UNIVERSIDAD DE GRANADA

Departamento de Ciencias de la Computación
e Inteligencia Artificial

Tesis doctoral

Bases de Datos Objeto-Relacionales Difusas: Modelo, Arquitectura y Aplicaciones

Autor

Carlos D. Barranco González

Director

Juan Miguel Medina Rodríguez

Septiembre, 2009

Editor: Editorial de la Universidad de Granada
Autor: Carlos Barranco González
D.L.: Gr. 3483-2009
ISBN: 978-84-692-6383-9

La memoria titulada "Bases de Datos Objeto-Relacionales Difusas: Modelo, Arquitectura y Aplicaciones", que presenta D. Carlos David Barranco González para optar al grado de Doctor, ha sido realizada en el Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada bajo la dirección del Doctor Juan Miguel Medina Rodríguez.

Granada, septiembre de 2009

El doctorando | El director

Carlos D. Barranco González | Juan Miguel Medina Rodríguez

Índice general

Agradecimientos

Aunque no serán pocas las ocasiones en que el desarrollo de un trabajo de investigación de este tipo se habrá comparado con la ascensión en montaña, permitidme la licencia de emplear, una vez más, esta analogía dada su adecuación y el amor que tengo por el tema. Desde mi punto de vista, una comparación más apropiada sería la ascensión para el paso de un collado, por cuanto no supone la cota más elevada, sino un hito en el camino que dará paso a nuevos horizontes y nuevos retos.

Para comenzar una gran ruta suele ser necesaria asistencia para aproximarnos al punto de partida, ya que, de otra forma, nuestras fuerzas se habrían agotado antes incluso de comenzar el camino. Afortunadamente, he tenido la inmensa suerte de contar con los mejores apoyos que uno pueda desear para que mi punto de partida fuese lo más ventajoso posible. Tengo que agradecer a mis padres su admirable esfuerzo, ímprobo y continuado, su cariño, educación, ejemplo, paciencia y confianza. Gracias a vosotros, aquellas lejanas tardes tomando un colacao delante de aquellas "lavadoras gigantes" que eran los ordenadores de principios de los 80 (durante las que, tengo entendido, causé hasta un incendio por error), los memorables días que pasé delante del *arrastrable* (posiblemente uno de los primeros modelos de ordenador "portátil"), mis primeros programas en el MSX y los millones de horas que habré pasado entre cacharros, son reflejo, hoy, del presente trabajo. Vaya, también, este agradecimiento para toda mi familia.

Una ruta no puede hacerse si uno no está bien preparado, guiado y pertrechado. Agradezco a Juan Miguel todo su esfuerzo, amabilidad y paciencia durante el desarrollo de este trabajo. Agradezco, también, su confianza, que me dio la oportunidad de comenzar el camino. Más allá de la dimensión académica, he de destacar su dimensión humana. Me siento enormemente afortunado por contar con su apoyo. Gracias, también al estado español, por pertrecharme con lo que necesité para emprender la ruta mediante diversas becas.

Durante todo el camino he ido encontrando excelentes personas que han sido fundamentales para llevar a buen término la ruta. Gracias a los que me dieron cariño, apoyo, compañía y amistad durante el camino. Gracias a los que me ayudaron cuando la carga de la mochila era muy pesada. Gracias a los que compartieron conmigo parte del camino. Gracias a los que me animaron a caminar cuando me sentía cansado. Gracias a los que, haciendo sus propios caminos, me acompañaron. Gracias a los que me crucé durante el camino y me ofrecieron su simpatía. Siento mucho no poder nombraros a cada uno y no quisiera caer en el error de olvidar a alguien, pero no puedo dejar de nombrar a Eva, Jesús, Elena, Raúl, Fede, Domingo, Alberto, Alejandro, Norberto, Paco, Dani, Merlos, Morón, Juanjo, Rocío, Nacho y Rafa. A todos, mi agradecimiento.

Capítulo 1

Introducción

El presente trabajo de investigación se marca como objetivo el establecimiento de unas bases para el desarrollo de Sistemas de Gestión de Bases de Datos (SGBDs) que permitan el manejo de información imperfecta. Usaremos los mecanismos de la lógica difusa para incorporar dicha capacidad para la representación y tratamiento de información imperfecta en aquellos SGBD actuales que implementan los estándares más recientes de SQL. Estos sistemas se denominan genéricamente SGBD objeto-relacionales, por lo que llamaremos *SGBD objeto-relacional difuso* (SGBDORD) a la extensión con capacidad para el manejo de información imperfecta que proponemos en esta memoria.

Para ello, este trabajo de investigación propondrá un modelo de bases de datos en el que se formalizarán las características anteriormente mencionadas, así como unas herramientas, concretamente un álgebra, que permita al acceso y manipulación de la información representada según dicho modelo.

Propuesto un modelo de bases de datos adecuado a las pretensiones del presente trabajo de investigación, se analizará la viabilidad del desarrollo de un sistema de estas características. Este análisis permitirá elegir, de la forma más adecuada a las posibilidades de los sistemas actuales, la aproximación empleada para abordar la propuesta. Para ello, en la presente memoria serán analizados los precedentes de este tipo de sistemas, realizando un análisis de los mismos con el objetivo de identificar las debilidades de éstos. De esta forma, dichas debilidades podrán ser tenidas en cuenta durante el desarrollo de esta investigación, con el fin de obtener soluciones para éstas. Posteriormente, se analizará la evolución y tendencias actuales de la industria de desarrollo de SGBDs. Tenidos todos estos aspectos en cuenta, como se analizará en detalle más delante, se propondrá el desarrollo de un SGBDORD en forma de una extensión de un SGBD objeto-relacional clásico. Esta extensión se desarrollará empleando los mecanismos que ofrecen este tipo de sistemas, como son los tipos de usuario y los operadores definidos por el usuario. Finalmente, el anterior estudio derivará en la implementación de un prototipo de la propuesta.

Generalmente, la adición a un SGBD de capacidades para la representación y manipulación de información imperfecta conlleva una disminución del rendimiento del mismo. Esta disminución es debida a la mayor complejidad en las operaciones de manipulación de este tipo de datos y a la imposibilidad de aplicar los mecanismos de acceso clásicos para este tipo de datos. Con el objetivo de incrementar el rendimiento de tales sistemas para facilitar su incorporación en

entornos de producción, el presente trabajo de investigación propondrá nuevos mecanismos de acceso diseñados específicamente para el tratamiento de datos imprecisos. Además, la concepción de éstos buscará el empleo de estructuras de indexado ya existentes en los SGBD clásicos para facilitar su incorporación en SGBD objeto-relaciones difusos desarrollados como la extensión de los primeros.

Finalmente, el presente trabajo de investigación mostrará varias aplicaciones prácticas de los SGBD objeto-relaciones difusos en diversas áreas. Entre éstas, se incluyen: las áreas de la gestión inmobiliaria, la búsqueda imágenes y la medicina.

La propuesta que describe esta memoria se desglosa mediante los siguientes capítulos:

- Capítulo 2: Este capítulo introducirá los conceptos previos que se utilizarán en el resto de capítulos de la memoria.
- Capítulo 3: El capítulo realiza un análisis de los precedentes en el área de la representación y tratamiento de información imprecisa en bases de datos.
- Capítulo 4: Este capítulo analiza los motivos que fundamentan la presentación de la propuesta objeto de este trabajo.
- Capítulo 5: El capítulo contiene la propuesta de modelo de bases de datos que combina las características de los SGBD objeto-relacionales junto con las capacidades de representación de datos imperfectos.
- Capítulo 6: Este capítulo propone un álgebra para manipular la información representada según el modelo definido en el Capítulo 5.
- Capítulo 7: En este capítulo se detalla la propuesta para el desarrollo de un SGBD basado en el modelo anterior.
- Capítulo 8: Este capítulo describe los mecanismos de acceso propuestos para incrementar el rendimiento de los SGBDs difusos.
- Capítulo 9: El capítulo describe un conjunto de aplicaciones prácticas que demuestran la utilidad del prototipo de SGBDORD propuesto en el Capítulo 7.
- Capítulo 10: Este capítulo presenta la conclusiones extraídas del presente trabajo, así como las líneas de trabajos futuras que quedan abiertas a partir del mismo.

Capítulo 2

Conceptos Previos

2.1. Introducción

El presente capítulo está dedicado a introducir al lector los conceptos que se utilizarán como base a lo largo de esta memoria.

Para ello, dividiremos el mismo en tres bloques. El primero de los bloques está compuesto por una introducción general a las bases de datos, ya que la presente memoria precisa de ciertos conocimientos básicos de este área por parte del lector. El segundo bloque está compuesto por una introducción a la teoría de conjuntos difusos y su derivado, la lógica difusa. Finalmente, el tercer bloque introducirá el concepto de multiconjunto difuso. A lo largo de la presente memoria, se hará un uso intenso de los conceptos básicos incluidos en este último par de bloques.

2.2. Conceptos generales sobre bases de datos

Esta sección presenta brevemente las nociones básicas sobre bases de datos que se presuponen en la presente memoria.

Para ello, se introducirá el concepto de *base de datos* y, posteriormente, los diferentes modelos de bases de datos existentes, así como la definición de *sistema de bases de datos* que emplearemos a lo largo del presente trabajo.

Tras la citada introducción, se examinarán en mayor detalle los modelos de bases de datos más significativos en la actualidad: el Modelo Relacional, el Modelo Orientado a Objetos y el Modelo Objeto-Relacional.

2.2.1. Concepto de base de datos

En [60] se define una *base de datos* como se indica seguidamente.

Definición 2.1 (Base de datos)**.** *Una* base de datos *es un conjunto de datos persistentes que es utilizado por los sistemas de aplicación de alguna empresa dada.*

Hay dos conceptos clave en la definición anterior:

- Datos: En la definición, el concepto de datos se usa de manera amplia, no haciendo distinción entre datos e información, como hacen otros autores.

Por tanto entenderemos como datos o información como cualquier hecho de importancia para un determinado individuo.

- Persistentes: Se dice que los datos en la base de datos son persistentes ya que éstos quedarán almacenados en la misma hasta que, de forma explícita, se indique que éstos han de ser eliminados.

En la anterior definición, se ha de entender el término empresa como individuo u organización con un determinado fin, según la propia interpretación de su autor.

2.2.2. Modelos de bases de datos

Toda base de datos, como colección de datos que es, debe seguir cierta organización para los datos que contiene y ciertos mecanismos para poder acceder de forma ordenada a los datos que ésta almacena. A dicha organización de los datos en una base de datos, y a los mecanismos de acceso asociados a ésta, se la denomina *modelo de datos*.

Este concepto se define en la bibliografía [60] como se indica a continuación.

Definición 2.2 (Modelo de datos)**.** *Un* modelo de datos *es una definición lógica, independiente y abstracta de los objetos, operadores y demás que en conjunto constituyen la máquina abstracta con la que interactúan los usuarios. Los objetos permiten modelar la estructura de los datos. Los operadores permiten modelar su comportamiento.*

Durante la historia de las bases de datos han ido surgiendo distintos modelos de datos, con propuestas diferentes para la organización y acceso a los datos por parte de los usuarios. Un listado de estos modelos, disponible con mayor detalle en [59], es el siguiente:

- Modelo jerárquico: Este modelo propone la organización de los datos en forma de jerarquía, de manera tal que, partiendo de un nodo raíz, los datos queden organizados en forma de árbol a través de relaciones padre-hijo. Debido a esta organización jerárquica, el acceso a los datos determinado por la estructura en la que éstos quedan organizados, suponiendo este hecho una limitación en muchos casos. Además de lo anterior, se ha de destacar la incapacidad del modelo para la representación de ciertos tipos relaciones múltiples entre los datos, como las relaciones *muchos a uno* y *muchos a muchos*.

- Modelo en red: Modelo similar al anterior, en el que se modifica la interrelación entre nodos, haciendo posible que un nodo tenga varios padres y, por tanto, haciendo posible la representación de relaciones muchos a muchos. Esta modificación permite la superación de la principal limitación del modelo anterior, por lo que hace posible la representación cualquier relación múltiple entre los datos. A pesar de la mejora que supuso, este modelo no gozó de popularidad durante mucho tiempo, a causa de la dificultad que suponía la localización y extracción de información de las bases de datos en red. Debido a la estructura, el proceso de localización y extracción debe hacerse navegando por las relaciones entre los distintos nodos de la red, lo que hacía necesario la implementación de programas de cierta complejidad incluso para las consultas más sencillas.

- Modelo relacional: Este modelo está basado en el concepto matemático de relación y, en la actualidad, es el más popular y utilizado. Se ha de destacar del mismo que ofrece un marco para el acceso sencillo a los datos. Postergaremos el análisis del mismo hasta la siguiente sección.

- Modelo orientado a objetos: El modelo orientado a objetos fue introducido en el ámbito de las bases de datos con el objetivo de ofrecer capacidades de representación y manejo de información de manera más sencilla e intuitiva para el usuario, resolviendo así ciertos problemas que el modelo relacional no era capaz de resolver de forma sencilla. En secciones posteriores profundizaremos en dicho modelo, donde se analizará éste en más detalle.

- Modelo objeto-relacional: El modelo objeto-relacional pretende unir las ventajas del modelo relacional (acceso sencillo a la información), junto con las ventajas de modelo orientado a objetos (potentes capacidades de modelado). De esta forma, se pretende contrarrestar las desventajas y sumar las ventajas de ambos modelos. Más adelante, profundizaremos en este modelo.

Finalmente, a partir del concepto de modelo de datos, podremos definir un *Sistema de Gestión de Bases de Datos* (SGBD) como una aplicación informática que implementa en un sistema informático la máquina abstracta definida por el modelo de datos determinado.

2.2.3. Bases de datos relacionales

Las bases de datos relacionales son bases de datos que emplean el modelo relacional como modelo de datos.

Se podría decir que, lo que las bases de datos pueden ser vistas como una colección de hechos ciertos, a partir de los cuales es posible inferir otros hechos adicionales mediante la ejecución de consultas sobre dicha base de datos. Un hecho cierto es lo que se denomina en lógica clásica *proposición verdadera*.

Las bases de datos relacionales, manejan de forma muy directa e intuitiva la representación y recuperación de estas proposiciones verdaderas y, de hecho, el modelo de datos relacional es toda una teoría formal derivada de la teoría de conjuntos y la lógica de predicados de primer orden.

Esta conjunción de simplicidad en la representación y acceso a los datos, así como el respaldo matemático del modelo, han llevado a las bases de datos relacionales a ser uno de los modelos más populares, implementados y estudiados tanto en el mundo académico como empresarial.

2.2.3.1. Modelo relacional

El Modelo Relacional fue propuesto en 1970 por el doctor E.F. Codd, investigador de los laboratorios de IBM en San José (California), siendo publicada su propuesta inicialmente en [48]. Dicho modelo surgió como un intento de simplificar la estructura de las bases de datos, que llegaba a ser demasiado compleja en otros modelos, y se popularizó rápidamente consolidándose como el modelo general de la inmensa mayoría de las bases de datos.

En el modelo relacional se establecen tres aspectos fundamentales para las bases de datos basadas en él: la estructura de los datos, la integridad de los

datos y la manipulación de los datos. A continuación, detallaremos cada uno de los citados aspectos.

Estructura de los datos.

El modelo relacional propone agrupar todos los datos o hechos ciertos, referentes a un tipo de entidad, o de interrelación entre entidades, en lo que se conoce como *relación*. Una relación no es más que un conjunto tuplas. Dichas relaciones suelen ser representadas de forma común e intuitiva como tablas.

Cada tupla de la relación se referirá a una entidad concreta del tipo que representa dicha relación. Las tuplas son representadas comúnmente como filas de las tablas que representan las relaciones.

Cada tupla contendrá una serie de valores correspondientes a los atributos de la relación. Cada uno de estos atributos es la información concreta que deseamos almacenar de cada una de las entidades representadas por las tuplas.

Cada valor correspondiente a un atributo deberá pertenecer al dominio asociado a dicho atributo. Los atributos de una relación suelen ser representados como las columnas de la tablas.

El siguiente ejemplo ilustra los anteriores conceptos.

Ejemplo 2.1. *Supongamos una base de datos que, entre otros, almacene datos sobre coches.*

Los datos relativos a los coches estarán agrupados en la tabla "coches". Dicha tabla tendrá asociada una fila que representará cada coche sobre el que tenemos datos, y poseerá tantas columnas como características de los coches queramos representar. Por ejemplo, "matricula" (en el dominio de las cadenas de caracteres), "puertas" (en el dominio de los números naturales), etc.

Dde manera más formal, se definen los elementos que conforma la estructura de una base de datos relación como se indica a continuación.

Definición 2.3 (Atributo)**.** *Llamaremos* Atributo *a cada una de las características importantes y propiedades que tiene cualquier tipo de entidad, o interrelación entre entidades, y que son almacenadas en la base de datos.*

Definición 2.4 (Dominio)**.** *Llamaremos* Dominio *al conjunto de todos los valores posibles que puede tomar un atributo. Por tanto, todos los atributos tendrán un dominio asociado.*

Definición 2.5 (Tupla)**.** *Llamaremos* Tupla *al conjunto de todos los valores concretos de todos los atributos de una determinada entidad o interrelación entre entidades.*

Definición 2.6 (Valor del dominio)**.** *Llamaremos* Valor del Dominio *a un valor concreto de un atributo concreto de una tupla.*

Definición 2.7 (Relación)**.** *Llamaremos* Relación *al conjunto constituido por dos partes llamadas* cabecera *y* cuerpo.

- *La* cabecera *es un conjunto fijo de pares atributo-dominio definido como se indica en la Ecuación 2.1, donde cada atributo A_j se corresponde exactamente con el dominio subyacente D_j con $j = 1, 2, \ldots, n$, siendo estos dominios no necesariamente distintos.*

$$\{(A_1 : D_1), (A_2 : D_2), \ldots, (A_n : D_n)\} \tag{2.1}$$

- *El* cuerpo *consta de un conjunto de tuplas, donde cada tupla consiste en un conjunto de n pares atributo-valor de la forma que se indica en la Ecuación 2.2, donde* $i = 1, 2, \ldots, m$*, siendo* m *el número de tuplas que contiene la relación.*

$$\{(A_1 : v_{i1}), (A_2 : v_{i2}), \ldots, (A_n : v_{in})\} \tag{2.2}$$

  Obsérvese que, para cada atributo A_j*, existe un par atributo-valor* $(A_j : v_{ij})$ *donde* v_{ij} *es el Valor de Dominio del atributo* A_j *perteneciente al dominio* D_j *de la tupla i-ésima.*

Integridad de los datos.

Con respecto a la integridad de los datos, el modelo relacional introduce un par conceptos y establece dos reglas para asegurar un acceso unívoco y consistente. Éstos son los siguientes:

- Clave primaria: Es un atributo o conjunto de atributos cuya combinación de valores identifican de forma unívoca una tupla en una relación.

- Clave externa: Es un atributo, o conjunto de atributos, que contiene un valor presente en la clave primaria de otra relación. Las claves externas permiten que una tupla de una relación pueda contener referencias a tuplas en otras relaciones.

Adicionalmente, el modelo relacional establece un importante concepto, el *valor nulo*. Todo atributo podrá contener el valor nulo, independientemente del dominio asociado a éste. Dicho valor nulo se interpreta de dos formas posibles:

- Desconocimiento: No se conoce el valor de un atributo para una determinada tupla. Por ejemplo, la edad de una persona puede ser desconocida.

- Inaplicabilidad: El atributo no es aplicable para la entidad representada. Por ejemplo, el nombre de la esposa para un soltero.

Estos conceptos previos nos permiten formular las reglas de integridad, a las que nos referíamos en el inicio de este apartado, de la forma siguiente:

- Integridad de entidad: Toda entidad, representada como una tupla en una relación, debe de ser distinguible de forma unívoca. Para ello toda relación deberá tener definida una clave primaria válida, quedando prohibido que algún atributo que forme parte de dicha clave primaria contenga el valor nulo.

- Integridad referencial: Si una relación contiene una clave externa, los valores de dicha clave deben coincidir con los valores presentes en la clave primaria de la relación que referencia o ser todos nulos.

Las anteriores reglas garantizan un estado consistente de la base de datos. La primera de ellas asegura que toda entidad almacenada en la base de datos puede ser accedida de forma unívoca conociendo el valor de su clave primaria y la relación a la que pertenece. En lo que respecta a la segunda, ésta garantiza que sea imposible la existencia de una referencia a una entidad que no sea parte de la base de datos.

Además de las anteriores reglas, el modelo relacional permite la imposición de reglas adicionales por parte de los usuarios o diseñadores de bases de datos. Estas reglas, llamadas reglas de negocio, permiten importer ciertas restricciones a los datos que no son posibles de expresar empleando solamente el modelo lógico, y que son esenciales para el mantenimiento de la integridad y coherencia de la base de datos. Por ejemplo, en el contexto empleado en el Ejemplo 2.1, se puede definir la regla que haga obligatorio que el atributo "número de puertas" de la relación "coches" sólo pueda contener los valores 3 o 5.

Manipulación de los datos.

Como mecanismos de manipulación del modelo relacional, Codd [49] propuso el *álgebra relacional* y el *cálculo relacional*.

El *álgebra relacional* es un lenguaje procedural para la manipulación de relaciones, mediante el cual se indican qué operaciones se han de aplicar sobre las relaciones de la base de datos para obtener el resultado que buscamos, es decir, cómo construir el resultado que buscamos. Éste es un lenguaje formal que posee una serie de operadores que se aplican sobre una o varias relaciones para ofrecer como resultado otra relación. El hecho de que tanto los operandos como los resultados sean relaciones permite el anidamiento de dichos operadores con el objeto de obtener como resultado final la relación que busca el usuario. En [60, 144], donde se pueden encontrar más detalles sobre éste lenguaje, se indican cinco operadores fundamentales (selección, proyección, producto cartesiano, unión y diferencia) y tres operadores no fundamentales (reunión, intersección y división), pudiendo estos tres últimos ser expresados a partir de los operadores fundamentales.

El *cálculo relacional*, en contraste con el álgebra relacional, es un lenguaje de consulta no procedural basado en el cálculo de predicados de primer orden, mediante el cual el usuario expresa la relación que desea obtener como resultado de la consulta. Por tanto, el usuario define la información que desea obtener de la base de datos, sin necesidad de especificar qué operaciones han de hacerse sobre las relaciones existentes en dicha base de datos para obtener la citada información.

Existen dos aproximaciones al cálculo relacional, llamadas *cálculo relacional de tuplas*, propuesto inicialmente por Codd [49], y *cálculo relacional de dominios*, propuesto por Lacroix y Pirotte en 1977 [89], en el que se sustituyen las variables de tupla por variables de dominio.

Ambos métodos para la manipulación de relaciones, el álgebra y el cálculo relacional, son equivalentes. Esto significa que cualquier relación obtenida como resultado del empleo de los operadores del álgebra relacional se podrá obtener haciendo uso de los operadores del cálculo relacional, tanto de tuplas como de dominios, y viceversa. Ullman [144] propuso el procedimiento para la transformación de una expresión en álgebra relacional a cálculo relacional, ya sea de tuplas o de dominios, y viceversa.

Cualquier lenguaje de consulta sobre bases de datos relacionales alternativo a los anteriores se califica como *relacionalmente completo* si ofrece, como mínimo, la potencia que del álgebra o el cálculo relacional. Nótese que la condición anterior es de mínimos, ya que es común que los lenguajes de consulta alternativos relacionalmente completos ofrezcan mayor potencia, de tal forma que se permita al usuario una mayor facilidad de uso y capacidad de obtención de resultados complejos.

2.2.3.2. Lenguaje SQL

Hoy en día, SQL (Structured Query Language) es el lenguaje estándar para trabajar con bases de datos relacionales y está soportado, en general en forma de dialectos, prácticamente por todos los productos del mercado.

En lenguaje SQL está dividido, según la funcionalidad de sus sentencias, en los dos sublenguajes siguientes:

- DDL (Data Definition Language): Es el conjunto de sentencias que nos permiten la definición, alteración y eliminación de las estructuras que contendrán los datos, es decir, las tablas y vistas.
- DML (Data Manipulation Language): Este sublenguaje contiene el subconjunto de las sentencias que nos permiten manipular los datos en la base de datos. Estas sentencias permitirán la consulta, inserción y borrado de dichos datos.
- DCL (Data Control Language): Este último subconjunto del SQL contiene la sentencias que permiten al administrar los derechos de acceso a los datos.

La historia de SQL es extensa. Después de que E.F. Codd publicara el artículo que supuso el nacimiento del modelo relacional, comenzó la investigación en los laboratorios de IBM en San José (California), sobre un lenguaje que implementase dicho modelo. En 1974, Donald Chamberlin y otros investigadores de dicho centro definieron el lenguaje SEQUEL (Structured English Query Language) [42], basado un lenguaje anterior llamado SQUARE [125], con una sintaxis de estilo marcadamente matemático. Dicho lenguaje fue implementado en el prototipo SEQUEL-XRM entre 1974 y 1975.

Los experimentos con dicho prototipo condujeron, entre 1976 y 1977, a una revisión del lenguaje llamada SEQUEL/2 [43] que, por motivos legales, cambió de nombre posteriormente convirtiéndose en SQL.

Un nuevo prototipo desarrollado en 1977, System R, fue el primero en implementar el lenguaje ya conocido como SQL. Dicho prototipo, que aun no se comercializaba, fue empleado internamente en IBM durante algún tiempo, así como por parte de algunos clientes seleccionados, antes de la incorporación de dicha tecnología en productos comerciales. El éxito cosechado por el prototipo animó a otras compañías a desarrollar sus propios productos relacionales basados en SQL, como Relational Software (actualmente conocida como Oracle) y Sybase. El comienzo de la venta de los primeros productos relacionales por parte de IBM fue en 1981, culminando con la aparición de DB/2 en 1983.

El éxito de los productos basados en SQL, convirtió a éste en un estándar industrial *de facto* que pronto se convertiría en un estándar *de jure*. En 1986, ANSI (American National Standards Institute) publicó el estándar SQL-86, que

era básicamente el dialecto IBM de SQL. Un año más tarde, en 1987, dicho estándar fue aceptado por ISO (International Standards Organization). Durante los siguientes años fueron apareciendo propuestas para la mejora y modificación de SQL-86 resultando en la publicación de un nuevo estándar denominado SQL-89 y, más tarde, en una revisión completa denominada SQL-92.

Posteriormente a la publicación de SQL-92, fueron surgiendo de forma paulatina nuevas adiciones parciales. En 1995 fue publicada la propuesta SQL/CLI (Call-Level Interface) conocida como CLI-95, que define una interfaz de programación de aplicaciones (API, del inglés *Application Programming Interface*), para el acceso a bases de datos relacionales. En 1996 se publicó SQL/PMS (Persistent Stored Module) conocida como PMS-96, que definía la sintaxis de la extensión procedural de SQL para especificar la lógica de módulos almacenados en el servidor de bases de datos (en otras palabras, procedimientos y funciones ejecutables en el servidor). La última adición, publicada en 1998, conocida como SQL/OLB o OLB-98, especificó los mecanismos de incrustación de sentencias SQL en programas escritos en lenguaje Java.

Finalmente, en 1999, se agruparon las adiciones mencionadas anteriormente en la versión del estándar conocida informalmente como SQL3. Dicha versión fue publicada bajo la denominación más formal de SQL:1999. Además de agrupar y consolidar las adiciones hechas al estándar SQL tras SQL-92, SQL:1999 incorpora nuevas características, denominadas objeto-relacionales, que serán comentadas en mayor detalle más adelante en el presente capítulo.

Tras la versión SQL:1999, se han publicado versiones adicionales que añaden diversa funcionalidad a este. La versión denominada SQL:2003 modifica el estándar SQL:1999 para añadir, como incorporación más relevante, la posibilidad de manejo de datos XML (Extensible Markup Language). Seguidamente, la versión SQL:2006 modifica a la anterior para profundizar en el manejo de datos XML. Finalmente, la versión SQL:2008 supone una adición a su antecesora para incorporar al estándar algunas características funcionales (como la posibilidad de emplear disparadores para gestionar las modificaciones sobre vistas) muy populares en distintos SGBD.

2.2.3.3. Evoluciones del modelo relacional

Desde la proposición del modelo relacional hasta la introducción del nuevo paradigma de los modelos orientados a objetos (que serán estudiados a continuación), se realizaron propuestas para la mejora del mismo a fin de reducir las desventajas que se evidenciaron tras su adopción en innumerables aplicaciones. De entre las diferentes propuestas, podemos destacar las *relaciones anidadas* y las *multirrelaciones*.

Relaciones anidadas.

Uno de los factores más relevantes que han contribuido al éxito del modelo relacional es que éste proporciona un soporte satisfactorio para las necesidades típicas en aplicaciones de gestión empresarial. Este tipo de aplicaciones manejan grandes cantidades de datos con una estructura relativamente simple, para los cuales, el modelo relacional resulta efectivo y eficiente, así como muy potente en lo que refiere a sus capacidades de consulta.

El éxito del modelo relacional en el campo empresarial estimuló la adopción de la tecnología de bases de datos relacionales en campos distintos como,

por ejemplo, CAD (Computer Aided Design), CAM (Computer Aided Manufacturing), CASE (Computer Aided Software Engineering), GIS (Geographic Information Systems), aplicaciones médicas y científicas, almacenamiento y recuperación de multimedia, etcétera. No obstante, pronto se hizo patente que el modelo relacional no es adecuado para el manejo de los datos y relaciones complejos que son necesarias en este tipo de aplicaciones.

Una de las primeras propuestas para solucionar la anterior situación es el *modelo de relaciones anidadas* [136]. Este modelo parte de la base del modelo relacional, extendiendo el mismo para permitir que los valores de los atributos sean, a su vez, relaciones. De esta forma, se permite el anidamiento de unas relaciones en otras, dando un soporte más adecuado para datos y relaciones complejos. La extensión que el modelo de relaciones anidadas hace sobre el modelo relacional implica la rotura que impone la Primera Forma Normal [48] de este último, por lo que también se conoce al modelo relacional anidado como modelo NF^2 (Non First Normal Form) y a las relaciones anidadas como *relaciones NF^2*.

Las relaciones NF^2 se manipulan con el mismo conjunto de operadores del álgebra relacional, añadiendo a éste los operadores de *anidación* y *desanidación*. Este par de nuevos operadores permiten que el juego de operadores relacionales puedan ser aplicados sobre las relaciones anidadas. Su funcionamiento es simple, el operador de desanidación reduce en un nivel las capas de anidamiento de una relación anidada, mientras que el operador de anidación realiza la operación contraria. Dado un atributo a de una relación r cuyos valores son, a su vez, relaciones, al aplicar el operador de desanidación sobre el mismo se obtiene una nueva relación, notada como $\mu_a(r)$, en que las tuplas de cada relación anidada se concatenan con la tupla que las contiene (operación equivalente a una suerte de producto relacional entre cada tupla y las tuplas de la relación anidada que ésta contiene). Por su parte, el operador de anidación funciona como una especie de operador de división. Dado un conjuntos de atributos $a_1, a_2, \ldots, a_n$ de una relación r y un nuevo nombre de atributo a, el anidamiento de los atributos en $a_1, a_2, \ldots, a_n$ de r en a resulta en una relación, notada como $\nu_{a=(a_1,a_2,\ldots,a_n)}(r)$, en la que todas las tuplas con valores similares para los atributos de r excluyendo los atributo $a_1, a_2, \ldots, a_n$ son agrupadas en una única tupla. Dicha tupla tendrá un nuevo atributo denominado a cuyo valor será una relación en la que estarán contenidos los valores de los atributos $a_1, a_2, \ldots, a_n$ para las tuplas de r que ésta agrupa.

El siguiente ejemplo ilustra el concepto de relación anidada y el funcionamiento de los operadores de anidación y desanidación.

Ejemplo 2.2. *Supongamos que disponemos de una relación anidada similar a la mostrada en la Tabla 2.1. Ésta relación, que denominaremos c, contiene los datos relativos a las colecciones de libros de una editorial. En ella, se puede apreciar que los valores del atributo* Libro *son relaciones anidadas. A su vez, éstas incluyen relaciones anidadas como valores del atributo* Capítulos.

El resultado de aplicar sobre c el operador de desanidación para el atributo Libros*, es una relación, notada como $\mu_{Libros}(c)$, de la forma que se indica en la Tabla 2.2. Si, sobre esta nueva relación que denominaremos d, desanidamos el atributo* Capítulos*, el resultado es la relación que se muestra en la Tabla 2.3. Por simplicidad, denominaremos a esta última relación como e.*

Hechas estas transformaciones, podemos recuperar la relación c original aplicando el operador de anidación. Concretamente, obtendremos que se cumple

Colección	Libro			
	Título	Precio	Capítulos	
			Número	Título del Capítulo
Col1	Lib1	18	1	Cap1
			2	Cap2
			3	Cap3
			4	Cap4
	Lib2	25	1	Cap5
			2	Cap6
			3	Cap7
Col2	Lib3	18	1	Cap8
			2	Cap9

Tabla 2.1: Cabecera y cuerpo de c

Colección	Título	Precio	Capítulos	
			Número	Título del Capítulo
Col1	Lib1	18	1	Cap1
			2	Cap2
			3	Cap3
			4	Cap4
Col1	Lib2	25	1	Cap5
			2	Cap6
			3	Cap7
Col2	Lib3	18	1	Cap8
			2	Cap9

Tabla 2.2: Cabecera y cuerpo de $\mu_{\text{Libros}}(c)$

Colección	Título	Precio	Número	Título del Capítulo
Col1	Lib1	18	1	Cap1
Col1	Lib1	18	2	Cap2
Col1	Lib1	18	3	Cap3
Col1	Lib1	18	4	Cap4
Col1	Lib2	25	1	Cap5
Col1	Lib2	25	2	Cap6
Col1	Lib2	25	3	Cap7
Col2	Lib3	18	1	Cap8
Col2	Lib3	18	2	Cap9

Tabla 2.3: Cabecera y cuerpo de $\mu_{\text{Capítulos}}(d)$

Precio	Capítulos	
	Número	Título del Capítulo
18	1	Cap1
	2	Cap2
	3	Cap3
	4	Cap4
25	1	Cap5
	2	Cap6
	3	Cap7
18	1	Cap8
	2	Cap9

Tabla 2.4: Cabecera y cuerpo de f

Precio	Número	Título del Capítulo
18	1	Cap1
18	2	Cap2
18	3	Cap3
18	4	Cap4
25	1	Cap5
25	2	Cap6
25	3	Cap7
18	1	Cap8
18	2	Cap9

Tabla 2.5: Cabecera y cuerpo de $\mu_{\text{Capítulos}}(f)$

la igualdad $\nu_{Capítulos=(Número,\,Título\ del\ Capítulo)}(e) = d$ *y, de la misma forma,* $\nu_{Libro=(Título,Precio,Capítulos)}(d) = c$.

El uso de los operadores de anidación y desanidación implica la necesidad del tratamiento de un problema que ha sido estudiado ampliamente en la bibliografía: la complementación de los operadores de anidación y desanidación.

En primer lugar, se ha de notar que los operadores de anidación y desanidación no son complementarios en cualquier caso. Emplearemos el siguiente ejemplo para ilustrar este hecho.

Ejemplo 2.3. *Supongamos que partimos de una relación, que denominaremos f, de la forma que se indica en la Tabla 2.4. Obsérvese que ésta es una proyección de la relación d, descrita en el Ejemplo 2.2.*

Si aplicamos sobre la relación f el operador de desanidación para el atributo Capítulos, *obtendremos la relación que se muestra en la Tabla 2.5. Denominaremos a esta última relación como g.*

Al aplicar el operador de anidación sobre g para obtener una relación equivalente a f, se obtiene la relación que se muestra en la Tabla 2.6. Obsérvese que, contrariamente a lo esperado, ésta relación, que denominaremos h, no es equivalente a la relación f. Este resultado es debido a que, una vez desanidada

Precio	Capítulos	
	Número	Título del Capítulo
18	1	Cap1
	2	Cap2
	3	Cap3
	4	Cap4
	1	Cap8
	2	Cap9
25	1	Cap5
	2	Cap6
	3	Cap7

Tabla 2.6: Cabecera y cuerpo de $\nu_{\text{Capítulos=(Título,Precio,Capítulos)}}(f)$

la relación f, no es posible distinguir aquellas tuplas en g que proceden de las tuplas primera y tercera (según el orden en que se muestran en la Tabla 2.4) de f. Como se indicó anteriormente, el operador de anidación agrupa, en una misma tupla, todas las tuplas de la relación sobre el que se aplica cuyo valor para los atributos no participantes en el proceso de anidación es igual.

Para asegurar que los operadores de anidación y desanidación son complementarios, propiedad deseable para manipular a cualquier nivel las relaciones anidadas con los operadores relacionales, se propone en la bibliografía la definición de diversas formas normales para la relaciones anidadas [112]. La más general y aceptada es la forma normal denominada *forma normal de particiones* (FNP) [129]. Ésta restringe las relaciones NF^2 a aquellas que poseen una clave primaria en la que no forma parte ningún atributo que toma como valores relaciones anidadas. Dadas las cualidades tan deseables del subconjunto de relaciones NF^2 en FNP, ha sido posible el desarrollo de un álgebra formal para las mismas [129], lo que permite la optimización de sus expresiones de forma equivalente al álgebra relacional clásica [96, 80].

Además del problema de la complementación de los operadores de anidación y desanidación en relaciones NF^2 genéricas, se ha estudiado ampliamente la complementación de éstos en presencia de valores ausentes cuya semántica asociada indica la inexistencia del valor para atributos cuyos valores son relaciones anidadas [92, 174].

Los valores inexistentes pueden modelarse de forma natural, en el ámbito de los modelos NF^2, como relaciones anidadas vacías. El problema llega cuando se aplica sobre los mismos el operador de desanidación. Al aplicar el comportamiento habitual del operador de desanidación, las tuplas que tengan como valor del atributo a desanidar una relación vacía no tendrán correspondencia en la relación resultante, ya que el producto de cada tupla con sus tuplas anidadas (ninguna en este caso) produce un resultado vacío. Este problema se torna aún más evidente cuando, al aplicar el operador de anidación para reconstruir la relación original, no es posible recuperar las citadas tuplas.

La solución más ampliamente aceptada para este problema es permitir la equivalencia de las relaciones vacías con una relación que contiene una única tupla cuyos atributos tienen asociado como valor un valor ausente de semántica

inexistente [130, 93, 131]. Esto permite la presencia, en la relación resultante de un proceso de desanidación, de las tuplas con valor inexistente para el atributo participante en la desanidación, así como la reconstrucción de la relación original mediante la aplicación de la operación de anidación complementaria. No obstante, se ha de tener en cuenta que ésta solución impide que se puedan distinguir los casos en que el valor de un atributo que contiene relaciones anidadas es inexistente o posee una relación con una única tupla en que los valores para todos sus atributos son valores inexistentes.

Multirrelaciones.

A pesar de que en el modelo relacional las relaciones son modeladas como conjuntos, la mayoría de los SGBD basados en éste y sus lenguajes de consulta (por ejemplo, SQL) admiten que en las relaciones puedan existir múltiples ocurrencias de una determinada tupla. Esta diferencia entre el modelo relacional y sus implementaciones se origina por motivos prácticos. Con objeto de incrementar el rendimiento en el procesamiento de consultas de los SGBDs, se permite la existencia de duplicados en las relaciones, ya que su eliminación puede resultar una tarea muy costosa en grandes bases de datos.

Lo que, en un principio, pudo ser una carencia en las implementación del modelo relacional, se ha convertido en una ventaja que incrementa la capacidad expresiva tanto de las implementaciones como de los lenguajes de consulta que la soportan, ofreciendo ventajas en campos de aplicación como sistemas de recuperación de información [111] o minería de datos [90].

Estas ventajas han incentivado la investigación para proveer unos fundamentos teóricos sólidos, similares a los del modelo relacional, para el caso de relaciones en que se admiten duplicados [76, 90]. Este nuevo tipo de relaciones, de nominadas *multirrelaciones* [83], se modelan empleando multiconjuntos (estructuras similares a los conjuntos que admiten duplicados, que serán introducidas con mayor detalle en la Sección 2.4 del presente capítulo) para permitir la mencionada multiplicidad de tuplas.

2.2.4. Bases de datos orientadas a objetos

Las bases de datos orientadas a objetos fueron propuestas como respuesta ante ciertos problemas, ante los cuales, las bases de datos relacionales no eran capaces de dar respuestas adecuadas. Dichos problemas fueron, principalmente, los siguientes:

- Dificultad de modelado: Los elementos presentes en los problemas reales, normalmente complejos e interrelacionados, no suelen ser modelados con facilidad empleando el modelo relacional. Como soluciones alternativas fueron apareciendo nuevos modelos de datos, llamados modelos semánticos, que permitían modelar la realidad de forma más directa cada vez. Uno de estos primeros modelos de datos post-relacional fue el modelo Entidad-Relación, propuesto por Chen en 1976. Dicho modelo ha sido empleado como herramienta para el diseño de bases de datos relacionales, ya que éste puede ser trasladado con facilidad al modelo relacional. Actualmente, el modelado orientado a objetos es uno de los más potentes y aceptados, ya que permite modelar la realidad de una forma próxima y menos rígida.

- Poca adecuación del modelo de datos relacional a las necesidades de las nuevas aplicaciones: Como se ha comentado anteriormente, con el paso de tiempo, han ido surgiendo nuevas aplicaciones apoyadas en sistemas de bases de datos, como CAD, CAM, CASE, GIS, etcétera. Con dichas aplicaciones, los sistemas de bases de datos relacionales tienden a incurrir en problemas, como la falta de eficiencia en la recuperación de datos y la dificultad de modelado de dichos datos empleando el modelo relacional.

- Ineficiencia en los servicios de persistencia ofrecidos a las aplicaciones programadas en lenguajes orientados a objetos: En la actualidad la mayor parte de las aplicaciones son desarrolladas empleando lenguajes orientados a objetos y, por tanto, las entidades que manipulan las aplicaciones son objetos. Cuando las aplicaciones necesitan hacer persistentes esos objetos para un uso futuro, deben acudir a los servicios de bases de datos relacionales, por lo que se hace necesario una conversión entre las estructuras de dichos objetos y las disponibles en la base de datos relacional (tuplas). Igualmente, cuando se desea recuperar un objeto desde la base de datos, es necesario realizar el proceso inverso. Este mapeo de objetos en relaciones provoca, en cierta medida, un descenso en el rendimiento de las aplicaciones, así como la inevitable necesidad de añadir una capa adicional al software. Dicho problema es conocido en la literatura como "Object-Relational Impedance Mismatch".

Los problemas señalados anteriormente motivaron la proposición de un nuevo tipo de bases de datos basadas en modelos orientados a objetos. Una de las primeras y más importantes propuestas que plantearon seriamente los fundamentos de las bases de datos orientada a objetos fue el documento conocido como "The Object Oriented Manifesto" [4] escrito por Atkinson. En dicho documento, se define qué características debe poseer una base de datos orientadas a objetos para ser considerada como tal y cuáles son las características opcionales que sería deseable que incluyesen.

Una base de datos orientada a objetos debe integrar las capacidades del paradigma orientado a objetos junto con las capacidades que proporciona una base de datos, obteniendo así una combinación de los beneficios de ambos elementos.

El paradigma de orientación a objetos ofrecerá las siguientes características:

- Tipos de datos abstractos.
- Herencia.
- Identidad de objetos.

Por su parte, como base de datos deberá de ofrecer las usuales características siguientes:

- Persistencia.
- Concurrencia.
- Transacciones.
- Recuperación ante fallos.

- Facilidades de consulta.
- Versionado.
- Integridad.
- Seguridad.
- Rendimiento.

Hasta la fecha, se han seguido diferentes aproximaciones para la implementación de bases de datos que incorporasen las capacidades del paradigma orientado a objetos. Dichas aproximaciones, recogidas en [82], son las siguientes:

1. La creación de un nuevo modelo/lenguaje de base de datos: Esta alternativa, la más agresiva, consiste en crear un nuevo lenguaje, derivado de un modelo de datos orientado a objetos, junto con el desarrollo de un sistema de base de datos orientada a objetos que implemente dicho lenguaje. La mayoría de los proyectos de investigación sobre bases de datos orientadas a objetos han optado por ésta alternativa.

2. Extensión de un lenguaje de bases de datos incorporando las capacidades del paradigma de orientación a objetos: La estrategia seguida consiste en incorporar a lenguajes clásicos de bases de datos, como puede ser SQL, los elementos necesarios que permitan disfrutar de las características ofrecidas por el paradigma orientado a objetos. Esta es la estrategia seguida por la industria de desarrollo de bases de datos relacionales comerciales, como son Oracle, IBM, Informix y Borland.

3. Extensión de un lenguaje de programación orientado a objetos incorporando capacidades de bases de datos: Esta aproximación pretende introducir capacidades de bases de datos dentro de lenguajes orientados a objetos existentes. El ejemplo más conocido es la extensión del lenguaje Smalltalk, llamada OPAL, que incluye la base de datos orientada a objetos GemStone.

4. Desarrollo de bibliotecas extensibles que proporcionen una gestión de datos orientada a objetos: Muchos fabricantes de bases de datos orientados a objetos, como Ontos y Versant, con sus librerías para C++, optaron por esta estrategia. Simplemente, se proporcionan ciertas librerías que suministran servicios de persistencia, así como métodos para la gestión de transacciones y demás características de las bases de datos.

5. Incrustar estructuras de lenguajes de bases de datos orientadas a objetos en lenguajes de programación convencionales: La propuesta de incrustar lenguajes de bases de datos en código de lenguajes de programación fue muy bien aceptada. En el caso de SQL ésta fue ampliamente empleada, existiendo sistemas que permiten incrustar sentencias SQL en código C, FORTRAN, Ada, etc. En el ámbito de las bases de datos orientadas a objetos, el ejemplo más significativo es O2 [6], que proporciona capacidades de incrustación de sentencias en código C (llamado CO2) y Basic.

6. Desarrollo de aplicaciones específicas con sistemas de gestión de bases de datos orientados a objetos ad-hoc: Como última aproximación, también bastante extendida, se puede considerar la creación de ciertos servicios de persistencia para una aplicación concreta. Estos servicios, a pesar de no constituir una base de datos orientada a objetos de propósito general, ofrecen las facilidades propias de las bases de datos orientadas a objetos que la aplicación requiere.

2.2.5. Comparativa de modelos

A continuación, se realiza una comparativa no exhaustiva entre bases de datos relacionales y bases de datos orientadas a objetos. En ésta, se enumerarán las ventajas y desventajas que presentan los citados modelos.

En primer lugar, enumeraremos las ventajas del modelo orientado a objetos con respecto al modelo relacional. Éstas son las siguientes:

1. Representación adecuada de objetos complejos: Las bases de datos orientas a objetos aceptan los objetos complejos de forma natural. Las bases de datos relacionales, en cambio, permiten la representación de dichos objetos empleando un número elevado de relaciones (al menos tantas como el número de componentes de la jerarquía de composición del objeto) las cuales deben ser reunidas por el sistema de gestión de bases de datos para permitir el acceso a los subcomponentes del objeto, lo que provoca un claro descenso del rendimiento.

2. Jerarquías de clases: La forma natural de presentar los datos del mundo real suele ser el empleo de jerarquías. Dichas jerarquías se asumen de forma directa en el modelo orientado a objetos. Los modelos relacionales, en cambio, no disponen de formas de representación directa para éstas y no aceptan el tratamiento semántico de las mismas.

3. Eliminación del problema de impedancia objeto-relacional: Como se indicó anteriormente, las bases de datos orientas a objetos pueden almacenar de forma directa los elementos con las que suelen trabajar las aplicaciones desarrolladas con lenguajes de tercera generación, los objetos. Esta capacidad evita la perdida de rendimiento que supone la conversión de dichos objetos en tuplas, operación necesaria en el caso que se empleen bases de datos relacionales.

4. Alta capacidad para definir restricciones sobre los datos y la manipulación de los mismos: Debido a que el modelo orientado a objetos incluye métodos para la manipulación del estado de los objetos, dicho proceso de manipulación puede ser controlado de la forma deseada por el usuario mediante la especificación, en forma procedural, de los procedimientos de modificación de los objetos.

Por otra parte, las desventajas de las bases de datos orientadas a objetos sobre las bases de datos relacionales son las siguientes:

1. Debilidad ante cambios en el esquema: En las bases de datos relacionales, añadir, modificar o eliminar una tabla no suele tener repercusión ninguna sobre las aplicaciones que trabajan sobre ésta. En una base de datos

orientada a objetos, añadir, modificar o eliminar clases suele requerir modificaciones en las clases de la aplicación que interactúan con las instancias de las clases modificadas. Estas modificaciones en la base de datos normalmente hacen necesario una recompilación integra de la aplicación. Adicionalmente, se añade la necesidad de la adaptación de las instancias ya existentes en la base de datos, pertenecientes a las clases modificadas, a las nuevas definiciones de dichas clases. Esto último puede suponer un largo tiempo de procesamiento en función del tamaño de la base de datos.

2. Dependencia del lenguaje: Los datos de las bases de datos relacionales son accesibles desde cualquier aplicación de forma prácticamente independiente al lenguaje de programación con el que haya sido desarrollada ésta. En cambio, las bases de datos orientadas a objetos normalmente están fuertemente ligadas a un lenguaje de programación orientado a objetos concreto por medio de la API de acceso a las mismas. Esto significa que la base de datos sólo puede ser accedida desde aplicaciones desarrolladas usando un determinado lenguaje para el que existe la API de acceso correspondiente.

3. Carencia de capacidad para procesamiento de consultas ad-hoc: En el modelo relacional se admite el procesamiento de consultas ad-hoc ya que, debido a la naturaleza de los datos, es posible crear nuevas tablas a partir de la unión de tablas ya existentes, relacionando los datos existentes entre sí para obtener resultados de cualquier consulta. Teniendo en cuenta que en el modelo orientado a objetos no es posible emular la semántica de la unión relacional mediante la unión de dos clases, se hace evidente que el modelo orientado a objetos posee una flexibilidad mucho menor a la hora de aceptar consultas por parte de los usuarios. De hecho, se puede decir que las consultas que pueden ser aplicadas sobre los datos en el modelo orientado a objetos están determinadas en gran manera por el diseño del sistema. Adicionalmente, se ha de tener en cuenta que la encapsulación de los objetos puede hacer imposible que ciertas consultas puedan emplear criterios en base a condiciones impuestas al estado interno de los objetos, cuando estos objetos no muestran dicho estado mediante una interfaz pública. Este hecho dificulta aun más la posibilidad de la ejecución de este tipo de consultas ad-hoc.

4. Carencia de un modelo teórico bien fundamentado: En contraste con las bases de datos relacionales, las bases de datos orientadas a objetos no disponen de un modelo ampliamente aceptado. Adicionalmente, se crítica que los modelos que se han propuesto no poseen una base matemática tan sólida como el modelo relacional.

2.2.6. Bases de datos objeto-relacionales

Las bases de datos objeto-relacionales, conocidas también como bases de datos relacionales extendidas, son la última tendencia en el mundo de las bases de datos. Dichas bases de datos pretender aunar las ventajas de las bases de datos relacionales con las ofrecidas por las bases de datos orientadas a objetos.

Las bases de datos objeto-relacionales constituyen una *evolución* de las bases de datos, en contraste con la *revolución* que constituyen las bases de datos orientas a objetos. Toda base de datos objeto-relacional debe ser compatible y

ofrecer todas las características de las bases de datos relacionales clásicas. De hecho, a los ojos del usuario si este no emplea las características extendidas, una base de datos objeto-relacional debe ser totalmente equivalente a una base de datos relacional.

Las bases de datos objeto-relacionales, toman de las bases de datos orientadas a objetos ciertas características que permiten abordar nuevos campos de aplicación y hacen mucho más cómoda y sencilla la interacción por parte de los usuarios y aplicaciones. Dichas características son las siguientes:

- Tipos complejos definidos por el usuario: Los sistemas de gestión de bases de datos relacionales ofrecen a los usuarios un número limitado de tipos de datos, normalmente muy simples. Las bases de datos objeto-relacionales permiten la definición, por parte del usuario, de tipos de datos estructurados en base a tipos básicos que se adecúen mejor a las necesidades de éste.

- Jerarquías de tipos: Las bases de datos objeto-relacionales incorporan la herencia de tipos como herramienta para modelar las jerarquías de tipos, que suelen ser muy usuales en la modelización de problemas del mundo real. Dicha herencia de tipos, incrementa las capacidades semánticas del modelo de datos, dotando a éste de capacidades adicionales para el modelado.

- Operaciones propias sobre los datos: La inclusión de tipos de datos definidos por el usuario proporciona al usuario la posibilidad de definir e implementar nuevos métodos y operadores ligados a estos, para tratar de forma correcta y eficiente los datos de estos nuevos tipos.

- Extensibilidad del SGBD: Se proporciona al usuario la capacidad de extensión de los mecanismos internos del SGBD con el objeto de permitir la incorporación de nuevas estructuras de indexado apropiadas para los tipos definidos por éste, así como la incorporación de lógica adicional para mejorar los mecanismos de optimización de consultas cuando se aplican sobre los citados tipos.

Algunos autores [60, 61], critican la denominación de las bases de datos objeto-relacionales, ya que indican que esta propuesta no es nueva. Según dichos autores, los sistemas de gestión de bases de datos objeto-relacionales no son más que SGBD relacionales en los que el modelo relacional ha sido implementado de forma correcta, dando soporte apropiado al concepto de *dominio* definido ya en el trabajo en que se propuso dicho modelo relacional a principios de los años 70 del siglo pasado [48]. En otras palabras, no se necesita extender el modelo relacional ya que éste ofrece todo lo necesario para abordar los tipos de problemas que llevaron a la proposición de dicha extensión. El único problema que realmente existía, es que los fabricantes de sistemas gestores de bases de datos habían realizado una implementación muy limitada del modelo relacional.

El concepto de base de datos objeto-relacional goza de una amplia aceptación entre los fabricantes de SGBD relacionales, los cuales incorporan características objeto-relacionales en los que los fabricantes suelen denominar "servidores universales" de bases de datos, calificativo que alude a la capacidad de dichos servidores de abordar cualquier tipo de problema. Estos tipos de SGBD son

conocidos también como *SGBD de tercera generación*, considerando las SGBD relacionales como de segunda generación, y los SGBD pre-relacionales como primera generación.

El estándar SQL:1999 [108, 107] ha establecido los fundamentos de los SGBDs basados en un modelo de base de datos objeto-relacional. El lenguaje SQL que define dicho estándar incorpora los elementos necesarios, tanto en el DDL como en el DML del lenguaje, para el empleo de las características objeto-relacionales.

Particularmente, en SQL:1999 se permite la definición de tipos complejos en base a dos construcciones básicas: los tipos estructurados y las colecciones de valores. La primera de ellas permite la construcción de nuevos tipos en base a un número determinado de valores procedentes de diversos tipos base. Nótese que este tipo de construcción permite representar información compleja de la forma en que se realiza en las bases de datos orientadas a objetos. La segunda construcción permite la creación de tipos cuyos valores son colecciones de tamaño variable de valores de un mismo tipo base. Concretamente, para el caso de SQL:1999, éstas colecciones son *multiconjuntos*, concepto que se introducirá posteriormente en el presente capítulo. Este último tipo de estructuras combinado con el primero ofrece unas capacidades de representación similares a las que permiten las relaciones anidadas y las multirrelaciones. Además de lo anterior, se considera la herencia entre tipos, lo que permite la definición de tipos estructurados como extensión de otros y la sustitución de valores. Finalmente, SQL:1999 da soporte a la asociación de lógica definida por el usuario a los tipos estructurados, constituyendo un mecanismo similar a la encapsulación de lógica en forma de métodos en em modelo orientado a objetos.

2.3. Teoría de conjuntos difusos

La teoría de conjuntos difusos [163] fue desarrollara por L.A. Zadeh en 1965. Dicha teoría constituye la base del paradigma más ampliamente aceptado en la actualidad para la representación y tratamiento de información imperfecta (información afectada por cierta imprecisión o incertidumbre).

Gracias a la teoría de conjuntos difusos es posible modelos matemáticamente *conceptos* imperfectos que son ampliamente empleados por los seres humanos en su comunicación y razonamiento. De esta forma, se dota a los sistemas automáticos de la capacidad para interactuar de una forma mucho más natural con los usuarios, así como de tomar decisiones similares a las que tomarían expertos.

2.3.1. Conjuntos difusos

La noción de conjunto está relacionada con el cumplimiento de una determinada propiedad que deben satisfacer los elementos pertenecientes a éste. Podemos considerar dicha propiedad como una función definida sobre dominio U, que hace corresponder cada uno los elementos de U con un valor de 0 ó 1. Un objeto es un elemento del conjunto si esa función le asigna el valor 1. En caso contrario, cuando se asigne el valor 0, no lo será. Teniendo en cuenta lo anterior, cualquier propiedad P determina un conjunto S_P que está formado como se indica en la Ecuación 2.3.

$$S_P = \{u \in U | P(u) = 1\} \tag{2.3}$$

Nombre	Edad	Sueldo
Ana	23	20.000
Bruno	26	30.000
Clara	62	40.000
David	34	20.000
Eva	30	20.000
Fidel	34	20.000

Tabla 2.7: Datos de los empleados

De la misma forma, cualquier subconjunto $S \subseteq U$ genera una propiedad P_S que viene dada por la expresión que se indica en la Ecuación 2.4.

$$P_S(u) = 1 \iff u \in S \tag{2.4}$$

La teoría de conjuntos difusos generaliza la idea de conjunto, considerando que las propiedades que definen un conjunto son igualmente funciones definidas sobre el universo U, pero que usan como imagen el intervalo cerrado $[0, 1]$. Una propiedad de estas características se denomina *difusa*, y el conjunto que dicha propiedad determina, también denominado *difuso*, está definido como se indica en la Ecuación 2.5

$$S_P = \{\langle u, i\rangle | P(u) = i, u \in U, i \in [0, 1]\} \tag{2.5}$$

Con esta descripción, podemos definir un *conjunto difuso* como se indica a continuación.

Definición 2.8 (Conjunto Difuso)**.** *Un* conjunto difuso *es un conjunto de objetos, cada uno de los cuales tiene asociado un nivel de pertenencia al mismo en el intervalo* $[0, 1]$.

Un conjunto difuso A, definido sobre el universo U, está definido por su función de pertenencia μ_A. Esta función es de la forma $\mu_A : U \to [0, 1]$, de tal manera que para cada elemento u del universo U, $\mu_A(u)$ representa el grado de pertenencia del elemento u al conjunto difuso A.

En los casos en los que el universo U es finito ($U = \{u_1, u_2, \ldots, u_n\}$), por comodidad, se suele definir el conjunto difuso (y, por extensión, su función de pertenencia) de la forma que se indica en la Ecuación 2.6.

$$A = \mu_A(u_1)/u_1 + \mu_A(u_2)/u_2 + \ldots + \mu_A(u_n)/u_n \tag{2.6}$$

Ilustraremos el concepto de conjunto difuso mediante el siguiente ejemplo.

Ejemplo 2.4. *Supongamos que partimos de un conjunto de empleados cuyos datos se indica en la Tabla 2.7. El subconjunto de los empleados jóvenes de la empresa es un subconjunto difuso.*

Obsérvese que el citado subconjunto no puede ser un conjunto crisp *(término que emplearemos para referirnos a los conjuntos clásicos) ya que no existe un criterio estricto y absoluto de edad para determinar los miembros de éste. Para solventar este problema, se podría emplear un criterio gradual que permita reducir paulatinamente la pertenencia de estas personas al subconjunto de los que*

son considerados jóvenes, lo que induciría un conjunto difuso. Este subconjunto difuso, que denominaremos $\tilde{J}$, se puede definir para el caso de los empleados que figuran en la Tabla 2.7 de la forma que se indica en la Ecuación 2.7, donde $\tilde{J}$ se caracteriza como un conjunto de pares $\mu_{\tilde{J}}(x)/x$.

$$\tilde{J} = \{1.0/Ana, 0.9/Bruno, 0.2/David, 0.5/Eva, 0.2/Fidel\} \quad (2.7)$$

Nótese que, en la anterior ecuación, Clara queda excluida totalmente de $\tilde{J}$ ya que, debido a su edad, no pertenece en ningún grado al subconjunto de los empleados jóvenes.

2.3.2. Etiquetas lingüísticas

Como ya se indicó anteriormente, los conjuntos difusos suelen ser empleados para la representación de conceptos vagos, como podría ser, por ejemplo siguiendo el planteamiento del Ejemplo 2.4, el concepto "joven". Podríamos definir el conjunto de edades para las que uno puede ser considerado "joven" como se indica en la Ecuación 2.8.

$$joven = \{1/20, \ldots, 1/25, 0.9/26, 0.8/27, 0.6/29, 0.5/30, \ldots, 0.1/34\} \quad (2.8)$$

El identificador "joven", que lleva asociada una determinada connotación semántica, recibe el nombre de *etiqueta lingüística*.

Según lo anterior, definiremos formalmente el concepto de etiqueta lingüística como se indica a continuación.

Definición 2.9 (Etiqueta lingüística)**.** *Llamaremos* etiqueta lingüística *a aquella palabra, en lenguaje natural, que exprese un conjunto difuso, que puede estar formalmente definido o no.*

Como se hace evidente al lector, en la vida diaria empleamos infinidad de etiquetas lingüísticas como, por ejemplo, "joven", "viejo", "alto", "bajo", "barato", "caro", etc. También es evidente que la definición intuitiva de los conjuntos difusos asociados a dichas etiquetas es muy subjetiva, variando de un individuo a otro y del momento particular. Más aún, se hace evidente que dicha definición es muy dependiente del contexto en que es aplicada. Por ejemplo, la etiqueta lingüística "barato" no tiene asociado el mismo rango de precios cuando se aplica para referirse al precio de un coche que cuando se aplica para referirse al precio de una bicicleta.

2.3.3. Conceptos sobre conjuntos difusos

En la literatura se definen ciertos conceptos básicos que en la presente sección, por brevedad, enumeraremos y definiremos de manera informal. Remitimos a las referencias [84, 85], entre otras, para la obtención de definiciones más formales de los mismos. Dichos conceptos son los siguientes:

- Igualdad de conjuntos difusos: Se dice que dos conjuntos difusos son *iguales* si los valores de pertenencia a uno y otro conjunto son iguales para cada elemento del universo.

- Inclusión de un conjunto difuso en otro: Se dice que un conjunto difuso está incluido en otro conjunto difuso, si el valor de pertenencia al primer conjunto es menor o igual al valor de pertenencia al segundo conjunto para cada elemento del universo.

- Soporte de un conjunto difuso: Es el subconjunto formado por los elementos del dominio cuyo valor de pertenencia al conjunto difuso es mayor que cero. En adelante, notaremos el soporte de un conjunto difuso A como $L_{>0}(A)$.

- α-corte de un conjunto difuso: Es un conjunto clásico compuesto por los elementos del universo cuyo valor de pertenencia al conjunto difuso es mayor o igual a α. Notaremos el α-corte de un conjunto difuso A como $L_\alpha(A)$.

- Núcleo de un conjunto difuso: Es el subconjunto formado por los elementos del dominio cuyo valor de pertenencia al conjunto difuso igual a 1. El núcleo de un conjunto difuso A será notado, en adelante, como $L_1(A)$.

- Altura de un conjunto difuso: La altura de un conjunto difuso es el mayor grado de pertenencia asignado a un elemento de dicho conjunto.

- Conjunto difuso normalizado: Un conjunto difuso se dice normalizado si su altura es 1.

2.3.4. Operaciones sobre conjuntos difusos

Al igual que en el caso anterior, en la literatura se define un cierto número de operaciones sobre conjuntos difusos, las cuales serán introducidas brevemente de manera informal en esta sección, por lo que recomendamos la consulta de las referencias [84, 85] para mayor detalle sobre las mismas. Dichas operaciones son las siguientes:

- Intersección de dos conjuntos difusos: La intersección de dos conjuntos difusos está definida por la función de pertenencia mostrada en la Ecuación 2.9, donde $\otimes$ es una *t-norma* [137].

$$\mu_{A\cap B}(x) = \otimes(\mu_A(x), \mu_B(x)) \qquad \forall x \in U \tag{2.9}$$

- Unión de dos conjuntos difusos: La unión de dos conjuntos difusos se define por una función de pertenencia indicada en la Ecuación 2.10, donde $\oplus$ es una *t-conorma* [137].

$$\mu_{A\cup B}(x) = \oplus(\mu_A(x), \mu_B(x)) \qquad \forall x \in U \tag{2.10}$$

- Complemento de un conjunto difuso: El complemento de un conjunto difuso queda definido según la función de pertenencia descrita en la Ecuación 2.11, donde C es una *negación fuerte* [142].

$$\mu\neg A(x) = C(x) \qquad \forall x \in U \tag{2.11}$$

Las t-normas y las t-conormas son funciones que verifican ciertas propiedades. Éstas que caracterizan la intersección y unión conjunto difusos respectivamente. Dichas funciones se suelen presentar en forma de par t-norma y t-conorma asociada. En la literatura han sido definidos varios de estos pares, siendo los más comunes:

- Operadores *idempotentes*: Las funciones *mínimo* y *máximo* son empleadas, respectivamente, como t-norma y t-conorma. Estos operadores suelen ser muy empleados ya que conservan gran cantidad de las propiedades de los operadores booleanos [13].
- Operadores *arquimedianos*: Se emplea el producto como t-norma y la suma probabilística ($x+y-x*y$) como t-conorma. Estos operadores no satisfacen la propiedad distributiva ni son idempotentes.
- Operadores *acotados*: Los operadores $max(0, x+y-1)$ y $min(1, x+y)$ hacen, respectivamente, de t-norma y de t-conorma. Dichos operadores no satisfacen la idempotencia, la propiedad distributiva ni la propiedad de absorción. Por contra, satisfacen las propiedades conmutativa, asociativa y de identidad.

La negación fuerte es un tipo de función que satisface ciertas condiciones establecidas por E. Trias [142], y que caracteriza la negación de los conjuntos difusos. La negación más empleada es la propuesta por Zadeh [163], que se define de la siguiente forma que se indica en la Ecuación 2.12

$$C(x) = 1 - x \qquad \forall x \in [0,1] \tag{2.12}$$

2.3.5. Números difusos

Los números difusos, ideados con el propósito de analizar y manipular valores aproximados, fueron introducidos inicialmente por Zadeh en la serie de trabajos [165, 166, 167], y posteriormente redefinidos de forma sucesiva por varios autores.

En el presente trabajo emplearemos la definición de número difuso hecha por Dubois y Prade [68]. Ésta es la que sigue.

Definición 2.10 (Número difuso)**.** *Un* número difuso A *es un subconjunto difuso de* U *cuya función de pertenencia es* μ_A*. Dicho subconjunto cumple las siguientes condiciones:*

1. μ_A *es convexa.*
2. μ_A *es semicontinua superiormente.*
3. *El soporte de* A *es un conjunto acotado.*

Algunos autores añaden una condición más al listado anterior, imponiendo la necesidad de que el subconjunto difuso esté normalizado.

La forma general de la función de pertenencia de un número difuso A se muestra en la Ecuación 2.13, donde r_A y s_A son funciones de la forma $r_A, s_A : U \to [0,1]$, siendo r_A no decreciente, y s_A no creciente. Además, las funciones

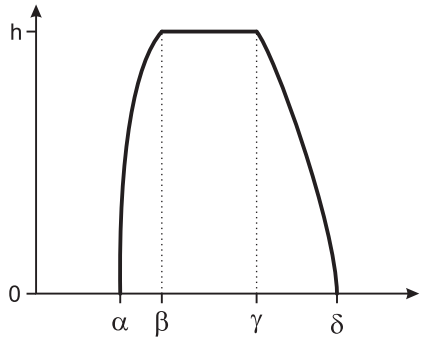

Figura 2.1: Número difuso generalizado.

anteriores han de cumplir la condición $r_A(\beta) = h = s_A(\gamma)$. Finalmente, se define, $h \in (0, 1]$ como la *altura* del número difuso, y $\alpha, \beta, \gamma, \delta \in U$.

$$\mu_A(x) = \begin{cases} r_A(x) & \text{si } x \in [\alpha, \beta) \\ h & \text{si } x \in [\beta, \gamma) \\ s_A(x) & \text{si } x \in [\gamma, \delta) \\ 0 & \text{en otro caso} \end{cases} \qquad \forall x \in [0, 1] \tag{2.13}$$

La Figura 2.1 muestra una representación gráfica de un número difuso genérico.

Un caso particular de los números difusos, muy empleado a lo largo de esta memoria, es el de los número difusos *trapezoidales*. Un número difuso es trapezoidal si las funciones r_A y s_A son funciones lineales. En dicho caso, la función de pertenencia queda definida de la forma que se indica en la Ecuación 2.14.

$$\mu_A(x) = \begin{cases} h + \frac{(x-\beta)h}{\beta-\alpha} & \text{si } x \in [\alpha, \beta) \\ h & \text{si } x \in [\beta, \gamma) \\ h - \frac{(x-\gamma)h}{\delta-\gamma} & \text{si } x \in [\gamma, \delta) \\ 0 & \text{en otro caso} \end{cases} \qquad \forall x \in [0, 1] \tag{2.14}$$

En general, se suele considera $h = 1$, por lo que un número trapezoidal A puede ser caracterizado por cuatro parámetros de la forma $A \equiv (\alpha, \beta, \gamma, \delta)$. Finalmente, se ha de destacar que en los casos en que $\beta = \gamma$ se puede denominar al número difuso trapezoidal como número difuso *triangular*.

Las figuras 2.2 y 2.3 muestran una representación gráfica de los números difusos trapezoidales y triangulares. Nótese que dichas representaciones se refieren a números difusos normalizados.

2.3.6. Principio de extensión

El principio de extensión fue propuesto por Zadeh [165, 166, 167] con el objeto de proveer de un método general que permitiese extender conceptos matemáticos no difusos para el tratamiento de cantidades difusas.

Este principio se define formalmente como se indica a continuación.

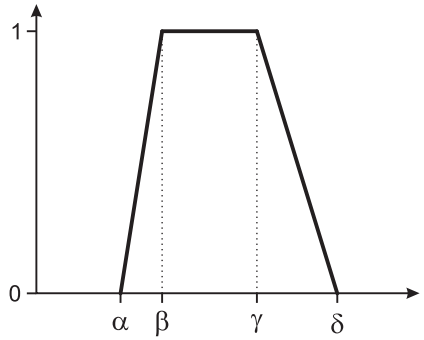

Figura 2.2: Número difuso trapezoidal normalizado.

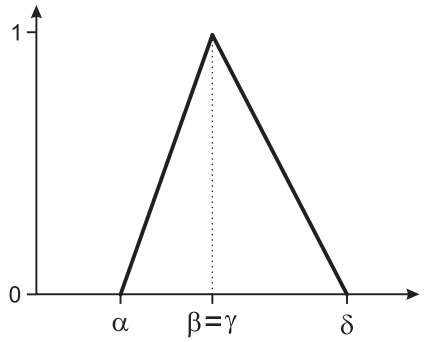

Figura 2.3: Número difuso triangular normalizado.

Definición 2.11 (Operadores extendidos para cardinalidades difusas)**.** *Sean* $S, S_1, S_2, \ldots, S_n$ *una serie de conjuntos* crisp. *Sea* f *una función de la forma* $f : S_1, S_2, \ldots, S_n \mapsto S$.

La extensión de f*, notada como* $\widetilde{f}$*, es la transposición de* f *al marco de los números difusos. Ésta es de la forma* $\widetilde{f} : \widetilde{P}(S_1), \widetilde{P}(S_2), \ldots, \widetilde{P}(S_n) \mapsto \widetilde{P}(S)$ *y está definida según la función de pertenencia descrita en la Ecuación 2.15.*

$$\mu_{\widetilde{f}(\widetilde{S}_1,\widetilde{S}_2,\ldots,\widetilde{S}_n)}(s) = \sup\{\mu_{\widetilde{S}_1}(s_1) \otimes \mu_{\widetilde{S}_2}(s_2) \otimes \ldots \otimes \mu_{\widetilde{S}_n}(s_n) | \\ s_1 \in S_1 \ \wedge \ s_2 \in S_2 \ \wedge \ \ldots \ \wedge \ s_n \in S_n \ \wedge \ f(s_1, s_2, \ldots, s_n) = s\}$$

$$\forall s \in S, \forall \widetilde{S}_1 \in \widetilde{P}(S_1), \forall \widetilde{S}_2 \in \widetilde{P}(S_2), \ldots, \forall \widetilde{S}_n \in \widetilde{P}(S_n) \quad (2.15)$$

2.3.7. Aritmética difusa

El principio de extensión, descrito anteriormente, hace posible transponer, con mucha facilidad, las operaciones aritméticas clásicas al tratamiento de números difusos. De esta forma, se definen las siguientes operaciones sobre conjuntos difusos:

- Suma extendida: La función de pertenencia de la suma extendida de dos cantidades difusas A_1 y A_2 está definida según la función de pertenencia indicada en la Ecuación 2.16.

$$\mu_{A_1 \boxplus A_2}(x) = \sup\{\text{mín}(\mu_{A_1}(y - x), \mu_{A_2}(y)) \mid y \in U\} \qquad \forall x \in U \quad (2.16)$$

- Diferencia extendida: La diferencia extendida de dos cantidades difusas A_1 y A_2 se define según la función de pertenencia descrita en la Ecuación 2.17.

$$\mu_{A_1 \boxminus A_2}(x) = \sup\{\text{mín}(\mu_{A_1}(y + x), \mu_{A_2}(y)) \mid y \in U\} \qquad \forall x \in U \quad (2.17)$$

- Producto extendido: El producto extendido de dos cantidades difusas A_1 y A_2 está caracterizado por la función que se muestra en la Ecuación 2.18.

$$\mu_{A_1 \boxtimes A_2}(x) = \begin{cases} \sup\{\text{ mín}(\mu_{A_1}(x/y), \mu_{A_2}(y)) \mid \\ \qquad y \in (U - \{0\})\} & \text{si } x \neq 0 \\ \sup\{\text{ mín}(\mu_{A_1}(y), \mu_{A_2}(z)) \mid \\ \qquad y \in U \ \wedge \ z \in U \ \wedge \ (y = 0 \ \vee \ z = 0)\} & \text{si } x = 0 \end{cases}$$

$$\forall x \in U \quad (2.18)$$

- División extendida: La división extendida de dos números difusos A_1 y A_2 queda definida por la función de pertenencia indicada en la Ecuación 2.19.

$$\mu_{A_1 \boxslash A_2}(x) = \sup\{\text{mín}(\mu_{A_1}(x \cdot y), \mu_{A_2}(y)) \mid y \in U\} \qquad \forall x \in U \quad (2.19)$$

Nótese que, en las anteriores definiciones, se ha asumido como t-norma la función mínimo.

2.3.8. Lógica difusa

La teoría de conjuntos difusos ha generado una extensión de la lógica, la lógica difusa, que nos permite obtener respuestas aproximadas a partir de un conjunto de hechos que pueden ser inciertos.

La lógica clásica sólo admite como valores de verdad los valores *verdadero* y *falso*. Desde la lógica clásica se ha evolucionado hacia las lógicas multivaluadas [143], en las que se puede elegir entre un conjunto finito de posibles valores de verdad. La lógica difusa permite utilizar valores de verdad graduados.

2.3.8.1. Operadores lógicos básicos

Anteriormente, se ha indicado que las *t-normas*, *t-conormas* y *negaciones* se han empleado, respectivamente, para la caracterización de las operaciones de *intersección*, *unión* y *negación* sobre conjuntos difusos. En el ámbito de la lógica difusa, las operaciones de *conjunción*, *disyunción* y *negación*, también se generalizan empleando dichas familias de operadores. La *conjunción* será modelada utilizando *t-normas*, la *disyunción* se modelará utilizando *t-conormas*, y la *negación* utiliza la familia de negaciones.

2.3.8.2. Implicación

La implicación clásica ha de ser también generalizada en la lógica difusa. Dicha implicación queda modelada por un operador que queda definido de tal como se indica en la siguiente definición.

Definición 2.12 (Implicación difusa)**.** *Una* implicación difusa *es una función* $I : [0,1] \times [0,1] \to [0,1]$ *que verifica las siguientes propiedades:*

1. $\beta \geq \gamma \Rightarrow I(\alpha,\beta) \geq I(\alpha,\gamma),\ \forall \alpha,\beta,\gamma \in [0,1]$
2. $\beta \geq \gamma \Rightarrow I(\beta,\alpha) \leq I(\gamma,\alpha),\ \forall \alpha,\beta,\gamma \in [0,1]$
3. $I(0,\alpha) = 1\ \forall \alpha \in [0,1]$
4. $I(\alpha,0) = \alpha\ \forall \alpha \in [0,1]$
5. $I(\alpha, I(\beta,\gamma)) = I(I(\alpha,\beta),\gamma)\ \forall \alpha,\beta,\gamma \in [0,1]$

En la literatura se proponen una amplia gama de operadores de implicación. La Tabla 2.8 relaciona los más significativos.

2.4. Multiconjuntos difusos

En la presente sección introduciremos los multiconjuntos difusos, elementos matemáticos resultantes de combinar los conceptos de la teoría de conjuntos difusos y los multiconjuntos.

Para ello, se introducirá brevemente en primer lugar el concepto de multiconjunto. Esto facilitará la posterior introducción de los multiconjuntos difusos,

Nombre	Definición
Dienes	$I(\alpha,\beta) = max(\beta, 1-\alpha)$
Lukasiewicz	$I(\alpha,\beta) = min(1, 1-\alpha+\beta)$
Kleene-Dienes	$I(\alpha,\beta) = max(1-\alpha, \beta)$
Reichenbach	$I(\alpha,\beta) = 1-\alpha+\alpha\cdot\beta$
Gödel	$I(\alpha,\beta) = \begin{cases} 1 & \text{si } \alpha \leq \beta \\ \beta & \text{si } \alpha > \beta \end{cases}$
Gödel Recíproca	$I(\alpha,\beta) = \begin{cases} 1 & \text{si } \alpha \leq \beta \\ 1-\alpha & \text{si } \alpha > \beta \end{cases}$
Goguen	$I(\alpha,\beta) = \begin{cases} 1 & \text{si } \alpha \leq \beta \\ \beta/\alpha & \text{si } \alpha > \beta \end{cases}$
N_1	$I(\alpha,\beta) = \begin{cases} max(\beta, 1-\alpha) & \text{si } \alpha \leq \beta \\ \beta & \text{si } \alpha > \beta \end{cases}$
N_2	$I(\alpha,\beta) = \begin{cases} max(\beta, 1-\alpha) & \text{si } \alpha \leq \beta \\ 1-\alpha & \text{si } \alpha > \beta \end{cases}$

Tabla 2.8: Operadores de implicación más significativos.

que vendrá seguida por un breve repaso de las caracterización de los mismos en la literatura. Para ambos artefactos matemáticos, se describirán las operaciones más comunes en la bibliografía.

2.4.1. Multiconjuntos

Al igual que la teoría de conjuntos difusos generaliza a la teoría de conjuntos clásicos, permitiendo que la pertenencia de los elementos de éstos sea graduada, otra posible generalización de los conjuntos clásicos consiste en permitir que la pertenencia de los elementos de éstos sea múltiple. Esta idea, aparentemente empleada por primera vez [21] en 1888 por Richard Dedekind [64], se introduce en la bibliografía relacionada con las Ciencias de la Computación gracias a Knuth [86] y, posteriormente, se estudia en profundidad por Blizard [20, 19, 22].

En la literatura podemos encontrar diferentes denominaciones para estas estructuras matemáticas (*heap*, *bunch*, etc.) [138] siendo los términos *multiset* y *bag* los más comunes. Dado que en castellano la traducción literal del término *bag* puede resultar inapropiada, y visto que algunos autores consideran esta denominación y otras alternativas un tanto vulgares [138], en el presente trabajo denominaremos a estas estructuras como *multisets* (Término acuñado por N. G. Bruijin [86]) o, en castellano, *multiconjuntos*.

A continuación, definiremos formalmente el concepto de multiconjunto.

Definición 2.13 (Multiconjunto)**.** *Sea U un conjunto.*

Un multiconjunto M definido sobre el universo U es un par $M = \langle U, \omega_M \rangle$, donde ω_M representa una función de la forma $\omega_M : U \mapsto \mathbb{N}$.

En adelante, la función ω_M que será denominada como función de conteo *de M.*

Nótese que, en la definición anterior, el símbolo $\mathbb{N}$ representa el conjunto de los números naturales. Además, obsérvese que la función de conteo de un multiconjunto es análoga a la función característica de un conjunto *crisp* o difuso.

Ilustraremos el concepto de multiconjunto mediante el siguiente ejemplo.

Ejemplo 2.5. *Siguiendo con el marco planteado en el Ejemplo 2.4, supongamos que deseamos obtener información sobre los sueldos de los empleados cuyos datos figuran en la Tabla 2.7.*

Partiendo de los datos contenidos en la citad tabla, podemos determinar el conjunto crisp *de los sueldos de los empleados. Éste, que denominaremos S, está definido como se indica en la Ecuación 2.20.*

$$S = \{20.000, 30.000, 40.000\} \tag{2.20}$$

El conjunto S informa sobre los distintos sueldos que son pagados a los empleados. No obstante, nótese que, debido a las características de la estructura matemática empleada para contener estos datos, la datos sobre los sueldos contenidos en S resultan menos informativos que los contenidos en la Tabla 2.7. Concretamente, si contamos con los datos contenidos en la citada tabla, podremos apreciar que, aunque los distintos sueldos de los empleados son los indicados en S, el primero de ellos (según el orden en que se indica en la Ecuación 2.20) es el más común entre los empleados.

Lo expuesto anteriormente evidencia que el uso de un conjunto crisp *en el caso que nos ocupa, ha provocado la pérdida de la dimensión cuantitativa de los datos. En cambio, si se emplea un multiconjunto como estructura para contener éstos, dicha dimensión se conservará. Partiendo de los datos contenidos en la Tabla 2.7 el multiconjunto de los sueldos de los empleados, notado como $\hat{S}$, se define como se indica en la Ecuación 2.21. Obsérvese que, en dicha ecuación, $\hat{S}$ está definido (siguiendo el mismo esquema que se emplea para los conjuntos difusos) como un conjunto de pares $\omega_{\hat{S}}(x)/x$.*

$$\hat{S} = \{4/20.000, 1/30.000, 1/40.000\} \tag{2.21}$$

Como se puede observar, contando con los datos (e información cuantitativa sobre éstos) contenidos en el multiconjunto $\hat{S}$, es posible determinar, además del rango de sueltos de los empleados, cómo de común es cada uno de ellos.

2.4.2. Operaciones sobre multiconjuntos

Con respecto a las operaciones que se pueden aplicar sobre los multiconjuntos, en la literatura se pueden encontrar versiones adaptadas de los operadores que se aplican habitualmente sobre conjuntos. Además de los anteriores, se pueden encontrar ciertos operadores exclusivos del ámbito de los multiconjuntos, definidos como extensiones de los operadores de conjuntos en el ámbito de estas estructuras. El siguiente listado describe brevemente los operadores más comunes. Remitimos al lector a [155, 138] para más detalles sobre éstas.

- Intersección: La intersección de dos multiconjuntos A y B definidos sobre un universo U, notada como $A \cup B$, es un multiconjunto definido por la función de conteo que se indica en la Ecuación 2.22.

$$\omega_{(A\cup B)}(u) = \text{mín}(\omega_A(u), \omega_B(u)) \qquad \forall u \in U \tag{2.22}$$

- Unión: La unión de dos multiconjuntos A y B definidos sobre un universo U, notada como $A \cap B$, es un multiconjunto definido por la función de conteo indicada en la Ecuación 2.23.

$$\omega_{(A\cap B)}(u) = \text{máx}(\omega_A(u), \omega_B(u)) \qquad \forall u \in U \tag{2.23}$$

- Unión aditiva: La union aditiva de dos multiconjuntos A y B definidos sobre un universo U, notada como $A \uplus B$, es un multiconjunto definido por la función de conteo descrita en la Ecuación 2.24.

$$\omega_{(A\uplus B)}(u) = \omega_A(u) + \omega_B(u) \qquad \forall u \in U \tag{2.24}$$

- Diferencia: La diferencia de dos multiconjuntos A y B definidos sobre un universo U, notada como $A - B$, es un multiconjunto definido por la función de conteo especificada en la Ecuación 2.25.

$$\omega_{(A-B)}(u) = \text{máx}(\omega_A(u) - \omega_B(u), 0) \qquad \forall u \in U \tag{2.25}$$

Obsérvese que las adiciones y omisiones, con respecto al juego habitual de operadores de conjuntos, del anterior listado. En primer lugar, se ha de notar la omisión del operador de complementación. Éste, dado que en el ámbito de los multiconjuntos la ocurrencia de los elementos no es absoluta, carece de sentido. En segundo lugar, se ha de notar la adición del operador de *unión aditiva*. Este operador viene motivado por las diversas formas en las que se pueden agregar las ocurrencias de los elementos de un par de multiconjuntos, variedad que no se da para el caso de los conjuntos.

Además de los anteriores operadores, se han definido en la bibliografía los siguientes operadores relacionales para multiconjuntos:

- Igualdad: Dos multiconjuntos A y B son iguales, notado como $A = B$, si, y sólo si, se cumple la condición establecida en la Ecuación 2.26.

$$A = B \iff \forall u \in U, \omega_A(u) = \omega_B(u) \tag{2.26}$$

- Inclusión: El multiconjunto A está incluido en el multiconjunto B , notado como $A \subseteq B$, si se satisface la expresión descrita en la Ecuación 2.27.

$$A \subseteq B \iff \forall u \in U, \omega_A(u) \leq \omega_B(u) \tag{2.27}$$

- Inclusión estricta: El multiconjunto A está incluido estrictamente en el multiconjunto B , notado como $A \subset B$, si es satisfecha la expresión indicada en la Ecuación 2.28.

$$A \subseteq B \iff \forall u \in U, \omega_A(u) < \omega_B(u) \tag{2.28}$$

Finalmente, definiremos la cardinalidad de un multiconjunto A definido sobre un universo U como se indica en la Ecuación 2.29.

$$|A| = \sum_{u \in U} \omega_A(u) \tag{2.29}$$

2.4.3. Multiconjuntos difusos

Las líneas de generalización sobre la teoría de conjuntos *crisp* de que la teoría de conjuntos difusos y de los multiconjuntos convergen en el concepto de *multiconjunto difuso*. Esta estructura matemática es una agrupación en la que se permiten múltiples ocurrencias de sus elementos y en la que, para permitir la pertenencia parcial de los elementos, cada ocurrencia está graduada.

2.4.3.1. Definición de multiconjunto difuso

A diferencia de otras estructuras, existen en la literatura múltiples propuestas de definición del concepto de multiconjunto difuso.

Este tipo de estructura fue introducida, junto con la primera propuesta de definición del concepto de multiconjunto difuso, por Yager [155]. Posteriormente, estas estructuras han sido estudiadas por diversos autores. Como resultado de estos estudios, se encuentran extensiones de la definición original y definiciones alternativas para ofrecer mayores posibilidades y solventar carencias de ésta.

Multiconjuntos difusos de Yager.

En [155], se define un multiconjunto difuso (que denominaremos *Multiconjunto Difuso de Yager* para distinguirlo de otras propuestas) de la forma que se indica a continuación.

Definición 2.14 (Multiconjunto difuso de Yager)**.** *Sea U un conjunto.*

Un multiconjunto difuso de Yager M definido sobre el universo U es un par $M = \langle U, \Psi_M \rangle$, donde Ψ_M es una función de la forma $\Psi_M : U \mapsto Q$. En la anterior expresión, Q es un conjunto definido como se indica en la Ecuación 2.30.

$$Q = \{\langle [0,1], \omega \rangle | \omega : [0,1] \mapsto \mathbb{N}\} \tag{2.30}$$

Obsérvese que, en la definición anterior, Q representa el conjunto de todas los posibles multiconjuntos definidos sobre los elementos del intervalo $[0,1]$. De esta forma, las ocurrencias de los elementos de un multiconjunto difuso están caracterizadas por un multiconjunto *crisp* de grados de pertenencia.

Ilustraremos el concepto de multiconjunto difuso de Yager mediante el siguiente ejemplo.

Ejemplo 2.6. *Continuando con el marco empleados en los ejemplos 2.4 y 2.5, supongamos que, partiendo de los datos en la Tabla 2.7, deseamos obtener información sobre los sueldos de los empleados jóvenes.*

En el Ejemplo 2.4 se determinó el subconjunto difuso de los empleados jóvenes, en el que los grados de pertenencia al citado subconjunto proporcionan información cualitativa sobre la juventud cada empleado. Por otra parte, en el Ejemplo 2.5 se determinó el multiconjunto de los sueldos de los empleados. En

el citado ejemplo, se mostró que la cardinalidad asociada a cada sueldo proporcionaba una valiosa información cuantitativa sobre éste.

Según lo anterior, al buscar información sobre el sueldo de los empleados jóvenes, obtendremos un resultado que combinará información cualitativa y cuantitativa sobre los datos. Por tanto, parece apropiado el uso de un multiconjunto difuso de Yager como estructura para contender los citados datos.

A partir de los datos de la Tabla 2.7 y de la definición de la etiqueta lingüística "joven" realizada en la Subsección 2.3.2, podemos definir el multiconjunto difuso de Yager que contiene los sueldos de los empleados jóvenes, notado como $\breve{S}$*, como se indica en la Ecuación 2.31. En dicha ecuación,* $\breve{S}$ *se caracteriza como un conjunto de pares* $\Psi_{\breve{S}}(x)/x$*, donde, a su vez, cada* $\Psi_{\breve{S}}(x)$ *es un multiconjunto* crisp *de grados caracterizado por pares cardinalidad/grado.*

$$\breve{S} = \{\ \{1/1.0, 1/0.5, 2/0.2\}/20.000\ ,\ \{1/0.9\}/30.000\ \} \tag{2.31}$$

Obsérvese que $\breve{S}$ *contiene los mismos elementos que el conjunto* S *definido en el Ejemplo 2.5, así como la información cuantitativa que figura en* $\hat{S}$*, multiconjunto definido en el mismo ejemplo. Los grados de cada una de las ocurrencias se corresponde con los grados asignados a los empleados de los que éstas provienen en el conjunto difuso* $\tilde{J}$ *definido en el Ejemplo 2.4. De esta forma, se aporta una dimensión cualitativa (en lo que se refiere a la pertenencia al grupo de empleados jóvenes) a cada una de las ocurrencias.*

La propuesta de Yager incluye, además de la definición anterior, la transposición de los operadores más habituales sobre multiconjuntos *crisp* al nuevo marco difuso. Para más detalles sobre este conjunto de operadores, remitimos al lector a [155, 156].

Definiciones Alternativas.

Podemos encontrar en la literatura diversas propuestas alternativas a la definición de Yager. De entre ellas, destacan la siguientes referencias según su orden cronológico:

- Chakrabarty et al. [41]: Estos autores proponen extender, al marco de los multiconjuntos difusos de Yager, el concepto de cardinalidad negativa introducido por Blizard [18]. Éste introdujo la posibilidad de contar con cardinalidades negativas en multiconjuntos *crisp* para ofrecer una herramienta que útil en aplicaciones del campo de la física como, por ejemplo, estudios sobre materia-antimateria, materia oscura, etcétera. La transposición de esta idea al marco de los multiconjuntos difusos permite combinar la potencia de ambos.

- Miyamoto [109]: En este estudio se trata el caso en que los multiconjuntos difusos contienen elementos con un número infinito de ocurrencias graduadas y su relación con los multiconjuntos *crisp*.

- Rocacher [126]: En el anterior trabajo se hace una propuesta alternativa (a la original realizada por Yager) en la que las cardinalidades de un multiconjunto difuso se caracterizan mediante cardinalidades difusas. Dado que esta propuesta será de interés para el presente trabajo, analizaremos ésta en más detalle a continuación. Además de la propuesta anterior, con

posterioridad a la misma y con intención de mejorar algunos aspectos de ésta, Rocacher junto con Bosc proponen la caracterización de las cardinalidades de un multiconjunto difuso empleando enteros difusos relativos [32] y números racionales difusos [127].

- Delgado et al. [66, 67]: En estos trabajos se propone extender la estructura de los multiconjuntos difusos de tal forma que se pueda incorporar en la misma información asociativa en forma de pares propiedad-valor. La información cuantitativa de las ocurrencias será empleada para indicar la cardinalidad de los valores en dichas asociaciones.

- Chakrabarty [40]: En esta ocasión, el autor propone la caracterización de las cardinalidades de un multiconjunto difuso como una lista de grados ordenados de forma decreciente. En esencia, esta propuesta es muy similar a la realizada por Rocacher [126]. No obstante, carece de la potencia de esta última, especialmente en lo que se refiere a la capacidad del marco para acomodar otro tipo de estructuras, concretamente multiconjuntos *crisp*, conjuntos difusos y conjuntos *crisp*, como casos particulares.

- Miyamoto [110]: En esta nueva propuesta, Miyamoto caracteriza las cardinalidades de un multiconjunto empleando una función monótona decreciente. Este marco, permite caracterizar de una misma manera los multiconjuntos *crisp*, los multiconjuntos reales [17] (en los que la cardinalidad está caracterizada por un número real) y multiconjuntos difusos. En lo que se refiere a este último tipo de multiconjuntos, la propuesta es similar a la realizada por Rocacher [126] aunque la caracterización empleada resulta inapropiada en la práctica.

Multiconjuntos difusos de Rocacher.

De entre la propuestas anteriores, como se ha señalado previamente, consideramos de especial relevancia la propuesta realizada por Rocacher [126]. Ésta introduce una caracterización alternativa de las cardinalidades de los elementos de los multiconjuntos difusos para solventar algunas deficiencias detectadas en la propuesta de Yager.

Se ha de notar que la definición de los multiconjuntos difusos de Yager permite considerar a los multiconjuntos *crisp* y a los conjuntos difusos como casos particulares. No obstante, como observa Rocacher [126], la aplicación de los operadores de intersección, unión e inclusión sobre multiconjuntos difusos de Yager equivalentes a determinados conjuntos difusos no resulta coherente con los resultados obtenidos al aplicar esos mismos operadores según se definen en el marco de la teoría de conjuntos difusos.

Detectada la problemática anterior, Rocacher proponer un nuevo marco de caracterización de los conjuntos difusos que la solvente, de tal forma que multiconjuntos difusos, multiconjuntos *crisp*, conjuntos difusos y conjuntos *crisp* puedan ser tratados de una manera uniforme. En dicha propuesta, las cardinalidades de los elementos de un multiconjunto difuso son caracterizadas mediante *cardinalidades difusas* [168]. Este concepto se define como se indica a continuación.

Definición 2.15 (Cardinalidad difusa de un conjunto difuso)**.** *Sea A un conjunto difuso.*

La cardinalidad difusa *de A es un número difuso, notado en adelante como $FGCOUNT(A)$, cuya función característica está definida como se indica en la Ecuación 2.32.*

$$\mu_{FGCOUNT(A)}(n) = \sup\{\alpha \in [0,1] : |L_\alpha(A)| \geq n\} \qquad \forall n \in \mathbb{N} \tag{2.32}$$

De la anterior definición, se puede observar que una cardinalidad difusa siempre será un número difuso definido sobre $\mathbb{N}$, normalizado y cuya función característica será decreciente.

El siguiente ejemplo ilustra el concepto de cardinalidad difusa.

Ejemplo 2.7. *Supongamos que A es un conjunto difuso definido de la forma $A = \{1.0/x_1, 0.5/x_2, 0.1/x_3\}$.*

Según la definición anterior, La cardinalidad difusa de A es un número difuso de la forma $FGCOUNT(A) = \{1.0/0, 1.0/1, 0.5/2, 0.1/3\}$.

Obsérvese que el grado de pertenencia α de cada número natural n de la cardinalidad difusa de un conjunto difuso A se interpreta como el grado en que A está compuesto de, al menos, n elementos. Dado que la función de pertenencia de una cardinalidad difusa es convexa, éstas pueden representarse de forma más compacta omitiendo los grados de los números naturales que poseen un grado de pertenencia similar al de su inmediato superior. De esta forma, la cardinalidad difusa descrita en el ejemplo anterior puede ser expresada según la siguiente notación $FGCOUNT(A) = \{1.0/1, 0.1/3\}_c$. Nótese que se emplea el subíndice 2 para permitir diferenciar los casos en que se hace uso de la notación abreviada descrita anteriormente.

Una vez definido el concepto de cardinalidad difusa, definiremos el concepto de multiconjunto difuso según la propuesta de Rocacher como se indica a continuación.

Definición 2.16 (Multiconjunto difuso de Rocacher)**.** *Sea U un conjunto.*

Un multiconjunto difuso de Rocacher M definido sobre el universo U es un par $M = \langle U, \Omega_M \rangle$, donde Ω_M es una función de la forma $\Omega_M : U \mapsto \mathbb{N}_f$. En la anterior expresión, $\mathbb{N}_f$ se corresponde con el conjunto todas las posibles cardinalidades difusas.

En adelante, la función Ω_M será denominada genéricamente como la función de conteo *de M.*

Ilustraremos el concepto de multiconjunto difuso de Rocacher mediante el siguiente ejemplo.

Ejemplo 2.8. *Traspongamos el requerimiento de información sobre el sueldo de los empleados jóvenes, abordado en el Ejemplo 2.6, empleando, en esta ocasión, un multiconjunto difuso de Rocacher como estructura para contener los datos.*

A partir de los datos de la Tabla 2.7 y de la definición de la etiqueta lingüística "joven" realizada en la Subsección 2.3.2, podemos definir el multiconjunto difuso de Rochacher que contiene los sueldos de los empleados jóvenes, notado como $\check{S}'$, como se indica en la Ecuación 2.33. En dicha ecuación, $\check{S}'$ está caracterizado como un conjunto de pares $\Omega_{\check{S}'}(x)/x$, donde $\Omega_{\check{S}'}(x)$ es una cardinalidad difusa expresada en forma compacta.

$$\breve{S}' = \{\ \{1.0/1, 0.5/2, 0.2/4\}_c/20.000\ ,\ \{1.0/0, 0.9/1\}_c/30.000\ \} \tag{2.33}$$

Nótese la equivalencia de $\breve{S}'$ y el multiconjunto difuso de Yager $\breve{S}$, definido en el Ejemplo 2.6, así como la diferencia la caracterización de sus cardinalidades.

Finalmente, se ha de destacar la enorme ventaja que la anterior propuesta presenta en lo que se refiere al tratamiento uniforme, bajo un mismo marco, de distintas estructuras conjuntivas (conjuntos *crisp*, conjuntos difusos, multiconjuntos *crisp* y conjuntos difusos). Este hecho, convierte la propuesta en una herramienta fundamental para la creación de sistemas que permitan el tratamiento de cualquiera de las estructuras conjuntivas mencionadas, así como la interoperatividad de las mismas. Precisamente, ésta es una de las características que se buscan para las propuestas que se realizan los siguientes capítulos del presente trabajo. Por tanto, en el ámbito de esta memoria, consideraremos cualquier multiconjunto difuso como un multiconjunto difuso de Rocacher y, en adelante, denominaremos a este tipo de estructuras simplemente como *multiconjuntos difusos.*

2.4.3.2. Operaciones sobre multiconjuntos difusos

Gracias a la aportación de Rocacher, las operaciones sobre multiconjuntos difusos se definen de la misma forma que las operaciones sobre multiconjuntos *crisp*. Recordemos que las operaciones de multiconjuntos *crisp* se basan en la aritmética de los números naturales que caracterizan las cardinalidades de sus elementos. Por tanto, para el caso de los multiconjuntos difusos, será necesario disponer de una aritmética para las cardinalidades difusas.

Dado que las cardinalidades difusas son un casos particulares de números difusos, podremos emplear la aritmética extendida de números difusos, introducida anteriormente en la Subsección 2.3.7, para operar con éstos. No obstante, se han de tener en cuenta ciertas particularidades de la aritmética de cardinalidades difusas ya que éstas se definen sobre $\mathbb{N}$. Concretamente, será necesario emplear definiciones alternativas para los operadores extendidos de substracción y división, como veremos en detalle más adelante, para superar ciertas dificultades.

Una vez disponemos de una aritmética para las cardinalidades difusas, la definición de los operadores de intersección, unión y unión aditiva de multiconjuntos difusos es inmediata a partir de los operadores de los multiconjuntos *crisp*. En el caso del operador de diferencia, dadas las dificultades que presenta el operador extendido de substracción, será necesario considerar de forma más detallada su definición, cuestión que abordaremos en el siguiente epígrafe.

En lo que respecta a los operadores relacionales de los multiconjuntos difusos, éstos también se definen de la misma manera que sus homólogos en el caso de multiconjuntos *crisp*. No obstante, se ha de tener en cuenta que los operadores relacionales $<$ y $\leq$, aplicados respectivamente en la definición de los operadores de inclusión e inclusión estricta, sólo inducen en $\mathbb{N}_f$ un orden parcial. Para solventar este inconveniente, el operador de inclusión de multiconjuntos difusos se define de la forma que se indica a continuación.

Definición 2.17 (Inclusión de multiconjuntos difusos)**.** *Sea U un conjunto. Sean A y B dos multiconjuntos difusos definidos sobre el universo U.*

Diremos que A está incluido en B, notado como $A \subseteq B$, si se satisface la condición indicada en la Ecuación 2.34, donde $\Rightarrow_f$ representa un operador de implicación difuso.

$$A \subseteq B \iff \forall u \in U, \forall n \in \mathbb{N}, \mu_{\Omega_A(u)}(n) \Rightarrow_f \mu_{\Omega_B(u)}(n) \tag{2.34}$$

Finalmente, la cardinalidad de un multiconjunto se define como la suma de las cardinalidades de sus elementos. Formalmente, éste concepto queda definido como se indica a continuación.

Definición 2.18 (Cardinalidad de un multiconjunto difuso)**.** *Sean A un multiconjunto difuso definido sobre el universo U.*

La cardinalidad de A es una cardinalidad difusa, notada como $|A|$, definida como se indica en la Ecuación 2.35. En dicha ecuación, la sumatoria se corresponde con la aplicación del operador extendido de suma para números difusos.

$$|A| = \sum_{u \in U} \Omega_A(u) \tag{2.35}$$

Diferencia de multiconjuntos difusos

La diferencia de multiconjuntos difusos ha de ser definida con especial cuidado ya que, para este tipo de estructuras, este operador no induce un operador de complemento. En la teoría de conjuntos *crisp*, el operador de diferencia puede ser definido para dos conjuntos *crisp* A y B sin necesidad de recurrir al operador de complemento de la forma que se indica en la Ecuación 2.36.

$$A - B = S \iff A = (A \cap B) \cup S \tag{2.36}$$

Para el caso de los multiconjuntos *crisp*, el operador de diferencia tiene asociada una semántica aditiva de tal forma que, para dos multiconjuntos *crisp* A y B, $A - B$ resulta en el multiconjunto *crisp* S que ha de ser *añadido* (unido aditivamente) a $A \cap B$ para que el resultado sea igual a A. El anterior planteamiento se formaliza en la Ecuación 2.37.

$$A - B = S \iff A = (A \cap B) \uplus S \tag{2.37}$$

La definición del operador de diferencia de multiconjuntos difusos, en base a la definición para el caso de multiconjuntos *crisp*, exige que se cumpla una igualdad análoga a la indicada en la Ecuación 2.37 en el caso de los multiconjuntos difusos. Dicha igualdad, para el caso de dos multiconjuntos difusos A y B definidos sobre un universo U, se corresponde con la indicada en la Ecuación 2.38.

$$A - B = S \iff \forall u \in U, \Omega_A(u) = \Omega_{(A \cap B)}(u) \boxplus \Omega_S(u) \tag{2.38}$$

Desafortunadamente, es conocido [126, 135] que $X = N \boxminus M$ no es la solución para la ecuación $M \boxplus X = N$, siendo M y N números difusos. La solución a la anterior ecuación, cuando ésta existe, se obtiene sustituyendo el operador extendido de diferencia por el la *diferencia optimista*. La diferencia optimista de dos números difusos N y M, notada como $N \widehat{\boxminus} M$, se define como se indica en la Ecuación 2.39, donde el operador $\wedge^{-1}$ está definido según la Ecuación 2.40.

$$\mu_{(N\widehat{\boxminus}M)}(x) = \inf_{(n,m)\in\mathbb{N}\times\mathbb{N}|m+x=n} \mu_M(m) \wedge^{-1} \mu_N(n) \qquad \forall x \in \mathbb{N} \tag{2.39}$$

$$x \wedge^{-1} y = \sup\{t|t \in [0,1] \ \wedge \ \text{mín}(x,t) \leq y\} \qquad \forall x, y \in [0,1] \tag{2.40}$$

Nótese que, como se ha señalado anteriormente, la solución a la ecuación $M \boxplus X = N$ no siempre existe. Si esta ecuación es debilitada a la forma $M \boxplus X \leq N$, entonces siempre se dispondrá de un conjunto de soluciones a la misma, de las cuales $X = N\widehat{\boxminus}M$ es la mayor de ellas [135]. Consecuentemente, siguiendo el mismo planteamiento, se puede definir un operador optimista de diferencia entre dos multiconjuntos difusos A y B como se indica en la Ecuación 2.41.

$$A\widehat{-}B = \bigcup\{S|(A \cap B) \uplus \subseteq A\} \tag{2.41}$$

Obsérvese que la expresión anterior no hace otra cosa que formalizar el concepto de que $A\widehat{-}B$ es el mayor multiconjunto difuso que se puede *añadir* (unir aditivamente) a $A \cap B$ para que el resultado esté contenido en A. Como se puede observar, esta definición no es contraintuitiva y es compatible con la definición usual del operador de diferencia.

Según lo anterior, definiremos formalmente el operador optimista de diferencia de multiconjuntos difusos como se indica a continuación.

Definición 2.19 (Diferencia optimista de multiconjuntos difusos)**.** *Sea U un conjunto. Sean A y B dos multiconjuntos difusos definidos sobre el universo U.*

La diferencia optimista de A y B, notada como $A\widehat{-}B$, es un multiconjunto difuso cuya función de conteo está definida como se indica en la Ecuación 2.42.

$$\Omega_{(A\widehat{-}B)}(u) = \text{máx}\left(\Omega_A(u)\widehat{\boxminus}\Omega_B(u), \{1.0/0\}_c\right) \qquad \forall u \in U \tag{2.42}$$

Dado que en el presente trabajo la diferencia de multiconjuntos difusos y cardinalidades difusas se hará siempre empleando su variante optimista, éstas se notarán de la misma forma que los operadores ordinarios, usando respectivamente los símbolos $-$ y $\boxminus$, con el objeto de simplificar la notación.

Finalmente, se ha de notar que, con el objetivo de solventar las dificultades de aplicación del operador de substracción extendido para cardinalidades difusas, Bosc y Rocacher han propuesto, con posterioridad a la propuesta descrita anteriormente, una caracterización alternativa para las cardinalidades de los multiconjuntos difusos [32]. Esta nueva propuesta emplea como caracterización los *enteros relativos difusos*, cardinalidades difusas definidas sobre $\mathbb{Z}$, con las que se puede obtener siempre un resultado exacto al aplicar el operador de diferencia. Desafortunadamente, la consecución de diferencias exactas entre multiconjuntos difusos mediante la caracterización de las cardinalidades difusas en $\mathbb{Z}_f$ (símbolo que representa el conjunto de los enteros relativos difusos) significa sacrificar la simplicidad de la caracterización en $\mathbb{N}_f$, lo que dificulta la interpretación de cardinalidades por parte de los usuarios.

División de multiconjuntos difusos.

Como los multiconjuntos contienen elementos con distinto número de ocurrencias, es posible derivar del operador de inclusión un nuevo operador que indique, no sólo si un multiconjunto B está incluido en otro multiconjunto A, si no *cuantas veces* B puede ser incluido en A. Este planteamiento da lugar al *operador de división.*

Este operador se define para el caso de los multiconjuntos *crisp* como se indica a continuación.

Definición 2.20 (División de multiconjuntos)**.** *Sea U un conjunto. Sean A y B dos multiconjuntos definidos sobre el universo U.*

La división de A por B, notada como $A \div B$, es un número entero que se determina como se indica en la Ecuación 2.43.

$$A \div B = \inf_{u \in U} \{\omega_A(u)/\omega_B(u)\} \tag{2.43}$$

Como en el caso de otros operadores, la anterior definición puede ser adoptada en el marco de los multiconjuntos difusos substituyendo los operadores aritméticos por operadores extendidos. Desafortunadamente, como ocurre con el caso de la substracción extendida, el operador extendido de división no siempre proporciona un resultado cuando es aplicado sobre dos cardinalidades difusas cualesquiera. En definitiva, el problema es que la ecuación $M \boxtimes X = N$ no siempre tiene una solución cuando N y M son números difusos. No obstante, al igual que el citado caso del operador extendido de substracción, es posible sustituir el operador extendido de división por una versión *optimista* que devuelva el mayor número difuso X que, aun no siendo la solución a la ecuación anterior, satisfaga su versión débil $M \boxtimes X \leq N$.

El operador optimista de división se construye en base al operador de diferencia optimista como una iteración del mismo [127]. Formalmente, la división optimista de dos números difusos está definida como se indica a continuación.

Definición 2.21 (División optimista de cardinalidades difusas)**.** *Sean A y B dos cardinalidades difusas.*

La división optimista de A por B, notada como $A \widehat{\oslash} B$, es una cardinalidad difusa definida según la función característica indicada en la Ecuación 2.44.

$$\mu_{(A \widehat{\oslash} B)}(n) = \sup_{\alpha \in [0,1]} \{\alpha | \{1.0/0, \alpha/n\}_c \boxtimes B \leq A\} \qquad \forall n \in \mathbb{N} \tag{2.44}$$

La sustitución en la Definición 2.20 del operador extendido de división por su versión optimista da lugar al *operador de división optimista de multiconjuntos difusos.* Dado que en el presente trabajo se hará uso exclusivamente de las versiones optimista del operador de división de multiconjuntos difusos y de cardinalidades difusas, con objeto de simplificar la notación, se emplearán los mismos símbolos de las versiones ordinarias para identificar los mismos.

Finalmente, como en el caso del operador extendido de diferencia, se ha de destacar que Rocacher y Bosc han propuesto, con posterioridad a la propuesta descrita anteriormente, una nueva caracterización de las cardinalidades difusas que permiten aplicar el operador extendido de división y obtener resultados exactos [127]. Más concretamente, se consigue que el citado operador sea complementario al operador extendido de producto para cualquier caso. No obstante,

de la misma forma que ocurre con el mencionado precedente, esta nueva caracterización supone sacrificar la simplicidad de la caracterización en $\mathbb{N}_f$. Para obtener los beneficios de la propuesta, resulta necesario caracterizar las cardinalidades difusas empleando *números racionales difusos* (elementos del conjunto $\mathbb{Q}_f$). Esto dificulta, más aún que la caracterización en $\mathbb{Z}_f$, la interpretación de cardinalidades por parte de usuarios.

2.5. Conclusiones

En el presente capítulo se han introducido los conceptos fundamentales en que se basará la presente memoria. En éste, se han proporcionado unas nociones básicas sobre bases de datos. A continuación, se han introducido la teoría de conjuntos difusos y la lógica difusa como herramientas que permitirán el tratamiento de información imperfecta. Finalmente, el capítulo aporta una introducción a los multiconjuntos difusos como estructuras matemáticas que permiten dotar a los datos una dimensión cuantitativa y cualitativa.

Capítulo 3

Información imperfecta en bases de datos

3.1. Introducción

Este capítulo introducirá los precedentes existentes en el área del tratamiento y representación de información imprecisa en bases de datos.

Para ello, se enumerarán las propuestas más importantes, agrupándolas en función de los paradigmas que emplean para la representación de información imprecisa. A continuación, se examinarán los diferentes lenguajes de consulta difusa que pueden ser empleados en las bases relacionales difusas, el grupo más importante de bases de datos de información imprecisas. Finalmente, se estudiarán las últimas propuestas en este campo, las bases de datos orientadas a objetos difusos.

3.2. Información imperfecta

El problema de la representación y tratamiento de información imprecisa ha sido ampliamente estudiado a lo largo de las últimas décadas. La primera cuestión a tratar en ésta sección será la descripción del concepto *información imperfecta*.

En la anterior sección, concretamente en la subsección 2.2.1, se propuso una definición para el término "información", tratado en esta memoria de manera amplia asimilándolo al término "dato", por lo que queda describir qué añade la calificación de "imperfecta". El termino "imperfecta", referido a la información, engloba varios significados que es interesante distinguir [31]:

- Incompletitud: La información que tenemos es incompleta, es decir, no tenemos toda la información disponible.
- Incertidumbre: No se conoce si la información es cierta o no.
- Desconocimiento: Se desconoce la información, no tenemos ninguna información
- Inaplicabilidad: La información no es aplicable a una determinada entidad.

Además de lo anterior, hay que considerar que, en ocasiones, dichos significados no son disyuntivos, por lo que para una determinada información se puede dar una combinación de los mismos.

Como hemos visto anteriormente, la teoría de conjuntos difusos y la lógica difusa han demostrado ser de gran utilidad para proporcionar dicha representación y tratamiento de información imperfecta. Con estas herramientas se trata de manera satisfactoria los distintos tipos de imperfección que pueden aparecer en los datos del mundo real.

3.3. Información imperfecta en bases de datos

La mayoría de las propuestas para la representación y tratamiento de información imperfecta en bases de datos emplean la lógica difusa para dicho propósito. No obstante, a continuación estudiaremos algunos modelos iniciales que no la emplearon.

La aplicación de la teoría de conjuntos difusos y de la lógica difusa para la representación y tratamiento de la información ha sido ampliamente estudiada sobre el modelo relacional de bases de datos, y recientemente sobre modelos de bases de datos orientados objetos, como se describirá más adelante.

Existen otros modelos para el tratamiento de la imprecisión en bases de datos que no emplean el paradigma difuso, pero éstos no han gozado de demasiada aceptación. Entre ellos están los basados en los conjuntos rugosos (*rough*) [116, 117] introducidos por Pawlak en 1982.

3.4. Aproximaciones sin emplear lógica difusa

En la presente sección analizaremos las aproximaciones que han sido propuestas para incorporar capacidades de tratamiento de la imprecisión en las bases de datos.

3.4.1. Valores nulos

La primera propuesta para dotar a una base de datos relacional con capacidad para la representación de información imperfecta fue realizada por E.F. Codd [50]. Ésta, tras su propuesta inicial, fue evolucionando en trabajos posteriores [51, 52, 53].

La citada propuesta consiste en la introducción de un nuevo valor, denominado NULL, en todo dominio disponible en la base de datos. Este valor especial se empleaba para indicar que no se conocía el valor que tomaba el atributo en dicha tupla, por lo que era posible que el valor de dicho atributo fuese cualquiera perteneciente al dominio asociado al citado atributo. Cuando se aplicaban comparadores sobre el valor nulo, la comparación no podía ser evaluada como *Verdadero* o *Falso*, ya que no se conocía el verdadero valor que toma el atributo. Como respuesta a las comparaciones se devolvía el valor *Quizás* (*Maybe* en el original) que es propio de las lógicas trivaluadas.

Con posterioridad, dicha propuesta evolucionó incluyendo la distinción entre valores nulos empleando una marca. Se propuso para este fin las dos marcas siguientes:

- A-marca: Indica que el valor nulo representa un valor desconocido pero aplicable.
- I-marca: Un valor nulo con esta marca indica que el valor no es aplicable.

3.4.2. Valores por defecto

Esta propuesta, realizada por Date [58], surge de la reflexión del autor sobre los valores nulos. Según el autor, éstos no habían sido bien tratados en el modelo relacional por lo que, según su opinión, no debían ser incorporados en el mismo hasta que se hubiese revisado correctamente la idea. Para suplir la funcionalidad que proporcionaban dichos valores nulos, propuso que cada dominio tuviese designado un *valor por defecto*. En caso de desconocimiento del valor de un atributo, se emplearía de forma automática el *valor por defecto* del dominio asociado.

3.4.3. Rangos de valores

El empleo de *rangos de valores*, como método para la representación de información imprecisa, fue propuesto por Grant [75]. La propuesta consistió en una extensión del modelo relacional de tal forma que los atributos pudiesen almacenar valores intervalares, junto con la redefinición de los operadores relacionales de manera que su evaluación pudiese resultar en los valores *Verdadero*, *Falso* y *Quizás*.

3.4.4. Bases de datos estadísticas

Este tipo de bases de datos fueron propuestas por Wong [152]. En ellas se trata la incertidumbre mediante la inferencia estadística. Con este enfoque, la consulta se plantea como un experimento estadístico en el que la información es incompleta y se centra en calcular la respuesta a dicha consulta caracterizando ésta como el conjunto de tuplas que minimiza los errores estadísticos.

3.4.5. Bases de datos probabilísticas

La Bases de Datos Probabilísticas fueron introducidas por Barbara et al. [7]. La propuesta consiste en el desarrollo de bases de datos en las que se asocian probabilidades a los valores de los atributos, de tal forma que se trata a cada a atributo como una distribución de probabilidad discreta.

3.5. Bases de datos que emplean lógica difusa

A continuación, realizaremos una breve descripción de los modelos de bases de datos que emplean de alguna forma lógica difusa como herramienta para el tratamiento de la imprecisión.

3.5.1. Modelo básico de base de datos difusa

Este modelo extiende el modelo relacional clásico mediante la incorporación a cada tupla de un grado, normalmente situado en el intervalo $[0, 1]$.

Dicho grado ha sido empleado, en diferentes propuestas, para los siguientes fines:

- Para indicar el *grado de pertenencia* de la tupla a la relación [73, 113]. Este uso ha sido el más extendido.
- Para indicar el *nivel de fuerza de dependencia* que existe entre dos atributos de la relación, representando de esta manera la relación que existe entre ellos [5].
- Para indicar el grado de cumpliendo de una condición.
- Para indicar el grado de importancia de la tupla en la relación [25].

El principal inconveniente de este modelo es que cada tupla asume su carácter difuso de forma global, por lo que no permite la representación de información difusa como valores en los atributos.

3.5.2. Modelo de Buckles y Petri

Este modelo, presentado inicialmente en [35] y desarrollado posteriormente en [34, 36], se basa en el empleo del concepto de *relación de similitud* [164].

Según la propuesta, cada par de elementos de cada dominio empleado en la base de datos ha de tener asociado un grado de similitud. Cuando dicho grado de similitud vale 1, indica que el par de elementos es completamente igual, y cuando dicho grado vale 0, se indica que los elementos del par son totalmente distintos.

El modelo permite representar valores conjuntivos, con una semántica disyuntiva. Los valores permitidos son:

1. Conjuntos finitos escalares, como puede ser el conjunto de los tipos de inmuebles $\{casa, piso, duplex\}$.
2. Conjuntos finitos de números, como por ejemplo el conjunto $\{1, 5, 25, 125\}$.
3. Conjuntos de números difusos, como el conjunto $\{\text{barato}, \text{módico}, \text{caro}\}$.

En las consultas se puede establecer un umbral para el grado de similitud entre elementos, por encima del cual los pares de elementos se consideran iguales, y por debajo, distintos.

3.5.3. Modelo de Prade Testemale

Este modelo de base de datos relacional difusa, descrito en mayor detalle en [122, 121, 124, 123], se basa en la *teoría de la posibilidad* [169] (Reeditado en 1999 en [170]) para la representación de datos imprecisos.

Los valores imprecisos se representan como una distribución de posibilidad. En concreto, el valor para un atributo A de una entidad x queda representado por una distribución de posibilidad $\Pi_{A(x)}$ definida según una función característica $\pi_{A(x)} : D \cup \{e\} \rightarrow [0, 1]$, siendo D el dominio asociado al atributo A, y e un elemento especial que indicar la inaplicabilidad del atributo A a la entidad x.

Este modelo exige que las distribuciones de posibilidad empleadas para la representación de datos imprecisos estén *normalizadas*, lo que significa que al menos debe existir un valor $d \in D \cup \{e\}$ para el cual se cumpla $\pi_{A(x)}(d) = 1$.

Como en todo modelo posibilístico, hay que tener en cuenta que para un valor $d \in D \cup \{e\}$, el hecho de que $\pi_{A(x)}(d) = 1$ sólo indica que dicho valor es *completamente* posible, y no que el valor sea *cierto*. De hecho, dicho valor d solo se podrá considerar *cierto* cuando sea el único valor *posible*.

3.5.4. Modelo de Umano-Fukami

Este modelo [69, 147, 145, 146] es uno de los primeros modelos de Bases de Datos Relacionales Difusas. Al igual que el modelo anterior, está basado en la *teoría de la posibilidad*, empleando ésta para la representación de información imprecisa. Dicho modelo es denominado por los autores como *possibility-distribution-fuzzy-relational*.

Como diferencia destacable, este modelo incluye una mejora importante, con respecto al modelo anterior, en lo que se refiere a la representación y tratamiento de *información desconocida o no aplicable*. Para la representación de este tipo de información, se introducen tres valores especiales:

- Desconocido: Este valor (UNKNOWN en el original) indica que no se conoce el valor del atributo para una determinada entidad, pero que dicho atributo es aplicable. Éste se describe mediante una distribución de posibilidad cuya función característica se indica en la Ecuación 3.1.

$$\pi_{\text{Unknown}}(d) = 1 \qquad \forall d \in D \tag{3.1}$$

- No aplicable: El valor (UNDEFINED en el original) indica que el atributo no es aplicable para una determinada entidad. El valor queda descrito por una distribución de posibilidad cuya función característica es de la forma que se indica en la Ecuación 3.2.

$$\pi_{\text{Undefined}}(d) = 0 \qquad \forall d \in D \tag{3.2}$$

- Nulo: Este valor (NULL en el original) representa el desconocimiento total del valor que toma el atributo, ni siquiera se conoce si el atributo es aplicable o no. La distribución de posibilidad que representa este valor está definida como se indica en la Ecuación 3.3.

$$\Pi_{\text{Null}} = \{1/\text{Unknown}, 1/\text{Undefined}\} \tag{3.3}$$

Además de la representación de información imprecisa en los valores de los atributos, este modelo permite la asociación de una distribución de posibilidad, dentro del intervalo $[0, 1]$, a cada tupla que indica el grado de pertenencia de dicha tupla a la relación.

Finalmente se ha destacar, que este modelo incluye la definición extendida de los operadores básicos y no básicos del álgebra relacional, que permiten operar con las relaciones de este modelo, aplicando para ello el principio de extensión.

3.5.5. Modelo de Zemankova y Kaendel

Este modelo, descrito en [171, 172], opta por una representación de la información imperfecta igual, que la de los modelos posibilísticos. Adicionalmente, desarrolla un lenguaje de manipulación de datos que analiza las relaciones entre posibilidad y certeza.

3.5.6. Modelo generalizado para bases de datos difusas

Este modelo, denominado originalmente *Generalized Model of Fuzzy Relational Databases* (GEFRED) y propuesto por Medina et al. [105], surge de la iniciativa de la integración de los modelos anteriores. El objetivo de la propuesta es ofrecer un modelo que incluya los aspectos más interesantes de las propuestas previas para ofrecer una solución completa al problema de la representación y consulta de información imprecisa inscrita en un modelo relacional.

Para solucionar el problema de la representación de información imperfecta, el modelo define el concepto de *dominio difuso generalizado*. Dicho dominio sustituye y extiende el dominio clásico de los atributos, de forma tal que se permite la representación de distribuciones de posibilidad sobre el dominio básico y el empleo de valores especiales (NULL,UNKNOWN,UNDEFINED) para indicar tanto el desconocimiento como la inaplicabilidad del valor de los atributos.

Mediante el empleo de dominios difusos generalizados, en la base de datos se podrán representar los siguientes tipos de datos:

1. Escalares. Por ejemplo, para el dominio $Rendimiento = \{bajo, medio, alto\}$, un valor *alto* se representa empleado la distribución de posibilidad $\{1/alto\}$.

2. Números. Por ejemplo, para un campo *Altura*, un valor de 1.73 se puede representar con la distribución de posibilidad $\{1/1.73\}$.

3. Conjuntos de asignaciones escalares posibles. Por ejemplo, para el anterior dominio *Rendimiento*, se puede indicar que el rendimiento puede ser tanto *medio* como *alto* empleando la siguiente distribución de posibilidad $\{1/medio, 1/alto\}$.

4. Conjuntos de asignaciones numéricas posibles. Por ejemplo, para el caso anterior del campo *Altura*, podemos indicar que es posible que ésta sea 1.70, 1.80 y 1.90 mediante la siguiente distribución de posibilidad $\{1/1.70, 1/1.80, 1/1.90\}$.

5. Distribuciones de posibilidad construidas sobre un dominio escalar. Por ejemplo, se puede definir sobre el dominio *Rendimiento* la siguiente distribución de posibilidad $\{0.3/bajo, 0.7/medio, 0.1/alto\}$.

6. Distribuciones de posibilidad construidas sobre un dominio numérico. Usando de nuevo el atributo *Altura*, se podría definir la distribución de posibilidad $\{0.1/1.70, 0.9/1.75, 1/1.80, 0.2/1.85\}$.

7. Grados (un número real en el intervalo $[0, 1]$) que indican el *grado de acoplamiento*. Por ejemplo, para un atributo *Calidad* se puede indicar $Calidad = 0.9$ indicando que la entidad se ajusta en gran medida al concepto de calidad.

8. Valores UNKNOWN (desconocido), que indican que no se conoce el valor concreto para el atributo, aunque éste es aplicable. Este valor se representará mediante la distribución de posibilidad $\{1/u|u \in U\}$, siendo U el universo de discurso.

9. Valores UNDEFINED (no aplicable), que indican que el atributo no es aplicable a la entidad. Este valor se representa mediante la distribución de posibilidad $\{0/u|u \in U\}$, siendo U el universo de discurso.

10. Valores NULL (nulo), que indican que se desconoce si el valor es aplicable o no, y además se desconoce que valor podría tomar. Se empleará la distribución de posibilidad $\{1/UNDEFINED, 1/UNKNOWN\}$ para representar este valor.

Basándose en el concepto de *dominio difuso generalizado*, el modelo define las *relaciones difusas generalizadas*. Éstas están basadas en la relaciones clásicas y se construyen empleando tanto dominios clásicos como dominios difusos generalizados. Además, en ellas se añade a cada atributo un *grado de compatibilidad* que es empleado para describir el grado de ajuste del atributo a las condiciones difusas impuestas en las consultas.

Para la manipulación de las citadas *relaciones difusas generalizadas*, el modelo define el concepto de *comparador difuso generalizado* para permitir establecer condiciones difusas sobre los valores de los atributos, y la *selección difusa generalizada* que permite realizar una selección sobre una *relación difusa generalizada* aplicando condiciones difusas definidas en forma de *comparador difuso generalizado*.

Finalmente, se ha de destacar que el modelo propone una extensión difusa del álgebra relacional.

3.5.7. Bases de datos difusas basadas en evoluciones del modelo relacional

Todos los trabajos descritos anteriormente se basan en el modelo relacional para la formulación de las propuestas. Como se indicó en el Capítulo 2, antes del salto del modelo relacional a las bases de datos orientadas a objetos se propusieron ciertas evoluciones del relacional con el objeto se solventar algunas de sus carencias en nuevas aplicaciones. De entre ellas, destacan las *relaciones anidadas* y las *multirrelaciones*.

Dado el breve lapso de tiempo en que la investigación en bases de datos difusas pasó de emplear como base el modelo relacional a las bases de datos orientadas a objetos, no se encuentra en la bibliografía muchos trabajos que se basen en las citadas evoluciones del modelo relacional. No obstante, la revitalización de estas propuestas tras la incorporación de sus características en SQL:1999 atrajo de nuevo la atención de los investigadores en bases de datos difusas sobre éstas. De entre ellas, podemos destacar el trabajo de Ma en relación al modelado de información difusa y su posterior mapeo en un modelo difuso de relaciones anidadas propuesto por el mismo autor [97], el trabajo pionero de Kim et al. sobre multirrelaciones difusas [83] y los trabajos de Rocacher y Bosc sobre el empleo de multiconjuntos difusos en consultas flexibles [126, 127].

3.6. Lenguajes de consulta difusa

La recuperación de datos desde una base datos es un aspecto fundamental de las mismas. El usuario debe poder obtener de la base de datos aquellos datos que necesite en cada momento, así como aplicar un tratamiento particular sobre dichos datos. En las consultas tradicionales sobre bases de datos, el usuario establece unas determinadas condiciones de consulta que le permiten especificar qué datos desea recuperar. La base de datos devolverá aquellas tuplas que satisfagan completamente dichas condiciones, constituyendo éstas el resultado de la consulta planteada por el usuario.

Una alternativa a la consulta clásica es lo que se conoce como *consulta flexible*, siendo ésta una consulta en la que las condiciones impuestas por el usuario pueden ser satisfechas parcialmente. Las consultas flexibles pueden ser aplicadas tanto en consultas sobre bases de datos clásicas (*crisp*), como sobre bases de datos difusas, proporcionando, frente a la consultas clásicas, la siguientes ventajas:

- Proveen un mayor número de respuestas a las consultas, en forma de respuestas aproximadas, cuando las consultas clásicas son demasiado restrictivas y no proporcionan suficiente número de resultados.
- Categorizar los resultados en función de su compatibilidad con las condiciones impuestas en la consulta. Cada resultado de la consulta tendrá un grado de compatibilidad asociado. Éste grado permitirá, por ejemplo, ordenar los resultados por relevancia o limitar la respuesta de la consulta a los n mejores resultados.

A continuación, describiremos brevemente las propuestas más significativas sobre lenguajes de consulta difusa.

3.6.1. Lenguaje de consulta difuso de Tahami

Uno de los primeros trabajos que abordaron las consultas flexibles mediante el uso de conjuntos difusos corresponde a Tahami [139], aunque dicho trabajo se enfocó sólo sobre la aplicación de este tipo de consultas sobre bases de datos clásicas. Aplicando un predicado difuso como condición para consultar relaciones clásicas, obtiene una relación en la que cada tupla posee un grado asociado que indica el nivel de satisfacción del predicado aplicado.

3.6.2. Lenguaje SQLf

Este lenguaje, propuesto por Bosc y Pivert [30] en 1995, representa una síntesis de las características y funcionalidades sugeridas en otras propuestas anteriores sobre consulta flexible aplicada a bases de datos clásicas, como son [27, 94, 115, 139, 153]. Al igual que en los citados trabajos, SQLf sólo considera la sentencia SELECT, empleada para la realización de consultas, no abordando el resto de sentencias que conforman los sublenguajes DDL y DML de SQL. El formato de las sentencias para consultas de dicho lenguaje es el siguiente:

```
SELECT  [N|T|N,T] <lista_de_selección>
FROM    <lista_de_tablas>
WHERE   <condición_difusa>
```

En la anterior descripción de la sintaxis, `<lista_de_selección>` indica la lista de atributos que se desean recuperar, `<lista_de_tablas>` indica la lista de tablas que serán reunidas para la consulta y `<condición_difusa>` la condición difusa que deben satisfacer las tuplas para que formen parte del resultado de la consulta. Los elementos opcionales `N` y `T`, que pueden ser aplicados por separado o de manera conjunta, permiten al usuario especificar que desea recuperar sólo los `N` mejores resultados y/o, según sea el caso, aquellas tuplas cuyo grado de cumplimiento de la condición difusa impuesta sea superior al umbral `T`.

3.6.3. Lenguaje FSQL

El lenguaje Fuzzy SQL (FSQL) (Fuzzy SQL), propuesto por Medina y Galindo [70, 71], está diseñado para permitir la definición de consultas sobre el modelo GEFRED de Medina, descrito en la sección anterior. Antes de entrar a tratar la sintaxis del lenguaje, estudiaremos la propuesta de implementación del modelo GEFRED, en la que se basa dicho modelo de consulta.

3.6.3.1. Aproximación computacional a GEFRED

La propuesta de implementación del modelo GEFRED se conoce con el nombre de FIRST (*Fuzzy Interface for Relational Systems* ó, en castellano, Interfaz Difusa para Sistemas Relacionales), y fue introducida en [70, 103]. Dicha propuesta de implementación introduce los siguientes tipos de datos difusos:

- Atributos difusos de tipo 1: Este tipo de atributo permite almacenar valores *crisp*. El empleo de este tipo de atributos, en lugar de los tipos incluidos en el SGBD anfitrión, permitirá que los atributos puedan participar en condiciones difusas, lo que permite compararlos con las distribuciones de posibilidad y las etiquetas lingüísticas que estén definidas sobre su dominio.

- Atributos difusos de tipo 2: Este tipo de atributo tomará valores en un dominio de números difusos. Dichos números difusos se representarán por distribuciones de posibilidad definidas sobre dominios ordenados continuos o discretos. Los posibles valores que podrá albergar este tipo son los siguientes:

  - Distribución de posibilidad trapezoidal: Este valor se describirá mediante la notación abreviada $[\alpha, \beta, \gamma, \delta]$ empleada para definir números difusos.
  - Etiqueta lingüística: Dichas etiquetas son referencias a conceptos empleados en el lenguaje natural y caracterizados por una distribución de posibilidad.
  - Valor aproximado: Representa el concepto difuso *aproximadamente* n, siendo n un valor de dominio subyacente. La distribución de posibilidad trapezoidal con la que se caracteriza este tipo de valor es $[n - margen, n, n, n + margen]$, siendo $margen$ un valor establecido de forma global para el dominio. Dicho margen identifica hasta donde pueden ser considerados *aproximados* los valores del dominio.

  - Intervalo de posibilidad: Este valor es una distribución de posibilidad que representa un rango de valores totalmente posibles. Un intervalo de posibilidad en el rango $[\alpha, \beta]$, se caracteriza por una distribución de posibilidad trapezoidal $[\alpha, \alpha, \beta, \beta]$.

- Atributos difusos de tipo 3: Este tipo de atributos están definidos sobre dominios discretos no ordenados, en los que se define una relación de similitud entre los valores del dominio para permitir comparaciones de proximidad entre los elementos. Los posibles valores que podrá albergar este tipo de atributo son los siguientes:

  - Valor escalar: Es el caso particular en el que en la distribución de posibilidad que puede albergar el tipo sólo existe un elemento y éste tiene un grado de posibilidad igual a 1. Es decir, la distribución de posibilidad que quedará representada es $\{1/d\}$ donde d es el valor escalar que se desea almacenar.
  - Distribución de posibilidad definida sobre un dominio escalar: Este tipo permite almacenar una distribución de posibilidad de la forma $\{p_1/d_1, p_2/d_2, \ldots, p_n/d_n\}$, donde p_i es el grado de posibilidad asociado al escalar d_i.

Adicionalmente, se definen varios valores especiales (heredados del modelo de Umano-Fukami) empleados en GEFRED. Estos valores son los siguientes:

- UNKNOWN: Representa el valor desconocido. Asigna un grado de posibilidad 1 a todos los elementos del dominio.
- UNDEFINED: Representa el hecho de que el atributo no es aplicable. Asigna un grado de posibilidad 0 a todos los elementos del dominio.
- NULL: Representa el desconocimiento absoluto. No se conoce si el valor es aplicable o no y, evidentemente, no se conoce el valor que toma. Se asigna el grado de posibilidad 1 a los valores UNKNOWN y UNDEFINED.

3.6.3.2. Sintaxis de FSQL

Esta propuesta parte de la estructura básica de la sentencia SELECT del lenguaje de consulta relacional SQL, que tiene la siguiente sintaxis:

```
SELECT <lista_de_selección>
FROM <lista_de_tablas>
WHERE <condición>;
```

Partiendo de dicha estructura, la propuesta introduce modificaciones para permitir la realización de consultas difusas sobre tipos difusos. Estas modificaciones permiten manejar los siguientes elementos en las consultas:

- Etiquetas lingüísticas: Se pueden introducir en la sentencia etiquetas lingüísticas, con el objeto de hacer uso de ellas en las condiciones que se definan en la consulta. Para su empleo irán precedidas del símbolo $, de la forma $*etiqueta*.

Constante	Sintaxis
Valor desconocido	UNKNOWN
Valor inaplicable	UNDEFINED
Valor nulo	NULL
Distribución de posibilidad trapezoidal	`$[` $\alpha, \beta, \gamma, \delta$ `]`
Etiqueta lingüística	`$etiqueta`
Intervalo	`[` n, m `]`
Valor aproximado	`#`n

Tabla 3.1: Constantes difusas de FSQL

- Comparadores difusos: Este lenguaje introduce una amplia gama de comparadores difusos, como pueden ser, por ejemplo, la igualdad difusa, mayor o igual entre números difusos, etcétera. Estos comparadores permitirán la construcción de condiciones difusas sobre los valores difusos de la base de datos. Cada comparador está disponible tanto con una semántica de posibilidad como de necesidad, de tal forma que el usuario pueda seleccionar la variante adecuada.

- Umbral de cumplimiento: El lenguaje introduce el elemento `THOLD<` t `>`, con $t \in [0, 1]$, que permite establecer un umbral mínimo de cumplimiento t para las condiciones difusas incluidas en la cláusula WHERE.

- Grado de compatibilidad: El lenguaje añade el operador `CDEG(<atributo>)`, que permite conocer el grado de cumplimiento de todas las condiciones en las que se ve implicado el atributo `<atributo>`. Cuando se usa el carácter `*` como parámetro de dicho operador, éste devuelve el grado de cumplimiento global de cada tupla para las condiciones difusas establecidas en la cláusula WHERE.

- Carácter comodín difuso: Siguiendo la filosofía del carácter comodín de SQL, el carácter `*`, que incluye en la selección todos los atributos de las tablas involucradas en la consulta, el lenguaje FSQL introduce el carácter comodín `%`. Este nuevo carácter, además de incluir en la selección todas las columnas de las tablas participantes en la consulta, incluye los grados de compatibilidad de todos los atributos difusos revelantes, entendiendo por relevante aquel atributo que aparece en alguna condición establecida en la consulta.

- Constantes difusas: El lenguaje incluye 7 tipos de constantes que permitirán al usuario definir valores difusos de forma sencilla. Dichas constantes se muestran en la tabla 3.1.

- Condición `IS`: La condición `IS` de SQL, permite comprobar si un valor es nulo. El lenguaje FSQL amplía esta condición para permitir distinguir si el valor es desconocido, inaplicable o nulo. La sintaxis de la condición queda como sigue:

```
<atributo_difuso> IS [NOT] (UNKNOWN | UNDEFINED | NULL)
```

- Cuantificadores difusos: El lenguaje FSQL permite el empleo de cuantificadores difusos en las consultas, ya sean absolutos o relativos.

Un ejemplo de consulta difusa expresada usando el lenguaje FSQL podría ser la siguiente:

```
SELECT CDEG(*),%
FROM Inmuebles
WHERE
    Precio FLEQ $[100000,150000,200000,250000] AND
    TIPO FEQ $Apartamento THOLD 0.6 AND
    Superficie IS NOT UNKNOWN AND
    Habitaciones FEQ #4;
```

3.6.3.3. Arquitectura del servidor de consultas FSQL

El Servidor de Consultas FSQL, llamado FSQL Server, fue implementado inicialmente por Medina [103] y mejorado por Galindo [70].

El proyecto de implementación fue ideado de forma que dicho servidor de consultas se apoya sobre un SGBD *crisp* ya existente. Este enfoque presenta ciertas ventajas y desventajas. Como ventaja destaca, principalmente, la facilidad para realizar la implementación, ya que no es necesario programar desde cero un SGBD completo. Con ello, obtendremos las ventajas del SGBD anfitrión (como, por ejemplo, seguridad, eficiencia, etc.) sin que deban considerar estas importantes características en el proceso de desarrollo. La desventaja fundamental de esta aproximación es que el servidor será menos óptimo que si éste se hubiese programado a bajo nivel.

Los componentes fundamentales de la arquitectura del servidor son:

- Datos: Los datos almacenados y gestionados por el SGBD anfitrión se componen de los datos propios de los usuarios del servidor de consultas FSQL, así como lo datos de la *base de metaconocimiento Difuso* (FMB, Fuzzy Meta-knowledge Base). La FMB almacena, en formato relacional, toda la información necesaria para el funcionamiento del servidor FSQL. Esta información está compuesta por la lista de todas las columnas de tipos difusos de la base de datos, así como información acerca de las mismas (como, por ejemplo, las etiquetas lingüísticas definidas sobre éstas).

- Servidor FSQL: Es un componente software incrustado en el SGBD que oculta el procesamiento de las consultas difusas al usuario. Está programado íntegramente en PL/SQL (la extensión procedural del lenguaje SQL, propietaria de Oracle) y está compuesto por tres grandes grupos de funciones:

  - Funciones de traducción: Este grupo está compuesto por una función principal, llamada función de traducción, y un conjunto de funciones auxiliares que ayudan en su cometido a la primera. La función de traducción realiza el análisis léxico, sintáctico y semántico de las consultas FSQL de los usuarios.

    Después de comprobar que no hay errores en la consulta FSQL, la función de traducción traduce la misma a una consulta SQL (que

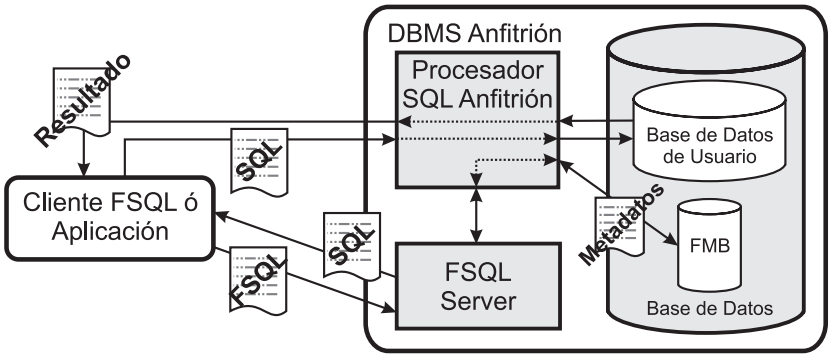

Figura 3.1: Arquitectura del servidor de consultas FSQL

contendrá llamadas a las funciones del siguiente grupo) que podrá ser procesada por el SGBD anfitrión para proporcionar la respuesta a la consulta difusa que envió el usuario.

- Funciones de representación: Las funciones de este grupo son empleadas para mostrar los atributos difusos, incluidos como columnas en las respuestas a las consultas, de forma legible por el usuario. De esta forma, se evita que el usuario deba interpretar el formato interno en que se representan dichos atributos.
- Funciones de comparación difusa: Este grupo de funciones se encarga de implementar las comparaciones entre atributos difusos, así como del cálculo de los grados de compatibilidad devueltos al usuario. Estos grados de compatibilidad son generados por el elemento `CDEG(<atributo_difuso>)` del lenguaje FSQL.

- Cliente FSQL: Este componente es un programa externo que hace de interfaz entre el usuario y el servidor FSQL, ocultando los detalles de llamada a las funciones que conforman dicho servidor.

La Figura 3.1 ilustra, de forma esquemática, cómo funciona el servidor FSQL. Según el esquema que se muestra en la figura, la secuencia que sigue el procesamiento de una consulta FSQL es la siguiente:

1. El cliente FSQL, o cualquier aplicación que desee obtener los resultados de una consulta FSQL, envía esta a la función de traducción de consultas de FSQL Server. Dicha función se encuentra incrustada en la base de datos, y por tanto, para realizar su invocación se emplearán los mecanismos de llamada a funciones almacenadas en dicho sistema anfitrión.

2. La función de traducción de FSQL Server realiza el análisis léxico, sintáctico y semántico de la consulta FSQL, apoyándose para ello en los metadatos de la FMB. Como resultado del proceso anterior, se obtiene una sentencia SQL que, empleando las funciones de representación y comparación difusa, devolverá el resultado de la consulta difusa que planteó el usuario.

3. El cliente FSQL, o la aplicación que hace uso de los servicios prestados por SQL Server, obtendrá la sentencia SQL resultado del proceso de traducción. Dicha sentencia se enviará, por parte del cliente o la aplicación, al procesador del consultas del SGBD anfitrión para su procesamiento.

4. El procesador de consultas del SGBD anfitrión recibe la consulta SQL, la procesa, y envía de vuelta el resultado de la misma al cliente FSQL.

3.6.3.4. Extensiones del modelo GEFRED y FSQL Server

El modelo GEFRED, y su implementación FSQL Server, han sido extendidos por varios autores con objeto de añadirles nuevas capacidades. Dichas extensiones son las siguientes:

- Blanco [15, 16] propone, en 2001, una extensión de GEFRED para dotarlo de capacidades deductivas. Para ello, introduce el concepto de Relación Intensiva Difusa y extiende la aproximación computacional de GEFRED (FIRST), obteniendo como resultado el modelo FREDDI, así como el lenguaje DFSQL (Deductive FSQL) para permitir manipular dichos tipos de relaciones y realizar deducciones sobre las mismas.

- Carrasco [37, 38] propone, en 2003, una extensión de GEFRED para añadir capacidades de *minería de datos* (Data mining). Dicha extensión añade nuevos tipos de datos, así como nuevas sentencias al lenguaje FSQL con el objeto de poder practicar técnicas de minería de datos sobre datos difusos.

3.7. Bases de datos orientadas a objetos difusos

La presente sección está dedicada al estudio de las bases de datos orientadas a objetos difusos. Para ello, estudiaremos qué razones llevan a la incorporación de este tipo de bases de datos al terreno difuso, y continuaremos estudiando a qué niveles se incorpora la vaguedad en dichas bases de datos. Finalmente, revisaremos brevemente los modelos de bases de datos, de éste tipo, mas significativos.

3.7.1. Motivos para la incorporación de borrosidad

Las bases de datos relacionales difusas dieron paso, en el terreno investigador, a las bases de datos orientadas a objetos difusos. Las razones por las que se lleva a cabo este cambio han sido, fundamentalmente, las mismas que las razones que llevaron al inicio de la investigación y desarrollo de bases de datos orientadas a objetos, en detrimento de las bases de datos relacionales. Estás razones se pueden reducir a una fundamental: la mayor capacidad de modelado de la realidad que posee el modelo orientado a objetos, lo que permite abordar nuevas aplicaciones complejas.

Las bases de datos orientadas a objetos difusos, poseen ciertas ventajas sobre las bases de datos orientadas a objetos clásicos. Dichas ventajas son las siguientes:

- Aumento de la capacidad de consulta: Las bases de datos orientadas a objetos ofrecen una mayor facilidad de consulta ya que permiten emplear condiciones difusas.

- Aumento de los problemas que pueden ser tratados: Al dotar a las bases de datos orientadas a objetos de capacidad para el almacenamiento y manipulación de información imperfecta, aumenta la cantidad de problemas que pueden ser tratados por las mismas. Así mismo, para los problemas que ya podían ser tratados por las bases de datos orientadas a objetos clásicas, la nueva orientación difusa puede aportar ventajas en forma de mayor flexibilidad y adecuación a los problemas.

3.7.2. Niveles de incorporación de la borrosidad

La imprecisión se incorporará al modelo orientado a objetos en diferentes niveles, coincidiendo éstos con los diferentes niveles de agrupación de información que ofrecen los modelos orientados a objetos. Dichos niveles son los siguientes:

- Nivel de atributos: Este primer nivel, permite la incorporación de datos difusos en las bases de datos. Para ello, se añaden nuevos dominios que ofrecen la posibilidad de almacenar información imperfecta como valores de los atributos de los objetos.

- Nivel de relaciones de instancia: La borrosidad en este nivel permite que un objeto pertenezca a una clase de manera incierta, es decir, con un cierto grado situado en el intervalo $[0, 1]$. En definitiva, lo que se permite a este nivel, es difuminar las relaciones de pertenencia a una clase por parte de los objetos.

- Nivel de relaciones de herencia: La incorporación de la borrosidad en este tercer nivel, permite suavizar las relaciones superclase-subclase existentes en los modelos orientados a objetos.

- Nivel de definición: Este nivel se centra en la definición del tipo de la clase y de los objetos. La borrosidad en este nivel, afecta a la estructura de dichas clases u objetos.

- Nivel de conducta: Como último nivel, se considera la borrosidad asociada a la conducta de los objetos. En este nivel, se ven afectados por la imprecisión los métodos que definen dicha conducta.

3.7.3. Modelos de bases de datos orientadas a objetos difusos

Como ya se indicó en anteriores secciones, los modelos de bases de datos orientadas a objetos tienen sus precursores en los diferentes modelos semánticos que fueron superando, en capacidad y facilidad de modelado, al modelo relacional.

En el caso de las bases de datos orientadas a objetos, sucede algo muy similar. Los precursores de dicho tipo de base de datos son los modelos semánticos avanzados que fueron incorporando la imprecisión, siendo entre ellos el más destacable el Modelo Entidad-Relación Difuso.

Durante dicho desarrollo, una serie de trabajos comienzan a estudiar la imprecisión en los datos y en las relaciones de los mismos:

- Ruspini [132] presenta los distintos tipos de información imprecisa que cualquier modelo de datos debe considerar, comenzando a introducir los atributos definidos de forma imperfecta, representados mediante términos lingüísticos y conjuntos difusos.

- Ziveli y Chen [175] comienzan a plantear los diferentes niveles en los que la información puede estar afectada por cierta imperfección. Entre otros se identifican el nivel conceptual, relaciones difusas entre entidades, conjuntos difusos de entidades, etcétera.

- Vanderberghe et al. [150] introducen nociones mas elaboradas de importantes conceptos como, por ejemplo, conjunto difuso de entidades, subclases difusas y categorías difusas.

Estos trabajos supusieron la gestación del tratamiento de la información imperfecta más allá del modelo relacional y abrieron la senda para la inclusión de la borrosidad en modelos orientados a objetos, desembocando dicha senda en el estudio de las bases de datos orientadas a objetos difusos.

A continuación, se estudiaran, de forma general, los modelos de bases de datos orientadas a objetos más relevantes que se encuentran la bibliografía.

3.7.3.1. Modelo de J.P. Rossaza et al.

Los autores presentan [128] una representación orientada a objetos cuya principal novedad es la de permitir la especificación de un *rango de valores permitidos* y un *rango de valores típicos* para los atributos que componen la estructura de una clase. Dichos rangos podrían ser definidos de forma difusa.

Haciendo uso de dichos rangos de valores, el modelo permite definir diferentes tipos de relaciones de inclusión, de forma graduada, entre clases.

De forma análoga, para los valores de los atributos en las instancias de las clases se introduce el *rango posible* y el *rango creíble*. Esto permite especificar de forma imprecisa los valores de los atributos de dichas instancias.

Considerando los cuatro tipos de rangos, el modelo permite graduar las relaciones instancia-clase, de forma tal, que es posible realizar un cálculo del grado en el que una determinada instancia podría ser clasificada como de una determinada clase.

Basándose en todo lo anterior, los autores proponen un nuevo mecanismo de herencia e instanciación.

Finalmente, los autores estudian los diferentes procedimientos necesarios para el tratamiento de los eventos que pueden ocurrir durante el mantenimiento y la actualización de la base de datos.

3.7.3.2. Modelo GBP de R. George et al.

El principal objetivo de este trabajo [72] es la representación de situaciones del mundo real dentro de un sistema gestor de bases de datos. Para ello, se emplea un paradigma orientado a objetos clásico en el que se introduce la distinción entre las nociones *dominio de un atributo* y *rango de un atributo*. Dichos conceptos se definen como:

Definición 3.1 (Dominio de un atributo)**.** *El* dominio de un atributo *se define como el conjunto de valores que un atributo puede tomar independientemente de la clase en la que éste sea empleado.*

Definición 3.2 (Rango de un atributo)**.** *Se define el* rango de un atributo *como el conjunto de valores permitidos para un atributo en una instancia de una determinada clase.*

El trabajo propone la representación de problemas mediante el empleo de una jerarquía difusa de clases. Dicha jerarquía difusa se distingue de la clásica ya que, en la primera, los conceptos que definen las clases tienen unos límites imprecisos, debido a la falta de precisión de los valores que toman los objetos que forman la clase. Los grados de pertenencia de los objetos a las clases se calcularán empleando relaciones de similitud, en sustitución de las relaciones de igualdad, definidas entre los valores de los dominios de los atributos.

3.7.3.3. Modelo FOOD de A. Yazici et al.

Este modelo [157, 162, 160] supone la continuación y mejora de las ideas propuestas en el modelo anterior, pudiéndose considerar éste una extensión del mismo. La imprecisión se soporta en este modelo a los niveles de atributo, de relaciones objeto-clase, de relaciones superclase-subclase, de relaciones clase-clase y de asociaciones entre clases.

La sustitución de las relaciones de igualdad por las de similitud, y el empleo de los conceptos de dominio y rango asociados a cada atributo, permiten el cálculo del grado de pertenencia de un determinado objeto a una clase. También, se define el método por el cual es posible el cálculo del grado de pertenencia de una determinada subclase a una superclase.

3.7.3.4. Modelo de G. Bordogna et al.

El modelo, presentado en [23, 24], está basado en una representación gráfica que permite modelar la presencia de información vaga en una base de datos orientada a objetos. Dicha representación gráfica resulta muy expresiva, ya que cada elemento capaz de representar información imperfecta tienen asociada una notación gráfica que permite la construcción del modelo conceptual de una forma muy explícita y clara.

Finalmente, la propuesta incluye el planteamiento de una implementación del modelo propuesto, que consiste básicamente en el almacenamiento y gestión del grafo que representa el modelo conceptual, en una base de datos orientada a objetos tradicional.

Se ha de destacar que el modelo se centra solamente en la representación de información imprecisa, y no proporciona ningún método para el cálculo del grado de relación entre los elementos que conforman el esquema, mientras que los modelos anteriores sí que proporcionan esta clase de métodos.

3.7.3.5. Modelo UFO de N. Van Gyseghem et al.

Ese modelo, presentado en [148, 149], se puede considerar el más completo y, por tanto, el más complejo de los modelos presentados hasta el momento en la presente sección. No propone métodos para el cálculo de los grados de

pertenencia de los diferentes elementos del modelo, pero considera la representación de información imperfecta en todas sus versiones y en todos los niveles del esquema.

La propuesta puede se divide en dos partes. La primera parte del modelo transforma cada elemento del modelo orientado a objetos que esté representado por un conjunto clásico por uno equivalente usando conjuntos difusos, siempre que esto proporcione alguna ventaja. Esta parte trata sobre los siguientes elementos:

- Modelado suave de propiedades difusas: La propuesta permite modelar de manera suave las clases, dividiendo el conjunto de propiedades de las clases entre *propiedades requeridas* y *propiedades opcionales*. Esta última clase de propiedades se corresponde con propiedades específicas de posibles subclases, que no están definidas dentro del esquema de la base de datos para una aplicación concreta. Las propiedades opcionales no son heredadas obligatoriamente por las subclases y, en caso de serlo, puede mantenerse, o no, su opcionalidad. La citada *opcionalidad* de las propiedades puede ser difuminada, permitiéndose que una determinada propiedad en una clase sea opcional en un determinado grado.

- Relaciones objeto-clase (clases difusas): Estas relaciones pueden ser graduadas, por lo que un conjunto difuso representa convenientemente el conjunto de objetos que pertenecen a una determinada clase.

- Relaciones superclase-subclase (herencia difusa): De igual forma, las relaciones superclase-subclase pueden ser difuminadas mediante el empleo de un grado en el rango $[0, 1]$. Esta situación queda perfectamente representada mediante el empleo de conjuntos difusos que indiquen cuáles son las subclases de una determinada clase.

- Valores difusos para atributos multivaluados: El modelo propone el empleo de conjuntos difusos para representar atributos multivaluados en sentido conjuntivo.

La segunda parte del modelo, la que se encarga del tratamiento de la incertidumbre e imprecisión, emplea la visión semántica de los conjuntos difusos como distribuciones de posibilidad. Con ello, se incluyen en el modelo:

- Valores imprecisos para los atributos: Este tipo de datos se modelan empleando distribuciones de posibilidad que, a su vez, son representadas como conjuntos difusos con semántica disyuntiva.

- Incertidumbre en objetos: Este tipo de incertidumbre se da cuando no se conoce con certeza la estructura del objeto. Para ello, se hace uso del concepto de rol, que modela aspectos determinados de los objetos. Cuanta mayor sea la incertidumbre que rodea al objeto, más roles tendrá asociados, pudiendo dicha asociación ser graduada.

- Modelado hipotético: De la misma forma que los roles permiten dotar de una estructura incierta a los objetos, estos mismos roles pueden ser empleados para permitir cierta incertidumbre en el esquema de las propias clases.

3.7.3.6. Modelo de Marín et al.

La propuesta Marín et at. [100, 101, 102, 99] busca la definición de un modelo de bases de datos orientado a objetos difusos que permita afrontar la imperfección que normalmente afecta a los datos del mundo real y que, también, permita hacer frente a la complejidad de las estructura de los mismos.

Para ello, se consolidan los aspectos más interesantes de las propuestas en la literatura, entre ellos los aportados por los modelos anteriores, y se proponen nuevos elementos que amplíen las capacidades del la propuesta.

El modelo propuesto trata la *vaguedad* de los datos del mundo real desde diversos niveles:

- Nivel de atributos: El modelo estudia distintas formas de representar un dominio, proponiendo tres formas fundamentales para la representación de la imprecisión a este nivel:
  - Datos imperfectos sin dominio básico asociado: Se aborda mediante la definición de dominios imperfectos sin representación semántica subyacente. Este tipo de datos pueden ser, por ejemplo, la *seriedad* de un diagnóstico, o la *calidad* de un producto.
  - Datos imperfectos sobre un dominio subyacente: Se aborda mediante la definición de dominios imperfectos con referencial subyacente con semántica disyuntiva. Un ejemplo de este tipo de datos puede ser la *edad* o la *altura* de una persona.
  - Datos imperfectos con multiples valores: Cuando un determinado atributo puede poseer múltiples valores, y cuando cada uno de esos valores tiene asociado un determinado grado, se pueden definir dicho tipo de atributos como un conjunto difuso de objetos. Con esta aproximación, se incluyen en el modelo los dominios imperfectos con referencial subyacente y semántica conjuntiva. Como ejemplo de este tipo de datos tenemos por ejemplo el *conjunto de idiomas* que habla un traductor.
- Nivel de similitud entre objetos: Para la recuperación de objetos desde la base de datos es fundamental disponer de un método para el cálculo del grado de similitud entre objetos. Este método se corresponde con el *Grado de Inclusión Guiado por Semejanza* (*Resemblance Driven Inclusión Degree*, en el original) [99], equiparando el grado de similitud de dos objetos como una agregación del par de grados de la doble inclusión de los mismos.

  En el trabajo, se propone el empleo de relaciones de similitud para el cálculo de semejanza de los valores de los atributos de los objetos comparados. En función del método empleado para el cálculo del grado de inclusión entre conjuntos difusos, se obtendrán diferentes operadores para calcular relaciones de parecido entre conjuntos difusos de objetos imprecisos. Las propuestas de operadores de similitud que hace el trabajo son: Semejanza Generalizada, Factor de Cardinalidad y Operador de Consistencia.
- Nivel de relaciones entre objetos: Las relaciones entre objetos, siendo estas difusas o no, quedan modeladas correctamente sin la necesidad de la inclusión de nuevos elementos en el modelo, todo ello de forma independiente al enfoque que se dé a dichas relaciones:

  - Enfoque funcional: Las relaciones con enfoque funcional están implementadas al nivel de los atributos. Por tanto, las propuestas realizadas para la representación de la vaguedad en dicho nivel son igualmente válidas para la representación de relaciones imperfectas entre objetos cuando se sigue el enfoque funcional.
  - Enfoque no funcional: Bajo este enfoque, las relaciones serán representadas mediante una clase que exprese explícitamente la agregación entre los objetos relacionados. En este caso, la vaguedad en la relaciones objeto-clase quedará en manos de la propuesta para la expresión de vaguedad en las relaciones objeto-clase, que describirán a continuación.

- Nivel de relaciones objeto-clase: La pertenencia de un objeto a una clase puede verse graduada debido a la vaguedad en los valores que caracterizan a los objetos. El modelo puede emplearse para describir las relaciones de inclusión, utilizando para ello atributos adicionales en la definición del tipo de una clase, que permitan representar la certeza que se tiene de que un objeto pertenezca a esa clase.

- Nivel de superclase-subclase: El trabajo estudia dos técnicas diferentes para la creación de subclases:

  - Adición de reglas: Se propone añadir reglas, que supongan restricciones sobre las propiedades que deben cumplir los objetos de una determinada superclase, para modelar la pertenencia a la subclase que se está definiendo. Este mecanismo permite obtener clases intensivas carentes de representación interna.
  - Técnica clásica: Esta técnica consiste en ampliar el tipo de la subclase mediante un conjunto adicional de atributos.

El modelo presenta las siguientes características destacadas:

- Empleo de los mecanismos propios del modelo orientado a objetos clásico para la representación de las nuevas estructuras propuestas.

- Estudio de la representación de la vaguedad diferenciando claramente sus dos matices: la imprecisión y la incertidumbre.

Como nota final, se ha destacar que, debido a que las estructuras propuestas en el modelo son representables haciendo únicamente uso de las características clásicas de la orientación a objetos, el trabajo presenta una arquitectura para la construcción de un Sistema de Gestión de Bases de Datos Orientadas a Objetos Difusos sobre la base de un sistema clásico ya existente.

De esta manera, es posible conseguir un sistema que incluya la nuevas características propuestas, y las características de un buen gestor de bases de datos, con un esfuerzo razonable. Dicho esfuerzo será mucho menor al que supondría la implementación de un nuevo sistema desde cero.

Implementación del modelo.

El trabajo de Marín incluye una implementación funcional basada en el modelo descrito anteriormente. FoodBi (Fuzzy Object Oriented DataBase Interface) es el prototipo, desarrollado por Marín, que se encarga de ofrecer una interfaz gráfica para la gestión de esquemas de objetos difusos.

Dicho prototipo permite al usuario definir de manera gráfica el esquema que define el modelo orientado a objetos difusos de las entidades presentes en un determinado problema. Una vez definido el modelo, el traductor difuso-clásico y el gestor de metadatos implementados en FoodBi, se encargan de generar las clases clásicas y los metadatos necesarios para que dicho modelo pueda ser representado en un sistema de gestión de bases de datos orientado a objetos clásico.Se ha de notar que FooBi no es un gestor de bases de datos orientadas a objetos difusos propiamente dicho. Éste implementa la interfaz para la definición de las estructuras que albergarán información imperfecta (parte que podríamos identificar como DDL) pero no ofrece capacidades para la manipulación de dicha información (cometido del sublenguaje DML).

La estrategia seguida por el autor para el diseño de una base de datos orientada a objetos difusos es la de implementar una capa de recubrimiento sobre un sistema de gestión de bases de datos orientas a objetos clásicos. Dicha capa de recubrimiento, llamada gestor conceptual de borrosidad, se compone de los siguientes elementos:

- Gestor de esquemas: Se encarga de resolver las tareas relacionadas con la definición de estructuras de la base de datos.

- Gestor de objetos: Ofrece funcionalidad para la creación, almacenamiento, manipulación y destrucción de objetos sobre las estructuras creadas por el anterior gestor.

- Gestor de metadatos: Se encarga del mantenimiento del catálogo que almacena los metadatos.

- Traductor clásico-difuso: Este elemento se encarga de traducir las peticiones de definición y manipulación de datos, producidas por los gestores anteriores, a las correspondientes peticiones que se han de hacer sobre el sistema de gestión de bases de datos orientadas a objetos *crisp* que se emplea como base. Este elemento es el único del sistema que es dependiente de la SGBD subyacente, por lo que, si se desea que el sistema emplee como base otro SGBD, dicho componente se deberá adecuar a la nueva situación.

El sistema de gestión de bases de datos orientada a objetos para el que está inicialmente diseñado el *traductor clásico-difuso* de FoodBi es una combinación de la máquina virtual Java y el sistema de gestión de bases de datos objeto-relacionales Oracle versión 8.1.6. Del primer componente, se aprovecha su lenguaje (Java) como lenguaje para la definición y manipulación de objetos. El segundo componente, Oracle, se emplea como plataforma encargada de la persistencia de dichos objetos.

Extensión del modelo

Cuevas [57] propone una extensión al modelo de Marín que se centra en la modelización de la vaguedad a nivel los valores de atributos y de la definición de objetos.

Una de las aportaciones incluidas en la propuesta es el concepto de *perspectiva de comparación*. Este concepto surge de la necesidad de la adaptación del operador de comparación de objetos en función del contexto en que ésta se realiza. Al dotar de perspectivas de comparación, se permite seleccionar la forma en que el objeto será comparado. De esta forma, se dotará a la operación de comparación de objetos una semántica específica según el contexto.

Además de lo anterior, se proponen diversas estrategias de comparación de objetos difusos y la incorporación de atributos con valores inferidos.

El tratamiento de la vaguedad a nivel de estructural se realiza gracias a los *tipos difusos*. Éstos modelan como un conjunto difuso la colección de atributos que tendrán los objetos de determinado tipo. De esta forma, es posible modelar la incertidumbre sobre composición estructural, entendida como el conjunto de atributos, de los objetos de un determinado tipo.

Finalmente, se ha de destacar que la propuesta del modelo de base de datos orientada a objetos difusos de Cuevas ha sido adaptada para acomodarla al modelo objeto-relacional e implementada sobre el SGBD objeto-relacional PostgresSQL [56]. Ésta implementación, denominada pg4DB, incorpora tanto las características del modelo de Marín como las extensiones propuestas por Cuevas.

3.7.3.7. Modelo de De Tré et al.

Para permitir el modelado simultaneo de los distintos tipos de imperfección que puede afectar a la información, De Tré et al. [63] proponen el uso de conjuntos difusos de nivel 2. Éstos son conjuntos difusos definidos sobre un universo que, a su vez, está formado por conjuntos difusos definidos sobre el dominio de discurso. Dado que los conjuntos difusos de nivel 2 son, en definitiva, un anidamiento de dos conjuntos difusos ordinarios, podemos distinguir en su estructura un *nivel interno* y un *nivel externo*.

La propuesta aprovecha el citado el anidamiento de los conjuntos difusos de nivel 2 para modelar simultáneamente dos tipos de imperfección. Por ejemplo, los grados de pertenencia de los elementos del dominio de discurso en el nivel interno pueden ser interpretados como grados de posibilidad que permitan modelar cierta información imprecisa o vaga. Por su parte, los grados de pertenencia del nivel externo pueden ser empleados para indicar la certidumbre asociada a la información.

En este marco, De Tré et al. [62] extienden el concepto de *tipo* empleado en ODMG[1] 3.0 [39], el estándar *de facto* para bases de datos orientas a objetos *crisp*, para definir el concepto de *tipo generalizado*. Éste concepto, al igual que su homólogo para el caso de las bases de datos orientas a objetos *crisp*, se emplea como pieza básica para la construcción del modelo de base de datos orientada a objetos difusos propuesto por los autores.

[1] Nombrado de esta forma debido a que fue propuesto por el Object Data Management Group (ODMG) ó, como se denominó a partir de 1998, Object Database Management Group.

3.8. Indexado en bases datos difusas

Uno de los aspectos que se tratan en el presente trabajo es el desarrollo de mecanismos que permitan a los SGBD difusos ser competitivos en entornos en producción. Dado que los mecanismos de indexado son una de las claves, junto con los mecanismos de optimización de consultas, para conseguir altas cotas de rendimiento en los SGBDs, dedicaremos la presente sección a la descripción del estado del arte en el campo del indexado de bases de datos difusas.

Aunque se ha dedicado mucho esfuerzo investigador en la proposición de modelos de bases de datos difusas, no han sido muchas las propuestas en el campo de las técnicas de indexado para éstas. Las primeras investigaciones sobre indexado de bases de datos difusas, hasta donde tenemos constancia, se remontan al trabajo pionero de Bosc et al. sobre los principios de indexado para bases de datos difusas [26]. En éste se expone la necesidad de técnicas específicas de indexado que puedan ser aplicadas en bases de datos difusas y se proponen dos principios de indexado para consultas flexibles que emplean, respectivamente, medidas de posibilidad y necesidad sobre atributos cuyos valores están definidos como distribuciones de posibilidad. Dado que las técnicas propuestas en el presente trabajo emplean como base estos principios de indexado, postergaremos la descripción de los mismos al Capítulo 8, en el que se podrá encontrar una descripción detalla de éstos.

Desde la publicación del trabajo pionero de Bosc et al., se han propuesto algunas técnicas de indexado. Podemos dividir estas propuestas entre las que emplean los principios de indexado de Bosc y las que no los emplean. De entre las técnicas basadas en los principios de indexado de Bosc, se pueden destacar los trabajos de Boss y Helmer [33, 78], así como el de Liu et al. [95]. Los primeros proponen una técnica de indexado para datos difusos definidos sobre dominios escalares. En esta propuesta se emplea como estructura base una *lista invertida* [176, 133]. Por su parte, Liu et al. proponen una técnica de indexado para números difusos. En éste caso, se emplea como estructura de indexado base un *árbol G* [88], un índice multidimensional compuesto de un directorio y un fichero de rejilla. Ambas propuestas permiten aplicar un número ilimitado de consultas flexibles diferentes, por lo que podemos considerarlas técnicas de indexado genéricas. Como en el caso de los principios de indexado de Bosc, postergaremos la descripción de ambas técnicas al Capítulo 8, ya que se emplearan como referencia para evaluar el rendimiento de las técnicas de indexado que se proponen en el presente trabajo.

En el grupo de las técnicas de indexado de datos difusos que no están basadas en los principios de indexado de Bosc, podemos destacar los trabajos de Bosc et al. [29], Petry et al. [118] y Yazici et al. [158, 159, 161]. Desafortunadamente, todas estas técnicas están diseñadas para aplicaciones en que el número de consultas flexibles distintas que se pueden realizar es fijo y, por tanto, finito y pequeño en la práctica. Por ésta razón, no consideramos a estas técnicas como técnicas de indexado de propósito general.

3.9. Conclusiones

En el presente capítulo se ha introducido brevemente el estado del arte en lo que se refiere al manejo de información imperfecta en bases de datos.

Tras la pertinente definición del concepto *información imperfecta* empleado en este trabajo, se han descrito las propuestas más relevantes de la bibliografía para el manejo de la misma en forma de base de datos, incluyendo propuestas basadas y no basadas en la lógica difusa, lenguajes de consulta flexible y tratamiento de la información imperfecta en bases de datos orientas a objetos.

Capítulo 4

Motivación de la propuesta

4.1. Introducción

El presente capítulo está dedicado a la exposición y al análisis de los motivos que nos conducen a la formulación de la propuesta que se realiza en el presente trabajo.

Para ello, analizaremos críticamente los principales modelos de bases de datos difusas propuestos en la literatura con el fin de detectar las carencias que éstos presentan. Así mismo, se analizarán las estrategias de implementación de dichos modelos. A continuación, analizaremos las tendencias actuales de la industria del desarrollo de sistemas de gestión de bases de datos. Finalmente, se extraerán las conclusiones pertinentes de los anteriores análisis y se emplearán éstas como directrices que determinarán el sentido de la propuesta del presente trabajo.

4.2. Evaluación de los modelos actuales de bases de datos difusas

En la presente sección, se realizará un análisis crítico de las propuestas actuales para la incorporación de capacidades de representación y manipulación de información imperfecta en bases de datos, centrándonos exclusivamente en el empleo del paradigma difuso. Este análisis tiene como objetivo detectar las carencias de dichas propuestas para, posteriormente, realizar una propuesta que permita paliarlas.

En el capítulo anterior, se ha realizado repaso a los modelos de bases de datos difusas más relevantes en la actualidad. En general, los modelos analizados se basan en un determinado modelo base de datos *crisp* que extienden para incluir capacidades de manejo de información imperfecta. Estas propuestas han quedado divididas claramente en dos grupos en función del modelo en que se basan: las *bases de datos relacionales difusas* y las *bases de datos orientadas a objetos difusos*.

A continuación, procederemos a realizar dicho análisis crítico particularizando éste en función de los grupos anteriormente mencionados.

4.2.1. Bases de datos relacionales difusas

Como ya se expuso en secciones anteriores, los primeros modelos de bases de datos que incorporaron capacidades para el tratamiento de información imperfecta, empleando la lógica difusa, se basaron el en modelo relacional.

A pesar de la gran capacidad para el almacenamiento y manipulación de información imperfecta que ofrecen los modelos de bases de datos relacionales difusas, éstos sufren de ciertos inconvenientes heredados del modelo relacional, en el que están basado estos modelos. Éstos, detallados en el capítulo anterior, se resumen a continuación:

- Dificultad de modelado: Los elementos presentes en el mundo real, normalmente complejos e interrelacionados, no suelen ser modelados con facilidad empleando el modelo relacional. Además, cuando la imperfección está presente no solo en los valores de la base de datos, si no también en la estructura de las entidades, en la organización de éstas, y en la interrelación de las mismas, las bases de datos relacionales difusas no permiten un modelado correcto de dichos elementos.

- Poca adecuación del modelo de datos relacional a las necesidades de las nuevas aplicaciones: Con las nuevas aplicaciones (por ejemplo, CAD, CAM, CASE, GIS, médicas, científicas, multimedia, etcétera) que se apoyan en bases de datos para la gestión de los datos que manejan, los sistemas de bases de datos relacionales tienden a incurrir en ciertos problemas, como la falta de eficiencia en la recuperación de datos y la dificultad de modelado de dichos datos empleando el modelo relacional.

- Ineficiencia en los servicios de persistencia ofrecidos a las aplicaciones programadas en lenguajes orientados a objetos: En la actualidad, la mayor parte de las aplicaciones se desarrollan empleando lenguajes orientados a objetos. Los servicios de persistencia de datos que necesitan dichas aplicaciones se ven forzados a realizar un proceso de mapeo entre el modelo orientado a objetos, del lado de la aplicación, y el modelo relacional, disponible en la base de datos. Dicho proceso de mapeo, que conlleva cierto coste computacional, junto con el acceso típico a los datos en los modelos orientados a objetos (el acceso navegacional) hacen que las bases de datos relacionales se muestren ineficientes en el proceso de recuperación y manejo de información de dichos sistemas.

4.2.2. Bases de datos orientadas a objetos difusos

Como evolución de los modelos de bases de datos relacionales difusas y con el objetivo de solventar los problemas que éstos presentan, las propuestas siguen la tendencia evolutiva de las bases de datos clásicas hacia la orientación a objetos. El resultado de este proceso son los modelos de bases de datos orientadas a objetos difusos.

Las bases de datos orientadas a objetos difusos ofrecen un entorno muy potente para la representación de información imperfecta a nivel de valores, estructura, así como de organización e interrelación. Esto permite modelar con facilidad entidades procedentes del mundo real incluso cuando éstas se ven afectadas por cierta imperfección, lo que permite dar soporte de forma conveniente

a las nuevas aplicaciones para bases de datos. Además, se añade el valor de permitir a éstas el almacenamiento y manipulación de información imperfecta, lo que abre nuevas posibilidades para las mismas.

A pesar de todo ello, como ocurre en el caso de las bases de datos relacionales difusas, las bases de datos orientadas a objetos difusos sufren de los problemas heredados del modelo en el que se basan: el modelo orientado a objetos. Estos problemas, expuestos en capítulos anteriores, se resumen a continuación:

- Debilidad ante cambios en el esquema: En una base de datos orientada a objetos, añadir, modificar o eliminar clases suele requerir modificaciones en las clases de la aplicación que interactúan con éstas. Además, se ha de considerar que es necesaria la adaptación de las instancias pertenecientes a las clases modificadas, ya existentes en la base de datos, a las nuevas definiciones de sus clases. Por tanto, estas modificaciones en la base de datos normalmente hacen necesario una recompilación íntegra del sistema.

- Dependencia del lenguaje: Las bases de datos orientadas a objetos suelen estar fuertemente ligadas a un lenguaje de programación orientado a objetos concreto por medio de la API de acceso a las mismas. Esto significa que la base de datos sólo puede ser accedida desde aplicaciones desarrolladas usando un determinado lenguaje para el que existe la API de acceso correspondiente.

- Carencias en la capacidad de consulta: Debido a que no es posible emular la semántica de la unión relacional en el modelo orientado a objetos mediante la unión de dos clases, se hace evidente que este modelo posee una flexibilidad mucho menor a la hora de aceptar consultas por parte de los usuarios. Además, las consultas que pueden ser aplicadas sobre los datos en el modelo orientado a objetos están determinadas por el diseño del sistema.

  Finalmente, la encapsulación de los objetos puede impedir que ciertas consultas empleen criterios en base a condiciones impuestas al estado interno de los objetos, cuando estos objetos no muestran dicho estado mediante una interfaz pública. Esto dificulta aun más la ejecución de consultas ad-hoc.

4.3. Análisis de las arquitecturas de los sistemas de gestión de bases de datos difusas

Las implementaciones de gestores de bases de datos difusas emplean fundamentalmente dos aproximaciones para el desarrollo de las mismas:

- Implementación de un SGBDD *ad-hoc*: Esta aproximación requiere un gran esfuerzo de implementación hasta obtener un SGBD con todas las características exigibles a este tipo de sistemas, estable y con un rendimiento aceptable. Normalmente este tipo de proyectos no pasan de ser prototipos muy incompletos, aceptables a nivel de investigación pero, normalmente, inadecuados para las necesidades en entornos en producción.

  Implementar el sistema gestor desde cero, permite a sus diseñadores dotar a dicho sistema de tantas capacidades de almacenamiento y manipulación

de información imperfecta como hayan podido idear, lo que proporciona a este tipo de sistemas una alta potencia en el almacenamiento y manipulación de información imperfecta.

- Empleo de un SGBD existente como anfitrión: Esta aproximación requiere menor esfuerzo de implementación que la anterior. Además, al estar basada en un sistema ya existente, siendo éste normalmente de nivel profesional, el SGBDD obtenido hereda la estabilidad y rendimiento del SGBD anfitrión. Debido a la estabilidad y rendimiento heredada del anfitrión, este tipo de desarrollos pueden ser empleados en grandes sistemas en producción.

  Ya que la implementación que siga esta aproximación debe ajustarse a las características de sistema anfitrión, es posible que ésta se vea forzada a restringir las capacidades de almacenamiento y tratamiento de información imperfecta ofrecidas.

Como ha quedado patente en el análisis anterior, las implementaciones basadas en extensiones de sistemas de gestión de bases de datos existentes, a pesar de que exista la posibilidad de que sean menos potentes en el almacenamiento y manipulación de información imperfecta que las implementaciones *ad-hoc*, son las más adecuadas a la hora de aplicar éstas a problemas reales, en los que el volumen de datos y de consultas, hacen necesario que el SGBDD ofrezca un gran rendimiento y estabilidad.

4.3.1. Sistemas basados en traductores

Dentro del conjunto de las implementaciones de SGBD difusos basas en un SGBD anfitrión, destacan aquellas que emplean como aproximación el uso de un sistema de traducción [70, 103, 100] que transforme el lenguaje de consulta difuso en el lenguaje de consulta del sistema anfitrión.

A pesar de las ventajas que ofrece esta estrategia de implementación, existen ciertos inconvenientes asociados a la misma:

- Imposibilidad de empleo de todas capacidades de consulta disponibles en el sistema anfitrión: Típicamente, el lenguaje de consulta difuso del SGBDD se creará como una extensión del lenguaje de consulta de sistema anfitrión. No obstante, por motivos de simplicidad la citada extensión no suele considerar todos los elementos disponibles en el sistema.

  Aún siendo total la base del lenguaje de consulta del sistema anfitrión que es considerada para la definición del lenguaje de consulta difuso, una vez establecido éste se implementará el traductor que permitirá la ejecución de las consultas expresadas en este lenguaje. A partir de este momento, el lenguaje de consulta difuso queda aislado de posibles evoluciones del lenguaje en que se basa. Según esto, los usuarios no podrán emplear en las consultas difusas determinadas facilidades que hayan sido incorporadas con posterioridad al lenguaje de consulta de sistema anfitrión (por ejemplo, al instalar el SGBDD sobre una versión posterior del sistema anfitrión), ya que el proceso de traducción no reconocerá la mismas y fallará.

- Imposibilidad de uso de las características de valor añadido que ofrece el sistema anfitrión: Algunos sistemas de gestión de bases de datos relacionales ofrecen características de valor añadido, como son facilidades para

el tratamiento de datos espaciales, series temporales, procesamiento de textos, clustering, tratamiento de XML, etcétera. Estas características se ofrecen en forma de extensiones del lenguaje SQL, para poder llevar a cabo de una forma sencilla cierto tipo de consultas y manipulación de datos.

Al igual que ocurre en caso descrito anteriormente, cuando el usuario desea hacer uso de dichas extensiones en consultas difusas, el traductor de sentencias no reconoce las mismas e indica un fallo al usuario.

- Imposibilidad de incorporar capacidades de optimización e indexado específicas para datos imperfectos: En esta aproximación, la naturaleza de los datos imperfectos sólo es conocida por el sistema de traducción de consultas difusas. Por tanto, el sistema anfitrión no percibe dichos datos como imperfectos, ya que éstos se representan mediante las estructuras ordinarias ofrecidas por dicho sistema.

  Esto imposibilita que el sistema anfitrión haga uso, si éste las ofrece, de capacidades para la definición por parte del usuario de mecanismos de optimización de consultas o indexado más apropiados y eficientes para los tipos de datos imperfectos, ya que es incapaz de detectar que la naturaleza de los mismos.

- Reducción del rendimiento en el procesamiento de sentencias: Debido a que es necesario un proceso de traducción para las sentencias de consultas difusas, cada vez que se ejecuta este tipo de sentencias se deben ejecutar doblemente procesos de análisis léxico, sintáctico y semántico. El primer grupo de análisis se ejecuta sobre la sentencia de consulta difusa que se desea traducir. El segundo se ejecuta, por parte del SGBD anfitrión, sobre la sentencia SQL obtenida del proceso de traducción cuando ésta es ejecutada por el mismo. Esta ejecución doble puede mermar el rendimiento del procesamiento de consultas.

Dentro de esta clase de sistemas, se puede diferenciar entre los que implementan el sistema de traductor haciendo uso de los mecanismos de extensión funcional del anfitrión [70, 103] o los que implementan el sistema de traducción como un componente externo al sistema anfitrión [100]. A continuación analizaremos cada una de las alternativas.

4.3.1.1. Traductor interno

La primera de las alternativas implica que el traductor de consultas y sus funciones auxiliares (por ejemplo, para la representación y comparación de datos imperfectos) están implementados internamente en el servidor. Ésta es la arquitectura empleada, por ejemplo, por FSQL Server [70, 103].

Este enfoque proporciona capacidades de acceso y manipulación de la información imperfecta de forma sencilla, tanto del lado del servidor como desde aplicaciones cliente, debido a que la invocación del sistema de traducción se realiza con los mismos mecanismos que se emplean para el envío de consultas al sistema anfitrión.

No obstante, el empleo de un traductor interno aumenta el impacto de alguna de las desventajas debidas al empleo de traductores. Se ha de tener en cuenta que el proceso de análisis y traducción de sentencias difusas se ha de implementar como un procedimiento almacenado en el servidor, empleando el lenguaje

procedural que ofrece el sistema anfitrión. En los SGBDs actuales, es común que el código escrito en dicho lenguaje se interprete o, a lo sumo, se compile a un código intermedio. Este último, aunque más próximo al código máquina, requiere también para su ejecución el empleo de un intérprete. Esto provoca que la ejecución de la rutina de traducción, así como las auxiliares de comparación y representación, añada más carga al sistema y, por tanto, repercute negativamente en el rendimiento del mismo.

4.3.1.2. Traductor externo

Otra posible alternativa en las arquitecturas basadas en el empleo de traductores es la de implementar éste de forma externa al SGBD anfitrión que se emplea. Esta aproximación permite dividir de la lógica del SGBDD. Por un lado se dispondrá de una parte funcional que se encargará en la traducción de las sentencias del lenguaje de consulta difuso y, por otro lado, se contará con un segundo componente funcional formado por un SGBD clásico.

Por ejemplo, el sistema FooBi [100], que sigue la aproximación descrita, realiza esta división de la siguiente forma:

- La interfaz de usuario y el gestor de borrosidad: Esta parte se ejecuta como cliente en la máquina del usuario del sistema gestor de bases de datos difusas.
- El sistema de persistencia de las clases: Esta parte se ejecuta de forma independiente a la anterior.

La ventaja fundamental de este enfoque es que el sistema de persistencia puede estar alojado en una máquina con suficiente potencia como para servir a un gran número de usuarios. Así, es posible que los usuarios del sistema compartan recursos como el espacio en disco y la capacidad de procesamiento, librando a la máquina cliente de dicha carga.

Por otra parte, al residir gran parte de la lógica del sistema gestor en el cliente, el uso compartido de dicha lógica es imposible, por lo que el mantenimiento de la misma se hace difícil en un entorno multiusuario. Cuando dicha lógica deba ser modificada con objeto de mejorarla, mantenerla o adaptarla a ciertos cambios en el sistema gestor de bases de datos orientadas a objetos subyacente, el proceso deberá actualizar, uno a uno, el software de que disponen los clientes.

4.3.2. Sistemas basados en la extensión nativa del anfitrión

Una alternativa a los sistemas basados en traductores es el desarrollo del SGBDD como una extensión nativa del SGBD anfitrión. La primera propuesta de este tipo de sistemas fue realizada por Cubero, Marín, Medina, Pons y Vila [55]. Con posterioridad, esta propuesta ha sido ampliada [10, 44, 45]. Dicha ampliación será descrita en el Capítulo 7 del presente trabajo. Finalmente, Cuevas [56, 57] también emplea esta aproximación en pg4DB.

Esta alternativa, posterior a la basada en traductores, resulta posible a partir de la incorporación en los SBGD *crisp* de mecanismos de extensión basados en las novedades incluidas en el estándar SQL:199 (que se tratarán en mayor profundidad en la siguiente sección). Gracias a las nuevas posibilidades que

ofrece el empleo de UDTs, es posible definir tipos específicos que permitan la presentación y manejo, mediante lógica encapsulada, de información imperfecta en un SGBD clásico.

Esta clase de sistemas mantienen los aspectos positivos de los sistemas de traducción interna y neutralizan, en la medida en que las características de extensión del SGBD anfitrión lo permiten, las desventajas señaladas para los mismos.

4.4. Tendencias actuales de la industria de sistemas gestores de bases de datos

Debido a que las implementaciones más completas de bases de datos difusas se basan en un SGBD anfitrión, la presente sección analizará las tendencias que siguen los desarrolladores y las entidades de estandarización de este tipo de sistemas. Particularmente, el análisis se centrará en las capacidades de extensión de los SGBDs, con el fin de identificar aquellos sistemas más adecuados para la implementación de un SGBDD siguiendo una aproximación que minimice los inconvenientes mencionados en la sección anterior.

Para ello, analizaremos los mecanismos de extensión por parte del usuario que contemplan los estándares actuales, SQL:1999, y el grado de implementación de dichos mecanismos en los sistemas más populares disponibles en la actualidad. El objetivo de esta sección es analizar qué tipo de mecanismo es el más común y conveniente para el desarrollo de una extensión de dichos sistemas para el almacenamiento y manipulación de información imperfecta, permitiendo así justificar las decisiones de diseño e implementación de dicha extensión.

4.4.1. Mecanismos de extensión de SQL:1999

La serie de estándares SQL que fueron publicados a lo largo de la vida de dicho lenguaje (SQL-86, SQL-89 y SQL-92) no recogieron, hasta 1996, la posibilidad de que los usuarios pudiesen definir ciertas extensiones para el tratamiento de sus datos. En dicho año, la publicación de SQL/PMS, una adición al estándar conocida como PMS-96, introduce en el estándar SQL la posibilidad de que el usuario defina procedimientos y funciones. La definición de tales procedimientos y funciones se realiza mediante el empleo de lenguajes procedurales. Éstos serán ejecutados y almacenados en el servidor y podrán ser invocados dentro de sentencias SQL. Esta incorporación al estándar responde al gran éxito de los mecanismos de extensión procedural propietarios que incluyeron ciertos fabricantes en sus productos.

La posibilidad de definir por parte del usuario funciones y procedimientos, permite la creación de conjuntos de dichas funciones y procedimientos para realizar procesamientos complejos de los datos dentro de las sentencias SQL.

Posteriormente, con la publicación del estándar SQL:1999, las capacidades de extensión de los sistemas SQL se incrementan de forma notable. El nuevo estándar contempla, además de las posibilidades de extensión en base a procedimientos y funciones ya incluidas en su parte fundacional, la posibilidad de definir tipos complejos, llamados *tipos definidos por el usuario* o, en el original, *User Defined Types* (UDTs).

Los UDTs, además de permitir la definición tipos de datos complejos, pueden incluir en su definición procedimientos y funciones asociados a los mismos. De esta forma, la lógica asociada a cada UDT se encuentra encapsulada en dicho tipo. Esto, en definitiva, abre la posibilidad de que las bases de datos SQL permitan al usuario definir y trabajar con objetos complejos con métodos asociados.

Como se puede apreciar, la incorporación de los UDTs en SQL:1999 abre las puertas a la extensión de los sistemas de gestión de bases de datos SQL. Es posible desarrollar un conjunto de UDTs empaquetados con el objeto de añadir los tipos y el procesamiento necesario para dotar de capacidades de representación y manipulación de datos procedentes de aplicaciones complejas como, por ejemplo, datos espaciales, médicos, multimedia, etcétera.

Adicionalmente, el estándar SQL:1999 incorpora la extensión de SQL-92 denominada SQL/OLB y una nueva parte conocida como SQL/JRT. La extensión SQL/OLB (Object Language Bindings), publicada en 1998 y conocida como OLB-98, define los mecanismos de incrustación de código SQL en código Java. Por su parte, SQL/JRT (Java Routines and Types) define los mecanismos necesarios para permitir la implementación de procedimientos y funciones almacenadas en el servidor empleando código Java, la inclusión de tipos definidos en Java como tipos de usuario de la base de datos y los mecanismos para el empaquetado y la carga automática de estos tipos y rutinas en un servidor SQL.

Estas últimas partes de SQL:1999 abren la posibilidad del desarrollo en el lenguaje Java de paquetes de UDTs y sus métodos asociados. Esta posibilidad permitiría que dichos desarrollos puedan ser incorporados de forma directa en cualquier sistema SQL, independientemente del fabricante. Teniendo esto en cuenta, la posibilidad de desarrollo de paquetes para el almacenamiento y tratamiento de datos complejos se ve reforzada, ya que, al dotar de independencia dichos paquetes con respecto de los fabricantes de sistemas SQL, estos podrían ser instalados y funcionar en virtualmente en cualquier base de datos que se ajuste al estándar SQL:1999.

Con esta visión del estándar, se hace patente que las posibilidades de extensión de un sistema SQL pasan por la definición de UDTs con lógica encapsulada que permita el almacenamiento y manipulación de los mismos. Esto es, en definitiva, el empleo de las características objeto-relacionales de los sistemas SQL.

4.4.2. Análisis de los sistemas disponibles en el mercado

La presente sección está dedicada al análisis de los sistemas SQL que, en mayor o menor medida, incorporan capacidades para la extensión de los mismos en forma de UDTs.

4.4.3. PostgreSQL

PostgreSQL, es actualmente el único SGBD de libre distribución que puede ser considerado objeto-relacional. El desarrollo de este sistema pretende ajustarse al máximo al estándar SQL:1999, por lo que, entre otras funcionalidades, ofrece soporte para los tipos definidos por el usuario. No obstante, este soporte es limitado ya que, aunque permite la definición de tipos estructurados, PostgreSQL no soporta colecciones [140].

Hasta el momento, PostgreSQL permite la definición de tipos estructurados, pero no incorpora mecanismos de herencia para ellos, ni permite la inclusión de métodos en los mismos, así como las referencias a objetos. En cambio, como contraprestación, ofrece herencia sobre tablas y funciones definidas por el usuario que permiten manipular, tanto tipos básicos, como tipos definidos por el usuario. Esta última capacidad permite incorporar lógica para dichos tipos en forma de funciones en lugar de métodos aunque, por otra parte, también significa la imposibilidad de encapsulado de ésta.

4.4.4. DB2 Universal Database

DB2 Universal Database es otro producto que puede ser considerado como un sistema de gestión de bases de datos objeto-relacionales. Propiedad de IBM, este producto es la combinación del sistema relacional de IBM, denominado DB2, e Informix, producto de la compañía de mismo nombre adquirida por IBM. En la combinación de los productos, Informix aporta a DB2 características objeto-relacionales. A su vez, las características objeto-relacionales de Informix fueron heredadas por éste del sistema Ilustra, adquirido por Informix Software previamente a su compra por IBM.

Este sistema ofrece una amplia variedad de características objeto-relacionales y de mecanismos de extensibilidad, como son la definición de tipos de usuario, las jerarquías de tipos mediante herencia, la definición de métodos, herencia de tablas, referencias a objetos, definición de índices por parte del usuario y operadores definidos por el usuario. El sistema está orientado a dar soporte a los objetos almacenados como valores de columnas, en detrimento de la representación de objetos en forma de filas, y se apoya fuertemente en Java para la definición e implementación de dichos objetos.

El fabricante da soporte a paquetes de UDTs y su funcionalidad asociada, conocidos en el mundo Informix como *DataBlades* y denominados por IBM como *extenders*. De esta forma, es posible desarrollar paquetes para el tratamiento de datos complejos que pueden ser instalados en el servidor de forma transparente. Dado el diseño modular de los paquetes, es posible la coexistencia de varios de ellos en un mismo servidor, permitiendo incluso la interacción y complementación entre los mismos.

4.4.5. SQL Server

Este producto de Microsoft es la apuesta de la compañía para su gama de servidores de bases de datos.

De entre características documentadas de este servidor se incluyen algunas que permiten calificarlo como un servidor de bases de datos objeto-relacionales. El servidor da soporte a tipos definidos por el usuario, cuyos métodos y propia definición puede hacerse empleando código intermedio CLR (Common Language Runtime). El código CRL puede ser generado compilando cualquier código escrito en un lenguaje de programación soportado por la plataforma .NET (por ejemplo, C# o Visual Basic), propiedad de Microsoft.

La definición de UDTs empleando CLR ofrece la ventajas de incorporar toda la potencia de los lenguajes orientados a objetos de la plataforma .NET, entre otras, las aportadas por la herencia y el polimorfismo. Por otra parte, el uso

de CLR implica el empleo una plataforma propietaria, lo que hace imposible la migración del código a plataformas de otros fabricantes.

4.4.6. Oracle RDBMS

Oracle Database, a partir de su versión 9, puede considerarse uno de los servidores de bases de datos más avanzados en lo que respecta a la tecnología objeto-relacional y a los mecanismos de extensión del servidor.

El sistema da soporte a UDTs, permitiendo herencia de tipos, sustitución de tipos, polimorfismo por herencia, métodos asociados a los UDTs, con sobrecarga de dichos métodos, e implementación de los mismos internamente, mediante el uso de diferentes lenguajes procedurales, o de forma externa.

Una diferencia fundamental, con respecto a otros, de este servidor es que, a pesar de que soporta el almacenamiento de objetos en columnas y filas de tablas, el soporte para objetos almacenados en forma de fila está más avanzado que el soporte para objetos en columnas.

El servidor permite que los UDTs tengan asociada una representación en lenguaje Java, así como que el cuerpo de los métodos de dichos UDTs sea implementado en dicho lenguaje. Esto favorece la cooperación entre aplicaciones y el servidor de base de datos, la independencia de las definiciones con respecto a la plataforma y la incorporación y reutilización de código Java procedente de otras aplicaciones.

Con respecto a los mecanismos de extensión, aparte de los citados UDTs, el servidor da soporte a la definición de operadores por parte del usuario (destacando el soporte que se da a los operadores auxiliares), el soporte para la definición de nuevas estructuras de indexado específicas para los UDTs y la definición por parte del usuario de funciones de evaluación de costo y selectividad para el optimizador. Los operadores auxiliares (o, en inglés, *ancillary operator*) [114] permiten la transferencia de datos entre los operadores aplicados en la parte WHERE de la sentencia a los operadores presentes en la parte SELECT. En lo que respecta a las funciones de evaluación de costo y selectividad definidas por el usuario, éstas permitirán modificar el comportamiento por defecto del optimizador de consultas, ayudándolo a optimizar correctamente las sentencias que contienen llamadas a operadores definidos por el usuario.

Un conjunto de UDTs, junto con los operadores definidos por el usuario asociados, así como sus estructuras de indexado y funciones auxiliares para el optimizador de consultas, pueden ser agrupados en un solo paquete que, en el caso de éste servidor, se conoce con el nombre de *Data Cartridge*.

Los *Data Cartridges* pueden ser instalados de forma sencilla en cualquier servidor Oracle, dotando a estos servidores de nuevas capacidades para manejar cierto tipo de datos complejos (espaciales, temporales, multimedia, etcétera). La funcionalidad incluida en los *Data Cartridges* podrá ser compartida por los usuarios del sistema, lo que reduce el coste de instalación y mantenimiento. Además, el acceso a la funcionalidad de un *Data Cartridge* deberá de estar autorizado por el administrador del sistema, lo que permite ofrecer cierto tipo de características sólo a un determinado grupo de usuarios.

4.4.7. Resumen de las capacidades de los sistemas disponibles en el mercado

A pesar de que la gran mayoría de los productos presentes en el mercado poseen ciertas facilidades para la extensión de los mismos, podemos señalar a *DB2 Universal Database* y a *Oracle* como los servidores con mayores capacidades de extensión por parte de los usuarios. Estas capacidades se pueden resumir en los elementos del siguiente listado:

- Potente mecanismo para la definición de tipos de usuario, incluyendo herencia de tipos, métodos, sobrecarga y polimorfismo.
- Operadores definidos por el usuario.
- Estructuras de indexado definidas por el usuario.

Además de lo anterior, se ha de destacar que el servidor Oracle ofrece ciertos mecanismos de extensión adicionales. Estos son los siguientes:

- Operadores auxiliares.
- Optimizador de consultas extensible.

4.5. Conclusiones

La presente sección resume los diferentes motivos, que han ido poniéndose de manifiesto en las secciones anteriores, que justifican la propuesta del presente trabajo y las determinaciones tomadas en base éstos en lo que respecta al modelo e implementación que se propone en el mismo.

4.5.1. Modelo de base de datos

Los modelos de bases de datos difusas analizados anteriormente presentan una serie de inconvenientes en función del modelo en el que se hayan basado, ya que los heredan del mismo.

Los sistemas basados en el modelo relacional presentan los siguientes problemas:

- Dificultad de modelado de datos complejos.
- Ineficiencia en la interacción de la base de datos con aplicaciones desarrolladas en lenguajes orientados a objetos.

Los sistemas basados en modelos orientados a objetos sufren los siguientes problemas:

- Debilidad ante cambios en el esquema.
- Fuerte dependencia del lenguaje en el que ha sido desarrollado el sistema de gestión de bases de datos.
- Poco o nulo soporte a consultas ad-hoc.

Dadas estas dificultades, se hace evidente que es deseable el desarrollo de un modelo de base de datos difuso basada en alguna alternativa. Como se indicó en el Capítulo 2, el modelo objeto-relacional mitiga en gran medida los inconvenientes citados anteriormente.

Por todas estas razones, nuestra propuesta aborda la definición de un modelo de bases de datos objeto-relacional difuso que permite beneficiarse de las ventajas del modelo relacional, en lo relativo a la simplicidad en la elaboración de consultas ad-hoc, al que incorpora capacidad para el tratamiento de datos complejos de naturaleza imperfecta. La estrategia adoptada por el modelo teórico propuesto busca extender las capacidades del enfoque objeto-relacional para tratar con información imperfecta, integrando el tratamiento crisp como un caso particular del modelo.

Además de lo anterior, dado el carácter integrador del modelo propuesto, se considerará en el mismo el tratamiento de multirrelaciones. De esta forma, se integra en el modelo la base estructural admitida de forma implícita en la inmensa mayoría de los SGBD relacionales y objeto-relacionales, así como en el estándar SQL:1999 para el caso de relaciones anidadas.

4.5.2. Estrategias de implementación de sistemas de gestión de bases de datos difusas

Anteriormente, ha quedado patente en el análisis de las implementaciones existentes, que la aproximación para la implementación de un SGBDD basándose en la extensión de un anfitrión clásico, posee muchas más ventajas que el desarrollo de un sistema ad-hoc. Las principales ventajas se relacionan a continuación:

- Herencia de las características del sistema anfitrión, en especial en lo que se refiere a rendimiento del sistema y a la funcionalidad añadida que ofrezca dicho servidor anfitrión (capacidades de clustering, tratamiento de datos espaciales, multimedia, texto, XML, etcétera).

- Menor esfuerzo de implementación.

Atendiendo a las ventajas de esta alternativa, nuestra propuesta de implementación se basa en la extensión de un SGBD objeto-relacional para dotarle de capacidades para el tratamiento de información imperfecta.

Siguiendo la mencionada arquitectura, podemos encontrar en la literatura varias estrategias de implementación: uso de traductores (tanto externos como internos) y empleo de tipos definidos por el usuario.

Como se ha indicado en el presente capítulo, la primera de las estrategias tiene asociados los siguientes inconvenientes:

- Los traductores no permiten emplear todas las capacidades que ofrece este lenguaje de consulta.

- Imposibilidad de uso de características adicionales incorporadas en el servidor anfitrión, en forma de extensiones propietarias de SQL, si estas no se han tenido en cuenta durante el diseño del traductor.

- Incapacidad del servidor para determinar la naturaleza imperfecta de los datos, lo que impide el uso de estructuras de indexado y mecanismos de optimización adecuados para este tipo de datos.
- Menor rendimiento en el procesamiento de sentencias debido a la necesidad de interpretar doblemente éstas.

Por otra parte, la segunda de las estrategias mencionadas, el empleo de tipos definidos por el usuario, neutraliza las anteriores desventajas.

Dadas las bondades de esta última alternativa, la implementación del modelo se realizará siguiendo esta estrategia. De esta forma, se consigue la integración con el SGBD anfitrión de manera que puedan usarse de forma conjunta todas las características de éste, siendo el lenguaje para la consulta y manipulación de datos imperfectos el mismo dialecto SQL proporcionado por el SGBD anfitrión. Adicionalmente, al no resultar necesario el empleo de traductores, se gana en eficiencia, extensibilidad e integración con el resto de características del sistema anfitrión. Finalmente se ha de destacar que esta estrategia de implementación permite la incorporación de mecanismos de indexado y optimización apropiados para los datos imperfectos.

4.5.3. Tendencias de la industria de sistemas de bases de datos

Tanto los estándares publicados como los productos analizados, indican que la industria de sistemas de bases de datos enfoca los nuevos desarrollos hacia servidores objeto-relacionales. Dichos servidores están dotados de altas capacidades de extensión para permitir la adaptación de los servicios de almacenamiento, acceso y manipulación de información, a las necesidades específicas del usuario. Estas capacidades extensivas se materializan de las siguientes formas:

- Tipos de datos definidos por el usuario, que permiten al usuario crear estructuras de almacenamiento más adecuadas para sus datos. Esta tarea se potencia gracias al soporte por parte de estos sistemas de herencia de tipos, métodos y polimorfismo.
- Operadores definidos por el usuario, que permiten incorporar a SQL nuevos operadores más apropiados para los UDTs.
- Extensibilidad de los sistemas de indexado y optimización de consultas, lo que permite adaptar éstos a los nuevos tipos de datos introducidos por los usuarios.
- Agrupamiento de los elementos anteriores en módulos que se incorporan fácilmente el sistema y lo dotan de funcionalidad para el tratamiento de un tipo específico de datos.

Las anteriores características hacen que los sistemas objeto-relacionales resulten muy apropiados para ser empleados como sistemas anfitrión de un SGBDD. Como se concluye en la Subsección 4.4.7 del presente capítulo, Oracle es en la actualidad el SGBD disponible en el mercado con un mayor soporte para las características objeto-relaciones. Considerando lo anterior, en el presente trabajo optaremos por el empleo de un sistema objeto-relacional, concretamente Oracle, como anfitrión.

Capítulo 5

Modelo objeto-multirrelacional difuso: modelo de datos

5.1. Introducción

Podríamos definir una base de datos como un conjunto de datos exhaustivo y no redundante relativo a las propiedades y relaciones entre las entidades a las que se refieren dichos datos. El modelo de una base de datos describe la forma, o primitivas, en que dicho conjunto de datos se organiza.

En este capítulo propone un modelo de base de datos que, a diferencia de otros modelos, presenta las siguientes características distintivas:

1. Permite representar información cuantitativas y cualitativas sobre las relaciones entre las entidades. Las participación de una entidad en una relación podrá ser múltiple y estar graduada.

2. Permite representar valores imperfectos (datos afectados por incertidumbre y/o imprecisión) relativos a las propiedades de las entidades.

3. Permite representar valores ausentes de diversa naturaleza.

4. No restringe los dominios de los atributos al subconjunto de los dominios de valores atómicos. El modelo admite dominios cuyos valores provienen de la composición de valores pertenecientes a otros dominios más básicos.

La concepción del presente modelo ha empleado como base al modelo relacional, debido a su indiscutible aceptación en el ámbito de las bases de datos y a sus excelentes cualidades. Por tanto, los elementos básicos en que se organiza la información en este modelo serán conceptualmente equivalentes a las relaciones. Éstos, que denominaremos *multirrelaciones complejas difusas*, poseen una estructura básica similar pero extendida, a fin de acomodar las extensiones del modelo relacional que se consideran en el presente trabajo y de las que surgen las características distintivas del nuevo modelo. Es por ello, que este modelo recibe el calificativo *multirrelacional*.

Este capítulo se organiza de la siguiente forma. En primer lugar, se describirán los conceptos básicos en que se apoyará la definición de modelo propuesto, a fin de definir un marco de referencia para el lector. Posteriormente, se definirán los tipos de dominios contemplados en el modelo, parte fundamental para lograr la expresividad requerida en lo que se refiere a los valores de las propiedades de las entidades. En tercer lugar, se definirán los elementos básicos de organización de información del modelo, las multirrelaciones complejas difusas. En cuarto lugar se describirá la capacidad del modelo para la representación de valores ausentes. Finalmente, se detallarán las conclusiones relativas al contenido del capítulo.

5.2. Conceptos básicos

Los conceptos sobre los que trata el presente apartado son genéricos y ampliamente conocidos dentro del campo de las bases de datos. No obstante, procederemos a definir los conceptos básicos en los que se apoya la propuesta antes de abordar la misma, todo ello con el objetivo de proporcionar al lector una referencia rápida sobre los mismos y puntualizar el significado de éstos en el ámbito de la presente propuesta.

Comenzaremos definiendo las dos bases principales: el concepto de atributo y de dominio.

Definición 5.1 (Atributo)**.** *Un atributo a es un nombre único empleado para identificar una determinada propiedad de una determinada clase de entidades.*

Denominaremos $\mathbb{A}$ *al conjunto de todos los posibles atributos.*

Ejemplo 5.1. *Dada una determinada clase de entidades, supongamos los trabajadores de una empresa, las propiedades asociadas a cada una de las entidades de esta clase podrían ser su nombre, edad, número de teléfono, fecha del contrato y sueldo.*

Las anteriores propiedades podrían estar identificadas por los siguientes nombres únicos:

- *Nombre:* nombre.
- *Edad:* edad.
- *Número de teléfono:* tlf.
- *Fecha de contrato:* fcontrato.
- *Sueldo:* sueldo.

Al conjunto de atributos de una entidad que representa a un empleado, se denominará A_{emp}*. Este está definido, según lo anterior, de la forma que muestra la Ecuación 5.1.*

$$A_{emp} = \{\text{nombre}, \text{edad}, \text{tlf}, \text{fcontrato}, \text{sueldo}\} \tag{5.1}$$

Identificadas las propiedades de las entidades, es necesario definir el conjunto de datos sobre las que éstas pueden adoptar valores. Para ello se define el concepto de *dominio*.

Definición 5.2 (Dominio)**.** *Un dominio D es un conjunto de valores homogéneos caracterizados cada uno de ellos por un nombre único.*

Denominaremos $\mathbb{D}$ al conjunto de todos los posibles dominios.

En la anterior definición, el término *homogéneo* se aplica en el sentido en que los valores de un mismo dominio son todos del mismo tipo. En definitiva, un dominio no agrupa a valores de distinta naturaleza.

Ejemplo 5.2. *Un ejemplo clásico de dominio es el conjunto de los números naturales, que notaremos como D_{nat}. El dominio D_{nat} es un conjunto de valores homogéneos, ya que está constituido por elementos del mismo tipo (todos son números, concretamente positivos y sin decimales) y cada uno de ellos está caracterizado por un nombre único que coincide con la forma en que se notan (ningún par de números naturales distintos se nota de la misma forma).*

Una vez definidos los dos conceptos base, definiremos la forma en que éstos quedan relacionados. En primer lugar definiremos el concepto de *Dominio de un atributo.*

Definición 5.3 (Dominio de un atributo)**.** *El dominio $D_a \in \mathbb{D}$ de un atributo $a \in \mathbb{A}$ es conjunto de los valores que puede tomar la propiedad que dicho atributo identifica.*

Ejemplo 5.3. *Los dominios de los atributos empleados en el Ejemplo 5.1 son los siguientes:*

- *Atributo* nombre*: El dominio D_{str}, compuesto por todas las posibles secuencias de caracteres alfabéticos.*
- *Atributo* edad*: El dominio D_{nat} definido en el Ejemplo 5.2.*
- *Atributo* tlf*: El dominio D_{tlf}, compuesto por todas las posibles secuencias de nueve cifras.*
- *Atributo* fcontrato*: El dominio D_{fecha} compuesto por las posibles fechas dentro del calendario gregoriano.*
- *Atributo* sueldo*: El dominio D_{mon}, compuesto por todos los números reales positivos con dos decimales, que se corresponden con cantidades monetarias.*

Definida la relación entre atributos y dominios mediante el anterior concepto, será necesario ahora una manera formal de expresar y fijar dicho emparejamiento. Para ello se define la *Función dominio.*

Definición 5.4 (Función dominio)**.** *Sea $A \subset \mathbb{A}$, $A = \{a_1, a_2, ..., a_n\}$, $a_i \in \mathbb{A}\ \forall i \in [1, n]$, un conjunto de atributos. Sea $\mathbb{D}_A \subset \mathbb{D}$, $\mathbb{D}_A = \{D_{a_1}, D_{a_2}, ..., D_{a_n}\}$, $D_i \in \mathbb{D}\ \forall i \in [1, n]$ el conjunto de los dominios de los atributos en A de tal forma que D_{a_i} es el dominio asociado con el atributo $a_i\ \forall i \in [1, n]$.*

Una función dominio dom es una función de la forma $dom : A \mapsto \mathbb{D}_A$, tal que para cada $a_i \in A$, $dom(a_i) = D_{a_i}$, siendo D_{a_i} es el dominio del atributo a_i.

Atributo	Valor
nombre	"Alberto Gutiérrez"
edad	39
tlf	999555555
fcontrato	16-3-2002
sueldo	38535.54

Tabla 5.1: Ejemplo de la definición en forma tabular de una tupla

Ejemplo 5.4. *La función dominio correspondiente al emparejamiento de atributos y dominios detallados en el Ejemplo 5.3 es una función de la forma* $dom_{emp} : A_{emp} \mapsto \{D_{str}, D_{nat}, D_{tlf}, D_{fecha}, D_{mon}\}$ *y está definida según la Ecuación 5.2.*

$$dom_{emp}(a) = \begin{cases} D_{str} & si\ a = \text{nombre} \\ D_{nat} & si\ a = \text{edad} \\ D_{tlf} & si\ a = \text{tlf} \\ D_{fecha} & si\ a = \text{fcontrato} \\ D_{mon} & si\ a = \text{sueldo} \end{cases} \quad \forall a \in A_{emp} \tag{5.2}$$

Finalmente, definidos todos los elementos anteriores, definiremos el concepto de *tupla*. Este elemento permite representar a una entidad, o una relación entre entidades, mediante la agrupación en una sola estructura de los datos relativos a ésta.

Definición 5.5 (Tupla)**.** *Sea* $A \subseteq \mathbb{A}$ *el conjunto de atributos que identifica las propiedades de una determinada clase de entidades. Sea dom una función de dominio para los atributos en A.*

Una tupla t es una función de la forma $t : A \mapsto dom(A)$*, tal que cumple la restricción* $\forall a \in A, t(a) \in dom(a)$*. En lo anterior,* $dom(A)$ *está definido como* $dom(A) := \bigcup_{a \in A} dom(a)$*.*

La restricción impuesta a las tuplas en la definición anterior, $\forall a \in A, t(a) \in dom(a)$, asegura que, a pesar de que el codominio de t es $dom(A)$, a cada atributo $a \in A$ le corresponde exclusivamente un valor de su propio dominio $dom(a)$.

Ejemplo 5.5. *Siguiendo la temática de los anteriores ejemplos, partiendo de un conjunto de atributos como el definido en el Ejemplo 5.1 y la función de dominio* dom_{emp} *definida en el Ejemplo 5.4, podríamos definir una tupla* $t : A_{emp} \mapsto dom_{emp}(A_{emp})$ *de la forma que se muestra en la Ecuación 5.3. Obviamente, la definición anterior se puede expresar en forma de tabla, tal y como se muestra en la Tabla 5.1.*

$$t(a) = \begin{cases} \textit{"Alberto Gutierrez"} & si\ a = \text{nombre} \\ 39 & si\ a = \text{edad} \\ 999555555 & si\ a = \text{tlf} \\ 16-3-2002 & si\ a = \text{fcontrato} \\ 38535.54 & si\ a = \text{sueldo} \end{cases} \tag{5.3}$$

Será de utilidad, como se verá en posteriores capítulos, definir el conjunto de las posibles tuplas que se pueden expresar para un conjunto de atributos y

una función de dominio determinados. A este tipo de tuplas las denominaremos *tuplas compatibles* y al conjunto que las contiene *conjunto de tuplas compatibles.* Este conjunto se define formalmente como se indica a continuación.

Definición 5.6 (Conjunto de tuplas compatibles)**.** *Sea A un conjunto de atributos. Sea dom una función de dominio de la forma $dom : A \mapsto \mathbb{D}$. Definimos el conjunto de tuplas compatibles basadas en A y dom, notado como $\mathbb{T}_{(A,dom)}$, según se indica en la Ecuación 5.4.*

$$\mathbb{T}_{(A,dom)} = \{t : A \mapsto dom(A) | \forall a \in A, t(a) \in dom(a)\} \quad \forall A \subset \mathbb{A}, \forall dom : A \mapsto \mathbb{D} \quad (5.4)$$

5.3. Tipos de dominios para una base de datos objeto-multirrelacional difusa

Como se indicó en la introducción, una base de datos objeto-relacional difusa ha de permitir almacenar datos de naturaleza imperfecta, no atómica y una combinación de ambas naturalezas. A este tipo de datos los denominaremos *datos complejos.*

En la presente propuesta se considerará, dentro de la primera categoría, aquellos datos de naturaleza difusa afectados por incertidumbre, que denominaremos *datos inciertos*, y aquellos afectados por imprecisión, que denominaremos *datos imprecisos.* Con respecto a los datos no atómicos, se considerarán los *datos compuestos*, datos que se forman como un conjunto de tamaño fijo de datos de igual o distinto tipo. Finalmente, dentro de la categoría de datos de naturaleza difusa y no atómica se considerarán los *multiconjuntos difusos*, colecciones de tamaño variable de datos de un mismo tipo en la que la participación de cada elemento está cuantificada y cualificada.

El presente apartado se dedica a definir formalmente el concepto de *dominio complejo* y cada uno de los dominos en los que se encuadran los anteriores tipos de datos.

5.3.1. Dominios atómicos

En lo que respecta a sus dominios, las bases de datos relacionales se caracterizan por admitir únicamente *dominios atómicos* como dominios de sus atributos, restricción impuesta por su primera formal normal. En cambio, la característica principal, en lo que se refiere a dominios, de las bases de datos objeto-relacionales es que éstas admiten dominios que no sean atómicos. Los valores de cualquier dominio no atómico son una composición de otros valores de dominios que podrían ser, a su vez, ser no atómicos. Esta composición recursiva ha de tener su base en los *dominios atómicos.*

Comenzaremos este apartado definiendo el concepto de *dominio atómico*, en el que se basan el resto de conceptos que se definirán en esta sección. Una definición formal de éste concepto, es la siguiente:

Definición 5.7 (Dominios atómicos)**.** *Sea $D \in \mathbb{D}$ un dominio. Diremos que D es un* dominio atómico*, $D \in \mathbb{S}$ (donde $\mathbb{S}$ denota el subconjunto de dominios que son atómicos, $\mathbb{S} \subset \mathbb{D}$), si, y solo sí, sus valores asociados son* atómicos*.*

Un valor atómico *es tal que no puede ser dividido en unidades más pequeñas de tal forma que estas tengan significado en el universo que se modela.*

La determinación de la atomicidad de un dominio, y por extensión de un valor, es algo controvertida. Para evitar en lo posible la controversia se emplea como referencia el universo que se modela. Por tanto, un mismo dominio puede resultar atómico o no atómico en función del ámbito en que se considera. Todo esto se ilustra en los siguientes ejemplos:

Ejemplo 5.6. *Como ejemplo de dominio atómico se podría considerar el dominio de los números naturales D_{nat} mencionado anteriormente en el Ejemplo 5.2. Para identificar el universo que se está modelando, consideraremos que este dominio es el dominio del atributo edad, tal y como se indica en el Ejemplo 5.3, atributo que identifica la edad de un trabajador de una determinada empresa.*

A pesar de que los valores que forman parte de este dominio están compuestos por cifras, éstas por si solas no tienen significado en el universo que se modela, ya que necesitamos la secuencia de cifras para determinar la edad de un trabajador.

Ejemplo 5.7. *Un ejemplo menos obvio podría ser el dominio de las fechas D_{fecha} definido en el Ejemplo 5.3. Los datos de este dominio están compuestos por un componente que indica el día de la fecha, otro para el mes y, finalmente, otro para indicar el año de ésta. Siendo, además, todos estos componentes números naturales dentro de determinados rangos, por lo que a su vez (como se indica en el ejemplo anterior) están compuestos por cifras.*

Se podría argumentar que cada uno de los citados componentes tiene significado propio, sin necesidad de que todos ellos sean agrupados. Por ejemplo, el componente que indica el año nos ofrece información que podría interpretarse de forma aislada. Una vez más, si empleamos como referencia el universo que se modela (recordemos que en este caso se refiere a la fecha en la que el trabajador inició su contrato), queda claro que cada uno de los componentes aislados no proporciona la información suficiente y necesaria para determinar con exactitud la fecha de inicio de contrato del trabajador. Por tanto, el dominio de las fechas (en el ámbito de este ejemplo) se considera atómico.

5.3.2. Dominios complejos

Los dominios que hacen que una base de datos objeto-relacional difusa se diferencie de las propuestas clásicas, los dominios cuyos valores son difusos y/o no atómicos, se encuadran dentro de una categoría que denominaremos *dominios complejos*. Esta categoría se define de la siguiente forma:

Definición 5.8 (Dominio complejo)**.** *Sea D un dominio, $D \in \mathbb{D}$.*

Se dice que D es un dominio complejo, $D \in \mathbb{C}$, donde $\mathbb{C}$ es la clase de dominios complejos, si, y solo si, es un dominio de los siguientes tipos:

- *Incierto: Dominios atómicos en los que sus valores están afectados por incertidumbre. Los valores de este tipo de dominios son la conjunción de un valor de un dominio complejo subyacente determinado y un grado que indica la certidumbre asociada a dicho valor.*

- *Impreciso: Dominios atómicos en los cuales sus valores están afectados por imprecisión. Los valores de estos dominios son distribuciones de posibilidad sobre los valores de un dominio complejo subyacente. Estas distribuciones de posibilidad permiten modelar la imprecisión asociada a un valor determinado.*

- *Compuesto: Dominios no atómicos formados por conjuntos de tamaño fijo de elementos pertenecientes a dominios complejos subyacentes posiblemente diferentes. Los valores de este tipo de dominio son tuplas del mismo tipo, es decir, tuplas definidas sobre un conjunto atributos y una función dominio preestablecidos, comunes y fijos para todos los elementos del dominio.*

- *Multiconjuntivo difuso: Dominios no atómicos en los que los valores que los forman son multiconjuntos difusos de elementos que pertenecen a un determinado dominio complejo subyacente.*

- *Básico: Dominios atómicos cuyos valores no están afectados por imprecisión ni incertidumbre.*

Se ha de hacer notar que el último tipo de dominios que pertenecen a la categoría de dominios complejos son los dominios básicos. Los valores de un dominio básico no son elementos compuestos y por tanto, de forma estricta, un dominio básico no debería considerarse como dominio complejo. A pesar de ello, se consideran los dominios básicos como un tipo de dominio complejo. Esto es debido a la definición recursiva de los anteriores tipos de dominios complejos, que hace necesario incluir un tipo de dominio complejo que suponga un caso base sobre el que se puedan construir el resto de casos.

5.3.3. Tipos de dominios complejos

En el siguiente apartado se definirán de manera más formal cada uno de los tipos de dominios complejos que se consideran en la definición anterior.

5.3.3.1. Dominios inciertos

El primero de los dominios complejos que describiremos será el de los *dominios inciertos*. En ciertas situaciones no se conoce con total certeza el valor de un atributo determinado. Este tipo de dominios ofrece la posibilidad de emplear valores afectados por incertidumbre, esto es, valores de un determinado dominio subyacente asociados a un determinado nivel de certeza.

Ejemplo 5.8. *Un ejemplo de dominio de este tipo podría ser el dominio de las confirmaciones de asistencia a una reunión, que denominaremos* D_{conf}*. Este dominio se podría emplear como valor de un atributo* asiste *asociado a las tuplas que representen los convocados a una reunión.*

En el caso clásico, el dominio D_{conf} *estaría compuesto exclusivamente por los valores* sí *y* no*. En el contexto del ejemplo, no se conoce con total certeza si uno de los convocados a una reunión asistirá o no a ésta. En estas situaciones, los sistemas clásicos no permiten indicar la certeza del dato, por lo que se deberá asumir una de las posibles alternativas dentro del conjunto de valores* $\{sí, no\}$*.*

Esto supone cierta pérdida de información (la certeza del dato) que puede inducir a resultados erróneos a un sistema que trabaje sobre dicha información.

En cambio, en una base de datos objeto-relacional difusa, se podría emplear toda la información disponible (el valor y su certidumbre asociada) de tal forma que ésta se pueda emplear en el procesamiento de los datos para ofrecer resultados más certeros.

Una definición formal de lo que se considera un dominio incierto en el presente modelo es la siguiente:

Definición 5.9 (Dominio Incierto)**.** *Sea $D \in \mathbb{D}$ un dominio complejo. Un dominio incierto U_D, de la clase de dominios inciertos $\mathbb{U}$, $U_D \in \mathbb{U}$, es un dominio compuesto por valores inciertos de la forma $\langle \alpha, d \rangle$, $\alpha \in (0,1]$ y $d \in D$.*

Formalmente, la definición de U_D se muestra en la Ecuación 5.5.

$$U_D = \{\langle \alpha, d \rangle | \alpha \in (0,1], d \in D\} \tag{5.5}$$

$\langle \alpha, d \rangle$ representa un valor d con un grado de certidumbre asociada α. El valor α será 1 cuando la certidumbre sea máxima, reduciéndose ésta cuanto más cercano sea el valor a 0, caso extremo (y excluido) en que no se poseería certidumbre alguna sobre el valor que el valor representado por $\langle \alpha, d \rangle$ sea d.

Por motivos simplicidad en la notación , denominaremos al dominio U_D simplemente como U. Por el mismo motivo, denominaremos a D como el dominio subyacente de *U y se notará como $sub(U)$, $sub(U) = D$.*

En ocasiones, como se verá en los siguientes capítulos, será necesario acceder a uno de los componentes de un valor incierto. Para ello, dado un valor incierto $u = \langle \alpha, d \rangle$, emplearemos la función $cert(u)$ para obtener el grado α de certidumbre asociado a un valor incierto u y $val(u)$ para el valor d asociado al mismo. A este último valor, lo denominaremos *valor base*. Definiremos formalmente estos conceptos como se indica a continuación.

Definición 5.10 (Grado de certidumbre de un valor incierto)**.** *Sea U un dominio incierto. Sea u un valor de U. El grado de certidumbre del valor U es el resultado de aplicar sobre el mismo la función cert. Dicha función es de la forma $cert : \{u | (\exists U \in \mathbb{U} | u \in U))\} \mapsto [0,1]$ y está definida según se indica en la Ecuación 5.6.*

$$cert(u) = \alpha \qquad \forall u = \langle \alpha, d \rangle | ((\exists U \in \mathbb{U} | u \in U) \wedge \alpha \in [0,1] \wedge d \in sub(U)) \tag{5.6}$$

Definición 5.11 (Valor base de un valor incierto)**.** *Sea U un dominio incierto. Sea u un valor de U. El valor base del valor U es el resultado de aplicar sobre el mismo la función val. Dicha función es de la forma $val : \{u | (\exists U \in \mathbb{U} | u \in U))\} \mapsto \{u | (\exists D \in \mathbb{D} | d \in D))\}$ y está definida según se indica en la Ecuación 5.7.*

$$val(u) = d \qquad \forall u = \langle \alpha, d \rangle | ((\exists U \in \mathbb{U} | u \in U) \wedge \alpha \in [0,1] \wedge d \in sub(U)) \tag{5.7}$$

Ejemplo 5.9. *Siguiendo el Ejemplo 5.8 y la definición anterior, se podría definir el dominio D_{conf} como un dominio incierto, $D_{conf} \in \mathbb{U}$, cuyo dominio subyacente es dominio booleano $D_{bool} = \{sí, no\}$, $sub(D_{conf}) = D_{bool}$.*

Certidumbre	Referencia
1	Sin duda
0.75	Seguramente
0.5	Posiblemente
0.25	Poco posible

Tabla 5.2: Etiquetas de referencia para los niveles de certidumbre

Los valores de este dominio tendrán, por tanto, la forma $\langle \alpha, d \rangle$*, donde* $d \in \{sí, no\}$*, de tal manera que se pueda indicar que se conoce que un convocado asistirá o no a la reunión con un determinado nivel de certeza* α*.*

Para facilitar la interpretación de los niveles de certeza en este dominio los usuarios podrían emplear la siguientes referencias mostradas en la Tabla 5.2.

Basándonos en la referencias descritas, el dominio D_{conf} *contendría a valores como, por ejemplo,* $\langle 1, sí \rangle$ *que serían interpretados por los usuarios como "el convocado asistirá sin duda a la reunión",* $\langle 0.75, sí \rangle$ *(el convocado asistirá seguramente a la reunión),* $\langle 0.5, sí \rangle$ *(el convocado posiblemente asistirá a la reunión), etcétera.*

Obviamente, el dominio también contendrá las contrapartidas negativas a los anteriores ejemplos como, por ejemplo, $\langle 0.5, no \rangle$ *(el convocado posiblemente no asistirá a la reunión).*

5.3.3.2. Dominios imprecisos

El segundo tipo de dominios que será definido será el de los *dominios imprecisos*. Estos tipos de dominios contienen valores afectados por un determinado grado de imprecisión. Este tipo de valores se da en ocasiones en que no se conoce el valor exacto de una determinada propiedad pero se posee una aproximación de dicho valor. Esta aproximación estará compuesta por un conjunto de semántica disyuntiva de los posibles valores para la propiedad, de los cuales uno de ellos será el verdadero valor para ésta.

Ejemplo 5.10. *Un ejemplo de un dominio de tipo impreciso, siguiendo con el ejemplo empleado anteriormente de los asistentes a una reunión, podría ser el dominio del* número *de asistentes a una reunión, que denominaremos como* D_{asist}*. Este dominio contiene como valores las posibles cifras de asistentes a una reunión. Esta cifras pueden ser imprecisas ya que, como recordaremos de ejemplos anteriores, es posible que no se conozca de forma certera si cada uno de los convocados asistirá o no a la reunión.*

Usando de nuevo la referencia del caso clásico, este dominio se podrían corresponder con el de los números naturales. Una vez más, los sistemas clásicos no permiten representar toda la información de que se dispone, obligando al usuario a suministrar un dato preciso derivado de la información aproximada de que se dispone.

Formalmente, definiremos el concepto de *dominio impreciso* como se indica a continuación.

Definición 5.12 (Dominio Impreciso)**.** *Sea* $D \in \mathbb{C}$ *un dominio complejo. Un dominio impreciso* I_D*, perteneciente a la clase de dominios imprecisos* $\mathbb{I}$*,* $I_D \in \mathbb{I}$*,*

es un dominio compuesto por valores imprecisos modelados como distribuciones de posibilidad sobre el dominio subyacente D.

Formalmente, se define el dominio I_D como se indica en la Ecuación 5.8, donde $\widetilde{P}(D)$ es el conjunto difuso de las partes de D *definido en la Ecuación 5.9.*

$$I_D = \widetilde{P}(D) \tag{5.8}$$

$$\widetilde{P}(D) = \{\widetilde{D} \mid \widetilde{D} : D \mapsto [0,1]\} \tag{5.9}$$

Por motivos de simplicidad, se empleará la misma notación abreviada que para el caso anterior de los dominios inciertos. De esta forma D_I se notará exclusivamente por I y D se denominará como el dominio subyacente de I, notado como $sub(I)$, $sub(I) = D$.

Ejemplo 5.11. *El dominio D_{asist}, descrito en el ejemplo anterior, se definiría siguiendo las directrices marcadas anteriormente como un dominio impreciso, $D_{asist} \in \mathbb{I}$, cuyo dominio subyacente sería el dominio de los números naturales, $sub(D_{asist}) = D_{nat}$. Este dominio representará cantidades imprecisas como distribuciones de posibilidad.*

Consideremos que partimos de las siguientes confirmaciones de asistencia a una reunión: se conoce que 3 de los convocados asistirá sin duda a la reunión (valores de confirmación $\langle 1, s\hat{i}\rangle$), 4 asistirán seguramente (valores $\langle 0.75, s\hat{i}\rangle$), 2 posiblemente no asistirán (valores $\langle 0.5, no\rangle$) y es poco posible que un último asista (valor $\langle 0.25, no\rangle$).

Teniendo en cuenta los datos anteriores podríamos decir, sobre el número de asistentes a la reunión, que sin duda asistirán al menos 3 personas a la misma, que seguramente asistirán 7 personas a la reunión, que posiblemente asistirán 9 y que es poco posible que asistan 10 de los convocados. Esta cantidad imprecisa se expresaría en el dominio impreciso D_{asist} como la distribución de posibilidad $\{1/3, 0.75/7, 0.5/9, 0.25/10\}$.

5.3.3.3. Dominios compuestos

Un tercer tipo de dominio complejo sería el de los *dominios compuestos*. En ciertas aplicaciones es necesario manejar ciertos datos compuestos de otros datos más simples. Estos tipos de datos se crean para agrupar en una sola unidad a otros datos simples que, poseyendo o no cada uno de los componentes semántica propia dentro del universo modelado, su unión posee una semántica propia, distinta a la de cada uno de sus componentes de forma aislada. La semántica de la agrupación es, por tanto, conjuntiva.

El tipo de agrupaciones que soportan los dominios de este tipo se restringe a aquellas que son de un tamaño fijo y cuyos componentes pueden ser de naturalezas heterogéneas.

Ejemplo 5.12. *Un ejemplo clásico en que la agrupación de varios datos posee una semántica propia es el de las direcciones postales.*

Las direcciones postales son datos compuestos de datos más simples, como el nombre de la vía, el tipo de ésta, el número, código postal y localidad. Algunos de los componentes de una dirección postal pueden tener significado propio en el universo en que se modeló, como por ejemplo puede ser el nombre de la localidad

o el nombre de la vía. A pesar de ello, la agregación de éstos en una única entidad crea un dato con una semántica propia y una información más completa.

Este tipo de datos se representan en el modelo relacional incluyendo cada uno de sus componentes como un atributo dentro de la relación en la que estos datos están incluidos. Esta práctica, sin embargo, supone una pérdida de información en el sentido en que se pierde la interrelación de cada uno de los componentes, dejándose a las aplicaciones o usuarios la responsabilidad de recomponer dicha relación reconstruyendo la agrupación de los mismos. Esta práctica, en el caso de datos de gran complejidad, como en el caso de sistemas CAD o de *office automation* se ha demostrado inapropiada y contraproducente.

En el marco de la presente propuesta, definiremos de manera formal un dominio de este tipo de la siguiente forma:

Definición 5.13 (Dominios Compuestos)**.** *Sea $A \subset \mathbb{A}$ un conjunto de atributos y dom una función dominio de la forma $dom : A \mapsto \mathbb{D}_A$, donde $\mathbb{D}_A$, $\mathbb{D}_A \subset \mathbb{D}$, es el conjunto de los dominios de los atributos de A.*

Un dominio compuesto $O_{(A,dom)}$, de la clase de dominios compuestos $\mathbb{O}$, $O_{(A,dom)} \in \mathbb{O}$, es un dominio cuyos valores son las posibles tuplas que se pueden formar sobre el conjunto de los dominios de los atributos en A según la función de dominio dom.

Formalmente, un dominio compuesto $O_{(A,dom)}$ se define como se indica la Ecuación 5.10.

$$O_{(A,dom)} = \{o | o : A \mapsto dom(A) \wedge \forall a \in A, o(a) \in dom(a)\} \tag{5.10}$$

Por simplicidad, de manera similar a los casos anteriores, denominaremos a un dominio compuesto $O_{(A,dom)}$ simplemente como O. El conjunto A se denominará como los atributos del dominio *O y se notará como $att(O)$. De la misma forma, la función dominio dom se notará como dom_O y se denominará como* la función dominio de *O.*

Finalmente, siguiendo con la terminología de los tipos de dominios anteriores, denominaremos al conjunto de los dominios de los atributos de un dominio compuesto O, $dom_O(att(O))$, como el conjunto de los dominios subyacentes *de O, notado como $sub(O)$, $sub(O) = dom_O(att(O))$. Nótese que, a diferencia de los tipos de dominio anteriores, $sub(O)$ es un conjunto de dominios y no un único dominio.*

Ejemplo 5.13. *Continuando con el ejemplo anterior, podemos definir el dominio compuesto D_{dir} cuyos valores son direcciones postales. Los atributos de este dominio, $att(D_{dir})$, serán el conjunto* $\{\text{nombreVia}, \text{tipoVia}, \text{numero}, \text{cp}, \text{localidad}\}$ *que representan respectivamente el nombre de la vía, el tipo de ésta, el número del inmueble, el código postal y el nombre de la localidad. La función dominio de D_{dir}, $dom_{D_{dir}}$, se representa de forma tabular en la Tabla 5.3. De estos dominios, D_{str} y D_{nat} fueron definidos en los ejemplos 5.3 y 5.2 respectivamente, y D_{cp} es un dominio compuesto por secuencias de 5 cifras.*

Un ejemplo de un valor del dominio D_{dir} podría ser la tupla ("Principal", "Calle", 23, 45231, "Granada").

5.3.3.4. Dominios multiconjuntivos difusos

El cuarto tipo de dominios complejos es el de los dominios *multiconjuntivos difusos*. En ciertas ocasiones es necesario manejar agrupaciones, de tamaño

Atributo	Dominio
nombreVia	D_{str}
tipoVia	D_{str}
numero	D_{nat}
cp	D_{cp}
localidad	D_{str}

Tabla 5.3: Dominios asociados a los atributos del dominio compuesto D_{dir}

variable, de valores de un determinado tipo. Una vez más, remitiéndonos a la semántica dentro del dominio de aplicación, estas agrupaciones de valores poseen semántica propia, tenga o no semántica propia el dominio de los componentes de la agrupación. Estas agrupaciones se pueden modelar empleando multiconjuntos difusos, lo que permite cuantificar y cualificar (empleando cardinalidades difusas) la participación de cada uno de los elementos en la agrupación.

Las agrupaciones de tamaño variable descritas anteriormente se pueden equiparar a las *relaciones anidadas* propias de los modelos NF^2 (introducidos en el apartado 2.2.3.3 del Capítulo 2). Esta equiparación se realiza con la salvedad de que, en lugar de relaciones simples, la presente propuesta considera multirrelaciones difusas, dando así la posibilidad de expresar información cuantitativa y cualitativa adicional sobre cada uno de sus elementos.

Se ha de remarcar que los valores de este tipo de dominio son agrupaciones de valores *homogéneos* y de tamaño *variable*, en contraste con los valores de los dominios compuestos, conjuntos de tamaño fijo y de elementos de naturaleza heterogénea. A diferencia de los valores los valores de tipo impreciso, la semántica de la agrupación es conjuntiva, es decir, cada uno de los valores de la agrupación es una parte del valor del atributo correspondiente.

Ejemplo 5.14. *Siguiendo el contexto de los anteriores ejemplos, un ejemplo de dominio de tipo multiconjuntivo difuso es el dominio cuyos valores indican los idiomas que hablan los participantes en una reunión. Estos valores serán conjuntivos ya que son varios los idiomas que los participantes pueden hablar. La participación de cada elemento, cada idioma, puede ser cuantificada, indicándose así el número de participantes que puede hablar cada idioma. También, la participación de cada elemento puede ser cualificada, indicando así el nivel que cada participante tiene en el uso del idioma.*

Al igual que ocurre en el caso de los valores de dominios complejos, los valores de este tipo de dominios se representan en el modelo relacional clásico de forma separada. Supongamos que se parte de una relación r que posee un atributo a cuyos valores son conjuntos de tamaño variable de valores de un determinado dominio D. Para representar estos datos en el modelo relacional clásico, se almacenarán los valores correspondientes al atributo a en una relación separada, que se podría denominar s, y se transformará r en una relación r' en la que se habrá eliminado el atributo a. En la relación s habrá un atributo a' que tendrá a D como dominio asociado, de tal forma que los elementos de los valores conjuntivos originales del atributo a de r se representarán como tuplas en s. Adicionalmente, la relación s deberá disponer de un conjunto de atributos F que se corresponderá con el conjunto de atributos K que forman la clave

primaria en r'. El conjunto de atributos F hará la veces de clave externa de tal forma que, al reunir r' con s se reconstruya la información contenida en la relación original r. De esta forma, las tuplas en s que representan los elementos del valor conjuntivo del atributo a de una determinada tupla t en r se reunirán con la tupla t' en r' que deriva de la tupla t en r.

De la misma forma, al igual que ocurre en el caso de los dominios complejos, la anterior técnica deja la responsabilidad de reconstruir los valores conjuntivos en manos de los usuarios o aplicaciones que trabajen sobre éstos. Además del inconveniente anterior, la operación de reconstrucción requiere emplear operaciones de reunión entre relaciones, lo que implica que en cuanto crezca la complejidad de la información las operaciones de reconstrucción se volverán prohibitivamente ineficientes. De ahí el interés de que el propio modelo soporte el tipo de datos conjuntivos.

En el marco de la presente propuesta, definiremos a los dominios multiconjuntivos difusos como sigue:

Definición 5.14 (Dominios Multiconjuntivo Difusos)**.** *Sea $D \in \mathbb{C}$ un dominio complejo. Un dominio multiconjuntivo difuso M_D, perteneciente a la clase de dominios multiconjuntivos difusos $\mathbb{M}$, $M_D \in \mathbb{M}$, es un dominio tal que sus valores son los posibles multiconjuntos difusos que se pueden crear sobre valores del dominio D.*

Formalmente, este tipo de dominios se define como se indica en la Ecuación 5.11. En dicha ecuación, $\widehat{\mathcal{P}}(D)$ denota el multiconjunto difuso de las partes *del conjunto D, definido como se indica en la Ecuación 5.12.*

$$M_D = \widehat{\mathcal{P}}(D), D \in \mathbb{C} \tag{5.11}$$

$$\widehat{\mathcal{P}}(D) = \{\widehat{D} \mid \widehat{D} : D \mapsto \mathbb{N}_f\} \tag{5.12}$$

Una vez más, por motivos de simplicidad, se hará uso la misma notación abreviada que para los casos de dominios inciertos e imprecisos. Por tanto, M_D se notará simplemente como M, y D se denominará como el dominio subyacente de M, notándose como $sub(M)$.

Ejemplo 5.15. *Retomando el ejemplo anterior, definiremos D_{spk} como el dominio de los idiomas hablados por los asistentes a la una reunión. El dominio D_{spk} es un dominio multiconjuntivo difuso, $D_{spk} \in \mathbb{M}$, cuyo dominio subyacente $dom(D_{spk})$ es el dominio de los diferentes idiomas existentes $D_{lang} = \{Español, Inglés, Alemán, Francés, Italiano \ldots\}$, $dom(D_{spk}) = D_{lang}$.*

Supongamos que entre los participantes a una reunión, la cantidad de ellos que pueden hablar un idioma y el nivel con que pueden hablarlo se corresponde con lo indicado en la Tabla 5.4. Supongamos también que el nivel de dominio del idioma de un asistente se expresa según los grados de pertenencia de dicho idioma al multiconjunto de los idiomas hablados en una reunión según se muestra en la Tabla 5.5.

La situación descrita anteriormente se expresaría mediante un único valor del dominio D_{spk}, esto es, un multiconjunto difuso de idiomas. En concreto, dicho multiconjunto difuso se corresponde con el presentado en la Ecuación 5.13.

	Nivel			
Idioma	Nativo	Alto	Medio	Bajo
Español	2	1	3	1
Inglés	1	1	4	3
Alemán	1	0	1	1
Francés	1	0	2	2
Italiano	0	0	1	5

Tabla 5.4: Idiomas hablados por los asistentes a una reunión

Nivel	Grado de pertenencia
Nativo	1.0
Alto	0.75
Medio	0.5
Bajo	0.25

Tabla 5.5: Asignación del grado de pertenencia al dominio D_{spk} para cada nivel de conocimiento de un idioma

$$\begin{array}{ll}[& \{\ldots, 1.0/2, 0.75/3, \ldots, 0.5/6, 0.25/7\} * \textit{Español}, \\ & \{1.0/0, 1.0/1, 0.75/2, \ldots, 0.5/6, \ldots, 0.25/9\} * \textit{Inglés}, \\ & \{1.0/0, 1.0/1, 0.5/2, 0.25/3\} * \textit{Alemán}, \\ & \{1.0/0, 1.0/1, 0.5/2, 0.5/3, 0.25/4, 0.25/5\} * \textit{Francés}, \\ & \{1.0/0, 0.5/1, \ldots, 0.25/6\} * \textit{Italiano}\ ] \end{array} \tag{5.13}$$

5.3.3.5. Dominios básicos

El quinto y último tipo de dominios complejos es el de los dominios *básicos*. Estos tipos de dominios son el último eslabón de la cadena de composición recursiva que se emplea para definir un dominio complejo, ya que éste tipo de dominios son dominios atómicos *crisp* (no afectados por imprecisión ni incertidumbre) y por tanto su definición no requiere otros dominios subyacentes.

Formalmente, los dominios de este tipo se definen de la siguiente forma:

Definición 5.15 (Dominios Básicos)**.** *Sea D un dominio, $D \in \mathbb{D}$. Se dice que D es un dominio de tipo básico, $D \in \mathbb{B}$ si, y solo sí, D es un dominio atómico y no es un tipo de dominio incierto ni impreciso.*

La definición anterior se expresa matemáticamente como se indica en la Ecuación 5.14.

$$D \in \mathbb{B} \iff D \in (\mathbb{S} - (\mathbb{U} \cup \mathbb{I})) \tag{5.14}$$

Ejemplo 5.16. *Un ejemplo de dominio de este tipo puede ser el dominio D_{nat} descrito en el ejemplo 5.2.*

El dominio D_{nat} es un dominio atómico, como ya se discutió en el Ejemplo 5.6. Además, los valores de este dominio no están afectados por incertidumbre (un valor de este dominio no expresa información añadida sobre la certidumbre

del valor que representa) ni imprecisión (un valor de este dominio representa un valor absolutamente preciso).

5.3.4. Dominios complejos acíclicos

Como se ha indicado anteriormente, los dominios complejos son dominios cuyos valores son composiciones de valores de otros dominios que denominamos dominios subyacentes. Estos dominios subyacentes pueden a su vez ser dominios complejos. Al estar los valores de un dominio complejo compuestos por otros valores de dominios complejos, se produce una definición recursiva de este tipo de dominios. Esta recursión tiene su caso base en los dominios básicos.

Dado un dominio complejo, podríamos determinar el conjunto de dominios que intervienen en toda su definición recursiva. A este conjunto de dominios lo denominaremos *conjunto de dominios descendientes* y estará definido como sigue:

Definición 5.16 (Conjunto de dominios descendientes)**.** *Sea D un dominio complejo, $D \in \mathbb{C}$. Su* conjunto de dominios descendientes*, notado como $des(D)$, se define como se indica en la Ecuación 5.15.*

$$des(D) = \begin{cases} \emptyset & si\ D \in \mathbb{B} \\ dom_D(att(D)) \cup \left(\cup_{a_i \in att(D)} des(dom_D(a_i))\right) & si\ D \in \mathbb{O} \\ \{sub(D)\} \cup des(sub(D)) & si\ D \in \mathbb{M} \cup \mathbb{U} \cup \mathbb{I} \end{cases} \tag{5.15}$$

Hasta el momento, basándose en las definiciones de los distintos tipos de dominios complejos, nada impide que un determinado dominio complejo se defina en base así mismo, esto es, que un dominio complejo D, $D \in \mathbb{C}$, pertenezca al conjunto de sus descendientes $D \in des(D)$. Esta situación provocaría un ciclo en la definición de dicho dominio lo que significaría que los valores de dicho dominio podrían estar compuestos por un conjunto no finito de datos. Esto conduciría a una situación no deseable en la que los valores de este tipo de dominios no podrían ser representados por un conjunto finito de valores y por tanto no tratables por un sistema de gestión de bases de datos que se base en el presente modelo.

Para evitar esta situación definiremos la clase de dominios *complejos acíclicos*. Esta clase se define como sigue:

Definición 5.17 (Dominio complejo acíclico)**.** *Sea D un dominio complejo, $D \in \mathbb{C}$. Se dice que el dominio D* es acíclico *si, y solo sí, $\{D\} \cap des(D) = \emptyset$.*

Se ha de notar que la presente propuesta considera exclusivamente aquellos dominios complejos que son acíclicos. Por tanto, por simplicidad se empleará a lo largo del texto exclusivamente la denominación *dominio complejo* omitiendo el calificativo de acíclico y se notará al conjunto de dominios complejos acíclicos como $\mathbb{C}$, símbolo que anteriormente se ha empleado para notar el conjunto de dominios complejos y que de ahora en adelante se emplea restringiéndolo al conjunto de los dominios complejos acíclicos. De la misma forma, los subconjuntos de $\mathbb{C}$, $\mathbb{U}$, $\mathbb{I}$, $\mathbb{O}$ y $\mathbb{M}$, quedan igualmente restringidos.

5.4. Multirrelaciones complejas difusas

Una vez definida la base del presente modelo, los dominios complejos, la presente sección definirá la unidad de representación de información de éste. Al igual que en modelo relacional clásico la unidad de representación de información son las relaciones, en el presente modelo se define su análogo, las *multirrelaciones complejas difusas*.

Las relaciones clásicas agrupan la información relativa a una clase de entidades o relación entre las mismas. En ellas, cada una de sus tuplas representa a una entidad (o relación entre entidades). Cada tupla está compuesta por los datos (que considera el modelo) relativos a la entidad que ésta representa. Las multirrelaciones complejas difusas propuestas en este modelo tienen el mismo cometido aunque, a diferencia del modelo clásico, permiten añadir si se desea información adicional sobre la relación de pertenencia de cada entidad a la clase y expresar la certidumbre que se posee sobre los datos relativos a dicha entidad.

La relación de pertenencia de una tupla a una multirrelación compleja difusa podrá ser cuantificada. Se podrá, por tanto, indicar el número de veces que una determinada entidad participa en la relación de pertenencia a la multirrelación compleja difusa, a diferencia del modelo clásico en que el enfoque conjuntista impedía esta posibilidad. Esta característica aumenta la capacidad expresiva del modelo y lo acerca a las estructuras que manejan la mayor parte de los sistemas de gestión de bases de datos (SGBD): los multiconjuntos.

Adicionalmente, la relación de pertenencia de una tupla a una multirrelación compleja difusa podrá ser cualificada. En definitiva, será posible graduar cada una de las participaciones de una tupla en la multirrelación compleja difusa a la que pertenezca, permitiendo de esta forma matizar dicha relación de pertenencia, posibilidad que no se ofrece en los modelos clásicos. Esta posibilidad dota al modelo de las características de las bases de datos posibilísticas.

Finalmente, como se ha comentado anteriormente, se incorpora la posibilidad de especificar la certidumbre que se posee de que las propiedades de cada entidad representada en forma de tupla se correspondan con los valores de los atributos para dicha tupla. Esta última posibilidad dota al modelo de las características de ciertos tipos de modelos de bases de datos difusas.

Definiremos de manera formal una multirrelación compleja difusa de la siguiente manera:

Definición 5.18 (Multirrelación compleja difusa)**.** *Una* multirrelación compleja difusa *(MRCD) m es un par definido como se indica en la Ecuación 5.16. En dicha ecuación, $\mathcal{H}_m$ denota la* cabecera *de m y $\mathcal{B}_m$ el* cuerpo *de m.*

$$m = (\mathcal{H}_m, \mathcal{B}_m) \tag{5.16}$$

En la definición anterior se ha hecho uso de los conceptos de *cabecera* y *cuerpo* de una MRCD. Definiremos el concepto de *cabecera* de una MRCD como sigue:

Definición 5.19 (Cabecera de una multirrelación compleja difusa)**.** *La* cabecera de una multirrelación difusa *m, $\mathcal{H}_m$, es un par definido como se indica en la Ecuación 5.17.*

$$\mathcal{H}_m = (A_m, dom_m) \tag{5.17}$$

Los componentes del par anterior están definidos como sigue:

- A_m,$A_m \subseteq \mathbb{A}$, *es el* conjunto de atributos *de* m.
- $dom_m : A_m \mapsto \mathbb{D}_A$ *es la* función dominio *de* m. *Ésta empareja cada atributo* a_i *de* m, $a_i \in A_m$, *con su dominio complejo asociado* D_{a_i}, $D_{a_i} \in \mathbb{D}_A$, $\mathbb{D}_A \subset \mathbb{C}$.

La cabecera de una MRCD m contiene la información de la estructura de la MRCD. Esta información permanece invariable durante la existencia de la MRCD y está compuesta por el conjunto de atributos de m (conjunto A_m) y los dominios asociados a dichos atributos (indicado por la función de dominio dom_m).

El segundo elemento del par que define una MRCD es el cuerpo de ésta. Definiremos el concepto de *cuerpo* de una MRCD de la siguiente forma:

Definición 5.20 (Cuerpo de una multirrelación compleja difusa)**.** *El cuerpo* $\mathcal{B}_m$ *de una multirrelación compleja difusa* m *es un par definido según la Ecuación 5.18.*

$$\mathcal{B}_m = (T_m, U_m) \tag{5.18}$$

Los componentes del par que conforma $\mathcal{B}_m$ *se definen de la siguiente forma:*

- T_m *es el* multiconjunto difuso de las tuplas de m, *formalmente definido como se indica en la Ecuación 5.19. En dicha ecuación,* $\mathbb{T}_m$ *representa al* conjunto de las posibles tuplas de m, *definido éste como se indica en la Ecuación 5.20.*

$$T_m : \mathbb{T}_m \mapsto \mathbb{N}_f \tag{5.19}$$

$$\mathbb{T}_m = \{t_m | t_m : A_m \mapsto dom_m(A_m) \wedge t_m(a) \in dom_m(a) \forall a \in A_m\} \tag{5.20}$$

- U_m *es la función de incertidumbre de* m. *Esta función es de la forma* $U_m : T_m \mapsto (0, 1]$ *tal que para una determinada tupla* t *de* m, $t \in T_m$, $U_m(t)$ *indica el grado de certidumbre asociado al conjunto de los valores de los atributos de dicha tupla.*

El cuerpo de una MRCD m contiene la información relativa a las tuplas que forman parte de la misma. Esta información permite representar la cantidad de veces que una tupla determinada participa en m junto con la calidad o grado en que participa cada una de estas veces (indicado por la función de pertenencia de T_m), así como la certidumbre que se posee sobre el conjunto de valores de los atributos de una determinada tupla (indicado por la función de certidumbre U_m).

Ejemplo 5.17. *Supongamos que se desea representar en una base de datos la información relativa a las reservas de reuniones de trabajo asociadas a los eventos organizados en un centro de negocios. Se ha de tener en cuenta que los datos de dichas reservas se obtienen desde el departamento comercial con una antelación considerable y que la configuración final de un evento complejo*

depende de multitud de factores. Debido a esto, la información que se manejará en la base de datos podrá ser vaga.

Cada evento tendrá asociada una serie de reuniones de trabajo paralelas, a las que los departamentos o empresas participantes en el evento envían delegados. Las reuniones de trabajo paralelas de un determinado evento serán de iguales características. Las reservas de estas reuniones de trabajo contendrán información acerca del número de reuniones que se celebrarán de forma paralela y del estado de confirmación de cada una de ellas. Además, se dispondrá de información temporal acerca de la celebración de las reuniones, del número de asistentes a cada una, de si se desea servicio de catering para las reuniones y de los servicios de traducción requeridos por cada una. Los datos del conjunto anterior pueden ser preliminares y por tanto no estar totalmente confirmados.

El estado de confirmación de la celebración de las reuniones paralelas asociadas a un evento podrá ser uno de los siguientes:

- *Confirmada.*
- *Aprobada, pendiente de confirmación.*
- *Propuesta, pendiente de aprobación.*

Los anteriores estados están ordenados de forma decreciente en función del nivel de confianza en que la reunión se celebrará finalmente.

De forma similar, el nivel de confirmación de los datos sobre las reuniones podrá ser uno de los siguientes:

- *Datos confirmados.*
- *Datos pendientes de confirmar.*

La información temporal sobre la celebración de las reuniones está compuesta por la fecha en la que se celebrarán las reuniones paralelas, la hora de inicio de éstas y la duración que tendrán las mismas.

El número de asistentes, que será igual en cada reunión de trabajo paralela, será un número preciso en caso de que se disponga de la confirmación de los asistentes. En las ocasiones en que no se disponga de la confirmación de los asistentes, se emplearán valores imprecisos expresados como rangos de valores (por ejemplo, entre 20 y 30 participantes*) o como valores aproximados (por ejemplo,* aproximadamente 50*).*

El servicio de catering se puede confirmar con poca antelación por lo que es posible que no se conozca con total certidumbre si se requerirá o no este servicio. Con objeto de planificar las reservas, al menos se dispondrá de una respuesta preliminar por parte del cliente a falta de confirmación por parte del mismo.

Los datos relativos al servicio de interpretación constarán del número de intérpretes necesarios por cada idioma y del nivel de servicio de interpretación requerido a cada intérprete. Se considerarán los siguientes niveles de servicio de interpretación:

- *Simultanea: Interpretación literal que se realiza al mismo tiempo (o dentro de un intervalo de tiempo mínimo) que el orador pronuncia su discurso.*
- *Consecutiva: Interpretación literal en la que, cuando el orador realiza una pausa en el discurso, el intérprete traduce el fragmento anterior.*

Grado	Referencia
1.0	Confirmada
0.6	Aprobada pendiente de confirmación
0.3	Propuesta pendiente de aprobación

Tabla 5.6: Etiquetas de referencia para los grados de confirmación de la celebración de una reunión de trabajo

Grado	Referencia
1.0	Datos confirmados
0.5	Datos pendientes de confirmar

Tabla 5.7: Etiquetas de referencia para los grados de confirmación de los datos relativos a las reuniones de trabajo de un determinado evento

- *Acompañamiento: Interpretación no literal, en la que el intérprete transmite el sentido del discurso del orador.*

Cada intérprete estará cualificado para ofrecer un determinado nivel de servicio de interpretación en una lengua determinada. Los anteriores tipos de interpretación están ordenados de manera descendente en función del nivel de dominio requerido al intérprete. Por tanto, los intérpretes cualificados para realizar interpretaciones simultáneas, lo están para realizar interpretaciones consecutivas y de acompañamiento. De la misma forma, los cualificados para realizar interpretaciones consecutivas lo están para realizar interpretaciones de acompañamiento.

Toda la información anterior se representará empleando el modelo propuesto en una MRCD. Esta MRCD será denominada r. En r cada tupla contendrá los datos que describen las reuniones de trabajo paralelas asociadas a un evento.

La cardinalidad de cada tupla en la relación r determinará la cantidad de reuniones de trabajo paralelas asociadas a un evento. Esta participación, además de cuantificada, será cualificada de tal forma que se pueda indicar el grado de confirmación que se posee sobre la celebración de dicha reunión de trabajo. Para simplificar la interpretación por parte de los usuarios de estos grados de confirmación, se emplearán tres niveles, cada uno de ellos asociado a una etiqueta, que se corresponden con los niveles de confirmación de las reuniones mencionados anteriormente. Estos niveles junto con las etiquetas empleadas son los que se indican en la Tabla 5.6.

El nivel confirmación que se posee sobre los detalles del evento puede ser especificado mediante el nivel de certidumbre asociado a los valores de los atributos de la tupla que representa estas reuniones. Una vez más, y con objeto de simplificar la interpretación, se empleará un número determinado de grados asociados a ciertas etiquetas. Estos grados y sus etiquetas asociadas, correspondientes a los niveles de confirmación mencionados anteriormente, son los que se muestran en la Tabla 5.7.

El resto de información relativa a las reuniones se expresará como atributos de las tuplas en r. Por tanto, el conjunto atributos de r, A_r, estará definido como se muestra en la Ecuación 5.21. Los valores del atributo celebración

Atributo	Dominio
fecha	D_{fecha}
inicio	D_{hora}
duración	D_{nat}

Tabla 5.8: Definición extensiva de la función de dominio $dom_{D_{celeb}}$

reflejarán la información temporal que se dispone sobre la celebración de las reuniones, los valores del atributo asistentes *reflejarán en número de asistentes a cada reunión, los valores del atributo* catering *reflejarán la información de que se dispone sobre el requerimiento de servicio de catering de las reuniones y, finalmente, los valores del atributo* intérpretes *reflejará los servicios de interpretación requeridos para cada reunión.*

$$A_r = \{celebración, asistentes, catering, intérpretes\} \tag{5.21}$$

Los datos relativos a la información temporal son valores compuestos por tres componentes: fecha, hora de inicio y duración. Por tanto, los valores del atributo celebración *serán valores de un dominio complejo compuesto que denominaremos* D_{celeb}, $D_{celeb} \in \mathbb{C}$. *El conjunto de atributos de* D_{celeb}, $att(D_{celeb})$, *está definido como* $att(D_{celeb}) = \{fecha, inicio, duración\}$, *siendo los valores del atributo* fecha *los que representan la fecha en la que se realizará el evento, los valores del atributo* inicio *son los que representarán la hora a la que se iniciará el evento, y los valores del atributo* duración *los que indicarán (en número de horas) la duración del evento. El dominio asociado al atributo* fecha *será el dominio* D_{fecha} *definido en el Ejemplo 5.3.* D_{fecha} *es un dominio complejo básico,* $D_{fecha} \in \mathbb{B}$, *ya que sus valores son atómicos (véase Ejemplo 5.7) y no están afectados por imprecisión ni incertidumbre. El atributo* inicio *tendrá asociados como valores los pertenecientes al dominio* D_{hora}. *Definiremos el dominio* D_{hora} *como aquél cuyos valores representan las horas del día. Estos valores son de la forma* $(h : m), h \in [0, 23], m \in [0, 59]$, *donde el componente* h *representa la hora del día y el componente* m *los minutos pasados de la hora en punto. Estos valores están compuestos por dos elementos, pero sus elementos de forma aislada no tienen semántica propia en el universo modelado. Sólo la conjunción de ambos transmite el significado apropiado. Por tanto, este tipo de valores son atómicos. Además, los valores de este dominio no están afectados por incertidumbre o imprecisión alguna. En consecuencia diremos que* D_{hora} *es un dominio complejo de tipo básico,* $D_{hora} \in \mathbb{B}$, *ya que cumple las condiciones necesarias para ello. El atributo* duración *tendrá como valores asociados a los elementos del dominio* D_{nat}. *El dominio* D_{nat} *está definido inicialmente en el Ejemplo 5.2, y es un dominio complejo de tipo básico (véase Ejemplo 5.16). Los valores de* D_{nat} *son números naturales que indicarán el número de horas que durarán las reuniones paralelas. Con todo lo anterior, especificaremos la función dominio de* D_{celeb}, $dom_{D_{celeb}}$, *de forma extensiva como se indica en la Tabla 5.8.*

Como se ha mencionado anteriormente, el número de asistentes a cada reunión de trabajo es un número natural que, en ciertas ocasiones, puede ser impreciso debido a la imposibilidad de confirmar con total certeza todos los asistentes a los eventos. Estos números imprecisos podrán ser expresados mediante conjuntos difusos de carácter disyuntivo (valores de dominios de tipo impreci-

so que representan distribuciones de posibilidad) definidos sobre el conjunto de los número naturales. Los distintos tipos de cantidades imprecisas empleadas anteriormente, rangos y números aproximados, serán modelados por conjuntos difusos cuyas funciones de pertenencia se especifican en las ecuaciones 5.22 y 5.23 respectivamente. Obsérvese que, para el caso de valores aproximados, se permite un margen del 10 % de la cantidad a la que se aproxima. Obviamente, los valores precisos se modelarán empleando una distribución de posibilidad unaria cuya función característica se indica en la Ecuación 5.24.

$$\mu_{entre(a,b)}(x) = \begin{cases} 1 & si\ a \leq x \leq b \\ 0 & en\ otro\ caso \end{cases} \tag{5.22}$$

$$\mu_{aprox(a)}(x) = \begin{cases} 1 & si\ x = a \\ \frac{10*x-11*a}{a} & si\ \frac{9}{10} * a \leq x < a \\ \frac{11*a-10*x}{a} & si\ a < x \leq \frac{110}{100} * a \\ 0 & en\ otro\ caso \end{cases} \tag{5.23}$$

$$\mu_{exact(a)}(x) = \begin{cases} 1 & si\ x = a \\ 0 & en\ otro\ caso \end{cases} \tag{5.24}$$

Según lo anterior, el dominio asociado al atributo asistentes *debe ser un dominio complejo de tipo impreciso cuyo dominio subyacente será el dominio de los números naturales. Por tanto, para este caso podremos emplear el dominio D_{asist} definido en el Ejemplo 5.11. Este dominio es un dominio complejo incierto, $D_{asist} \in \mathbb{I}$, cuyo dominio subyacente es el dominio complejo básico D_{nat} empleado anteriormente, $sub(D_{asist}) = D_{nat}$.*

La información acerca del requerimiento de un servicio de catering, como se ha mencionado anteriormente, estará afectada por cierta incertidumbre en función de la confirmación por parte del cliente de su decisión final. Por tanto, el dominio asociado al atributo catering *se corresponderá con un valor de un dominio de tipo incierto definido sobre un dominio booleano subyacente. Un dominio de estas características es el dominio D_{conf} definido en el Ejemplo 5.8. D_{conf} es un dominio de tipo incierto, $D_{conf} \in \mathbb{I}$, cuyo dominio subyacente es el domino D_{bool}, $sub(D_{conf}) = D_{bool}$, también definido en el Ejemplo 5.8. D_{bool} a su vez es un dominio complejo básico, $D_{bool} \in \mathbb{B}$, ya que sus valores son atómicos y no están afectados por incertidumbre ni imprecisión.*

Los diferentes grados que se emplearán para codificar la certidumbre que se tiene sobre este valor serán iguales a los que se han indicando anteriormente en la Tabla 5.7.

En lo que se refiere a la información relativa a los servicios de interpretación que se requerirán en cada una de las reuniones paralelas, ésta será un valor múltiple, ya que se podrán requerir servicios de interpretación en distintos idiomas. La participación de cada elemento del valor conjuntivo será cuantificada, debido a que se pueden requerir varios intérpretes de un mismo idioma. Estas participaciones estarán, a su vez, graduadas en función del nivel de servicio requerido al intérprete. Según lo anterior, el dominio asociado al atributo intérpretes *deberá ser un dominio complejo de tipo multiconjuntivo difuso. Este dominio, que denominaremos D_{inter}, deberá tener como dominio subyacente a uno cuyos valores*

Grado	Nivel de servicio de interpretación
1.0	Simultánea
0.75	Consecutiva
0.5	Acompañamiento

Tabla 5.9: Grados asociados a los diferentes niveles de servicio de interpretación

representen los distintos idiomas para los que se requiere interpretación, como puede ser el dominio D_{lang} definido en el Ejemplo 5.15, $sub(D_{inter}) = D_{lang}$. El dominio D_{lang} es un dominio de tipo complejo básico, debido a que sus valores son atómicos y no se encuentran afectados por incertidumbre o imprecisión.

La correspondencia entre cada uno de los niveles de servicio de interpretación y los grados es la que se muestra en la Tabla 5.9.

Por tanto, y según lo anterior, la cabecera de la MRCD r, $\mathcal{H}_r$, se define como se indica en la Ecuación 5.25, estando A_r definido como se indica en la Ecuación 5.21 y dom_r según se indica en la Ecuación 5.26.

$$\mathcal{H}_r = (A_r, dom_r) \tag{5.25}$$

$$dom_r(a) = \begin{cases} D_{celeb} & si\ a = celebración \\ D_{asist} & si\ a = asistentes \\ D_{conf} & si\ a = catering \\ D_{inter} & si\ a = intérpretes \end{cases} \qquad \forall a \in A_r \tag{5.26}$$

Definamos ahora el cuerpo de r. Supongamos que deseamos representar en r la información relativa a las siguientes reservas:

- *Reserva de un evento para el día 4 de junio de 2010 a las 9:00, y hasta las 14:00, en que se celebrarán 6 reuniones paralelas, de las cuales 2 están confirmadas, 3 están aprobadas y pendientes de confirmación, y 1 está pendiente de aprobación. El número de asistentes a cada reunión será entre 20 y 25 personas. Se ha confirmado que se requerirá servicio de catering en las reuniones. Cada reunión requerirá los siguientes servicios de interpretación:*

  - *8 intérpretes en inglés, de los cuales 1 será capaz de hacer interpretación simultánea, 2 serán capaces de realizar interpretaciones consecutivas y 5 serán capaces de realizar interpretación de acompañamiento.*
  - *4 intérpretes en alemán, de los cuales 1 será capaz de hacer interpretación simultánea, otro será capaz de realizar interpretación consecutiva y 2 más serán capaces de realizar interpretación de acompañamiento.*

  Los datos anteriores están confirmados por el cliente.

- *Reserva para el día 14 de octubre de 2010 a las 17:00, y hasta las 20:00, en que se celebrarán 3 reuniones paralelas las cuales están aprobadas y pendientes de confirmación. El número de asistentes a cada reunión será aproximadamente 40 personas. De forma preliminar se indica que no se*

necesitará servicio de catering, pero este extremo no ha sido confirmado por el cliente. Cada reunión requerirá los servicios de interpretación de 10 intérpretes en inglés capaces de realizar interpretación de acompañamiento. Los datos anteriores no están confirmados por el cliente.

- *Reserva para el día 21 de noviembre de 2010 a las 15:00, y hasta las 21:00, en que se celebrarán 2 reuniones paralelas ya confirmadas. El número de asistentes a cada reunión será de 20 personas. De forma preliminar se indica que se requerirá servicio de catering en las reuniones, y se está a la espera de confirmación de este extremo por parte del cliente. Cada reunión requerirá los siguientes servicios de interpretación:*

  - *6 intérpretes en inglés, de los cuales 3 serán capaces de hacer interpretación simultánea y los otros 3 serán capaces de realizar interpretación de acompañamiento.*
  - *6 intérpretes en francés, de los cuales 1 será capaz de hacer interpretación simultánea y los otros 5 serán capaces de realizar interpretación consecutiva.*
  - *4 intérpretes en italiano capaces de realizar interpretación de acompañamiento.*

  Los datos anteriores han sido confirmados por el cliente.

- *Reserva para el día 9 de diciembre de 2010 a las 17:00 y hasta las 21:00 en que se celebrarán 4 reuniones paralelas propuestas y pendientes de aprobación. El número de asistentes a cada reunión será como máximo 5 personas. Se ha confirmado que no se requerirá servicio de catering en las reuniones. No se requerirán servicios de interpretación. Los datos anteriores están pendientes de confirmación por parte del cliente.*

Con toda la información de reservas anterior se define el cuerpo de la MRCD r, $\mathcal{B}_r$, como se indica en la Ecuación 5.27, donde T_R y U_R están definidos de forma extensiva como se indica en la Tabla 5.10.

$$\mathcal{B}_r = (T_r, U_r) \tag{5.27}$$

5.5. Bases de datos multirrelacionales complejas difusas

Finalmente, definido el concepto de dominio complejo y multirrelación compleja difusa en los que se basa el presente modelo, queda tan sólo definir el concepto de *base de datos multirrelacional compleja difusa*.

Las bases de datos relacionales tradicionales no son más que una agrupación de relaciones. De manera análoga, una base de datos multirrelacional compleja difusa agrupa una serie de multirrelaciones complejas difusas. Formalmente, definiremos una *base de datos multirrelacional compleja difusa* de la siguiente forma:

U_r	T_r	t_r						
		celebración			asistentes	catering	intérpretes	
		fecha	inicio	duración				
1.0	$\{1.0/2, 0.6/5, 0.3/6\}_c$	4-6-2010	9:00	5	$entre(20, 25)$	$\langle 1.0, \text{sí}\rangle$	$\{1.0/1, 0.75/3, 0.5/8\}_c$ $\{1.0/1, 0.75/2, 0.5/4\}_c$	Inglés Alemán
0.5	$\{0.6/3\}_c$	14-10-2010	17:00	3	$aprox(40)$	$\langle 0.5, \text{no}\rangle$	$\{0.5/10\}_c$	Inglés
1.0	$\{1.0/2\}_c$	21-11-2010	15:00	6	20	$\langle 0.5, \text{sí}\rangle$	$\{1.0/3, 0.5/6\}_c$ $\{1.0/1, 0.75/5\}_c$ $\{0.5/4\}_c$	Inglés Francés Italiano
0.5	$\{0.3/4\}_c$	9-12-2010	17:00	4	$hasta(5)$	$\langle 1.0, \text{no}\rangle$	$\emptyset_{inter}$	

Tabla 5.10: Reservas de reuniones de trabajo paralelas

Definición 5.21 (Base de datos multirrelacionales complejas difusas)**.** *Una* base de datos multirrelacional compleja difusa M *es un conjunto de* multirrelaciones complejas difusas. *Formalmente* M *está definida como se indica en la Ecuación 5.28. En dicha ecuación, cada* $m_i \in M$, $\forall i \in [1, n]$, *detona una de las multirrelaciones complejas difusas que forman parte de* M.

$$M = \{m_1, m_2, ..., m_n\}, M \subseteq \Delta, \tag{5.28}$$

Las bases de datos relacionales se dividen en dos componentes claramente diferenciados. El primer componente, denominado *esquema*, agrupa la información (de carácter invariante en el tiempo) relativa a la estructura de la base de datos. El segundo componente, denominado *instancia*, agrupa la información (de carácter variante en el tiempo) contenida en la base de datos siguiendo el esquema de ésta.

De la misma forma, una base de datos multirrelacional difusa M estará divida en su esquema y su instancia. Definiremos el esquema de M, $\mathcal{S}_M$, como el conjunto de las cabeceras de las MRCD que forman parte de ella, $\mathcal{S}_M = \{\mathcal{H}_m | m \in M\}$. La instancia de M, $\mathcal{I}_M$, estará definida como el conjunto de los cuerpos de las MRCD que la componen, $\mathcal{I}_M = \{\mathcal{B}_m | m \in M\}$.

5.6. Valores ausentes

Como se ha destacado en los capítulos dedicados al estado del arte de este trabajo, existen múltiples tipos, según la semántica, de valores ausentes en una base de datos.

En el presente modelo se considerarán únicamente los valores ausentes con las siguientes semánticas asociadas:

- Valor no existente.
- Valor desconocido pero existente.
- No existe información sobre el valor, ni siquiera si éste existe.

Este conjunto de valores viene impuesto por el paradigma de representación de valores ausentes que se ha elegido para el presente modelo, siendo en este caso el de Prade-Testemale [122, 121, 124, 123]. Éste destaca de entre las demás propuestas en la bibliografía por su simplicidad y coherencia con el modelo propuesto:

- Simplicidad: El paradigma requiere exclusivamente el empleo de un sólo elemento especial adicional a los propios de cada dominio para modelar los tipos de valores ausentes, en contraste con los múltiples símbolos de otras propuestas.
- Coherencia con el modelo propuesto: El paradigma modela los valores ausentes en forma de distribuciones de posibilidad, tipos de valores que el presente modelo puede tratar de forma natural.

5.6.1. Tipos de valores ausentes

El presente modelo considera tres tipos de valores ausentes, coincidiendo cada uno con una de las semánticas indicas anteriormente. A continuación definiremos cada uno de estos tipos.

5.6.1.1. Valor no existente

Este primer tipo de valor ausente se emplea en los casos en que el valor para un determinado atributo de una determinada entidad no existe debido a que dicho atributo no es aplicable a dicha entidad.

Ejemplo 5.18. *Supongamos que el centro de negocios empleado en los ejemplos anteriores organiza como parte de eventos empresariales, además de las reuniones paralelas que nos han servido de ejemplo anteriormente, cenas de gala para agasajar a los invitados a dichos eventos de empresa. Para organizar dichas cenas se mantendrá un listado con el nombre de los invitados y de sus acompañantes.*

Dicho listado, naturalmente, se traducirá en una relación con al menos un par de atributos: uno para representar el nombre del invitado y otro para representar el nombre del acompañante. En ocasiones, algún invitado no lleva acompañante a la cena, y por tanto no se dispone de nombre para el acompañante. Este caso es el ejemplo clásico de uso de valores inexistentes. Al no existir acompañante, el valor para la propiedad que representa el nombre del acompañante a la cena no existe. Para indicar la concurrencia de esta situación se empleará el valor no existente *como valor el citado atributo.*

Este tipo de valor ausente será representado en el presente modelo mediante un valor especial notado como *dne* (does not exists).

Todo dominio empleado en una base de datos objeto-multirrelacional difusa incluirá este valor especial, ya sea como un valor propio del dominio, o como un valor añadido a éste, según el tipo de dominio. Así pues, el valor ausente *dne* estará definido según el tipo de dominio al que pertenezca.

Definición 5.22. *Valor no existente. Sea D un dominio complejo, $D \in \mathbb{C}$. Un valor no existente asociado al dominio D, notado como dne_D, se define, en función del tipo de dominio complejo al que pertenece D, de la siguiente forma:*

- *Dominios inciertos: Si D es un dominio incierto, $D \in \mathbb{U}$, el valor ausente dne_D se corresponde con un valor del dominio D de la forma en que se indican en la Ecuación 5.29, donde $dne_{sub(D)}$ es el símbolo empleado para notar un valor inexistente en $sub(D)$, el dominio subyacente de D.*

$$dne_D = (1, dne_{sub(D)}) \ \ \forall D \in \mathbb{U} \tag{5.29}$$

La anterior definición de valor inexistente asigna un grado de certidumbre 1, total certidumbre, al valor ausente dne_D. Nótese que cualquier dominio incierto D, $D \in \mathbb{U}$, además del anterior valor incierto dne_D, también incluye valores inexistentes con distintos grados de certidumbre $\alpha \in (0, 1]$. Estos valores se notarán como dne_D^{α} y se denominarán valores inexistentes α-inciertos. Este tipo de valores están definidos como se indica en la Ecuación 5.30.

$$dne_D^{\alpha} = (\alpha, dne_{sub(D)}) \ \ , \forall D \in \mathbb{U}, \alpha \in (0,1] \tag{5.30}$$

- *Dominios imprecisos: Si D es un dominio impreciso, $D \in \mathbb{I}$, el valor ausente dne_D se corresponde con la siguiente distribución de posibilidad definida en la Ecuación 5.31. En la citada ecuación $dne_{sub(D)}$ es, como en el caso anterior, el símbolo empleado para notar un valor ausente en $sub(D)$, el dominio subyacente de D.*

$$dne_D = \{1/dne_{sub(D)}\} \ \ \forall D \in \mathbb{I} \tag{5.31}$$

- *Dominios compuestos: En el caso de que D sea un dominio compuesto, $D \in \mathbb{O}$, el valor ausente dne_D es una tupla de la forma $dne_D : att(D) \mapsto dom_D(att(D))$ definida como se indica en la Ecuación 5.32. En dicha ecuación, $dne_{dom(a)}$ es el símbolo para el valor ausente con semántica tipo 1, valor no existente, asociado al dominio de un atributo a, $a \in att(D)$.*

$$dne_D(a) = dne_{dom(a)} \ \forall a \in att(D), D \in \mathbb{O} \tag{5.32}$$

- *Dominios multiconjuntivos difusos: Cuando el dominio D es un dominio multiconjuntivo difuso, $D \in \mathbb{M}$, el valor ausente dne_D se corresponderá con el* conjunto vacío*, notado como $\emptyset_D$. Se ha de notar que $\emptyset_D$ es un valor propio de cualquier dominio multiconjuntivo difuso. Éste se define de la forma que se indica en la Ecuación 5.33.*

$$dne_D = \emptyset_D = [\{1/0\} * v \mid v \in sub(D)] \tag{5.33}$$

- *Dominios básicos: Si D es un dominio básico, $D \in \mathbb{B}$, el valor ausente dne_D será un valor especial añadido al conjunto de los valores del dominio.*

Ejemplo 5.19. *Un valor ausente de tipo no existente, un valor dne, ya se ha usado en los ejemplos anteriores, en concreto en el cuerpo de la MRCD r definido en el Ejemplo 5.17. El uso del citado valor ha pasado inadvertido ya que en ese caso el valor dne es un valor propio del dominio. Dicho caso es el de la última tupla que forma el cuerpo de r, definido en la Tabla 5.10. Las reuniones paralelas del evento representado por dicha tupla no requieren servicios de interpretación, y por tanto se usa el valor $\emptyset_{inter}$ para indicar tal circunstancia. Debido a que el dominio del atributo* intérpretes*, D_{inter}, es de tipo multiconjunto difuso, el valor no existente del dominio, dne_{inter}, se corresponde (como indica la definición anterior) con el conjunto vacío, $\emptyset_{inter} = dne_{inter}$.*

5.6.1.2. Valor desconocido pero existente

El segundo tipo de valores ausentes se emplea para indicar que el valor concreto para un determinado atributo de una determinada entidad es desconocido aunque se tiene la certeza de que éste existe.

Ejemplo 5.20. *En ocasiones cuando en una base de datos se reflejan los datos que se tienen disponibles por el momento, es posible que se tenga que tratar con valores desconocidos pero existentes. Éste podría ser el caso de la base de*

datos que almacena la información relativa a eventos empresariales que se ha empleado en los ejemplos anteriores.

Supongamos que los agentes comerciales conocen algunos de los datos relativos a las reuniones paralelas asociadas a un determinado evento, pero el cliente no ha proporcionado una aproximación (y mucho menos precisado) el número de asistentes a cada una de la reuniones. Con el objetivo de que los datos disponibles acerca de dichas reuniones queden reflejados en la base de datos, los agentes comerciales incluyen una tupla con esos datos. El problema llega cuando se ha de indicar un valor para el atributo asistentes. *En el caso descrito, no se dispone de un valor concreto para dicho atributo. El motivo por el que no se conoce dicho valor no es porque la propiedad que representa el atributo no sea aplicable a la entidad como ocurría en el ejemplo 5.18. Todo lo contrario, se conoce que dicho atributo debe tener un valor, pero en este momento no se puede precisar un determinado valor para el citado atributo.*

Una solución aplicada comúnmente consiste en dar un valor temporal para el atributo del que no se conoce el valor. Este dato, una vez que se conozca el verdadero valor del atributo, será sustituido. El problema es que, una vez más, el sistema no es capaz de determinar la naturaleza de la información introducida y por tanto trabajará sobre esta información temporal (y que generalmente no corresponde con la realidad) de la misma forma que lo hace con la correcta. Finalmente, esto supone que el sistema, al emplear la información temporal en sus procesos, ofrecerá resultados erróneos.

En estos casos es mucho más adecuado y ventajoso emplear el *valor desconocido pero existente* como valor del atributo para el que se desconoce el valor. De esta forma queda reflejada la información que se posee acerca del citado atributo: exclusivamente su existencia.

Este tipo de valor ausente se representará en el presente modelo haciendo uso de una distribución de posibilidad sobre los valores del dominio asociado al atributo en cuestión, de tal forma que todos los valores de dicho dominio, excepción hecha del valor *dne*, son posibles valores para el atributo.

Formalmente, un valor de este tipo se define de la siguiente forma:

Definición 5.23 (Valor desconocido pero existente)**.** *Sea A un atributo, $A \in \mathbb{A}$, sea dom una función de dominio que asocia al atributo A con su respectivo dominio D, $dom(A) = D$. Un valor desconocido pero existente de un determinado dominio D, notado por simplicidad como unk_{I_D} (unknown), se define como la siguiente distribución de posibilidad definida en la Ecuación 5.34.*

$$unk_{I_D} = \{1/d \mid d \in D, d \neq dne\} \tag{5.34}$$

Nótese que la distribución de posibilidad unk_{I_D} para un determinado dominio D no es un valor propio de dicho dominio si no una distribución de posibilidad sobre los valores de D. Por tanto unk_{I_D} es un valor correspondiente al dominio impreciso I_D, $I_D \in \mathbb{I}$, donde D sea el dominio subyacente, $sub(I_D) = D$.

Por tanto, para permitir la representación de este tipo de valores ausentes el dominio del atributo debe ser, en lugar de D, un dominio de tipo impreciso cuyo dominio subyacente sea D, $dom(A) = I_D$.

Ejemplo 5.21. *Supongamos que se desea incorporar a la MRCD r definida en el ejemplo 5.17 una nueva tupla para representar la información que se dispone*

U_r	T_r	t_r					
		celebración			asistentes	catering	intérpretes
		fecha	inicio	duración			
0.5	$\{1.0/1\}_c$	4-3-2010	8:00	7	unk_{asist}	$\langle 1.0, \text{si}\rangle$	$\emptyset_{inter}$

Tabla 5.11: Reserva de reunión de trabajo para la que no se conoce el número de asistentes.

acerca de la reserva un determinado evento para el cual se están valorando aún determinados aspectos. La información disponible en el momento es la siguiente:

> *Reserva para el día 4 de marzo de 2010 a las 8:00, y hasta las 15:00, en que se celebrará una reunión. El número de reuniones se ha confirmado. No se dispone por el momento de una aproximación sobre el número de asistentes. Se confirma que se requerirá un servicio de catering. No se requerirán intérpretes. La información está pendiente de ser confirmada por parte del cliente.*

Para representar esta información se emplearía la siguiente tupla que se muestra en la Tabla 5.11.

5.6.1.3. Valores sobre los que no existe información

Finalmente, el tercer tipo de valores ausentes tiene asociada la semántica que representa el caso en que no se posee ninguna información del valor para un determinado atributo de una determinada entidad. Esta carencia de información significa, en definitiva, que no se conoce el valor concreto y ni siquiera se tiene la certeza de que dicho valor exista.

Ejemplo 5.22. *Empleemos de nuevo el contexto de la base de datos que almacena los datos relativos a las reuniones paralelas asociadas a eventos empresariales. Supongamos que ahora permitimos que la información relativa a los servicios de interpretación requeridos sea imprecisa, de tal forma que se permita representar las diferentes alternativas de requerimientos de este tipo de servicios que baraja el cliente. Por ejemplo, un cliente puede estar indeciso en el número de intérpretes para cada uno de los idiomas de los participantes. Se debate entre tres intérpretes en inglés y dos en alemán o dos en inglés y tres en alemán, ya que algunos de los participantes dominan ambas lenguas pero no conoce cual de ellas prefieren. Esta información imprecisa se podrá representar como una distribución de posibilidad sobre valores del dominio D_{inter}.*

Un caso extremo de imprecisión en los servicios de interpretación requeridos es la ignorancia absoluta sobre qué servicios serán finalmente necesarios. El cliente no sabe exactamente qué servicios requerirá. En una primera aproximación se podría pensar que esta situación se podría representar por el valor desconocido unk *definido anteriormente. No obstante, el desconocimiento absoluto sobre los servicios de interpretación que se requerirán incluye la posibilidad de que no se requiera ninguno. El hecho de que no se requiera ningún servicio de interpretación se representa por el conjunto vacío $\emptyset_{inter}$. Debido a que D_{inter} es un dominio multiconjuntivo difuso, $D_{inter} \in \mathbb{M}$, el conjunto vacío*

$\emptyset_{inter}$ se corresponde con el valor inexistente dne_{inter} . Recordemos que el valor desconocido excluye explícitamente al valor inexistente, por lo que la situación descrita no puede ser representada por el citado valor. Por tanto, éste es un caso típico donde el valor sobre el que no existe información *permite reflejar completamente la situación descrita. Este valor ausente incluye la posibilidad de que el valor para un determinado atributo finalmente no exista así como que éste sea cualquiera de los valores del dominio.*

Al igual que el caso de valores desconocidos pero existentes, valores *unk*, este tipo de valor ausente se representará en el presente modelo haciendo uso de una distribución de posibilidad sobre los valores del dominio asociado al atributo en cuestión, de tal forma que todos los valores de dicho dominio, en este caso incluso el valor *dne*, sean posibles valores para el atributo.

Formalmente, un valor de este tipo se define de la siguiente forma:

Definición 5.24 (Valor sobre el que no existe información)**.** *Sea A un atributo, $A \in \mathbb{A}$, sea dom una función de dominio que asocia al atributo A con su respectivo dominio D, $dom(A) = D$. Un valor sobre el que no existe información de un determinado dominio D, notado por simplicidad como ni_{I_D} (no information), se define como la distribución de posibilidad definida en la Ecuación 5.35.*

$$ni_{I_D} = \{1/d \mid d \in D\} \tag{5.35}$$

Se ha de notar que en la anterior distribución de posibilidad se incluye el valor dne_D ya que éste se considera un valor más de cualquier dominio complejo D. Igual que en el caso de la distribución unk_{I_D}, la distribución de posibilidad ni_{I_D} para un determinado dominio D no es un valor contemplado en dicho dominio. Como anteriormente se ha indicado para el valor ausente unk_{I_D}, para permitir que se representen valores ausentes de tipo ni_{I_D}, el atributo A deberá asociarse, en substitución del dominio D, con un dominio impreciso I_D, $I_D \in \mathbb{I}$, donde D sea el dominio subyacente, $sub(I_D) = D$.

Ejemplo 5.23. *Supongamos que el sustituimos el dominio asociado al atributo* intérpretes *por el dominio D_{iint}. Éste es un dominio impreciso, $D_{iint} \in \mathbb{I}$, cuyo dominio subyacente es D_{int}, el anterior dominio asociado al atributo* intérpretes*, $sub(D_{iint}) = D_{int}$. Este cambio abre la posibilidad de expresar valores imprecisos para el atributo* intérpretes*, tal y como se ha sugerido en el Ejemplo 5.22.*

Supongamos ahora que deseamos a añadir a la MRCD r, definida en el ejemplo 5.17 y aplicada la modificación que se acaba de proponer, una tupla que represente la información que se dispone acerca de la siguiente reserva de un evento:

> *Reserva para el día 8 de febrero de 2010 a las 9:00, y hasta las 18:00, en que se celebrarán dos reuniones paralelas, ambas confirmadas. El número de asistentes será de 20. No se requerirá servicio de catering (confirmado). No se conoce aún ninguna información relativa a los servicios de interpretación requeridos. La información está pendiente de ser confirmada por parte del cliente.*

La información anterior se representará mediante una tupla similar a la indicada en la Tabla 5.12.

U_r	T_r	t_r					
		celebración			asistentes	catering	intérpretes
		fecha	inicio	duración			
0.5	$\{1.0/2\}_c$	8-2-2010	9:00	9	20	$\langle 1.0, \text{no}\rangle$	ni_{iinter}

Tabla 5.12: Reserva de reunión de trabajo para la que no se conoce los servicios de interpretación requeridos.

5.6.2. Restricciones de uso de los valores ausentes en función del dominio

Como se ha mencionado anteriormente, los tipos de valores ausentes que se considerarán en el presente modelo serán los valores no existentes, los valores desconocidos pero existentes y los valores sobre los que no existe información alguna.

De los anteriores, el primer tipo de valor será representable en cualquiera de los dominios de una base de datos objeto-multirrelacional difusa. En cambio, los dos últimos tipos de valores serán representables exclusivamente en dominios imprecisos.

La exclusividad de representación de los valores ausentes de tipo desconocido pero existente y de los valores para los que no existe ninguna información se debe al paradigma empleado para la representación de este tipo de valores. Éste, el de Prade-Testemale como se ha indicado anteriormente, modela estos tipos de valores ausentes como distribuciones de posibilidad sobre el dominio de que forman parte estos valores. Este enfoque requiere, por tanto, que cualquier dominio de la base de datos contenga como valores las posibles distribuciones de posibilidad sobre un dominio subyacente determinado.

El modelo de base de datos objeto-multirrelacional difusa que propone este trabajo ha sido creado con el objeto de proporcionar un modelo capaz de representar información compleja, proporcionando dominios que puedan adaptarse a la distinta naturaleza de ésta. Dotar a cualquier dominio de la posibilidad de contener como propios valores ausentes de tipo 2 y 3 obligaría a que todo dominio fuese un dominio impreciso que contenga como valores las distribuciones de posibilidad que se refieren. Este requisito introduciría artificialmente un nivel de complejidad adicional para los casos en el que no es preciso representar estos tipos de información ausente. Por tanto, se ha optado por que los valores ausentes de tipo 2 y 3 estén exclusivamente disponibles para los dominios imprecisos, dejando al modelador de la base de datos la decisión de incorporar el citado nivel de complejidad adicional sólo en aquellos casos donde es necesario representar estos tipos de valores ausentes.

5.7. Conclusiones

En el presente capítulo se ha propuesto un modelo para las bases de datos objeto-multirrelacionales difusas. Este modelo, basado en el relacional, representa las entidades de una determinada clase como tuplas en un multiconjunto difuso. Gracias a ello, el modelo soporta la representación de información cuan-

titativa y cualitativa sobre las ocurrencias de una entidad en una determinada clase y, como consecuencia de lo anterior, sobre las relaciones entre las diversas entidades presentes en una base de datos.

Gracias a los diversos dominios complejos definidos para el presente modelo, los valores asociados a los atributos de una determinada entidad pueden ser imperfectos, compuestos o, incluso, una combinación de ambas características. En lo que refiere a la capacidad del modelo para la representación de datos imperfectos, éste da la posibilidad de representar datos imprecisos y datos inciertos. Además de dar la posibilidad de indicar el grado de incertidumbre sobre el un determinado valor, el modelo permite indicar la certidumbre que se posee sobre la totalidad de los valores de las propiedades de una determinada entidad. Con respecto a la capacidad de representación de datos compuestos, el modelo soporta las colecciones de datos tanto homogéneos como heterogéneos. Las colecciones de datos homogéneos son modeladas como multiconjuntos difusos, por lo que el modelo soportará que éstas posean un tamaño arbitrario y que sus elementos mantengan información cuantitativa y cualitativa de su participación. Las colecciones de datos heterogéneos se representarán como composiciones, a modo de tuplas, de tamaño fijo en que cada elemento debe ser un valor de un dominio determinado.

Finalmente, se ha de destacar la capacidad del modelo para la representación de valores ausentes. Éste ha sido dotado de capacidad suficiente para la representación de valores ausentes con tres tipos de semánticas asociadas, coincidiendo con las semánticas más empleadas en los modelos de bases difusas presentes en la bibliografía.

Capítulo 6

Modelo objeto-multirrelacional difuso: álgebra

6.1. Introducción

El motivo fundamental por el que la información desestructurada se representa de forma estructurada en una base de datos no es otro que poder el acceder rápida y cómodamente a ésta y derivar nueva información de la anterior. Con el empleo de sistemas automatizados, particularmente ordenadores, las deseables características anteriormente mencionadas se maximizan.

En el capítulo anterior se ha descrito la estructura de datos del modelo propuesto que permite la representación de información imperfecta en una base de datos. Para dar utilidad a dicha representación, es necesario definir un lenguaje que permita el acceso y manipulación de los datos representados. El objetivo del presente capítulo es definir tal lenguaje. Ésta es un álgebra, basada en el álgebra relacional, que permitirá el acceso y manipulación de las MRCDs presentes en una base de datos objeto-multirrelacional difusa. Para ello, adaptaremos el funcionamiento de los operadores del álgebra relacional a las particularidades de las MRCDs. Además de lo anterior, introduciremos un nuevo conjunto de operadores, que denominaremos *operadores de transformación de dominios*, basados en el par de operadores de anidación y desanidación del álgebra de las relaciones NF^2, que permitirán la transformación de los dominios complejos asociados a un atributo en la MRCD en sus componentes y viceversa.

Para dar consistencia al álgebra propuesta, se introducirá una forma normal, inspirada en la FNP del álgebra NF^2, que asegurará, al igual que su precedente, que las MRCDs que manipula el álgebra propuesta cumplen determinadas condiciones para garantizar el comportamiento adecuado sus operadores. Particularmente, al igual que la FNP, se pretende asegurar que los operadores de transformación de dominios (los equivalentes a los operadores de anidación y desanidación para el caso de la FNP) son complementarios entre sí.

El presente capítulo está organizado de la siguiente forma. En primer lugar, se introducirá la mencionada forma normal para las MRCD. Seguidamente, se

definirán los operadores del álgebra de MRCDs derivados del álgebra relacional, abordando en primer lugar el subconjunto de operadores básicos para, posteriormente, definir los operadores derivados del álgebra. Posteriormente, se definirán los mencionados operadores de transformación de dominios. Finalmente, se expondrán las conclusiones relativas al contenido del presente capítulo.

6.2. Forma normal de particiones extendida

Como se ha visto en los capítulos introductorios del presente trabajo, el álgebra para las relaciones NF^2 requirió que dichas relaciones cumplieran la FNP para asegurar resultados coherentes en la aplicación de los operadores de anidación y desanidación. Al igual que dicha álgebra, el álgebra para la manipulación de MRCDs requerirá que éstas cumplan una serie de requisitos. A este conjunto de requisitos los denominaremos *Forma Normal de Particiones Extendida* (FNPE).

La FNPE tiene el mismo espíritu que la FNP: asegurar que las operaciones de anidación y desanidación (agregación y desagregación en el caso de las MRCDs) sean complementarias. Para garantizar esto, la FNP se ocupa de evitar que las tuplas de una relación NF^2 se diferencien exclusivamente por atributos cuyos valores no son atómicos. En otras palabras, las tuplas de una relación NF^2 se identifican de manera unívoca por el valor de sus valores atómicos. De esta forma, los conjuntos de tuplas resultantes de un proceso de desanidación de cada una de las tuplas originales son disjuntos. Esto garantizará que dicho proceso de desanidación pueda ser revertido por un proceso de anidación, dado que hay una relación unívoca entre las tuplas de la relación original y las tuplas de la relación resultante.

La FNPE extiende a la FNP en el sentido de que considera varios tipos de dominios complejos, a diferencia de la FNP que consideraba en exclusiva a las relaciones anidadas. Una MRCD puede contener atributos asociados a diferentes dominios de tipo complejo, como son los dominios de tipo incierto, compuesto, impreciso y multiconjuntivo difuso. Todos ellos son susceptibles de participar en un proceso de *desagregación*, equivalente al proceso de desanidación para las relaciones NF^2, que descompondrá sus valores. Estas desagregaciones podrán ser revertidas posteriormente por un proceso de *agregación*.

Al contrario que la FNP, que sólo empleaba valores atómicos, la FNPE extenderá el conjunto de valores que se emplearán para la identificación unívoca de tuplas. Esto ampliará el rango de MRCDs que podrán ser tratadas por el álgebra que se define en este trabajo. Además de los valores de tipo básico, la FNPE permitirá que se puedan identificar a las tuplas por valores compuestos e inciertos. Esta posibilidad, no obstante, estará limitada al cumplimiento de ciertas condiciones por parte de los dominios subyacentes de éstos.

Antes de comenzar la definición formal de la FNPE, definiremos una serie de conceptos en los que ésta se apoyará y que ayudarán en dicho cometido.

6.2.1. Dominios signatura

En primer lugar, será necesario determinar los dominios cuyos valores servirán para identificar unívocamente cualquier tupla de una MRCD. A este tipo

de dominios los denominaremos *dominios signatura*. Como se mencionó anteriormente, el objetivo de la obligatoriedad de esta identificación unívoca es el mismo que para la FNP: poder determinar una relación unívoca entre tuplas resultantes y originales de un proceso de desagregación para poder revertir éste en un proceso de agregación.

Al igual que ocurre con la FNP, los dominios de tipo básico serán incluidos en el conjunto de dominios signatura. Los dominios de tipo básico son atómicos y simples, por lo que no se podrán ver involucrados en un proceso de desagregación y, por tanto, sus valores no cambiarán en dichos procesos.

Por los motivos que se comentaron en el caso de la FNP, los dominios que contengan valores que causen que una tupla original derive en varias tuplas resultantes en un proceso de desagregación quedarán descartados. Este es el caso de los dominios de tipo impreciso y multiconjuntivo difuso.

En lo que respecta al caso de los dominios compuestos y dominios de tipo incierto, permitiremos identificar a las tuplas empleando los valores asociados a este tipo de dominios si cumplen determinadas condiciones. Debido a las particularidades de estos tipos de dominios, emplearemos sólo algunos componentes del valor para realizar esta identificación unívoca. En el caso de dominios compuestos, emplearemos exclusivamente aquellos componentes del valor que sean de un dominio signatura. En lo que respecta a los dominios de tipo incierto, emplearemos sólo el valor base de éstos, siempre y cuando, éste sea un valor de un dominio signatura. El motivo de esta restricción es garantizar que, después de un proceso de desagregación que involucre a uno de estos valores, los valores empleados para la identificación unívoca en la relación original sigan siendo válidos para realizar dicha identificación en la relación resultante. En caso de que esto no fuese así, dos tuplas que se pueden identificar de forma unívoca en la relación original, gracias exclusivamente a uno de estos valores, no podrán serlo en la relación resultante.

Según lo anterior, definiremos formalmente el concepto de dominio signatura como se indica a continuación.

Definición 6.1 (Dominios signatura)**.** *Definiremos el conjunto de dominios signatura, notado como* $\mathbb{D}^{sig}$*, de la forma que se indica en la Ecuación 6.1.*

$$\mathbb{D}^{sig} = \{D | (D \in \mathbb{B}) \vee (D \in \mathbb{U} \wedge sub(D) \in \mathbb{D}^{sig}) \vee \\ \big(D \in \mathbb{O} \wedge (\exists D' \in dom_D(att(D)) | (D' \in \mathbb{D}^{sig}))\big)\} \quad (6.1)$$

Obsérvese que en la definición anterior se establece que un dominio de tipo compuesto será un dominio signatura si alguno de sus componentes está asociado a un dominio signatura. En el caso de los dominios de tipo incierto, estos serán dominios signatura si su dominio subyacente es un dominio signatura.

6.2.2. Atributos signatura

Una vez determinados cuáles serán los dominios signatura, será útil determinar, como veremos más adelante, para un conjunto de atributos dado cuáles de ellos tendrán asociados valores que se emplearán para identificar unívocamente tuplas. Claramente, los atributos seleccionados serán aquellos que tengan asociado un dominio que sea parte del conjunto de dominios signatura.

Según lo anterior, definiremos el conjunto de atributos cuyos valores (o parte de los mismos) servirán para la identificación unívoca de tuplas, conjunto que denominaremos *conjunto de atributos signatura*, de la forma que se indica a continuación.

Definición 6.2 (Conjunto de atributos signatura)**.** *Sea A un conjunto de atributos y dom una función dominio que empareja cada atributo de A con su dominio asociado.*

El conjunto de atributos signatura en A según la función dominio dom, notado como $sig(A, dom)$, se define de la forma indicada en la Ecuación 6.2.

$$sig(A, dom) = \{a | a \in A \wedge dom(a) \in \mathbb{D}^{sig}\} \qquad \forall A \subseteq \mathbb{A}, \forall dom : A \mapsto \mathbb{D} \quad (6.2)$$

Cuando el conjunto de atributos A y la función de dominio dom son las de una MRCD m, podremos determinar el conjunto de atributos signatura de m aplicando la definición anterior. Por simplicidad en la notación, $sig(m)$ notará dicho conjunto de atributos signatura. Por tanto, $sig(m) = sig(A_m, dom_m)$.

6.2.3. Valor signatura

Como se mencionó anteriormente, para los dominios signatura de los tipos compuesto e incierto se emplearán exclusivamente algunos de los componentes de sus valores para realizar la tarea de identificación de tuplas. En dichos casos, es necesario determinar, dado un valor de dichos tipos de dominios, cual será el valor efectivo que se empleará como signatura para la identificación de tuplas.

Para ello, definiremos el concepto de *valor signatura*. El valor signatura de un valor compuesto será un nuevo valor compuesto cuyos componentes serán los valores signatura de aquellos componentes del valor original que estén asociados a dominios signatura. En el caso de un valor impreciso, se empleará la signatura de la base de dicho valor, ignorando el grado de certidumbre del mismo. Obviamente, el valor signatura de un valor básico será él mismo.

Según lo anterior, definiremos el *valor signatura* de un valor de un dominio signatura como sigue.

Definición 6.3 (Valor signatura)**.** *Sea D un dominio signatura. Sea d un valor de D. El* valor signatura *del valor d de un dominio D se define como se indica en la Ecuación 6.3.*

$$sig(D, d) =$$

$$= \begin{cases} d & si\ D \in \mathbb{B} \\ sig(sub(D), value(d)) & si\ D \in (\mathbb{U} \cap \mathbb{D}^{sig}) \\ d' \in O | (O \in \mathbb{O} | \\ \quad (att(O) = sig(att(D), dom_D)\ \wedge\ dom_O = dom_D^{sig}) \wedge \\ \quad \forall a \in att(O), d'(a) = sig(dom_O(a), d(a))) & si\ D \in (\mathbb{O} \cap \mathbb{D}^{sig}) \end{cases}$$

$$\forall D \in \mathbb{D}^{sig}, \forall d \in D \quad (6.3)$$

En la Ecuación 6.3 incluida en la definición anterior, dom_D^{sig} es una función dominio derivada de la función dominio dom_D de un dominio complejo D a la que denominaremos *función de dominio de signaturas*. Ésta relaciona cualquier atributo $a \in att(D)$ asociado a un dominio signatura D_a, $D_a = dom_D(a)$, con el *dominio de signaturas* de D_a, notado como $sig(D_a)$. A su vez, definimos el *dominio de signaturas* de un dominio signatura D_a, notado como $sig(D_a)$, como un dominio compuesto por los valores signatura de los valores incluidos en el dominio D_a. Definiremos el par de conceptos empleados anteriormente como se indica en las siguientes definiciones.

Definición 6.4 (Dominio de signaturas)**.** *Sea D un dominio signatura, $D \in \mathbb{D}^{sig}$. El dominio de signaturas de D es un dominio definido según se indica en la Ecuación 6.4.*

$$sig(D) = \{sig(D,d) | d \in D\} \qquad \forall D \in \mathbb{D}^{sig} \tag{6.4}$$

Definición 6.5 (Función de signaturas)**.** *Sea dom una función dominio de la forma $dom : A \mapsto \mathbb{D}$, donde A es un conjunto de atributos, $A \subset \mathbb{A}$. La* función de dominio de signaturas *derivada de dom, notada como dom^{sig}, es una función dominio de la forma $dom^{sig} : sig(A, dom) \mapsto \mathbb{D}$ definida como se indica en la Ecuación 6.5.*

$$dom^{sig}(a) = sig(dom(a)) \qquad \forall dom : A \mapsto \mathbb{D} | (A \subset \mathbb{A}), \forall a \in sig(A, dom) \tag{6.5}$$

6.2.4. Signatura de una tupla

Una vez determinado el conjunto de atributos cuyos valores nos servirán para identificar unívocamente las tuplas y las partes de dichos valores que servirán para la citada tarea, simplemente queda por determinar en qué consistirá dicho identificador. Al identificador unívoco de una tupla lo denominaremos *signatura de una tupla*. La signatura de una tupla consistirá en una tupla que enlace los atributos que forman parte de su conjunto de atributos signatura con los valores signatura de éstos.

Definiremos formalmente la signatura de una tupla, según lo anterior, como se indica a continuación.

Definición 6.6 (Signatura de una tupla)**.** *Sea $t \in \mathbb{T}_{(A,dom)}$ una tupla, donde $A \subset \mathbb{A}$ es un conjunto de atributos y dom una función dominio de la forma $dom : A \mapsto \mathbb{D}$.*

La signatura de una tupla t es una tupla $sig(t) \in \mathbb{T}_{(sig(A,dom),dom^{sig})}$ que se define como se indica a en la Ecuación 6.6.

$$\begin{aligned} &sig(t)(a) = sig(t(a), dom(a)) \\ &\quad \forall t \in \mathbb{T}_{(A,dom)} | (A \subset \mathbb{A} \ \wedge \ dom : A \mapsto \mathbb{D}), \forall a \in A | (dom(a) \in \mathbb{D}^{sig}) \end{aligned} \tag{6.6}$$

6.2.5. Signatura restringida de una tupla

Como se verá más adelante en el presente capítulo, en ocasiones es interesante obtener la signatura de una tupla considerando exclusivamente los valores

de un subconjunto de sus atributos signatura. A este tipo de signatura la denominaremos *signatura restringida*. Este concepto se define formalmente como se indica a continuación.

Definición 6.7 (Signatura restringida de una tupla)**.** *Sea $t \in \mathbb{T}_{(A,dom)}$ una tupla, donde $A \subset \mathbb{A}$ es un conjunto de atributos y dom una función dominio de la forma $dom : A \mapsto \mathbb{D}$. Sea $B \subseteq A$ un subconjunto de los atributos de t.*

La signatura restringida a B de una tupla t , notada como $sig^{|B}(t)$, se define según se indica a en la Ecuación 6.7.

$$sig^{|B}(t)(b) = sig(t(b), dom(b))$$
$$\forall t \in \mathbb{T}_{A,dom} | (A \subset \mathbb{A} \ \wedge \ dom : A \mapsto \mathbb{D}), \forall B \subseteq A, \forall b \in B | (dom(b) \in \mathbb{D}^{sig}) \tag{6.7}$$

6.2.6. Forma normal de particiones extendida

Vista la forma de identificación unívoca de tuplas en una MRCD mediante sus signaturas, queda finalmente definir en qué consistirá la FNPE. Ésta, básicamente, introduce la restricción de que ninguna tupla tendrá asociada una signatura igual a la de otra tupla en una MRCD determinada.

Adicionalmente, se ha de tener en cuenta que, para que la restricción anterior se cumpla tras las desagregación de un atributo con un dominio compuesto de valores múltiples (valores imprecisos o multiconjuntivos), se han de restringir los posibles dominios de esta naturaleza a aquellos que poseen como dominio subyacente un dominio signatura. En caso contrario, las tuplas del conjunto en que derive una determinada tupla de la MRCD, participante en un proceso de desagregación, no podrían ser distinguidas por su signatura.

Estas restricciones, como veremos más adelante, garantizarán el buen comportamiento de los operadores del álgebra de MRCDs y particularmente la complementación de los operadores de agregación y desagregación.

Según los argumentos anteriormente expuestos, definiremos la FNPE según se indica a continuación.

Definición 6.8 (Forma normal de particiones extendida)**.** *Sea Δ^* un conjunto de MRCDs definido como se indica en la Ecuación 6.8. Sea m una MRCD. Se dice que m está en* forma normal de particiones extendida *(FNPE) si $m \in \Delta^*$.*

$$\Delta^* = \{m | m \in \Delta \ \wedge \ \nexists (t, t') \in (T_m \times T_m) | (t \neq t' \wedge sig(t) = sig(t')) \wedge$$
$$\forall d \in (dom_m(A_m) \cap (\mathbb{I} \cup \mathbb{M})), sub(d) \in \mathbb{D}^{sig}\} \tag{6.8}$$

Por simplicidad, a partir de este punto, siempre se entenderá que cualquier MRCD estará en FNPE sin que sea necesario mencionar esta particularidad. De la misma forma, emplearemos de manera indistinta el conjunto de las posibles MRCDs (el conjunto Δ) y el conjunto de las posibles MRCD que cumplen la FNPE (el conjunto Δ^*).

6.3. Álgebra sobre multirrelaciones complejas difusas

La presente sección aporta una transposición al contexto de las MRCDs de los operadores del álgebra relacional clásica. Esta sección comprende exclusivamente los operadores básicos del citado álgebra, ya que el resto de operadores, que pueden derivarse de los anteriores, serán tratados en una sección posterior. La adaptación de la mayoría de los citados operadores, como se verá en las siguientes subsecciones dedicas a cada uno de ellos, tendrá en cuenta las particularidades de las MRCDs en lo que se refiere a identificación unívoca de tuplas e información asociada a las mismas.

6.3.1. Proyección

El operador de proyección permite crear nuevas MRCDs a partir de una MRCD determinada mediante la selección de un subconjunto de los atributos de la MRCD original. Dada una MRCD m y un subconjunto de sus atributos A, $A \subset A_m$, obtendremos una nueva MRCD similar a m pero eliminando de ésta los atributos no incluidos en A. Este proceso, conceptualmente sencillo y de aplicación directa en el modelo relacional clásico, ha de tener en cuenta las características distintivas de las MRCDs para aplicarse de forma coherente en el nuevo marco que define el modelo propuesto.

En una relación clásica cada tupla perteneciente a ésta es distinta del resto. Las tuplas se distinguen unas de otras por el valor de sus atributos. Se ha de tener en cuenta que al eliminar atributos en una relación es posible que algunas tuplas de ésta pasen a ser redundantes en la relación resultante. Al tener las relaciones naturaleza conjuntista, en la que se no se permiten duplicados, estas tuplas redundantes son descartadas, quedando sólo una de ellas en la citada relación resultante.

En la presente propuesta se han de considerar ciertas diferencias existente entre el caso clásico descrito anteriormente y el caso de las MRCDs. Las tuplas en la relación se distinguen en este último caso por su signatura. Al eliminar algún atributo que forme parte del conjunto de atributos signatura es posible que algunas tuplas de la MRCD original resulten redundantes en la MRCD resultante tras la aplicación del operador de proyección. La naturaleza de una MRCD es multiconjuntista y por tanto admite duplicados. Esto implica que las cardinalidades de un determinado grupo de tuplas redundantes habrán de ser agregadas en una sola tupla que las represente. Una agregación similar deberá ocurrir para el caso del grado de certidumbre asociado a las tuplas. Desafortunadamente, debido a que las tuplas se diferencian exclusivamente por los valores signatura de los atributos que forman parte de la signatura, se puede dar el caso de que las tuplas redundantes posean valores distintos para los atributos que no conformen la signatura (atributos con dominios de tipo multiconjuntivo o impreciso) o en parte de los valores de los atributos que no se considera en el valor signatura de éstos (atributos con dominios inciertos o complejos). En esos casos habrá que determinar cómo quedan reflejados dichos valores en la tupla que represente en la MRCD resultante a un determinado conjunto de tuplas redundantes.

Trataremos el problema descrito anteriormente en tres fases. En primer lu-

gar, determinaremos cuáles son los conjuntos de tuplas redundantes al restringir el conjunto de atributos de una MRCD. En segundo lugar, resolveremos el problema de determinar una tupla representante para un determinado conjunto de tuplas redundantes. Finalmente, determinaremos cómo se agregarán las cardinalidades y certidumbres de las tuplas redundantes.

6.3.1.1. Equivalencia entre tuplas con respecto a un subconjunto de atributos signatura

Para tratar de determinar los conjuntos de tuplas redundantes tras la aplicación del operador de proyección deberemos, en primer lugar, contar con una relación de equivalencia entre tuplas que nos indique, dada una tupla de una MRCD, si otra se puede considerar redundante si se restringe el conjunto de los atributos de dicha MRCD. Para ello, definiremos a continuación el concepto de *equivalencia entre tuplas con respecto a un subconjunto de sus atributos signatura.*

Definición 6.9 (Equivalencia entre tuplas con respecto a sus signaturas restringidas)**.** *Sean t_1 y t_2 dos tuplas pertenecientes a $\mathbb{T}_{(A,dom)}$, donde A es un conjunto de atributos y dom una función de dominio de la forma $dom : A \mapsto \mathbb{D}$. Sea B un subconjunto de A, $B \subseteq A$.*

Definiremos la relación de equivalencia entre dos tuplas t_1 y t_2 con respecto a sus signaturas restringidas a B*, notada como $t_1 \sim^{|B} t_2$, como se indica en la Ecuación 6.9.*

$$t_1 \sim^{|B} t_2 \iff sig^{|B}(t_1) = sig^{|B}(t_2) \quad \forall t_1, t_2 \in \mathbb{T}_{(A,dom)} | (A \subset \mathbb{A} \wedge dom : A \mapsto \mathbb{D}), \forall B \subseteq A \tag{6.9}$$

La relación de equivalencia descrita en la definición anterior permitirá dividir el conjunto de las tuplas de m en distintas clases de equivalencia. Dada la naturaleza de la relación de equivalencia definida anteriormente, las tuplas pertenecientes a una misma clase de equivalencia son redundantes entre sí cuando se reduce su conjunto de atributos a B. Este mecanismo, por tanto, será útil para agrupar las tuplas redundantes en un determinado conjunto. A continuación definiremos formalmente las clases de equivalencia descritas anteriormente.

Definición 6.10 (Clase de equivalencia de una tupla con respecto a su signatura restringida)**.** *Sea t una tupla perteneciente a $\mathbb{T}_{(A,dom)}$, donde A es un conjunto de atributos y dom una función de dominio de la forma $dom : A \mapsto \mathbb{D}$. Sea B un subconjunto de A, $B \subseteq A$. Sea T un conjunto de tuplas pertenecientes a $\mathbb{T}_{(A,dom)}$.*

Definiremos la clase de equivalencia de t dentro de T según su signatura restringida a B*, notada como $[t]_T^{|B}$, como se indica en la Ecuación 6.10*

$$[t]_T^{|B} = \{t' \in T | t' \sim^{|B} t\} \quad \forall t \in \mathbb{T}_{(A,dom)} | (A \subset \mathbb{A} \wedge dom : A \mapsto \mathbb{D}), \forall B \subseteq A, \forall T \subseteq \mathbb{T}_{(A,dom)} \tag{6.10}$$

6.3.1.2. Tupla representante de un conjunto de tuplas redundantes con respecto a un subconjunto de atributos signatura

Cada clase de equivalencia de una tupla, siguiendo la definición anterior, contendrá a dicha tupla y, si se da el caso, a las tuplas redundantes en relación con ésta. El siguiente paso consiste en determinar, dada una clase de equivalencia, cuál será la tupla que represente a ésta en la MRCD resultante al aplicar un operador de proyección que restrinja los atributos de la MRCD resultante a los incluidos en el subconjunto B.

Para el caso de clases de equivalencia compuestas por una sola tupla, la solución es evidente, pero determinar ésta para el caso de las clases de equivalencia que estén compuestas por más de una tupla no es algo directo. Obviamente, en el segundo caso, el valor en la tupla representante para los atributos asociados a dominios de tipo básico, y que por tanto forman parte de la signatura, será el valor único compartido por todo el conjunto de tuplas redundantes. Como se indicó anteriormente, la problemática se genera cuando se ha de determinar un valor para los atributos en cuyo dominio sea de tipo multiconjuntivo, impreciso, incierto o complejo. Para esos atributos, recordemos, es posible que las tuplas dentro de un conjunto de tuplas redundantes no tengan valores iguales. Nótese que esta desigualdad se puede dar tanto para el caso de atributos asociados a dominios no signatura, en cuyo caso no es sorprendente que dichos atributos no compartan un mismo valor, como para atributos asociados a dominio signatura. En este último caso, los atributos en esta situación compartirán obligatoriamente valor signatura pero podrán no compartir el valor concreto del dominio que tienen asociado (recuérdese que no todos los componentes de un determinado valor compuesto se emplean para determinar la signatura del mismo).

Para solventar el problema, en estos casos introduciremos el concepto de *valor agregado* de un subconjunto de valores de un mismo dominio. Éste será un valor del mismo dominio del que se extraen los valores del subconjunto y agregará de alguna forma a éstos. La forma en que se determinará dicho valor agregado dependerá de la semántica asociada al tipo de dominio complejo al que éstos pertenezcan.

Valores agregados en dominios imprecisos

En el caso de atributos con dominios imprecisos, los valores de dichos atributos indican el conjunto de los posibles valores que se estima serían el valor real del atributo. Parece lógico pensar que la tupla representante debería tener como valor para estos atributos una agregación de todas estas posibilidades. Tratándose de distribuciones de posibilidad modeladas como conjuntos difusos, esta agregación se realizará empleando el operador de unión. El resultado de esta unión será una distribución de posibilidad que incluye a todos los posibles valores para el atributo en el conjunto de tuplas redundantes.

No obstante, una unión ordinaria de dos valores cualesquiera de tipo impreciso puede resultar en una distribución de posibilidad que contenga distintos valores que se corresponden a un mismo valor signatura. Esta situación conduciría a la obtención de una MRCD que viole la FNPE. El resultando de dicho proceso de unión será un valor impreciso que respete la FNPE, por tanto, se habrá de definir una operador de unión entre valores imprecisos que tenga en cuenta dicha restricción. Un operador de unión de tal naturaleza, al unir dos distribuciones de posibilidad, deberá evitar la existencia de valores redundan-

tes, valores con un mismo valor signatura, en el resultando de dicha unión. En el caso de que estos existan, deberá sustituir éstos por un único valor que los represente. Dicho valor representante tendrá asociado un grado de certidumbre que será la combinación de los grados de certidumbre de los valores redundantes a los que represente.

El planteamiento anterior es similar al empleando para la eliminación de tuplas redundantes tras la aplicación del operador de proyección de una MRCD. Así pues, aplicaremos una solución similar al mismo.

En primer lugar, definiremos una relación de equivalencia entre los elementos de este tipo de valores. Dicha relación, que permitirá determinar si dos valores son redundantes, se define como se indica a continuación.

Definición 6.11 (Equivalencia entre valores con respecto a su valor signatura)**.** *Sea D un dominio signatura. Sean a y b dos valores del dominio D. Definiremos la relación de equivalencia entre dos valores a y b según su valor signatura, notada como $a \overset{sig}{\sim} b$, como se indica en la Ecuación 6.11.*

$$a \overset{sig}{\sim} b \iff sig(D,a) = sig(D,b) \qquad \forall a,b \in D | D \in \mathbb{D}^{sig} \tag{6.11}$$

Una vez definida la equivalencia entre valores redundantes, definiremos la clase de equivalencia, basada en dicha relación, para un valor dentro de un conjunto. Dicha clase permitirá dividir un determinado conjunto de valores en subconjuntos disjuntos cuyos miembros se consideran redundantes entre sí.

Definición 6.12 (Clase de equivalencia de un valor en un conjunto con respecto a su valor signatura)**.** *Sea D un dominio signatura. Sea V un subconjunto de valores de D Sea v un valor en V. La clase de equivalencia de v en V con respecto a su valor signatura es un subconjunto de V notado como $[v]_V$ y definido como se indica en la Ecuación 6.12.*

$$[v]_V = \{v' | (v \in V \wedge v' \overset{sig}{\sim} v)\} \qquad \forall V \subseteq D | D \in \mathbb{D}^{sig}, \forall v \in V \tag{6.12}$$

Una vez dispuesta la herramienta que nos permite agrupar los distintos valores redundantes, queda determinar cómo se calculará el representante de dichos valores. Esta tarea se pospondrá hasta que se determine la forma en que se agregarán los valores de los distintos tipos de dominios complejos que pueden actuar como dominio subyacente de un dominio impreciso. Independientemente del proceso de agregación de valores redundantes, para realizar la operación de unión evitando los citados valores redundantes deberemos tener la capacidad de determinar, dado un conjunto de valores, un nuevo conjunto en el que éstos han sido sustituidos el valor agregado que los representa. Con tal objeto definimos en *conjunto de valores agregados* como se indica a continuación.

Definición 6.13 (Conjunto de valores agregados)**.** *Sea D un dominio signatura y V un subconjunto de sus valores. El* conjunto de valores agregados *de V es un conjuntos de valores del dominio D notado como $G(D,V)$ y definido como indica la Ecuación 6.13. En la citada ecuación se hace uso de la expresión $agg(D,[v]_V)$ que representa el valor agregado que representa a un conjunto de valores redundantes $[v]_V$ de un determinado dominio D y que está definida como se indica en la Definición 6.16.*

$$G(D,V) = \{agg(D,[v]_V)\ v \in V\} \qquad \forall D \in \mathbb{D}^{sig}, \forall V \subseteq D \tag{6.13}$$

Finalmente, dispuestos los instrumentos que nos permitirán evitar la existencia de elementos redundantes en un valor de tipo impreciso, definimos el proceso de unión de este tipo de valores como se indica a continuación.

Definición 6.14 (Unión de valores imprecisos con respecto al valor signatura de sus elementos)**.** *Sea I un dominio complejo de tipo impreciso cuyo dominio subyacente es un dominio signatura. Sean $\tilde{a}$ y $\tilde{b}$ dos valores del dominio I.*

La unión de $\tilde{a}$ y $\tilde{b}$ con respecto al valor signatura de sus elementos es un valor del dominio I, notado como $(\tilde{a} \overset{sig}{\cup} \tilde{b})$, cuya función característica está definida como se indica en la Ecuación 6.14.

$$\mu_{(\tilde{a} \overset{sig}{\cup} \tilde{b})}(d) = \begin{cases} \bigotimes_{d' \in [d]_{supp(\tilde{a} \cup \tilde{b})}} \mu_{(\tilde{a} \cup \tilde{b})}(d') & si\ d \in G(sub(I), supp(\tilde{a} \cup \tilde{b})) \\ 0 & en\ otro\ caso \end{cases}$$

$$\forall \tilde{a}, \tilde{b} \in I | (I \in \mathbb{I} \wedge sub(I) \in \mathbb{D}^{sig}), \forall d \in sub(I) \tag{6.14}$$

Valores agregados en dominios multiconjuntivos difusos

En el segundo caso, en el caso de atributos con dominios multiconjuntivos, los valores de estos atributos son multiconjuntos difusos que contienen distintos elementos con distintas cardinalidades. Un buen candidato para representar una serie de multiconjuntos difusos podría ser uno tal que contenga todos los elementos de los originales. Esta operación se traduce en una unión aditiva de los diferentes multiconjuntos a agregar.

Al igual que ocurre en el caso anterior, el de la agregación de valores imprecisos, la aplicación de la unión aditiva para la agregación de este tipo de valores, sin tener en cuenta el valor signaturas de los elementos que los componen, puede resultar en un valor de tipo impreciso no admitido por la FNPE.

Para evitar esto se ha optado, como en el caso anterior, por la definición de un operador de unión que evite la existencia de valores redundantes en el multiconjunto resultante de la operación. De la misma forma que antes, se realizará el proceso de unión aditiva ordinario y se eliminarán del resultado aquellos valores redundantes, agregando éstos en un único representante. Dicho representante poseerá la cardinalidad agregada de cada uno de los elementos a los que sustituye, realizándose dicha agregación de forma aditiva de tal forma que se mantenga la semántica del operador.

Según las consideraciones iniciales, definimos el nuevo operador de unión aditiva como se indica a continuación.

Definición 6.15 (Unión aditiva de valores multiconjuntivos difusos con respecto al valor signatura de sus elementos)**.** *Sea M un dominio complejo de tipo multiconjuntivo difuso cuyo dominio subyacente es un dominio signatura. Sean $\hat{a}$ y $\hat{b}$ dos valores del dominio M.*

La unión aditiva de $\hat{a}$ y $\hat{b}$ con respecto al valor signatura de sus elementos es un valor del dominio M, notado como $(\hat{a} \overset{sig}{\uplus} \hat{b})$, cuya función característica está definida como se indica en la Ecuación 6.15.

$$\omega_{(\hat{a} \overset{sig}{\uplus} \hat{b})}(d) = \begin{cases} \sum_{d' \in [d]_{supp(\hat{a} \uplus \hat{b})}} \omega_{(\hat{a} \uplus \hat{b})}(d') & si\ d \in G(sub(M), supp(\hat{a} \uplus \hat{b})) \\ 0 & en\ otro\ caso \end{cases}$$

$$\forall \hat{a}, \hat{b} \in M | (M \in \mathbb{M} \wedge sub(M) \in \mathbb{D}^{sig}), \forall d \in sub(M) \quad (6.15)$$

Valores agregados en dominios inciertos

En tercer lugar, es necesario tratar el caso de los dominios de tipo incierto. Para agregar los valores de este tipo de dominio se han de agregar sus dos componentes. La primera de ellas es el grado de certidumbre asociado a cada valor incierto a agregar. Recordemos que los valores signatura de los valores de este tipo de dominio se componen exclusivamente del valor signatura de su valor base, ignorando, por tanto, en la composición de la signatura el grado de certidumbre asociado a éste. Puede darse el caso, por consiguiente, que sea necesario agregar en una sola tupla representante a un conjunto de tuplas que coinciden en signatura pero no coinciden en el grado de certidumbre asociado al valor del atributo en consideración. Parece lógico que al ser la tupla representante una agregación conjuntiva de las tuplas coincidentes en signatura, el conjunto los grados de certidumbre asociados al valor incierto del atributo en cuestión sean agregados de forma conjuntiva empleando una t-norma. El segundo componente a agregar será el valor base del valor incierto. En el caso de que el dominio incierto sea un dominio signatura, los valores a agregar compartirán el valor signatura de sus valores base. No obstante, éstos podrán no compartir un mismo valor base. Como se indicó anteriormente, varios valores de un dominio signatura pueden compartir un mismo valor signatura debido a que para la determinación de éste no se tienen en cuenta todos los componentes de los valores complejos. La agregación de estos valores base se realizará según la naturaleza del dominio al que pertenezcan. En el caso de que el dominio impreciso no sea signatura, no sorprenderá que los valores a agregar no compartan el valor base ni el grado de certidumbre asociado a éste. Sea cual sea el caso, la agregación de los grados de certidumbre y valores base seguirá el mismo procedimiento que el usado para el caso en el que el dominio impreciso sea un dominio signatura.

Valores agregados en dominios compuestos

En cuarto y último lugar, trataremos la agregación de valores de dominios de tipo compuesto. La agregación de un conjunto de este tipo de valores será un valor de tipo compuesto cuyos componentes se corresponden con la agregación de los componentes de cada uno de los valores del citado conjunto. El proceso de agregación de los componentes se realizará según el dominio de cada uno de éstos.

Valor agregado

Los procedimientos de agregación descritos anteriormente se relacionan de forma recursiva en tanto que los valores de un dominio pueden estar compuestos por valores de otros dominios subyacentes que se han de agregar. El caso base para esta recursión está en los valores de dominios de tipo básico ya que éstos no provienen de la agregación de valores de otros dominios.

Según lo anterior, definiremos el valor agregado de un conjunto de valores de un mismo dominio, que en el caso de dominios signatura comparten una misma signatura, como se indica a continuación.

Definición 6.16 (Valor agregado)**.** *Sea D un dominio complejo, $D \in \mathbb{D}$. Sea V un subconjunto no vacío de los valores del dominio D. En caso de que D sea un dominio signatura, V será de tal forma que los valores que contiene comparten un mismo valor signatura, esto es, $\forall d_1, d_2 \in V, sig(D, d_1) = sig(D, d_2)$.*

El valor agregado de un subconjunto de valores V de un dominio D, notado como $aggr(D, V)$, se define como se indica en la Ecuación 6.16.

$$
\begin{aligned}
&aggr(D,V) = \\
&\quad = \begin{cases}
d & si\ D \in \mathbb{B} \wedge V = \{d\} \\
d \in D | (cert(d) = \bigotimes_{v \in V} cert(v) \wedge \\
\quad val(d) = aggr(sub(D), \{val(v) | v \in V\})) & si\ D \in \mathbb{U} \\
\bigcup_{\tilde{v} \in V}^{sig} \tilde{v} & si\ D \in \mathbb{I} \\
d \in D | \forall a \in att(D), d(a) = \\
\quad aggr(dom_D(a), \{v(a) | v \in V\}) & si\ D \in \mathbb{O} \\
\biguplus_{\hat{v} \in V}^{sig} \hat{v} & si\ D \in \mathbb{M}
\end{cases}
\end{aligned}
$$

$$
\forall D \in \mathbb{D}, \forall V \subset D | (D \in \mathbb{D}^{sig} \Rightarrow \forall d_1, d_2 \in V, sig(D, d_1) = sig(D, d_2)) \tag{6.16}
$$

En la definición anterior, encontramos una restricción instrumental en el primer caso de la Ecuación 6.16. La restricción $V = \{d\}$ se cumple en todos los casos cuando el dominio D es de tipo básico. Para el caso de dominios signatura, como es el caso de los dominios básicos, se exige en la definición anterior que las signaturas de todos los elementos contenidos en V sean iguales. El valor signatura de un valor de un dominio básico es él mismo, por tanto V sólo puede estar compuesta por un único elemento. Esta restricción instrumental se aprovecha para obtener el único valor d que está contenido en V.

Tupla representante

Visto lo anterior, definiremos la tupla representante de un conjunto de tuplas redundantes, como se indica en la siguiente definición.

Definición 6.17 (Tupla representante de una clase de equivalencia con respecto a la signatura de una tupla dentro de un conjunto de tuplas)**.** *Sea t una tupla de $\mathbb{T}_{(A,dom)}$, donde A es un conjunto de atributos y dom una función de dominio de la forma $dom : A \mapsto \mathbb{D}$. Sea B un subconjunto de A, $B \subseteq A$. Sea T un conjunto de tuplas de $\mathbb{T}_{(A,dom)}$. Sea $[t]_T^{|B}$ la clase de equivalencia de la tupla t dentro de T según su signatura restringida a B.*

La tupla representante de la clase de equivalencia $[t]_T^{|B}$ es una tupla $r_{[t]_T^{|B}} \in \mathbb{T}_{(B,dom)}$ definida como indica la Ecuación 6.17.

$$r_{[t]_T^{|B}}(b) = aggr(dom(b), \{t'(b) | t' \in [t]_T^{|B}\})$$
$$\forall t \in \mathbb{T}_{(A,dom)} | (A \subset \mathbb{A} \wedge dom : A \mapsto \mathbb{D}), \forall B \subseteq A, \forall T \subseteq \mathbb{T}_{(A,dom)}, \forall b \in B \tag{6.17}$$

6.3.1.3. Conjunto de tuplas representantes

Según lo anterior, la MRCD resultante de aplicar el operador de proyección estará compuesta por las tuplas representantes de cada una de las clases de equivalencia en que se pueda dividir el conjunto de tuplas de la MRCD original. Por tanto, definiremos el concepto de *conjunto de tuplas representantes con respecto a un subconjunto de atributos* tal y como se indica a continuación.

Definición 6.18 (Conjunto de tuplas representantes con respecto a la signatura restringida de un conjunto de tuplas)**.** *Sea $T \subseteq \mathbb{T}_{(A,dom)}$ un conjunto de tuplas donde $A \subset \mathbb{A}$ es un conjunto de atributos y dom una función dominio de la forma $dom : A \mapsto \mathbb{D}$. Sea B un subconjunto de los atributos de A, $B \subseteq A$.*

El conjunto de tuplas representantes de las tuplas en T con respecto a su signatura restringida a B *se define como se indica en la Ecuación 6.18.*

$$R^{|B}(T) = \{r_{[t]_T^{|B}} | t \in T\} \qquad \forall T \subset \mathbb{T}_{(A,dom)} | (A \subset \mathbb{A} \wedge dom : A \mapsto \mathbb{D}), \forall B \subseteq A \tag{6.18}$$

En la definición anterior y en las que ésta se apoya, la determinación de representantes se restringe a un conjunto de tuplas T. Como en el caso que nos ocupa las MRCDs m poseen un multiconjunto difuso de tuplas T_m en lugar de un conjunto clásico, para aplicar esta definición tendremos que usar la imagen de T_m como conjunto clásico. Esta transformación implica una complicación de la notación. Por simplicidad en la misma, no indicaremos explícitamente tal conversión y, por tanto, emplearemos multiconjuntos difusos en el lugar que corresponde a conjuntos clásicos asumiendo de forma implícita ésta.

6.3.1.4. Agregación de cardinalidades y certidumbres de tuplas redundantes

Finalmente, determinado el conjunto de tuplas de la MRCD resultante de la proyección, resta determinar cuál será la cardinalidad y valor de certidumbre de dichas tuplas.

Con respecto a la cardinalidad parece lógico, teniendo en cuenta la semántica conjuntiva asociada a las tuplas representantes, que estas tengan asignada la suma de la cardinalidad de las tuplas redundantes a las que representan. Además, de esta forma la relación resultante conservará la cardinalidad de la relación original.

En lo que respecta a la certidumbre, se ha de tener en cuenta la tupla representante agrega los valores de los atributos de las tuplas redundantes. La naturaleza de esta agregación, conjuntiva en este caso, induce el empleo de una t-norma para determinar a partir de cada uno de los componentes de la agregación la certidumbre asociada al resultado de ésta.

6.3.1.5. Definición del operador de proyección para multirrelaciones complejas difusas

Con todo lo anterior, se está en disposición de definir formalmente el operador de proyección para el caso de las MRCDs. Ésta es como se indica a continuación.

Definición 6.19 (Proyección)**.** *Sea m una MRCD y A un subconjunto de sus atributos, $A \subseteq A_m$. La proyección de m sobre el subconjunto de sus atributos A, notada como $\pi_A(m)$, es una MRCD definida de la forma que se indica en la Ecuación 6.19.*

$$\pi_A(m) = (\mathcal{H}_{\pi_A(m)}, \mathcal{B}_{\pi_A(m)}) \tag{6.19}$$

La cabecera y el cuerpo de $\pi_A(m)$ están definidos como se indica en las Ecuaciones 6.20 y 6.21 respectivamente.

$$\mathcal{H}_{\pi_A(m)} = (A, dom_{\pi_A(m)}) \tag{6.20}$$

$$\mathcal{B}_{\pi_A(m)} = (T_{\pi_A(m)}, U_{\pi_A(m)}) \tag{6.21}$$

Cada uno de los componentes de cabecera y cuerpo de $\pi_A(m)$ se definen como sigue:

- *La función de dominio de la MRCD resultante, $dom_{\pi_A(m)}$, es una restricción de la función de dominio dom_m al subconjunto de atributos A tal y como se indica en la Ecuación 6.22.*

$$dom_{\pi_A(m)} = dom_m^{|A} \tag{6.22}$$

- *El multiconjunto de tuplas de la MRCD resultante, $T_{\pi_A(m)}$, se define según la función características que se indica en la Ecuación 6.23.*

$$\omega_{T_{\pi_A(m)}}(t) = \begin{cases} \sum\limits_{t' \in T_m | r_{[t']_{T_m}^{|A}} = t} \omega_{T_m}(t') & si\ t \in R^{|A}(T_m) \\ \{1.0/0\}_2 & en\ otro\ caso \end{cases} \qquad \forall t \in \mathbb{T}_{\pi_A(m)} \tag{6.23}$$

- *La función de certidumbre de la MRCD resultante, $U_{\pi_A(m)}$, tendrá la forma que se indica en la Ecuación 6.24.*

$$U_{\pi_A(m)}(t) = \bigotimes_{t' \in T_m | t = r_{[t']^{|A}_{T_m}}} U_m(t') \qquad \forall t \in T_{\pi_A(m)} \tag{6.24}$$

El siguiente ejemplo ilustrará el funcionamiento del operador de proyección definido anteriormente.

Ejemplo 6.1. *Para ilustrar la aplicación del operador de proyección sobre MRCDs, aplicaremos éste sobre una MRCD derivada de la MRCD r definida en el Ejemplo 5.17.*

La nueva MRCD, que denominaremos r', reorganiza algunos de los atributos originales de r de tal forma que agrupa éstos en un atributo asociado a un dominio complejo. Este cambio permitirá mostrar la forma en que se agregan los valores de distintos dominios. Además, con objeto de simplificar el ejemplo, en r' se han desagregado los datos temporales de un evento en un dato de tipo complejo y se ha suprimido algún componente de los mismos.

Con todo esto, r' resulta en una MRCD con una cabecera compuesta por un conjunto de atributos definido como se indica en la Ecuación 6.25 y una función dominio de la forma que se muestra en la Ecuación 6.26. En dicha ecuación, se emplean dos nuevos dominios, D_{meses} y $D_{requisitos}$. El dominio D_{meses} es un dominio básico que está compuesto por los números naturales en el rango $[1, 12]$ que representan los meses del año. En lo que se refiere al dominio $D_{requisitos}$, éste es un dominio compuesto cuyo conjunto de atributos se muestra en la Ecuación 6.27 y cuya función de dominio está definida según se indica en la Ecuación 6.28. Los valores de este dominio representarán el conjunto de requisitos para la celebración de una reunión paralela dentro de un determinado evento.

$$A_{r'} = \{mes, año, requisitos\} \tag{6.25}$$

$$dom_{r'}(a) = \begin{cases} D_{meses} & si\ a = mes \\ D_{nat} & si\ a = año \\ D_{requisitos} & si\ a = requisitos \end{cases} \qquad \forall a \in A_{r'} \tag{6.26}$$

$$A_{requisitos} = \{asistentes, catering, intérpretes\} \tag{6.27}$$

$$dom_{requisitos}(a) = \begin{cases} D_{asist} & si\ a = asistentes \\ D_{conf} & si\ a = catering \\ D_{inter} & si\ a = intérpretes \end{cases} \qquad \forall a \in A_{requisitos} \tag{6.28}$$

Finalmente, el cuerpo de la MRCD r' estará definido según se indica en la Tabla 6.1.

Una vez definida la MRCD r', apliquemos sobre ella el operador de proyección. En el caso del presente ejemplo, pretendemos crear una nueva MRCD que derive de r' mediante la eliminación del atributo mes*, la cual se corresponda*

$U_{r'}$	$T_{r'}$	$t_{r'}$					
		mes	año	requisitos			
				asistentes	catering	intérpretes	
1.0	$\{1.0/2, 0.6/5, 0.3/6\}_c$	6	2010	$entre(20, 25)$	$\langle 1.0, \text{sí}\rangle$	$\{1.0/1, 0.75/3, 0.5/8\}_c$ $\{1.0/1, 0.75/2, 0.5/4\}_c$	Inglés Alemán
1.0	$\{0.6/3\}_c$	9	2010	$aprox(40)$	$\langle 0.5, \text{no}\rangle$	$\{0.5/10\}_c$	Inglés
0.5	$\{1.0/2\}_c$	11	2010	20	$\langle 0.5, \text{sí}\rangle$	$\{1.0/3, 0.5/6\}_c$ $\{1.0/1, 0.75/5\}_c$ $\{0.5/4\}_c$	Inglés Francés Italiano
0.5	$\{0.3/4\}_c$	4	2011	$hasta(5)$	$\langle 1.0, \text{no}\rangle$	$\emptyset_{inter}$	

Tabla 6.1: Cuerpo de la MRCD r'

con $\pi_{\{año,requisitos\}}(r')$*. El conjunto de atributos de esta MRCD se corresponde con el mostrado en la Ecuación 6.29. En lo que respecta a la función dominio, ésta es de la forma que se muestra en la Ecuación 6.30.*

$$A_{\pi_{\{año,requisitos\}}(r')} = \{año, requisitos\} \tag{6.29}$$

$$dom_{\pi_{\{año,requisitos\}}(r')}(a) = \begin{cases} D_{nat} & si\ a = año \\ D_{requisitos} & si\ a = requisitos \end{cases}$$

$$\forall a \in A_{\pi_{\{año,requisitos\}}(r')} \tag{6.30}$$

En lo relativo al cuerpo de $\pi_{\{año,requisitos\}}(r')$*, éste está definido según se indica en la Tabla 6.2.*

Obsérvese que el número de tuplas de $\pi_{\{año,requisitos\}}(r')$ *es menor que el de la MRCD original* r'*. Este cambio se debe a que algunas de las tuplas de* r' *han sido agregadas. Concretamente, al eliminar el atributo* mes *de* r'*, las tuplas primera y tercera (siempre según el orden en que éstas se muestran en la Tabla 6.1) coinciden en signatura y por tanto se consideran redundantes. Esta redundancia se solventa mediante la introducción en el cuerpo de* $\pi_{\{año,requisitos\}}(r')$ *de la tupla representante de la clase de equivalencia a la que pertenecen ambas tuplas en sustitución de éstas. Esta tupla representante se corresponde con la primera (según el orden en que se muestran la tuplas en la Tabla 6.2).*

Como se indicó anteriormente, los valores de la tupla representante están constituidos por la agregación de los valores de las tuplas a las que ésta representa. En particular, obsérvese que el número de asistentes se corresponde a la unión de las distribuciones de posibilidad originales (siempre considerando la signatura de sus elementos, como se indicó al describir el proceso de agregación de valores imprecisos), la confirmación del requerimiento del servicio de catering posee un grado de certidumbre asociado que se corresponde con la agregación mediante una t-norma de los originales, y el multiconjunto que representa los servicios de interpretación requeridos se corresponde con la unión conjuntiva de los originales (igualmente, considerando la signatura de los elementos que componen los multiconjuntos participantes). Asimismo, la cardinalidad y grado de certidumbre asociados a la citada tupla se corresponde con la agregación en forma de suma (en el caso de la cardinalidad) y empleando una t-norma (en el caso del grado de certidumbre). Por simplicidad, en el anterior ejemplo se ha empleado como t-norma la función mínimo.

6.3.2. Selección

El operador de selección permite obtener una nueva MRCD a partir de una MRCD original de tal forma que la resultante contenga exclusivamente aquellas tuplas que satisfacen una determinada condición.

Esta condición se expresará mediante un predicado que, debido a que las MRCD permiten la pertenencia graduada de las tuplas, puede ser difuso. La cardinalidad difusa de cada una de las tuplas en la MRCD resultante se verá afectada en función del grado de satisfacción del predicado difuso que se emplea como condición. El grado de cada una de las posibles cardinalidades de la

$U_{\pi_{\{\text{año,requisitos}\}}(r')}$	$T_{\pi_{\{\text{año,requisitos}\}}(r')}$	$t_{\pi_{\{\text{año,requisitos}\}}(r')}$				
		año	requisitos			
			asistentes	catering	intérpretes	
0.5	$\{1.0/4, 0.6/7, 0.3/8\}_c$	2010	$entre(20, 25)$	$\langle 0.5, \text{sí}\rangle$	$\{1.0/4, 0.75/6, 0.5/14\}_c$	Inglés
					$\{1.0/1, 0.75/2, 0.5/4\}_c$	Alemán
					$\{1.0/1, 0.75/5\}_c$	Francés
					$\{0.5/4\}_c$	Italiano
1.0	$\{0.6/3\}_c$	2010	$aprox(40)$	$\langle 0.5, \text{no}\rangle$	$\{0.5/10\}_c$	Inglés
0.5	$\{0.3/4\}_c$	2011	$hasta(5)$	$\langle 1.0, \text{no}\rangle$	$\emptyset_{inter}$	

Tabla 6.2: Cuerpo de $\pi_{\{\text{año,requisitos}\}}(r')$

tupla se agregará, de forma conjuntiva, al grado de satisfacción de ésta para la condición de selección.

Formalmente, definiremos el operador de agregación como se indica a continuación.

Definición 6.20 (Selección)**.** *Sea m una MRCD. Sea $\tilde{p}$ un predicado difuso cuya función característica es de la forma $\mu_{\tilde{p}} : T_m \mapsto [0,1]$.*

La selección de las tuplas de m según el predicado difuso $\tilde{p}$, notada como $\sigma_{\tilde{p}}(m)$, es una MRCD definida de la forma que se indica en la Ecuación 6.31.

$$\sigma_{\tilde{p}}(m) = (\mathcal{H}_{\sigma_{\tilde{p}}(m)}, \mathcal{B}_{\sigma_{\tilde{p}}(m)}) \tag{6.31}$$

La cabecera y cuerpo de $\sigma_{\tilde{p}}(m)$ son de la forma que indican, respectivamente, las ecuaciones 6.32 y 6.33.

$$\mathcal{H}_{\sigma_{\tilde{p}}(m)} = (A_m, dom_m) \tag{6.32}$$

$$\mathcal{B}_{\sigma_{\tilde{p}}(m)} = (T_{\sigma_{\tilde{p}}(m)}, U_{\sigma_{\tilde{p}}(m)}) \tag{6.33}$$

Los componentes del cuerpo de $\sigma_{\tilde{p}}(m)$ definido anteriormente se definen cómo se indica a continuación:

- *El conjunto de tuplas de $\sigma_{\tilde{p}}(m)$, $T_{\sigma_{\tilde{p}}(m)}$, se define según la función característica que se indica en la Ecuación 6.34. En dicha ecuación, la función ord está definida como se indica en la Ecuación 6.35.*

$$\omega_{T_{\sigma_{\tilde{p}}(m)}}(t) = \begin{cases} \omega_{T_m}(t) \boxtimes ord(\mu_{\tilde{p}}(t)) & si\ t \in T_m \\ \{1.0/0\}_2 & en\ caso\ contrario \end{cases} \qquad \forall t \in \mathbb{T}_{\sigma_{\tilde{p}}(m)} \tag{6.34}$$

$$ord(\alpha) = \{1/0, \alpha/1\} \qquad \forall \alpha \in [0,1] \tag{6.35}$$

- *La función de certidumbre $U_{\sigma_{\tilde{p}}(m)}$ es de la forma que se indica en la Ecuación 6.36.*

$$U_{\sigma_{\tilde{p}}(m)}(t) = U_m(t) \qquad \forall t \in T_{\sigma_{\tilde{p}}(m)} \tag{6.36}$$

Nótese que en la Ecuación 6.36 se emplea la función de certidumbre U_m sobre tuplas pertenecientes a $T_{\sigma_{\tilde{p}}(m)}$. Recordemos que el dominio de U_m está compuesto por las tuplas pertenecientes al multiconjunto T_m y no $T_{\sigma_{\tilde{p}}(m)}$. No obstante, este uso es lícito ya que, según la definición de $T_{\sigma_{\tilde{p}}(m)}$ en la Ecuación 6.34, se puede concluir que $T_{\sigma_{\tilde{p}}(m)} \subseteq T_m$.

A continuación, se muestra un ejemplo de uso del operador de selección.

Ejemplo 6.2. *Supongamos que el centro de negocios está valorando alquilar nuevos salones para la celebración de sesiones paralelas asociadas a eventos que cuentan con una capacidad de aproximadamente 35 personas. Se estima que el alquiler de estos salones es rentable para eventos con al menos 25 asistentes aproximadamente. El margen con el que trabaja esta aproximación es de* ± 5

asistentes	$\mu_{[20,25,35,40]}(asistentes)$
$entre(20,25)$	1.0
$aprox(40)$	0.4
20	0.0
$hasta(5)$	0.0

Tabla 6.3: Grados de pertenencia al conjunto difuso $[20,25,35,40]$ de los diferentes valores del atributo *asistentes* en la MRCD r

personas, de tal forma que los salones admiten reuniones de más de 35 personas perdiendo un poco de comodidad hasta un límite de 40 asistentes, situación que ya no se considera aceptable. De la misma forma, la rentabilidad decrece ligeramente si se celebran reuniones con menos de 25 asistentes, hasta un límite de 20 asistentes en que la perdida no será tolerable. El equipo directivo desea obtener un listado de los posibles eventos cuyas sesiones paralelas podrían ser celebradas en este tipo de salones ajustándose a las condiciones anteriores. Además, dicho listado deberá indicar cuántos de estos salones serán necesarios alquilar para cada evento.

Para crear el informe anterior, debemos obtener una MRCD derivada de r (MRCD definida en el Ejemplo 5.17) en la que figuren las tuplas que para el atributo asistentes *poseen un valor entre 25 y 35 aproximadamente. La anterior condición puede ser modelada empleando el predicado difuso $\tilde{p}$ que se indica en la Ecuación 6.37. Recuérdese que los valores del dominio asociado al atributo* asistentes *en r son distribuciones de posibilidad modeladas como conjuntos difusos. Esta última forma es la que se emplea en la definición de $\tilde{p}$. El componente $[20,25,35,40]$ en la Ecuación 6.37 representa al conjunto difuso de diferentes cantidades de asistentes para las cuales el alquiler de los salones de reunión es admisible. La función característica de este conjunto difuso es una función trapezoidal definida tal y como se indicó en la Ecuación 2.14. Finalmente, se ha de hacer notar que el predicado $\tilde{p}$ se corresponde con una medida de posibilidad.*

$$\tilde{p}(t) \iff \exists d \in sub(dom_r(asistentes))|(d \in t(asistentes) \wedge d \in [20,25,35,40]) \quad \forall t \in T_r \qquad (6.37)$$

Una vez definido el predicado que permitirá filtrar a las tuplas de r que cumplen la restricción descrita anteriormente (cuyo grado de satisfacción se muestra en la Tabla 6.3), el listado requerido se obtendrá del resultado de la operación $\sigma_{\tilde{p}}(r)$. Éste es una MRCD con cabecera similar a la de r, ya que el operador de selección no modifica la cabecera de la MRCD resultando. En lo que respecta al cuerpo de $\sigma_{\tilde{p}}(r)$, $\mathcal{B}_{\sigma_{\tilde{p}}(r)}$ está definido según se indica en la Tabla 6.4.

Obsérvese que en el cuerpo de $\sigma_{\tilde{p}}(r)$ aparecen únicamente las dos primeras tuplas de r (siempre referido al orden en que se muestran en la Tabla 5.10). La cardinalidad de la primera tupla es idéntica a la que ésta posee en r ya que la tupla satisface completamente (con grado 1.0) el predicado $\tilde{p}$. En cambio, la cardinalidad de la segunda tupla no es la misma que en r. Ésta se ve afectada

$U_{\sigma_{\tilde{p}}(r)}$	$T_{\sigma_{\tilde{p}}(r)}$	$t_{\sigma_{\tilde{p}}(r)}$						
		celebración			asistentes	catering	intérpretes	
		fecha	inicio	duración				
1.0	$\{1.0/2, 0.6/5, 0.3/6\}_c$	4-6-2010	9:00	5	$entre(20, 25)$	$\langle 1.0, \text{sí}\rangle$	$\{1.0/1, 0.75/3, 0.5/8\}_c$ $\{1.0/1, 0.75/2, 0.5/4\}_c$	Inglés Alemán
0.5	$\{0.4/3\}_c$	14-10-2010	17:00	3	$aprox(40)$	$\langle 0.5, \text{no}\rangle$	$\{0.5/10\}_c$	Inglés

Tabla 6.4: Cuerpo de $\sigma_{\tilde{p}}(r)$

ya que dicha tupla satisface parcialmente, concretamente con un grado 0.4*, el predicado* $\tilde{p}$*. Debido a esto, los grados de posibilidad en su cardinalidad quedan restringidos a un máximo de* 0.4*. En lo que respecta a las tuplas de* r *no incluidas en* $\sigma_{\tilde{p}}(r)$*, estas quedan descartadas de la MRCD resultante ya que no satisfacen en grado alguno la condición establecida en* $\tilde{p}$ *y, por tanto, su cardinalidad asociada en* $\sigma_{\tilde{p}}(r)$ *resulta en valor* $\{1.0/0\}_2$*.*

Las cardinalidades de las tuplas que conforman el resultado de la operación poseen una semántica híbrida que refleja la posibilidad de que se celebre un número concreto de reuniones de trabajo (la semántica original) y la posibilidad de que la celebración de dichas reuniones en los locales de alquiler resulte rentable y cómoda para los asistentes (semántica introducida por la condición establecida). El análisis de estas cardinalidades ayudará al equipo del centro de negocios a determinar el número de salones que será conveniente alquilar en firme, reservar formalmente y apalabrar en función del grado de posibilidad asignado.

6.3.3. Renombrado

El operador de renombrado permite derivar de una MRCD otra similar a la anterior a excepción del nombre de los atributos, siendo éstos (o un subconjunto de ellos) sustituidos por otros distintos a los de la MRCD original. Este operador no fue incluido originalmente en el álgebra relacional propuesta por Codd. No obstante, los inventores de ISBL (Information Systems Base Language) [141], un lenguaje de consulta basado en el álgebra relacional para uno de los primeros prototipos de SGBD basado en el modelo relacional desarrollado por IBM a finales de la década de 1970, demostraron la necesidad de su inclusión en el subconjunto de operadores básicos del álgebra relacional [77].

La trasposición de este operador al caso de las MRCDs es directa y su definición formal es la que se presenta a continuación.

Definición 6.21 (Renombrado)**.** *Sea* m *una MRCD. Sea* $A \subseteq A_m$ *un subconjunto de sus atributos. Sea* A' *un conjunto de nombres de atributos,* $A' \subset \mathbb{A}$*, tal que* $A' \cap A_m \neq \emptyset$ *y* $|A| = |A'|$*. Sea* s *una función biyectiva de la forma* $s : A \mapsto A'$ *que denominaremos* función de renombrado*.*

El renombrado del conjunto de atributos A *por un nuevo conjunto de atributos* A' *según una función de renombrado* s *en la MRCD* m *es una MRCD notada como* $\rho_s(m)$ *y definida según se indica en la Ecuación 6.38.*

$$\rho_s(m) = (\mathcal{H}_{\rho_s(m)}, \mathcal{B}_{\rho_s(m)}) \tag{6.38}$$

La cabecera y cuerpo de $\rho_s(m)$ *tienen la forma que indica, respectivamente, las ecuaciones 6.39 y 6.40.*

$$\mathcal{H}_{\rho_s(m)} = (A_{\rho_s(m)}, dom_{\rho_s(m)}) \tag{6.39}$$

$$\mathcal{B}_{\rho_s(m)} = (T_{\rho_s(m)}, U_{\rho_s(m)}) \tag{6.40}$$

Los componentes de la cabecera y cuerpo definidos anteriormente están definidos como sigue:

- *El conjunto de atributos de* $\rho_s(m)$*,* $A_{\rho_s(m)}$*, se define como se indica en la Ecuación 6.41.*

$$A_{\rho_s(m)} = (A_m - A) \cup A' \tag{6.41}$$

- *La función de dominio de $\rho_s(m)$ es de la forma está definida según se indica en la Ecuación 6.42. En dicha ecuación, s^{-1} representa la función inversa de s. Esta función es de la forma $s^{-1} : A' \mapsto A$ y está definida como se indica en la Ecuación 6.43.*

$$dom_{\rho_s(m)}(a) = \begin{cases} dom_m(a) & si\ a \in (A_m - A) \\ dom_m(s^{-1}(a)) & si\ a \in A' \end{cases} \qquad \forall a \in A_{\rho_s(m)} \tag{6.42}$$

$$s^{-1}(a') = a$$
$$\forall s : A \mapsto A' | (\forall a' \in A', (\exists! a \in A | (s(a) = a'))),$$
$$\forall a' \in A' | (\exists! a \in A, s(a) = a') \tag{6.43}$$

- *El multiconjunto de tuplas de $\rho_s(m)$, $T_{\rho_s(m)}$, está definido según la función característica que se indica en la Ecuación 6.44. En dicha ecuación, t^s_m es una tupla de la forma que se indica en la Ecuación 6.45.*

$$\omega_{T_{\rho_s(m)}}(t) = \omega_{T_m}(t^s_m) \qquad \forall t \in \mathbb{T}_{\rho_s(m)} \tag{6.44}$$

$$t^s_m(a) = \begin{cases} t(a) & si\ a \in (A - A') \\ t(s^{-1}(a)) & si\ a \in A' \end{cases}$$

$$\forall m \in \Delta, \forall s : A \mapsto A' | (A \subset A_m \ \wedge\ A' \subset \mathbb{A} \ \wedge\ A' \cap A = \emptyset \wedge$$
$$\forall a' \in A' : (\exists! a \in A | s(a) = a')), \forall t \in \mathbb{T}_{\rho_s(m)}, \forall a \in A_{\rho_s(m)} \tag{6.45}$$

- *La función de certidumbre de la MRCD resultante, $U_{\rho_s(m)}$, es de la forma que se indica en la Ecuación 6.46.*

$$U_{\rho_s(m)}(t) = U_m(t^s_m) \qquad \forall t \in T_{\rho_s(m)} \tag{6.46}$$

Cuando se aplica el operador anterior no se suele definir en la práctica una función de renombrado como tal. Por los citados motivos prácticos, se emplea directamente la notación $\rho_{(a'_1,a'_2,...,a'_n)\leftarrow(a_1,a_2,...,a_n)}(m)$. Esta notación presupone la existencia de una función de renombrado s de la forma $s : \{a_1, a_2, ..., a_n\} \mapsto \{a'_1, a'_2, ..., a'_n\}$ tal que $s(a_i) = a'_i$. Por tanto, $\rho_{(a'_1,a'_2,...,a'_n)\leftarrow(a_1,a_2,...,a_n)}(m)$ es equivalente a la aplicación de la expresión algebraica $\rho_s(m)$.

6.3.4. Producto cartesiano

El producto cartesiano en el álgebra relacional clásica es una operación que permite crear, a partir de dos relaciones, una nueva relación cuyas tuplas se corresponden con la combinación de cada posible par de tuplas formado por una tupla de la primera relación y otra de la segunda. Esta combinación de pares de tuplas se realiza en forma de concatenación, de tal forma que cada tupla de la relación resultante tendrá tanto los atributos propios de las tuplas de una de las relaciones originales como de la otra. Los valores asociados a dichos atributos para cada una de las tuplas de la relación resultante son iguales a los valores que poseen las tuplas originales. Debido a la forma en que se combinan las tuplas, la aplicación del producto cartesiano está restringida a un par de relaciones que no tengan atributos en común.

En el caso de las MRCDs, la operación de producto cartesiano es esencialmente igual a la descrita. No obstante, en este caso se deberán combinar, además, las cardinalidades y el grado de certidumbre de las tuplas de las MRCDs originales.

Las cardinalidades de las tuplas originales serán combinadas mediante el producto de las mismas. De esta forma, la cardinalidad de cada tupla de la MRCD resultante es equivalente a la que ésta poseería el producto de las MRCD originales en su versión no compacta.

En lo que respecta a los grados de certidumbre de las tuplas originales, éstos se agregarán mediante la aplicación de una t-norma ya que la tuplas de las MRCD resultante son combinaciones cuya semántica asociada es conjuntiva.

Según lo anterior, la definición formal del producto cartesiano de dos MRCDs es como se indica a continuación.

Definición 6.22 (Producto cartesiano)**.** *Sean m_1 y m_2 dos MRCDs tales que éstas no comparten ningún atributo, $A_{m_1} \cap A_{m_2} = \emptyset$.*

El producto cartesiano de m_1 y m_2, notado como $(m_1 \times m_2)$, es una MRCD que está definida de la forma indicada en la Ecuación 6.47.

$$(m_1 \times m_2) = (\mathcal{H}_{(m_1 \times m_2)}, \mathcal{B}_{(m_1 \times m_2)}) \tag{6.47}$$

La cabecera y cuerpo de $(m_1 \times m_2)$ responde a lo indicado en las ecuaciones 6.48 y 6.49 respectivamente.

$$\mathcal{H}_{(m_1 \times m_2)} = (A_{(m_1 \times m_2)}, dom_{(m_1 \times m_2)}) \tag{6.48}$$

$$\mathcal{B}_{(m_1 \times m_2)} = (T_{(m_1 \times m_2)}, U_{(m_1 \times m_2)}) \tag{6.49}$$

Los componentes de la cabecera y el cuerpo anteriores son como se indica a continuación:

- *El conjunto de los atributos de $(m_1 \times m_2)$ se define como se indica en la Ecuación 6.50.*

$$A_{(m_1 \times m_2)} = A_{m_1} \cup A_{m_2} \tag{6.50}$$

- *La función dominio de $(m_1 \times m_2)$, $dom_{(m_1 \times m_2)}$, es una función de la forma indicada en la Ecuación 6.51.*

$$dom_{(m_1 \times m_2)}(a) = \begin{cases} dom_{m_1}(a) & si\ a \in A_{m_1} \\ dom_{m_2}(a) & si\ a \in A_{m_2} \end{cases} \qquad \forall a \in A_{(m_1 \times m_2)} \quad (6.51)$$

- *La función característica del multiconjunto difuso de tuplas de* $(m_1 \times m_2)$ *se define como indica la Ecuación 6.52.*

$$\omega_{T_{(m_1 \times m_2)}} = \omega_{T_{m_1}}(t^{|A_{m_1}}) \boxtimes \omega_{T_{m_2}}(t^{|A_{m_2}}) \qquad \forall t \in \mathbb{T}_{(m_1 \times m_2)} \quad (6.52)$$

- *La función de certidumbre de* $(m_1 \times m_2)$, $U_{(m_1 \times m_2)}$, *se define como se indica en la Ecuación 6.53.*

$$U_{(m_1 \times m_2)} = U_{m_1}(t^{|A_{m_1}}) \otimes U_{m_2}(t^{|A_{m_2}}) \qquad \forall t \in T_{(m_1 \times m_2)} \quad (6.53)$$

Ilustraremos el funcionamiento del operador anteriormente definido empleando el siguiente ejemplo.

Ejemplo 6.3. *Supongamos que el centro de negocios, además de la información relativa a las características de las reservas de eventos, recopila información sobre el material de papelería y equipamiento audiovisual de los que el cliente desea disponer en las salas en que se realizarán las reuniones paralelas asociadas a cada uno de ellos.*

Supongamos también, por simplicidad, que empleamos la fecha de celebración de cada evento para identificar el mismo, aunque ello implique restringir el número de eventos que el centro puede organizar cada día a uno. La información relativa al material requerido puede representarse en una MRCD en la que en cada tupla se asocie un determinado material a un determinado evento (identificado unívocamente, como se ha indicado anteriormente, por su fecha).

La cardinalidad asociada a cada una de las tuplas representará el número de unidades necesarias de dicho material para cada una de las sesiones paralelas asociadas al evento. Esta cardinalidad, al ser difusa, permitirá (al igual que ocurría con las propias reservas) indicar dicho número de unidades de forma imprecisa. Así, se permitirá indicar tanto el número de unidades como el grado de confirmación por parte del cliente de esta estimación. La codificación de los diferentes grados de imprecisión se realizará de la misma forma que las reservas, basándonos en la correspondencia que se indicó en la Tabla 5.6.

El grado de certidumbre asociado a cada tupla indicará el grado de confirmación de los datos referidos acerca del pedido de material que ésta representa. De esta forma, se puede indicar que los datos han sido confirmados por el cliente o si se trata de una estimación pendiente de confirmar del comercial que lleva su cuenta. La semántica de los distintos grados de confirmación será similar a la aplicada a los datos relativos a las reservas, por lo que se empleará la correspondencia que se indicó en la Tabla 5.7.

Basándonos en la descripción anterior, podemos definir una MRCD, que denominaremos m, *representando los pedidos de material. Ésta será de tal forma que su cabecera contenga un conjunto de atributos como el mostrado en la Ecuación 6.54. El atributo* fecha *se corresponde con la fecha del evento para el cual es necesario dicho material en cada una de las salas en las que se celebrarán*

U_m	T_m	t_m	
		fecha	material
1.0	$\{1.0/1\}_c$	4-6-2010	videoproyector
1.0	$\{1.0/20, 0.6/25, 0.3/30\}_c$	4-6-2010	cuaderno
0.5	$\{0.3/1\}_c$	14-9-2010	videoproyector
0.5	$\{0.3/5\}_c$	9-12-2010	cuaderno

Tabla 6.5: Cuerpo de la MRCD m

sus reuniones paralelas asociadas. El atributo material *indicará el material que es requerido. La función dominio de esta MRCD tendrá la forma que se indica en la Ecuación 6.55. En dicha ecuación, se emplea el dominio D_{fecha}, que fue definido en el Ejemplo 5.3, y el nuevo dominio $D_{material}$. Este último es un dominio complejo básico cuyos valores representan los distintos materiales y medios audiovisuales que puede proveer la agencia. Dicho dominio está definido como se indica en la Ecuación 6.56, donde, por motivos de brevedad, se han indicado sólo dos valores del dominio.*

$$A_m = \{fecha, material\} \tag{6.54}$$

$$dom_m(a) = \begin{cases} D_{fecha} & si\ a = fecha \\ D_{material} & si\ a = material \end{cases} \qquad \forall a \in A_m \tag{6.55}$$

$$D_{material} = \{videoproyector, cuaderno, \ldots\} \tag{6.56}$$

En lo que respecta al cuerpo de m, éste estará definido, para el propósito del presente ejemplo, como se indica en la Tabla 6.5.

Una vez disponemos de una MRCD que contiene las reservas de eventos y otra que contiene los requisitos materiales necesarios para la celebración de cada una de las reuniones paralelas de dichos eventos, esta información se puede cruzar para obtener los requisitos materiales totales de cada evento. El cruce de la información contenida en las dos MRCDs se realizará empleando el operador de producto cartesiano sobre ambas relaciones.

Antes de aplicar dicho operador, será necesario introducir ciertas modificaciones a las relaciones participantes. En primer lugar, por motivos de brevedad, en lugar de emplear la MRCD r tal y como está definida en el Ejemplo 5.17, emplearemos una versión reducida de la misma en la que se mantiene exclusivamente como único atributo presente la fecha en la que se celebrará el evento. Esta MRCD, que denominaremos r'' se deriva de la original de la forma que se indica en la Ecuación 6.57. El cuerpo de r'' es de la forma que se indica en la Tabla 6.6.

$$r'' = \pi_{\{fecha\}}(\phi_{celebración}(r)) \tag{6.57}$$

En segundo lugar, se ha de tener en cuenta que la aplicación del operador de producto cartesiano exige a las MRCDs participantes que sus conjuntos de

$U_{r''}$	$T_{r''}$	$t_{r''}$
		fecha
1.0	$\{1.0/2, 0.6/5, 0.3/6\}_c$	4-6-2010
0.5	$\{0.6/3\}_c$	14-9-2010
1.0	$\{1.0/2\}_c$	21-11-2010
0.5	$\{0.3/4\}_c$	9-12-2010

Tabla 6.6: Cuerpo de la MRCD r''

atributos sean disjuntos. Como se puede apreciar, éste no es el caso para m *y* r'' *ya que ambas comparten el atributo* fecha. *En estos casos, el operador de renombrado, definido en la sección anterior, demuestra su utilidad. Éste nos permitirá renombrar el atributo* fecha *de* m *y* r'' *de tal forma que los nuevos nombres asignados no coincidan en las MRCDs participantes en el producto. De esta forma, las MRCDs participantes en el producto cartesiano serán* $\rho_{(fechaR)\leftarrow(fecha)}(r'')$ *y* $\rho_{(fechaM)\leftarrow(fecha)}(m)$*, ambas MRCDs derivadas respectivamente de* r'' *y* m *sustituyendo al atributo* fecha *por, según el caso,* fechaR *y* fechaM.

Hechas las consideraciones anteriores, podemos proceder al cruce de los datos de r'' *y* m. *Éste proceso resulta en una MRCD, que denominaremos* p *por motivos de brevedad, obtenida por la aplicación de la expresión algebraica indicada en la Ecuación 6.58. En lo que respecta a la cabecera de esta MRCD, su conjunto de atributos es de la forma que se muestra en la Ecuación 6.59 y su función dominio es la que se indica en la Ecuación 6.60. El cuerpo de* p *es la forma que se muestra en la Tabla 6.7.*

$$p = \rho_{(fechaR)\leftarrow(fecha)}(r'') \times \rho_{(fechaM)\leftarrow(fecha)}(m) \tag{6.58}$$

$$A_p = \{fechaR, fechaM, material\} \tag{6.59}$$

$$dom_p(a) = \begin{cases} D_{fecha} & si\ a = fechaR \\ D_{fecha} & si\ a = fechaM \\ D_{material} & si\ a = material \end{cases} \qquad \forall a \in A_p \tag{6.60}$$

La información contenida en p *se corresponde con un cruce indiscriminado de las tuplas en* $\rho_{(fechaR)\leftarrow(fecha)}(r'')$ *y* $\rho_{(fechaM)\leftarrow(fecha)}(m)$. *Esta información será útil siempre que se haga un filtrado posterior, conservando sólo a aquellas tuplas que provienen de la concatenación de dos tuplas originales que se refieren al mismo evento. Para ello, se puede emplear la fecha expresada en cada una de ellas ya que, como se comentó anteriormente, éste dato identifica unívocamente el evento al que se refiere la información contenida en éstas. El emparejamiento descrito anteriormente se puede conseguir mediante la selección de aquellas tuplas de* p *cuyo valor para los atributos* fechaR *y* fechaM *coincide. Este proceso se realizará mediante la aplicación de la expresión algebraica* $\sigma_{fechaR=fechaM}(p)$.

Denominaremos, por motivos de brevedad, a la MRCD resultante de la anterior expresión algebraica como p'. *En lo que respecta a la definición de* p', *su*

U_p	T_p	t_p		
		fechaR	fechaM	material
1.0	$\{1.0/2, 0.6/5, 0.3/6\}_c$	4-6-2010	4-6-2010	videoproyector
1.0	$\{1.0/40, 0.6/125, 0.3/180\}_c$	4-6-2010	4-6-2010	cuaderno
0.5	$\{0.3/6\}_c$	4-6-2010	14-9-2010	videoproyector
0.5	$\{0.3/30\}_c$	4-6-2010	9-12-2010	cuaderno
0.5	$\{0.6/3\}_c$	14-9-2010	4-6-2010	videoproyector
0.5	$\{0.6/75, 0.3/90\}_c$	14-9-2010	4-6-2010	cuaderno
0.5	$\{0.3/3\}_c$	14-9-2010	14-9-2010	videoproyector
0.5	$\{0.3/15\}_c$	14-9-2010	9-12-2010	cuaderno
1.0	$\{1.0/2\}_c$	21-11-2010	4-6-2010	videoproyector
1.0	$\{1.0/40, 0.6/50, 0.3/60\}_c$	21-11-2010	4-6-2010	cuaderno
0.5	$\{0.3/2\}_c$	21-11-2010	14-9-2010	videoproyector
0.5	$\{0.3/10\}_c$	21-11-2010	9-12-2010	cuaderno
0.5	$\{0.3/4\}_c$	9-12-2010	4-6-2010	videoproyector
0.5	$\{0.3/120\}_c$	9-12-2010	4-6-2010	cuaderno
0.5	$\{0.3/4\}_c$	9-12-2010	14-9-2010	videoproyector
0.5	$\{0.3/20\}_c$	9-12-2010	9-12-2010	cuaderno

Tabla 6.7: Cuerpo de la MRCD p

$U_{p'}$	$T_{p'}$	$t_{p'}$		
		fechaR	fechaM	material
1.0	$\{1.0/2, 0.6/5, 0.3/6\}_c$	4-6-2010	4-6-2010	videoproyector
1.0	$\{1.0/40, 0.6/125, 0.3/180\}_c$	4-6-2010	4-6-2010	cuaderno
0.5	$\{0.3/3\}_c$	14-9-2010	14-9-2010	videoproyector
0.5	$\{0.3/20\}_c$	9-12-2010	9-12-2010	cuaderno

Tabla 6.8: Cuerpo de la MRCD p'

cabecera será similar a la de p y su cuerpo está definido como se indica en la Tabla 6.8.

La información contenida en p' es precisamente la que buscábamos. En ella se muestra el material reservado para cada uno de los eventos. La cardinalidad de cada tupla muestra el número (en este caso difuso) de unidades totales que serán necesarias para cada evento. El grado de dicha cardinalidad difusa muestra la combinación de los diferentes grados de confirmación del número de reuniones paralelas a realizar y el número de unidades requeridas por reunión. De igual manera, el grado de certidumbre asociado a la tuplas es una combinación de los grados de certidumbre asociados a los datos relativos a los eventos y las reservas de material.

Adicionalmente al proceso de filtrado descrito anteriormente, por motivos estéticos podremos prescindir en p' uno de los atributos del par fechaR *y* fechaM*, ya que sus valores son redundantes, y restaurar el nombre del que permanezca*

al original fecha. *Esta trasformación se consigue aplicando el operador de proyección y posteriormente el de renombrado de la forma que se muestra en la expresión algebraica* $\rho_{(fecha)\leftarrow(fechaR)}(\pi_{\{fechaR,material\}}(p'))$. *El resultado de dicha expresión es una MRCD equivalente a la obtenida en el posterior Ejemplo 6.6, en el que (como se verá en su momento) se cruza la información de las MRCDs* r'' *y* m *de forma directa, sin necesidad de estas transformaciones.*

6.3.5. Unión

El operador de unión de relaciones es un operador binario que se aplica sobre dos relaciones *compatibles*. Diremos que dos relaciones son compatibles si sus tuplas son de la misma forma. Definiremos formalmente este concepto para el caso de las MRCDs (siendo esta definición igualmente aplicable para el caso de las relaciones clásicas) de la forma que se indica en la siguiente definición.

Definición 6.23 (MRCDs compatibles)**.** *Sean* m_1 *y* m_2 *dos MRCDs. Se dice que* m_1 *y* m_2 *poseen cabeceras compatibles, o simplemente que* m_1 *y* m_2 *son compatibles, si* $A_{m_1} = A_{m_2}$ *y* $dom_{m_1} = dom_{m_2}$.

En el modelo clásico, la aplicación de este operador resulta en una nueva relación cuyo conjunto de tuplas se corresponde con la unión conjuntiva de los conjuntos de tuplas de la relaciones originales. A pesar del sencillo funcionamiento de este operador, la transposición del mismo a las MRCDs no es directa y se han de considerar varios aspectos. En primer lugar se ha de considerar que una unión de dos MRCDs, empleando el operador de unión aditiva de multiconjuntos difusos en sustitución del operador de unión aplicado a las relaciones clásicas, puede dar lugar a una MRCD que no se encuentre en FNPE. Aun cuando las MRCDs participantes estén en la citada forma normal, pueden existir tuplas pertenecientes a una y otra MRCD que se consideren redundantes. Al unir los multiconjuntos de tuplas de ambas MRCDs, el multiconjunto de tuplas de la MRCD resultante contendrá tuplas redundantes. El segundo aspecto a considerar será la forma adecuada en que se agregarán las cardinalidades y grados de certidumbre de las tuplas procedentes de las MRCDs originales para formar la MRCD resultante del proceso de unión.

Para ello, en primer lugar, se abordará en el siguiente apartado el tratamiento de tuplas redundantes en la MRCD resultante de la aplicación del operador de unión. Posteriormente, se dedicará un apartado en el que se tratará la agregación de cardinalidades y grados certidumbre en el proceso de unión.

6.3.5.1. Tratamiento de tuplas redundantes

La unión de dos MRCDs en una sola pueden conllevar la aparición de tuplas redundantes en la MRCD resultante. Esta situación es fácilmente resoluble en el modelo clásico. En cambio, en el marco del álgebra de MRCDs, esta situación requiere, como se indicó en la subsección dedicada al operador de proyección (Subsección 6.3.1), de una solución algo más compleja. Las tuplas en una MRCD están identificadas por su signatura, y al unir dos MRCDs se puede dar el caso de que el resultado contenga tuplas con una misma signatura pero no un mismo valor para todos los atributos.

La solución a esta situación será similar a la que se propuso para el caso de las proyección, siendo estas tuplas redundantes agregadas en una tupla repre-

sentante. En dicho caso, ha sido de interés el establecimiento de los conceptos de equivalencia, clases de equivalencia, tuplas representantes y conjunto de representantes según la signatura restringida de las tuplas debido a que el operador de proyección implica la eliminación de ciertos atributos de la MRCD original. En el caso del operador de unión no existe tal eliminación y por tanto los citados conceptos han de ser definidos en función de la signatura (no restringida) de las tuplas. Estos conceptos, adaptados al nuevo marco, se definen de manera formal como se indica a continuación.

Definición 6.24 (Equivalencia entre tuplas con respecto a sus signaturas)**.** *Sean t_1 y t_2 dos tuplas de $\mathbb{T}_{(A,dom)}$, donde A es un conjunto de atributos y dom una función de dominio de la forma $dom : A \mapsto \mathbb{D}$.*

Definiremos la relación de equivalencia entre dos tuplas t_1 y t_2 con respecto a sus signaturas*, notada como $t_1 \sim t_2$, como se indica en la Ecuación 6.61.*

$$t_1 \sim t_2 \iff sig(t_1) = sig(t_2)$$
$$\forall t_1, t_2 \in \mathbb{T}_{(A,dom)} | (A \subset \mathbb{A} \wedge dom : A \mapsto \mathbb{D}) \quad (6.61)$$

Definición 6.25 (Clase de equivalencia de una tupla con respecto a su signatura)**.** *Sea t una tupla de $\mathbb{T}_{(A,dom)}$, donde A es un conjunto de atributos y dom una función de dominio de la forma $dom : A \mapsto \mathbb{D}$. Sea T un conjunto de tuplas de $\mathbb{T}_{(A,dom)}$.*

Definiremos la clase de equivalencia de t dentro de T según su signatura*, notada como $[t]_T$, como se indica en la Ecuación 6.62*

$$[t]_T = \{t' \in T | t' \sim t\}$$
$$\forall t \in \mathbb{T}_{(A,dom)} | (A \subset \mathbb{A} \wedge dom : A \mapsto \mathbb{D}), \forall T \subseteq \mathbb{T}_{(A,dom)} \quad (6.62)$$

Definición 6.26 (Tupla representante de una clase de equivalencia con respecto a la signatura de una tupla dentro de un conjunto de tuplas)**.** *Sea t una tupla de $\mathbb{T}_{(A,dom)}$, donde A es un conjunto de atributos y dom una función de dominio de la forma $dom : A \mapsto \mathbb{D}$. Sea T un conjunto de tuplas de $\mathbb{T}_{(A,dom)}$. Sea $[t]_T$ la clase de equivalencia de la tupla t dentro de T según su signatura.*

La tupla representante de la clase de equivalencia $[t]_T$ es una tupla $r_{[t]_T} \in \mathbb{T}_{(A,dom)}$ definida como indica la Ecuación 6.63.

$$r_{[t]_T}(a) = aggr(dom(a), \{t'(a) | t' \in [t]_T\})$$
$$\forall t \in \mathbb{T}_{(A,dom)} | (A \subset \mathbb{A} \wedge dom : A \mapsto \mathbb{D}), \forall T \subseteq \mathbb{T}_{(A,dom)}, \forall a \in A \quad (6.63)$$

Definición 6.27 (Conjunto de tuplas representantes con respecto a la signatura de un conjunto de tuplas)**.** *Sea $T \subseteq \mathbb{T}_{(A,dom)}$ un conjunto de tuplas donde $A \subset \mathbb{A}$ es un conjunto de atributos y dom una función dominio de la forma $dom : A \mapsto \mathbb{D}$.*

El conjunto de tuplas representantes con respecto a la signatura de las tuplas en T *se define como indica la Ecuación 6.64.*

$$R(T) = \{r_{[t]_T} | t \in T\} \quad (6.64)$$

Al igual que en el caso de las definiciones basadas en signaturas restringidas, y con la intención de no introducir complicaciones innecesarias en la notación, se emplearán indistintamente conjuntos clásicos o multiconjuntos difusos de tuplas para el caso de T.

6.3.5.2. Agregación de cardinalidades y certidumbres

Una vez detectadas las tuplas redundantes y agregado el valor de sus atributos en una única tupla representante, se ha de determinar el valor de la cardinalidad y certidumbre de ésta.

En lo que respecta a la cardinalidad, se debe considerar que el operador unión del álgebra relacional clásica se corresponde semánticamente con el operador de unión de conjuntos. En el caso de los multiconjuntos difusos, como se vio en los capítulos introductorios, existen dos operadores de unión y, por tanto, esta correspondencia no es directa. La diferencia entre estos dos operadores radica precisamente en la forma en que se agregan las cardinalidades que poseen los elementos en los multiconjuntos participantes en la unión. En la presente propuesta hemos optado por la aplicación de la semántica aditiva. Este semántica es coherente con el soporte de los multiconjuntos a la pertenencia múltiple de elementos o, en otras palabras, la existencia de duplicados. Parece lógico pensar que si existen n duplicados de una tupla en una de las MRCDs a unir y m duplicados en la otra MRCD participante en la unión, el número de duplicados en la MRCD resultante del proceso sea la suma de éstos. Por tanto, la agregación de las cardinalidades de tuplas redundantes se realizará de forma aditiva. Se ha de remarcar que esta propuesta es coherente con el proceso de agregación de cardinalidades de tuplas redundantes propuesto para el operador de proyección.

Adicionalmente, se ha de determinar cómo se agregarán los grados de certidumbre asociados a las tuplas de signatura similar de ambas MRCDs participantes. Para ello, tendremos en cuenta que la agregación de tuplas redundantes tiene asociada una semántica conjuntiva en lo que se refiere a los valores asociados a los atributos. Por tanto, parece lógico que la agregación de los grados de certidumbre relativos al valor de dichos atributos siga esa misma semántica y se emplee una t-norma para tal propósito.

6.3.5.3. Definición del operador de unión

Según lo anterior, definiremos formalmente el operador de unión de MRCDs de la forma que se indica a continuación.

Definición 6.28 (Unión)**.** *Sean m_1 y m_2 dos MRCDs compatibles. La unión de m_1 y m_2, notada como $(m_1 \cup m_2)$, es una MRCD de la forma que se indica en la Ecuación 6.65.*

$$(m_1 \cup m_2) = (\mathcal{H}_{m_1}, \mathcal{B}_{(m_1 \cup m_2)}) \tag{6.65}$$

En la anterior ecuación, el cuerpo de $(m_1 \cup m_2)$ está definido según se indica en la Ecuación 6.66.

$$\mathcal{B}_{(m_1 \cup m_2)} = (T_{(m_1 \cup m_2)}, U_{(m_1 \cup m_2)}) \tag{6.66}$$

Los componentes del cuerpo de la MRCD $(m_1 \cup m_2)$ están definidos de la siguiente forma:

- *El conjunto de tuplas de* $(m_1 \cup m_2)$, $T_{(m_1 \cup m_2)}$, *está definido según la función característica que se indica en la Ecuación 6.67.*

$$\omega_{T_{(m_1 \cup m_2)}}(t) = \\ = \begin{cases} \omega_{T_{m_1}}(t_1) \boxplus \omega_{T_{m_2}}(t_2) & si\ \exists! t_1 \in T_{m_1} | r_{[t_1]_{(T_{m_1} \cup T_{m_2})}} = t \wedge \\ & \exists! t_2 \in T_{m_2} | r_{[t_2]_{(T_{m_1} \cup T_{m_2})}} = t \\ \{1.0/0\}_2 & en\ otro\ caso \end{cases}$$

$$\forall t \in \mathbb{T}_{(m_1 \cup m_2)} \quad (6.67)$$

- *La función de certidumbre de la MRCD resultante de la unión,* $U_{(m_1 \cup m_2)}$, *es de la forma que se indica en la Ecuación 6.68.*

$$U_{(m_1 \cup m_2)}(t) = \\ = \begin{cases} U_{m_1}(t_1) \otimes U_{m_2}(t_2) & si\ \exists! t_1 \in T_{m_1} | (r_{[t_1]_{(T_{m_1} \uplus T_{m_2})}} = t) \wedge \\ & \exists! t_2 \in T_{m_2} | (r_{[t_2]_{(T_{m_1} \uplus T_{m_2})}} = t) \\ U_{m_1}(t) & si\ t \in T_{m_1}\ \wedge\ \nexists t_2 \in T_{m_2} | (r_{[t_2]_{(T_{m_1} \uplus T_{m_2})}} = t) \\ U_{m_2}(t) & si\ t \in T_{m_2}\ \wedge\ \nexists t_1 \in T_{m_1} | (r_{[t_1]_{(T_{m_1} \uplus T_{m_2})}} = t) \end{cases}$$

$$\forall t \in T_{(m_1 \cup m_2)} \quad (6.68)$$

En las ecuaciones 6.67 y 6.68 se indica, en la condición que determina la aplicabilidad de sus primeros casos, que las tuplas t_1 y t_2 sean las únicas tuplas de m_1 y m_2, respectivamente, cuya tupla representante sea la tupla t. Con respecto a lo anterior, se ha de remarcar que, en el caso de que las tuplas t_1 y t_2 existan, éstas son únicas. Lo anterior se puede garantizar ya que, al estar m_1 y m_2 en FNPE, no pueden existir en dichas MRCDs dos tuplas con una misma signatura y, por tanto, con una misma tupla representante.

El siguiente ejemplo ilustra el funcionamiento del operador de unión de MRCDs según la definición anterior.

Ejemplo 6.4. *Supongamos que la dirección del centro de negocios desea obtener un informe consolidado por años de las reservas registradas. Para el caso de este ejemplo, supongamos también que el centro de negocios posee varias sucursales y que éstas organizan los datos relativos a sus eventos en MRCDs (o simplemente bases de datos) distintas.*

Sería posible obtener el informe requerido por la dirección, pero de forma separada para cada sucursal, siguiendo el procedimiento descrito en el Ejemplo 6.1. Una vez obtenidas en cada sucursal las MRCDs conteniendo los datos que componen los fragmentos del informe, éstas se podrían combinar en la central

para generar el informe general requerido. Ilustraremos el funcionamiento del operador de unión propuesto anteriormente mediante la unión las MRCDs conteniendo los datos consolidados por sucursal para la creación de una MRCD que contenga los datos generales.

Partiremos de las mismas asunciones hechas en el Ejemplo 6.1 en lo que se refiere a la forma de representación de las reservas de eventos. Por tanto, emplearemos la misma cabecera asociada a la MRCD r' (definida en el citado ejemplo) para la representación de las mismas. Supondremos, por simplicidad, la existencia de sólo dos sucursales que denominaremos A y B.

Supongamos que los datos de reservas consolidados por años de la sucursal A se representan en una MRCD denominada r_A. La cabecera de esta MRCD coincidirá con la de la expresión $\pi_{\{año,requisitos\}}(r')$, estando la MRCD resultante de dicha expresión definida en el Ejemplo 6.1. El cuerpo de r_A (derivado del mostrado en la Tabla 6.2 para la expresión $\pi_{\{año,requisitos\}}(r')$ y adaptado al presente ejemplo) está definido como se indica en la Tabla 6.9.

En lo que respecta a los datos consolidados por años de la sucursal B, éstos se representarán una MRCD denominada r_B. Obviamente, el esquema de r_B será igual al de r_A. El cuerpo de r_B estará definido como se indica en la Tabla 6.10.

Nótese que r_A y r_B son MRCDs compatibles ya que comparten cabeceras similares. Una vez se dispone de los cuerpos de r_A y r_B, obtendremos la MRCD conteniendo la información consolidada por años para la totalidad del centro de negocios aplicando el operador de unión sobre éstas. En lo que respecta a la cabecera de $(r_A \cup r_B)$, ésta es de la misma forma que lo es la cabecera de las MRCDs originales. Con respecto al cuerpo de $(r_A \cup r_B)$, éste está definido según se indica en la Tabla 6.11.

Obsérvese cómo las tuplas con signatura idéntica en r_A y r_B (y por tanto, tuplas redundantes en la unión) quedan agregadas en una sola tupla que las representa en $(r_A \cup r_B)$. En concreto, las tupla primera de r_A (siempre siguiendo el orden en que se muestran en la Tabla 6.9) y primera de r_B (igualmente, siempre siguiendo el orden en que se muestran en la Tabla 6.10) quedan representadas por una única tupla en $(r_A \cup r_B)$, siendo ésta la primera según en el orden en que se muestran las tuplas en la Tabla 6.11. Nótese que dicha tupla representante agrega los valores para el atributo requisitos *de las dos tuplas originales mencionadas anteriormente y por tanto ilustra la agregación de valores de tipo compuesto así como, por extensión, la agregación de valores de tipo impreciso, incierto y multiconjuntivo difuso. En este caso, además, se evidencia el proceso de agregación de los diferentes grados de certidumbre asociados a las tuplas que ésta representa. El proceso de agregación de tuplas queda también ilustrado en el caso de las tuplas tercera de r_A y segunda de r_B. Estas quedan representadas por la tupla tercera de $(r_A \cup r_B)$.*

6.3.6. Diferencia

En el caso clásico, el operador de diferencia crea una nueva relación a partir de dos relaciones compatibles preexistentes en la cual estarán incluidas las tuplas de la primera relación original (o relación minuendo) siempre que no exista una tupla similar en la segunda de ellas (o relación sustraendo).

Al extender este operador al caso de las MRCDs se ha de realizar una serie de consideraciones previas. En primer lugar, se ha de tener en cuenta que las

U_{r_A}	T_{r_A}	t_{r_A}				
		año	requisitos			
			asistentes	catering	intérpretes	
1.0	$\{1.0/4, 0.6/7, 0.3/8\}_c$	2010	$entre(20, 25)$	$\langle 0.5, \text{sí}\rangle$	$\{1.0/4, 0.75/6, 0.5/14\}_c$	Inglés
					$\{1.0/1, 0.75/2, 0.5/4\}_c$	Alemán
					$\{1.0/1, 0.75/5\}_c$	Francés
					$\{0.5/4\}_c$	Italiano
0.5	$\{0.6/3\}_c$	2010	$aprox(40)$	$\langle 0.5, \text{no}\rangle$	$\{0.5/10\}_c$	Inglés
1.0	$\{0.3/4\}_c$	2011	$hasta(5)$	$\langle 1.0, \text{no}\rangle$	$\emptyset_{inter}$	

Tabla 6.9: Cuerpo de r_A

U_{r_B}	T_{r_B}	t_{r_B}				
		año	requisitos			
			asistentes	catering	intérpretes	
0.5	$\{1.0/1, 0.3/2\}_c$	2010	26	$\langle 1.0, \text{sí}\rangle$	$\{1.0/5\}_c$ $\{1.0/2\}_c$	Inglés Alemán
1.0	$\{1.0/1, 0.6/2\}_c$	2011	$hasta(10)$	$\langle 0.5, \text{no}\rangle$	$\{0.75/3\}_c$	Italiano
0.5	$\{0.6/4\}_c$	2011	5	$\langle 0.5, \text{sí}\rangle$	$\emptyset_{inter}$	

Tabla 6.10: Cuerpo de r_B

tuplas pueden pertenecer de manera múltiple a las MRCDs. Esto implicará que la existencia de una misma tupla en la MRCD minuendo y en la MRCD sustraendo no resultará en todos los casos, al contrario de como ocurre en el caso clásico, en la exclusión de ésta en la MRCD resultante. Al realizar la operación de substracción, se deberá tener en cuenta la cardinalidad de la tupla en la MRCD minuendo y en la sustraendo, quedando la cardinalidad de dicha tupla en la MRCD resultante como la resta de ambas. Esto implicará que, sólo en el caso de que la cardinalidad de la tupla en la MRCD sustraendo supere a la que ésta posee en la MRCD minuendo, ésta quede excluida en la MRCD resultante.

En segundo lugar, como en el caso de la unión, se ha de considerar la forma en que las tuplas en las MRCDs participantes en el proceso de diferencia se considerarán similares. En el caso clásico bastará con que los valores de los atributos para éstas sean similares. En el caso de las MRCDs se empleará la signatura de dichas tuplas.

La identificación de tuplas mediante su signatura lleva, una vez más, a considerar el caso en que las tuplas consideradas iguales no posean el mismo valor para todos sus atributos. En el caso de la unión se optó por la inclusión en la MRCD resultante de una tupla representante que agregara a las tuplas redundantes. En el caso de la diferencia no parece lógico seguir dicho procedimiento. Se ha de tener en cuenta que en caso la unión las MRCDs participantes aportan tuplas al cuerpo de la MRCD resultante, por lo que resulta lógico que en el caso de tuplas redundantes sus valores se agreguen en una tupla representante. En el caso de la diferencia de MRCDs, sólo la MRCD substraída aportará tuplas a la MRCD resultante, por lo que parece lógico en este caso que sean las tuplas de ésta, y no unas tuplas procedentes de la agregación de las tuplas presentes en la MRCD minuendo y sustraendo, las que formen parte del cuerpo de la MRCD resultante.

En lo relativo al grado de certidumbre asociado a las tuplas en la MRCD resultante, se ha de tener en cuenta que, según lo anterior, los valores de los atributos de las tuplas en dicha MRCD serán tomados directamente de las tuplas en la MRCD minuendo. Recordemos que el grado de certidumbre de una tupla se refiere al nivel de certidumbre que se posee sobre que el valor de los atributos que ésta presenta. De esta forma, parece lógico que el grado de certidumbre de las tuplas en la MRCD resultante sea el mismo que éstas poseen en la MRCD minuendo, ya que poseen los mismos valores que las tuplas de dicha MRCD. Por otra parte, se ha de considerar que para determinar qué tuplas de la MRCD

$U_{(r_A \cup r_B)}$	$T_{(r_A \cup r_B)}$	$t_{(r_A \cup r_B)}$				
		año	requisitos			
			asistentes	catering	intérpretes	
0.5	$\{1.0/5, 0.6/8, 0.3/10\}_c$	2010	$entre(20, 26)$	$\langle 0.5, \text{sí} \rangle$	$\{1.0/9, 0.75/11, 0.5/19\}_c$	Inglés
					$\{1.0/3, 0.75/4, 0.5/6\}_c$	Alemán
					$\{1.0/1, 0.75/5\}_c$	Francés
					$\{0.5/4\}_c$	Italiano
0.5	$\{0.6/3\}_c$	2010	$aprox(40)$	$\langle 0.5, \text{no} \rangle$	$\{0.5/10\}_c$	Inglés
1.0	$\{1.0/1, 0.6/2, 0.3/6\}_c$	2011	$hasta(10)$	$\langle 0.5, \text{no} \rangle$	$\{0.75/3\}_c$	Italiano
0.5	$\{0.6/4\}_c$	2011	5	$\langle 0.5, \text{sí} \rangle$	$\emptyset_{inter}$	

Tabla 6.11: Cuerpo de $(r_A \cup r_B)$

minuendo formarán parte del cuerpo de la MRCD resultante se ha de realizar el proceso de emparejamiento, descrito en el párrafo anterior, entre éstas y las tuplas de la MRCD sustraendo. Éste proceso se basa en el valor asociado a los atributos signatura de las tuplas de ambas MRCDs participantes, valores que están afectados por cierto grado de incertidumbre. Por tanto, parece también lógico que el grado de certidumbre asociado a las tuplas de la MRCD resultante indique bajo qué condiciones de certeza se realizó tal emparejamiento. Ambas consideraciones pueden ser atendidas si los grados de certidumbre de las tuplas de las MRCDs minuendo y sustraendo emparejadas son agregados, siendo el resultado de la agregación el valor de certidumbre que se asociará a la tupla en que éste emparejamiento derive en la MRCD resultante. Se empleará una t-norma en el citado proceso de agregación de grados de certidumbre debido a la semántica conjuntiva asociada al mismo.

Según lo anterior, el cuerpo de la MRCD resultante del proceso de diferencia estará formado por las tuplas de la MRCD minuendo, siendo la cardinalidad de cada una de éstas la resta entre la que posee en la MRCD minuendo y la cardinalidad de la tupla (si ésta existe) que comparte su misma signatura en la MRCD sustraendo.

Finalmente, en lo que respecta a la cabecera de la MRCD resultante, ésta compartirá, obviamente, la misma que la que comparten las MRCDs participantes.

Según las consideraciones anteriores, definiremos formalmente el operador de diferencia de MRCDs como se muestra a continuación.

Definición 6.29 (Diferencia)**.** *Sean m_1 y m_2 dos MRCDs compatibles. La diferencia de m_1 y m_2, notada como $(m_1 - m_2)$, es una MRCD de la forma que se indica en la Ecuación 6.69.*

$$(m_1 - m_2) = (\mathcal{H}_{m_1}, \mathcal{B}_{(m_1 - m_2)}) \tag{6.69}$$

En lo anterior, el cuerpo de $(m_1 - m_2)$, $\mathcal{B}_{(m_1-m_2)}$, está definido según se indica en la Ecuación 6.70.

$$\mathcal{B}_{(m_1-m_2)} = (T_{(m_1-m_2)}, U_{(m_1-m_2)}) \tag{6.70}$$

Los componentes del cuerpo de $(m_1 - m_2)$ están definidos de la siguiente forma:

- *El multiconjunto de tuplas de la MRCD resultante, está definido según la función característica que se indica en la Ecuación 6.71. En la citada ecuación, el operador $\boxminus$ representa la diferencia optimista de cardinalidades difusas.*

$$\omega_{T_{(m_1-m_2)}}(t) = \begin{cases} \omega_{T_{m_1}}(t) \boxminus \omega_{T_{m_2}}(t_2) & si\ t \in T_{m_1} \wedge \\ & \exists! t_2 \in [t]_{T_{m_2}} | (\omega_{T_{m_2}}(t_2) \leq \omega_{T_{m_1}}(t)) \\ \{1.0/0\}_2 & en\ otro\ caso \end{cases}$$

$$\forall t \in \mathbb{T}_{(m_1-m_2)} \tag{6.71}$$

- *La función de certidumbre de $(m_1 - m_2)$ está definida según se indica en la Ecuación 6.72.*

$$U_{(m_1-m_2)}(t) = U_{m_1}(t) \otimes U_{m_2}(t_2)$$

$$\forall t \in T_{(m_1-m_2)} | (\ \exists! t_2 \in [t]_{T_{m_2}}) \quad (6.72)$$

En relación a la anterior definición, concretamente de la Ecuación 6.71, se han de hacer las siguientes consideraciones:

- En primer lugar, la segunda componente de la condición para su primer caso, concretamente la expresión $\exists! t_2 \in [t]_{T_{m_2}}$, exige la unicidad de la tupla t_2 en $[t]_{T_{m_2}}$. Esta unicidad, en el caso de que $[t]_{T_{m_2}}$ no esté vacío, está garantizada ya que al encontrarse m_2 en FNPE no será posible encontrar dos o más tuplas en la citada MRCD que compartan la signatura de t. Evidentemente, en caso de que esto no fuese así, existirían varias tuplas en m_2 con una misma signatura, lo que las convertiría en redundantes y no permitirían que m_2 cumpliera la FNPE.

- En segundo lugar, en la citada componente se exige que la cardinalidad asociada a t_2 sea menor que la cardinalidad asociada a la tupla t en m_1. Esta condición es necesaria ya que la diferencia optimista de dos cardinalidades difusas no está definida en los casos en que el minuendo es menor que el substraendo, tal y como se indicó en los capítulos introductorios del presente trabajo. Nótese también que la condición $\omega_{T_{m_2}}(t_2) \leq \omega_{T_{m_1}}(t)$ es una condición crisp, estando ésta definida para dos cardinalidades difusas Q y Q' como se indica en la Ecuación 6.73.

$$Q \leq Q' \iff \forall x \in \mathbb{N}, \mu_Q(x) \leq \mu_{Q'}(x) \quad (6.73)$$

Finalmente, en la Ecuación 6.72, obsérvese que la condición que restringe el conjunto de las tuplas para la que está definida la función de certidumbre, se ha introducido una restricción adicional para las tuplas que forman parte del multiconjunto $T_{(m_1-m_2)}$. La citada restricción se satisface para cualquier tupla de dicho multiconjunto, ya que esta condición está incluida en la definición que se hace de $T_{(m_1-m_2)}$ en la Ecuación 6.71. Esta restricción, por tanto, es exclusivamente instrumental y tiene como objetivo definir la tupla t_2, siendo ésta la tupla en la MRCD m_2 que se ha emparejado con la tupla t (procedente inicialmente de la MRCD m_1).

Ilustraremos el funcionamiento del operador de diferencia de MRCDs descrito anteriormente mediante el siguiente ejemplo.

Ejemplo 6.5. *Volvamos al escenario empleado en el Ejemplo 6.4, donde se pretendía obtener un informe general en el que se mostraran las reservas del centro de negocios consolidadas anualmente a partir de los informes parciales de cada una de las sucursales. Supongamos ahora que la dirección del citado centro desea comparar el rendimiento comercial anual de cada una de las sucursales (medido en función de las reservas que poseen en su base de datos) a partir de*

dichos informes parciales. Concretamente la dirección desea obtener un informe que, dadas dos sucursales A y B, muestre en qué aventaja, en lo que se refiere a rendimiento comercial, la sucursal A a la B.

Partiendo de las MRCDs r_A y r_B definidas en el Ejemplo 6.4, será posible dar respuesta al informe requerido. Para ello, crearemos una nueva MRCD que sea la diferencia de las dos anteriores. La MRCD $(r_A - r_B)$ contendrá aquellas reservas que se encuentren en r_A y no en r_B. Al permitir las MRCDs la asociación de información cuantitativa y cualitativa a sus tuplas, $(r_A - r_B)$ contendrá información de en cuantas reservas aventaja la sucursal A a la B y qué grado de confirmación mínimo poseen éstas.

La MRCD $(r_A - r_B)$ se define como una MRCD cuya cabecera coincide con la de sus componentes, r_A y r_B. En lo que respecta al cuerpo de $(r_A - r_B)$, éste se define según se indica en la Tabla 6.12.

En el cuerpo de $(r_A - r_B)$ puede observarse cómo la cardinalidad de la primera tupla se ve reducida con respecto a la que ésta poseía en r_A. Como en anteriores ejemplos, el orden con el que se refiere a las tuplas se corresponde con el orden en que éstas se muestran, en función de la MRCD a la que cada tupla pertenezca, en las tablas 6.9, 6.10 y 6.12. Concretamente, la cardinalidad que la referida tupla presenta en $(r_A - r_B)$ procede de la substracción de la que ésta posee en r_A y la cardinalidad asociada a la primera tupla de r_B. Obsérvese que, a pesar de que dichas tuplas no son idénticas, éstas poseen la misma signatura, lo que semánticamente se interpreta como que éstas representan el mismo tipo *de reserva. En lo que respecta al valor de los campos de la citada tupla, así como al grado de certidumbre asociado a éstos, cada tupla resultante conserva los valores que posee originalmente en r_A y éstos no proceden de agregación alguna con la referida tupla de r_B que ha provocado la reducción de su cardinalidad.*

La tupla resultante en que deriva el anterior par de tuplas también evidencia el mecanismo de agregación de grados de certidumbre de minuendo y sustraendo. Obsérvese que, a pesar de que el valor asociado a los atributos para la primera tupla de $(r_A - r_B)$ se corresponde con los asociados a la primera tupla de r_A, los grados de certidumbre asociados a éstas no son coincidentes. El citado proceso de agregación se hace evidente en este caso debido a que la tupla sustraendo posee un grado de certidumbre menor que el asociado a la tupla minuendo, lo que provoca que la tupla resultante de este emparejamiento vea disminuido su grado de certidumbre con respecto a la tupla original de la que procede. En el presente ejemplo se ha usado la función mínimo como t-norma para la agregación de grados de certidumbre.

La cardinalidad de la segunda tupla de $(r_A - r_B)$ se conserva idéntica a la que ésta posee en r_A ya que dicha tupla no existe en r_B una tupla con su misma signatura.

Finalmente, obsérvese que la tercera tupla de r_A no deriva en una tupla $(r_A - r_B)$ a pesar de que en la MRCD sustraendo la tupla con la que se empareja, la segunda tupla de r_B, posee una cardinalidad menor de forma absoluta. Es precisamente el detalle de que es menor exclusivamente *si se comparan las cardinalidades de forma absoluta lo que impide que dicha tupla esté incluida en $(r_A - r_B)$. Debido a que los grados de posibilidad de la cardinalidad de la tercera tupla de r_A son menores que los asociados a la segunda de r_B, la primera cardinalidad no se puede declarar mayor que la segunda. Esta situación impide que la segunda cardinalidad pueda ser substraída de la primera, y por tanto (en apli-*

$U_{(r_A - r_B)}$	$T_{(r_A - r_B)}$	$t_{(r_A - r_B)}$				
		año	requisitos			
			asistentes	catering	intérpretes	
0.5	$\{1.0/3, 0.6/6\}_c$	2010	$entre(20, 25)$	$\langle 0.5, \text{sí} \rangle$	$\{1.0/4, 0.75/4, 0.5/14\}_c$	Inglés
					$\{1.0/1, 0.75/2, 0.5/4\}_c$	Alemán
					$\{1.0/1, 0.75/5\}_c$	Francés
					$\{0.5/4\}_c$	Italiano
0.5	$\{0.6/3\}_c$	2010	$aprox(40)$	$\langle 0.5, \text{no} \rangle$	$\{0.5/10\}_c$	Inglés

Tabla 6.12: Cuerpo de $(r_A - r_B)$

cación del segundo caso de la Ecuación 6.71) la cardinalidad que la citada tupla posee asociada en la MRCD resultante es nula (lo que implica en la práctica su exclusión).

6.4. Operadores derivados en el álgebra sobre multirrelaciones complejas difusas

Los operadores del álgebra relacional pueden ser categorizados en dos grupos fundamentales, los operadores básicos y los operadores derivados. El primer grupo está compuesto por aquellos operadores que no pueden ser expresados en base a otros operadores del álgebra relacional. El segundo grupo está formado por aquellos operadores que pueden ser expresados en base a operadores básicos. Este último grupo está formado por las diferentes variantes de la de reunión así como por los operadores de intersección y división.

El presente apartado estudia la transposición de los citados operadores derivados al ámbito de las MRCDs, centrándose particularmente en la validez de las expresiones que permiten recrear el comportamiento de éstos en base a la aplicación de operadores básicos del álgebra de MRCDs.

6.4.1. Reunión

En el contexto del álgebra relacional clásica, los operadores de reunión son un tipo de operadores relacionales binarios que producen como resultado una única relación resultante que proviene del emparejamiento de las tuplas de las relaciones que hacen las veces de operandos. Existe una amplia variedad de operadores de reunión cuya diferencia radica precisamente en la forma en que se realiza dicho emparejamiento.

La presente sección analizará la transposición de este tipo de operadores al ámbito de las MRCDs.

6.4.1.1. Reunión natural

Podríamos decir que el operador de reunión natural del álgebra relacional es una derivación restringida del producto cartesiano de relaciones. Recordemos que el producto cartesiano de dos relaciones resulta en una única relación formada por la combinación, en forma de concatenación, de las tuplas de las relaciones participantes.

El operador de reunión natural realiza la misma operación pero, en lugar de hacer una combinación de las tuplas de las relaciones participantes de forma indiscriminada, éste sólo combina aquellas tuplas que poseen el mismo valor para los atributos que poseen en común. Este último requisito pone en evidencia que para la aplicación de éste operador, y al contrario que ocurría en el caso del producto cartesiano, las relaciones participantes deben poseer algunos atributos con nombre y dominio coincidentes. Notaremos la *reunión natural* entre dos relaciones r y r' como $(r \bowtie r')$.

Como se puede deducir de lo anterior, el operador de reunión natural puede derivarse del producto cartesiano (que se encargará de realizar la combinación de las tuplas de ambas relaciones) y del operador de selección (que será el encargado de seleccionar aquellas tuplas procedentes de la combinación que

compartan el mismo valor para los atributos comunes). Además de los anteriores operadores, será necesario hacer uso del operador de renombrado para evitar que las relaciones participantes posean atributos comunes en el momento de la aplicación del producto cartesiano y del operador de proyección para eliminar los atributos redundantes una vez terminada la selección. Concretamente, el operador de reunión natural se expresa en función de los operadores básicos de la forma que se indica a continuación.

Definición 6.30 (Reunión natural en función de operadores básicos del álgebra relacional)**.** *Sean r y r' dos relaciones tales que sus conjuntos de atributos son de la forma $A_r = \{a_1, a_2, \ldots, a_n, c_1, c_2, \ldots, c_m\}$, donde $n \in \mathbb{N}$ y $m \in \mathbb{N}^+$, y $A_{r'} = \{a'_1, a'_2, \ldots, a'_l, c_1, c_2, \ldots, c_m\}$, siendo $l \in \mathbb{N}$.*

Podemos definir la reunión natural *de r y r' en base a los operadores básicos del álgebra relacional según se indica en la Ecuación 6.74.*

$$r \bowtie r' = \pi_{(A_r \cup A_{r'})}(\sigma_{(c_1=c'_1 \wedge c_2=c'_2 \wedge \ldots \wedge c_m=c'_m)}(r \times \rho_{(c'_1,c'_2,\ldots,c'_m)\leftarrow(c_1,c_2,\ldots,c_m)}(r'))) \tag{6.74}$$

En lo que respecta al álgebra de MRCDs, la anterior equivalencia sigue siendo válida. Para la trasposición del operador de reunión natural al contexto de las MRCDs se han de hacer las mismas consideraciones que se hicieron para el operador de producto cartesiano, en particular en lo que se refiere a la agregación de las cardinalidades difusas y grados de certidumbre asociados a las tuplas que se combinan en la MRCD resultante. De hecho, la única diferencia entre la reunión natural y dicho operador es la selección posterior que se hace de aquellas tuplas que proceden de la combinación de tuplas cuyos valores asociados a los atributos que poseen en común son iguales. Esta última es una condición *crisp* cuya aplicación es directa.

Según lo anterior, definiremos el operador de reunión natural de dos MRCDs como se indica a continuación.

Definición 6.31 (Reunión natural)**.** *Sean m_1 y m_2 dos MRCDs tales que un subconjunto de sus atributos son compatibles, esto es, $A_{m_1} \cap A_{m_2} \neq \emptyset$ y $dom_{m_1}^{|(A_{m_1} \cap A_{m_2})} = dom_{m_2}^{|(A_{m_1} \cap A_{m_2})}$.*

La reunión natural *de m_1 y m_2, notada como $(m_1 \bowtie m_2)$, es una MRCD definida como se indica en la Ecuación 6.75.*

$$(m_1 \bowtie m_2) = (\mathcal{H}_{(m_1 \bowtie m_2)}, \mathcal{B}_{(m_1 \bowtie m_2)}) \tag{6.75}$$

En lo anterior, la cabecera de $(m_1 \bowtie m_2)$ está definida como se indica en la Ecuación 6.76 y su cuerpo según se indica en la Ecuación 6.77.

$$\mathcal{H}_{(m_1 \bowtie m_2)} = (A_{(m_1 \bowtie m_2)}, dom_{(m_1 \bowtie m_2)}) \tag{6.76}$$

$$\mathcal{B}_{(m_1 \bowtie m_2)} = (T_{(m_1 \bowtie m_2)}, U_{(m_1 \bowtie m_2)}) \tag{6.77}$$

Los componentes de cabecera y cuerpo están definidos como sigue:

- *El conjunto de atributos de $(m_1 \bowtie m_2)$ de define como se indica en la Ecuación 6.78.*

$$A_{(m_1 \bowtie m_2)} = A_{m_1} \cup A_{m_2} \tag{6.78}$$

- *Su función dominio, $dom_{(m_1 \bowtie m_2)}$, está definida según se indica en la Ecuación 6.79.*

$$dom_{(m_1 \bowtie m_2)}(a) = \begin{cases} dom_{m_1}(a) & si\ a \in A_{m_1} \\ dom_{m_2}(a) & si\ a \in A_{m_2} \end{cases} \qquad \forall a \in A_{(m_1 \bowtie m_2)} \quad (6.79)$$

- *El conjunto de tuplas de la MRCD resultante, $T_{(m_1 \bowtie m_2)}$, se define según la función característica que se muestra en la Ecuación 6.80.*

$$\omega_{T_{(m_1 \bowtie m_2)}}(t) = \\ = \begin{cases} \omega_{T_{m_1}}(t_1) \boxtimes \omega_{T_{m_2}}(t_2) & si\ (\exists t_1 \in T_{m_1} | t_1 = t^{|A_{m_1}}) \wedge \\ & \qquad (\exists t_2 \in T_{m_2} | t_2 = t^{|A_{m_2}}) \\ \{1.0/0\}_2 & en\ caso\ contrario \end{cases}$$

$$\forall t \in \mathbb{T}_{(m_1 \bowtie m_2)} \quad (6.80)$$

- *La función de certidumbre de la MRCD resultante, $U_{(m_1 \bowtie m_2)}$, se define tal y como se indica en la Ecuación 6.81.*

$$U_{(m_1 \bowtie m_2)}(t) \ = \ U_{m_1}(t^{|A_{m_1}}) \otimes U_{m_2}(t^{|A_{m_2}}) \qquad \forall t \ \in \ T_{(m_1 \bowtie m_2)} \quad (6.81)$$

Ilustraremos el funcionamiento del operador definido anteriormente empleando el siguiente ejemplo.

Ejemplo 6.6. *En el presente ejemplo continuaremos con el marco empleado en el anterior Ejemplo 6.3. Recordemos que en dicho ejemplo se buscaba cruzar la información disponible en un par de MRCDs. La primera de ellas, denominada r'' y definida en el ejemplo mencionado anteriormente, representa las reservas de eventos a organizar por el centro de negocios. La segunda MRCD, denominada m y definida en el mismo ejemplo mencionado anteriormente, representa el material requerido por los clientes que habrá de estar disponible en los salones en que se celebren cada una de las sesiones paralelas asociadas a dichos eventos. Ambas MRCDs compartían el atributo* fecha*, el cuál permitía identificar los eventos y relacionar éstos con sus respectivas reservas de material.*

El cruce de la información en ambas MRCDs se realizó en el Ejemplo 6.3 aplicando el producto cartesiano sobre las mismas. Como se indicó en el citado ejemplo, se obtuvo como resultado una MRCD que realizaba un cruce indiscriminado de las tuplas pertenecientes respectivamente a las MRCDs r'' y m, sin tener en cuenta la relación que establecía el atributo fecha *entre ellas. Para refinar dicho resultado se propuso la aplicación del operador de selección sobre dicha MRCD de tal forma que se obtenga una nueva MRCD que contenga sólo aquellas tuplas que se refieren al mismo evento.*

En el presente ejemplo ilustraremos cómo realizar el cruce de información que se pretendía realizar en el Ejemplo 6.3 empleando un único operador sobre

$U_{(r'' \bowtie m)}$	$T_{(r'' \bowtie m)}$	$t_{(r'' \bowtie m)}$	
		fecha	material
1.0	$\{1.0/2, 0.6/5, 0.3/6\}_c$	4-6-2010	videoproyector
1.0	$\{1.0/40, 0.6/125, 0.3/180\}_c$	4-6-2010	cuaderno
0.5	$\{0.3/3\}_c$	14-9-2010	videoproyector
0.5	$\{0.3/20\}_c$	9-12-2010	cuaderno

Tabla 6.13: Cuerpo de la MRCD $(r'' \bowtie m)$

las MRCDs r'' y m. Tal objetivo se consigue mediante la expresión algebraica $(r'' \bowtie m)$. El resultado de dicha expresión es una MRCD cuya cabecera está formada por el conjunto de atributos y función de dominio definidos respectivamente en las ecuaciones 6.82 y 6.83. En lo que se refiere al cuerpo de $(r'' \bowtie m)$, éste está definido como se muestra en la Tabla 6.13.

$$A_{(r'' \bowtie m)} = \{fecha, material\} \tag{6.82}$$

$$dom_{(r'' \bowtie m)}(a) = \begin{cases} D_{fecha} & si\ a = fecha \\ D_{material} & si\ a = material \end{cases} \qquad \forall a \in A_{(r'' \bowtie m)} \tag{6.83}$$

La información contenida en $(r'' \bowtie m)$ se corresponde con el cruce de información de las MRCDs r'' y m empleando la fecha del evento como relación entre reservas de eventos y material asociado al mismo. Al igual que ocurría en el caso de la MRCD p' del Ejemplo 6.3, en la MRCD $(r'' \bowtie m)$ se muestra la información relativa al material reservado para cada uno de los eventos. La cardinalidad de cada tupla de $(r'' \bowtie m)$ muestra el número (en este caso difuso) de unidades totales que serán necesarias para cada evento. El grado de dicha cardinalidad difusa muestra la combinación de los diferentes grados de confirmación del número de reuniones paralelas a realizar y el número de unidades requeridas por reunión. De igual manera, el grado de certidumbre asociado a las tuplas de la MRCD resultante es una combinación de los grados de certidumbre asociados a los datos relativos a los eventos y las reservas de material.

6.4.1.2. Otros tipos de reunión

Además de la reunión natural, la más común, existen en la bibliografía diferentes tipos de operadores de reunión. Estos se diferencian de la anterior en la variación del proceso de emparejamiento. En el presente apartado analizaremos cada uno de ellos.

θ-reunión.

Este operador de reunión sustituye en el proceso de emparejamiento el operador de igualdad, aplicado a los valores de los atributos que comparten las relaciones participantes en la reunión natural, por cualquier otro operador relacional que se denomina genéricamente Θ. Estos operadores pueden ser, por

ejemplo, los operadores $<$, $\leq$, $>$, $\geq$ o incluso $=$ (en cuyo caso el operador se denomina *equireunión* y es equivalente a la reunión natural).

Al ser posible, en el caso de la θ-reunión, que el emparejamiento de tuplas de las relaciones participantes no se haga empleando el operador de igualdad, los valores de los atributos empleados para tal emparejamiento pueden no ser iguales. Por tanto, no se podrá considerar ningún atributo participante en el proceso de emparejamiento redundante y se requerirá que todos los atributos de las relaciones participantes estén presentes en la relación resultante. Debido a esto, y al contrario que el operador de reunión natural, este operador requiere que las relaciones participantes no tengan atributos en común. Obsérvese que, debido a este requisito, la aplicación de este operador requiere la indicación explícita de las parejas de atributos de las relaciones participantes que serán comparadas durante el proceso de emparejamiento empleando el operador relacional θ correspondiente.

La θ-reunión de dos relaciones r y r' se nota como $(r \underset{l\theta l'}{\bowtie} r')$, donde l y l' son dos listas de la misma longitud que contienen, respectivamente, a un subconjunto de los atributos de r y r'. Estas listas serán de la forma $l = (a_1, a_2, \ldots, a_n)$ y $l' = (a'_1, a'_2, \ldots, a'_n)$, donde n es la longitud de ambas listas y los a_i y a'_i representan, respectivamente, a atributos de r y r'. Dicha notación indica que cada atributo a_i de la primera lista será comparado, empleando el operador relacional θ, con el atributo a'_i de la segunda lista.

La θ-reunión se define en función de operadores relaciones básicos como se muestra a continuación.

Definición 6.32 (θ-reunión en función de operadores básicos del álgebra relacional)**.** *Sean r y r' dos relaciones tales que no comparten ningún atributo, $A_r \cap A_{r'} = \emptyset$. Sea θ un operador relacional. Sean l y l' dos listas de la forma $l = (a_1, a_2, \ldots, a_n)$ y $l' = (a'_1, a'_2, \ldots, a'_n)$, siendo n la longitud de ambas listas y cada a_i y a'_i, respectivamente, un atributo de r y r'.*

Podemos definir la θ-reunión *de r y r' según las listas de atributos l y l' en base a los operadores básicos del álgebra relacional según se indica en la Ecuación 6.84.*

$$r \underset{l\theta l'}{\bowtie} r' = \sigma_{(a_1 \theta a'_1 \wedge a_2 \theta a'_2 \wedge \ldots \wedge a_n \theta a'_n)}(r \times r') \tag{6.84}$$

Semireunión.

Este tipo de operador de reunión tiene un funcionamiento similar al operador de reunión natural exceptuando la forma en que se determinan el conjunto de atributos de la relación resultante de su aplicación. En el caso de la reunión natural, la relación resultante posee un conjunto de atributos proveniente de la unión de los conjuntos de atributos de las relaciones participantes en la misma. En cambio, en el caso de la semireunión, el conjunto de atributos de la relación resultante estará compuesto exclusivamente por aquellos pertenecientes al conjunto de atributos de la relación que se emplea como primer operando.

La semireunión de dos relaciones r y r' se nota como $(r \ltimes r')$ y se define en función de operadores relaciones básicos como se muestra a continuación.

Definición 6.33 (Semireunión en función de operadores básicos del álgebra relacional)**.** *Sean r y r' dos relaciones tales que sus conjuntos de atributos son*

de la forma $A_r = \{a_1, a_2, \ldots, a_n, c_1, c_2, \ldots, c_m\}$, *donde* $n \in \mathbb{N}$ *y* $m \in \mathbb{N}^+$, *y* $A_{r'} = \{a'_1, a'_2, \ldots, a'_l, c_1, c_2, \ldots, c_m\}$, *siendo* $l \in \mathbb{N}$.

Podemos definir la semireunión *de* r *y* r' *en base a los operadores básicos del álgebra relacional según se indica en la Ecuación 6.85.*

$$r \ltimes r' = \pi_{A_r}(\sigma_{(c_1=c'_1 \wedge c_2=c'_2 \wedge \ldots \wedge c_m=c'_m)}(r \times \rho_{(c'_1,c'_2,\ldots,c'_m)\leftarrow(c_1,c_2,\ldots,c_m)}(r'))) \quad (6.85)$$

Antireunión.

El funcionamiento del operador de antireunión es inverso al del operador de reunión natural. El operador de reunión natural selecciona exclusivamente aquellas tuplas de la relación que se emplea como primer operando que poseen una correspondencia con alguna tupla de la relación que se emplea como segundo operando (o viceversa, ya que el procedimiento es equivalente). En el caso de la antireunión, sólo aquellas tuplas del primer operando que no poseen correspondencia con ninguna tupla del segundo operador son seleccionadas. Obviamente, el resultado de este operador no es una relación que contenga la unión de las tuplas emparejadas como en el caso de la reunión natural. Evidentemente, en este caso las tuplas de la relación resultante son de la misma forma que las del primer operando.

La antireunión de dos relaciones r y r' se nota como $(r \triangleright r')$ y se define en función de operadores relaciones básicos como se muestra a continuación.

Definición 6.34 (Antireunión en función de operadores básicos del álgebra relacional)**.** *Sean* r *y* r' *dos relaciones tales que sus conjuntos de atributos son de la forma* $A_r = \{a_1, a_2, \ldots, a_n, c_1, c_2, \ldots, c_m\}$, *donde* $n \in \mathbb{N}$ *y* $m \in \mathbb{N}^+$, *y* $A_{r'} = \{a'_1, a'_2, \ldots, a'_l, c_1, c_2, \ldots, c_m\}$, *siendo* $l \in \mathbb{N}$.

Podemos definir la antireunión *de* r *y* r' *en base a los operadores básicos del álgebra relacional según se indica en la Ecuación 6.86.*

$$\begin{aligned} r \triangleright r' &= r - (r \ltimes r') = \\ &= r - \left(\pi_{A_r}(\sigma_{(c_1=c'_1 \wedge c_2=c'_2 \wedge \ldots \wedge c_m=c'_m)}(r \times \rho_{(c'_1,c'_2,\ldots,c'_m)\leftarrow(c_1,c_2,\ldots,c_m)}(r')))\right) \end{aligned} \quad (6.86)$$

Reunión natural externa.

La reunión natural externa posee un comportamiento similar al asociado a la reunión natural en lo que se refiere al emparejamiento y unión de las tuplas de las relaciones que se emplean como operandos. No obstante, y a diferencia de ésta, la relación resultante de la aplicación del operador de reunión natural externa contiene, además de los pares de tuplas de las relaciones participantes que han podido ser emparejadas, a aquellas tuplas de las relaciones participantes que no obtuvieron emparejamiento.

La inclusión en la relación resultante de las tuplas no emparejadas requiere la asignación de un valor para los atributos no existentes en la relación de la que provienen. Recordemos que el conjunto de atributos que posee la relación resultante de la aplicación del operador de reunión natural sobre un par de relaciones se corresponde con la unión del conjunto de atributos de éstas. En el caso de una tupla de la relación resultante que representa un par de tuplas

emparejadas, el valor de sus atributos proviene del valor de éstos para una u otra tupla del par, en función de la relación de la que proviene el atributo en cuestión. Para los atributos comunes a las dos relaciones participantes en la reunión, el origen de su valor será indiferente ya que las tuplas emparejadas poseerán el mismo valor asociado para estos atributos. En el caso de una tupla no emparejada, el valor de los atributos de la tupla que representa a ésta en la relación resultante provendrán exclusivamente de la misma. De esta forma, los valores para los atributos existentes en la relación de la que ésta proviene serán los mismos que los asociados a la tupla original de la que procede. En cambio, para el caso de los atributos provenientes de la relación en la que la tupla no encontró emparejamiento, éstos se asociarán a valores ausentes en su dominio correspondiente.

La reunión natural externa de dos relaciones r y r' se nota de la forma que se muestra en la expresión $(r \Rightarrow\!\bowtie\!\Leftarrow r')$ y se define en función de operadores relaciones básicos como se muestra a continuación.

Definición 6.35 (Reunión natural externa en función de operadores básicos del álgebra relacional)**.** *Sean r y r' dos relaciones tales que sus conjuntos de atributos son de la forma $A_r = \{a_1, a_2, \ldots, a_n, c_1, c_2, \ldots, c_m\}$, donde $n \in \mathbb{N}$ y $m \in \mathbb{N}^+$, y $A_{r'} = \{a'_1, a'_2, \ldots, a'_l, c_1, c_2, \ldots, c_m\}$, siendo $l \in \mathbb{N}$. Sean $\omega_{(r,r')}$ y $\omega_{(r',r)}$ dos relaciones cuyo conjunto de atributos es de la forma $A_{\omega_{(r,r')}} = A_{r'} - A_r$ y $A_{\omega_{(r',r)}} = A_r - A_{r'}$. Dichas relaciones contendrán una única tupla, siendo ésta de la forma $(\omega_{dom(a'_1)}, \omega_{dom(a'_2)}, \ldots, \omega_{dom(a'_l)})$ para el caso de $\omega_{(r,r')}$ y para $\omega_{(r',r)}$ de la forma $(\omega_{dom(a_1)}, \omega_{dom(a_2)}, \ldots, \omega_{dom(a_n)})$. En dichas tuplas, $\omega_{dom(a)}$ representa un valor ausente dentro del dominio asignado al atributo a.*

Podemos definir la reunión natural externa *de r y r' en base a los operadores básicos del álgebra relacional según se indica en la Ecuación 6.87.*

$$
\begin{aligned}
r \Rightarrow\!\bowtie\!\Leftarrow r' &= (r \bowtie r') \cup \left((r \triangleright r') \times \omega_{(r,r')}\right) \cup \left((r' \triangleright r) \times \omega_{(r',r)}\right) = \\
&= (r \bowtie r') \cup \left((r - (r \ltimes r')) \times \omega_{(r,r')}\right) \cup \left((r' - (r' \ltimes r)) \times \omega_{(r',r)}\right) = \\
&= \left(\pi_{(A_r \cup A_{r'})} \left(\sigma_{(c_1 = c'_1 \wedge c_2 = c'_2 \wedge \ldots \wedge c_m = c'_m)} \left(r \times \rho_{(c'_1, c'_2, \ldots, c'_m) \leftarrow (c_1, c_2, \ldots, c_m)}(r')\right)\right)\right) \cup \\
&\quad \Big(\left(r - \left(\pi_{A_r} \left(\sigma_{(c_1 = c'_1 \wedge c_2 = c'_2 \wedge \ldots \wedge c_m = c'_m)} \left(r \times \rho_{(c'_1, c'_2, \ldots, c'_m) \leftarrow (c_1, c_2, \ldots, c_m)}(r')\right)\right)\right)\right) \\
&\qquad \times \omega_{(r,r')} \Big) \cup \\
&\quad \Big(\left(r' - \left(\pi_{A_{r'}} \left(\sigma_{(c_1 = c'_1 \wedge c_2 = c'_2 \wedge \ldots \wedge c_m = c'_m)} \left(r' \times \rho_{(c'_1, c'_2, \ldots, c'_m) \leftarrow (c_1, c_2, \ldots, c_m)}(r)\right)\right)\right)\right) \\
&\qquad \times \omega_{(r',r)} \Big)
\end{aligned}
\tag{6.87}
$$

De la reunión natural externa se derivan dos variaciones llamadas *reunión natural externa izquierda* y *reunión natural externa derecha*. Éstas se diferencian en que la relación resultante que produce su aplicación incluye, además de las tuplas de ambos operandos emparejadas, sólo a las tuplas no emparejadas de uno de los operandos, el izquierdo o el derecho, según sea el caso. La reunión natural externa izquierda de dos relaciones r y r' se nota como $(r \Rightarrow\!\bowtie r')$ y la reunión natural externa derecha de ésta de la forma $(r \bowtie\!\Leftarrow r')$. Estos operadores están definidos en base a los operadores básicos del álgebra según se indica a continuación.

Definición 6.36 (Reunión natural externa izquierda en función de operadores básicos del álgebra relacional)**.** *Sean r y r' dos relaciones tales que sus conjuntos de atributos son de la forma $A_r = \{a_1, a_2, \ldots, a_n, c_1, c_2, \ldots, c_m\}$, donde $n \in \mathbb{N}$ y $m \in \mathbb{N}^+$, y $A_{r'} = \{a'_1, a'_2, \ldots, a'_l, c_1, c_2, \ldots, c_m\}$, siendo $l \in \mathbb{N}$. Sea $\omega_{(r,r')}$ una relación cuyo conjunto de atributos es de la forma $A_{\omega_{(r,r')}} = A_{r'} - A_r$. Dicha relación contendrán una única tupla, siendo ésta de la forma $(\omega_{dom(a'_1)}, \omega_{dom(a'_2)}, \ldots, \omega_{dom(a'_l)})$. En dicha tupla, $\omega_{dom(a)}$ representa un valor ausente dentro del dominio asignado al atributo a.*

Podemos definir la reunión natural externa izquierda *de r y r' en base a los operadores básicos del álgebra relacional según se indica en la Ecuación 6.88.*

$$
\begin{aligned}
r \Rightarrow\!\bowtie r' &= (r \bowtie r') \cup \left((r \triangleright r') \times \omega_{(r,r')}\right) = \\
&= (r \bowtie r') \cup \left((r - (r \ltimes r')) \times \omega_{(r,r')}\right) = \\
&= \left(\pi_{(A_r \cup A_{r'})} \left(\sigma_{(c_1 = c'_1 \wedge c_2 = c'_2 \wedge \ldots \wedge c_m = c'_m)} \left(r \times \rho_{(c'_1, c'_2, \ldots, c'_m) \leftarrow (c_1, c_2, \ldots, c_m)}(r')\right)\right)\right) \cup \\
&\quad \Big(\left(r - \left(\pi_{A_r} \left(\sigma_{(c_1 = c'_1 \wedge c_2 = c'_2 \wedge \ldots \wedge c_m = c'_m)} \left(r \times \rho_{(c'_1, c'_2, \ldots, c'_m) \leftarrow (c_1, c_2, \ldots, c_m)}(r')\right)\right)\right)\right) \\
&\qquad \times \omega_{(r,r')} \Big)
\end{aligned}
\tag{6.88}
$$

Definición 6.37 (Reunión natural externa derecha en función de operadores básicos del álgebra relacional)**.** *Sean r y r' dos relaciones tales que sus conjuntos de atributos son de la forma $A_r = \{a_1, a_2, \ldots, a_n, c_1, c_2, \ldots, c_m\}$, donde $n \in \mathbb{N}$ y $m \in \mathbb{N}^+$, y $A_{r'} = \{a'_1, a'_2, \ldots, a'_l, c_1, c_2, \ldots, c_m\}$, siendo $l \in \mathbb{N}$. Sea $\omega_{(r',r)}$ una relación cuyo conjunto de atributos es de la forma $A_{\omega_{(r',r)}} = A_r - A_{r'}$. Dicha relación contendrán una única tupla, siendo ésta de la forma $(\omega_{dom(a_1)}, \omega_{dom(a_2)}, \ldots, \omega_{dom(a_l)})$. En dicha tupla, $\omega_{dom(a)}$ representa un valor ausente dentro del dominio asignado al atributo a.*

Podemos definir la reunión natural externa derecha *de r y r' en base a los operadores básicos del álgebra relacional según se indica en la Ecuación 6.89.*

$$
\begin{aligned}
r \bowtie\!\Leftarrow r' &= (r \bowtie r') \cup \left((r' \triangleright r) \times \omega_{(r',r)}\right) = \\
&= (r \bowtie r') \cup \left((r' - (r' \ltimes r)) \times \omega_{(r',r)}\right) = \\
&= \left(\pi_{(A_r \cup A_{r'})} \left(\sigma_{(c_1 = c'_1 \wedge c_2 = c'_2 \wedge \ldots \wedge c_m = c'_m)} \left(r \times \rho_{(c'_1, c'_2, \ldots, c'_m) \leftarrow (c_1, c_2, \ldots, c_m)}(r')\right)\right)\right) \cup \\
&\quad \Big(\left(r' - \left(\pi_{A_{r'}} \left(\sigma_{(c_1 = c'_1 \wedge c_2 = c'_2 \wedge \ldots \wedge c_m = c'_m)} \left(r' \times \rho_{(c'_1, c'_2, \ldots, c'_m) \leftarrow (c_1, c_2, \ldots, c_m)}(r)\right)\right)\right)\right) \\
&\qquad \times \omega_{(r',r)} \Big)
\end{aligned}
\tag{6.89}
$$

Adaptación de las variantes del operador de reunión al álgebra de multirrelaciones complejas difusas.

Como se ha expuesto anteriormente, las diferentes variaciones del operador de reunión pueden ser expresadas en función de operadores básicos del álgebra relacional. Como en el caso del operador de reunión natural, estos operadores pueden ser adaptados al contexto del álgebra de MRCDs aplicando las mismas

expresiones algebraicas, siempre y cuando se sustituyan los operadores básicos del álgebra relacional clásica por los correspondientes operadores básicos del álgebra de MRCDs.

Hecha esta adaptación, se obtendrán los diferentes operadores de reunión para el álgebra de MRCDs con un comportamiento similar al descrito para sus homólogos del álgebra relacional clásica. No obstante, se deberá tener en cuenta que los nuevos operadores, además de tener un comportamiento similar a nivel de tupla, afectarán a la cardinalidad y grado de certidumbre que éstas poseen en la MRCD resultante de su aplicación debido al marco multirrelacional en que se aplican.

Con respecto a la cardinalidad de las tuplas de la MRCD resultante de la aplicación de uno de estos operadores de reunión, se ha de destacar que ésta indica el número de emparejamientos que cada una de las tuplas representa (contabilizando los duplicados de la tuplas emparejadas que son indicados por la cardinalidad asociada a éstas en la MRCD de la que proceden). Esto resulta evidente cuando se trata de una reunión natural o una θ-reunión. En el caso de una semireunión, este hecho no resulta tan intuitivo ya que la MRCD resultante contiene exclusivamente a tuplas que pertenecen a una de las MRCD operando, por lo que puede resultar sorprendente encontrar una variación (con respecto a las que éstas poseen en la MRCD de la que provienen) en la cardinalidad asignada en dichas tuplas aun siendo este comportamiento coherente con la semántica del operador. Por último, para el caso de la reunión natural externa se habrá de tener en cuenta la cardinalidad asignada a la tupla en las MRCDs $\omega_{(r,r')}$ y $\omega_{(r',r)}$. Parece lógico que ésta sea $\{1.0/1\}_c$ para que actúe de forma neutra en el proceso de agregación con la cardinalidad de las tuplas desemparejadas.

En lo que se refiere al grado de certidumbre las tuplas de la MRCD resultante de la aplicación de uno de estos operadores de reunión, éste procederá de la agregación de los grados asociados a cada una de las tuplas emparejadas. Una vez más, este proceso de agregación resulta evidente para la reunión natural y la θ-reunión. Al igual que anteriormente, éste comportamiento puede resultar poco intuitivo en el caso de la semireunión. Téngase en cuenta que las tuplas pertenecientes a la MRCD resultante de la semireunión de dos MRCDs r y r', $(r \ltimes r')$, reflejan a aquellas tuplas de r que han podido emparejarse con alguna tupla en r'. Obsérvese, por tanto, que esta operación de emparejamiento ha de tener en cuenta los valores de los atributos comunes de las tuplas de la MRCD r', aunque éstos no formen parte de las tuplas de $(r \ltimes r')$. Por ello, parece lógico que se tenga en cuenta el grado de certidumbre asociado a estos valores mediante la agregación del mismo con el grado de certidumbre asociado a las tuplas de r. De esta forma, el nuevo grado de certidumbre asociado a cada tupla en la MRCD resultante indica bajo qué condiciones de certidumbre se ha realizado el emparejamiento (o los emparejamientos) que ésta representa. En lo que se refiere a la reunión natural externa, la tupla de las MRCDs $\omega_{(r,r')}$ y $\omega_{(r',r)}$ deberá tener asociado un grado de certidumbre de 1.0. De esta forma, al igual que se planteó en el caso de las cardinalidades, éste actuará como elemento neutro en el proceso de agregación del grado de certidumbre con las tuplas desemparejadas.

Además de lo anterior, se ha de hacer una consideración expresa en lo que se refiere a la aplicación del operador de diferencia de MRCDs en el caso de la antireunión. Como se mencionó en la sección dedicada a dicho operador, la semántica asociada al mismo en el marco relacional clásico es muy diferente a la que éste tiene cuando se aplica sobre MRCDs. En el primer caso, este operador

se emplea genéricamente como la negación del cuantificador existencial. Por ejemplo, en el caso de la antireunión de dos relaciones r y r', lo que se pretende al aplicar el operador de diferencia es obtener aquellas tuplas de r que *no existen* en $(r \ltimes r')$. Cuando el operador de diferencia se aplica sobre MRCDs éste no es siempre el caso ya que en el marco de las MRCDs se ha de tener en cuenta la cardinalidad asociada a las tuplas de sus operandos. De esta forma, el resultado de la aplicación del operador de diferencia no es eliminar de forma absoluta de la MRCD minuendo aquellas tuplas presentes en la MRCD substraendo si no substraer las cardinalidades asociadas a las mismas. Por tanto, es perfectamente posible que en la MRCD resultante de la diferencia de dos MRCDs esté presente una tupla que a su vez está presente en las MRCDs minuendo y substraendo. En el caso de las MRCDs, el operador de diferencia sólo se comporta como la negación del cuantificador existencial cuando las cardinalidades de las tuplas de la MRCD minuendo son menores o iguales que las asociadas a éstas en la MRCD substraendo. Éste, precisamente, es el caso de la expresión $r - (r \ltimes r')$ cuando r y r' son MRCDs. La cardinalidad de cualquier tupla en r que sea susceptible de ser emparejada con alguna tupla en r' siempre será menor o igual a la de ésta en $(r \ltimes r')$, ya que la aplicación del operador de semireunión sólo puede resultar en un incremento o mantenimiento de la cardinalidad asociada a estas tuplas. Téngase en cuenta que el operador de semireunión se expresa como el producto cartesiano de sus operandos junto con una aplicación posterior, sobre el resultado del producto, del operador de selección.

Finalmente, también se ha de considerar la naturaleza de los valores ausentes que se emplean en la tupla existente en las MRCDs $\omega_{(r,r')}$ y $\omega_{(r',r)}$ empleadas en la expresiones en las que se expresan los diferentes operadores de reunión natural externa considerados anteriormente. Como se indicó en el capítulo anterior, el presente modelo considera tres tipos de valores ausentes, por lo que habrá de determinarse cuál de ellos es empleado como valor en la única tupla de las MRCDs $\omega_{(r,r')}$ y $\omega_{(r',r)}$. Dado que la semántica de los tipos de valores ausentes *unk* y *ni* incluye la posibilidad de que el valor exista, no parece lógico que se emplee estos en el contexto de los operadores de reunión natural externa, en el que los valores ausentes se emplean concretamente para indicar la carencia de un valor para el atributo. Precisamente, ese es el cometido de los valores ausentes de tipo *dne*, por lo que será este tipo de valores el que se deba emplear en la tupla de las citadas MRCDs.

Hechas las consideraciones anteriores, la adaptación de las distintas variaciones del operador de reunión consideradas anteriormente es directa. Por motivos de brevedad no se incluirá en este trabajo cada una de ellas. A modo de ejemplo, y aprovechando que se requerirá en secciones posteriores, se incluirá la definición del operador de reunión natural externa izquierda en el ámbito del álgebra de MRCDs. Ésta es como se indica a continuación.

Definición 6.38 (Reunión natural externa izquierda)**.** *Sean m_1 y m_2 dos MRCDs tales que un subconjunto de sus atributos son compatibles, esto es, $A_{m_1} \cap A_{m_2} \neq \emptyset$ y $dom_{m_1}^{|(A_{m_1} \cap A_{m_2})} = dom_{m_2}^{|(A_{m_1} \cap A_{m_2})}$.*

La reunión natural externa izquierda *de m_1 y m_2, notada como $(m_1 \Rightarrow\!\!\!\bowtie m_2)$, es una MRCD definida como se indica en la Ecuación 6.90.*

$$(m_1 \Rightarrow\!\!\!\bowtie m_2) = (\mathcal{H}_{(m_1 \Rightarrow\!\!\!\bowtie m_2)}, \mathcal{B}_{(m_1 \Rightarrow\!\!\!\bowtie m_2)}) \tag{6.90}$$

En lo anterior, la cabecera de $m_1 =\!\!\bowtie m_2$ está definida como se indica en la Ecuación 6.91 y su cuerpo según se indica en la Ecuación 6.92.

$$\mathcal{H}_{(m_1=\!\!\bowtie m_2)} = (A_{(m_1=\!\!\bowtie m_2)}, dom_{(m_1=\!\!\bowtie m_2)}) \tag{6.91}$$

$$\mathcal{B}_{(m_1=\!\!\bowtie m_2)} = (T_{(m_1=\!\!\bowtie m_2)}, U_{(m_1=\!\!\bowtie m_2)}) \tag{6.92}$$

Los componentes de cabecera y cuerpo están definidos como sigue:

- *El conjunto de atributos de $(m_1 =\!\!\bowtie m_2)$ de define como se indica en la Ecuación 6.93.*

$$A_{(m_1=\!\!\bowtie m_2)} = A_{m_1} \cup A_{m_2} \tag{6.93}$$

- *Su función dominio, $dom_{(m_1=\!\!\bowtie m_2)}$, está definida según se indica en la Ecuación 6.94.*

$$dom_{(m_1=\!\!\bowtie m_2)}(a) = \begin{cases} dom_{m_1}(a) & si\ a \in A_{m_1} \\ dom_{m_2}(a) & si\ a \in A_{m_2} \end{cases} \qquad \forall a \in A_{(m_1=\!\!\bowtie m_2)} \tag{6.94}$$

- *El conjunto de tuplas de la MRCD resultante, $T_{(m_1=\!\!\bowtie m_2)}$, se define según la función característica que se muestra en la Ecuación 6.95*

$$\omega_{T_{(m_1=\!\!\bowtie m_2)}}(t) = \\ = \begin{cases} \omega_{T_{m_1}}(t_1) \boxtimes T_{m_2}(t_2) & si\ \exists! t_1 \in T_{m_1} | (t_1^{|A_{m_1}} = t^{|A_{m_1}}) \wedge \\ & \qquad \exists! t_2 \in T_{m_2} | (t_2^{|A_{m_2}} = t^{|A_{m_2}}) \\ \omega_{T_{m_1}}(t_1) & si\ \exists! t_1 \in T_{m_1} | (t_1^{|A_{m_1}} = t^{|A_{m_1}}) \wedge \\ & \qquad \nexists t_2 \in T_{m_2} | (t_2^{|A_{m_2}} = t^{|A_{m_2}}) \wedge \\ & \qquad \forall a \in (A_{m_2} - A_{m_1}), (t(a) = dne_{dom_{m_2}(a)}) \\ \{1.0/0\}_2 & en\ caso\ contrario \end{cases}$$

$$\forall t \in \mathbb{T}_{(m_1=\!\!\bowtie m_2)} \tag{6.95}$$

- *La función de certidumbre de la MRCD resultante, $U_{m_1=\!\!\bowtie m_2}$, se define tal y como se indica en la Ecuación 6.96.*

$$U_{(m_1=\!\!\bowtie m_2)}(t) = \begin{cases} U_{m_1}(t^{|A_{m_1}}) \otimes U_{m_2}(t^{|A_{m_2}}) & si\ t^{|A_{m_2}} \in T_{m_2} \\ U_{m_1}(t^{|A_{m_1}}) & en\ caso\ contrario \end{cases}$$

$$\forall t \in T_{(m_1=\!\!\bowtie m_2)} \tag{6.96}$$

Obsérvese que, en la anterior definición, las condiciones de aplicación de los dos primeros casos de la Ecuación 6.95 exigen la unicidad de las tuplas t_1 y t_2 en, respectivamente, las MRCDs m_1 y m_2. Debido a que las condiciones impuestas a estas tuplas determinan los valores que éstas han de tener asociados a la totalidad de sus atributos, la unicidad exigida será satisfecha (siempre que éstas existan) ya que las MRCDs a las que pertenecen han de estar en FNPE. Lo contrario implicaría que dos tuplas de una misma MRCD poseerían una misma signatura, lo cual incurriría en una contradicción. Este misma situación se da para la condición de aplicación del primer caso de la Ecuación 6.96.

Ilustraremos el funcionamiento del operador de reunión natural externa izquierda, formalizado anteriormente, empleando el siguiente ejemplo.

Ejemplo 6.7. *Continuemos empleado el marco usado en los ejemplos 6.3 y 6.6, en el que se deseaba cruzar la información de las reservas de eventos que maneja el centro de negocios (representada en la MRCD r'') y la de las reservas de material necesario para la celebración de cada una de la sesiones paralelas asociadas a dichos eventos (información que está contenida en la MRCD m). Ambas MRCDs, r'' y m, están definidas en el Ejemplo 6.3.*

En el Ejemplo 6.6 la información de r'' y m se cruzó aplicando sobre las mismas el operador de reunión natural. La MRCD resultante del proceso de reunión se describió en dicho ejemplo, estando su cuerpo concretamente definido según se indica en la Tabla 6.13. En dicho cuerpo se muestran una serie de tuplas que proceden del emparejamiento de las pertenecientes a las MRCDs r'' y m, representando éstas los requisitos materiales totales necesarios para la celebración de cada uno de los eventos representados en r''. Se ha de notar que en dicha MRCD no se muestran aquellos eventos que no requieren de ningún material para la celebración de sus reuniones paralelas, ya que estos no poseen correspondencia con tupla alguna en m.

Supongamos que deseamos que la MRCD procedente de la reunión de r'' y m descrita anteriormente incluya de forma explícita (al contrario de como sucede en el caso descrito anteriormente) a aquellos eventos que no tienen asociado ningún requisito material. Para ello, sustituiremos el operador de reunión natural aplicado en el Ejemplo 6.6 por el operador de reunión natural externa izquierda. De esta forma, procederemos a aplicar la expresión algebraica $(r'' =\!\!\bowtie m)$. Esta expresión resulta en una MRCD cuya cabecera es idéntica a la descrita para $(r'' \bowtie m)$ en el Ejemplo 6.6, ya que la diferencia entre ambos operadores radica exclusivamente en la forma de determinar las tuplas que forman parte del cuerpo de la MRCD resultante del proceso de reunión. El cuerpo de la MRCD $(r'' =\!\!\bowtie m)$ es de la forma que se indica en la Tabla 6.14.

Obsérvese que el cuerpo de $(r'' =\!\!\bowtie m)$ posee una tupla más que el de $(r'' \bowtie m)$. Ésta, la cuarta según el orden en que se muestran las tuplas en la Tabla 6.14, se corresponde con la tupla tercera de r'' (según el orden en que se muestran las tuplas en la Tabla 6.6). Ésta no posee correspondencia en m, por lo que quedó excluida del cuerpo de $(r'' \bowtie m)$. La forma en la que ésta tupla se refleja en el cuerpo de $(r'' =\!\!\bowtie m)$ muestra tal falta de correspondencia, concretamente por el valor ausente que refleja el atributo material.

$U_{(r'' \Rightarrow\!\bowtie m)}$	$T_{(r'' \Rightarrow\!\bowtie m)}$	$t_{(r'' \Rightarrow\!\bowtie m)}$	
		fecha	material
1.0	$\{1.0/2, 0.6/5, 0.3/6\}_c$	4-6-2010	videoproyector
1.0	$\{1.0/40, 0.6/125, 0.3/180\}_c$	4-6-2010	cuaderno
0.5	$\{0.3/3\}_c$	14-9-2010	videoproyector
1.0	$\{1.0/2\}_c$	21-11-2010	$dne_{D_{material}}$
0.5	$\{0.3/20\}_c$	9-12-2010	cuaderno

Tabla 6.14: Cuerpo de la MRCD $(r'' \Rightarrow\!\bowtie m)$

6.4.2. Intersección

El operador de intersección, aplicado sobre un par de relaciones compatibles, resulta en una relación de la que forman parte aquellas tuplas existentes en ambos operandos. En la presente sección estudiaremos la forma en que dicho operador puede ser expresado en función de operadores básicos del álgebra relacional y su transposición al álgebra de MRCDs.

6.4.2.1. El operador de intersección en función de operadores básicos del álgebra relacional

Este operador puede ser expresado en base a operadores básicos del álgebra relacional de la siguiente forma.

Definición 6.39 (Intersección en función de operadores básicos del álgebra relacional)**.** *Sean r y r' dos relaciones compatibles.*

Podemos definir la intersección *de r y r' en base a los operadores básicos del álgebra relacional según se indica en la Ecuación 6.97.*

$$r \cap r' = r - (r - r') \tag{6.97}$$

Obsérvese que en la definición anterior el operador de intersección ha sido definido exclusivamente en base al operador de diferencia. El objetivo de tal expresión es hallar aquellas tuplas de r que no existen en r', y posteriormente eliminar de r aquellas tuplas que no existen en el conjunto anterior. Una vez más, y doblemente en este caso, el operador de diferencia de relaciones se ha empleado como la negación del cuantificador existencial. Como ya se indicó en el caso del operador de antireunión, este uso del operador de diferencia puede no ser equivalente en el contexto del álgebra de MRCDs y, por tanto, significar alguna dificultad en la transposición del operador de intersección al citado contexto.

De hecho, el uso que se hace del operador de diferencia en la Definición 6.39 como negación de cuantificador existencial no es equiparable en el contexto de la expresión indicada en la Ecuación 6.97 cuando r y r' son MRCDs. Como se indicó anteriormente, en el caso del operador de antireunión este uso era equiparable porque se puede asegurar que, para las tuplas comunes de las MRCDs participantes, sus cardinalidades asociadas en la MRCD minuendo son menores o iguales a las que estas poseen en la MRCD sustraendo. En cambio, en el caso de la expresión indicada en la Ecuación 6.97, tal condición no se cumple

de manera general. Este hecho, por tanto, impide la definición del operador de intersección en función de los operadores básicos del álgebra de MRCDs según la equivalencia indicada para el caso del álgebra relacional clásica mostrado anteriormente.

6.4.2.2. El operador de intersección en el álgebra de multirrelaciones complejas difusas

Al no ser posible definir el operador de intersección en función de operadores básicos del álgebra de MRCDs, éste deberá definirse de forma específica dentro del contexto de las MRCDs. Para ello, será necesario considerar diversos aspectos del funcionamiento de éste operador.

Como se indicó al inicio del presente apartado, el cometido básico del operador de intersección es detectar aquellas tuplas que están presentes en sus dos MRCDs operandos o, lo que es lo mismo, los pares de tuplas (donde las tuplas del par proceden de uno y otro operando) que son equivalentes. Por tanto, antes de abordar ésta definición habrá de determinarse en qué forma consideraremos equivalentes dos tuplas dadas. Parece lógico que en este aspecto apliquemos, como se ha hecho en el caso del operador de unión, la equivalencia de tuplas en función de su signatura.

Cada pareja de tuplas equivalente dará lugar a una tupla en la MRCD resultante de la aplicación del operador de intersección. Como en el caso del operador de unión MRCDs, será necesario determinar una única tupla que represente dicho par cuando, a pesar de que las signaturas de las tuplas coincidan, no coincidan los valores de todos sus atributos. Para este cometido podremos hacer uso del concepto de *tupla representante* definido en el apartado dedicado al operador de unión de MRCDs, ya que el caso es equivalente.

Además de determinar una tupla representante para cada par, será necesario determinar la manera en que se agregarán las cardinalidades y grados de certidumbre asociados a ambas tuplas. En primer lugar, para el caso de la agregación de cardinalidades, emplearemos como referencia al operador de intersección de multiconjuntos difusos. Éste emplea la función mínimo para agregar los pares de cardinalidades, elección que es coherente con la semántica asociada al operador. Por tanto, la presente propuesta aplicará la misma función. En el caso de la agregación de grados de certidumbre, habrá que tener en cuenta que la tupla representante es una agregación conjuntiva de las tuplas representadas. Por tanto, para mantener la semántica del proceso de agregación de tuplas, parece lógico que los grados de certidumbre sobre los valores de los atributos asociados al par de tuplas sean agregados empleando una t-norma.

Según lo anterior, definiremos formalmente el operador de intersección de MRCDs como se indica a continuación.

Definición 6.40 (Intersección)**.** *Sean m_1 y m_2 dos MRCDs compatibles. La unión de m_1 y m_2, notada como $(m_1 \cap m_2)$, es una MRCD de la forma que se indica en la Ecuación 6.98.*

$$(m_1 \cap m_2) = (\mathcal{H}_{m_1}, \mathcal{B}_{(m_1 \cap m_2)}) \tag{6.98}$$

En la anterior ecuación, el cuerpo de $(m_1 \cap m_2)$ está definido según se indica en la Ecuación 6.99.

$$\mathcal{B}_{(m_1 \cap m_2)} = (T_{(m_1 \cap m_2)}, U_{(m_1 \cap m_2)}) \tag{6.99}$$

Los componentes del cuerpo de $m_1 \cap m_2$ están definidos de la siguiente forma:

- *El conjunto de tuplas de $(m_1 \cap m_2)$, $T_{(m_1 \cap m_2)}$, está definido según la función característica que se indica en la Ecuación 6.100.*

$$\omega_{T_{(m_1 \cap m_2)}}(t) = \\ = \begin{cases} \text{mín}(\omega_{T_{m_1}}(t_1), \omega_{T_{m_2}}(t_2)) & si\ \exists! t_1 \in T_{m_1} | (r_{[t_1]_{(T_{m_1} \cup T_{m_2})}} = t) \wedge \\ & \qquad \exists! t_2 \in T_{m_2} | (r_{[t_2]_{(T_{m_1} \cup T_{m_2})}} = t) \\ \{1.0/0\}_2 & en\ otro\ caso \end{cases}$$

$$\forall t \in \mathbb{T}_{(m_1 \cap m_2)} \tag{6.100}$$

- *La función de certidumbre de la MRCD resultante de la intersección, $U_{(m_1 \cup m_2)}$, es de la forma que se indica en la Ecuación 6.101.*

$$U_{(m_1 \cap m_2)}(t) = U_{m_1}(t_1) \otimes U_{m_2}(t_2)$$

$$\forall t \in T_{(m_1 \cup m_2)} | (\exists! t_1 \in T_{m_1} | (r_{[t_1]_{(T_{m_1} \cup T_{m_2})}}) \wedge \\ \exists! t_2 \in T_{m_2} | (r_{[t_2]_{(T_{m_1} \uplus T_{m_2})}} = t)) \tag{6.101}$$

Como se indicó en el caso del operador de unión, en las ecuaciones 6.100 (en la condición que determina la aplicación de su primer caso) y 6.101 (en la condición que determina la aplicabilidad de la misma) se requiere que las tuplas t_1 y t_2 sean las únicas tuplas de m_1 y m_2, respectivamente, cuya tupla representante sea la tupla t. Se ha de remarcar que la unicidad de estas tuplas, cuando éstas existen, se puede garantizar gracias a que las MRCDs m_1 y m_2 han de estar en FNPE. Debido a ello, no pueden existir en dichas MRCDs dos tuplas con una misma signatura y, en definitiva, con una misma tupla representante.

Ilustraremos el funcionamiento del operador de intersección de MRCDs empleando el siguiente ejemplo.

Ejemplo 6.8. *Siguiendo con el marco descrito en el Ejemplo 6.4, en el que se pretende obtener un informe consolidado por años de las reservas registradas a partir de las tablas de reserva de cada una de las sucursales del centro de negocios, buscaremos ahora obtener un informe anual sobre las características mínimas, en lo que se refiere a número de reuniones paralelas, de dichas reservas.*

Para realizar esta tarea, supondremos, por simplicidad y siguiendo el marco anteriormente mencionado, que el centro de negocios tiene asociadas dos sucursales que denominaremos A y B. Los datos anuales de las reservas de estas

sucursales se encuentran reflejados en las MRCDs r_A y r_B, MRCDs que fueron definidas en el Ejemplo 6.4. Nótese que r_A y r_B son MRCDs compatibles.

Podremos obtener el informe requerido a partir de las MRCDs r_A y r_B realizando la intersección de las mismas. La MRCD resultante, debido a la semántica que este operador tiene en el contexto del álgebra de MRCDs, contendrá las tuplas comunes a ambas MRCDs con una cardinalidad asociada que se corresponde con el mínimo de la que éstas tienen asociada en las MRCDs participantes.

La intersección de r_A y r_B, $(r_A \cap r_B)$, es una MRCD cuya cabecera es de la misma forma que lo es la cabecera de las MRCDs participantes. Con respecto al cuerpo de $(r_A \cap r_B)$, éste está definido según se indica en la Tabla 6.15.

Obsérvese cómo el proceso para la determinación de las tuplas comunes en r_A y r_B se basa en la búsqueda de tuplas con signatura idéntica y no en la igualdad exhaustiva del valor de todos los atributos de las tuplas. De hecho, las tuplas en los pares detectados en el anterior proceso, aún con una misma signatura, no poseen un valor igual para todos los atributos. Concretamente, las tuplas primeras de las MRCDs r_A y r_B (según el orden en que éstas se muestran respectivamente en las tablas 6.2 y 6.10) son emparejadas, aún no coincidiendo todos los valores de sus atributos, ya que comparten una misma signatura. El mismo caso se da para el par formado por las tuplas tercera de r_A y segunda de r_A.

Obsérvese también que la discrepancia en los componentes que no forman la signatura en las tuplas emparejadas provoca que éstas se vean representadas en la MRCD $(r_A \cap r_B)$ mediante una tupla que agrega sus valores. Concretamente, el par de formado por la primeras tuplas de r_A y r_B está representado por la primera tupla de $(r_A \cap r_B)$ (siguiendo el orden en que éstas se muestran en la Tabla 6.15), la cual posee unos valores para los atributos no signatura que proceden de la agregación de los valores asociados a las tuplas originales. Este mismo caso se da para la segunda tupla de $(r_A \cap r_B)$, que representa a las tuplas del par formado por la tercera tupla de r_A y la segunda tupla de r_B.

Nótese que las cardinalidades asociadas a las dos tuplas del cuerpo de la MRCD $(r_A \cap r_B)$ se corresponden con el mínimo de las cardinalidades de asociadas a las tuplas que forman el par que éstas representan. De la misma forma, el grado de certidumbre asociado a las tuplas de la MRCD $(r_A \cap r_B)$ procede de la agregación, empleando una t-norma, del grado de las tuplas que forman el par que éstas representan. Se ha de destacar que en el presente ejemplo se ha empleado como t-norma la función mínimo. La agregación de grados de certidumbre queda ilustrada en el grado de certidumbre asociado a la segunda tupla de $(r_A \cap r_B)$, ya que las tuplas que forman el par que ésta representa no poseen grados de certidumbre coincidentes.

6.4.3. División

El operador de división es un operador relacional binario no conmutativo. Por simplicidad, siempre diremos que éste se aplica sobre un par de relaciones que denominaremos r y r' y que respectivamente harán las veces de operador izquierdo y derecho. Para la aplicación del mismo, y al igual que ocurre con otros operadores como el producto cartesiano, la unión o la intersección, los conjuntos de atributos de sus operandos han de cumplir una determinada condición. En el caso de la división, el conjunto de atributos de r' ha de ser un subconjunto del conjunto de atributos de r, esto es $A_{r'} \subset A_r$. La aplicación de este operador

$U_{(r_A \cap r_B)}$	$T_{(r_A \cap r_B)}$	$t_{(r_A \cap r_B)}$				
		año	requisitos			
			asistentes	catering	intérpretes	
0.5	$\{1.0/1, 0.3/2\}_c$	2010	$entre(20, 26)$	$\langle 0.5, \text{sí} \rangle$	$\{1.0/9, 0.75/11, 0.5/19\}_c$	Inglés
					$\{1.0/3, 0.75/4, 0.5/6\}_c$	Alemán
					$\{1.0/1, 0.75/5\}_c$	Francés
					$\{0.5/4\}_c$	Italiano
1.0	$\{0.3/2\}_c$	2011	$hasta(10)$	$\langle 0.5, \text{no} \rangle$	$\{0.75/3\}_c$	Italiano

Tabla 6.15: Cuerpo de $(r_A \cap r_B)$

resultará en una relación, que notaremos como $(r \div r')$, cuyas tuplas representarán a una parte de los distintos conjuntos de tuplas de r que poseen un valor común para los atributos que posee exclusivamente r. El criterio de selección de estos conjuntos de tuplas será satisfecho sólo por aquellos conjuntos para los que, por cada tupla en r', exista al menos una del conjunto con un valor similar para los atributos que r y r' comparten.

De manera más informal, se puede ver la división como un operador que permite seleccionar de entre un grupo de elementos (representados por las tuplas de r), aquellos que cumplen todo un conjunto de requisitos (representados por las tuplas de r').

6.4.3.1. El operador de división en función de operadores básicos del álgebra relacional

Este operador puede ser expresado en base a operadores básicos del álgebra relacional tal y como se muestra a continuación.

Definición 6.41 (División en función de operadores básicos del álgebra relacional)**.** *Sean r y r' dos relaciones tales que $A_{r'} \subset A_r$.*

Podemos definir la división *de r y r' en base a los operadores básicos del álgebra relacional según se indica en la Ecuación 6.102.*

$$r \div r' = \pi_{(A_r - A'_r)}(r) - \pi_{(A_r - A'_r)}((\pi_{(A_r - A'_r)}(r) \times r') - r) \tag{6.102}$$

La expresión equivalente al operador de división indicada anteriormente resulta compleja. No obstante, ésta puede ser fácilmente comprendida si se conoce el objetivo de cada una de sus subexpresiones.

La descomposición en subexpresiones está descrita en la secuencia de ecuaciones comenzando por la Ecuación 6.103 y finalizando en la Ecuación 6.106. Ésta facilita la comprensión, permite brevedad y claridad de la notación. En primer lugar, se ha de notar que la expresión s, mostrada en la Ecuación 6.103, busca obtener una relación cuyas tuplas representan a todos los conjuntos de tuplas de r que poseen un valor común para los atributos del conjunto $(A_r - A_{r'})$. Posteriormente, la expresión p, mostrada en la Ecuación 6.104, se empleará para obtener una relación compatible con r cuyas tuplas representan todas las posibles combinaciones entre los conjuntos de tuplas de r, representados por las tuplas de s, y las tuplas en r'. La aplicación del operador de diferencia en la expresión mostrada en la Ecuación 6.105 busca obtener aquellas combinaciones posibles de las tuplas en s y r' que no existen en r. Posteriormente, la aplicación del operador de proyección sobre $(p - r)$, para obtener q, resultará en una relación cuyas tuplas representarán a los conjuntos de tuplas de r, que son representados en su totalidad por las tuplas de s, que no satisfacen el criterio de selección descrito al inicio del presente apartado. Finalmente, la expresión $s - q$ en la Ecuación 6.106 resultará en una relación que incluirá sólo a aquellas tuplas de s que cumplen el citado criterio de selección.

$$s = \pi_{(A_r - A'_r)}(r) \tag{6.103}$$

$$p = s \times r' \tag{6.104}$$

$$q = \pi_{(A_r - A'_r)}(p - r) \tag{6.105}$$

$$r \div r' = s - q \tag{6.106}$$

Nótese que, al igual que en el caso de las equivalencias mostradas para la antireunión y la intersección, el operador de diferencia entre relaciones se emplea como la negación del cuantificador existencial en la Ecuación 6.102, concretamente en las subexpresiones 6.105 y 6.106 en las que ésta se descompone. Se ha remarcar, una vez más, que este uso del operador de diferencia no es equiparable de manera general en el marco del álgebra de MRCDs. Concretamente, en el caso de la expresión mostrada en la Ecuación 6.105 la equiparación de este uso no puede ser garantizada, ya que es posible que las tuplas comunes de p y r posean una cardinalidad mayor en la primera MRCD que en la segunda. Se ha de tener en cuenta que las tuplas de p proceden de la combinación de las tuplas de s y de las de r'. Las tuplas de s, al proceder esta MRCD de la aplicación de operador de proyección sobre r, tendrán una cardinalidad igual o superior a las tuplas de las que éstas proceden. Además, al aplicar el producto cartesiano $s \times r'$ para obtener p, esta cardinalidad podrá incrementarse nuevamente.

La imposibilidad de garantizar, en el caso concreto de la expresión mostrada en la Ecuación 6.102, la equiparación del operador de diferencia entre MRCDs como la negación del cuantificador existencial, hace dicha expresión inaplicable en el ámbito de las MRCDs.

6.4.3.2. El operador de división en el álgebra de multirrelación complejas difusas

Dada la imposibilidad de la definición del operador de división en base a operadores básicos del álgebra de MRCDs, será necesario definir éste dentro del contexto de las MRCDs de forma específica. Antes de crear la citada definición, será necesario tener en cuenta diversos aspectos del funcionamiento de éste operador.

En primer lugar, será necesario estipular la forma en que se determinarán los diferentes conjuntos de tuplas en r que tienen asociado un mismo valor para los atributos que posee exclusivamente dicha MRCD. Como se ha discutido a lo largo de este trabajo, la identidad de una tupla en el modelo relacional viene dada por el valor de sus atributos. En cambio, en el contexto de la MRCDs, la identidad de una tupla está determinada por su signatura. Por tanto, parece lógico que la transposición del operador de división al citado ámbito, determine los diferentes conjuntos en los que se dividen las tuplas de r para esta operación algebraica en función de la signatura restringida de las mismas y no en función de los valores de todos sus atributos. Este cambio implicará que cada conjunto de tuplas de r deba ser representado en la MRCD $(r \div r')$ por una tupla cuyos valores asociados a sus atributos procedan de la agregación de los valores (posiblemente distintos) asociados a dichos atributos para las tuplas del conjunto que ésta representa. Este tipo de agregación coincide con el empleado para la determinación de tuplas representantes en el caso de operador de proyección. Finalmente, este cambio en la forma en que se determinarán los conjuntos de tuplas de r y sus representantes en $(r \div r')$ asegurará que esta última MRCD estará en FNPE.

En segundo lugar, se ha de tener en cuenta que la consideración anterior sobre la identificación de tuplas por su signatura, y no sobre el valor de todos sus atributos, implica que el criterio de selección que aplica el operador de división sobre los conjuntos de tuplas en que se divide r se basará en la signatura restringida al conjunto de atributos de r' de las tuplas en r en lugar de en el

valor de todos los atributos de estas.

En tercer lugar, se debe determinar el significado de las cardinalidades y grados de certidumbre asociados a las tuplas de las MRCDs participantes en la operación de división y en la MRCD resultante de la misma. Parece natural que, en la trasposición del operador de división en el ámbito de las MRCDs, se aprovechen las capacidades expresivas adicionales que ofrecen las cardinalidades asociadas a sus tuplas para indicar restricciones cuantitativas en r'. Estas restricciones cuantitativas permitirán indicar la cardinalidad mínima que ha de poseer cualquier tupla de r que se emplee para satisfacer la condición que imponga una tupla en r'. Dado que las cardinalidades que se emplean son difusas, éstas, además de las citadas restricciones cuantitativas, permitirán la especificación de restricciones cualitativas. Como se verá en el ejemplo incluido en el presente apartado, éstas restricciones cualitativas podrán ser aplicadas de forma flexible.

Según los planteamientos anteriores, el criterio de selección de los conjuntos de tuplas de r se reformulará, dentro del ámbito de álgebra de las MRCDs, con respecto al que se aplica en el ámbito relacional y se subdivide en dos condiciones referentes a la signatura restringida y a la cardinalidad asociada a la tupla evaluada. Éste se satisfará si, por cada tupla t' de r', existe al menos una tupla t del conjunto de tuplas evaluado cuya signatura restringida al conjunto de atributos de r' coincide con la signatura de t' y la cardinalidad asociada a t en r es igual o superior a la asociada a t' en r'.

En cuarto lugar, podemos considerar que el operador de división en el ámbito de las MRCDs puede ofrecer un resultado más rico en el que se indique, por cada conjunto de tuplas de r, la cantidad (difusa) de veces que puede éste satisfacer los requisitos impuestos por r'. Éste precisamente será el significado asociado a la cardinalidad difusa de cada tupla en $(r \div r'))$. De esta forma, la cardinalidad de cada tupla en la MRCD $(r \div r')$ se obtendrá como el mínimo de ocasiones que cada tupla del conjunto que ésta representa puede satisfacer los requisitos cuantitativos del criterio de selección. A su vez, este número de ocasiones en que se satisfacen los requisitos se obtendrá para cada tupla t de r aplicando el operador de división optimista de cardinalidades difusas empleando como dividendo la cardinalidad asociada a t en r y como divisor la cardinalidad asociada a la tupla t' de r' con la que t se empareja.

En quinto y último lugar, se ha de considerar cómo se determinará el grado de certidumbre de las tuplas en la MRCD $(r \div r')$. Debido a que los valores de éstas tuplas proceden de la agregación de los valores asociados a las tuplas de uno de los conjuntos de tuplas en que se divide r, parece lógico que el grado de certidumbre asociado a éstas sea una agregación conjuntiva de los grados de certidumbre de cada una de las tuplas de las que procede el valor de sus atributos. No obstante, dado que una parte de las condiciones impuestas por cada tupla t' de r' están basadas en el valor de sus atributos signatura, parece lógico que también se considere el grado de certidumbre asociado a los citados valores. En definitiva, el grado de certidumbre de una tupla de la MRCD $(r \div r')$ debe indicar bajo qué condiciones de certidumbre se cumplieron las condiciones establecidas por las tuplas de r'. Para ello, el grado de certidumbre asociado a las tuplas de r' deberá ser agregado al asociado a las tuplas de cada conjunto de r con las que se emparejan. Estas agregaciones, debido a su carácter conjuntivo, se realizarán aplicando una t-norma.

Según lo anterior, definiremos formalmente el operador de división de MRCDs

como se indica a continuación.

Definición 6.42 (División). *Sean m_1 y m_2 dos MRCDs tales que $A_{m_2} \subset A_{m_1}$.*

La división de m_1 entre m_2, notada como $(m_1 \div m_2)$, es una MRCD de la forma que se indica en la Ecuación 6.107.

$$(m_1 \div m_2) = (\mathcal{H}_{(m_1 \div m_2)}, \mathcal{B}_{(m_1 \div m_2)}) \tag{6.107}$$

La cabecera y cuerpo de $(m_1 \div m_2)$ son de la forma indicada en las ecuaciones 6.108 y 6.109 respectivamente.

$$\mathcal{H}_{(m_1 \div m_2)} = (A_{(m_1 \div m_2)}, dom_{(m_1 \div m_2)}) \tag{6.108}$$

$$\mathcal{B}_{(m_1 \div m_2)} = (T_{(m_1 \div m_2)}, U_{(m_1 \div m_2)}) \tag{6.109}$$

Los componentes de la cabecera y el cuerpo anteriores son como se indica a continuación:

- *El conjunto de los atributos de $(m_1 \div m_2)$ se define como se indica en la Ecuación 6.110.*

$$A_{(m_1 \div m_2)} = A_{m_1} - A_{m_2} \tag{6.110}$$

- *La función dominio de $(m_1 \div m_2)$, $dom_{(m_1 \div m_2)}$, es de la forma indicada en la Ecuación 6.111.*

$$dom_{(m_1 \div m_2)}(a) = dom_{m_1}^{|(A_{m_1} - A_{m_2})} \qquad \forall a \in A_{(m_1 \div m_2)} \tag{6.111}$$

- *El multiconjunto difuso de tuplas de $(m_1 \div m_2)$ está definido según la función característica que se indica en la Ecuación 6.112. En dicha ecuación, c es una función definida como se indica en la Ecuación 6.113 y el operador $\boxslash$ representa la división optimista de cardinalidades difusas.*

$$\omega_{T_{(m_1 \div m_2)}}(t) = \inf_{t_2 \in T_{m_2}} (c(t_2, t, m_1) \boxslash \omega_{T_{m_2}}(t_2))$$

$$\forall t \in \mathbb{T}_{(m_1 \div m_2)} \tag{6.112}$$

$$c(t_2, t, m_1) = \begin{cases} \omega_{T_{m_1}}(t_1) & si\ \exists! t_1 \in T_{m_1} | (r_{[t_1]^{|(A_{m_1} - A_{m_2})}_{T_{m_1}}} = t \wedge \\ & \qquad sig^{|A_{m_2}}(t_1) = sig(t_2) \\ \{1.0/0\}_2 & en\ caso\ contrario \end{cases}$$

$$\forall m_1 \in \Delta, \forall t_2 \in \mathbb{T}_{m_2} | (m_2 \in \Delta \ \wedge \ A_{m_2} \subset A_{m_1}), \forall t \in \mathbb{T}_{(m_1 \div m_2)} \tag{6.113}$$

- *La función de certidumbre de la MRCD* $(m_1 \div m_2)$, $U_{(m_1 \div m_2)}$, *es de la forma que se indica en la Ecuación 6.114.*

$$U_{(m_1 \div m_2)}(t) = \left(\bigotimes_{\substack{t_1 \in T_{m_1} | \\ (r_{[t_1]_{T_{m_1}}|(A_{m_1} - A_{m_2})} = t)}} U_{m_1}(t_1) \right) \otimes \left(\bigotimes_{t_2 \in T_{m_2}} U_{m_2}(t_2) \right)$$

$$\forall t \in T_{(m_1 \div m_2)} \quad (6.114)$$

Se ha de destacar en la definición anterior, el papel que juega la función c en la Ecuación 6.112. Esta función c facilita la determinación, dada una tupla t_2, de la tupla t_1 (si es que existe) que satisface el requisito que ésta representa. Cuando existe una tupla t_1 que satisfaga el requisito establecido por t_2, en el conjunto de tuplas de m_1 representadas en $(m_1 \div m_2)$ por la tupla t, la función c devolverá la cardinalidad asociada en m_1 a la citada tupla t_1. En caso contrario, la función devolverá cero.

Asimismo, se ha de remarcar que la unicidad exigida para el cumplimiento del primer caso de la Ecuación 6.113 siempre es satisfecha en el caso de que exista una tupla t_1 de m_1 que cumpla las condiciones adicionales establecidas. Esto es debido a que dichas condiciones determinan el valor que t_1 ha de tener asociado para todos sus atributos signatura (la primera condición para los atributos signatura en el conjunto $(A_{m_1} - A_{m_2})$ y la segunda para los atributos signatura pertenecientes a A_{m_2}). Dado que las anteriores condiciones determinan la signatura de t_1, podemos afirmar que no habrá otra tupla en m_1 que las cumpla ya que lo contrario implicaría que m_1 no está en FNPE.

Finalmente, se ha de indicar que la definición de la función de certidumbre de $(m_1 \div m_2)$, en la Ecuación 6.114, se presenta de forma simplificada. En lugar de realizar en primer lugar la agregación de los grados de certidumbre de las tuplas de m_1 y m_2 que son emparejadas y posteriormente agregar los grados de certidumbre resultantes, se ha optado por reagrupar las agregaciones (aprovechando la propiedad asociativa de las t-normas) de tal forma que la expresión resulte más sencilla.

Ilustraremos el funcionamiento de operador de división de MRCDs, descrito anteriormente, con el siguiente ejemplo.

Ejemplo 6.9. *Como se ha indicado anteriormente, el operador de división se emplea fundamentalmente para obtener el subconjunto de entidades, de un conjunto dado, que satisfacen la totalidad de una serie de condiciones. Para ilustrar el funcionamiento de este operador plantearemos un escenario, dentro del marco del centro de negocios, en el que se desea determinar el subconjunto de agencias de traducción, de la cartera del centro de negocios, que pueden ser subcontratadas para encargarse de los servicios de interpretación requeridos para una sesión paralela, dentro de un evento organizado por el citado centro. Para ello, contaremos con una MRCD que representa la cartera de agencias de interpretación que trabajan para el centro de negocios y otra que representa los requisitos, en lo que*

respecta a servicios de interpretación, de cada sesión paralela de un determinado evento.

La primera de estas MRCDs, que como se ha mencionado anteriormente representa la cartera de agencias de interpretación del centro de negocios, será denominada r_C. Las tuplas de r_C indican, para cada una de las agencias, la disponibilidad de intérpretes para un determinado idioma. La cardinalidad de cada una ellas representa el número concreto de intérpretes disponibles que tiene la agencia correspondiente para el idioma indicado. Los grados de las diferentes cardinalidades difusas indican el nivel de servicios de interpretación que los intérpretes pueden prestar. La semántica asociada a estos grados será la misma que la indicada en la Tabla 5.9, definida en el Ejemplo 5.17. El grado de certidumbre asociado a cada tupla indica el nivel de confirmación de los datos que en ella figuran. La semántica asociada a estos grados se corresponde con la indicada en la Tabla 5.7. La MRCD r_C posee una cabecera compuesta por el conjunto de atributos indicado en la Ecuación 6.115 y una función dominio definida en la Ecuación 6.116. De los atributos del conjunto A_{r_C}, el atributo agencia *identificará, mediante su nombre, a la agencia de interpretación a que se refiere cada tupla y el atributo* idioma *el idioma de interpretación a que se refiere ésta. En la función de dominio de r_C, el dominio D_{ags} se corresponde con el dominio de los nombres de las agencias de interpretación con las que cuenta el centro de negocios en cartera. Por simplicidad, dicho dominio está definido como se indica en la Ecuación 6.117. El cuerpo de r_C está definido según se indica en la Tabla 6.16.*

$$A_{r_C} = \{agencia, idioma\} \tag{6.115}$$

$$dom_{r_C}(a) = \begin{cases} D_{ags} & si\ a = agencia \\ D_{lang} & si\ a = idioma \end{cases} \qquad \forall a \in A_{r_C} \tag{6.116}$$

$$D_{ags} = \{Abecedarium, Babel, Celeris, Deletraria, Europa\} \tag{6.117}$$

La segunda MRCD, que representa los requisitos de interpretación requeridos por una de las sesiones paralelas de que se compone un determinado evento, será denominada r_D. Cada una de las tuplas de r_D representa el requerimiento de un servicio de interpretación para un determinado idioma. La cardinalidad asociada a éstas servirá para indicar el número de intérpretes y nivel de servicio requerido, empleando para ello la misma semántica que se ha usado en el caso de r_C. El grado de certidumbre de las tuplas de r_D indica el nivel de confirmación de los datos que representan cada uno de los requisitos. La semántica asociada a estos grados de certidumbre se corresponde con la indicada en la Tabla 5.7. Esta MRCD posee una cabecera formada por un conjunto de atributos cuyo único miembro es el atributo idioma y una función dominio que asocia éste con el dominio D_{lang} (definido en el Ejemplo 5.15). Finalmente, el cuerpo de r_D está definido según se indica en la Tabla 6.17.

Una vez definidas las MRCDs dividendo y divisor, obtendremos la información requerida aplicando sobre ellas el operador de division. La MRCD resultante de esta operación, $(r_C \div r_D)$, posee una cabecera compuesta por un conjunto de atributos cuyo único miembro es el atributo agencia y una función dominio que

U_{r_C}	T_{r_C}	t_{r_C}	
		Agencia	Idioma
1.0	$\{1.0/12, 0.75/31, 0.5/32\}_c$	Abecedarium	Inglés
1.0	$\{1.0/13, 0.75/19\}_c$	Abecedarium	Alemán
0.5	$\{1.0/2, 0.75/4, 0.5/10\}_c$	Babel	Inglés
1.0	$\{1.0/1, 0.75/6\}_c$	Babel	Alemán
1.0	$\{1.0/20\}_c$	Celeris	Inglés
1.0	$\{1.0/6, 0.75/12, 0.5/20\}_c$	Deletraria	Inglés
1.0	$\{1.0/4, 0.75/10, 0.5/25\}_c$	Deletraria	Alemán
0.5	$\{1.0/2, 0.5/6\}_c$	Deletraria	Italiano
0.5	$\{1.0/4, 0.75/5, 0.5/7\}_c$	Europa	Inglés
0.5	$\{1.0/8\}_c$	Europa	Alemán

Tabla 6.16: Cuerpo de r_C

U_{r_D}	T_{r_D}	t_{r_D}
		Idioma
1.0	$\{1.0/4, 0.75/6, 0.5/8\}_c$	Inglés
1.0	$\{1.0/1, 0.75/2, 0.5/4\}_c$	Alemán

Tabla 6.17: Cuerpo de r_D

$U_{(r_C \div r_D)}$	$T_{(r_C \div r_D)}$	$t_{(r_C \div r_D)}$
		Agencia
1.0	$\{1.0/3, 0.75/4\}_c$	Abecedarium
0.5	$\{0.5/1\}_c$	Babel
1.0	$\{1.0/1, 0.75/2\}_c$	Deletraria

Tabla 6.18: Cuerpo de $(r_C \div r_D)$

asocia éste con el dominio D_{ags}. En lo que respecta al cuerpo de $(r_C \div r_D)$, éste está definido según se indica en la Tabla 6.18.

En el cuerpo de $(r_C \div r_D)$ se puede observar que las agencias de interpretación que satisfacen los criterios establecidos en r_D son Abecedarium, Babel y Deletraria. La cardinalidad asociada en la MRCD resultante a la tuplas indica el número de sesiones paralelas de las características descritas en r_D de que cada agencia podría hacerse cargo.

*Concretamente, se indica que la agencia Abecedarium podría encargarse de los servicios de interpretación de tres sesiones paralelas. El grado asociado a esta cardinalidad (*1.0*) indica que la agencia podrá satisfacer completamente los requisitos establecidos para estas tres sesiones. Obsérvese que ésta cardinalidad difusa indica que la agencia Abecedarium podría hacerse cargo de cuatro sesiones paralelas, pero en esta ocasión con un grado asociado de* 0.75*. El anterior grado indica el nivel de flexibilidad con el que se han aplicado las restricciones, en lo que respecta al nivel de servicio de los intérpretes requeridos, impuestas en r_D. En concreto, este valor indica que, si el nivel de servicio máximo de los intérpretes requeridos se reduce a* 0.75 *(que se corresponde con interpretación consecutiva), se podrán satisfacer dichas condiciones en cuatro ocasiones.*

Obsérvese que la cardinalidad asociada a cada una de las tuplas de la MRCD resultante se obtiene como el mínimo de las cardinalidades difusas que indican el número de ocasiones que la agencia puede satisfacer los requisitos, en términos de intérpretes, para cada uno de los idiomas indicados en r_D. Por ejemplo, la agencia Abecedarium tiene capacidad para satisfacer los requisitos de intérpretes de inglés para $\{1.0/3, 0.75/5\}_c$ sesiones paralelas. En lo que se refiere a los intérpretes de alemán requeridos, la agencia puede satisfacer los requisitos de $\{1.0/4\}_c$ sesiones paralelas. El mínimo de ambas cardinalidades difusas resulta en $\{1.0/3, 0.75/4\}_c$, que es la cardinalidad asociada a la tupla de Abecedarium en la Tabla 6.18.

La cardinalidad asociada a la tupla que representa a la agencia Babel en la MRCD resultante ilustra de nuevo las capacidades del operador de división para flexibilizar los requisitos impuestos por la MRCD divisor. Ésta agencia tiene capacidad para satisfacer completamente los requisitos de intérpretes de alemán de una sesión paralela. En cambio, en lo que se refiere a los intérpretes de inglés, la agencia no tiene recursos para satisfacer completamente los requisitos de ninguna sesión paralela. Si estos requisitos se flexibilizan, concretamente si se reduce el nivel de servicio máximo requerido para los intérpretes de inglés a acompañamiento *(grado* 0.5 *según la Tabla 5.9), la agencia podría satisfacerlos en una ocasión. El mínimo de ambas cardinalidades difusas se corresponde con $\{0.5/1\}_c$, cardinalidad que indica la situación descrita anteriormente.*

En lo que respecta a la agencia Deletraria, su cardinalidad asociada indica el número de sesiones paralelas que puede atender según la flexibilidad que se aplique en el requisito del nivel de servicio de los intérpretes. Obsérvese que la MRCD r_C indica que la agencia tiene personal habilitado para realizar interpretación en otro idioma distinto al inglés y el alemán. Dado que éste está fuera de los requisitos marcados por r_D, la tupla que lo representa es ignorada por el operador de división y su existencia no influye en absoluto en el contenido del cuerpo de la MRCD resultante.

La agencia Celeris queda excluida del subconjunto que satisface las condiciones establecidas en r_D ya que, a pesar de que ésta satisface los requisitos establecidos en lo que se refiere a los intérpretes de inglés, no tiene alguna capacidad para realizar interpretación de alemán.

Por su parte, la agencia Europa posee recursos para realizar interpretación del par de idiomas requeridos en r_D. No obstante, y a pesar de que la agencia es capaz de satisfacer los requisitos de interpretación de alemán, estos recursos no son suficientes para satisfacer el número de intérpretes de inglés requeridos (independientemente del grado de flexibilidad que se aplique al nivel de servicio que éstos han de satisfacer).

En lo que respecta a los grados de certidumbre de las tuplas de $(r_C \div r_D)$, estos indican el grado de certidumbre máximo que se puede asociar a los datos presentes en la MRCD resultante en función del grado de certidumbre que está asociado a sus operandos. El grado de certidumbre asociado a la agencia Abecedarium indica que los datos sobre ella están confirmados. Este grado proviene de la agregación de los datos sobre sus intérpretes indicados en r_C y los datos que indican los requisitos en r_D (ambos grupos de datos con un nivel de certidumbre que indica su confirmación). En el caso de la agencia Babel, su grado de certidumbre es menor ya que no todos los datos sobre sus intérpretes en r_C indican confirmación (concretamente los datos relativos a sus intérpretes de inglés no están confirmados). La agencia Deletraria tiene un grado de certidumbre asignado que indica la confirmación de sus datos, a pesar de que en r_C no todos ellos tienen asignado un grado de certidumbre máximo. No obstante, la única tupla correspondiente a esta agencia que no tiene un grado de 1.0 *es la que indica la disponibilidad de ésta en lo que respecta a intérpretes de italiano, tupla que, como se comentó anteriormente, es ignorada por el operador de división ya que en r_D no se expresan requisitos para éste idioma.*

Obsérvese que, debido a que los grados de certidumbre de las tuplas de r_D son máximos, éstos no han evidenciado su influencia en la determinación de los grados de certidumbre de la MRCD resultante. Si éstos hubiesen sido menores, habría quedado patente el impacto que éstos ejercen sobre los grados ed certidumbre de la MRCD resultante de la división. Para ilustrar este hecho, supongamos que la MRCD r_D es sustituida en el actual ejemplo por una nueva MRCD denominada $r_{D'}$ con una cabecera similar a la anterior y un cuerpo definido según se indica en la Tabla 6.19. Obsérvese que el único cambio con respecto a r_D es el grado de certidumbre asignado a la tupla que establece los requisitos sobre el número de intérpretes de alemán.

Con esta nueva MRCD divisor, la MRCD resultante, $(r_C \div r_{D'})$, tendrá una cabecera similar a la indicada anteriormente para $(r_C \div r_D)$ y un cuerpo de la forma que se indica en la Tabla 6.20. Obsérvese que el único cambio con respecto al cuerpo de $(r_C \div r_D)$ se da en los grados de certidumbre asignados a las tuplas. En el caso de $(r_C \div r_{D'})$ todas las tuplas tienen un grado de certidum-

$U_{r_{D'}}$	$T_{r_{D'}}$	$t_{r_{D'}}$
		Idioma
1.0	$\{1.0/4, 0.75/6, 0.5/8\}_c$	Inglés
0.5	$\{1.0/1, 0.75/2, 0.5/4\}_c$	Alemán

Tabla 6.19: Cuerpo de $r_{D'}$

$U_{(r_C \div r_{D'})}$	$T_{(r_C \div r_{D'})}$	$t_{(r_C \div r_{D'})}$
		Agencia
0.5	$\{1.0/3, 0.75/4\}_c$	Abecedarium
0.5	$\{0.5/1\}_c$	Babel
0.5	$\{1.0/1, 0.75/2\}_c$	Deletraria

Tabla 6.20: Cuerpo de $(r_C \div r_{D'})$

bre asociado de 0.5 *y, concretamente, las tuplas que representan a la agencias Abecedarium y Deletraria ven disminuido su grado de certidumbre con respecto al que poseen en* r_C*. Esta disminución indica las condiciones de certidumbre en que se han emparejado las tuplas de* r_C *y* $r_{D'}$*. En particular, el grado de certidumbre asociado a la segunda tupla de* $r_{D'}$ *(según el orden en que se muestran en la Tabla 6.20) limita el grado de certidumbre de las tuplas de* $(r_C \div r_{D'})$*.*

6.5. Operadores de transformación de dominios

Esta sección está dedicada a los operadores que permiten la creación de nuevas MRCD, a partir de MRCD previas, mediante la conversión del dominio complejo de un determinado atributo en un dominio de otro tipo. Los dominios complejos, definidos anteriormente, se caracterizan por la jerarquía definida entre sus dominios descendientes. Los operadores de transformación de dominios permitirán añadir o eliminar niveles en dicha jerarquía.

La adición de niveles a la jerarquía de dominios se practicará haciendo uso de los operadores que denominaremos de *agregación*. Este tipo de operadores permitirán crear nuevos valores complejos a partir de la composición de valores presentes en las tuplas de la MRCD original. Esto se traducirá en la adición de un nivel superior a la jerarquía de dominios.

La reducción de niveles en la jerarquía se realizará a través de los operadores que denominaremos de *desagregación*. Este tipo de operadores permitirán eliminar el nivel superior de la jerarquía de dominios asociados a un atributo, lo que implicará la descomposición de valores complejos del dominio original en sus componentes.

A continuación definiremos cada uno de los operadores de desagregación y agregación contemplados. Se seguirá precisamente ese orden, en primer lugar los operadores de desagregación y en segundo lugar los operadores de agregación, para permitir una mejor comprensión a lector de los factores que afectan a los operadores de agregación.

6.5.1. Operadores de desagregación

Como se ha mencionado anteriormente, los operadores de desagregación permiten eliminar el nivel superior de la jerarquía de dominios subyacentes del dominio asociado a un determinado atributo. De esta forma, se obtiene una nueva MRCD derivada de la original que contiene los componentes de un determinado dominio complejo difuso de forma desagregada. Existe un operador de desagregación por cada tipo de dominio complejo, excepción hecha de los dominios básicos ya que estos contienen exclusivamente valores atómicos crisp. A continuación detallaremos el funcionamiento de cada uno de ellos.

6.5.1.1. Desagregación de atributos con dominio incierto

En primer lugar, definiremos el operador para los atributos de dominios inciertos. Este operador transformará el dominio de un atributo con dominio incierto de tal forma que se separarán los componentes de los valores inciertos asociados a este atributo para cada tupla.

El dominio resultante de la transformación será el dominio subyacente del dominio incierto original. Los valores originales, compuestos por un grado de certidumbre y un valor del dominio subyacente, se descompondrán de tal forma que la certidumbre de cada valor será agregada con la certidumbre asociada a su tupla correspondiente, y el valor del dominio subyacente permanecerá como valor del atributo participante en la transformación.

El proceso de desagregación descrito anteriormente se define formalmente como se indica a continuación.

Definición 6.43 (Desagregación de atributos con dominio incierto)**.** *Sea m una MRCD. Sea u un atributo de m, $u \in A_m$, cuyo dominio asociado $dom_m(u)$ es de tipo incierto, $dom_m(u) \in \mathbb{U}$. La desagregación en m del atributo u con dominio incierto, notada como $\psi_u(m)$, es una MRCD de la forma indicada en la Ecuación 6.118.*

$$\psi_u(m) = (\mathcal{H}_{\psi_u(m)}, \mathcal{B}_{\psi_u(m)}) \tag{6.118}$$

La cabecera y cuerpo de $\psi_u(m)$ están definidos como se indica, respectivamente, en las Ecuaciones 6.119 y 6.120.

$$\mathcal{H}_{\psi_u(m)} = (A_m, dom_{\psi_u(m)}) \tag{6.119}$$

$$\mathcal{B}_{\psi_u(m)} = (T_{\psi_u(m)}, U_{\psi_u(m)}) \tag{6.120}$$

Finalmente, los componentes de cabecera y cuerpo se definen como se indica a continuación:

- *La función de dominio de la MRCD resultante, $dom_{\psi_u(m)}$, se define de la forma que se indica en la Ecuación 6.121.*

$$dom_{\psi_u(m)}(a) = \begin{cases} dom_m(a) & si\ a \neq u \\ sub(dom_m(u)) & si\ a = u \end{cases} \quad \forall a \in A_m \tag{6.121}$$

- *El multiconjunto difuso de tuplas de $\psi_u(m)$ está definido según la función característica que se indica en la Ecuación 6.122. En la citada ecuación, $t_m^{|A_m-\{u\}}$ representa la restricción del dominio de t_m, que originalmente es A_m, a su subconjunto $(A_m - \{u\})$. En general, la restricción de una función $f : J \mapsto K$ a un conjunto L tal que $L \subset J$, notada a partir de este momento como $f^{|L}$, es una función de la forma $f^{|L} : L \mapsto K$ y se define según se indica en la Ecuación 6.123.*

$$\omega_{T_{\psi_u(m)}}(t) = \begin{cases} \omega_{T_m}(t_m) & si\ \exists \alpha \in (0,1], \exists t_m \in T_m | \\ & \quad (t_m^{|A_m-\{u\}} = t^{|A_m-\{u\}} \ \wedge\ t_m(u) = \langle \alpha, t(u) \rangle) \\ \{1.0/0\}_c & en\ otro\ caso \end{cases}$$

$$\forall t \in \mathbb{T}_{\psi_u(m)} \quad (6.122)$$

$$f^{|L}(l) = f(l)\ \forall l \in L, f : J \mapsto K, L \subset J \quad (6.123)$$

- *La función de certidumbre de la relación resultante, $\psi_u(m)$, queda definida de la forma indicada en la Ecuación 6.124. En la citada ecuación, el símbolo $\otimes$ representa una t-norma.*

$$U_{\psi_u(m)}(t) = U_m(t_m) \otimes\ \alpha$$

$$\forall t \in T_{\psi_u(m)} | (\exists \alpha \in (0,1] \ \wedge\ \exists t_m \in T_m | \\ (sig(t_m) = sig(t) \ \wedge\ t_m(u) = \langle \alpha, t(u) \rangle)) \quad (6.124)$$

Nótese que en la Ecuación 6.124, la condición establecida para que la función de certidumbre $U_{\psi_u(m)}$ esté definida se satisface para cualquier tupla $t \in T_{\psi_u(m)}$ debido a la definición de $T_{\psi_u(m)}$, indicada anteriormente en la Ecuación 6.122. Esta condición se incluye solamente al efecto de determinar la tupla t_m de la MRCD m que se corresponde con una determinada tupla t de la MRCD $\psi_u(m)$, así como el grado de certidumbre α asociado al valor incierto del atributo u en t_m. A pesar de que existe una única tupla $t_m \in T_m$ que se corresponde con una tupla $t \in T_{\psi_u(m)}$ (hecho garantizado ya que m se ha de encontrar en FNPE), se ha de recurrir al uso de la citada condición, ya que no se encuentra en t toda la información necesaria para reconstruir la tupla t_m que le corresponde. Concretamente, no se puede obtener de t el grado de certidumbre α asociado al valor del atributo u en t_m ya que este se encuentra agregado con el grado de certidumbre asociado a la totalidad de atributos de t.

Ejemplo 6.10. *Supongamos que deseamos desagregar el atributo* catering *de la relación r introducida en el Ejemplo 5.17. La MRCD resultante de esta operación, $\psi_{catering}(r)$, conservará el mismo conjunto de atributos de r. Su función*

dominio estará definida como se indica en la Ecuación 6.125 Obsérvese que, según se indicó en el Ejemplo 5.9, D_{bool} es el dominio subyacente de D_{conf}.

$$dom_{\psi_{catering}(r)}(a) = \begin{cases} D_{celeb} & si\ a = celebración \\ D_{asist} & si\ a = asistentes \\ D_{bool} & si\ a = catering \\ D_{inter} & si\ a = intérpretes \end{cases} \qquad \forall a \in A_{\psi_{catering}(r)} \quad (6.125)$$

El cuerpo de $\psi_{catering}(r)$ está definido según se indica en la Tabla 6.21. Se ha de notar que para la agregación de los grados de certidumbre de cada tupla original y del valor del atributo desagregado para ésta se ha empleado la función mínimo *como t-norma.*

En dicha Tabla se puede apreciar como dichos grados de certidumbre son agregados y que el resultando de la agregación conforma el nuevo valor de certidumbre para las tuplas resultantes. Particularmente, esta agregación se aprecia para las tuplas tercera y cuarta (siguiendo el orden en que éstas se muestran en la Tabla 6.21) ya que en el resto el proceso de agregación produce los mismos valores que en la MRCD original.

6.5.1.2. Desagregación de atributos con dominio impreciso

Tratado anteriormente el operador de desagregación de atributos con dominio incierto, definiremos, en segundo lugar, el operador para la desagregación de atributos con dominio de tipo impreciso.

Recordemos que los valores imprecisos son colecciones disyuntivas de elementos modeladas como distribuciones de posibilidad. Cada atributo con dominio de tipo impreciso tiene asignada, como valor para una tupla determinada, una de estas colecciones con semántica disyuntiva. Cada elemento de la colección representa un posible valor para el citado atributo. El operador de desagregación para este tipo de dominios descompondrá estos valores de tal forma que, dada una tupla, la relación resultante contenga tantas tuplas como valores posibles haya para el atributo que participa en el proceso de agregación. Cada una de dichas tuplas tendrá asociado el mismo valor que la tupla original para los atributos que no participan en el proceso de agregación. En lo que se refiere al atributo que se desagrega, las tuplas resultantes tendrán como valor uno de los posibles valores incluidos en valor impreciso original del citado atributo. Obviamente, el dominio asociado al atributo desagregado será el dominio subyacente del original.

Cada tupla resultante verá variada su cardinalidad en función del grado de posibilidad correspondiente al valor que tiene asignado para el atributo desagregado. De esta forma, las tuplas resultantes tendrán una cardinalidad igual a la de la tupla original pero limitada a dicho grado de posibilidad.

Nótese que, según el planteamiento anterior, cada elemento de las distribuciones de posibilidad del atributo con dominio impreciso que participa en la desagregación genera una nueva tupla en la MRCD resultante. Debido a ello, en el presente trabajo se asume que los valores imprecisos empleados son distribuciones de posibilidad discretas. En caso contrario, estas distribuciones habrán de ser discretizadas a fin de que la MRCD resultante contenga un número finito de tuplas.

$U_{\psi_{\text{catering}}(r)}$	$T_{\psi_{\text{catering}}(r)}$	$t_{\psi_{\text{catering}}(r)}$						
		celebración			asistentes	catering	intérpretes	
		fecha	inicio	duración				
1.0	$\{1.0/2, 0.6/5, 0.3/6\}_c$	4-6-2010	9:00	5	$entre(20, 25)$	sí	$\{1.0/1, 0.75/3, 0.5/8\}_c$ $\{1.0/1, 0.75/2, 0.5/4\}_c$	Inglés Alemán
0.5	$\{0.6/3\}_c$	14-10-2010	17:00	3	$aprox(40)$	no	$\{0.5/10\}_c$	Inglés
0.5	$\{1.0/2\}_c$	21-11-2010	15:00	6	20	sí	$\{1.0/3, 0.5/6\}_c$ $\{1.0/1, 0.75/5\}_c$ $\{0.5/4\}_c$	Inglés Francés Italiano
0.5	$\{0.3/4\}_c$	9-12-2010	17:00	4	$hasta(5)$	no	$\emptyset_{inter}$	

Tabla 6.21: Cuerpo de $\psi_{\text{catering}}(r)$

El proceso de desagregación descrito anteriormente queda definido, de una manera formal, como se indica a continuación.

Definición 6.44 (Desagregación de atributos con dominio impreciso)**.** *Sea m una MRCD. Sea i un atributo de m, $i \in A_m$, cuyo dominio asociado $dom_m(i)$ es de tipo impreciso, $dom_m(i) \in \mathbb{I}$. La desagregación en m del atributo i con dominio impreciso, notada como $\eth_i(m)$, es una MRCD definida tal como se indica en la Ecuación 6.126.*

$$\eth_i(m) = (\mathcal{H}_{\eth_i(m)}, \mathcal{B}_{\eth_i(m)}) \tag{6.126}$$

En la anterior ecuación, $\mathcal{H}_{\eth_i(m)}$ está definida como se indica en la Ecuación 6.127 y $\mathcal{B}_{\eth_i(m)}$ está definido según se indica en la Ecuación 6.128.

$$\mathcal{H}_{\eth_i(m)} = (A_m, dom_{\eth_i(m)}) \tag{6.127}$$

$$\mathcal{B}_{\eth_i(m)} = (T_{\eth_i(m)}, U_{\eth_i(m)}) \tag{6.128}$$

Cada componente de la cabecera y cuerpo de $\eth_i(m)$ está definido según lo siguiente:

- *La función de dominio de la MRCD resultante, $dom_{\eth_i(m)}$ se define de la forma que se indica en la Ecuación 6.129.*

$$dom_{\eth_i(m)}(a) = \begin{cases} dom_m(a) & si\ a \neq i \\ sub(dom_m(i)) & si\ a = i \end{cases} \quad \forall a \in A_m \tag{6.129}$$

- *El multiconjunto difuso de tuplas de $\eth_i(m)$ está definido según la función característica que se indica en la Ecuación 6.130. En dicha ecuación, ord es una función definida según se indicó en la Ecuación 6.35.*

$$\omega_{T_{\eth_i(m)}}(t) = \\ = \begin{cases} \omega_{T_m}(t_m) \boxtimes ord(\alpha) & si\ \exists \alpha \in (0,1], \exists t_m \in T_m | \\ & \quad (t_m^{|A_m - \{i\}} = t^{|A_m - \{i\}} \ \wedge\ t_m(i)(t(i)) = \alpha) \\ \{1.0/0\}_2 & en\ otro\ caso \end{cases}$$

$$\forall t \in \mathbb{T}_{\eth_i(m)} \tag{6.130}$$

- *La función de certidumbre de la relación resultante, $\eth_i(m)$, queda definida de la forma que se indica en la Ecuación 6.131.*

$$U_{\eth_i(m)}(t) = U_m(t_m) \\ \forall t \in T_{\eth_i(m)} | (\exists t_m \in T_m | (sig(t_m) = sig^{(A_{\eth_i(m)} - \{i\})}(t))) \tag{6.131}$$

Como se ha hecho en el caso de la Definición 6.43, se ha de remarcar que la condición establecida en la Ecuación 6.131 para que la función de certidumbre $U_{\eth_i(m)}(t)$ esté definida se cumplirá para cualquier tupla $t \in T_{\eth_i(m)}$. Esto es debido a la forma en que se ha definido $T_{\eth_i(m)}$ en la Ecuación 6.130. Como en el caso anterior, esta condición se establece únicamente de manera instrumental para obtener, dada una tupla t de la MRCD $\eth_i(m)$, la tupla perteneciente a la MRCD m de la que ésta proviene. Dada una tupla $t \in T_{\eth_i(m)}$, se puede garantizar que existe una única tupla en $t_m \in T_m$ que posea las misma signatura restringida excluyendo el atributo i de t. Esto es debido a que es requisito fundamental que m se encuentre en FNPE y se ha de tener en cuenta que el atributo i no es un atributo signatura de la MRCD original. Por tanto, la existencia de más de una tupla en las condiciones descritas implicaría que m no se encuentra en FNPE.

Ejemplo 6.11. *Continuemos empleando, como en el ejemplo anterior, la MRCD r para ilustrar el proceso de desagregación descrito anteriormente. Si desagregamos su atributo* asistentes*, cuyo dominio es de tipo impreciso, el resultado será la MRCD* $\eth_{asistentes}(r)$.

En lo que respecta a la cabecera de la $\eth_{asistentes}(r)$*, el conjunto de sus atributos será el mismo que el de r y su función dominio estará definida como se indica en la Ecuación 6.132. Obsérvese que, en dicha ecuación, se hace uso del dominio* D_{nat}*, dominio subyacente del dominio* D_{asist} *según se indicó en el Ejemplo 5.11.*

$$dom_{\eth_{asistentes}(r)}(a) = \begin{cases} D_{celeb} & si\ a = celebración \\ D_{nat} & si\ a = asistentes \\ D_{conf} & si\ a = catering \\ D_{inter} & si\ a = intérpretes \end{cases} \qquad \forall a \in A_{\eth_{asistentes}(r)} \tag{6.132}$$

El cuerpo de $\eth_{asistentes}(r)$ *está definido según se indica en la Tabla 6.22. Por motivos de espacio, en dicha tabla se han omitido algunas tuplas. Concretamente la tuplas segunda y decimotercera representan a las tuplas omitidas. Como se puede observar, las tuplas omitidas tienen el mismo valor para los atributos no participantes en la desagregación. En lo que respecta al atributo* asistentes*, cada tupla tendrá un valor de la secuencia de valores naturales entre 21 y 24, para el primer grupo de tuplas omitidas, y 2 y 4, para el segundo grupo de tuplas omitidas.*

En la citada tabla, se puede observar el resultado de la agregación de los grados de posibilidad de cada elemento del soporte del valor para el atributo asistentes *en r y la cardinalidad de la tupla correspondiente. Particularmente, el efecto de esta agregación se puede observar en la cardinalidad de las tuplas que se refieren al evento a celebrar el 14-10-2010. Aunque la agregación se produce para cada una de las tuplas de* $\eth_{asistentes}(r)$*, para el citado grupo esta agregación supone un cambio en la cardinalidad de las tuplas resultantes debido a los distintos grados de posibilidad de los elementos del soporte del valor asociado al atributo* asistentes *en la tupla original. Para el resto de tuplas no se produce ningún cambio ya que el grado de posibilidad para los elementos del soporte del valor en las tuplas para el citado atributo es, en todos los casos,* 1.0.

$U_{\eth_{asistentes}(r)}$	$T_{\eth_{asistentes}(r)}$	$t_{\eth_{asistentes}(r)}$						
		celebración			asistentes	catering	intérpretes	
		fecha	inicio	duración				
1.0	$\{1.0/2, 0.6/5, 0.3/6\}_c$	4-6-2010	9:00	5	20	$\langle 1.0, \text{sí}\rangle$	$\{1.0/1, 0.75/3, 0.5/8\}_c$ $\{1.0/1, 0.75/2, 0.5/4\}_c$	Inglés Alemán
1.0	$\{1.0/2, 0.6/5, 0.3/6\}_c$	4-6-2010	9:00	5	⋮	$\langle 1.0, \text{sí}\rangle$	$\{1.0/1, 0.75/3, 0.5/8\}_c$ $\{1.0/1, 0.75/2, 0.5/4\}_c$	Inglés Alemán
1.0	$\{1.0/2, 0.6/5, 0.3/6\}_c$	4-6-2010	9:00	5	25	$\langle 1.0, \text{sí}\rangle$	$\{1.0/1, 0.75/3, 0.5/8\}_c$ $\{1.0/1, 0.75/2, 0.5/4\}_c$	Inglés Alemán
0.5	$\{0.25/3\}_c$	14-10-2010	17:00	3	37	$\langle 0.5, \text{no}\rangle$	$\{0.5/10\}_c$	Inglés
0.5	$\{0.5/3\}_c$	14-10-2010	17:00	3	38	$\langle 0.5, \text{no}\rangle$	$\{0.5/10\}_c$	Inglés
0.5	$\{0.6/3\}_c$	14-10-2010	17:00	3	39	$\langle 0.5, \text{no}\rangle$	$\{0.5/10\}_c$	Inglés
0.5	$\{0.6/3\}_c$	14-10-2010	17:00	3	40	$\langle 0.5, \text{no}\rangle$	$\{0.5/10\}_c$	Inglés
0.5	$\{0.6/3\}_c$	14-10-2010	17:00	3	41	$\langle 0.5, \text{no}\rangle$	$\{0.5/10\}_c$	Inglés
0.5	$\{0.5/3\}_c$	14-10-2010	17:00	3	42	$\langle 0.5, \text{no}\rangle$	$\{0.5/10\}_c$	Inglés
0.5	$\{0.25/3\}_c$	14-10-2010	17:00	3	43	$\langle 0.5, \text{no}\rangle$	$\{0.5/10\}_c$	Inglés
1.0	$\{1.0/2\}_c$	21-11-2010	15:00	6	20	$\langle 0.5, \text{sí}\rangle$	$\{1.0/3, 0.5/6\}_c$ $\{1.0/1, 0.75/5\}_c$ $\{0.5/4\}_c$	Inglés Francés Italiano
0.5	$\{0.3/4\}_c$	9-12-2010	17:00	4	1	$\langle 1.0, \text{no}\rangle$	$\emptyset_{inter}$	
0.5	$\{0.3/4\}_c$	9-12-2010	17:00	4	⋮	$\langle 1.0, \text{no}\rangle$	$\emptyset_{inter}$	
0.5	$\{0.3/4\}_c$	9-12-2010	17:00	5	5	$\langle 1.0, \text{no}\rangle$	$\emptyset_{inter}$	

Tabla 6.22: Cuerpo de $\eth_{asistentes}(r)$

6.5.1.3. Desagregación de atributos con dominio compuesto

Definido el operador de desagregación para atributos con dominios imprecisos, proseguiremos definiendo, en tercer lugar, el operador para desagregación de atributos con dominio compuesto.

Este operador de desagregación sustituye un atributo con dominio compuesto por el conjunto de atributos que forman dicha composición. De esta forma, cada uno de los componentes del atributo a desagregar pasará a ser un atributo en la relación resultante. Cada tupla de la relación original da lugar a una tupla en la relación resultante. Finalmente, cada tupla de la relación resultante tendrá asociado como valor para los atributos procedentes de la desagregación el valor que tenía originalmente el atributo desagregado para el componente correspondiente.

La aplicación de este operador de desagregación no supondrá cambios en las cardinalidades y grados de certidumbre, por lo que tupla original y resultante compartirán los mismos valores.

La aplicación de este operador estará restringida exclusivamente a aquellas MRCDs m y a aquellos atributos o de dominio compuesto $O = dom_m(o)$ en los que los valores de dicho dominio tengan asociado un conjunto de atributos $att(O)$ cuyos nombres no coincidan con alguno de los atributos de m que no participan en el proceso de desagregación. De otra forma, al incorporar los atributos asociados a O como nuevos atributos en la MRCD resultante se produciría una colisión de nombres de atributos. En el caso de que se dé esta situación en una MRCD, ésta se podrá solventar renombrando los atributos con nombres iguales a los asociados a O, empleando para ello el operador de renombrado introducido en la subsección 6.3.3.

Con todas las consideraciones anteriores, el procedimiento de desagregación descrito anteriormente queda definido formalmente como se indica a continuación.

Definición 6.45 (Desagregación de atributos con dominio compuesto)**.** *Sea m una MRCD. Sea o un atributo de m, $o \in A_m$, cuyo dominio asociado $dom_m(o)$ es de tipo compuesto, $dom_m(o) \in \mathbb{O}$ tal que $att(dom_m(o)) \cap (A_m - o) = \emptyset$. La desagregación en m del atributo o con dominio compuesto, notada como $\phi_o(m)$, es una MRCD de la forma que se indica en la Ecuación 6.133.*

$$\phi_o(m) = (\mathcal{H}_{\phi_o(m)}, \mathcal{B}_{\phi_o(m)}) \tag{6.133}$$

La cabecera y el cuerpo de la MRCD $\phi_o(m)$ se definen según se indica, respectivamente, en las ecuaciones 6.134 y 6.135.

$$\mathcal{H}_{\phi_o(m)} = (A_{\phi_o(m)}, dom_{\phi_o(m)}) \tag{6.134}$$

$$\mathcal{B}_{\phi_o(m)} = (T_{\phi_o(m)}, U_{\phi_o(m)}) \tag{6.135}$$

Cada uno de los componentes que forman la cabecera y el cuerpo de $\phi_o(m)$, definidas anteriormente, son de la siguiente forma:

- *El conjunto de atributos de $\phi_o(m)$, $A_{\phi_o(m)}$, se define como se indica en la Ecuación 6.136.*

$$A_{\phi_o(m)} = (A_m - \{o\}) \cup att(dom_m(o)) \tag{6.136}$$

- *La función de dominio de la MRCD resultante, $dom_{\phi_o(m)}$ se define de la forma indicada en la Ecuación 6.137.*

$$dom_{\phi_o(m)}(a) = \begin{cases} dom_m(a) & si\ a \in (A_m - \{o\}) \\ dom_{dom_m(o)}(a) & si\ a \in att(dom_m(o)) \end{cases} \qquad \forall a \in A_{\phi_o(m)} \tag{6.137}$$

- *El multiconjunto difuso de tuplas de $\phi_o(m)$ está definido según la función característica que está indicada en la Ecuación 6.138, donde $t^o_m(a)$ es una tupla definida según se indica en la Ecuación 6.139.*

$$\omega_{T_{\phi_o(m)}}(t) = \omega_{T_m}(t^o_m) \qquad \forall t \in \mathbb{T}_{\phi_o(m)} \tag{6.138}$$

$$t^o_m(a) = \begin{cases} t(a) & si\ a \neq o \\ t^{|att(dom_m(o))} & si\ a = o \end{cases}$$

$$\forall m \in \Delta, \forall o \in A_m | dom_m(o) \in \mathbb{O}, \forall a \in A_m \tag{6.139}$$

- *La función de certidumbre de la relación resultante, $\phi_o(m)$, queda definida de la forma que se indica en la Ecuación 6.140.*

$$U_{\phi_o(m)}(t) = U_m(t^o_m) \qquad \forall t \in \mathbb{T}_{\phi_o(m)} \tag{6.140}$$

En el caso de la definición anterior, y a diferencia de lo que ocurría en las definiciones de los dos operadores de desagregación anteriores, no es necesario recurrir a la restricción instrumental de la función característica del conjunto de tuplas y de certidumbre de $\phi_o(m)$. En este caso la tupla en m que se corresponde con una tupla $t \in T_{\phi_o(m)}$ se puede determinar directamente a partir de la información contenida en t. Para este cometido, se ha definido la tupla t^o_m en la Ecuación 6.139.

Ilustraremos el funcionamiento del operador de desagregación descrito anteriormente mediante el siguiente ejemplo.

Ejemplo 6.12. *Continuaremos en el presente ejemplo transformando la MRCD r definida en el Ejemplo 5.17. Supongamos que deseamos desagregar los valores de su atributo* celebración. *El resultado del anterior proceso de desagregación, $\phi_{celebración}(r)$, es una MRCD cuyo conjunto de atributos está definido como se indica en la Ecuación 6.141. La función dominio de $\phi_{celebración}(r)$ está definida según se indica en la Ecuación 6.142. En lo que respecta al cuerpo de $\phi_{celebración}(r)$, éste está definido según se indica en la Tabla 6.23.*

En dicha tabla se puede observar que el único cambio que se produce es la incorporación a las tuplas de los atributos del dominio D_{celeb}.

$$A_{\phi_{celebración}(r)} = \{fecha, inicio, duración, asistentes, catering, intérpretes\} \tag{6.141}$$

$$dom_{\phi_{celebración(r)}}(a) = \begin{cases} D_{fecha} & si\ a = fecha \\ D_{hora} & si\ a = hora \\ D_{nat} & si\ a = duración \\ D_{asist} & si\ a = asistentes \\ D_{conf} & si\ a = catering \\ D_{inter} & si\ a = intérpretes \end{cases} \qquad \forall a \in A_{\phi_{celebración(r)}} \tag{6.142}$$

6.5.1.4. Desagregación de atributos con dominio multiconjuntivo difuso

Por último, definiremos el operador de desagregación de atributos con dominio multiconjuntivo difuso. La definición de este operador se basará en la del operador de desanidación del modelo NF^2 teniendo en cuenta las particularidades del presente modelo. Particularmente, y además del tratamiento obligado de cardinalidades difusas y grados de certidumbre asociados a las tuplas participantes y las cardinalidades difusas de los elementos de los multiconjuntos a desagregar, esta adaptación deberá tener en cuenta la posible presencia de valores ausentes asociados a un atributo de dominio multiconjuntivo difuso participante en un proceso de desagregación de este tipo.

Para ello, el presente apartado abordará en primer lugar una transposición directa del operador de desanidación sustituyendo las relaciones anidadas por MRCDs. Posteriormente, se mostrarán las dificultades que se plantean para este operador de desagregación en presencia de valores ausentes. Seguidamente, dedicaremos un apartado a determinar el origen de tales dificultades. A continuación, se realizará una propuesta para superar las dificultades mencionadas y, a partir de la misma, una definición formal del operador de desagregación basada en dicha propuesta. Finalmente, ilustraremos las capacidades para el tratamiento de valores ausentes de la mencionada propuesta.

Definición del operador de desagregación de atributos con dominio multiconjuntivo difuso basada en el operador de desanidación.

La concepción operador de desagregación de atributos con dominio multiconjuntivo difuso es similar a la del operador de desanidación para relaciones NF^2. Al aplicar el operador de desanidación sobre un atributo n de una relación de tipo NF^2, se realiza una transformación en la misma equivalente a realizar la reunión de cada tupla de la relación con las tuplas de la relación anidada que tiene asociada dicha tupla como valor del atributo n.

El operador de desagregación de atributos con dominio multiconjuntivo difuso se comportará de la misma manera aunque, al tratar con multiconjuntos difusos en lugar de con relaciones anidadas, se deberá tener en cuenta el tratamiento de las cardinalidades difusas asociadas a los elementos de éstos. De esta forma, al aplicar el operador de desagregación de atributos con dominio multiconjuntivo difuso, cada tupla de la MRCD original dará lugar a un conjunto de tuplas en la MRCD resultante, una por cada elemento en el multiconjunto difuso que tiene como valor la tupla original para el atributo a desagregar. El dominio

$U_{\phi_{celebración}(r)}$	$T_{\phi_{celebración}(r)}$	$t_{\phi_{celebración}(r)}$						
		fecha	inicio	duración	asistentes	catering	intérpretes	
1.0	$\{1.0/2, 0.6/5, 0.3/6\}_c$	4-6-2010	9:00	5	$entre(20, 25)$	$\langle 1.0, \text{sí}\rangle$	$\{1.0/1, 0.75/3, 0.5/8\}_c$ $\{1.0/1, 0.75/2, 0.5/4\}_c$	Inglés Alemán
0.5	$\{0.6/3\}_c$	14-10-2010	17:00	3	$aprox(40)$	$\langle 0.5, \text{no}\rangle$	$\{0.5/10\}_c$	Inglés
1.0	$\{1.0/2\}_c$	21-11-2010	15:00	6	20	$\langle 0.5, \text{sí}\rangle$	$\{1.0/3, 0.5/6\}_c$ $\{1.0/1, 0.75/5\}_c$ $\{0.5/4\}_c$	Inglés Francés Italiano
0.5	$\{0.3/4\}_c$	9-12-2010	17:00	4	$hasta(5)$	$\langle 1.0, \text{no}\rangle$	$\emptyset_{inter}$	

Tabla 6.23: Cuerpo de $\phi_{celebración}(r)$

asociado a dicho atributo en la MRCD resultante será el dominio subyacente del dominio del citado atributo en la MRCD original.

Las tuplas resultantes tendrán como valor para los atributos los mismos que tenía la tupla original a excepción del atributo que se ha desagregado. Para este atributo, cada tupla tendrá asociado como valor uno de los elementos del multiconjunto difuso asociado a la tupla correspondiente en la MRCD original.

Según lo anterior, en la MRCD resultante de la desagregación de un atributo de dominio multiconjuntivo difuso existirá una tupla por cada tupla t en la MRCD original y por cada elemento perteneciente al multiconjunto difuso asociado a t para el atributo desagregado. Por tanto, la cardinalidad de estas tuplas ha ser una agregación de la cardinalidad de los componentes de que ésta deriva. Debido a la relación de inclusión de los citados componentes (la tupla contiene al multiconjunto difuso desagregado), parece lógico que esta agregación se realice empleando el producto de las cardinalidades difusas de los citados componentes. Esta forma de agregar las cardinalidades permitirá conservar la cardinalidad total de cada uno de los elementos de los multiconjuntos difusos desagregados, estando ésta compuesta tanto por el número de veces que dicho elemento participaba en el citado multiconjunto como por el número de veces que la tupla participa en la relación.

Como veremos más adelante, el proceso de reunión de cada tupla con los elementos contenidos en el multiconjunto difuso asociado al valor para ésta del atributo desagregado será la fuente de las dificultades para la aplicación de este operador definido como la transposición directa del operador de desanidación de NF^2. Para remarcar este hecho, denominaremos al operador de desagregación procedente de tal transposición *desagregación débil de atributos con dominio multiconjuntivo difuso*. Dicho operador se define formalmente como se indica a continuación.

Definición 6.46 (Desagregación débil de atributos con dominio multiconjuntivo difuso)**.** *Sea m una MRCD. Sea n un atributo de m, $n \in A_m$, cuyo dominio asociado $dom_m(n)$ es de tipo multiconjuntivo difuso, $dom_m(n) \in \mathbb{M}$. La desagregación débil en m del atributo n con dominio multiconjuntivo difuso, notada como $\mu_n^*(m)$* [1]*, es una MRCD de la forma que muestra la Ecuación 6.143.*

$$\mu_n^*(m) = (\mathcal{H}_{\mu_n^*(m)}, \mathcal{B}_{\mu_n^*(m)}) \tag{6.143}$$

La cabecera y el cuerpo de $\mu_n^*(m)$ están definidos como se indica, respectivamente, en las ecuaciones 6.144 y 6.145.

$$\mathcal{H}_{\mu_n^*(m)} = (A_m, dom_{\mu_n^*(m)}) \tag{6.144}$$

$$\mathcal{B}_{\mu_n^*(m)} = (T_{\mu_n^*(m)}, U_{\mu_n^*(m)}) \tag{6.145}$$

Los componentes de la cabecera y cuerpo indicados en las ecuaciones anteriores están definidos como se indica a continuación:

[1] A pesar de que el símbolo μ se emplea para notar la función de pertenencia de los conjuntos difusos, emplearemos el mismo para notar el operador de desagregación de atributos con dominio multiconjuntivo difuso por coherencia con la notación empleada para el operador de desagregación de relaciones NF^2, del cual es transposición directa.

- *La función de dominio de la MRCD resultante, $dom_{\mu_n^*(m)}$, se define de la siguiente forma indicada en la Ecuación 6.146.*

$$dom_{\mu_n^*(m)}(a) = \begin{cases} dom_m(a) & si\ a \neq n \\ sub(dom_m(n)) & si\ a = n \end{cases} \qquad \forall a \in A_m \tag{6.146}$$

- *El multiconjunto difuso de tuplas de $\mu_n^*(m)$ está definido según la función característica que se indica en la Ecuación 6.147.*

$$\omega_{T_{\mu_n^*(m)}}(t) = \begin{cases} \omega_{T_m(t_m)} \boxtimes \omega_{t_m(n)}(t(n)) & si\ \exists t_m \in T_m | (t(n) \in t_m(n) \wedge \\ & \quad t_m^{|A_m - \{n\}} = t^{|A_m - \{n\}}) \\ \{1.0/0\}_2 & en\ otro\ caso \end{cases}$$

$$\forall t \in \mathbb{T}_{\mu_n^*(m)} \tag{6.147}$$

- *La función de certidumbre de la relación resultante, $\mu_n^*(m)$, queda definida de la forma que se indica en la Ecuación 6.148.*

$$U_{\mu_n^*(m)}(t) = U_m(t_m)$$
$$\forall t \in T_{\mu_n^*(m)} | (\exists t_m \in T_m | t_m^{|A_m - \{n\}} = t^{|A_m - \{n\}}) \tag{6.148}$$

Una vez más, al igual que ocurre en el caso de los operadores de desagregación de atributos con dominios inciertos e imprecisos, se aplica una restricción instrumental a la función de certidumbre de $\mu_n^*(m)$. Igual que en casos anteriores, debido a la definición del multiconjunto de tuplas de $\mu_n^*(m)$ mostrada en la Ecuación 6.147 y a que m se ha de encontrar en FNPE, dicha condición siempre se satisface y la tupla t_m es única.

Operadores de desagregación y valores ausentes.

Como se ha mencionado anteriormente, la participación de valores ausentes en procesos de desagregación puede resultar conflictiva. Los valores ausentes de un determinado dominio complejo están compuestos en su mayoría por uno o varios valores del dominio o dominios subyacentes de éste. Éste es el caso de los valores ausentes *unk* y *ni* de los dominios de tipo impreciso así como de los valores ausentes *dne* de dominios complejos de tipo incierto, impreciso y compuesto. En estos casos, la aplicación del proceso de desagregación no supone problema alguno ya que éste proceso simplemente resultará en la descomposición de dichos valores. La dificultad se presenta cuando tratamos con el valor ausente *dne* de un dominio de tipo multiconjuntivo difuso. Este valor se corresponde con el multiconjunto difuso vacío $\emptyset_N$ (siendo N un dominio de tipo multiconjuntivo difuso) y al aplicar el proceso de desagregación no hay elementos en qué dividir este valor.

Ya, desde la proposición de los modelos NF^2, se evidenció que el comportamiento del operador de desanidación (que, como se ha mencionado anteriormente, es un operador equivalente al de desagregación de atributos con dominio

multiconjuntivo difuso) cuando se aplica a relaciones anidadas vacías no es satisfactorio.

Si se emplea el operador de desagregación débil de atributos con dominio multiconjuntivo difuso (equivalente a la citada versión clásica de operador de desanidación), al desagregar en una MRCD m un atributo n asociado a un dominio de tipo multiconjuntivo difuso N, las tuplas que tengan asociado como valor para dicho atributo el multiconjunto vacío $\emptyset_N$ no tendrán correspondencia alguna en las tuplas de la MRCD resultante. Esta falta de correspondencia se debe a que no existe en el multiconjunto vacío valor alguno que permita derivar de la tupla original alguna tupla resultante.

A continuación se muestra un ejemplo que ilustra lo descrito anteriormente.

Ejemplo 6.13. *Antes de comenzar el proceso de desagregación que ilustra el presente ejemplo será necesario hacer una consideración previa. En esta ocasión, y a diferencia de los ejemplos anteriores, no operaremos sobre la MRCD r sino sobre una MRCD derivada de ésta que denominaremos* r^3*. La MRCD* r^3*, que definiremos a continuación, mantiene el atributo* intérpretes *de r, atributo que procederemos a desagregar en el presente ejemplo. El motivo de operar con* r^3 *en lugar de con r es de índole semántica, tal y como se detalla a continuación.*

La sustitución de r por r^3 *como MRCD sobre la se operarará en el presente ejemplo, tal y como se ha mencionado anteriormente, tiene como objetivo modificar la semántica asignada a las cardinalidades de los elementos de los multiconjuntos que hacen las veces de valores para el atributo* intérpretes*. En esta nueva MRCD, los grados de posibilidad de las cardinalidades difusas de los elementos de los valores multiconjuntivos difusos del atributo* intérpretes *indicarán la confirmación por parte del cliente sobre la necesidad de un determinado número de intérpretes para cada una de las sesiones paralelas. Esta modificación es necesaria para dotar de coherencia semántica a las agregaciones entre las cardinalidades asociadas a las tuplas de la MRCD y a los elementos de los valores del atributo* intérpretes*. La correspondencia entre los grados de posibilidad de las cardinalidades y la semántica asociada al mismo será similar a la que se indicó en la Tabla 5.6 y que se corresponde con la que se ha venido empleando con anterioridad para la cardinalidad de las tuplas en r. Debido a ello, los grados de posibilidad de las cardinalidades difusas de los elementos que componen los valores multiconjuntivos asociados originalmente a las tuplas en r como valor del atributo* intérpretes *han sido adaptados en* r^3 *al grado más cercano presente en la citada tabla.*

Según la modificación descrita anteriormente, la cabecera de r^3 *será igual a la de r. En cambio, el cuerpo de* r^3 *está definido como se indica en la Tabla 6.24. Según la nueva semántica asociada con las cardinalidades de los elementos de los multiconjuntos que representan los valores del atributo* intérpretes*, por ejemplo, la información que representa la primera tupla de* r^3 *(según el orden en que se muestran en la Tabla 6.24) indica que para cada una de las reuniones paralelas a celebrar en el evento del día 4–6–2010 se requerirá al menos un intérprete de inglés y otro de alemán, es posible (pendiente de confirmación) que se requieran hasta tres intérpretes de inglés y dos de alemán y está valorando (pendiente de aprobación) que sean necesarios hasta ocho intérpretes de inglés y cuatro de alemán.*

Descrita la MRCD sobre la que se trabajará en el presente ejemplo, veamos qué ocurre si permitimos que participen en el proceso de desagregación tuplas con

U_{r^3}	T_{r^3}	t_{r^3}						
		celebración			asistentes	catering	intérpretes	
		fecha	inicio	duración				
1.0	$\{1.0/2, 0.6/5, 0.3/6\}_c$	4-6-2010	9:00	5	$entre(20, 25)$	$\langle 1.0, \text{sí}\rangle$	$\{1.0/1, 0.6/3, 0.3/8\}_c$ $\{1.0/1, 0.6/2, 0.3/4\}_c$	Inglés Alemán
0.5	$\{0.6/3\}_c$	14-10-2010	17:00	3	$aprox(40)$	$\langle 0.5, \text{no}\rangle$	$\{0.3/10\}_c$	Inglés
1.0	$\{1.0/2\}_c$	21-11-2010	15:00	6	20	$\langle 0.5, \text{sí}\rangle$	$\{1.0/3, 0.3/6\}_c$ $\{1.0/1, 0.6/5\}_c$ $\{0.3/4\}_c$	Inglés Francés Italiano
0.5	$\{0.3/4\}_c$	9-12-2010	17:00	4	$hasta(5)$	$\langle 1.0, \text{no}\rangle$	$\emptyset_{inter}$	

Tabla 6.24: Cuerpo de r^3

valor ausente para el atributo participante en dicho proceso. Para ello, desagregaremos en r^3 *el atributo* intérpretes *empleando el operador de desagregación débil de atributos con dominio multiconjuntivo difuso descrito en la Definición 6.46. La desagregación en dicha MRCD del atributo* intérpretes *es una MRCD notada como* $\mu^*_{intérpretes}(r^3)$*. En lo que respecta a la cabecera de esta MRCD, ésta posee como conjunto de atributos el mismo que* r^3 *pero su función dominio, definida en la Ecuación 6.149, varía con respecto a la MRCD original. En la definición de dicha función de dominio se hace uso del dominio* D_{lang} *definido previamente en el Ejemplo 5.15. Recordemos que* D_{lang} *es el dominio subyacente de* D_{inter}*, tal y como se indicó en el Ejemplo 5.17.*

$$dom_{\mu^*_{intérpretes}(r^3)}(a) = \begin{cases} dom_{r^3}(a) & si\ a \neq intérpretes \\ D_{lang} & si\ a = intérpretes \end{cases} \qquad \forall a \in A_{\mu^*_{intérpretes}(r^3)} \tag{6.149}$$

En lo que se refiere al cuerpo de $\mu_{intérpretes}(r^3)$*, éste está definido como se indica en la Tabla 6.25.*

En la citada tabla se puede observar cómo cada tupla de r^3 *(salvo una excepción que se comentará a continuación) da lugar en* $\mu^*_{intérpretes}(r^3)$ *a una serie de tuplas, tantas como elementos tuviese el valor del atributo* intérpretes *en la MRCD original. Cada una de las tuplas resultantes representa la petición de una serie de intérpretes para un determinado evento. Las cardinalidades de estas tuplas indican hasta cuantos intérpretes de un determinado idioma serán necesarios para un determinado evento. Los grados de posibilidad en estas cardinalidades difusas pueden ser interpretados según la semántica establecida en la Tabla 5.6. Según esto, por ejemplo, la primera tupla de* $\mu^*_{intérpretes}(r^3)$ *(según el orden en que éstas muestran en la Tabla 6.25) indica que para el evento a celebrar el 4–6–2010 (teniendo en cuenta el total de sesiones paralelas asociadas a este evento, siendo dicha cantidad una cardinalidad difusa) se ha confirmado que se requerirán hasta dos intérpretes de inglés, es posible (dependiendo de confirmación) que se requieran al menos 15 intérpretes de este idioma, y se está valorando (pendiente de aprobación) que sean necesarios hasta 48. Todo ello dependerá del número final de reuniones paralelas celebradas y del número de intérpretes fijado finalmente para cada una de éstas.*

Como se señaló anteriormente, el hecho observado anteriormente de que por cada tupla en r^3 *se derivan una o varias tuplas en* $\mu^*_{intérpretes}(r^3)$ *se cumple para todas las presentes en el cuerpo de dicha MRCD salvo una excepción. Esta excepción es la última tupla de* r^3 *(siempre siguiendo el orden en que se muestran las tuplas en la Tabla 6.24). Esta tupla no produce ninguna tupla resultante en* $\mu^*_{intérpretes}(r^3)$*. Esto es debido a que el valor para el atributo desagregado de esta tupla es el valor ausente* $\emptyset_{inter}$*. En concreto, al aplicar la expresión en la Ecuación 6.147 de la Definición 6.46 para obtener el conjunto de tuplas de* $\mu^*_{intérpretes}(r^3)$*, la tupla en cuestión (que denotaremos* t_m*) no satisface la subcondición* $t(n) \in t_m(n)$ *de la citada ecuación para ninguna tupla* $t \in \mathbb{T}_{\mu^*_{intérpretes}(r^3)}$ *debido a que* $t_m(n)$ *es un conjunto vacío.*

Como se ha mostrado en el anterior ejemplo, la ausencia de una correspondencia en la relación resultante de la desagregación de aquellas tuplas que tienen como valor del atributo a el valor $\emptyset_D$ supone una pérdida de información acerca de las tuplas originales en el proceso de desagregación de un atributo con

$U_{\mu^*_{intérpretes}(r^3)}$	$T_{\mu^*_{intérpretes}(r^3)}$	$t_{\mu^*_{intérpretes}(r^3)}$					
		celebración			asistentes	catering	intérpretes
		fecha	inicio	duración			
1.0	$\{1.0/2, 0.6/15, 0.3/48\}_c$	4-6-2010	9:00	5	$entre(20, 25)$	$\langle 1.0, \text{sí}\rangle$	Inglés
1.0	$\{1.0/2, 0.6/10, 0.3/24\}_c$	4-6-2010	9:00	5	$entre(20, 25)$	$\langle 1.0, \text{sí}\rangle$	Alemán
0.5	$\{0.3/30\}_c$	14-10-2010	17:00	3	$aprox(40)$	$\langle 0.5, \text{no}\rangle$	Inglés
1.0	$\{1.0/6, 0.3/12\}_c$	21-11-2010	15:00	6	20	$\langle 0.5, \text{sí}\rangle$	Inglés
1.0	$\{1.0/2, 0.6/10\}_c$	21-11-2010	15:00	6	20	$\langle 0.5, \text{sí}\rangle$	Francés
1.0	$\{0.3/8\}_c$	21-11-2010	15:00	6	20	$\langle 0.5, \text{sí}\rangle$	Italiano

Tabla 6.25: Cuerpo de $\mu^*_{intérpretes}(r^3)$

dominio complejo de tipo multiconjuntivo difuso. Esta pérdida de información implica, como se ilustrará posteriormente en el apartado dedicado al operador complementario de agregación, que la MRCD original m no podrá ser reconstruida aplicando un operador de agregación sobre la MRCD resultante $\mu_a(m)$. Esto, en definitiva, supone que, en este caso particular, el operador de agregación no es complementario al operador de desagregación. Por tanto, a pesar de las restricciones impuestas por la FNPE a las MRCDs para que el operador de agregación y desagregación sean inversos, en el caso de relaciones que contenga el valor $\emptyset_D$ en los atributos con dominio de tipo multiconjuntivo difuso no se da esta deseable propiedad.

Desagregación de atributos con dominio multiconjuntivo difuso basada en el álgebra de MRCDs.

La citada pérdida de información en las tuplas que tengan asociado el valor ausente $\emptyset_N$ para el atributo sobre el que se desagrega (siendo N el dominio de tipo multiconjuntivo difuso asociado a dicho atributo) se produce por la propia concepción del proceso de desagregación. Para ilustrar este hecho veamos el proceso de desagregación de atributos con dominio multiconjuntivo difuso como una expresión del álgebra de MRCDs. Dicho proceso puede ser definido en base a otros operadores del álgebra de MRCDs como se muestra en la Ecuación 6.150.

$$\mu_n^*(m) = \pi_{(A_m - \{n\})}(m) \bowtie m_n \tag{6.150}$$

En la anterior ecuación, la MRCD m_n es una MRCD derivada de m de la forma que se indica a continuación.

Definición 6.47 (MRCD m_n)**.** *Sea m una MRCD. Sea n un atributo de m, $n \in A_m$, cuyo dominio asociado $dom_m(n)$ es de tipo multiconjuntivo difuso, $dom_m(n) \in \mathbb{M}$. La MRCD m_n es una MRCD derivada de m que está definida según se indica en la Ecuación 6.151.*

$$m_n = (\mathcal{H}_{m_n}, \mathcal{B}_{m_n}) \tag{6.151}$$

La cabecera y cuerpo de la m_n se definen como indican, respectivamente, las ecuaciones 6.152 y 6.153.

$$\mathcal{H}_{m_n} = (A_{m_n}, dom_{m_n}) \tag{6.152}$$

$$\mathcal{B}_{m_n} = (T_{m_n}, U_{m_n}) \tag{6.153}$$

Los componentes de las cabecera y cuerpo de las ecuaciones anteriores son de la siguiente forma:

- *El conjunto de atributos de m_n es de la forma indicada en la Ecuación 6.154.*

$$A_{m_n} = sig(m) \cup \{n\} \tag{6.154}$$

- *La función de dominio de m_n se define según se indica en la Ecuación 6.155.*

$$dom_{m_n}(a) = \begin{cases} dom_m(a) & si\ a \neq n \\ sub(dom_m(n)) & si\ a = n \end{cases} \qquad \forall a \in A_{m_n} \tag{6.155}$$

- *El multiconjunto de tuplas T_{m_n} está definido según la función característica que se indica en la Ecuación 6.156.*

$$\omega_{T_{m_n}}(t) = \begin{cases} \omega_{t_m(n)}(t(n)) & si\ \exists t_m \in T_m | (t(n) \in t_m(n) \wedge \\ & \qquad t_m^{|sig(m)} = t^{|sig(m)}) \\ \{1.0/0\}_2 & en\ otro\ caso \end{cases}$$

$$\forall t \in \mathbb{T}_{m_n} \tag{6.156}$$

- *La función de certidumbre de m_n se define según se muestra en la Ecuación 6.157.*

$$U_{m_n}(t) = 1.0 \qquad \forall t \in T_{\mu_n(m)} \tag{6.157}$$

Como se muestra en la Ecuación 6.150, la desagregación en una MRCD m de un atributo n con dominio multiconjuntivo difuso se puede ver como la reunión de dos MRCDs. La primera de ellas, $\pi_{(A_m-\{n\})}(m)$, contendrá las tuplas de m eliminando el atributo n. La segunda, m_n, es una MRCD que contiene una tupla por cada valor en el multiconjunto difuso $t(n)$ de cada tupla t de m. Según lo anterior, cada una de las tuplas de m_n representa uno de los elementos del valor multiconjuntivo difuso asociado al atributo n de una de las tuplas de m.

Las tuplas de m_n se han de reunir de manera coherente con las tuplas en $\pi_{(A_m-\{n\})}(m)$. En definitiva, cada tupla en m_n, representando a un elemento $e \in t(n)$, se ha de reunir con la tupla correspondiente a t en $\pi_{(A_m-\{n\})}(m)$, $t^{|(A_m-\{n\})}$. Con este objetivo, la tuplas de m_n incluirán los atributos signatura de m y compartirán con la tupla de m de la que proceden los valores para dichos atributos. De esta forma, se empleará la signatura de cada tupla t como información para realizar la reunión de forma coherente.

Se ha de indicar que las tuplas de m_n tendrán asignada la cardinalidad asociada al elemento $e \in t(n)$ en lugar de la cardinalidad original de la tupla original de la que derivan. De esta forma, al realizarse la reunión de m_n con $\pi_{(A_m-\{n\})}(m)$, cuyas tuplas conservan la cardinalidad de las tuplas originales, ambas cardinalidades se agregarán mediante su producto. Esto propiciará que la cardinalidad de las tuplas de la MRCD resultante sea coherente con lo dispuesto en la Definición 6.46.

El grado de la certidumbre de las tuplas en m_n se corresponde siempre con 1.0 para que éste actúe como elemento neutro. De esta forma, los grados de certidumbre asociados a las tuplas de $\pi_{(A_m-\{n\})}(m)$ no se verán afectados por el proceso de reunión, teniendo cada tupla de la MRCD resultante del proceso de desagregación asociado el mismo grado de certidumbre que el de la tupla en la MRCD original de la cual deriva. Dicho resultando es coherente con la descripción del proceso de desagregación de atributos con dominio multiconjuntivo difuso descrito en la Definición 6.46.

El funcionamiento del procedimiento de desagregación descrito anteriormente puede ilustrarse mediante el siguiente ejemplo.

Ejemplo 6.14. *Apliquemos, en esta ocasión, el proceso de desagregación propuesto en el Ejemplo 6.13 aplicando la expresión algebraica equivalente propuesta anteriormente en la Ecuación 6.150. Según ésta, la MRCD* $\mu^*_{intérpretes}(r^3)$ *se puede obtener a partir de la expresión que se muestra en la Ecuación 6.158.*

$$\mu^*_{intérpretes}(r^3) = \pi_{(A_{r^3}-\{intérpretes\})}(r^3) \bowtie r^3_{intérpretes} \tag{6.158}$$

En la anterior ecuación, $\pi_{(A_{r^3}-\{intérpretes\})}(r^3)$ *es una MRCD cuya cabecera está definida según lo indicado en la Definición 6.19 y cuyo cuerpo es de la forma que se indica en la Tabla 6.26. En lo que respecta a* $r^3_{intérpretes}$*, su cabecera está definida según se indica en la Definición 6.47 y su cuerpo según muestra la Tabla 6.27.*

Obviamente, el resultado de la aplicación de la expresión de la Ecuación 6.158 es una MRCD similar a la que se obtuvo en el Ejemplo 6.13. Como en aquel caso, la última tupla de r^3 *queda sin representación en* $\mu_{intérpretes}(r^3)$*. Nótese que, aunque dicha tupla posee representación en* $\pi_{(A_{r^3}-\{intérpretes\})}(r^3)$*, ésta no encuentra emparejamiento con ninguna tupla de* $r^3_{intérpretes}$*. Por tanto, el resultando de la reunión entre ambas MRCDs no incluye a la misma.*

Propuesta de un nuevo método de desagregación.

Como puede observarse fácilmente en el ejemplo anterior, la pérdida de información al desanidar multiconjuntos difusos vacíos se produce en el momento en que se reúnen $\pi_{(A_m-\{n\})}(m)$ y m_n. En ese proceso son descartadas aquellas tuplas de la MRCD $\pi_{(A_m-\{n\})}(m)$ que no tienen correspondencia en m_n.

Como solución al anterior problema, proponemos la sustitución del operador de reunión natural por otro que no descarte aquellas tuplas de $\pi_{(A_m-\{n\})}(m)$ sin correspondencia. El operador de *reunión natural externa*, en concreto en su variante izquierda, es un buen candidato para realizar la operación de emparejamiento sin la pérdida de tuplas sin correspondencia.

Al aplicar este operador en lugar de la reunión natural en la expresión algebraica de la Ecuación 6.150 evitaremos que las tuplas con valores ausentes para el atributo participante en un proceso de desagregación queden sin correspondencia en la MRCD resultante. El siguiente ejemplo ilustra el resultado de tal cambio.

Ejemplo 6.15. *Según la propuesta anterior, reemplazaremos el operador de reunión natural por el de reunión natural externa izquierda en la Ecuación 6.158 del anterior Ejemplo 6.14. Por tanto, la expresión algebraica resultante será la mostrada en la Ecuación 6.159.*

$$s = \pi_{(A_{r^3}-\{intérpretes\})}(r^3) =\!\bowtie r^3_{intérpretes} \tag{6.159}$$

Tanto, la MRCD $\pi_{(A_{r^3}-\{intérpretes\})}(r^3)$ *como* $r^3_{intérpretes}$ *están definidas según lo indicado en el Ejemplo 6.14. El resultado de la expresión algebraica anterior,* $\pi_{(A_{r^3}-\{intérpretes\})}(r^3) =\!\bowtie r^3_{intérpretes}$*, que denominaremos por motivos de brevedad como* s*, tiene asociada una cabecera similar a la que se indicó para* $\mu^*_{intérpretes}(r^3)$ *en el Ejemplo 6.14 ya que ambos operadores de reunión (la*

$U_{\pi_{(A_{r^3}-\{intérpretes\})}(r^3)}$	$T_{\pi_{(A_{r^3}-\{intérpretes\})}(r^3)}$	$t_{\pi_{(A_{r^3}-\{intérpretes\})}(r^3)}$				
		celebración			asistentes	catering
		fecha	inicio	duración		
1.0	$\{1.0/2, 0.6/5, 0.3/6\}_c$	4-6-2010	9:00	5	$entre(20, 25)$	$\langle 1.0, \text{sí}\rangle$
0.5	$\{0.6/3\}_c$	14-10-2010	17:00	3	$aprox(40)$	$\langle 0.5, \text{no}\rangle$
1.0	$\{1.0/2\}_c$	21-11-2010	15:00	6	20	$\langle 0.5, \text{sí}\rangle$
0.5	$\{0.3/4\}_c$	9-12-2010	17:00	4	$hasta(5)$	$\langle 1.0, \text{no}\rangle$

Tabla 6.26: Cuerpo de $\pi_{(A_{r^3}-\{intérpretes\})}(r^3)$

$U_{r^3_{intérpretes}}$	$T_{r^3_{intérpretes}}$	$t_{r^3_{intérpretes}}$				
		celebración			catering	intérpretes
		fecha	inicio	duración		
1.0	$\{1.0/1, 0.6/3, 0.3/8\}_c$	4-6-2010	9:00	5	$\langle 1.0, \text{sí}\rangle$	Inglés
1.0	$\{1.0/1, 0.6/2, 0.3/4\}_c$	4-6-2010	9:00	5	$\langle 1.0, \text{sí}\rangle$	Alemán
1.0	$\{0.3/10\}_c$	14-10-2010	17:00	3	$\langle 0.5, \text{no}\rangle$	Inglés
1.0	$\{1.0/3, 0.3/6\}_c$	21-11-2010	15:00	6	$\langle 0.5, \text{sí}\rangle$	Inglés
1.0	$\{1.0/1, 0.6/5\}_c$	21-11-2010	15:00	6	$\langle 0.5, \text{sí}\rangle$	Francés
1.0	$\{0.3/4\}_c$	21-11-2010	15:00	6	$\langle 0.5, \text{sí}\rangle$	Italiano

Tabla 6.27: Cuerpo de $r^3_{intérpretes}$

reunión natural y la reunión natural externa izquierda) producen MRCDs resultantes con cabeceras similares. La diferencia entre ambos radica exclusivamente en el cuerpo de la MRCD resultante. El cuerpo de la MRCD s es como se indica en la Tabla 6.28.

Nótese que la única diferencia entre el cuerpo de s y el de $\mu^*_{intérpretes}(r^3)$ *reside en la última tupla de s. Ésta se corresponde con la última tupla de* r^3 *y se obtiene gracias a la sustitución del operador de reunión natural por el de reunión natural externa natural izquierda.*

Definición del operador de desagregación de atributos con dominio multiconjuntivo difuso.

La anterior solución, como se ha ilustrado en el ejemplo previo, relaciona el valor ausente dne_N de un dominio multiconjunto difuso N, o lo que es lo mismo el conjunto vacío $\emptyset_N$, con el valor $dne_{sub(N)}$ del dominio subyacente de N.

Según esta propuesta, definiremos el operador de desagregación de atributos con dominio multiconjuntivo difuso como se indica a continuación.

Definición 6.48 (Desagregación de atributos con dominio multiconjuntivo difuso)**.** *Sea* m *una MRCD. Sea* n *un atributo de* m*,* $n \in A_m$*, cuyo dominio asociado* $dom_m(n)$ *es de tipo multiconjuntivo difuso,* $dom_m(n) \in \mathbb{M}$*.*

La desagregación en m *del atributo* n *con dominio multiconjuntivo difuso, notada como* $\mu_n(m)$*, es una MRCD tal y como se indicó en la Definición 6.46 exceptuando a su multiconjunto de tuplas. Éste está definido según la función característica que se indica en la Ecuación 6.160.*

$$\omega_{T_{\mu_n(m)}}(t) = \begin{cases} \omega_{T_m}(t_m) \boxtimes \omega_{t_m(n)}(t(n)) & si\ \exists t_m \in T_m | (t(n) \in t_m(n) \wedge \\ & \quad t_m^{|A_m - \{n\}} = t^{|(A_m - \{n\})}) \\ \omega_{T_m}(t_m) & si\ \exists t_m \in T_m | (t_m(n) = \emptyset_{dom_m(n)} \wedge \\ & \quad t_m^{|A_m - \{n\}} = t^{|(A_m - \{n\})}) \wedge \\ & \quad t(n) = dne_{dom_{\mu_n(m)}(n)} \\ \{1.0/0\}_2 & en\ otro\ caso \end{cases}$$

$$\forall t \in \mathbb{T}_{\mu_n(m)} \quad (6.160)$$

La anterior definición se diferencia de su versión débil en la adición de un caso más en la definición del multiconjunto de tuplas de $\mu_n(m)$. Este nuevo caso, el segundo de la Ecuación 6.160, incluye la circunstancia de que la tupla original t_m tenga como valor para el atributo n al conjunto vacío $\emptyset_{dom_m(n)}$. Este extremo en la definición original no satisfacía la condición del primer caso de la Ecuación 6.147 y por tanto la tupla t correspondiente a t_m quedaba excluida de la MRCD resultante. Con la nueva adición, una tupla de este tipo satisface el caso introducido en la nueva definición y por tanto posee representante en la MRCD resultante. El citado caso establece un vínculo entre cada tupla de la MRCD original que posea $\emptyset_N$ como valor para el atributo participante en la desagregación (siendo N el dominio asociado a dicho atributo) y una tupla en la

U_s	T_s	t_s					
		celebración			asistentes	catering	intérpretes
		fecha	inicio	duración			
1.0	$\{1.0/2, 0.6/15, 0.3/48\}_c$	4-6-2010	9:00	5	$entre(20, 25)$	$\langle 1.0, \text{sí}\rangle$	Inglés
1.0	$\{1.0/2, 0.6/10, 0.3/24\}_c$	4-6-2010	9:00	5	$entre(20, 25)$	$\langle 1.0, \text{sí}\rangle$	Alemán
0.5	$\{0.3/30\}_c$	14-10-2010	17:00	3	$aprox(40)$	$\langle 0.5, \text{no}\rangle$	Inglés
1.0	$\{1.0/6, 0.3/12\}_c$	21-11-2010	15:00	6	20	$\langle 0.5, \text{sí}\rangle$	Inglés
1.0	$\{1.0/2, 0.6/10\}_c$	21-11-2010	15:00	6	20	$\langle 0.5, \text{sí}\rangle$	Francés
1.0	$\{0.3/8\}_c$	21-11-2010	15:00	6	20	$\langle 0.5, \text{sí}\rangle$	Italiano
0.5	$\{0.3/4\}_c$	9-12-2010	17:00	4	$hasta(5)$	$\langle 1.0, \text{no}\rangle$	dne_{lang}

Tabla 6.28: Cuerpo de s

MRCD resultante que es equivalente a la anterior en los valores de los atributos excepto para el atributo participante en la desagregación, para el cuál posee asociado el valor $dne_{sub(N)}$. En este caso, tanto la cardinalidad como el grado de certidumbre de las tuplas originales será transferido de forma directa a las resultantes.

6.5.2. Operadores de agregación

La contraparte de los operadores de desagregación, vistos anteriormente, la forman los operadores de agregación. Como se indicó en la introducción, los operadores de agregación permiten añadir un nivel más a la jerarquía del dominio asociado a un atributo o conjunto de atributos, según sea el caso. De esta manera, para el caso de dominios complejos que no sean de tipo compuesto, sustituiremos el dominio de un determinado atributo por otro dominio cuyo dominio subyacente será el antiguo dominio del atributo. En el caso de dominios de tipo compuesto, participará en esta transformación un grupo de atributos que serán agrupados en un único atributo cuyo dominio tenga como conjunto de atributos a los originales, siendo los dominios de los atributos del citado conjunto los descendientes directos del nuevo dominio compuesto.

Estos cambios, sea cual sea el caso, implicarán que el nuevo dominio asociado al atributo tendrá un nivel más en su jerarquía de dominios subyacentes que el anterior. Existe un operador de agregación de cuya aplicación resultan valores de cada uno de los tipos de dominios complejos a excepción, claro está, de los dominios básicos (ya que sus valores son atómicos). De esta forma, los operadores de agregación y desagregación asociados a un tipo de dominio concreto son complementarios, permitiendo uno deshacer la transformación realizada por el contrario.

6.5.2.1. Agregación en un atributo con dominio incierto

Para comenzar, definiremos el operador de agregación en atributos con dominio incierto. Este operador de agregación modifica el dominio del atributo participante en la agregación convirtiéndolo en un dominio incierto. Este nuevo dominio estará compuesto por valores inciertos sobre el dominio original del citado atributo. Por tanto, el antiguo dominio será el dominio subyacente del nuevo.

Cada tupla en la MRCD original dará lugar a una tupla en la MRCD resultante. Los valores de los atributos no participantes en la agregación serán los mismos en cada una de las tuplas originales y en las resultantes. Los valores imprecisos del atributo participante en la agregación se crearán a partir de dos fuentes. El componente que dota de valor base al atributo impreciso se obtendrá del valor que tiene la tupla original para el atributo participante en la agregación. El grado de certidumbre asociado a dicho valor se extraerá del grado de certidumbre asociado a la tupla original.

Recuérdese que el proceso de desagregación de un atributo con dominio incierto agrega de forma conjuntiva, en el valor de certidumbre asociado a la tupla resultante, el grado de certidumbre de la tupla original y el grado de certidumbre asociado al valor del atributo desagregado. El proceso, en el caso del operador de agregación en un atributo con dominio incierto, debe ser el inverso. Del grado de certidumbre de la tupla original se extraerá, además del

grado de certidumbre para el valor del atributo participante en la agregación, el grado de certidumbre de la tupla resultante. Dada la imposibilidad de revertir la agregación de dos grados de certidumbre partiendo solamente del resultado de dicha agregación, será necesario proporcionar al operador dos funciones que denominaremos *funciones desagregadoras*. Cada una de ellas proporcionará por cada tupla original un grado de certidumbre. El valor proporcionado por la primera de ellas será el grado de certidumbre de la tupla resultante. La segunda función proporcionará como resultado el grado de certidumbre asignado al valor del atributo participante en la agregación. Estas funciones serán de tal forma que dada una determinada tupla original, la agregación conjuntiva de los grados que proporcionan sea equivalente al grado de certidumbre asociado a dicha tupla. De esta forma, se garantiza un resultado coherente.

Definiremos formalmente este operador de agregación de la forma que se indica a continuación.

Definición 6.49 (Agregación en un atributo con dominio incierto)**.** *Sea m una MRCD. Sea u un atributo de m, $u \in A_m$. Sean d y d' unas funciones de la forma $d : sig(T_m) \mapsto (0,1]$ y $d' : sig(T_m) \mapsto (0,1]$ tales que $\forall t_m \in T_m, U_m(t_m) = d(t_m) \otimes d'(t_m)$. En lo anterior, $sig(T_m)$ representa al conjunto de las signaturas de las tuplas en T_m que está definido según se indica en la Ecuación 6.161. A las anteriores funciones d y d' se las denominará* funciones desagregadoras.

$$sig(T_m) = \{sig(t_m) | t_m \in T_m\} \qquad \forall m \in \Delta \tag{6.161}$$

La agregación en m del atributo u en un atributo con dominio incierto determinada por el par de funciones desagregadoras d y d', notada como $\upsilon_{(u,d,d')}(m)$, se define de la forma que se indica en la Ecuación 6.162.

$$\upsilon_{(u,d,d')}(m) = (\mathcal{H}_{\upsilon_{(u,d,d')}(m)}, \mathcal{B}_{\upsilon_{(u,d,d')}(m)}) \tag{6.162}$$

En lo anterior, la cabecera y cuerpo de $\upsilon_{(u,d,d')}(m)$ se definen como se indica en las ecuaciones 6.163 y 6.164.

$$\mathcal{H}_{\upsilon_{(u,d,d')}(m)} = (A_m, dom_{\upsilon_{(u,d,d')}(m)}) \tag{6.163}$$

$$\mathcal{B}_{\upsilon_{(u,d,d')}(m)} = (T_{\upsilon_{(u,d,d')}(m)}, U_{\upsilon_{(u,d,d')}(m)}) \tag{6.164}$$

Los componentes de la cabecera y el cuerpo indicados anteriormente, se definen a continuación como sigue:

- *La función de dominio de la MRCD resultante, $dom_{\upsilon_{(u,d,d')}(m)}$ se define de la forma indicada en la Ecuación 6.165.*

$$dom_{\upsilon_{(u,d,d')}(m)}(a) = \begin{cases} dom_m(a) & si\ a \neq u \\ U \in \mathbb{U} | sub(U) = dom_m(u) & si\ a = u \end{cases}$$

$$\forall a \in A_m \tag{6.165}$$

- *El multiconjunto difuso de tuplas de $\upsilon_{(u,d,d')}$ está definido según la función característica que se indica la Ecuación 6.166, donde t_m^u es una tupla definida según se indica en la Ecuación 6.167*

$$\omega_{T_{\upsilon_{(u,d,d')}(m)}}(t) = \begin{cases} \omega_{T_m}(t_m^u) & si\ cert(t(u)) = d'(sig(t_m^u)) \\ \{1.0/0\}_2 & en\ caso\ contrario \end{cases}$$

$$\forall t \in \mathbb{T}_{\upsilon_{(u,d,d')}(m)} \quad (6.166)$$

$$t_m^u(a) = \begin{cases} t(a) & si\ a \neq u \\ val(t(a)) & si\ a = u \end{cases}$$

$$\forall m \in \Delta, \forall u \in A_m | dom_m(u) \in \mathbb{U}, \forall t \in \mathbb{T}_{\upsilon_{(u,d,d')}(m)}, \forall a \in A_m \quad (6.167)$$

- *La función de certidumbre de la relación resultante, $U_{\upsilon_{(u,d,d')}}$, queda definida de la forma que se muestra en la Ecuación 6.168.*

$$U_{\upsilon_{(u,d,d')}(m)}(t) = d(sig(t_m^u)) \qquad \forall t \in T_{\upsilon_{(u,d,d')}(m)} \quad (6.168)$$

En la definición anterior, se puede apreciar que la condición que impone la Ecuación 6.166 para la pertenencia de una tupla t a $T_{\upsilon_{(u,d,d')}(m)}$ asegura que a dicho multiconjunto sólo pertenece aquella tupla cuyo grado de certidumbre asociado al valor del atributo u se corresponde con lo determinado por la función desagregadora d'. En dicha ecuación se introduce la tupla t_m^u que permitirá relacionar tuplas originales y resultantes del proceso de agregación. Dada una tupla t de $\upsilon_{(u,d,d')}(m)$, t_m^u es la tupla correspondiente a t en la MRCD original m.

Ejemplo 6.16. *Supongamos que deseamos realizar una agregación en la que participe el atributo* catering *de la MRCD $\psi_{catering}(r)$, definida en el Ejemplo 6.10, con el objetivo de obtener de nuevo la MRCD r. Para realizar el proceso de agregación y obtener dicha MRCD, definiremos las funciones desagregadoras d y d' tal y como se indica en la Tabla 6.29.*

La agregación en $\psi_{catering}(r)$ del atributo catering *en un atributo con dominio incierto determinada por el par de funciones desagregadoras d y d', es una MRCD notada como $\upsilon_{(catering,d,d')}(\psi_{catering}(r))$. Ésta posee el mismo conjunto de atributos que la MRCD $\psi_{catering}(r)$ (que a su vez posee el mismo conjunto de atributos que r) y tiene una función dominio definida como se indica en la Ecuación 6.169. En dicha ecuación, se emplea el dominio D_{conf} que fue definido en el Ejemplo 5.8. Recordemos que este dominio es un dominio de tipo incierto cuyo dominio subyacente es el dominio D_{bool}. Según lo anterior, podemos decir que la función de dominio de $\upsilon_{(catering,d,d')}(\psi_{catering}(r))$ es equivalente a la función dominio de la MRCD r.*

d	d'	$sig(t_{\psi_{\text{catering}}(r)})$			
		celebración			catering
		fecha	inicio	duración	
1.0	1.0	4-6-2010	9:00	5	sí
0.5	0.5	14-10-2010	17:00	3	no
1.0	0.5	21-11-2010	15:00	6	sí
0.5	1.0	9-12-2010	17:00	4	no

Tabla 6.29: Definición de las funciones desagregadoras d y d' para la MRCD $\psi_{\text{catering}}(r)$

$$dom_{\upsilon_{(catering,d,d')}(\psi_{catering}(r))}(a) = \begin{cases} dom_{\psi_{catering}(r)}(a) & si\ a \neq catering \\ D_{conf} & si\ a = catering \end{cases}$$

$$\forall a \in A_{\upsilon_{(catering,d,d')}(\psi_{catering}(r)} \quad (6.169)$$

En lo que respecta al cuerpo de $\upsilon_{(catering,d,d')}(\psi_{catering}(r))$, éste está definido de la misma forma que el cuerpo de la MRCD r, esto es, según se indica en la Tabla 5.10. Esta equivalencia responde a que los atributos no participantes en la agregación no varían su valor y a que las funciones desagregadoras d y d' indicadas anteriormente permiten restablecer los grados de certidumbre de las tuplas y de los valores del atributo catering *en éstas de la misma forma en que eran originalmente.*

Dado que $\upsilon_{(catering,d,d')}(\psi_{catering}(r))$ y r poseen una cabecera y cuerpo equivalentes, podemos concluir que ambas MRCD son iguales.

6.5.2.2. Agregación en un atributo con dominio impreciso

En segundo lugar, definido el operador de agregación en atributos con dominio incierto, definiremos el operador de agregación en atributos con dominio impreciso. Este operador de agregación reemplazará el dominio del atributo participante en la agregación por uno de tipo impreciso basado en el original. En otras palabras, el dominio del atributo participante en la agregación en la MRCD resultante será un dominio impreciso cuyo dominio subyacente será el dominio del citado atributo en la MRCD original.

Cada tupla en la MRCD resultante será una agregación de un conjunto de tuplas en la MRCD original. Concretamente, todas las tuplas cuyos valores para los atributos no participantes en la agregación coincidan serán agregadas en una única tupla en la MRCD resultante. El valor para los atributos no participantes en la agregación de una tupla de la MRCD resultante será igual que el valor para el conjunto de tuplas de la MRCD original que ésta agrega. Para el atributo participante en la agregación, el valor será una distribución de posibilidad cuyo soporte estará compuesto por el conjunto de los distintos valores que poseen para en citado atributo las tuplas agregadas. La información correspondiente al grado de posibilidad de cada uno de estos valores y de la cardinalidad de la tupla resultante será extraída de la cardinalidad de las tuplas originales.

Recordemos que la cardinalidad de las tuplas resultantes de un proceso de desagregación de un atributo con dominio impreciso procede de la agregación del grado de posibilidad de cada uno de los valores posibles para el atributo desagregado y de la cardinalidad de la tupla original. En este caso, este proceso deberá ser invertido. Al igual que en el caso del operador de agregación en atributos con dominio incierto, la agregación de grados de posibilidad y cardinalidades no puede ser revertida con la información disponible. Para realizar esta tarea, y de la misma forma que anteriormente, se suministrará al operador de agregación un par de funciones desagregadoras. En este caso, la primera de ellas devolverá la cardinalidad de la tupla resultante. La segunda devolverá el grado de posibilidad de cada uno de los elementos de que componen los valores imprecisos del atributo participante en la agregación en las tuplas resultantes.

En lo que respecta a los grados de certidumbre, el grado de certidumbre de cada tupla resultante procederá de la agregación conjuntiva de los grados de las tuplas originales que ésta agrega.

Definiremos formalmente el proceso de agregación descrito anteriormente como sigue.

Definición 6.50 (Agregación en un atributo con dominio impreciso)**.** *Sea m una MRCD. Sea i un atributo de m, $i \in A_m$. Sean d y d' unas funciones de la forma $d : sig^{|(sig(m)-\{i\})}(T_m) \mapsto \mathbb{N}_f$ y $d' : sig(T_m) \mapsto (0,1]$ tales que satisfacen la condición indicada en la Ecuación 6.170. En lo anterior, $sig^{|(sig(m)-\{i\})}(T_m)$ representa al conjunto de las signaturas restringidas al subconjunto de atributos signatura $(sig(m)-\{i\})$ de las tuplas de m que está definido según se indica en la Ecuación 6.171. A las anteriores funciones d y d' se las denominará* funciones desagregadoras.

$$\forall t_m \in T_m, T_m(t_m) = d(sig^{|(sig(m)-\{i\})}(t_m)) \boxtimes \{1/0, d'(sig(t_m))/1\} \quad (6.170)$$

$$sig^{|A}(T_m) = \{sig^{|A}(t_m) | t_m \in T_m\} \qquad \forall m \in \Delta, \forall A \subseteq A_m \quad (6.171)$$

La agregación en m del atributo i en un atributo con dominio impreciso determinada por el par de funciones desagregadoras d y d', notada como $\delta_{(i,d,d')}(m)$, se define de la forma indicada en la Ecuación 6.172.

$$\delta_{(i,d,d')}(m) = (\mathcal{H}_{\delta_{(i,d,d')}(m)}, \mathcal{B}_{\delta_{(i,d,d')}(m)}) \quad (6.172)$$

La cabecera y cuerpo de la anterior MRCD están definidos según se indica, respectivamente, en las ecuaciones 6.173 y 6.174.

$$\mathcal{H}_{\delta_{(i,d,d')}(m)} = (A_m, dom_{\delta_{(i,d,d')}(m)}) \quad (6.173)$$

$$\mathcal{B}_{\delta_{(i,d,d')}(m)} = (T_{\delta_{(i,d,d')}(m)}, U_{\delta_{(i,d,d')}(m)}) \quad (6.174)$$

Los componentes de la cabecera y cuerpo descritos anteriormente están definidos como sigue:

- *La función de dominio de la MRCD resultante, $dom_{\delta_{(i,d,d')}(m)}$ se define según se indica en la Ecuación 6.175.*

$$dom_{\delta_{(i,d,d')}(m)}(a) = \begin{cases} dom_m(a) & si\ a \neq i \\ I \in \mathbb{I} | sub(I) = dom_m(i) & si\ a = i \end{cases} \quad \forall a \in A_m \tag{6.175}$$

- *El multiconjunto difuso de tuplas de $\delta_{(i,d,d')}(m)$ está definido según la función característica que se indica en la Ecuación 6.176.*

$$\omega_{T_{\delta_{(i,d,d')}(m)}}(t) = \\ = \begin{cases} d(sig(t)) & si\ \exists t_m \in T_m | (t_m^{|A_m - \{i\}} = t^{|(A_m - \{i\})}) \wedge \\ & \quad t(i) = \{d'(sig(t_m))/t_m(i) | t_m \in T_m \wedge \\ & \quad sig^{|(sig(m)-\{i\})}(t_m) = sig(t)\} \\ \{1.0/0\}_2 & en\ caso\ contrario \end{cases}$$

$$\forall t \in \mathbb{T}_{\delta_{(i,d,d')}(m)} \tag{6.176}$$

- *La función de certidumbre de la relación resultante, $\delta_{(i,d,d')}(m)$, queda definida de la forma indicada en la Ecuación 6.177. En la citada ecuación, el símbolo $\bigotimes$ representa a una t-norma.*

$$U_{\delta_{(i,d,d')}(m)}(t) = \bigotimes_{\substack{t_m \in T_m | \\ sig^{|(sig(m)-\{i\})}(t_m) = sig(t)}} U_m(t_m)$$

$$\forall t \in T_{\delta_{(i,d,d')}(m)} \tag{6.177}$$

Nótese que, en lo anterior, las signaturas contenidas en $sig^{|(sig(m)-\{i\})}(T_m)$ son de la misma forma que las contenidas en $sig(T_{\delta_{(i,d,d')}(m)})$. Las signaturas de $\delta_{(i,d,d')}(m)$ incluyen valores para todos los atributos signatura de m a excepción del atributo i, exactamente de la misma forma que las signaturas restringidas excluyendo i de las tuplas de m. Obviamente, esta restricción en las signaturas es necesaria sólo en el caso de que i fuese un atributo signatura de m, ya que en otro caso los conjuntos de atributos signatura de ambas MRCD son equivalentes sin necesidad de restricción alguna. Como se ha descrito anteriormente, el proceso de agregación sustituye el dominio asociado al atributo i en m por un dominio de tipo multiconjuntivo difuso. Ya que los dominios de este tipo no son dominios signatura, el atributo i queda excluido del conjunto de atributos signatura de $\delta_{(i,d,d')}(m)$. Por tanto, y según lo anterior, podemos afirmar que las signaturas restringidas excluyendo el atributo i de m son de la misma forma que las contenidas en $sig(T_{\delta_{(i,d,d')}(m)})$. Debido a esto, las signaturas de las tuplas de $\delta_{(i,d,d')}(m)$ y las signaturas restringidas excluyendo el atributo i de m se equiparan y se usan de forma indistinta en la Ecuación 6.166.

d	$sig^{\|(sig(\eth_{asistentes}(r))-\{asistentes\})}(t_{\eth_{asistentes}(r)})$			
	celebración			catering
	fecha	inicio	duración	
$\{1.0/2, 0.6/5, 0.3/6\}_c$	4-6-2010	9:00	5	sí
$\{0.6/3\}_c$	14-10-2010	17:00	3	no
$\{1.0/2\}_c$	21-11-2010	15:00	6	sí
$\{0.3/4\}_c$	9-12-2010	17:00	4	no

Tabla 6.30: Definición de la función desagregadora d para $\eth_{asistentes}(r)$

El nexo descrito anteriormente entre las signaturas de $\delta_{(i,d,d')}(m)$ y las asignaturas restringidas excluyendo el atributo i de m permite establecer un vínculo entre tuplas originales y resultantes. Como se comentó con anterioridad, el presente operador agrega en una sola tupla a las tuplas cuyos valores atributos, distintos del atributo i, coincidan. Debido a que la MRCD original se ha de encontrar en FNPE, podemos restringir este conjunto de atributos a los atributos signatura excluyendo al atributo i. Esta relación de *muchos a uno* queda reflejada en la expresión de la condición de pertenencia a $T_{\delta_{(i,d,d')}(m)}$ impuesta en la Ecuación 6.166. Dicha condición relaciona a una tupla t de $T_{\delta_{(i,d,d')}(m)}$ con el conjunto de tuplas pertenecientes a m que son agregadas por t. Nótese que en dicha expresión, dada una tupla $t \in \mathbb{T}_{\delta_{(i,d,d')}(m)}$, su signatura $sig(t)$ coincide con la signatura restringida $sig^{|(sig(m)-\{i\})}(t_m)$ de las tuplas t_m de m que son agregadas por t.

De la misma forma que se ha indicado para el conjunto de tuplas, y según se muestra en la expresión de la función de certidumbre indicada en la Ecuación 6.177, los grados de certidumbre de las tuplas origen quedan agregados de forma conjuntiva en la tupla destino.

Ejemplo 6.17. *En esta ocasión, aplicaremos el operador de agregación definido anteriormente sobre la MRCD $\eth_{asistentes}(r)$, definida en el Ejemplo 6.11. En esta agregación participará el atributo* asistentes *y nos proponemos obtener como resultado la MRCD r. Para que el proceso de agregación resulte en la MRCD r, definiremos las funciones desagregadoras d y d' como se indica en las tablas 6.30 y 6.31.*

Notaremos el resultado del proceso de agregación descrito anteriormente de la forma $\delta_{(asistentes,d,d')}(\eth_{asistentes}(r))$. Éste es una MRCD cuyo conjunto de atributos es igual al de $\eth_{asistentes}(r)$. Su función dominio varía con respecto a su precursora, estando ésta definida como se indica en la Ecuación 6.178. En dicha ecuación, se emplea el dominio D_{asist} que, según se indicó en el Ejemplo 5.11, es un dominio de tipo impreciso cuyo dominio subyacente es el dominio D_{nat}.

$$dom_{\delta_{(asistentes,d,d')}(\eth_{asistentes}(r))}(a) = \begin{cases} dom_{\eth_{asistentes}(r)}(a) & si\ a \neq asistentes \\ D_{asist} & si\ a = asistentes \end{cases}$$

$$\forall a \in A_{\delta_{(asistentes,d,d')}(\eth_{asistentes}(r))} \quad (6.178)$$

d'	$sig(t_{\delta_{asistentes}(r)})$				
	celebración			asistentes	catering
	fecha	inicio	duración		
1.0	4-6-2010	9:00	5	20	sí
1.0	4-6-2010	9:00	5	⋮	sí
1.0	4-6-2010	9:00	5	25	sí
0.25	14-10-2010	17:00	3	37	no
0.5	14-10-2010	17:00	3	38	no
0.75	14-10-2010	17:00	3	39	no
1.0	14-10-2010	17:00	3	40	no
0.75	14-10-2010	17:00	3	41	no
0.5	14-10-2010	17:00	3	42	no
0.25	14-10-2010	17:00	3	43	no
1.0	21-11-2010	15:00	6	20	sí
1.0	9-12-2010	17:00	4	1	no
1.0	9-12-2010	17:00	4	⋮	no
1.0	9-12-2010	17:00	5	5	no

Tabla 6.31: Definición de la función desagregadora d' para $\delta_{asistentes}(r)$

En lo que respecta al cuerpo de $\delta_{(asistentes,d,d')}(\delta_{asistentes}(r))$, éste se define de la misma manera que r, cuya definición es tal y como se indicó en la Tabla 5.10.

Visto que $\delta_{(asistentes,d,d')}(\delta_{asistentes}(r))$ y r poseen una cabecera y cuerpo equivalentes (obsérvese la equivalencia de sus conjuntos de atributos y de sus funciones de dominio), podremos decir que son dos MRCD iguales.

6.5.2.3. Agregación en un atributo con dominio compuesto

Una vez definido el operador de agregación en atributos con dominio impreciso, continuaremos en tercer lugar con la definición del operador de agregación en atributos con dominio compuesto. Este operador de agregación creará en la MRCD resultante un nuevo atributo con dominio compuesto a partir de un subconjunto de los atributos en la MRCD original. Como es natural, los atributos de la MRCD original agregados en el atributo de dominio compuesto en la MRCD resultante serán eliminados de esta última. El dominio compuesto del nuevo atributo será de tal forma que tendrá como atributos a los agregados, coincidiendo nombres y dominios.

Cada tupla de la MRCD original se corresponderá con una en la MRCD resultante. Obviamente, los valores para los atributos no agregados serán los mismos en tupla original y resultante. El valor para el nuevo atributo producto de la agregación en la tupla resultante será un valor compuesto por los valores de los atributos agregados en la tupla original. Las cardinalidades y grados de certidumbre no se verán afectados por el proceso de agregación, y por tanto serán iguales en las MRCDs original y resultante.

El operador de agregación descrito anteriormente se define de manera formal según se indica a continuación.

Definición 6.51 (Agregación en un atributo con dominio compuesto)**.** *Sea m una MRCD. Sea A un subconjunto de los atributos de m, $A \subseteq A_m$. Sea o un nombre de atributo, $o \in \mathbb{A}$, que no forma parte del conjunto de atributos de m, $o \notin A_m$.*

La agregación en m de su subconjunto de atributos A en un atributo o de dominio compuesto, notada como $\theta_{(A,o)}(m)$, se define de como se indica en la Ecuación 6.179.

$$\theta_{(A,o)}(m) = (\mathcal{H}_{\theta_{(A,o)}(m)}, \mathcal{B}_{\theta_{(A,o)}(m)}) \tag{6.179}$$

Las cabecera y cuerpo de $\theta_{(A,o)}(m)$ se definen de la forma indicada en las ecuaciones 6.180 y 6.181 respectivamente.

$$\mathcal{H}_{\theta_{(A,o)}(m)} = (A_{\theta_{(A,o)}(m)}, dom_{\theta_{(A,o)}(m)}) \tag{6.180}$$

$$\mathcal{B}_{\theta_{(A,o)}(m)} = (T_{\theta_{(A,o)}(m)}, U_{\theta_{(A,o)}(m)}) \tag{6.181}$$

Los componentes de los elementos anteriores se definen de la forma siguiente:

- *El conjunto de atributos de $\theta_{(A,o)}(m)$, $A_{\theta_{(A,o)}(m)}$, se define como indica la Ecuación 6.182.*

$$A_{\theta_{(A,o)}(m)} = (A_m - A) \cup \{o\} \tag{6.182}$$

- *La función de dominio de la MRCD resultante, $dom_{\theta_{(A,o)}(m)}$ se define de la forma indicada en la Ecuación 6.183. En dicha ecuación, O_m^A es un dominio de tipo compuesto, $O_m^A \in \mathbb{O}$, tal que $att(O_m^A) = A$ y su función de dominio está definida como indica la Ecuación 6.184.*

$$dom_{\theta_{(A,o)}(m)}(a) = \begin{cases} dom_m(a) & si\ a \in (A_m - A) \\ O_m^A & si\ a = o \end{cases} \qquad \forall a \in A_{\theta_{(A,o)}(m)} \tag{6.183}$$

$$dom_{O_m^A}(a) = dom_m(a) \qquad \forall m \in \Delta, \forall A \subseteq A_m, \forall a \in A \tag{6.184}$$

- *El multiconjunto difuso de tuplas de $\theta_{(A,o)}(m)$ está definido según la función característica que se indica en al Ecuación 6.185. En dicha ecuación, $t_m^{(A,o)}$ es una tupla de $\mathbb{T}_m$ derivada de $t \in \mathbb{T}_{\theta_{(A,o)}(m)}$ definida como se indica en la Ecuación 6.186.*

$$\omega_{T_{\theta_{(A,o)}(m)}}(t) = \omega_{T_m}(t_m^{(A,o)}) \qquad \forall t \in \mathbb{T}_{\theta_{(A,o)}(m)} \tag{6.185}$$

$$t_m^{(A,o)}(a) = \begin{cases} t(a) & si\ a \notin A \\ t(o)(a) & si\ a \in A \end{cases}$$

$$\forall m \in \Delta, \forall A \subseteq A_m, \forall o \in (\mathbb{A} - A_m), \forall t \in \mathbb{T}_{\theta_{(A,o)}(m)}, \forall a \in A_m \tag{6.186}$$

- *La función de certidumbre de la relación resultante, $\theta_{(A,o)}$, queda definida de la forma indicada en al Ecuación 6.187.*

$$U_{\theta_{(A,o)}(m)}(t) = U_m(t_m^{(A,o)}) \qquad \forall t \in T_{\theta_{(A,o)}(m)} \tag{6.187}$$

En lo anterior, se ha hecho uso de la tupla derivada $t_m^{(A,o)}$ para emparejar a una tupla resultante con su original. De esta forma, dada una tupla $t \in \theta_{(A,o,O)}(m)$, se puede determinar con facilidad la tupla en m que se le corresponde para transferirle su cardinalidad y certidumbre asociada.

Nótese que en la definición anterior no ha sido necesario el uso de funciones desagregadoras como en casos anteriores. En el presente caso, al no existir agregación conjuntiva de cardinalidades ni grados de certidumbre, no resulta necesario restaurar dichos valores.

Ejemplo 6.18. *En este caso, siguiendo la línea de los ejemplos anteriores para el caso de los operadores de agregación, procederemos a realizar la agregación del conjunto de atributos $A = \{fecha, inicio, duración\}$ de la MRCD $\phi_{celebración}(r)$ en el atributo* celebración *con un dominio de tipo compuesto. El objetivo, como anteriormente, será que la MRCD resultante sea equivalente a r.*

La MRCD que resulta del proceso de agregación descrito será notada como $\theta_{(A,celebración)}(\phi_{celebración}(r))$. En lo que respecta a la cabecera de esta MRCD, ésta tendrá como conjunto de atributos el que se muestra en la Ecuación 6.188 y como función dominio la que se define en la Ecuación 6.189. En la definición de la función dominio se emplea el dominio $O^A_{\phi_{celebración}(r)}$ cuyo conjunto de atributos se corresponde con $att(O^A_{\phi_{celebración}(r)}) = A$ y cuya función dominio se define en la Ecuación 6.190. Obsérvese que tanto el conjunto de atributos como la función de dominio de $O^A_{\phi_{celebración}(r)}$ equivalen a las de D_{celeb}. Este dominio, D_{celeb}, ha sido definido en el Ejemplo 5.17.

$$A_{\theta_{(A,celebración)}(\phi_{celebración}(r))} = \{celebración, asistentes, catering, intérpretes\} \tag{6.188}$$

$$dom_{\theta_{(A,celebración)}(\phi_{celebración}(r))}(a) = \\ = \begin{cases} dom_{\phi_{celebración}(r)}(a) & si\ a \in \{asistentes, catering, intérpretes\} \\ O^A_{\phi_{celebración}(r)} & si\ a = celebración \end{cases}$$

$$\forall a \in A_{\theta_{(A,celebración)}(\phi_{celebración}(r))} \tag{6.189}$$

$$dom_{O^A_{\phi_{celebración}(r)}}(a) = dom_{\phi_{celebración}(r)}(a) \qquad \forall a \in A \tag{6.190}$$

En lo que se refiere al cuerpo de la MRCD resultante del proceso de agregación, $\theta_{(\{fecha,inicio,duración\},celebración)}(\phi_{celebración}(r))$ tendrá un cuerpo similar al de r. El cuerpo de esta última MRCD está definido según se mostró anteriormente en la Tabla 5.10.

Dado que $\theta_{(A,celebración)}(\phi_{celebración}(r))$ y r poseen una cabecera y cuerpo equivalentes (obsérvese la equivalencia de sus funciones de dominio dado que

los dominios $O^{A}_{\phi_{celebración}(r)}$ *y* D_{celeb} *son equivalentes), podremos decir que ambas MRCDs son iguales.*

6.5.2.4. Agregación en un atributo con dominio multiconjuntivo difuso

Finalmente, definiremos el operador de agregación en atributos con dominio multiconjuntivo difuso. Este operador de agregación tendrá, al igual que ocurría con su contrapartida, un funcionamiento muy similar al descrito para el operador de agregación en un atributo con dominio impreciso.

El dominio del atributo participante en la agregación será sustituido por uno de tipo multiconjuntivo difuso con un dominio subyacente igual al dominio original del citado atributo. Las tuplas de la MRCD resultante se corresponderán con el conjunto de las tuplas de la MRCD original que comparten el mismo valor para los atributos no participantes en la agregación.

Los valores para dichos atributos serán iguales tanto en la tupla resultante como en las tuplas originales agregadas. El valor para el atributo participante en la agregación será un multiconjunto difuso cuyo soporte estará formado por los distintos valores que poseen para dicho atributo las tuplas agregadas.

La cardinalidad asociada a cada uno de elementos del soporte así como la cardinalidad asociada a la tupla resultante serán extraídas de las cardinalidades de las tuplas originales. Recordemos, una vez más, que dichas cardinalidades son agregadas mediante su producto en el operador de desagregación complementario. El resultado de dicha agregación se emplea como cardinalidad de las tuplas resultantes. Revertir este proceso de agregación no es posible exclusivamente con la información contenida en el resultado de dicho proceso. Debido a esto, se ha optado, al igual que en los anteriores casos, por el suministro al operador de agregación de un par de funciones desagregadoras. La primera de ellas proporcionará la cardinalidad que se asociará a cada tupla de la MRCD resultante del proceso de agregación. La segunda proporcionará, para cada una de las tuplas de la relación resultante, las cardinalidades de cada uno de los elementos que forman los multiconjuntos difusos que se emplean como valores del atributo participante en la agregación.

En las definiciones de los operadores de agregación descritos en anteriores apartados no fue necesario prestar especial atención al tratamiento de valores ausentes por los motivos que se comentaron en el anterior apartado dedicado a la definición del operador de desagregación de atributos con dominio multiconjuntivo difuso. A diferencia de los citados casos, en el presente sí se tendrá que prestar atención a la presencia de valores ausentes. Concretamente, se tendrá en cuenta la relación entre los valores ausentes $\emptyset_N$ y $dne_{sub(N)}$ (donde N es un dominio complejo de tipo multiconjuntivo difuso) que establece el operador de desagregación de atributos con dominio multiconjuntivo difuso. Debido a esto, será necesario variar la definición usual de las funciones desagregadoras d y d' que ha sido empleada en los anteriores apartados de tal forma que se considere esta particularidad. En concreto, se deberá introducir una variación en las condiciones usuales que se aplican a dichas funciones de forma tal que, en el caso especial de presencia de valores ausentes descrito anteriormente, la función desagregadora d transfiera a la tupla resultante la cardinalidad de la tupla original y la función d' sea de tal forma que el valor multiconjuntivo difuso resultante de la agregación sea el multiconjunto difuso vacío.

En lo que respecta al grado de certidumbre asociado a las tuplas de la MRCD resultante del proceso de agregación, parece lógico, como en casos anteriores, que éste proceda de la agregación conjuntiva, empleando una t-norma, del grado de certidumbre asociado a las tuplas de la MRCD original de las que éstas deriven.

Según lo anterior, el proceso de agregación descrito anteriormente se define formalmente como se indica a continuación.

Definición 6.52 (Agregación en un atributo con dominio multiconjuntivo difuso)**.** *Sea m una MRCD. Sea n un atributo de m, $n \in A_m$. Sean d y d' unas funciones de la forma $d : sig^{|(sig(m)-\{n\})}(T_m) \mapsto \mathbb{N}_f$ y $d' : sig(T_m) \mapsto \mathbb{N}_f$ tales que cumplen las condiciones indicadas en las ecuaciones 6.191 y 6.192. A las anteriores funciones se las denominará funciones desagregadoras.*

$$\forall t_m \in T_m | (t_m(n) \neq dne_{dom_m(n)}),$$
$$T_m(t_m) = d(sig^{|(sig(m)-\{n\})}(t_m)) \boxtimes d'(sig(t_m)) \quad (6.191)$$

$$\forall t_m \in T_m | (t_m(n) = dne_{dom_m(n)}),$$
$$T_m(t_m) = d(sig^{|(sig(m)-\{n\})}(t_m)) \ \wedge \ d'(sig(t_m)) = \{1.0/0\}_2 \quad (6.192)$$

La agregación en m del atributo n en un dominio multiconjuntivo difuso determinada por el par de funciones desagregadoras d y d', notada como $\nu_{(n,d,d')}(m)$, se define de la forma que se indica en la Ecuación 6.193.

$$\nu_{(n,d,d')}(m) = (\mathcal{H}_{\nu_{(n,d,d')}(m)}, \mathcal{B}_{\nu_{(n,d,d')}(m)}) \quad (6.193)$$

La cabecera y cuerpo de $\nu_{(n,d,d')}(m)$ se definen como se indica en las ecuaciones 6.194 y 6.195 respectivamente.

$$\mathcal{H}_{\nu_{(n,d,d')}(m)} = (A_m, dom_{\nu_{(n,d,d')}(m)}) \quad (6.194)$$

$$\mathcal{B}_{\nu_{(n,d,d')}(m)} = (T_{\nu_{(n,d,d')}(m)}, U_{\nu_{(n,d,d')}(m)}) \quad (6.195)$$

Cada uno de los componentes de los elementos anteriores se define como se indica a continuación:

- *La función de dominio de la MRCD resultante, $dom_{\nu_{(n,d,d')}(m)}$ se define de la como se indica en la Ecuación 6.196.*

$$dom_{\nu_{(n,d,d')}(m)}(a) = \begin{cases} dom_m(a) & si\ a \neq n \\ M \in \mathbb{M} | (sub(M) = dom_m(n)) & si\ a = n \end{cases}$$
$$\forall a \in A_m \quad (6.196)$$

- *El multiconjunto difuso de tuplas de $\nu_{(n,d,d')}(m)$ está definido según la función característica que se indica en la Ecuación 6.197.*

$$\omega_{T_{\nu_{(n,d,d')}(m)}}(t) = \\ = \begin{cases} d(sig(t)) & si\ \exists t_m \in T_m | (t_m^{|A_m - \{n\}} = t^{|(A_m - \{n\})} \wedge \\ & \quad t(n) = [d'(sig(t_m))/t_m(n) | (t_m \in T_m \wedge \\ & \quad sig^{|(sig(m)-\{n\})}(t_m) = sig(t))] \\ \{1.0/0\}_2 & en\ caso\ contrario \end{cases}$$

$$\forall t \in \mathbb{T}_{\nu_{(n,d,d')}(m)} \quad (6.197)$$

- *La función de certidumbre de la relación resultante, $\nu_n(m)$, queda definida de la forma indicada en la Ecuación 6.198.*

$$U_{\nu_{(n,d,d')}(m)}(t) = \bigotimes_{\substack{t_m \in T_m | \\ (sig^{|(sig(m)-\{n\})}(t_m) = sig(t))}} U_m(t_m)$$

$$\forall t \in T_{\nu_{(n,d,d')}(m)} \quad (6.198)$$

Como en el caso del operador de agregación en atributos con dominio impreciso, la relación *uno a muchos* de una tupla en $\nu_{(n,d,d')}(m)$ con las tuplas en m facilita la definición de su multiconjunto de tuplas y de la función de certidumbre. Al igual que en el mencionado caso, las signaturas en $sig(T_{\nu_{(n,d,d')}(m)})$ son de la misma forma que las contenidas en $sig^{|(sig(m)-\{n\})}(T_m)$. Por tanto, estas se usan de manera indistinta y se igualan en las ecuaciones 6.197 y 6.198. De la misma forma, este nexo entre signaturas permite la formación de una relación entre un conjunto de tuplas originales y la tupla resultante que las agrega.

Además de lo anterior, se ha de destacar la forma en que se definen las funciones desagregadoras en la definición anterior. La definición de éstas diferencia los casos en que el valor del atributo participante en el proceso de agregación de la tupla tratada sea o no un valor ausente *dne*. En el primer caso, se asegurará que la función desagregadora d transfiere por completo la cardinalidad de la tupla original a la que tupla que se corresponde con ésta en la MRCD resultante. Con respecto a la función desagregadora d', se asignará una cardinalidad nula a cualquier elemento del multiconjunto difuso que resulta como valor, de la tupla en la MRCD resultante que se corresponde con la tratada, para el atributo participante en el proceso de agregación. En dicho caso, por tanto, el valor multiconjuntivo asociado al atributo participante en la agregación será el multiconjunto vacío. En cualquier otro caso en que no participe un valor ausente, las funciones desagregadoras se comportarán de manera usual.

El siguiente ejemplo ilustra el comportamiento del operador de agregación de atributos en dominios de tipo multiconjuntivo según la definición anterior.

d	$sig^{\mid(sig(\mu_{intérpretes}(r^3))-\{intérpretes\})}(t_{\mu_{intérpretes}(r^3)})$			
	celebración			catering
	fecha	inicio	duración	
$\{1.0/2, 0.6/5, 0.3/6\}_c$	4-6-2010	9:00	5	sí
$\{0.6/3\}_c$	14-10-2010	17:00	3	no
$\{1.0/2\}_c$	21-11-2010	15:00	6	sí
$\{0.3/4\}_c$	9-12-2010	17:00	4	no

Tabla 6.32: Definición de la función desagregadora d para $\mu_{intérpretes}(r^3)$

d'	$sig(t_{\mu_{intérpretes}(r^3)})$				
	celebración			catering	intérpretes
	fecha	inicio	duración		
$\{1.0/1, 0.6/3, 0.3/8\}_c$	4-6-2010	9:00	5	sí	Inglés
$\{1.0/1, 0.6/2, 0.3/4\}_c$	4-6-2010	9:00	5	sí	Alemán
$\{0.3/10\}_c$	14-10-2010	17:00	3	no	Inglés
$\{1.0/3, 0.3/6\}_c$	21-11-2010	15:00	6	sí	Inglés
$\{1.0/1, 0.6/5\}_c$	21-11-2010	15:00	6	sí	Francés
$\{0.3/4\}_c$	21-11-2010	15:00	6	sí	Italiano
$\{1.0/0\}_c$	9-12-2010	17:00	4	no	dne_{lang}

Tabla 6.33: Definición de la función desagregadora d' para $\mu_{intérpretes}(r^3)$

Ejemplo 6.19. *Para el caso de este ejemplo aplicaremos el operador de agregación definido anteriormente sobre la MRCD $\mu_{intérpretes}(r^3)$, MRCD que es equivalente a la MRCD s definida en el Ejemplo 6.15. El atributo implicado en la agregación será, en este caso, el atributo* intérpretes. *A diferencia de los ejemplos anteriores, el objetivo de la aplicación de la agregación anterior no es obtener r. Por los motivos que se expusieron en el Ejemplo 6.13, la MRCD base con la que trabajaremos en el presente ejemplo, y que se obtendrá como resultado, será r^3. La definición de dicha MRCD puede encontrarse en el citado ejemplo. Para la consecución de este objetivo, proporcionaremos al operador de agregación las funciones desagregadoras d y d' definidas según se indica respectivamente en las tablas 6.32 y 6.33.*

La MRCD resultante será notada cómo $\nu_{(intérpretes,d,d')}(\mu_{intérpretes}(r^3))$. En lo que respecta a su cabecera, el conjunto de atributos de esta MRCD será equivalente a la de r^3, pues ni el presente operador de agregación ni su operador complementario de desagregación introducen cambios en la cabecera de las MRCDs resultantes. Con respecto a la función dominio de $\nu_{(intérpretes,d,d')}(\mu_{intérpretes}(r^3))$, ésta está definida según se indica en la Ecuación 6.199. En dicha ecuación se hace uso del dominio D_{lang} definido previamente en el Ejemplo 5.15.

$$dom_{\nu_{(intérpretes,d,d')}(\mu_{intérpretes}(r^3))}(a) =$$
$$= \begin{cases} dom_{\mu_{intérpretes}(r^3))}(a) & si\ a \neq intérpretes \\ M \in \mathbb{M} | sub(M) = D_{lang} & si\ a = intérpretes \end{cases}$$

$$\forall a \in A_{\nu_{(intérpretes,d,d')}(\mu_{intérpretes}(r^3))} \quad (6.199)$$

Obsérvese la equivalencia de la función de dominio anterior y la asociada a r^3. Para el caso de atributos distintos al atributo intérpretes *ésta es obvia, ya que la función de dominio $dom_{\mu_{intérpretes}(r^3)}$, definida en el Ejemplo 6.15, es equivalente a dom_{r^3}. En lo que respecta al caso del atributo* intérpretes*, éste tiene asociado un dominio de tipo multiconjuntivo difuso cuyo dominio subyacente es D_{lang}. Este dominio es equivalente al dominio D_{inter}, dominio asociado en r^3 al citado atributo.*

En lo que se refiere al cuerpo $\nu_{(intérpretes,d,d')}(\mu_{intérpretes}(r^3))$, éste es equivalente al que se indica para la MRCD r^3 en la Tabla 6.24 del Ejemplo 6.13. Obsérvese que la última tupla del cuerpo de $\nu_{(intérpretes,d,d')}(\mu_{intérpretes}(r^3))$ (siguiendo el orden en que se muestran las tuplas en la Tabla 6.24) se incorpora gracias al caso especial para el tratamiento de valores ausentes de la definición propuesta de las funciones desagregadoras d y d'. Este caso especial, recordemos, permite la asignación de cardinalidades nulas por parte de d' en caso de la presencia de valores ausentes de tipo dne. Dicha asignación nula resulta en el establecimiento del multiconjunto difuso vacío como valor del atributo intérpretes *para la citada tupla. Finalmente, y gracias de nuevo a la excepción incorporada a la definición de las funciones desagregadoras d y d', la función d permite asignar la cardinalidad original que dicha tupla poseía en r^3.*

Con todo lo anterior, y visto que la cabecera y cuerpo de la MRCD resultante del proceso de agregación, $\nu_{(intérpretes,d,d')}(\mu_{intérpretes}(r^3))$, y r^3 son similares, podemos concluir que éstas son MRCDs equivalentes.

6.6. Conclusiones

En el presente capítulo se ha propuesto un álgebra que permite la manipulación de MRCDs. Parte de los operadores que componen este álgebra están concebidos como una transposición al ámbito de las MRCDs del conjunto de operadores del álgebra relacional clásica, por lo que el álgebra propuesta hereda las capacidades de manipulación de datos que posee su análoga clásica.

Además de los operadores derivados del álgebra relacional, el álgebra de MRCDs contiene un subconjunto de operandos destinados a la transformación de los dominios complejos de los atributos de las MRCDs, que se dividen en pares complementarios de agregadores y desagregadores. El motivo fundamental de este tipo de operadores es permitir desagregación de los componentes de un valor complejo de tal forma que los operadores derivados del álgebra relacional clásica pueden acceder a los citados componentes. Lógicamente, por cada operador de desagregación hará un operador de agregación complementario que permita restaurar los dominios complejos originales en caso de ser necesario.

Para el adecuado funcionamiento de este álgebra se ha propuesto una forma normal para la MRCDs. Ésta tiene como objeto asegurar la complementación de este subconjunto de operandos destinado a la transformación de dominios, a fin que, tras la descomposición de un dominio complejo de una determinada MRCD, se pueda obtener la misma a la aplicación posterior del operador de agregación correspondiente.

Capítulo 7

Prototipo de sistema de gestión de bases de datos objeto-relacional difuso

7.1. Introducción

La presente sección está dedicada a presentar y analizar la propuesta de extensión de un SGBD objeto-relacional, con el objeto de dotarlo de capacidades para el almacenamiento y manipulación de datos imperfectos, modelados empleando lógica difusa.

Para ello, se realizará una descripción de la propuesta detallando los tipos de datos imperfectos soportados, junto con los operadores introducidos para la manipulación de este tipo de datos. Posteriormente, se presentará la jerarquía que modela la descripción de la citada extensión. En tercer lugar, se describirán los aspectos más relevantes del prototipo que implementa la mencionada extensión. Finalmente, se valorarán las ventajas que esta propuesta ofrece.

7.2. Descripción de la propuesta

Como hemos concluido en el Capítulo 4, en el que se motivan las propuestas realizadas en el presente trabajo, la estrategia basada en la extensión de un SGBD existente, con objeto de incorporar en éste capacidades para el almacenamiento y manipulación de datos imprecisos, es la más ventajosa de las analizadas. En concreto, los SGBD objeto-relacionales parecen los más apropiados para la implantación de este tipo de extensiones debido, fundamentalmente, a la gran variedad de mecanismos de extensión que éstos ofrecen a los usuarios.

La presente propuesta sigue precisamente la aproximación anterior, planteando la extensión de un sistema de gestión objeto-relacional existente, mediante la implementación en el mismo de una jerarquía de tipos de datos definidos por el usuario, que permitirían aumentar las capacidades del sistema anfitrión dotándole de la capacidad para almacenar y manipular datos imperfectos.

El almacenamiento de datos imperfectos se llevará a cabo empleando la estructura interna de los UDT, siendo los métodos asociados a éstos los encargados

de implementar la lógica necesaria para manipular dicho tipos datos.

Dado que la presente propuesta es un prototipo, cuyo cometido principal es demostrar la viabilidad de un SGBDD siguiendo la estrategia que se propone en el presente trabajo, circunscribiremos ésta a la propuesta realizada en [10], basada en los trabajos iniciales [104, 55]. Esta constituye un subconjunto de las capacidades ofrecidas por el modelo objeto-multirrelacional descrito en los capítulos 5 y 6. En estos trabajos se comienza a perfilar la propuesta de la creación de una extensión difusa para un SGBD objeto-relacional, definiendo UDTs para la representación de tipos difusos atómicos, los definidos en [70], junto con dos tipos de colecciones difusas (con semántica conjuntiva y disyuntiva, respectivamente), y objetos complejos con atributos imprecisos, similares a los propuestos en [100].

7.3. Tipos de datos imperfectos

La extensión propuesta está diseñada para permitir el almacenamiento y la manipulación varios tipos de datos imperfectos, analizados en la siguientes subsecciones.

7.3.1. Datos atómicos imperfectos

La extensión consta de una parte básica que permite el almacenamiento y la manipulación de datos atómicos imperfectos. El calificativo *atómico* indica que los datos que se definen sobre dominios básicos, cuyos elementos no están compuestos de otros elementos con semántica propia. A continuación, se describen las variantes de este tipo de datos que soporta la extensión propuesta.

7.3.1.1. Datos imprecisos sobre un dominio ordenado

Este tipo de datos están representados, en la presente propuesta, en forma de distribuciones de posibilidad trapezoidales y etiquetas lingüísticas asociadas a las anteriores. Denominaremos a este tipo de datos como OAFT (Ordered Atomic Fuzzy Type). Los diferentes casos concretos de distribuciones de posibilidad trapezoidales contemplados por la extensión son las siguientes:

- Datos crisp: Es un caso especial que se da cuando los cuatro valores que definen la distribución de posibilidad trapezoidal son iguales, esto es, distribuciones de posibilidad trapezoidales de la forma $[\alpha, \alpha, \alpha, \alpha]$. Este caso permite representar valores precisos, donde α es el valor que se desea representar.

- Datos intervalares: En este caso, los valores que definen la distribución de posibilidad trapezoidal son de la forma $[\alpha, \alpha, \beta, \beta]$, siendo $\alpha \leq \beta$. Esta distribución de posibilidad trapezoidal permite representar intervalos de valores de la forma $[\alpha, \beta]$.

- Datos trapezoidales: Es el caso general que se da cuando cada valor que define la distribución de posibilidad trapezoidal posee un valor propio, es decir, distribuciones de posibilidad trapezoidales de la forma $[\alpha, \beta, \gamma, \delta]$ donde $\alpha \leq \beta \leq \gamma \leq \delta$.

	Despejado	Intervalos nubosos	Nuboso
Lluvioso	0.1	0.4	0.7
Nuboso	0.4	0.7	
Intervalos nubosos	0.7		

Tabla 7.1: Relación de similitud para los valores del dominio meteorológico

Un ejemplo de este tipo de datos es el rango de edad de una determinada persona. La edad de una persona *joven* podría ser descrita mediante una distribución de posibilidad trapezoidal como, por ejemplo, $[14, 17, 24, 27]$. En este caso, el calificador *joven* se puede considerar la etiqueta lingüística asociada a dicha distribución.

7.3.1.2. Datos imprecisos sobre un dominio escalar no ordenado

La extensión representará estos datos como distribuciones de posibilidad sobre el dominio escalar definido por el usuario. Debido a que no existe una relación de orden establecida entre los elementos del dominio, el usuario habrá de definir una relación de similitud entre dichos elementos, con el objeto de permitir la comparación de los valores almacenados. Llamaremos a este tipo de datos NOAFT (Non Ordered Atomic Fuzzy Type).

Un ejemplo de este tipo de datos es el estado meteorológico reinante. Los valores del dominio podrían ser $\{despejado, intervalos\ nubosos, nuboso, lluvioso\}$. La relación de similitud entre los elementos del dominio podría ser como la mostrada en la Tabla 7.1.

7.3.2. Colecciones difusas

Otro grupo de tipos de datos que contempla la extensión propuesta es el de las colecciones difusas. Estas son una generalización difusa de las colecciones ordinarias definidas en SQL:1999.

El hecho de incluir borrosidad en las colecciones, supone que los elementos pertenecientes a la colección no son miembros de ésta de forma absoluta, sino que su pertenencia es gradual. Esto implica que cada elemento de la colección tendrá asignado un grado de pertenencia que estará incluido en el rango $[0, 1]$.

Se ha de hacer notar que, dado que los elementos de la colección pueden ser de cualquier tipo, éstos pueden ser de naturaleza *crisp* o imperfecta. De esta forma, es posible modelar de forma simultánea diversos tipos o niveles de imperfección.

A continuación, procederemos a describir los distintos tipos de colecciones, diferenciadas en función de su semántica, que soporta la extensión propuesta.

7.3.2.1. Colecciones imprecisas con semántica conjuntiva

La semántica de este tipo de colecciones condiciona la visión que el usuario tiene del conjunto que forman los elementos de la colección, considerándose que todos los elementos que forman parte de ésta son valores que se dan de forma simultánea para el atributo para el cual es valor. Este tipo de datos será denominado como CFC (Conjunctive Fuzzy Collection).

Un ejemplo de este tipo de datos, que servirá para ilustrar el concepto de semántica conjuntiva, puede ser el siguiente. Consideremos un atributo que albergará el conjunto de idiomas que puede hablar una persona, siendo el dominio escalar de los idiomas que pueden ser hablados {*español*, *inglés*, *frances*, *alemán*, *italiano*, *portugués*, *ruso*}. Para una determinada persona, el conjunto de idiomas que habla puede representarse por el conjunto difuso {1/*español*, 0.8/*inglés*, 0.75/*italiano*}. Como vemos, esta colección es una colección difusa puesto que los elementos tienen graduada su pertenencia. En este caso, dicho grado de pertenencia indica el nivel de dominio del idioma. La semántica de la colección es conjuntiva porque la persona domina, en cierto grado, cada uno de los idiomas presentes en la misma.

7.3.2.2. Colecciones difusas con semántica disyuntiva

Las colecciones con semántica disyuntiva son la antítesis, semánticamente hablando, al tipo de colecciones descritas anteriormente. En este caso, la semántica indica que sólo un elemento de la colección será el valor real del atributo para el cual ésta es valor. Por tanto, la colección incluirá entre sus elementos los distintos valores posibles del atributo, donde el grado de pertenencia de los elementos de la colección indicará el grado de posibilidad de que éstos sean el valor real. Denominaremos a este tipo de datos DFC (Disjunctive Fuzzy Collection).

Un ejemplo ilustrativo de este tipo de colecciones puede ser la colección que indica el diagnóstico previo, hecho en función de los síntomas que presenta un paciente inconsciente al ingresar con un problema de salud desconocido en un servicio de urgencias. Supongamos que el dominio de las patologías posibles es el dominio escalar {*ataque cardiaco*, *edema pulmonar*, *infarto cerebral*, *intoxicación etílica*}. El médico valorará el estado de paciente e indicará una lista de posibles patologías, en la que se indica, adicionalmente, el grado de posibilidad asociado a cada una para priorizar ciertas pruebas y tratamientos. Para un determinado paciente, este diagnostico previo podría ser representado por la colección disyuntiva {0.8/*ataque cardiaco*, 0.6/*infarto cerebral*, 0.2/*intoxicación etílica*}. Queda claro que sólo una de estas tres patologías propuestas será la que sufre realmente el paciente. De ahí la semántica disyuntiva de la colección.

7.3.3. Objetos difusos

La extensión permitirá al usuario definir sus propias clases para modelar objetos complejos difusos. Los objetos complejos difusos podrán poseer atributos de tipo tanto *crisp* como imperfecto, incluyendo otros objetos difusos. Precisamente, esta última característica es por la que se califica a este tipo de objetos como complejos. Denominaremos a este tipo de datos como FO (Fuzzy object).

Cada atributo tendrá asociado un grado de importancia dentro del objeto. Dicho grado de importancia será empleado por el mecanismo de cálculo de semejanza entre objetos difusos que todos los objetos difusos deberán implementar. Este grado de importancia determinará el peso que tendrá el atributo en el cálculo del citado grado de semejanza.

Un ejemplo de este tipo de datos puede ser una clase empleada para modelar inmuebles en venta en una base de datos. Los inmuebles poseerán los siguientes atributos:

- Identificador del inmueble: Éste será un número entero secuencial.
- Dirección del inmueble: Será un objeto que a su vez estará compuesto por los siguientes atributos:
  - Tipo de la vía: Será un valor del dominio escalar {*avenida*, *calle*, *plaza*, *carretera*}.
  - Nombre de la vía: Una cadena de caracteres que contendrá dicho nombre:
  - Número: Un atributo de tipo impreciso sobre dominio ordenado, que permitirá almacenar valores clásicos (un solo número) o intervalos (para el caso de fincas con un rango de números).
- Precio: Es un atributo de tipo impreciso sobre dominio ordenado que permitirá almacenar tanto precios concretos como rangos de precios.

Como podemos apreciar, la clase anterior modela objetos complejos, ya que incluyen dentro de su estructura a otros objetos, con algunos atributos que albergan valores imprecisos, y otros precisos.

7.4. Comparadores para los tipos definidos

A continuación, procederemos a describir los comparadores que contempla la presente propuesta para los tipos de datos anteriormente descritos.

7.4.1. Comparadores para OAFT

La presente subsección está dedicada a describir los comparadores que pueden ser aplicados sobre pares de datos del tipo OAFT.

7.4.1.1. Comparador de semejanza

El primer comparador que propondremos para este tipo de datos, es el *comparador de semejanza*.

Para el caso de los datos OAFT, este comprador está disponible en dos variantes, en función de la medida que se emplee para calcular la semejanza entre las citadas distribuciones de posibilidad. Estas variantes son la de *posibilidad*, denominada FEQ (Fuzzy EQual), cuando se aplica una medida de posibilidad para el cálculo de dicho grado, y la de *necesidad*, denominada NFEQ (Necessarily Fuzzy EQUal), cuando se aplica una medida de necesidad para el cálculo del grado de semejanza.

Las definición de las dos variantes, extraídas de [70], es la siguiente:

Definición 7.1 (Comparador de semejanza para tipos OAFT)**.** *Sean A y B dos distribuciones de posibilidad caracterizadas por las funciones $\mu_A(x)$ y $\mu_B(x)$, respectivamente, y definidas sobre el dominio U. Se definen el comparador de semejanza con medida de posibilidad (FEQ) y el comparador de semejanza con medida de necesidad (NFEQ), entre las citadas distribuciones de posibilidad, como se indica en las ecuaciones 7.1 y 7.2 respectivamente.*

Posibilidad	Necesidad	Significado
FEQ	NFEQ	(Necessarily) Fuzzy Equal: *Igual que* difuso
FGT	NFGT	(Necessarily) Fuzzy Greater Than: *Mayor que* difuso
FGEQ	NFEGQ	(Necessarily) Fuzzy Greater or Equal: *Mayor o igual que* difuso
FLT	NFLT	(Necessarily) Fuzzy Less Than: *Menor que* difuso
FLEQ	NFLEQ	(Necessarily) Fuzzy Less or Equal: *Menor o igual que* difuso

Tabla 7.2: Comparadores difusos para datos atómicos difusos numéricos

$$FEQ(A,B) = \sup_{x \in U} \text{mín}(\mu_A(x), \mu_B(x)) \tag{7.1}$$

$$NFEQ(A,B) = \inf_{x \in U} \text{máx}(1 - \mu_A(x), \mu_B(x)) \tag{7.2}$$

7.4.1.2. Operadores relacionales difusos

Debido a que el dominio subyacente de un tipo OAFT es un dominio ordenado, cabe la posibilidad de definir extensiones difusas de los operadores relacionales clásicos. Dichos comparadores, extraídos inicialmente de [70], se indican en la Tabla 7.2.

Las definiciones formales de estos comparadores son las siguientes:

Definición 7.2 (Operadores relacionales difusos)**.** *Sean A y B dos distribuciones trapezoidales de posibilidad definidas por $[\alpha_A, \beta_A, \gamma_A, \delta_A]$ y $[\alpha_B, \beta_B, \gamma_B, \delta_B]$ respectivamente. Se definen los operadores relaciones de posibilidad y necesidad, como sigue se indica en las ecuaciones 7.3 a 7.10.*

$$FGT(A,B) = \begin{cases} 1 & si\ \gamma_A \geq \delta_B \\ \frac{\delta_A - \gamma_B}{(\delta_B - \gamma_B) - (\gamma_A - \delta_A)} & si\ \gamma_A < \delta_B\ y\ \delta_A > \gamma_B \\ 0 & en\ otro\ caso\ (\delta_A \leq \gamma_B) \end{cases} \tag{7.3}$$

$$NFGT(A,B) = \begin{cases} 1 & si\ \alpha_A \geq \delta_B \\ \frac{\beta_A - \gamma_B}{(\delta_B - \gamma_B) - (\alpha_A - \beta_A)} & si\ \alpha_A < \delta_B\ y\ \beta_A > \gamma_B \\ 0 & en\ otro\ caso\ (\beta_A \leq \gamma_B) \end{cases} \tag{7.4}$$

$$FGEQ(A,B) = \begin{cases} 1 & si\ \gamma_A \geq \beta_B \\ \frac{\delta_A - \alpha_B}{(\beta_B - \alpha_B) - (\gamma_A - \delta_A)} & si\ \gamma_A < \beta_B\ y\ \delta_A > \alpha_B \\ 0 & en\ otro\ caso\ (\delta_A \leq \alpha_B) \end{cases} \tag{7.5}$$

$$NFGEQ(A,B) = \begin{cases} 1 & si\ \alpha_A \geq \beta_B \\ \frac{\beta_A - \alpha_B}{(\beta_B - \alpha_B) - (\alpha_A - \beta_A)} & si\ \alpha_A < \beta_B\ y\ \beta_A > \alpha_B \\ 0 & en\ otro\ caso\ (\beta_A \leq \alpha_B) \end{cases} \quad (7.6)$$

$$FLT(A,B) = \begin{cases} 1 & si\ \beta_A \leq \alpha_B \\ \frac{\alpha_A - \beta_B}{(\alpha_B - \beta_B) - (\beta_A - \alpha_A)} & si\ \beta_A > \alpha_B\ y\ \alpha_A < \beta_B \\ 0 & en\ otro\ caso\ (\alpha_A \geq \beta_B) \end{cases} \quad (7.7)$$

$$NFLT(A,B) = \begin{cases} 1 & si\ \delta_A \leq \alpha_B \\ \frac{\gamma_A - \beta_B}{(\alpha_B - \beta_B) - (\delta_A - \gamma_A)} & si\ \delta_A > \alpha_B\ y\ \gamma_A < \beta_B \\ 0 & en\ otro\ caso\ (\gamma_A \geq \beta_B) \end{cases} \quad (7.8)$$

$$FLEQ(A,B) = \begin{cases} 1 & si\ \beta_A \leq \gamma_B \\ \frac{\delta_B - \alpha_A}{(\beta_A - \alpha_A) - (\gamma_B - \delta_B)} & si\ \beta_A > \gamma_B\ y\ \alpha_A < \delta_B \\ 0 & en\ otro\ caso\ (\alpha_A \geq \delta_B) \end{cases} \quad (7.9)$$

$$NFLEQ(A,B) = \begin{cases} 1 & si\ \delta_A \leq \gamma_B \\ \frac{\gamma_A - \delta_B}{(\gamma_B - \delta_B) - (\delta_A - \gamma_A)} & si\ \delta_A > \gamma_B\ y\ \gamma_A < \delta_B \\ 0 & en\ otro\ caso\ (\gamma_A \geq \delta_B) \end{cases} \quad (7.10)$$

7.4.2. Comparadores para NOAFT

Los pares de datos del tipo NOAFT, al no poseer una relación de orden entre ellos, sólo disponen de un comparador, el comparador de semejanza.

Dicho operador hace uso de la relación de semejanza definida por el usuario entre los elementos del dominio escalar, así como de los grados de posibilidad de dichos elementos en las distribuciones de posibilidad, para el cálculo del grado de similitud entre cada par de elementos.

La definición de este comparador se indica a continuación.

Definición 7.3 (Comparador de semejanza para tipos OAFT)**.** *Sean A y B dos distribuciones de posibilidad definidas sobre el dominio escalar U, siendo $\mu_A(x)$ y $\mu_B(y)$ las funciones características que definen respectivamente dichas distribuciones. Sea $S(x,y)$ la función característica que define la relación de semejanza sobre los elementos del dominio. Se define el comparador de similitud FEQ como se indica en la Ecuación 7.11.*

$$FEQ(A,B) = \sup_{x \in U, y \in U} \{S(x,y) \cdot \mu_A(x) \cdot \mu_B(x)\} \quad (7.11)$$

7.4.3. Comparadores para CFC

La propuesta considera, para la comparación de las colecciones difusas conjuntivas, los operadores de inclusión y el comparador de semejanza que serán analizados a continuación.

7.4.3.1. Operador de inclusión

Esta extensión propone el operador de inclusión que permitirá el cálculo del grado en que una colección difusa conjuntiva está incluida en otra.

El cálculo del grado de inclusión de dos conjuntos difusos [128] puede ser calculado como muestra la Ecuación 7.12.

Definición 7.4. *Sean A y B dos conjuntos difusos definidos sobre el universo U, y $\mu_A(x)$ y $\mu_B(x)$ sus funciones características. Sea $I(x,y)$ un operador de implicación. El grado de inclusión de A en B, $A \subseteq B$, se calcula como sigue:*

$$N(B|A) = \min_{u \in U}\{I(\mu_A(x), \mu_B(x))\} \tag{7.12}$$

La anterior propuesta no contempla que los elementos de los conjuntos difusos sean a su vez datos imperfectos, por lo que no se adecúa totalmente a los requisitos del tipo de dato CFC, ya que este puede contener tanto elementos de tipo *crisp* como imperfecto.

Para solventar el anterior problema, se puede emplear el proceso de cálculo del *Grado de Inclusión Guiado por Semejanza* entre conjuntos difusos, propuesto en [99]. Éste se define como se indica a continuación.

Definición 7.5 (Grado de inclusión guiado por semejanza)**.** *Sean A y B dos conjuntos difusos definidos, sobre el universo finito de referencia U, por sus funciones características $\mu_A(x)$ y $\mu_B(y)$. Sea S la relación de semejanza definida sobre los elementos del dominio y $\mu_S(x,y)$ su función característica. Sea $\otimes$ una t-norma e I un operador de implicación. El grado de inclusión guiado por semejanza de A en B se calcula como se indica en la Ecuación 7.13, donde $\theta_{A,B,S}(x,y)$ se define según la Ecuación 7.14.*

$$\Theta_S(B|A) = \min_{x \in U} \max_{y \in U} \theta_{A,B,S}(x,y) \tag{7.13}$$

$$\theta_{A,B,S}(x,y) = \otimes(I(\mu_A(x), \mu_B(y)), \mu_S(x,y)) \tag{7.14}$$

Esta función es la propuesta empleada para la implementación del operador de inclusión sobre colecciones del tipo CFC. Para ello, se definirá la relación de similitud entre dos elementos de la colección como se muestra en la Ecuación 7.15.

$$S(x,y) = \begin{cases} FEQ(x,y) & \text{si } x \text{ e } y \text{ son de un tipo imperfecto} \\ 1 & \text{si } x = y \text{ y } x \text{ e } y \text{ son de un tipo } \textit{crisp} \\ 0 & \text{si } x \neq y \text{ y } x \text{ e } y \text{ son de un tipo } \textit{crisp} \end{cases} \tag{7.15}$$

7.4.3.2. Comparador de semejanza

El operador de semejanza entre colecciones difusas conjuntivas calculará el grado en que dos conjuntos de tipo CFC son similares.

La igualdad entre conjuntos clásicos se puede definir empleando el concepto de la doble inclusión, mostrado en la Ecuación 7.16.

$$A = B \text{ si, y solo si, } (A \subseteq B) \wedge (B \subseteq A) \tag{7.16}$$

Cuando estos conjuntos son conjuntos difusos se propone el empleo, para el cálculo del grado de semejanza entre colecciones del tipo CFC, del *Grado de Semejanza Generalizado entre Conjuntos difusos* propuesto en [99]. Este grado de semejanza está basado en el concepto de la doble inclusión y emplea como estimador del grado de inclusión entre conjuntos difusos el propuesto anteriormente en la Ecuación 7.13.

El procedimiento para el cálculo de este grado de semejanza se define a continuación.

Definición 7.6 (Grado de semejanza generalizado entre conjuntos difusos)**.** *Sean A y B dos conjuntos difusos definidos sobre un dominio de referencia finito U, sobre el que se ha definido una relación de semejanza S entre cada par de sus elementos (como muestra la Ecuación 7.15), y $\otimes$ una t-norma. El grado de semejanza generalizado entre dos conjuntos difusos A y B restringido por $\otimes$ se calcula como se indica en la Ecuación 7.17.*

$$\beth_{S,\otimes}(A,B) = \otimes(\Theta_S(B|A), \Theta_S(A|B)) \tag{7.17}$$

7.4.4. Comparadores para DFC

Las colecciones del tipo DFC son colecciones en las que, debido a su semántica asociada, sólo uno de sus elementos es el valor que se aplica al atributo. Por tanto, no cabe calcular en ellas las operaciones tradicionales sobre colecciones con semántica conjuntiva (como, por ejemplo, el grado de inclusión), ya que en realidad la colección sólo puede tomar el valor de uno de sus elementos.

La única operación con sentido sobre este tipo de colecciones es el comparador de semejanza, el operador FEQ. Este operador comparará cada par de elementos (uno de la primera colección y otro de la segunda colección) y obtendrá el grado de similitud de dichos elementos, que será modulado por el grado de participación de los elementos en la colección. Estos grados de semejanza modulados serán agregados usando la función máximo (empleada como t-conorma) ya que este agregador es el más adecuado para la semántica disyuntiva (semántica OR). La definición del operador FEQ descrito anteriormente se indica a continuación.

Definición 7.7. *Sean A y B dos conjuntos difusos, con semántica disyuntiva, definidos sobre el dominio referencia U y cuyas funciones características son $\mu_A(x)$ y $\mu_B(y)$ respectivamente. Sea $S(x,y)$ una relación de similitud definida entre los elementos del dominio, similar a la que ha sido definida en la Ecuación 7.15. El grado de semejanza entre los dos conjuntos difusos disyuntivos A y B se calcula como se muestra a continuación:*

$$FEQ(A,B) = \max_{x \in U, y \in U}(\mu_A(x) \cdot \mu_B(y) \cdot S(x,y)) \tag{7.18}$$

7.4.5. Comparadores para FO

Los objetos difusos definidos por el usuario sólo dispondrán de un comparador entre ellos, el comparador de similitud entre objetos difusos de una misma clase. Éste calculará el grado de semejanza entre dicho par de objetos, en función de la similitud de los valores de los atributos de los objetos y el grado de importancia de dichos atributos.

El valor de similitud para los valores de los atributos se obtendrá comparando el valor que presenta un determinado atributo en el primer objeto con el valor presente en el otro objeto para dicho atributo. Esta comparación se realiza para cada atributo definido en la clase a la que pertenecen ambos objetos

Para el cálculo del grado de similitud entre objetos se propone aplicar el *algoritmo de cálculo de semejanza entre objetos* definido en [55], basado en operador V_Q propuesto en [99].

Dicho operador se define como se indica a continuación.

Definición 7.8. *Sean A y B dos conjuntos difusos definidos sobre el dominio $X = \{a_1, a_2, \ldots, a_n\}$, donde $\{a_1, a_2, \ldots, a_n\}$ es el conjunto de atributos definidos en la clase C. Se define el operador V_Q como:*

$$V_Q(A|D) = \gamma_Q \max_{x \in X}(\mu_D(x) \wedge \mu_A(x)) + (1 - \gamma_Q) \min_{x \in X}(\mu_A(x) \vee (1 - \mu_D(x))) \quad (7.19)$$

donde:

- $\mu_D(a_i) = p_{a_i}$*, siendo* p_{a_i} *el grado de relevancia del atributo* a_i*.*
- $\mu_A(a_i) = S_{a_i}(o_1, o_2)$*, siendo* S_{a_i} *la relación de similitud definida entre los valores del atributo* a_i*, calculada como muestra la Ecuación 7.15.*
- γ_Q *un valor asociado al cuantificador* Q*, verificando* $\gamma_\exists = 1$*,* $\gamma_\forall = 0$*,* $\forall Q$*,* $Q' \in Q$*,* $Q \succeq Q' \Rightarrow \gamma_Q \geq \gamma_{Q'}$*.*

El citado algoritmo, basado en el anterior operador, se define a continuación.

Definición 7.9. *Sean o_1 y o_2 dos objetos difusos de la clase C, se define el grado de similitud entre dicho par de objetos de la siguiente forma:*

$$FEQ(C, o_1, o_2) = FEQ(C, o_1, o_2, \emptyset, \emptyset) \quad (7.20)$$

donde $FEQ(C, o_1, o_2, \Omega_{visited}, \Omega_{approx}) =$

1. *Si $o_1 = o_2$ (igualdad de identidad entre objetos), entonces:*
   $FEQ(C, o_1, o_2, \Omega_{visited}, \Omega_{approx}) = 1$
2. *Si o_1 y o_2 son conjuntos difusos conjuntivos, entonces:*
   $FEQ(C, o_1, o_2, \Omega_{visited}, \Omega_{approx}) = \beth_{S,\otimes}(A, B) = \otimes(\Theta_S(B|A), \Theta_S(A|B))$
3. *Si existe una relación de similitud S definida entre los objetos de la clase C (son tipos simples), entonces:*
   $FEQ(C, o_1, o_2, \Omega_{visited}, \Omega_{approx}) = S(o_1, o_2)$
4. *Si $\{o_1, o_2\} \notin \Omega_{visited}$, entonces:*
   $FEQ(C, o_1, o_2, \Omega_{visited}, \Omega_{approx}) = V_Q(W, R)$
   donde:
   - *W contiene la lista de grados de importancia de los atributos.*

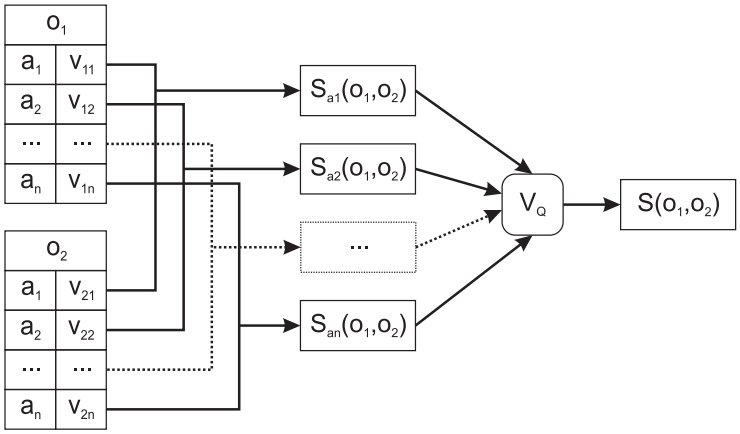

Figura 7.1: Mecanismo de comparación de objetos difusos complejos

- *R contiene la lista de los grados de semejanza de los valores de los atributos de los objetos, calculados de la forma:*
  $FEQ(C_{a_i}, o_1.a_i, o_2.a_i, \Omega_{visited} \cup \{\{o_1, 0_2\}\}, \Omega_{approx})$
  donde C_{a_i} *es la clase del atributo* a_i.

5. *Si* $\{o_1, o_2\} \in \Omega_{visited}$ *y* $\{o_1, o_2\} \notin \Omega_{approx}$*, entonces:*
   $FEQ(C, o_1, o_2, \Omega_{visited}, \Omega_{approx}) = FEQ(C, o_1, o_2, \emptyset, \Omega_{approx} \cup \{\{o_1, 0_2\}\})$

6. *En otro caso, cuando* $\{o_1, o_2\} \in \Omega_{visited}$ *y* $\{o_1, o_2\} \in \Omega_{approx}$*, el valor de* $FEQ(C, o_1, o_2, \Omega_{visited}, \Omega_{approx})$ *es indefinido.*

Esta formulación tan compleja describe un mecanismo relativamente simple que se lustra en la Figura 7.1. La figura muestra cómo se compraran los valores de los atributos, empleando el operador de similitud adecuado a la clase del atributo al que pertenecen, par a par (un valor del objeto o_1 y otro valor del objeto o_2). Una vez calculados, estos valores se agregan empleando el operador V_Q y, finalmente, el resultado de dicho operador corresponde con el valor buscado.

La única complicación que se puede dar en el procedimiento anterior es que el par de valores de un atributo que comparamos sean a su vez objetos complejos. En ese caso, se aplicará recursivamente el proceso de comparación y, en el caso de que se encuentren ciclos en las relaciones de los objetos que forman parte de la estructura compleja de los objetos, el valor de la comparación se declarará indefinido.

7.5. Jerarquía de tipos para datos difusos

Esta sección presenta la propuesta de una jerarquía de tipos definidos por el usuario diseñados para permitir el almacenamiento y manipulación de datos difusos. Dicha jerarquía de tipos se presenta, empleando el formato UML simplificado, en la Figura 7.2 y sus elementos se describen en las siguientes subsecciones.

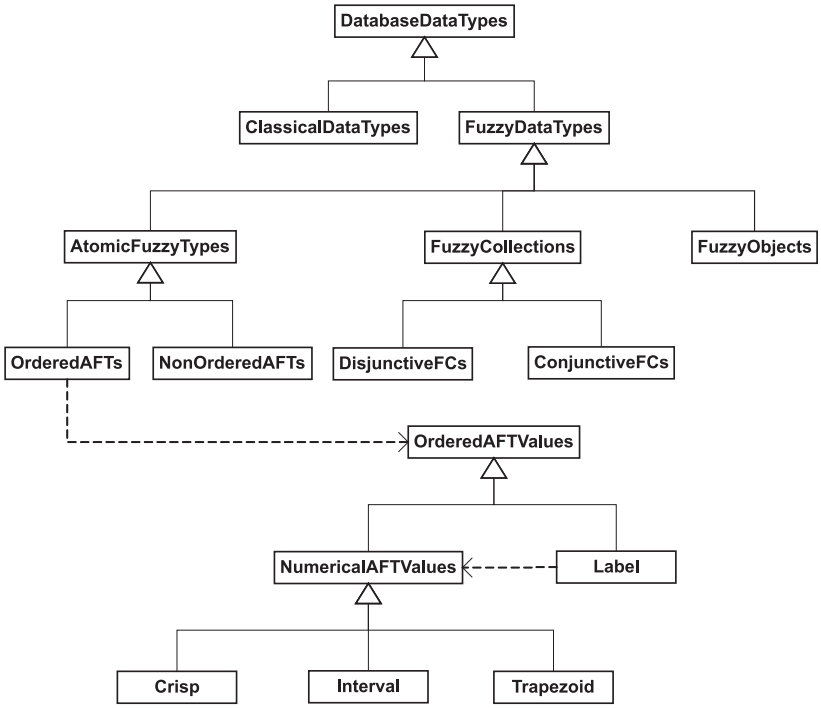

Figura 7.2: Jerarquía básica de tipos para la extensión difusa

7.5.1. Clases generales de la jerarquía

A continuación, procederemos a describir brevemente las clases mas generales de la jerarquía de tipos para datos difusos, diseñadas para ofrecer un marco en el que situar las clases de la extensión difusa propuesta.

7.5.1.1. Clase DatabaseDataTypes

Esta clase, abstracta por definición, representa de forma genérica los tipos de datos que puede manipular el sistema anfitrión. Ésta engloba tanto a los tipos de datos que dicho sistema anfitrión incluye por defecto, como los diferentes tipos que han sido definidos por el usuario.

7.5.1.2. Clase ClassicalDataTypes

La presente clase abstracta engloba, en forma de representación abstracta, los tipos de datos incluidos por defecto en el sistema anfitrión.

7.5.1.3. Clase FuzyDataTypes

Esta clase es una abstracción de todos los datos difusos soportados por la jerarquía de tipos propuesta. La clase puede ser considerada como una interfaz, la cual, declara métodos comunes de carácter general que deberán ser implementados por los subtipos. Su especificación se muestra con mayor detalle en la Figura 7.3.

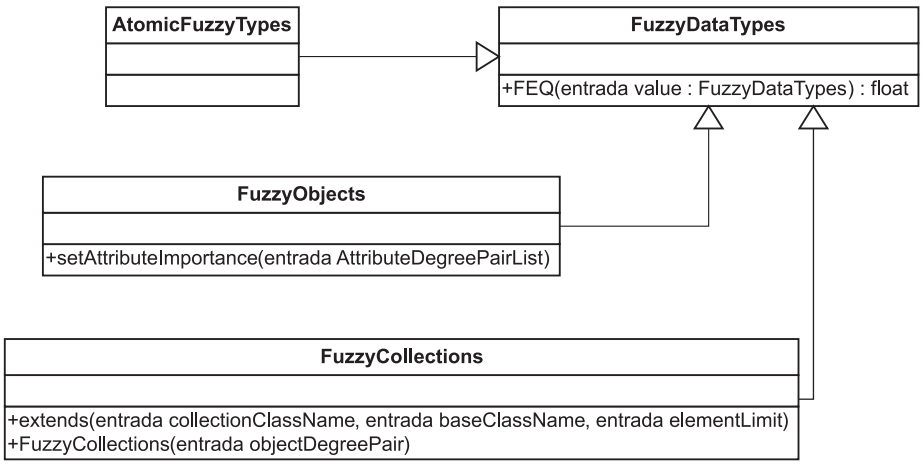

Figura 7.3: Detalle de las clases generales de la jerarquía

El único método definido en esta clase es el método `FEQ(value)`. Este método es un método abstracto que obligará a todas las subclases a implementar un método para el cálculo de la igualdad difusa. Éste devuelve un valor, en el intervalo $[0, 1]$, indicando el grado de igualdad calculado. En definitiva, todos los tipos de datos contemplados deberán de poseer un mecanismo para el cálculo de semejanza entre sus valores.

7.5.2. Clases para la representación de datos atómicos difusos

Esta sección está dedicada a la descripción de las clases que modelan parte de la jerarquía diseñada para el almacenamiento y manipulación de datos atómicos difusos.

7.5.2.1. Clase AtomicFuzzyTypes

La presente clase abstracta agrupa e implementa el comportamiento común a las clases de la jerarquía que permiten almacenar y manipular datos difusos atómicos, esto es, extensiones difusas de datos atómicos escalares y datos numéricos. Los detalles de estas clases se muestran en la Figura 7.4.

7.5.2.2. Clase OrderedAFTs

La clase abstracta OrderedAFTs proporciona la estructura y el comportamiento para el almacenamiento y manipulación de datos atómicos representados por una distribución de posibilidad definida sobre un domino ordenado.

La clase posee una serie de métodos para implementar los operadores propios que pueden ser aplicados sobre los números difusos que dicha clase modela. Estos operadores, en su versión de posibilidad, son los siguientes:

- `FGT(value)`: Este método implementa el operador *mayor que* difuso, aplicado entre el valor que representa el objeto que ejecuta el método y el

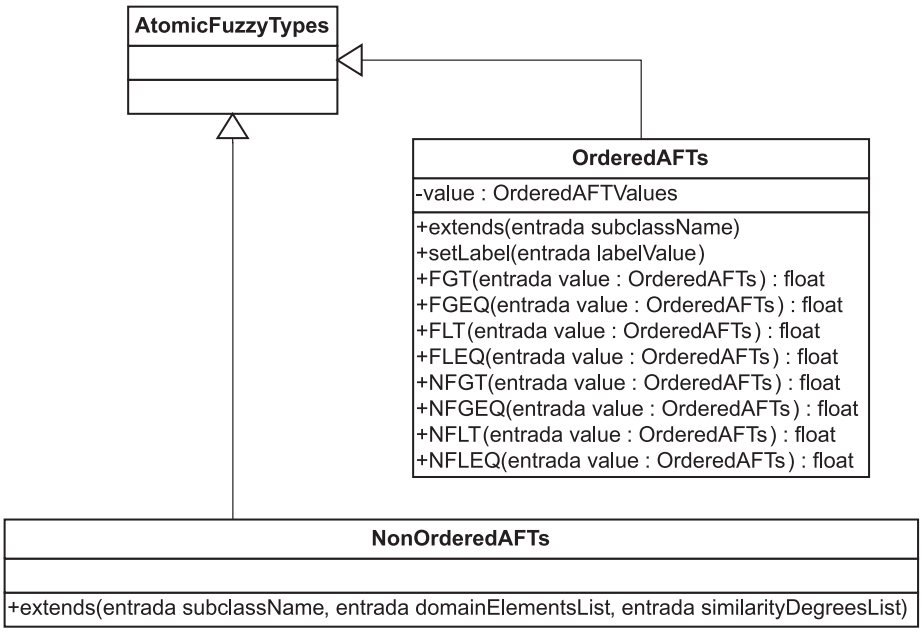

Figura 7.4: Detalle de las clases para representar datos atómicos difusos

valor representado por el objeto `value`. En definitiva, este método devuelve el valor resultante de la expresión $FGT(this, value)$, siendo *this* la distribución de posibilidad asociada al valor representado el objeto que ejecuta en el método, y *value* la distribución de posibilidad asociada al valor representado por el objeto `value` que se pasa como argumento.

- `FGEQ(value)`: El método contiene la implementación del operador *mayor o igual que* difuso, aplicado entre el valor representado por el objeto `value` y el que representa el objeto que ejecuta el método. Por tanto, este método devuelve el valor que resulta al aplicar la expresión $FGEQ(this, value)$.
- `FLT(value)`: Este método implementa el operador *menor que* difuso, que se aplica entre el valor representado por el objeto que ejecuta el método y el valor que representa el objeto `value`. El valor devuelto es el resultado de aplicar la expresión $FLT(this, value)$.
- `FLEQ(value)`: El método alberga la implementación del operador *menor o igual que* difuso. Este operador será aplicado entre el valor que representa el objeto `value` y el que representa el valor que ejecuta el método. El valor de retorno de este método es el resultante de la aplicación de la expresión $FLEQ(this, value)$.

Estos mismos operadores, pero en su versión de necesidad, son los siguientes:

- `NFGT(value)`: Este método que devuelve el valor resultante de la expresión $NFGT(this, value)$.
- `NFGEQ(value)`: Este método devuelve el valor obtenido de la aplicación de $NFGEQ(this, value)$.

- `NFLT(value)`: El método devuelve el valor resultante de la aplicación de la expresión $NFLT(this, value)$.

- `NFLEQ(value)`: El presente método ofrece como valor de retorno el resultante de la expresión $NFLEQ(this, value)$.

Además de los métodos descritos anteriormente, esta clase posee el método estático `extends(SubclassName)`. Dicho método permite al usuario la creación de forma automática de subclases, cuyo nombre se especifica en el parámetro `SubclasName`. Estas subclases serán las que emplearán los usuarios para definir los tipos de los atributos en los que se almacenarán los valores atómicos difusos que deseen emplear en su base de datos.

El atributo `value` es el encargado de almacenar el número difuso que albergan las instancias de dicha clase. Esta clase está diseñada para almacenar cualquier tipo de valor atómico numérico difuso o etiqueta lingüística, como veremos a continuación en la descripción de las clases empleadas para representar este tipo de valores, mostradas en mayor detalle la Figura 7.5 y analizadas en los siguientes apartados.

Los usuarios obtendrán, mediante el empleo del método `extends`, subclases parametrizadas de la clase OrderedAFT, similares a la mostrada en la Figura 7.6. Dichas subclases serán las que los usuarios emplearán para modelar los dominios de este tipo que empleen en sus bases de datos.

7.5.2.3. Clase OrderedAFTValues

Esta clase agrupa, de forma abstracta, los valores que pueden ser empleados en los tipos modelados por la clase OAFT. Estos valores pueden ser distribuciones trapezoidales de posibilidad, representadas de forma abstracta por la subclase NumericalAFTValues, o etiquetas lingüísticas, modeladas por instancias la clase Label, que representan dichas distribuciones de posibilidad.

Esta clase incluye un único método, el método `getValue()`. Este método es abstracto, de tal forma que se obligue a sus subclases a que implementen la lógica necesaria para devolver el valor de la clase NumericalAFTValues que representan.

La razón de ser de este método no es otra que permitir tratar transparentemente tanto valores numéricos como etiquetas lingüísticas. En el caso de objetos pertenecientes a las subclases de NumericalAFTValues, el método devolverá una referencia a ellos mismos y, en el caso de objetos de la clase Label (que modela las etiquetas lingüísticas), se devolverá el objeto de la clase NumericalAFTValues que representa la distribución de posibilidad a la cual la etiqueta está asociada.

7.5.2.4. Clase NumericalAFTValues

La clase abstracta NumericalAFTValues posee, en el modelo propuesto, un conjunto de subclases que representan los posibles tipos de valores, en forma de casos concretos o generales, de distribuciones trapezoidales de posibilidad que la extensión propuesta considera inicialmente.

Éstas, junto con los atributos que emplean para almacenar los valores imprecisos que modelan, son las siguientes:

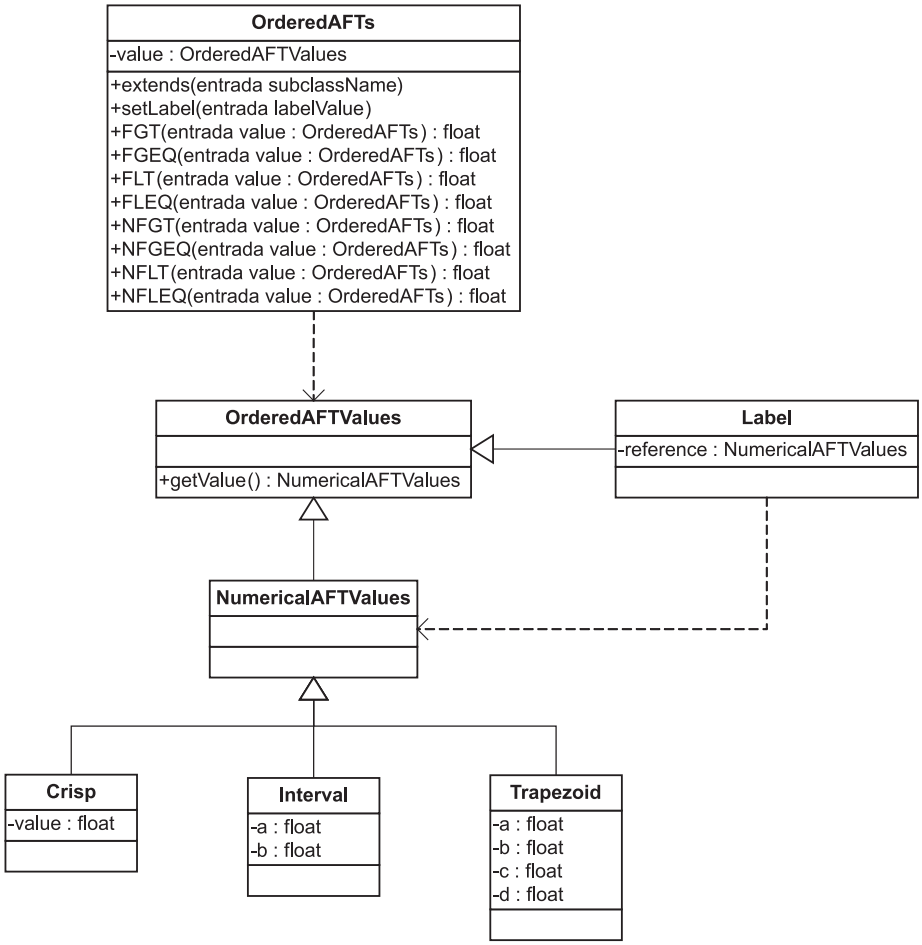

Figura 7.5: Detalle de las clases para la representación valores numéricos difusos

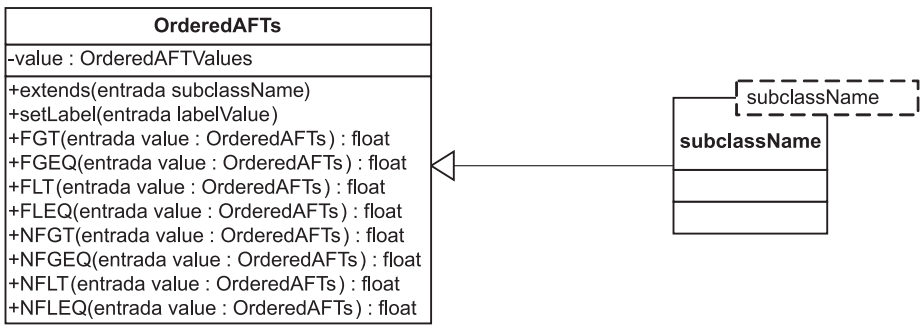

Figura 7.6: Subclases parametrizadas de la clase OrderedAFTs

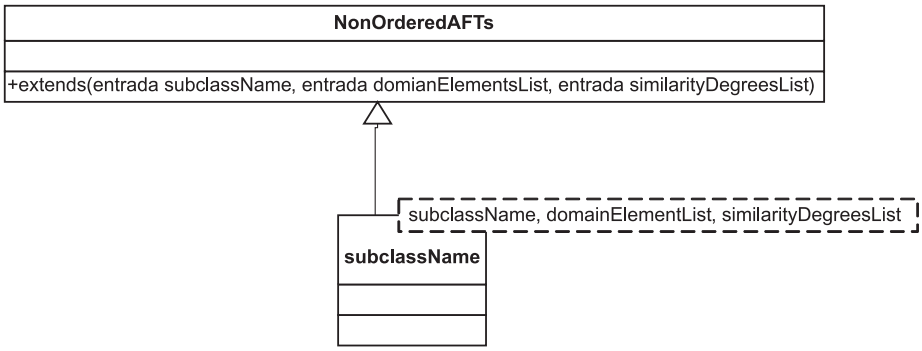

Figura 7.7: Subclases parametrizadas de la clase NonOrderedAFTs

- Clase Crisp: Las instancias de esta clase almacenan datos numéricos precisos. Esta clase posee un único atributo, llamado `value`, que se emplea para almacenar el valor preciso que representan los objetos de la clase.
- Clase Interval: Las instancias de esta clase almacenan rangos intervalares de valores. La clase posee dos atributos, llamados `a` y `b`, que son los necesarios para almacenar los límites del intervalo que representan los objetos pertenecientes a la clase.
- Clase Trapezoid: Las instancias de esta clase almacenan distribuciones de posibilidad trapezoidales generales. La presente clase posee cuatro atributos llamados `a`, `b`, `c` y `d`. Éstos son empleados para la representación de la distribución de posibilidad trapezoidal $[a, b, c, d]$ que representan las instancias de dicha clase.

Inicialmente el modelo de clases propuesto contempla estos tres tipos de valores atómicos numéricos difuso. No obstante, el diseño de dicho modelo hace muy sencilla su extensión para incluir nuevos tipos. Para ello, sólo habrá que especificar una nueva clase que herede de la clase NumericalAFTValues. Ésta proporciona la estructura y el comportamiento adecuado para el almacenamiento y manipulación del nuevo tipo de dato que se incorpora.

7.5.2.5. Clase NonOrderedAFTs

Esta clase abstracta proporciona la estructura y el comportamiento necesario para la manipulación y almacenamiento de datos difusos definidos sobre dominios escalares sin una relación de orden entre sus miembros.

Los usuarios de este tipo de datos deberán definir una relación de similitud entre los elementos del dominio, de tal forma que el operador FEQ pueda obtener el grado de similitud entre dos elementos de éste.

Como en el caso de la clase OrderedAFTs, los usuarios no harán uso directo de esta clase en su base de datos. Para modelar los dominios de este tipo, los usuarios crearán subclases parametrizadas de ésta. Para ello, emplearán el método estático `extends(...)`, cuya ejecución creará automáticamente dichas clases. Este tipo de clases parametrizadas se ilustran en la Figura 7.7.

Los argumentos del método `extends(...)`, para el caso de la clase NonOrderedAFTs, son los siguientes:

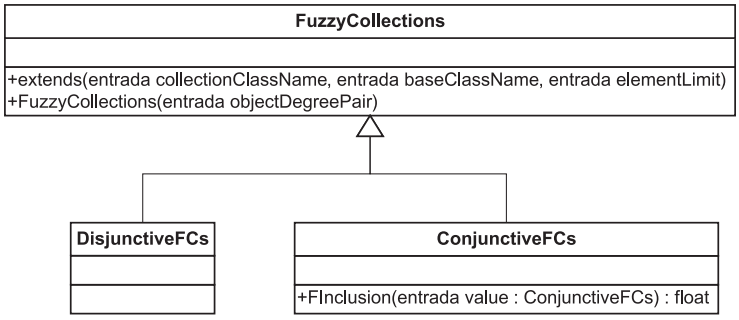

Figura 7.8: Detalle de las clases para la representación de colecciones difusas

- `subclassName`: Este argumento indica el nombre de la subclase que será creada por la llamada del método.
- `domainElementsList`: El argumento es una lista de los elementos del dominio escalar no ordenado que se desea asociar al subtipo que se crea.
- `similarityDegressList`: Este argumento es una lista en la que se incluyen, de forma ordenada, los grados de similitud entre cada par de elementos del dominio.

7.5.3. Clases para la representación de colecciones difusas

A continuación, procederemos a la descripción de las clases que conforman el subconjunto de clases del modelo propuesto, orientadas al almacenamiento y manipulación de colecciones difusas. La estructura de estas clases está detallada en la Figura 7.8.

7.5.3.1. Clase FuzzyCollections

Esta clase es la abstracción que engloba, proveyendo estructura y comportamiento común, las diferentes clases que permiten el almacenamiento de colecciones difusas.

Las colecciones difusas son una generalización borrosa del concepto de colección definido en SQL:1999. Esto implica que los objetos que forman parte de la colección pueden pertenecer a ésta en cierto grado, dentro del rango $[0, 1]$, al igual que ocurre con los elementos de un conjunto difuso.

Esta clase tiene asociados los siguientes métodos:

- `extends(collectionClassName,baseClassName,elementLimit)`: Éste es un método abstracto definido para obligar a las subclases de la clase FuzzyCollections a implementar un mecanismo para la creación automática de subclases de la misma. Estas subclases serán las empleadas por los usuarios para la modelización de las colecciones difusas que utilicen en sus bases de datos. Dichas subclases se generan en función de los valores de los siguientes parámetros del método:
  - `collectionClassName`: El valor de este parámetro es el nombre de la subclase que será creada por el método.

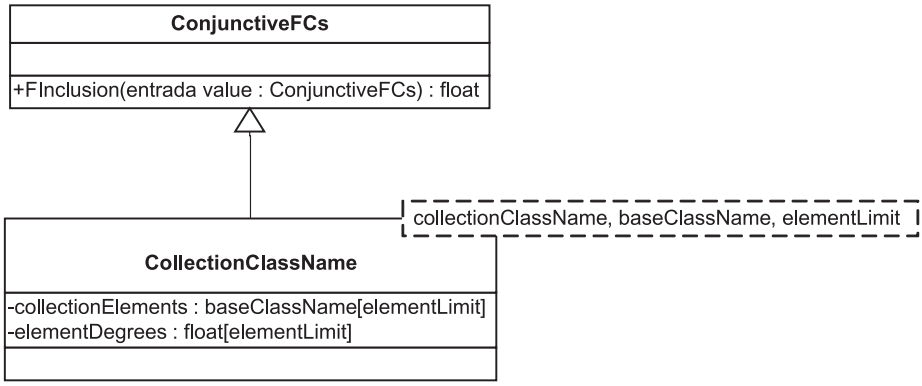

Figura 7.9: Subclases parametrizadas de la clase ConjunctiveFCs

- `baseClassName`: Este parámetro indica el nombre de la clase a la que pertenecerán los elementos de la colección que se está definiendo.
- `elementLimit`: El valor de este parámetro numérico indica el número máximo de elementos que podrá contener la colección.

- `FuzzyCollection(ObjectDegreePair)`: Este método abstracto es la especificación del constructor que deberán implementar, como mínimo, las subclases de las clases ConjunctiveFCs y DisjunctiveFCs. La implementación de este constructor ofrecerá una forma sencilla para inicializar la colección, en base a la lista de pares objeto-grado.

7.5.3.2. Clase ConjunctiveFCs

La clase abstracta ConjunctiveFCs proporciona la estructura necesaria para el almacenamiento de colecciones de objetos de una determinada clase y el comportamiento necesario para manipular los mismos.

La característica distintiva de esta clase es que modela colecciones de objetos con *semántica conjuntiva*, es decir, todos los objetos incluidos en la colección pertenecen a la misma en un determinado grado, lo que permite representar atributos con valores múltiples.

Esta clase posee un único método, el método `FInclusion(value)`. Éste calcula el grado de inclusión de la colección representada por el objeto `value` en la colección que representa el objeto sobre el que se invoca el método.

Los usuarios no emplearán la clase ConjunctiveFCs directamente. En lugar de ello, utilizarán subclases parametrizadas de ésta adaptadas al dominio subyacente que el usuario desee emplear. Para ello, se invocará método `extends(...)` heredado de la clase FuzzyCollection, que generará una nueva clase en función de los parámetros suministrados. La Figura 7.9 muestra un ejemplo de este tipo de clases.

7.5.3.3. Clase DisjunctiveFCs

La estructura y comportamiento de la clase *DisjunctiveFCs* están diseñados para el almacenamiento y manipulación de colecciones de objetos con *semántica disyuntiva*.

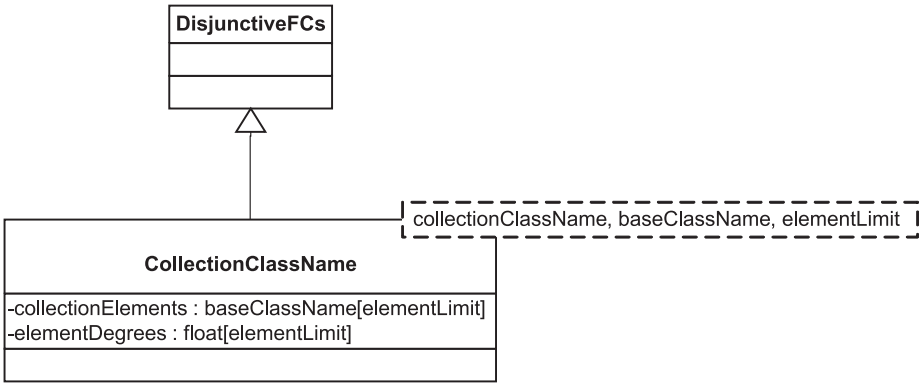

Figura 7.10: Subclases parametrizadas de la clase DisjunctiveFCs

Esta distinción semántica, hace que los elementos de la colección se vean como valores posibles, en un determinado grado, pudiéndose dar sólo uno de ellos. En definitiva, esta semántica es similar a la relacionada con datos imprecisos sobre un domino escalar no ordenado, modelados por las clase NonOrderedAFTs, con la salvedad de que los valores, en éste caso, pueden ser objetos complejos.

Al igual que ocurre con la clase ConjunctiveFCs, los usuarios no harán uso directo de la clase DisjunctiveFCs en su base de datos. En lugar de ello, emplearán subclases propias de la misma que serán parametrizadas adecuadamente para trabajar con los elementos de la clase seleccionada por el usuario. Un ejemplo de este tipo de clases parametrizadas puede verse en la Figura 7.10.

7.5.4. Clase para la representación de objetos difusos

La presente sección analiza la clase empleada como marco, a partir del cual, crear clases definidas por el usuario que modelen sus propios objetos difusos.

7.5.4.1. Clase FuzzyObjects

Esta clase abstracta es la encargada de proveer la estructura básica, así como el comportamiento necesario, para proporcionar las capacidades necesarias para el almacenamiento y manipulación de objetos difusos a las subclases que modelen los objetos difusos definidos por el usuario.

Los usuarios podrán crear sus propias subclases a partir de ésta, heredando directamente de ella. Esto permite a los usuarios definir tanto los atributos como los métodos que necesiten para diseñar adecuadamente sus clases. Un ejemplo de lo anterior se muestra en la Figura 7.11.

Gracias a la herencia, las clases de usuario heredarán la implementación, hecha en la clase FuzzyObjects, del método FEQ, que permitirá comparar entre sí objetos difusos de la misma clase.

7.6. Descripción de la implementación

La presente sección describirá el prototipo de la extensión para el tratamiento de datos imperfectos, introducida anteriormente, que ha sido implementado

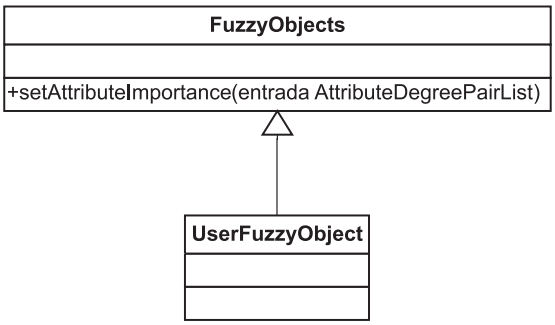

Figura 7.11: Subclases de usuario de la clase FuzzyObjects

como parte del presente trabajo.

Para ello, se describirá el sistema anfitrión sobre el que ha sido implementado dicho prototipo. Seguidamente, se detallaran los tipos de datos imperfectos a los que se da soporte en la implementación, las clases que modelan los mismos y los operadores definidos para la manipulación de este tipo de datos. Finalmente, se ilustrará la funcionalidad del prototipo mediante la combinación, en un ejemplo, de los elementos descritos.

7.6.1. Sistema anfitrión

El sistema anfitrión elegido para la implementación del prototipo, que introduce el presente capítulo, es Oracle. Como se indicó en el Capítulo 4, desde nuestro punto de vista, este SGBD es el más indicado para la implementación de dicho prototipo, debido a las capacidades de extensión a las que da soporte el sistema. Estas características son las siguientes:

- Tipos de Datos Definidos por el Usuario, con los que se podrán crear las clases que han sido definidas en el capítulo anterior.
- Operadores definidos por el usuario, que permiten definir los operadores con los que el usuario manipulará los tipos con los que se representan los tipos difusos.
- Extensibilidad de los sistemas de indexado y optimización de consultas, lo que permitirá incorporar al prototipo mejoras en el rendimiento del procesamiento de consultas flexibles, en forma de índices adaptados a la naturaleza imperfecta de los datos.
- Agrupamiento, de los elementos anteriores, en módulos que se integran fácilmente en el sistema y lo dotan de funcionalidad para el tratamiento de un tipo específico de datos. En el caso de Oracle estos módulos se denominan *Data Cartridges*.

Además de las razones anteriores, hay que tener en cuenta que Oracle es uno de los líderes indiscutibles en el mercado de SGBD, lo que nos asegura un alto rendimiento, un gran parque de instalaciones en las que se podrá emplear el prototipo implementado y gran variedad de capacidades de valor añadido (como, por ejemplo, pueden ser paquetes para el tratamiento de tipos de datos específicos, capacidades de distribución de bases de datos, de clustering, etc.).

7.6.2. Tipos de datos imperfectos

La presente sección describe los UDTs incluidos en el prototipo de la extensión difusa introducida en secciones anteriores.

Para ello, enumeraremos las clases pertenecientes a dicho prototipo encargadas de implementar los *tipos de datos* imprecisos, que coinciden con las clases *hoja* de la jerarquía propuesta en las secciones anteriores.

7.6.2.1. Clase OrderedAFTs

Esta clase proporciona la estructura y comportamiento adecuados para permitir al sistema anfitrión el almacenamiento y manipulación de datos imprecisos atómicos numéricos, denominados en el anterior capítulo como datos de tipo OAFT. Los datos del citado tipo serán representados en forma de distribuciones de posibilidad trapezoidales sobre el dominio numérico proporcionado por el sistema anfitrión.

Las diferentes clases de distribuciones de posibilidad consideradas en la propuesta son modeladas por las siguientes subclases de la clase OrderedAFTValues:

- Crisp: Esta clase permite representar valores clásicos. Su constructor, `Crisp(value)`, acepta un único argumento numérico, que corresponde con el valor que se desea representar.

- Interval: La clase está diseñada para permitir la representación de valores intervalares. El constructor de que dispone, `Interval(a,b)`, acepta los argumentos numéricos `a` y `b`, permitiendo crear una instancia de la clase que represente el intervalo $[a, b]$.

- Trapezoid: Esta clase permitirá representar distribuciones de posibilidad trapezoidales genéricas. Su constructor, `Trapezoid(a,b,c,d)`, acepta cuatro valores numéricos, lo que permite crear instancias de la clase representando distribuciones trapezoidales de la forma $[a, b, c, d]$.

- Label: Esta clase permite utilizar como valor representado el valor asociado a una etiqueta lingüística. El constructor de esta clase, `Label(labelName)`, crea una instancia de la clase que representa la etiqueta lingüística indicada en el argumento `labelName`.

  Los objetos de esta clase proporcionan al usuario el método `getValue()`, cuya ejecución devuelve la distribución de posibilidad, representada por el objeto de la clase NumerivalAFTValues asociado a dicha etiqueta lingüística.

La clase OrderedAFTs es una clase abstracta, por lo que no puede ser empleada directamente por parte de los usuarios para caracterizar los tipos de datos imprecisos que deseen contener en sus base de datos. Para poder hacer uso de este tipo de datos, el usuario deberá de emplear una subclase de OrderedAFTs definida por éste. Este requisito se impone, entre otros motivos, para permitir al usuario definir sus propias etiquetas lingüísticas, asociadas éstas al dominio que modela la subclase en cuestión.

Teniendo en cuenta lo anterior, la principal funcionalidad que ofrece al usuario esta clase es el método de clase `extends(typeName)`. Dicho método proporciona al usuario un mecanismo para la creación automática de subclases de la clase OrderedAFTs, cuyo nombre corresponde con el argumento `typeName`. Así, el usuario dispondrá, de forma sencilla, de todas las subclases que necesite para modelar adecuadamente los dominios de este tipo de datos presentes en su base de datos.

Adicionalmente, las subclases de OrderedAFTs proporcionan, de cara al usuario, los siguientes métodos estáticos para definir y obtener etiquetas lingüísticas sobre el dominio modelado por dicha subclase:

- `labelDef(label,value)`: Este método permite al usuario definir etiquetas lingüísticas asociadas a la clase. Para ello, el argumento `label` será empleado para la especificación de la etiqueta que se desea definir y el argumento `value`, que será un objeto de la clase NumericalAFTValues, permitirá indicar el valor asociado a la etiqueta.
- `getLabel(label)`: El presente método permite la obtención del objeto de la clase Label que representa la etiqueta lingüística identificada mediante la cadena de caracteres que el usuario pasa como valor del argumento `label`.

7.6.2.2. Clase ConjunctiveFCs

La clase proporciona la estructura y comportamiento necesarios para el almacenamiento y manipulación de colecciones difusas con semántica conjuntiva, es decir, colecciones de tipo CFC. Las colecciones difusas pondrán contener elementos de cualquier tipo soportado en la base de datos, ya sea *crisp* o imperfecto, siendo siempre dichos elementos objetos de la misma clase.

Para que la citada clase soporte cualquier tipo de elementos, ésta se ha definido como abstracta y, por tanto, no puede ser usada directamente por parte de los usuarios. Para el uso de este tipo de colecciones, es necesario que los usuarios creen subclases parametrizadas de la clase CFC. Estos parámetros servirán para determinar el tipo de elementos que contendrá la clase, así como su tamaño máximo.

Para el propósito anterior, la clase ofrece un método abstracto llamado `extends(typeName,elementTypeName,size)` que, al ser ejecutado, crea automáticamente una subclase parametrizada de la clase CFC. Los valores de los parámetros de este método sirven para especificar el nombre de la subclase parametrizada que se desea crear (parámetro `typeName`), para indicar la clase a la que pertenecen los elementos que manejará la colección (parámetro `elementTypeName`) y el tamaño máximo de la colección (parámetro `size`).

Además del método anterior, los objetos de las subclases parametrizadas por el usuario ofrecen, de cara a éste, los siguientes métodos para el manejo de la colección:

- `getElement(index)`: Este método devuelve el elemento de la colección cuyo índice se corresponde con el valor del argumento `index`.
- `getElementMembership(index)`: El método devuelve el grado de pertenencia del elemento de la colección cuyo índice se corresponde con el valor del argumento `index`.

- `getSize()`: Este método devuelve el número de elementos que forman parte de la colección.

- `getMaximumSize()`: El método devuelve el número máximo de elementos que puede contener la colección.

7.6.2.3. Clase FO

Esta clase proporciona la estructura y comportamiento para permitir el almacenamiento y manipulación de objetos difusos, tal y como se han definido en secciones anteriores. La clase FO se corresponde con los tipos de datos FO definidos en las citadas secciones.

Los usuarios definirán sus propias clases difusas heredando de la clase FO. De esta forma, los objetos difusos, pertenecientes a clases difusas definidas por el usuario dispondrán de la estructura y comportamientos generales de los objetos difusos y, a su vez, podrán ser modelados para incluir estructura y comportamiento adicional según las necesidades del usuario.

El método `FEQ(fuzzyObject)` de la clase FO calcula el grado de semejanza entre el objeto sobre el que se ejecuta el método y un segundo objeto que se corresponde con el valor del argumento `fuzzyObject`. En caso de que las subclases de FO no aporten una implementación concreta, los objetos de éstas heredarán de la clase FO una implementación por defecto. Ésta devolverá el resultado de aplicar el algoritmo de cálculo de semejanza de objetos difusos, descrito en secciones anteriores, asumiendo el valor 1 como peso para todos los atributos de los objetos. Finalmente, téngase en cuenta que este método no será empleado directamente por el usuario, ya que el operador `FEQ`, descrito posteriormente, se encargará de hacer la llamada a este método.

7.6.3. Operadores sobre valores imperfectos

La presenten sección describirá los operadores ofrecidos por la extensión para la comparación de los tipos de datos difusos que ésta soporta.

7.6.3.1. Operador FEQ

Este operador, en su formato de dos argumentos, `FEQ(a,b)`, devuelve el grado de semejanza entre el valor imperfecto `a` y el valor imperfecto `b`.

Se ha de destacar que el operador siempre opera con pares de valores del mismo tipo. Esto quiere decir que `a` y `b` deben ser objetos de la misma clase o, al menos, tener un ancestro común (por ejemplo, el caso de objetos de subclases de la clase OrderedAFTs).

El papel que cumple el operador es sencillo, simplemente ofrece el valor devuelto por la llamada `a.FEQ(b)`, empleando el método `FEQ` implementado en todos los tipos de datos soportados en el prototipo.

La forma en que se calculará el grado de semejanza entre los operandos variará en función de la implementación asociada al tipo de los mismos. De esta forma, es posible invocar, en función del tipo de los operandos, el procedimiento adecuado para el cálculo de grado de semejanza empleando un mismo operador.

7.6.3.2. Operador NFEQ

En el caso de los datos de tipo OAFT, además del operador `FEQ`, se dispone de una variante de éste que aplica una medida de necesidad. Para ello, se invocará el operador `NFEQ(a,b)`, donde `a` y `b` son los datos de tipo OAFT que se desean comparar.

7.6.3.3. Operadores relacionales difusos

Para los pares de datos de tipo OAFT, se ha definido una serie de operadores que se corresponden con los *operadores relacionales difusos* definidos, anteriormente, en la subsección 7.4.1. La signatura de estos operadores es de la forma `OP(a,b)`, donde `OP` representa el nombre del comparador que se desea emplear y los argumentos, `a` y `b`, se corresponden con los dos valores de tipo OAFT a comparar. Los nombres de estos comparadores coinciden con los indicados en la Tabla 7.2.

7.6.3.4. Operador FInclusion

Este operador, `FInclusion(a,b)`, puede ser aplicado sobre pares de objetos que representen colecciones difusas conjuntivas, es decir, colecciones del tipo CFC.

Este operador devuelve el resultado ofrecido por la llamada `a.FInclusion(b)`, que calcula el grado de inclusión de la colección difusa conjuntiva `b` en la colección difusa conjuntiva `a`.

El método `FInclusion(value)` está disponible en todos los objetos que representen colecciones de tipo CFC, ya que las clases de dichos objetos heredarán la definición de la clase padre ConjunctiveFCs. En la implementación actual, dicho método aplica el algoritmo para el cálculo de inclusión de un conjunto difuso en otro, descrito en el capítulo anterior.

7.6.4. Operadores para la construcción de condiciones flexibles complejas

La presente sección introduce los operadores booleanos extendidos, empleando como base la lógica difusa, para la agregación de condiciones flexibles simples, que han sido construidas con los operadores introducidos en la sección anterior.

7.6.4.1. Operador FzAND

Este operador se corresponde con la operación de conjunción tal y como se define en la lógica difusa. Dicho operador se define por la signatura `FzAND(a,b)`, donde sus dos argumentos, `a` y `b`, son dos condiciones flexibles, ya sean estas simples o complejas.

El operador agrega los grados de cumplimiento resultantes de las condiciones `a` y `b`, empleando para ello una t-norma, tal y como se define la conjunción en la literatura.

En el caso del prototipo implementado, se usa la t-norma del mínimo, descrita en la Ecuación 7.21.

$$\otimes(x, y) = \text{mín}(x, y) \tag{7.21}$$

7.6.4.2. Operador FzOR

El operador `FzOR(a,b)` permite aplicar la operación de disyunción, tal y como se ha definido en la lógica difusa, con el objeto de agregar los grados de cumplimiento de dos condiciones flexibles, `a` y `b`, ya sean estas simples o complejas. El proceso de agregación consiste en la aplicación de una t-conorma, siguiendo la descripción del operador de disyunción de la lógica difusa.

El presente prototipo emplea en la implementación del operador la t-conorma del máximo, descrita en la Ecuación 7.22.

$$\otimes(x, y) = \text{máx}(x, y) \tag{7.22}$$

7.6.4.3. Operador FzNOT

Este operador es empleado para aplicar la negación, tal y como se define en la lógica difusa, sobre el grado de cumplimiento de una condición difusa, ya sea esta simple o compleja.

La signatura de este operador es `FzNOT(a)`, donde el argumento `a` se corresponde con la condición imprecisa que se desea negar.

La negación del grado de cumplimiento de una condición se hace empleado una negación fuerte. En el caso del presente prototipo, la negación fuerte empleada se define en la Ecuación 7.23.

$$C(x) = 1 - x \tag{7.23}$$

7.6.5. Operadores para la identificación y extracción de grados de cumplimiento de condiciones flexibles

La presente sección describe los operadores proporcionados por el prototipo con el fin de identificar condiciones flexibles. Este agrupamiento será útil para que el usuario pueda extraer el grado de cumplimiento de dichas condiciones con el objeto de incluirlas como columnas en la tabla resultante de la consulta que plantee el usuario.

7.6.5.1. Operador FCond

Este operador se emplea para la identificación de condiciones flexibles. La signatura de dicho operador corresponde con la siguiente `FCond(cond,numid)`.

Los argumentos de este operador corresponden con la condición flexible que se desea identificar (`cond`) y el identificador numérico que se desea asociar a dicha condición (`numid`).

7.6.5.2. Operador CDEG

Este operador, empleado en la parte de selección de columnas de la condiciones, permite obtener el grado de cumplimiento de una determinada condición flexible identificada empleando el operador FCond descrito anteriormente.

La signatura de este método es `CDEG(numid)`, donde su único argumento, `numid`, es el identificador numérico de la condición flexible de la cual se desea obtener su grado de cumplimiento.

7.7. Ejemplo de uso del prototipo

La presente sección mostrará un ejemplo completo del presente prototipo, con el objeto de ilustrar sus capacidades para el almacenamiento y manipulación de información flexible.

7.7.1. Descripción del ejemplo

En el desarrollo del ejemplo se creará un esquema de base de datos que permitirá almacenar los datos de los pilotos de una línea aérea. Sobre dichos pilotos, se almacenarán los siguientes atributos:

- *Nombre*: Será un atributo preciso que contendrá el nombre completo del piloto.
- *HorasDeVuelo*: Este atributo almacenará las horas de vuelo de un determinado piloto. El atributo será de tipo impreciso, concretamente de tipo OAFT, ya que permitirá almacenar tanto valores concretos, indicando horas de vuelo, así como valores imprecisos en forma de etiquetas lingüísticas como las siguientes: $\{pocas, medias, muchas\}$. Éstas están definidas por las distribuciones de posibilidad trapezoidales $[0, 0, 500, 750]$, $[600, 1000, 5000, 9000]$ y $[7500, 9000, 150000, 15000]$, respectivamente.
- *Conocimientos*: Este campo almacenará los conocimientos que el piloto posee de técnicas relacionadas con la aeronáutica, como puede ser $\{navegación, mecánica, electrónica, psicología, \ldots\}$. Este campo será de tipo CFC, ya que los conocimientos pueden ser modelados como un conjunto, de semántica conjuntiva, y los grados de pertenencia al conjunto indicará el nivel de conocimiento de piloto de dicha área.

Teniendo en cuenta lo anterior, un piloto será representado en la base de datos como un objeto difuso cuyos atributos serán los descritos anteriormente.

7.7.2. Creación de clases de usuario

Para modelar el ejemplo anterior, se requieren dos tipos de datos imperfectos: uno, derivado de la clase OrderedAFTs, para el atributo *HorasDeVuelo* y otro, derivado de la clase CFC, para el atributo *Conocimientos*.

Para crear una clase que modele el dominio del atributo *HorasDeVuelo*, que denominaremos *HorasDeVueloT*, deberemos invocar convenientemente el método estático `extends(typeName)` de la clase OrderedAFTs, que creará automáticamente una subclase de ésta.

La sentencia que se necesita ejecutar para crear el tipo que modela el citado dominio es la siguiente:

```
EXEC OrderedAFTs.extends('HorasDeVueloT');
```

La anterior sentencia, si su ejecución ha sido satisfactoria, crea la clase HorasDeVueloT, subclase de OrderedAFTs, que ofrece la capacidad de representar datos de tipo OAFT.

Sobre el anterior dominio, hay definidas varias etiquetas lingüísticas que se crearán empleando el método `labelDef(label,value)` descrito anteriormente.

Las sentencias necesarias para crear dichas etiquetas lingüísticas son las siguientes:

```
EXEC HorasDeVueltoT.labelDef(
    'pocas',Trapezoid(0,0,500,750)
    );

EXEC HorasDeVueltoT.labelDef(
    'medias',Trapezoid(600,1000,5000,9000)
    );

EXEC HorasDeVueltoT.labelDef(
    'muchas',Trapezoid(7500,9000,150000,150000)
    );
```

La segunda clase, que modelará el dominio del atributo *Conocimientos*, heredará de la clase ConjunctiveFCs para que permita almacenar conjuntos de tipo CFC. Esta subclase estará parametrizada en función del tipo de los elementos que contendrá. En este caso, consideraremos que éstos son cadenas de caracteres correspondientes a las etiquetas que se emplean para identificar el área de conocimiento que se quiere representar y que número máximo de elementos que podrá contener la colección será 10.

Para obtener subclases parametrizadas de la clase ConjunctiveFCs, dicha clase ofrece el método estático `extends(typeName,elementTypeName,size)`, como ya hemos visto anteriormente. Dicho método se empleará en la siguiente sentencia para crear la clase deseada:

```
EXEC ConjunctiveFCs.extends(
    'ConocimientosT','VARCHAR2(10)',10
    );
```

Con la anterior sentencia, habremos creado la clase ConocimientoT, que modela una colección de tipo CFC, cuyos elementos serán cadenas de caracteres y cuya capacidad máxima será de 10 elementos.

Una vez creadas las clases para modelar los atributos básicos, crearemos la clase difusa que agrupa los anteriores atributos y permite la creación de objetos difusos que van a representar los pilotos de una linea aérea. Denominaremos a ésta clase como *PilotoT*.

La sentencia que para la creación de dicha clase es la siguiente:

```
CREATE TYPE PilotoT UNDER FO (
    nombre VARCHAR2(50),
    horasDeVuelo HorasDeVueloT,
    conocimientos ConocimientosT
    );
```

La sentencia para la creación de la tabla que contendrá los datos de los pilotos de la compañía es la siguiente:

```
CREATE TABLE Pilotos OF PilotoT (
    PRIMARY KEY (nombre)
    );
```

Nombre	Horas de Vuelo	Conocimientos
Alejandro Malatesta	8000	{0.6/*mecánica*}
Miguel Brunete	*pocas*	{}
Rodrigo Girón	*muchas*	{0.5/*psicología*, 0.2/*mecánica*}

Tabla 7.3: Pilotos de la línea aérea

7.7.3. Inserción de datos

En éste ejemplo supondremos que se desean representar los pilotos que aparecen en la Tabla 7.3.

Una vez creada la tabla que albergará los datos de los pilotos en la base se datos, se pueden ejecutar la siguientes sentencias de inserción para representar los pilotos que aparecen en la Tabla 7.3:

```
INSERT INTO Pilotos VALUES (
    'Alejandro Malatesta',
    Crisp(8000),
    ConocimientosT(0.6,'mecánica')
    );

INSERT INTO Pilotos VALUES (
    'Miguel Brunete',
    HorasDeVueltoT.getLabel('pocas'),
    ConocimientosT()
    );


INSERT INTO Pilotos VALUES (
    'Rodrigo Girón',
    HorasDeVueltoT.getLabel('muchas'),
    ConocimientosT(0.5,'psicología',0.2,'mecánica')
    );
```

7.7.4. Consultas flexibles

Siguiendo el anterior ejemplo, supongamos que la sección de personal de la compañía aérea desea extraer un listado de los pilotos con *muchas* horas de vuelo para organizar unos seminarios de reciclaje. La consulta, que devuelve el nombre del piloto, las horas de vuelto con que cuenta y el grado en que se ajusta al criterio *muchas horas de vuelo*, es la siguiente:

```
SELECT nombre, horasDeVuelo, CDEG(1)
FROM Pilotos
WHERE FCond(
    FEQ(horasDeVuelto,HorasDeVueltoT.getLabel('muchas')), 1
    ) > 0;
```

El resultado de la anterior consulta se muestra en la Tabla 7.4.

Nombre	Horas de Vuelo	Grado de Cumplimiento
Alejandro Malatesta	8000	0.33
Rodrigo Girón	muchas	1

Tabla 7.4: Pilotos con muchas horas de vuelo

Nombre	Horas de Vuelo	Grado de Cumplimiento
Alejandro Malatesta	8000	0.33
Rodrigo Girón	muchas	0.2

Tabla 7.5: Pilotos con muchas horas de vuelo y conocimientos en mecánica

Si además de lo anterior, el conocimiento de mecánica fuese un requisito para participar en el curso de reciclaje, la consulta anterior se podría extender de la siguiente forma:

```
SELECT nombre, horasDeVuelo, CDEG(1)
FROM Pilotos
WHERE FCond(
    FzAND(
        FEQ(horasDeVuelto,HorasDeVueltoT.getLabel('muchas')),
        FInclusion(Conocimientos,ConocimientosT(1,'mecánica'))
        ),1
    )>0 ;
```

El resultado de la consulta anterior se corresponde con la Tabla 7.5.

7.8. Conclusiones

En el presente capítulo se ha propuesto una extensión de un SGBD objeto-relacional, así como el prototipo que la implementa, para permitir la representación y manejo de datos imperfectos.

La propuesta incluye capacidades para representar datos de naturaleza imperfecta, tanto a nivel atómico como complejo. Dentro del nivel atómico es capaz de representar datos imprecisos definidos sobre dominios ordenados y sobre dominios escalares en los que se define una relación de similitud. En lo que respecta a los datos imperfectos complejos, la propuesta admite datos estructurados y colecciones, tanto con semántica conjuntiva como disyuntiva. Estas últimas capacidades están inspiradas en las construcciones soportadas en los sistemas SQL que implementan el estándar SQL:1999.

Gracias a que la propuesta se realiza partiendo de un modelo de bases de datos objeto-relacional, ésta reduce o elimina las dificultades detectadas en propuestas anteriores. En primer lugar, la propuesta facilita el modelado de datos complejos, ya que contempla la posibilidad de que los usuarios puedan definir sus propios objetos complejos difusos. En segundo lugar, la solución propuesta facilita la interacción con la base de datos de aplicaciones orientadas a objetos, minimizando el problema de acoplamiento entre aplicaciones programadas

en lenguajes orientados a objetos y la base de datos gracias a que ofrece mecanismos para representar éstos de forma eficiente. En tercer lugar, debido a que se emplean relaciones para representar los datos, la debilidad ante cambios en el esquema se minimiza. En último lugar, al ofrecer en SGBD anfitrión una interfaz independiente del lenguaje de programación, no se manifiesta la fuerte dependencia del lenguaje en que ha sido desarrollado el SGBD que se observa en otras propuestas.

En lo que respecta a la estrategia de implementación empleada, la propuesta ofrece diversas ventajas. En primer lugar, la extensión propuesta se integra totalmente en la estructura del lenguaje de consulta del servidor anfitrión, ya que hace uso de elementos ya contemplados en éste como UDTs, métodos asociados a los mismos y operadores definidos por el usuario. Esto evita el empleo de un traductor, dejando que sea el propio servidor anfitrión el que analice las sentencias, lo cual permite que éstas contengan cualquier elemento contemplado en el lenguaje de consulta del anfitrión. De esta forma, se ofrece al usuario la posibilidad de usar todas las capacidades del lenguaje de consulta del servidor anfitrión, incluso ante cambios futuros en el lenguaje, así como el empleo de características adicionales, o de valor añadido, para el tratamiento de datos especializados. Finalmente, se abre la posibilidad a la incorporación de mecanismos para incrementar el rendimiento en el procesamiento de consultas, particularmente índices especializados y optimización de consultas.

Capítulo 8

Indexado de datos difusos

8.1. Introducción

Como se comentó anteriormente en el presente trabajo, la disposición de mecanismos de indexado que incrementen el rendimiento de la ejecución de consultas difusas en los SGBDDs es crucial para la exitosa implantación de los mismos en entornos reales de producción. Esta situación es similar a la ocurrida, a principios de los 70, en el caso de los primeros prototipos de SGBD basados en el modelo relacional. El modelo relacional no reemplazó al modelo de red como modelo dominante hasta que los SGBD basados en el mismo fueron lo suficientemente eficientes. En la misma línea, esta necesidad de técnicas de indexado específicas para datos imprecisos ha sido señalada por Kraft, Petry y Bosc en sus trabajos sobre el estado del arte de la investigación en bases de datos difusas [87, 28].

El presente capítulo propone varias técnicas de indexado para dos dominios básicos de los SGBDs, el de los datos imprecisos de naturaleza numérica y el de los datos imprecisos escalares. Estas técnicas han sido concebidas con la premisa de que sus estructuras de indexado subyacentes han de estar basadas en las ya existentes en los SGBD *crisp* actuales, los árboles B^+ y los mapas de bits. El motivo de tal condición es proporcionar técnicas de indexado de datos imprecisos que puedan ser fácilmente incorporadas en los SGBDs desarrollados, tal y como se propone en el presente trabajo, como extensión de un SGBD clásico.

Este capítulo está organizado de la siguiente forma. El primer lugar, se describirá con detalle el tipo de datos y de consultas que las técnicas de indexado propuestas han de manejar. Seguidamente, se describirán los principios de indexado que permiten la transformación de consultas de naturaleza difusa en unos filtros construidos mediante consultas conjuntivas *crisp* que servirán como base para la concepción de las técnicas de indexado propuestas. En tercer y cuarto lugar, se describirán las técnicas propuestas para el indexado de datos imprecisos de naturaleza numérica y escalar, respectivamente. Finalmente, se detallarán las conclusiones del contenido del presente capítulo.

8.2. Consultas flexibles

Como se ha indicado en la introducción, el presente capítulo está dedicado al estudio de estructuras para el indexado de datos imprecisos o, genéricamente, datos difusos. El objetivo de tales estructuras es facilitar y acelerar el procesamiento de consultas flexibles que se refieran a los datos indexados. La presente sección está dedicada a definir de manera concreta la naturaleza de tales *consultas flexibles*. Para ello, definiremos previamente los diferentes conceptos básicos que servirán para facilitar dicha definición, no sin antes especificar un marco que especifique los tipos de datos sobre los que éstas se aplicarán.

8.2.1. Datos imprecisos

Antes de comenzar con las citadas definiciones, estableceremos el marco sobre el que éstas se apoyarán. Supongamos, para el caso que nos ocupa y sin pérdida de generalidad, que disponemos de una MRCD, que denominaremos m, cuyo cuerpo contiene n tuplas $(t_1, t_2, \ldots, t_n)$ y cuyo conjunto de atributos está compuesto por al menos un atributo que denominaremos a. Los valores asociados a las tuplas de m para dicho atributo son valores imprecisos pertenecientes al dominio asociado al atributo, que notaremos como D_a. Éstos valores imprecisos son modelados como distribuciones de posibilidad sobre los valores del dominio subyacente de D_a, que notaremos como $sub(D_a)$. Notaremos a la distribución de posibilidad que modela el valor impreciso asociado al atributo a para una determinada tupla t_i como $\Pi_{t_i(a)}$.

8.2.2. Restricciones flexibles

Una *restricción flexible* es una restricción gradual impuesta a los valores de un determinado atributo. Ésta puede modelarse como un conjunto difuso C donde los elementos de dicho conjunto indican los valores considerados aceptables por la restricción.

Supongamos que disponemos de una MRCD cuyas tuplas representan personas y éstas poseen un atributo que identifica la propiedad que se corresponde con la edad de éstas. Un ejemplo de este tipo de restricciones sobre el citado atributo puede ser la restricción que selecciona a la personas cuya edad está "alrededor de 30 años".

Naturalmente, una restricción flexible puede ser parcialmente satisfecha. Cuando una restricción flexible se aplica sobre las tuplas de una MRCD que posee un atributo asociado a valores imprecisos (incluyendo valores crisp como caso particular), ésta induce dos conjuntos difusos sobre el conjunto tuplas de la citada MRCD. Éstos son los siguientes:

1. El conjunto de tuplas que *posiblemente* satisfacen la restricción flexible.

2. El conjunto de tuplas que *necesariamente* satisfacen la restricción flexible.

El grado en que cada tupla pertenece a cada uno de los anteriores conjuntos se denomina, respectivamente, *grado de posibilidad* (o, por brevedad, simplemente *posibilidad*) y *grado de necesidad* (o simplemente *necesidad*). Evidentemente, éstos indican para cada tupla del conjunto en qué grado ésta satisface

posiblemente ó *necesariamente* la restricción flexible que se ha aplicado. De forma genérica, diremos que se aplica una *medida de posibilidad* o una *medida de necesidad* en función del grado que se tenga en cuenta para determinar si una restricción flexible ha sido satisfecha.

La función de pertenencia de una determinado tupla t_i a cada uno de los conjuntos difusos inducidos por una restricción flexible C está definida como se indica en la Ecuación 8.1, para el caso de posibilidad, y según se indica en la Ecuación 8.2 para el caso de necesidad. En dichas ecuaciones $\Pi\left(C/t_i(a)\right)$ y $N\left(C/t_i(a)\right)$ representan, respectivamente, el grado de posibilidad y necesidad en que la restricción flexible con un conjunto difuso de valores aceptables C es satisfecha por el valor asociado al atributo a para la tupla t_i.

$$\Pi\left(C/t_i(a)\right) = \sup_{d \in sub(D_a)} \left(\pi_{t_i(a)}(d) \otimes \mu_C(d)\right)$$
$$\forall t_i \in m | (m \in \Delta), \forall a \in A_m | (dom_m(a) \in \mathbb{I}), \forall C \in \tilde{P}(sub(D_a)) \quad (8.1)$$

$$N\left(C/t_i(a)\right) = \inf_{d \in sub(D_a)} \left(\left(1 - \pi_{t_i(a)}(d)\right) \oplus \mu_C(d)\right)$$
$$\forall t_i \in m | (m \in \Delta), \forall a \in A_m | (dom_m(a) \in \mathbb{I}), \forall C \in \tilde{P}(sub(D_a)) \quad (8.2)$$

Aunque, como acabamos de ver, una restricción flexible puede ser posiblemente o necesariamente satisfecha, no siendo exclusivas éstas medidas, en el presente trabajo asociaremos a toda restricción flexible una medida determinada. De esta forma, cada restricción flexible especificará si ésta ha de satisfacerse posiblemente o necesariamente. El resultado de la aplicación de una restricción flexible sobre el valor asociado a un determinado atributo para una tupla t_i será el grado en que ésta se satisface (ya sea posiblemente o necesariamente, según el caso) que se corresponderá con un valor numérico en el rango $[0, 1]$. Debido a esto, emplearemos la notación $M(C/a)$ para representar a una restricción flexible que se aplica sobre los valores asociados a un determinado atributo a, con un conjunto difuso de valores aceptables C y que aplica una medida M (donde M será Π para el caso de una medida de posibilidad y N para el caso de una medida de necesidad).

Las restricciones flexibles pueden ser combinadas mediante el empleo de conectivos difusos (como t-normas, t-conormas, implicaciones, etc.). Tales combinaciones resultan en nuevas restricciones flexibles. Para diferenciar las restricciones flexibles en función de si éstas proceden o no de la combinación de otras, denominaremos a las combinadas como *restricciones flexibles compuestas* y a las no combinadas como *restricciones flexibles atómicas*.

8.2.3. Condiciones flexibles

Para seleccionar un subconjunto de tuplas de una MRCD empleando una restricción flexible, ésta se suele combinar con un umbral de cumplimiento. A la anterior combinación la denominaremos *condición flexible*.

El umbral de una condición flexible especificará el grado máximo de flexibilidad con el que la restricción flexible asociada será aplicada o, lo que es

equivalente, un mínimo para el grado de cumplimiento de la restricción flexible por parte de los elementos seleccionados. El valor del umbral variará desde 1 (la restricción se aplica sin ninguna flexibilidad) hasta 0 (máxima flexibilidad de aplicación).

Los grados de cumplimiento de la restricción flexible por parte de las tuplas de una MRCD se relacionan con el umbral empleando un comparador crisp. Típicamente, los grados de cumplimiento y umbral se relacional mediante el operador "mayor o igual que", a excepción del caso en que el umbral sea 0. En este último caso se emplea el operador "mayor que" para evitar la inclusión en el conjunto de elementos seleccionados a aquellos que no cumplen en absoluto la restricción.

Para notar el hecho de que una condición flexible está compuesta por una restricción flexible de naturaleza atómica o compuesta, ésta se denominará *condición flexible atómica* ó *condición flexible compuesta*, según el caso.

Dada una restricción flexible $M(C/a)$ y un umbral λ, notaremos la condición flexible que deriva de su combinación como $\langle M(C/a), \lambda \rangle$.

8.2.4. Consultas flexibles

Finalmente, diremos que una consulta sobre un determinado conjunto de datos es flexible cuando ésta incluye una o varias condiciones flexibles. Un ejemplo de este tipo de consultas es el siguiente:

```
SELECT * FROM table WHERE FEQ(att,[a,b,c,d])>=T
```

La consulta anterior está expresada empleando la sintaxis de consulta del prototipo de SGBDORD introducido anteriormente en el presente trabajo. Ésta selecciona las tuplas de la tabla `table` que cumplen una determinada condición impuesta sobre los valores del atributo con dominio impreciso `att`. La citada condición aplica el comparador difuso `FEQ` que devuelve el grado de posibilidad en que los valores del mencionado atributo cumplen la restricción modelada por el conjunto difuso definido según la función de pertenencia trapezoidal representada por [a, b, c, d]. Este comparador devolverá un valor numérico en el intervalo $[0, 1]$ que posteriormente será comparado con el umbral `T` empleando el operador relacional "mayor o igual que".

En definitiva, la anterior consulta devolverá aquellas tuplas de `table` cuyo valor asociado al atributo `att` sea *posiblemente igual* a alguno de los valores en el conjunto difuso [a, b, c, d] siempre que el grado de cumplimiento de la anterior restricción se a igual o superior al valor umbral `T`.

8.3. Principios para el indexado de conjuntos difusos

La presente sección presenta los principios de indexado en que se basarán las distintos mecanismos de indexado propuestos en el presente capítulo. En primer lugar, se presentará una forma por la cual se podrá evaluar eficientemente la satisfacción de condiciones flexibles. Los principios de indexado propuestos se basarán en dicha forma de evaluación. Posteriormente, se presentarán dichos principios y se discutirá su aplicabilidad para su uso como base de mecanismos de indexado.

8.3.1. Evaluación eficiente de condiciones flexibles

Hasta el momento, para determinar si una consulta flexible es satisfecha por el valor asociado a una determinada tupla sería necesario emplear las expresiones de las ecuaciones 8.1 ó 8.2 según el caso. Desafortunadamente, la evaluación de tales expresiones es un proceso que puede resultar costoso. En [26] se nos propone un mecanismo más eficiente que nos permita determinar la satisfacción de tales expresiones basado en operaciones sobre conjuntos crisp finitos.

El presente apartado ilustra tales mecanismos, abordando en primer lugar el mecanismo para la evaluación de condiciones flexibles cuya restricción flexible asociada aplica una medida de posibilidad y, posteriormente, un mecanismo homólogo para el caso en que se empleen una medida de necesidad.

8.3.1.1. Evaluación eficiente de condiciones flexibles que aplican una medida de posibilidad

Dada una condición flexible $\langle \Pi(C/a), \lambda \rangle$, podemos determinar si una tupla t_i satisface posiblemente ésta de la forma que se muestra en la Ecuación 8.3.

$$\begin{aligned} \Pi\left(C/t_i(a)\right) \geq \lambda \iff & \left[\sup_{d \in sub(D_a)} \left(\pi_{t_i(a)}(d) \otimes \mu_C(d)\right)\right] \geq \lambda \iff \\ \iff & \exists d \in sub(D_a) | (\pi_{t_i(a)}(d) \geq \lambda \ \wedge \ \mu_C(d) \geq \lambda) \iff \\ \iff & L_\lambda(\Pi_{t_i(a)}) \cap L_\lambda(C) \neq \emptyset \end{aligned}$$

$$\forall t_i \in m | (m \in \Delta), \forall a \in A_m | (dom_m(a) \in \mathbb{I}), \forall C \in \tilde{P}(sub(D_a)), \forall \lambda \in (0,1] \quad (8.3)$$

Como se puede observar en la anterior ecuación, una determinada tupla t_i satisface una condición flexible $\langle \Pi(C/a), \lambda \rangle$ si los λ-cortes de $\Pi_{t_i(a)}$ (la distribución de posibilidad que representa el valor impreciso asociado al atributo a para la tupla t_i) y de C (el conjunto difuso de valores aceptables por la restricción flexible que impone la condición) se solapan.

Para el caso particular en que el umbral establecido es 0, la expresión aplicada es la que se muestra en la Ecuación 8.4.

$$\begin{aligned} \Pi(C/t_i(a)) > 0 \iff & \left[\sup_{d \in sub(D_a)} \left(\pi_{t_i(a)}(d) \otimes \mu_C(d)\right)\right] \geq 0 \iff \\ \iff & \exists d \in sub(D_a) | (d \in \Pi_{t_i(a)} \ \wedge \ d \in C) \iff \\ \iff & \Pi_{t_i(a))} \cap C \neq \emptyset \iff L_{>0}(\Pi_{t_i(a)}) \cap L_{>0}(C) \neq \emptyset \end{aligned}$$

$$\forall t_i \in m | (m \in \Delta), \forall a \in A_m | (dom_m(a) \in \mathbb{I}), \forall C \in \tilde{P}(sub(D_a)) \quad (8.4)$$

En este último caso, los λ-cortes han sido sustituidos por el soporte de $\Pi_{A(o_i)}$ y C para asegurar que el grado de posibilidad es estrictamente superior a 0.

8.3.1.2. Evaluación eficiente de condiciones flexibles que aplican una medida de necesidad

De forma similar, podemos determinar si un elemento o_i satisface una condición flexible $\langle N(C/a), \lambda\rangle$. Para el caso de necesidad se aplica el razonamiento mostrado en la Ecuación 8.5.

$$\begin{aligned} N\left(C/t_i(a)\right) \geq \lambda \iff & \left[\inf_{d \in sub(D_a)} \left(\left(1-\pi_{t_i(a)}(d)\right) \oplus \mu_C(d)\right)\right] \geq \lambda \\ \iff & \forall d \in sub(D_a) | \left((1-\pi_{t_i(a)}(d)) \geq \lambda \ \vee \ \mu_C(d) \geq \lambda\right) \iff \\ \iff & \forall d \in sub(D_a) | \left(\pi_{t_i(a)}(d) \leq (1-\lambda) \ \vee \ \mu_C(d) \geq \lambda\right) \iff \\ \iff & \forall d \in sub(D_a) | \left(d \notin L_{>(1-\lambda)}(\Pi_{t_i(a)}) \ \vee \ d \in L_\lambda(C)\right) \iff \\ \iff & \forall d \in sub(D_a) | \left(d \in L_{>(1-\lambda)}(\Pi_{t_i(a)}) \Rightarrow d \in L_\lambda(C)\right) \iff \\ \iff & L_{>(1-\lambda)}(\Pi_{t_i(a)}) \subseteq L_\lambda(C) \end{aligned}$$

$$\forall t_i \in m | (m \in \Delta), \forall a \in A_m | (dom_m(a) \in \mathbb{I}), \forall C \in \tilde{P}(sub(D_a)), \forall \lambda \in (0,1] \quad (8.5)$$

Podemos observar que la anterior ecuación establece que un determinado elemento o_i satisface una condición flexible $\langle N(C/a), \lambda\rangle$ si el $(1-\lambda)$-corte estricto de $\Pi_{t_i(a)}$ es un subconjunto del λ-corte de C.

De forma análoga al caso de las condiciones flexibles que aplican medidas de posibilidad, cuando el umbral establecido es 0, la expresión aplicada para determinar el cumplimiento de la condición flexible es distinta a la del caso genérico. En este caso particular se ha de aplicar la expresión que se muestra en la Ecuación 8.6.

$$\begin{aligned} N\left(C/t_i(a)\right) > 0 \iff & \left[\inf_{d \in sub(D_a)} \left(\left(1-\pi_{t_i(a)}(d)\right) \oplus \mu_C(d)\right)\right] \geq 0 \\ \iff & \forall d \in sub(D_a) | \left((1-\pi_{t_i(a)}(d)) > 0 \ \vee \ \mu_C(d) > 0\right) \iff \\ \iff & \forall d \in sub(D_a) | \left(\pi_{t_i(a)}(d) < 1 \ \vee \ \mu_C(d) > 0\right) \iff \\ \iff & \forall d \in sub(D_a) | \left(d \notin L_1(\Pi_{t_i(a)}) \ \vee \ d \in L_{>0}(C)\right) \iff \\ \iff & \forall d \in sub(D_a) | \left(d \in L_1(\Pi_{t_i(a)}) \Rightarrow d \in L_{>0}(C)\right) \iff \\ \iff & L_1(\Pi_{t_i(a)}) \subseteq L_{>0}(C) \end{aligned}$$

$$\forall t_i \in m | (m \in \Delta), \forall a \in A_m | (dom_m(a) \in \mathbb{I}), \forall C \in \tilde{P}(sub(D_a)) \quad (8.6)$$

De forma similar al caso en que se aplica una medida de posibilidad, en la anterior ecuación, los diferentes *alpha*-cortes han sido sustituidos por el soporte o núcleo, según el caso. Concretamente, el $(1-\lambda)$-corte de $\Pi_{A(o_i)}$ ha sido sustituido por su núcleo y el λ-corte de C ha sido sustituido por su soporte. De esta forma, se garantiza que el grado de necesidad de los elementos es estrictamente superior a 0.

8.3.2. Principios de indexado

Las anteriores expresiones mostradas en las ecuaciones 8.3 y 8.5, así como sus casos particulares mostrados en las ecuaciones 8.4 y 8.6, reducen la evaluación de condiciones flexibles a la sencilla tarea de determinar si dos conjuntos se solapan o si un conjunto es parte de otro, según el caso.

Dada una condición flexible $\langle M(C/a), \lambda \rangle$, estas sencillas operaciones podrían incluso ser rápidamente resueltas empleando una estructura de índice adecuada. No obstante, ya que el umbral de una determinada condición flexible puede ser cualquier grado en $[0, 1]$, este planteamiento se encuentra con el problema de determinar qué α-cortes de los conjuntos difusos indexados serían incluidos en dicha estructura.

Para solventar tal inconveniente Bosc propone en [26] derivar las expresiones de las ecuaciones 8.3, 8.6, 8.5 y 8.6 hacia expresiones que requieran la participación de determinados α-cortes de los conjuntos difusos evaluados, en lugar de λ-cortes o $(1 - \lambda)$-cortes, según el caso, arbitrarios.

8.3.2.1. Principio débil de indexado para condiciones que aplican medidas de posibilidad

Para el caso de condiciones flexibles que aplican una medida de posibilidad, la expresión 8.3 es derivada de la forma que se muestra en la Ecuación 8.7.

$$\Pi\left(C/t_i(a)\right) \geq \lambda \iff L_\lambda(\Pi_{t_i(a)}) \cap L_\lambda(C) \neq \emptyset \Rightarrow L_{>0}(\Pi_{t_i(a)}) \cap L_\lambda(C)$$
$$\forall t_i \in m | (m \in \Delta), \forall a \in A_m | (dom_m(a) \in \mathbb{I}), \forall C \in \tilde{P}(sub(D_a)), \forall \lambda \in (0, 1] \quad (8.7)$$

Obsérvese que en la anterior expresión se ofrece un principio débil por el que se puede determinar la satisfacción de una condición flexible $\langle \Pi(C/a), \lambda \rangle$ para una tupla t_i conociendo exclusivamente el soporte de $\Pi_{t_i(a)}$. Nótese, además, que para el caso particular en el que el umbral es 0 se empleará igualmente el soporte de $\Pi_{t_i(a)}$ para determinar la satisfacción de la condición flexible, tal y como se indica en la Ecuación 8.4.

8.3.2.2. Principio débil de indexado para condiciones que aplican medidas de necesidad

En el caso de condiciones flexibles que empleen medidas de necesidad, la expresión 8.5 puede ser derivada de la forma que se muestra en la Ecuación 8.8.

$$N\left(C/t_i(a)\right) \geq \lambda \iff L_{>(1-\lambda)}(\Pi_{t_i(a)}) \subseteq L_\lambda(C) \Rightarrow L_1(\Pi_{t_i(a)}) \subseteq L_\lambda(C)$$
$$\forall t_i \in m | (m \in \Delta), \forall a \in A_m | (dom_m(a) \in \mathbb{I}), \forall C \in \tilde{P}(sub(D_a)), \forall \lambda \in (0, 1] \quad (8.8)$$

En la ecuación anterior, se ofrece un principio débil con el que se puede determinar si una condición flexible $\langle N(C/a), \lambda \rangle$ se satisface para una tupla t_i, siendo sólo necesario conocer el núcleo de $\Pi_{t_i(a)}$. Nótese que en el caso particular de que el umbral sea 0, y tal y como se muestra en la Ecuación 8.6, también se emplea el núcleo de $\Pi_{t_i(a)}$ para determinar la satisfacción de la condición flexible.

8.3.2.3. Ventajas y limitaciones de los principios débiles de indexado

Como se puede observar, las expresiones mostradas en las ecuaciones 8.7 y 8.8 ofrecen la capacidad de resolver una condición flexible $\langle N(C/a), \lambda \rangle$ necesitando solamente conocer el soporte o núcleo, según sea la medida que esta aplique, de la distribución de posibilidad del valor asociado al atributo a para el elemento evaluado. Esta nueva situación hace viable el empleo de estructuras de indexado que contendrán, según el caso y por cada elemento indexado, uno de estos conjuntos.

No obstante, se ha de tener en cuenta que los anteriores principios débiles han sido derivados, mediante implicación, de los introducidos por las ecuaciones 8.3 y 8.5. De esta forma, las condiciones que éstos imponen son necesarias pero no suficientes para determinar la satisfacción de una condición flexible. Por tanto, su uso sólo permite hacer una preselección de los elementos que potencialmente podrán satisfacer una determinada condición flexible, siendo necesaria una criba posterior de éstos para determinar con seguridad cuáles satisfacen ésta realmente.

A partir de este punto, denominaremos al conjunto de tuplas que potencialmente satisfacen una condición flexible $\langle N(C/a), \lambda \rangle$, determinado mediante los principios de indexado débiles anteriormente descritos, como *conjunto candidato* de ésta. El conjunto candidato de una condición flexible $\langle N(C/a), \lambda \rangle$ será notado como $PS_{\langle N(C/a), \lambda \rangle}$.

8.4. Indexado de datos imprecisos numéricos

La primera de las propuestas del presente capítulo plantea la adaptación de los árboles B^+ para permitir el indexado de datos imprecisos numéricos. Como se indicó en la introducción del presente capítulo, la creación de técnicas de indexado que empleen estructuras de indexado subyacente empleadas comúnmente por sistemas de SGBD clásicos facilitará la incorporación de éstas en los SGBDs que se desarrollen como extensión de los mismos.

Las técnicas propuestas, una vez descritas, serán evaluadas con respecto a una técnica de indexado de referencia. En este caso, se ha optado por el uso como referencia de los árboles G. Éstos han sido adaptados previamente para el indexado de este tipo de datos imprecisos [95] y son los únicos métodos presentes en la bibliografía que permiten la resolución de consultas arbitrarias, cualidad indispensable si esta técnica de indexado se desea emplear en un SGBD de propósito general.

8.4.1. Datos imprecisos numéricos

Como se vió en secciones previas del presente trabajo, un dato impreciso puede ser modelado como una distribución de posibilidad sobre un determinado dominio subyacente. Cuando dicho dominio posee una relación de orden definida, denominaremos a dicho tipo de datos como *datos imprecisos numéricos*.

Concretamente, ceñiremos este trabajo a los datos imprecisos numéricos definidos como distribuciones de posibilidad convexas sobre dominios subyacentes en los que existe una relación lineal de orden establecida entre sus elementos. Un ejemplo de estos tipos de datos asociados a un atributo que represente la

edad de una persona puede ser la cantidad imprecisa "alrededor de 5" ó "entre 20 y 30" años.

Los datos imprecisos numéricos pueden ser modelados empleando cualquier distribución de posibilidad siempre que ésta cumpla los requisitos establecidos anteriormente. Las técnicas de indexado que se introducen en la presente sección son válidas para cualquiera de estas representaciones. No obstante, en el presente trabajo emplearemos exclusivamente distribuciones de posibilidad trapezoidales por motivos de simplicidad. Éstas, aún poseyendo funciones características simples, poseen una gran capacidad expresiva. Empleando distribuciones de posibilidad trapezoidal se pueden expresar distintos tipos de datos imprecisos numéricos como, por ejemplo, los siguientes:

- Intervalos flexibles: Cuando la distribución de posibilidad trapezoidal es de la forma $[p-m, p, q, q+m]$, donde $[p, q]$ es un intervalo crisp y m es el margen que define la flexibilidad del intervalo.

- Números aproximados: Este tipo de valores es un caso particular del tipo anterior cuando la distribución de posibilidad trapezoidal se define como $[n-m, n, n, n+m]$, donde n es el valor que es aproximado y m es el margen que expresa la flexibilidad de la aproximación.

- Intervalos crisp: Este tipo es un caso particular de intervalo flexible en que la flexibilidad con el que éste se aplica es nula. Los intervalos crisp se representan empleando una distribución de posibilidad que asigna el máximo grado de posibilidad a todos los valores dentro de un intervalo. Este es el caso cuando la distribución de posibilidad trapezoidal se define de la forma $[a, a, b, b]$, donde a y b son, respectivamente, los límites inferior y superior del intervalo.

- Números crisp: Este tipo de valor impreciso numérico es un caso particular de todos los tipos anteriores, donde la imprecisión y la flexibilidad es nula. La distribución de posibilidad de un número crisp se define de la forma $[n, n, n, n]$, donde n es el número crisp que se desea representar.

En este marco, asumiremos también que los conjuntos difusos empleados para definir una restricción difusa sobre este tipo de datos cumple, al igual que los datos, las restricciones establecidas anteriormente para éstos. Concretamente, será necesario que la función de pertenencia de los conjuntos difusos que definen las restricciones flexibles sea convexa.

8.4.2. Principios de indexado aplicados a datos imprecisos numéricos

Cuando los principios de indexado introducidos anteriormente se aplican a datos imprecisos numéricos, representados como distribuciones de posibilidad definidas sobre un dominio en el que existe una relación de orden lineal entre sus elementos, éstos pueden ser reformulados atendiendo a dicha relación de orden.

El presente apartado presenta la citada reformulación propuesta por Bosc. Posteriormente, describe las condiciones de aplicación establecidas para ésta por el autor y analiza el conflicto que dichas condiciones generan cuando se

aplican sobre datos imprecisos numéricos descritos por funciones características trapezoidales. Finalmente, se proponen unos nuevos principios de indexado compatibles con el uso de funciones características trapezoidales.

8.4.2.1. Reformulación de los principios de indexado

Bajo las condiciones que se establecieron en el apartado anterior para los datos imprecisos numéricos y las restricciones flexibles que se aplican sobre ellos, podemos asumir que cualquier α-corte de un conjunto difuso (ya sea un conjunto difuso que modela una distribución de posibilidad que describe un dato impreciso numérico o un conjunto difuso que define los valores aceptables por una restricción difusa) se corresponde con un intervalo definido sobre el dominio subyacente ordenado de éste.

Teniendo en cuenta la anterior consideración, los principios débiles de indexado mostrados en la ecuaciones 8.7 y 8.8, así como sus casos particulares para umbral 0 mostrados en la ecuaciones 8.3 y 8.5, pueden ser reformulados como una conjunción de condiciones relacionales. La reformulación de los principios débiles de indexado se corresponde, respectivamente y en función de si se trata una restricción flexible que aplique una medida de posibilidad o necesidad, con las expresiones mostradas en las ecuaciones 8.9 y 8.11. En lo que respecta a sus casos particulares, éstos han sido reformulados como se muestra en las ecuaciones 8.10 y 8.12. En dichas ecuaciones, las funciones ínf y sup se corresponden, respectivamente con el ínfimo y supremo del conjunto crisp que le es suministrado como argumento.

$$\begin{aligned}&\Pi\left(C/t_i(a)\right) \geq \lambda \Rightarrow L_{>0}(\Pi_{t_i(a)}) \cap L_\lambda(C) \iff \\ &\iff \text{ínf}\left(L_{>0}\left(\Pi_{t_i(a)}\right)\right) \leq \sup\left(L_\lambda(C)\right) \wedge \sup\left(L_{>0}\left(\Pi_{t_i(a)}\right)\right) \geq \text{ínf}\left(L_\lambda(C)\right)\end{aligned}$$

$$\forall t_i \in m|(m \in \Delta), \forall a \in A_m|(dom_m(a) \in \mathbb{I}), \forall C \in \tilde{P}(sub(D_a)), \forall \lambda \in (0,1] \quad (8.9)$$

$$\begin{aligned}&\Pi\left(C/t_i(a)\right) > 0 \iff L_{>0}(\Pi_{t_i(a)}) \cap L_{>0}(C) \iff \\ &\iff \text{ínf}\left(L_{>0}\left(\Pi_{t_i(a)}\right)\right) \leq \sup\left(L_{>0}(C)\right) \wedge \sup\left(L_{>0}\left(\Pi_{t_i(a)}\right)\right) \geq \text{ínf}\left(L_{>0}(C)\right)\end{aligned}$$

$$\forall t_i \in m|(m \in \Delta), \forall a \in A_m|(dom_m(a) \in \mathbb{I}), \forall C \in \tilde{P}(sub(D_a)) \quad (8.10)$$

$$\begin{aligned}N\left(C/t_i(a)\right) \geq \lambda \Rightarrow L_1(\Pi_{t_i(a)}) \subseteq L_\lambda(C) \iff \\ \iff \text{ínf}\left(L_\lambda(C)\right) \geq \text{ínf}\left(L_1\left(\Pi_{t_i(a)}\right)\right) \geq \sup\left(L_\lambda(C)\right) \wedge \\ \text{ínf}\left(L_\lambda(C)\right) \geq \sup\left(L_1\left(\Pi_{t_i(a)}\right)\right) \geq \sup\left(L_\lambda(C)\right)\end{aligned}$$

$$\begin{aligned}\forall t_i \in m|(m \in \Delta), \forall a \in A_m|(dom_m(a) \in \mathbb{I}), \\ \forall C \in \tilde{P}(sub(D_a)), \forall \lambda \in (0,1] \quad (8.11)\end{aligned}$$

$$N\left(C/t_i(a)\right) > 0 \iff L_1(\Pi_{t_i(a)}) \subseteq L_{>0}(C) \iff$$
$$\iff \text{ínf}\left(L_{>0}(C)\right) \geq \text{ínf}\left(L_1\left(\Pi_{t_i(a)}\right)\right) \geq \sup\left(L_{>0}(C)\right) \wedge$$
$$\text{ínf}\left(L_{>0}(C)\right) \geq \sup\left(L_1\left(\Pi_{t_i(a)}\right)\right) \geq \sup\left(L_{>0}(C)\right)$$

$$\forall t_i \in m|(m \in \Delta), \forall a \in A_m|(dom_m(a) \in \mathbb{I}), \forall C \in \tilde{P}(sub(D_a)) \quad (8.12)$$

Como se puede observar, las nuevas expresiones en que derivan los principios de indexado limitan el rango de los límites inferiores y superiores del intervalo que representa el soporte o núcleo, según el caso, de $\Pi_{t_i(a)}$ en función de los límites inferiores y superiores del intervalo que representa la base de la condición flexible aplicada.

8.4.2.2. Condiciones de aplicación de los principios de indexado

La reformulación de los principios de indexado presentada anteriormente ha sido propuesta por Bosc en [26]. En dicho trabajo se advierte que ésta es válida si se mantienen las siguientes asunciones:

1. Los dominios subyacentes de los conjuntos difusos poseen una relación de orden lineal asociada.
2. Todo valor impreciso numérico está representado por una distribución de posibilidad cuyo soporte ha de ser caracterizado como un intervalo cerrado sobre el dominio subyacente.
3. Toda restricción difusa ha de poseer un conjunto difuso de valores aceptables cuyo soporte ha de ser, de la misma forma que para los valores imprecisos numéricos, un intervalo cerrado.

Si consideramos el marco establecido en el apartado en el que se define la naturaleza de los datos imprecisos numéricos, concretamente el punto en que indicamos que en el presente trabajo emplearemos distribuciones de posibilidad trapezoidales, queda patente que las dos últimas asunciones de la anterior lista no son mantenidas ya que los soportes en el caso de funciones de pertenencia trapezoidales son intervalos abiertos.

8.4.2.3. Principios de indexado para distribuciones y restricciones flexibles trapezoidales

En nuestra opinión el par de asunciones que impide la aplicación de los principios de indexado sobre distribuciones de posibilidad y restricciones flexibles trapezoidales puede ser omitido. Esta omisión es posible si de los principios de indexado propuestos en las ecuaciones 8.9, 8.10 y 8.12 se derivan en unos nuevos principios débiles de indexado como los mostrados, respectivamente, en las ecuaciones 8.13, 8.14 y 8.15. En dichas ecuaciones, $\Pi_{t_i(a)}$ es una distribución de posibilidad definida por una función de posibilidad trapezoidal que se corresponde con $[\alpha_{\Pi_{t_i(a)}}, \beta_{\Pi_{t_i(a)}}, \gamma_{\Pi_{t_i(a)}}, \delta_{\Pi_{t_i(a)}}]$. De la misma forma, C es un conjunto difuso definido por una función de pertenencia trapezoidal que se corresponde con $[\alpha_C, \beta_C, \gamma_C, \delta_C]$.

$$\Pi\left(C/t_i(a)\right) \geq \lambda \Rightarrow$$
$$\text{ínf}\left(L_{>0}\left(\Pi_{t_i(a)}\right)\right) \leq \sup\left(L_\lambda(C)\right) \ \wedge \ \sup\left(L_{>0}\left(\Pi_{t_i(a)}\right)\right) \geq \text{ínf}\left(L_\lambda(C)\right) \Rightarrow$$
$$\alpha_{\Pi_{t_i(a)}} \geq \sup\left(L_\lambda(C)\right) \ \wedge \ \delta_{\Pi_{t_i(a)}} \geq \text{ínf}\left(L_\lambda(C)\right)$$

$$\forall t_i \in m | (m \in \Delta), \forall a \in A_m | (dom_m(a) \in \mathbb{I}),$$
$$\forall C \in \tilde{P}(sub(D_a)), \forall \lambda \in (0,1] \quad (8.13)$$

$$\Pi\left(C/t_i(a)\right) > 0 \iff$$
$$\text{ínf}\left(L_{>0}\left(\Pi_{t_i(a)}\right)\right) \leq \sup\left(L_{>0}(C)\right) \ \wedge \ \sup\left(L_{>0}\left(\Pi_{t_i(a)}\right)\right) \geq \text{ínf}\left(L_{>0}(C)\right) \Rightarrow$$
$$\alpha_{\Pi_{t_i(a)}} \leq \delta_C \ \wedge \ \delta_{\Pi_{t_i(a)}} \geq \alpha_C$$

$$\forall t_i \in m | (m \in \Delta), \forall a \in A_m | (dom_m(a) \in \mathbb{I}), \forall C \in \tilde{P}(sub(D_a)) \quad (8.14)$$

$$N\left(C/t_i(a)\right) > 0 \iff$$
$$\text{ínf}\left(L_{>0}(C)\right) \geq \text{ínf}\left(L_1\left(\Pi_{t_i(a)}\right)\right) \geq \sup\left(L_{>0}(C)\right) \ \wedge$$
$$\text{ínf}\left(L_{>0}(C)\right) \geq \sup\left(L_1\left(\Pi_{t_i(a)}\right)\right) \geq \sup\left(L_{>0}(C)\right) \Rightarrow$$
$$\alpha_C \geq \beta_{\Pi_{t_i(a)}} \geq \delta_C \ \wedge \ \alpha_C \geq \gamma_{\Pi_{t_i(a)}} \geq \delta_C$$

$$\forall t_i \in m | (m \in \Delta), \forall a \in A_m | (dom_m(a) \in \mathbb{I}), \forall C \in \tilde{P}(sub(D_a)) \quad (8.15)$$

Nótese que no ha sido necesario reformular el principio de indexado mostrado en la Ecuación 8.11 (que se aplica cuando la restricción difusa aplica una medida de necesidad y el umbral es superior a 0) ya que los α-cortes no estrictos de un conjunto difuso con función de pertenencia trapezoidal son intervalos cerrados.

Las nuevos principios de indexado , como se ha comentado, permiten la omisión de la segunda y tercera asunción hecha por Bosc para sus principios de indexado descritos en las ecuaciones 8.9, 8.10, 8.11 y 8.12. Como es natural, la obtención de esta ventaja conlleva cierto sacrificio. Al ser los nuevos principios presentados en las ecuaciones 8.13, 8.14 y 8.15 derivaciones débiles de los propuestos por Bosc, su uso como mecanismo de preselección supone la posible ocurrencia de un número mayor de falsos positivos con respecto al introducido por los originales. Éste incremento en el número de falsos positivos es producto de considerar los intervalos abiertos que se corresponde con el soporte de $\Pi_{t_i(a)}$ y C, que están descritos por funciones características trapezoidales, como intervalos cerrados.

En cualquier caso, estos nuevos principios solo introducirán nuevos falsos positivos en determinados casos extremos. Para el principio de indexado propuesto en la Ecuación 8.13, se introducen falsos positivos en los casos en que $\alpha_{\Pi_{t_i(a)}} = \sup\left(L_\lambda(C)\right)$ y $\alpha_{\Pi_{t_i(a)}}$ no sea parte del soporte de $\Pi_{t_i(a)}$. También, se introducirán falsos positivos en el caso de que $\delta_{\Pi_{t_i(a)}} = \alpha_C$ y el valor $\delta_{\Pi_{t_i(a)}}$

no sea parte del soporte de $\Pi_{t_i(a)}$. En lo que respecta al principio de indexado descrito en la Ecuación 8.14, la introducción de falsos positivos adicionales se da cuando $\alpha_{\Pi_{t_i(a)}} = \delta_C$ y esos valores no pertenecen al soporte de $\Pi_{t_i(a)}$ ó de C. De la misma manera, se introducen falsos positivos adicionales cuando $\delta_{\Pi_{t_i(a)}} = \alpha_C$ y estos valores no pertenecen al soporte de $\Pi_{t_i(a)}$ ó de C. Finalmente, para el caso del principio de indexado descrito en la Ecuación 8.15 se introducen falsos positivos adicionales en el caso de que $\alpha_C = \beta_{\Pi_{t_i(a)}}$ ó $\alpha_C = \gamma_{\Pi_{t_i(a)}}$ y el valor α_C no pertenece al soporte de C. Asimismo, se introducirán falsos positivos cuando $\delta_C = \beta_{\Pi_{t_i(a)}}$ ó $\delta_C = \gamma_{\Pi_{t_i(a)}}$ y el valor δ_C no sea parte del soporte de C.

8.4.3. Indexado de datos imprecisos numéricos como datos bidimensionales

La reformulación de los principios de indexado para datos imprecisos cuando éstos son numéricos, expuesta en el punto anterior, deriva las expresiones originales de éstos en condiciones que imponen un rango exclusivamente sobre los límites inferior y superior de los intervalos que representan el soporte o núcleo, según el caso, de las distribuciones de posibilidad que modelan los datos indexados. Los límites inferior o superior del soporte pueden ser sustituidos, como se ha indicado anteriormente, por los valores $\alpha_{\Pi_{t_i(a)}}$ y $\delta_{\Pi_{t_i(a)}}$ para el caso de distribuciones de posibilidad caracterizadas por una función de distribución de posibilidad trapezoidal de la forma $[\alpha_{\Pi_{t_i(a)}}, \beta_{\Pi_{t_i(a)}}, \gamma_{\Pi_{t_i(a)}}, \delta_{\Pi_{t_i(a)}}]$.

Este nuevo marco abre la posibilidad de que los datos imprecisos, independientemente de la forma en que se definan sus funciones de distribución de posibilidad asociadas, sean caracterizados en mecanismos de indexado empleando sólo cuatro valores. Más aún, este número puede verse reducido a la mitad si tenemos en cuenta que para la resolución de una determinada restricción flexible se emplean exclusivamente un mismo par de valores que es exclusivo y viene determinado en función de la medida, ya sea de posibilidad o necesidad, que se aplica en cada caso. En el caso de que se aplique una medida de posibilidad, este par estará formado por los límites inferior y superior del soporte de la distribución de posibilidad que caracteriza los datos indexados. Cuando se aplica una medida de necesidad, el par estará compuesto por el límite inferior y superior del núcleo de la distribución de posibilidad que caracteriza los datos indexados.

Dado que los datos necesarios para resolver una restricción flexible que aplica una de las medidas no son necesarios para la resolución de la otra, parece conveniente que se empleen estructuras de indexado separadas para indexar cada par de valores. De esta forma, se consigue reducir al mínimo el tamaño de la estructura de indexado empleada y por tanto el costo de acceso a la misma.

Teniendo en cuenta todo lo anterior, se puede concluir que los datos imprecisos numéricos pueden ser caracterizados por un par de valores ó, en definitiva, como puntos en un espacio bidimensional. Dada una tupla t_i cuyo valor asociado para el atributo a es un valor impreciso $t_i(a)$ modelado como una distribución de posibilidad $\Pi_{t_i(a)}$ caracterizada por una función de distribución de posibilidad trapezoidal de la forma $[\alpha_{\Pi_{t_i(a)}}, \beta_{\Pi_{t_i(a)}}, \gamma_{\Pi_{t_i(a)}}, \delta_{\Pi_{t_i(a)}}]$, su valor asociado puede ser representado en una estructura de indexado como el punto $(\alpha_{\Pi_{t_i(a)}}, \delta_{\Pi_{t_i(a)}}])$ ó $(\beta_{\Pi_{t_i(a)}}, \gamma_{\Pi_{t_i(a)}})$, en función de si el índice está dedicado a resolver condiciones flexibles cuyas restricciones flexibles asociadas aplican una medida de posibilidad o de necesidad.

8.4.4. Estructuras de indexado para datos imprecisos numéricos

Dado que, como se ha mencionado anteriormente, los datos imprecisos pueden ser caracterizados como puntos en un espacio bidimensional, la búsqueda de estructuras para el indexado de datos imprecisos numéricos es equivalente, dentro del marco de los principios de indexado anteriormente expuestos, a la búsqueda de estructuras para el indexado de datos bidimensionales. Éstas estructuras deben cumplir, como único requisito, que tengan la capacidad de resolver consultas de rango como las empleadas en los citados principios.

El presente capítulo introduce las estructuras de indexado para datos imprecisos numéricos propuestas en el presente trabajo y así como la estructura de indexado que se empleará como referencia para evaluar el rendimiento de las propuestas durante la fase experimental. Como se mencionado anteriormente, el propósito del presente trabajo es adaptar las estructuras de indexado más extendidas e implementadas en el ámbito de los SGBD relacionales y adaptarlas para poder indexar datos imprecisos con ellas. En el caso de los datos numéricos, la estructura de referencia son los árboles B^+. Por tanto, en primer lugar se dedicará un apartado a introducir dicha estructura. Dado que los árboles B^+ son estructuras para el indexado de datos escalares, se dedicará posteriormente un apartado en el que se describen las técnicas empleadas en el presente trabajo para poder hacer uso de dicha estructura de indexado sobre datos bidimensionales. Seguidamente, se escribirán las diferentes estructuras de indexado para datos imprecisos basadas en árboles B^+ propuestas en el presente trabajo. Finalmente, se describirá el funcionamiento de la estructura de indexado que se empleará como referencia, el árbol G.

8.4.4.1. Árboles B^+

Un árbol B^+ [54] es una de las variaciones descritas en la bibliografía de la estructura de indexado denominada árbol B [12]. Esta clase de estructuras está constituida por un tipo de árboles balanceados empleados para mantener una colección de registros ordenados en función de un determinado valor clave. La característica más destacada de estas estructuras es que están diseñadas para permitir una inserción, recuperación y borrado eficiente de registros.

Un árbol B almacena en cada uno de sus nodos las claves indexadas y punteros a los registros asociadas a éstas. Los nodos están organizados de tal forma que, cuando el árbol se recorre *en-orden*, los registros pueden ser recuperados según el orden determinado por sus valores clave. Para mantener el árbol balanceado, el número de entradas de un nodo ha de mantenerse entre un mínimo d y un máximo $2d$. Esta restricción garantiza una ocupación mínima del espacio disponible en cada nodo del 50 %. El valor d se denomina *orden* del árbol [54].

Un árbol B^+, en contraste con un árbol B, almacena las claves indexadas y los punteros a sus registros asociados sólo en sus nodos hoja. Los nodos internos de un árbol B^+ no almacenan punteros a los registros. Éstos funcionan como índices de acceso rápido a las entradas de los nodos hoja y sólo mantienen valores clave y punteros a otros nodos. Debido a que todas los punteros a los registros indexados se encuentran en los nodos hoja y a que dichos nodos están enlazados con el nodo hoja que les sigue, el rendimiento de un procesamiento secuencial de los datos indexados en un árbol B^+ es superior al de los árboles

registro	r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	r11	r12
clave	5	7	8	3	10	1	6	4	15	2	11	9

Tabla 8.1: Registros de ejemplo

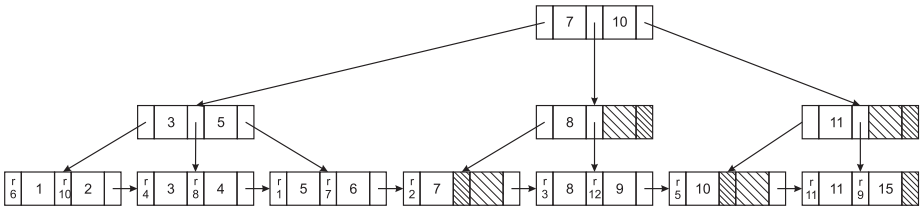

Figura 8.1: Árbol B^+ de orden 1 indexando los registros de la Tabla 8.1

B, particularmente cuando el árbol está almacenado en memoria secundaria.

El siguiente ejemplo ilustra la estructura de un árbol B^+.

Ejemplo 8.1. *Supongamos que deseamos indexar los registros que se muestran en la Tabla 8.1. Cada registro en dicha tabla se identifica por el valor indicado en la fila denominada* registro *y será indexado por el valor clave indicado en la fila* clave.

Los registros anteriores pueden ser indexados empleando un árbol B^+ que, por simplicidad, para el ámbito del presente ejemplo será de orden 1. Éste se representa gráficamente en la Figura 8.1. En dicha figura, las entradas rayadas de los nodos están vacías y los punteros a los registros han sido representados empleando el mismo identificador al que éstos hacen referencia.

En la figura citada en el ejemplo anterior, se puede apreciar que cada entrada en un nodo del árbol B^+ tiene asociados dos punteros a su lado izquierdo y derecho. En el caso de los nodos internos, el puntero en el lado izquierdo de una entrada apunta a subárbol que indexa los registros con una clave asociada menor que la clave de la entrada. Respectivamente, el puntero en el lado derecho apunta al subárbol que indexa los registros con una clave asociada mayor o igual que la clave de la entrada. Los nodos hoja hacen uso de los punteros izquierdo y derecho de una entrada de forma diferente. En este caso, el puntero izquierdo apunta al registro cuyo valor clave asociado coincide con el de la entrada, mientras que el puntero más a la derecha del nodo (el único que no está asociado como un puntero izquierdo a una entrada) apunta al siguiente nodo hoja en el árbol.

El algoritmo de búsqueda de un determinado registro con una clave asociada k en un árbol B^+ es sencillo. Comenzado desde el nodo raíz del árbol, buscamos el nodo hoja que debe contener la entrada asociada a k. Para determinar el camino a dicho nodo hoja seguimos, en cada nodo, el puntero p_i entre las dos entradas consecutivas k_{i-1} y k_i que satisfacen la condición $k_{i-1} \leq k < k_i$. Alternativamente, si la condición anterior no se satisface para ningún par de entradas, seguiremos el puntero p_0 si se satisface la condición $k < k_0$. Finalmente, si la anterior condición no es satisface, seguiremos el puntero p_{2d+1} en caso de que la condición $k \geq k_{2d}$ se satisfaga. Una vez se ha encontrado el nodo hoja correspondiente, buscamos en éste la entrada k_i que satisfaga la condición $k_i = k$. Si ésta existe, el registro asociado a la clave k se corresponde al que está apuntado por p_i. Si no se encuentra la entrada que sea igual a k, no existe

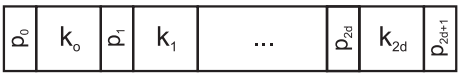

Figura 8.2: Estructura de un nodo de árbol B^+

ningún registro indexado en el árbol que se corresponda con dicha clave. En la descripción anterior, las claves y punteros en los nodos han sido numerados como se muestra en la Figura 8.2.

Los árboles B^+ son estructuras de indexado muy apropiadas para incrementar el rendimiento en la resolución de consultas de rango. El algoritmo para la búsqueda de todos aquellos registros cuyas claves asociadas están en el rango $[k, K]$ es también muy sencillo. en primer lugar buscaremos el nodo hoja de que debe contener la entrada asociada a k como se ha descrito anteriormente para la búsqueda simple. Una vez se ha encontrado el nodo hoja correspondiente, el algoritmo devuelve todos aquellos registros asociados a los punteros p_i para los que se cumpla la condición $k \leq k_i \leq K$. Si la última entrada del presente nodo tiene asociada una clave que cumple la condición $k_{2d} < K$, se recuperará el siguiente nodo hoja empleando el puntero p_{2d+1} y se repetirá el procedimiento descrito anteriormente hasta encontrar una entrada asociada a una clave k_i tal que $k_i \geq K$.

En lo que respecta a los detalles de los algoritmos de inserción y borrado en árboles B^+, los cuales son los responsables de mantener el árbol balanceado, remitimos al lector al trabajo original de Comer [54] ya que para el propósito del presente trabajo éstos no son necesarios.

8.4.4.2. Indexado de datos bidimensionales empleado árboles B^+

Como se ha mencionado en anteriores ocasiones, los árboles B^+ son estructuras de indexado para datos escalares. Dado que el presente trabajo busca emplear dichas estructuras para indexar datos imprecisos numéricos y que éstos se caracterizan como puntos en un espacio bidimensional, se hace necesario encontrar alguna forma que permita adaptar los árboles B^+ para soportar este tipo de datos.

Para conseguir el anterior objetivo, en el presente trabajo se han empleado las siguientes técnicas:

- Indexado de datos multidimensionales por componentes: Dado que un árbol B^+ sólo puede indexar datos escalares, se podrán indexar datos multidimensionales empleando un árbol B^+ por dimensión. En el caso particular de los datos bidimensionales se emplearán dos árboles B^+, uno por cada componente. Ésta es la aproximación que se empleará en el índice denominado 2BPT que describiremos a continuación.

- Indexado de datos multidimensionales por reducción: Una alternativa para permitir que una estructura de indexado de datos escalares pueda soportar datos multidimensionales consiste en el empleo de un método de mapeo que relacione biyectivamente un dato multidimensional con uno escalar. Esta conversión se denomina comúnmente *reducción*, ya que reduce la dimensiones de un dato multidimensional a un número menor (en éste caso a uno solo). Una vez reducidos los datos multidimensionales a escalares,

éstos pueden ser indexados de manera natural en un árbol B^+. Ésta aproximación es la empleada por el índice HBPT que, junto con el método de reducción, será descrito posteriormente en el presente apartado.

De las anteriores técnicas, la elección entre una técnica u otra depende de la naturaleza de las consultas a ejecutar. La reducción de datos multidimensionales, como se explicará en mayor detalle más adelante, muestra un buen rendimiento cuando se resuelven consultas de rango muy selectivas o, lo que es lo mismo, el área de consulta es pequeña. Según el área de consulta del método crece, el rendimiento de éste decrece rápidamente. Por otra parte, el indexado de datos multidimensionales por componentes no es competitivo, con respecto a otros métodos de indexado multidimensional, cuando el área de consulta es pequeña. En cambio, cuando el área de las consultas de rango es amplio el rendimiento de este método se aproxima al de técnicas muy específicas.

Las características, en lo que a rendimiento se refiere, de ambas técnicas las hacen apropiadas, respectivamente, para resolver un tipo específico de condiciones flexibles. Dadas las características de la técnica de indexado de datos multidimensionales por componentes, hacen ésta más apropiada para la resolución de consultas flexibles en que se aplica una medida de posibilidad, donde los principios de indexado estudiados requieren la resolución de una consulta de rango cuya área es amplia. En cambio, para el caso de consultas flexibles que aplican una medida de necesidad, que se resuelven aplicando los principios de indexado estudiados empleando una consulta de rango de área más pequeña que para el caso de posibilidad, la técnica de indexado de datos multidimensionales por reducción se muestra más apropiada.

8.4.4.3. Índice 2BPT

La primera de las propuestas del presente trabajo para el indexado de datos imprecisos difusos la denominaremos 2BPT. Esta técnica de indexado aprovecha las estructuras clásicas de indexado de los árboles B^+ combinándolas de tal forma que sea posible indexar los intervalos que representan los soportes de las distribuciones de posibilidad que modelan los datos imprecisos numéricos.

Concretamente, la técnica de indexado 2BPT indexa un intervalo por medido del uso de dos árboles B^+. Uno de esos árboles B^+ indexa los valores correspondientes al límite inferior de los intervalos indexados por 2BPT. Respectivamente, el otro árbol B^+ indexa los valores correspondientes al límite superior de dichos intervalos.

Dada una condición flexible $\langle \Pi(C/a), \lambda \rangle$, determinaremos su conjunto de tuplas candidatas según los siguientes pasos:

1. Aplicaremos sobre el conjunto de tuplas indexado, empleando el árbol B^+ que indexa al límite inferior de los intervalos que caracterizan los soportes de las distribuciones de posibilidad que modelan los datos imprecisos numéricos indexados, una de las condiciones de rango mostradas en las ecuaciones 8.16 ó 8.17, en función de si el umbral aplicado es o no superior a cero. En caso de que se emplee una función de pertenencia trapezoidal para la definición de C y el umbral aplicado sea cero se aplicará la condición mostrada en la Ecuación 8.18. En dichas ecuaciones, k se corresponde con la clave asociada a los elementos indexados. Dichas condiciones han sido extraídas de los principios descritos en la ecuaciones 8.9, 8.10, 8.13

y 8.14. El conjunto de tuplas que satisface dicha condición será notado como $L_{\langle \Pi(C/a),\lambda \rangle}$.

$$k \geq \text{ínf}\,(L_\lambda(C)) \tag{8.16}$$

$$k \geq \text{ínf}\,(L_{>0}(C)) \tag{8.17}$$

$$k \leq \alpha_{\Pi_{t_i(a)}} \leq \delta_C \tag{8.18}$$

2. De la misma forma, aplicaremos sobre el conjunto de tuplas indexado, empleando en este caso el árbol B^+ que indexa el límite superior de los intervalos que representan los soportes de las distribuciones de posibilidad que modelan los datos imprecisos numéricos indexados, una de las condiciones de rango que se muestran en las ecuaciones 8.19 ó 8.20, según si el umbral aplicado es superior o igual a cero. De forma similar al caso anterior, si se emplea una función de pertenencia trapezoidal para la definición de C y el umbral aplicado sea cero se aplicará la condición mostrada en la Ecuación 8.21. El conjunto de tuplas que satisface dicha condición será notado como $U_{\langle \Pi(C/a),\lambda \rangle}$.

$$k \geq \text{ínf}\,(L_\lambda(C)) \tag{8.19}$$

$$k \geq \text{ínf}\,(L_{>0}(C)) \tag{8.20}$$

$$k \leq \alpha_{\Pi_{t_i(a)}} \geq \alpha_C \tag{8.21}$$

3. Finalmente, el conjunto de tuplas candidatas a satisfacer se compone de aquellas tuplas que satisfacen las dos condiciones de rango anteriormente indicadas. Este conjunto se calcula de forma trivial de la forma que se indica en la Ecuación 8.22.

$$PS_{\langle \Pi(C/a),\lambda \rangle} = L_{\langle \Pi(C/a),\lambda \rangle} \cap U_{\langle \Pi(C/a),\lambda \rangle} \tag{8.22}$$

El siguiente ejemplo ilustra cómo se determina el conjunto candidato de una determinada condición flexible que aplique una medida de posibilidad siguiendo el procedimiento descrito anteriormente.

Ejemplo 8.2. *Supongamos que deseamos indexar las tuplas mostradas en la Tabla 8.2. Las tuplas de dicha tabla poseen dos atributos denominados* rowID *y* att. *El atributo* rowID *hace las veces de clave primaria de la tabla y sus valores son números enteros. Por su parte, el dominio asociado con el atributo* att *es de tipo OAFT y, por tanto, sus valores son imprecisos numéricos definidos mediante funciones características trapezoidales.*

Con el objeto de ilustrar el funcionamiento del mecanismo de indexado propuesto anteriormente, se definirá un índice de tipo 2BPT sobre los valores del atributo att *de la tabla descrita anteriormente. Los intervalos que caracterizan a los valores del atributo* att *(y que por extensión caracterizan a los soportes de*

rowID	att
1	[8,10,10,21]
2	[42,46,47,51]
3	[1,9,15,19]
4	[41,50,55,57]
5	[38,46,47,47]
6	[10,12,12,14]
7	[26,28,30,32]
8	[51,57,60,60]
9	[18,22,27,37]
10	[25,35,39,45]

Tabla 8.2: Tabla de datos de ejemplo

rowID	$[\alpha_{att}, \delta_{att}]$
1	[8,21]
2	[42,51]
3	[1,19]
4	[41,57]
5	[38,47]
6	[10,14]
7	[26,32]
8	[51,60]
9	[18,37]
10	[25,45]

Tabla 8.3: Intervalos que caracterizan los soportes de los valores del atributo *att*

sus distribuciones de posibilidad), se muestran en la Tabla 8.3. Los dos árboles B^+ *de orden 2 (dicho orden se ha seleccionado arbitrariamente para el caso del presente ejemplo) que indexan los límites inferior y superior de dichos intervalos están representados, respectivamente, en las figuras 8.3 y 8.4. En dichas figuras, los punteros de los nodos hoja apuntan a las tuplas cuyo valor para el atributo* rowID *se corresponde con el valor indicado entre paréntesis.*

Una vez descrito el índice que emplearemos, supongamos que deseamos resolver la consulta siguiente:

```
SELECT rowID,att FROM table WHERE FEQ(att,[20,25,35,40])>=0.3
```

La anterior consulta incluye la condición `FEQ(att,[20,25,35,40])>=0.3`*. Ésta, trasladada al marco teórico, se traduce en una condición flexible que aplica la restricción flexible* $\Pi(C_{[20,25,35,40]}/att)$*, donde* $C_{[20,25,35,40]}$ *representa el conjunto difuso de valores definido por la función de pertenencia trapezoidal* $[20, 25, 35, 40]$*, limitando los resultados a aquellos que poseen un grado de cumplimiento asociado igual o superior a* 0.3*.*

Cuando el procesador de consultas del SGBD detecta que existe un índice definido sobre el atributo att*, se inicia el procesamiento de dicha consulta según los pasos siguientes:*

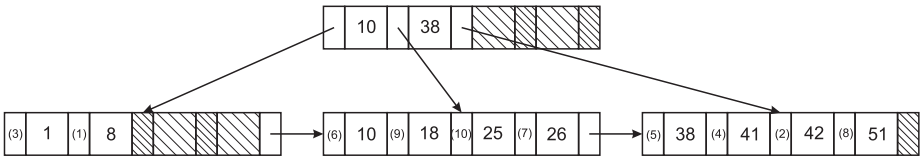

Figura 8.3: Árbol B$^+$ ($d = 2$) que indexa los límites inferiores de los intervalos que caracterizan el soporte de los valores del atributo *att*

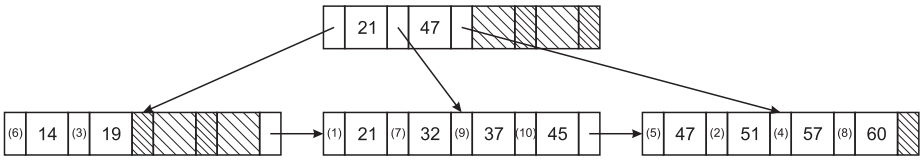

Figura 8.4: Árbol B$^+$ ($d = 2$) que indexa los límites superiores de los intervalos que caracterizan el soporte de los valores del atributo *att*

1. *Determinar los rangos a aplicar sobre los índices: En función del tipo de función de pertenencia de la restricción flexible a aplicar y del umbral aplicado en la condición flexible a resolver, se determina emplear una par de condiciones de rango (una para los valores límite inferior y otra para los valores límite superior de los intervalos que caracterizan los datos indexados) de entre los descritos en las Ecuaciones 8.16 a 8.21. Dado que en el presente ejemplo se emplea un umbral superior a cero, se seleccionarán las condiciones de rango descritas en las ecuaciones 8.16 y 8.19.*

2. *Determinar el conjunto de tuplas* $L_{\langle \Pi(C_{[20,25,35,40]}/att),0.3\rangle}$ *empleando el árbol* B^+ *que indexa los límites inferiores de los intervalos que caracterizan los soportes de las distribuciones de posibilidad que modelan los valores indexados. En este ejemplo, el conjunto* $L_{\langle \Pi(C_{[20,25,35,40]}/att),0.3\rangle}$ *es equivalente a* $\{3, 1, 6, 9, 10, 7, 5\}$*, donde cada tupla está identificada por el valor que ésta posee asociado al atributo* rowID*.*

3. *Determinar el conjunto de tuplas* $U_{\langle \Pi(C_{[20,25,35,40]}/att),0.3\rangle}$ *empleando, en este caso, el árbol* B^+ *que indexa los límites superiores de los intervalos que caracterizan los soportes de las distribuciones de posibilidad que modelan los valores indexados. Para el caso del presente ejemplo, el conjunto* $U_{\langle \Pi(C_{[20,25,35,40]}/att),0.3\rangle}$ *se corresponde con* $\{7, 9, 10, 5, 2, 4, 8\}$*.*

4. *Determinar el conjunto de candidatos para* $\langle \Pi(C_{[20,25,35,40]}/att), 0.3\rangle$*. Éste puede se determinado como la intersección de* $L_{\langle \Pi(C_{[20,25,35,40]}/att),0.3\rangle}$ *y* $L_{\langle \Pi(C_{[20,25,35,40]}/att),0.3\rangle}$*. En este ejemplo,* $PS_{\langle \Pi(C_{[20,25,35,40]}/att),0.3\rangle}$ *equivale al conjunto* $\{9, 10, 7, 5\}$*.*

5. *Determinar el resultado de la consulta. Tras la determinación del conjunto de candidatos de la condición flexible, se ha de proceder a filtrar éstos para excluir aquellos falsos positivos que no satisfacen la restricción flexible aplicada por encima del umbral especificado. En el caso del presente ejemplo, el grado de cumplimiento de cada una de las tuplas del conjunto de candidatos se muestra en la Tabla 8.4. Vistos dichos datos se puede*

rowID	$FEQ(att, [20, 25, 35, 40])$
5	0.1538
7	1
9	1
10	1

Tabla 8.4: Grados de cumplimiento asociados a las tuplas del conjunto de candidatos $PS_{\langle \Pi(C_{[20,25,35,40]}/att), 0.3 \rangle}$

concluir que el resultado de la consulta planteada equivale al conjunto de tuplas $\{7, 9, 10\}$. *Obsérvese que la tupla con valor 5 para el atributo* rowID *ha quedado excluida de este resultado final ya que su grado de cumplimiento asociada está por debajo del umbral.*

8.4.4.4. Índice HBPT

La segunda técnica para el indexado de datos imprecisos numéricos, propuesta en el presente trabajo, será denominada HBPT. Ésta aprovecha la estructura clásica de indexado denominada árbol B^+ [12, 54] y la adapta para permitir el indexado de datos bidimensionales que representan los intervalos que caracterizan los núcleos de las distribuciones de posibilidad que modelan los datos imprecisos numéricos indexados.

Dado que, como se comentó anteriormente, los árboles B^+ son estructuras diseñadas exclusivamente para indexar datos escalares, será necesario adaptar las mismas para hacer posible que éstas indexen datos bidimensionales. Además de la alternativa empleada por 2BPT, que indexa cada componente de los datos en un árbol B^+, es posible indexar éstos datos empleando un sólo árbol B^+. Esta alternativa se basa en la reducción (o mapeo) de datos multidimensionales a escalares para, posteriormente, indexar estos últimos empleando un árbol B^+.

La reducción de datos multidimensionales a escalares puede hacerse con la ayuda de una curva de Peano o *space-filling curve* [134]. Partiendo de una región cuadrada dividida en subregiones cuadradas más pequeñas y del mismo tamaño, una curva de Peano es capaz de, empleando exclusivamente segmentos de línea que conectan subregiones adyacentes, pasar por todas las subregiones que componen todo el cuadrado sin pasar por la misma más de una ocasión. La descripción anterior de una curva de Peano es válida para un espacio bidimensional, suficiente para los propósitos del presente trabajo. En caso de que se consideren espacios n-dimensionales, los cuadrados de la anterior descripción deberán ser reemplazados por hipercubos. La Figura 8.5 muestra varias ilustraciones de regiones cuadradas dividas en subregiones cuadradas de menor tamaño, como la descrita anteriormente, así como una curva de Peano que recorre dichas subregiones.

Las curvas de Peano son útiles para reducir un espacio bidimensional finito, discreto y cuadrado a uno unidimensional equivalente. Los puntos de este tipo de espacio bidimensional pueden ser vistos como subregiones cuadras que lo dividen. Entonces, una curva de Peano puede pasar a través, y una sola vez, de todos los puntos de éste espacio. El orden en que la curva de Peano pasa a través de los puntos induce un orden lineal sobre los mismos. Este orden

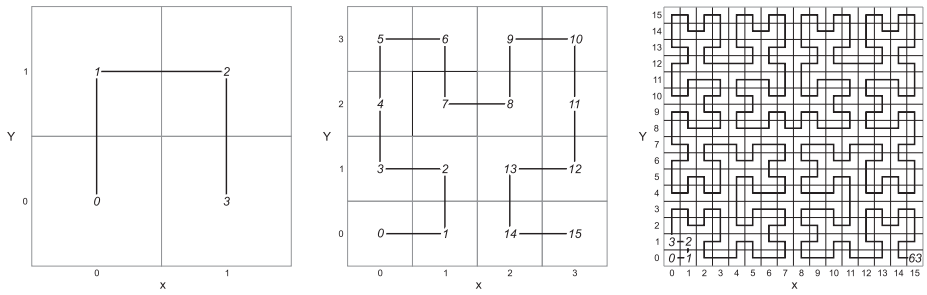

Figura 8.5: Diferentes curvas de Peano del tipo de Hilbert

lineal puede ser empleado para asociar los puntos del espacio bidimensional a una secuencia ordenada de enteros. Cada entero de la anterior secuencia es el equivalente escalar para el punto bidimensional con el que se corresponde.

La técnica de reducción descrita anteriormente se ha ilustrado en la Figura 8.5. En dicha figura, el centro de cada subregión cuadrada es interpretado como un punto bidimensional perteneciente a un espacio bidimensional finito, discreto y cuadrado. Los números en cursiva que se pueden encontrar a lo largo de la curva de Peano que recorre estas regiones se corresponden con la secuencia de enteros en el intervalo $[0, n]$, enteros a los que cada punto bidimensional puede ser reducido empleando el orden lineal inducido por la citada curva de Peano.

Entre la variedad de curvas de Peano existentes [134], en el presente trabajo emplearemos la curva de Hilbert. La anterior decisión ha sido motivada por las excelentes propiedades de agrupamiento (y, por tanto, alto rendimiento) que posee dicha curva en comparación con otras posibilidades populares como la curva de Morton (ó comúnmente *z-order curve*). Las curvas de Peano mostradas en la Figura 8.5 son ejemplos de curvas de Hilbert.

Una vez que los datos bidimensionales han sido reducidos, éstos pueden ser indexados empleado cualquier técnica diseñada para datos escalares [91] como, por ejemplo, un árbol B^+. El procesamiento de una consulta bidimensional de rango empleado la combinación de un árbol B^+ y una curva de Peano es un proceso iterativo que comprende varias consultas de rango unidimensional sobre la estructura de indexado. Cada una de las anteriores iteraciones implica la ejecución de una consulta de rango unidimensional que se corresponde con uno de los segmentos de la curva de Peano dentro de la región definida por la consulta multidimensional.

La Figura 8.6 ilustra el anterior proceso, donde el centro de cada cuadrado que forma parte de la rejilla de color gris representa un punto en un espacio bidimensional, el orden inducido para los puntos de dicho espacio por la curva de Hilbert está representado por una línea continua negra y la condición de rango bidimensional empleada como ejemplo está representada por un rectángulo punteado. En dicha ilustración se puede observar que la curva de Hilbert entra y sale de la región que representa la condición de rango en varias ocasiones. Para encontrar los datos bidimensionales indexados que están incluidos en dicha área es necesario resolver, por cada uno de los segmentos de la curva de Hilbert dentro de dicha área, una consulta de rango unidimensional empleando el árbol B^+ de nuestro índice. Por ejemplo, para el caso de la consulta de rango representa-

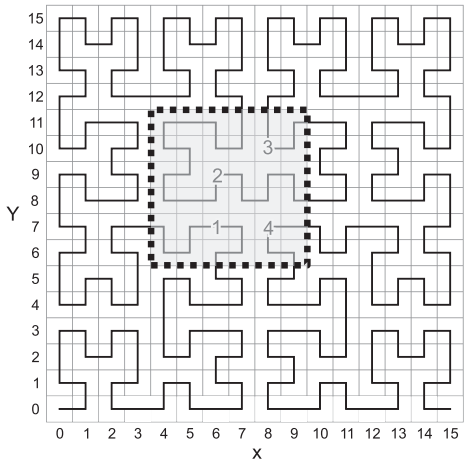

Figura 8.6: Procesamiento de una consulta de rango bidimensional

da en la Figura 8.6, es necesario resolver cuatro consultas unidimensionales de rango empleando el árbol B^+. Estas cuatro consultas se corresponden con los cuatro segmentos de línea numerados en la anterior figura.

Si el método de resolución de consultas de rango multidimensionales descrito anteriormente se aplica directamente su eficiencia es impredecible. Esto es debido a que el número de consultas de rango unidimensionales que es necesario realizar para resolver una consulta de rango multidimensional varía en función del área consultada y a que la eficiencia decrece en función del incremento en el número de éstas. Este decremento en la eficiencia es debido a que, cada vez que se procesa una de las consultas unidimensionales en las que deriva una consulta bidimensional, la mayor parte de los nodos internos del árbol B^+ que se leyeron en anteriores consultas han de ser leídos de nuevo.

Con el objeto de solventar el problema de eficiencia descrito anteriormente, proponemos el empleo de una memoria cache de nodos internos que mantenga el último nodo interno leído de cada nivel del árbol B^+. Teniendo en cuenta que las consultas de rango unidimensionales en que deriva una consulta de rango bidimensional son resueltas en el orden inducido por la curva de Hilbert, no es necesario disponer de una memoria cache de mayor tamaño. El orden con el que se procesan las consultas unidimensionales del ejemplo mostrado en la Figura 8.6 se corresponde con el orden ascendente de los números con el que se han identificado los segmentos de líneas a los que se corresponden. Procesar las consultas unidimensionales en este orden implica que al procesar cada una de ellas, la mayor parte de los nodos internos de árbol B^+ leídos en la anterior consulta serán posiblemente leídos de nuevo. Por tanto, si se emplea la memoria cache propuesta, ésta evitará las lecturas redundantes. El tamaño de la memoria cache necesaria no es muy significativo. Por ejemplo, una base de datos con 10.000.000 registros de las mismas características que los empleados en los experimentos descritos posteriormente en el presente trabajo requiere el mantenimiento de sólo dos nodos internos en el peor de los casos. Si se emplean bloques de disco de 16 KB, el tamaño de la memoria cache será de 32 KB.

8.4.4.5. Árboles G

Un árbol G (originalmente *G-tree* ó *Grid tree*) [88] es una estructura de indexado para datos multidimensionales cuyas componentes se definen sobre dominios ordenados. Ésta combina un directorio (comúnmente un árbol B^+) y un fichero de rejilla (originalmente *Grid file*). Los árboles G soportan tanto la resolución de consultas puntuales como consultas de rango. Esta última cualidad se empleará en la resolución de consultas flexibles sobre datos imprecisos.

Los árboles G organizan los datos indexados en particiones, cada una de ellas correspondiéndose directamente con un bloque de datos (normalmente correspondiéndose éste, a su vez, con un bloque de disco) de tamaño fijo. Dado que cada partición puede contener un máximo de elementos, el número de particiones crece conforme crece el número datos indexados. Inicialmente, el espacio D-dimensional de los datos está dividido en dos mitades a lo largo de la primera de las dimensiones (que denominaremos *dimensión 0*). Cada dato indexado en el árbol G se corresponderá, según su posición en el espacio D-dimensional dividido, con una de esas dos particiones y será almacenado en el bloque datos asociado a ésta. Cuando una de las particiones llega a su máxima capacidad y una nueva inserción provoca su desbordamiento, ésta es dividida en dos mitades a lo largo de la dimensión $(d+1) mod D$, donde d es la dimensión por la que fue dividida la partición padre de la que ésta proviene. Cada una de éstas mitades será una nueva partición asociada a un nuevo bloque de datos y los datos indexados en la partición padre se distribuirán convenientemente entre cada una de éstas. El anterior proceso se repetirá, mientras sea necesario, hasta que ninguna de las particiones se encuentre desbordada.

Cada partición será identificada unívocamente mediante una cadena binaria. Esta cadena se creará partiendo de la cadena que identifica al padre de la partición y anexándole a ésta un 0, si la partición es el hijo menor de su partición padre (con respecto a la dimensión por la que ésta ha sido dividida), o 1, si ésta es el hijo mayor.

Ejemplo 8.3. *En la Figura 8.7 se ilustra el esquema de particionado e identificación descrito anteriormente para datos bidimensionales y un tamaño máximo de bloque de datos de 2 elementos. La Figura 8.7a ilustra las particiones iniciales con 3 elementos indexados. Las figuras 8.7b, 8.7c y 8.7d ilustran, respectivamente, las particiones resultantes tras la división de las originales debido a un desbordamiento por la inserción de nuevos elementos en las particiones* 0, 01 *y* 011.

Cada vez que un elemento es insertado, eliminado o recuperado en un árbol G, se ha de repetir el mismo procedimiento descrito a continuación:

1. Determinar la partición a la que pertenece el elemento.
2. Recuperar el bloque de datos asociado a la partición e insertar, eliminar o recuperar el elemento según el caso.

El primero de los pasos anteriores puede realizarse rápidamente empleando operaciones geométricas, por lo que no será necesario que ninguna información al respecto sea almacenada y por tanto leída. En cambio, el segundo paso requiere una estructura de datos asociativa, que denominaremos *directorio*, que relacione cada partición con su bloque de datos correspondiente. En este caso, el directorio

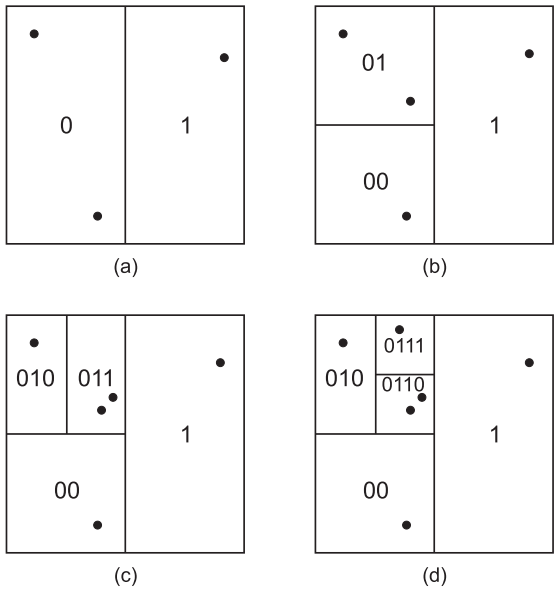

Figura 8.7: Esquema de particionado e identificación de un árbol G

será un árbol B^+ donde las claves de indexado serán los identificadores de las particiones. Para habilitar este mecanismo de indexado será necesario que exista un criterio de orden entre las claves de indexado. En este caso, el criterio de orden que se aplicará sobre los identificadores de particiones se corresponde con el definido en la Ecuación 8.23, donde las particiones p_1 y p_2 tienen asociado, respectivamente, un identificador de longitud b_1 y b_2 bits, b es un valor definido como $b = min(b_1, b_2)$ y $MSB(p, b)$ representa a los b bits más significativos del identificador asociado a la partición p.

$$p_1 > p_2 \iff MSB(p_1, b) > MSB(p_2, b) \tag{8.23}$$

Ejemplo 8.4. *En la Figura 8.8 se muestra una representación gráfica del árbol B^+ empleado como directorio para el árbol G que indexa los elementos (o puntos bidimensionales) representados en la Figura 8.7d.*

Los árboles G son particularmente adecuados para la resolución de consultas de rango. Este tipo de consultas en un contexto multidimensional se equipara a determinar, dado un hiper-rectángulo q, todos los puntos indexados que se encuentren en su interior. Éste tipo de consultas se resuelve según el siguiente procedimiento:

1. Determinar las particiones p_l y p_u que deberían, respectivamente, contener a los puntos l y u que se corresponden con la esquina inferior y superior de q. Esta operación puede realizarse geométricamente y, por tanto, no necesita acceder a ningún dato del índice.

2. Determinar el conjunto de particiones $P = \{p_i | p_l \leq p_i \leq p_u\}$ (y los punteros a sus bloques de datos) empleando una búsqueda de rango sobre el árbol B^+ que implementa el directorio del índice.

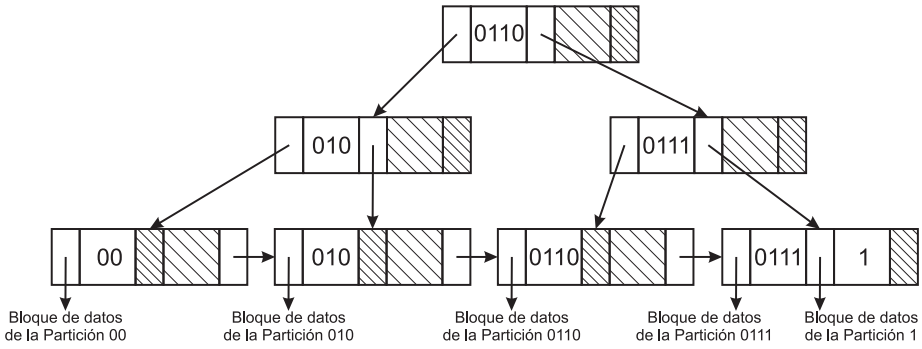

Figura 8.8: Directorio de acceso a los bloques de datos asociados a las particiones de un árbol G

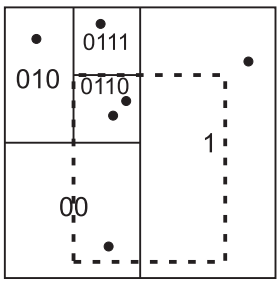

Figura 8.9: Consulta de rango

3. Por cada $p_i \in P$:

   a) Si la partición p_i está contenida en q, incluir en el conjunto resultante todos los elementos indexados pertenecientes a dicha partición que se encuentran almacenados en su bloque de datos.

   b) Si p_i interseca a q pero no está totalmente contenida en ésta, por cada elemento indexado perteneciente a dicha partición se comprobará si éste está contenido en q. En caso afirmativo, el elemento se incluirá en el conjunto resultante.

   c) En caso de que la partición p_i no interseque a q, ésta será descartada.

El siguiente ejemplo ilustra las resolución de una consulta de rango empleando un árbol G según el procesamiento descrito anteriormente.

Ejemplo 8.5. *Supongamos que deseamos recuperar de entre el conjunto de puntos incluidos en la Figura 8.7d aquellos que están incluidos en la consulta de rango que se representa gráficamente empleando una línea punteada en la Figura 8.9. Para ello, supondremos que dichos puntos están indexados por el árbol G descrito en los ejemplos 8.3 y 8.4.*

En primer lugar, determinaremos las particiones p_l y p_u asociadas a la esquina inferior y superior del hiper-rectángulo que define la consulta. En el caso del presente ejemplo, éstas son $p_l = 00$ y $p_u = 1$.

Seguidamente, determinaremos el conjunto de particiones $P = \{p_i | p_l \leq p_i \leq p_u\}$ empleando el árbol B^+ mostrado en al Figura 8.8 que, como se mencionó

anteriormente, hace las veces de directorio del árbol G. Para el presente ejemplo, el conjunto P está definido como $P = \{00, 010, 0110, 0111, 1\}$.

En tercer lugar comprobaremos si cada una de las particiones en P interseca el hiper-rectángulo de consulta. Como resultado de esta comprobación las particiones 010 *y* 0111 *son excluidas de cualquier procesamiento posterior ya que no intersecan dicha área. En lo que respecta a la partición* 0110*, todos los puntos indexados que se encuentran dentro de ella serán incluidos en el conjunto resultante de la consulta ya que ésta está totalmente incluida en el hiper-rectángulo que define la condición de rango. Finalmente, los puntos incluidos en las particiones* 00 *y* 1 *deberán ser comprobados uno a uno para determinar si éstos están dentro del hiper-rectángulo que define la condición de rango. Como resultado de ésta comprobación, el único punto perteneciente a la partición* 00 *es incluido en el conjunto de resultados de la consulta, ya que está contenido en el hiper-rectángulo que define la condición de rango, y el punto perteneciente a la partición* 1 *será descartado ya que no se encuentra dentro de éste.*

Remitimos al lector a [88] para obtener detalles sobre los algoritmos de inserción y borrado en árboles G, ya que éstos quedan fuera de ámbito del presente trabajo.

8.4.5. Evaluación del rendimiento de las propuestas

Con el objetivo de evaluar el rendimiento de las técnicas de indexado propuestas, así como la influencia sobre éste de diferentes factores, y comparar éste con el ofrecido por la técnica de referencia, se ha de realizar una evaluación cuantitativa de dicho rendimiento. El presente apartado describe los experimentos llevados a cabo para realizar dicha evaluación. Estos experimentos evaluarán el rendimiento de las técnicas de forma genérica y en distintos escenarios particulares. Para ello, se detallará la naturaleza de la medida empleada y los factores estudiados que puedan incidir en una variación en el rendimiento de éstas.

8.4.5.1. Medida de rendimiento

El rendimiento de un índice puede ser evaluado como el ahorro de tiempo que resulta de su aplicación cuando se procesa una consulta en contraposición al caso en que ésta es procesada sin la ayuda del índice. Este ahorro de tiempo es inversamente proporcional al tiempo requerido por el índice para resolver una determinada consulta, que puede ser dividido en el tiempo necesario para acceder a los datos del índice y el requerido para el procesamiento de los mismos.

Los datos de un índice, especialmente en el caso de que éste sea de gran tamaño, son almacenados usualmente en memoria secundaria. Por tanto, el tiempo necesario para acceder a los datos de índice es mucho mayor que el tiempo necesario para procesarlos. Además, se ha de notar que esta diferencia se incrementa continuamente ya que la potencia de computación evoluciona más rápido que el ancho de banda de los dispositivos de memoria secundaria. Debido a las razones anteriores, el tiempo de procesado no es, comúnmente, tenido en cuenta en la evaluación del rendimiento de índices y sólo el tiempo de acceso a los datos es considerado.

El tiempo requerido para acceder a los datos de un índice es claramente dependiente de la cantidad de datos accedidos. Los dispositivos de memoria

secundaria son dispositivos de bloques, por lo que el bloque de datos es su unidad de transferencia. El tiempo requerido para acceder a los datos en este tipo de dispositivos se divide en el tiempo necesario para localizar los bloques de datos, que denominaremos *tiempo de acceso*, en que éstos están almacenados y el tiempo necesario para leer dichos bloques. El tiempo de lectura de los bloques es proporcional al número de bloques leídos. En cambio, el tiempo de acceso a los bloques depende fuertemente de la distribución de los bloques en el medio de almacenamiento y de la tecnología que emplea el dispositivo para acceder a los mismos. Además, teniendo en cuenta que un SGBD se ejecuta sobre un sistema operativo (SO), los mecanismos de acceso a disco de este último se han de tener en cuenta para determinar el tiempo de lectura de los datos de un índice. Los SOs incluyen, comúnmente, técnicas que emplean memoria cache para reducir el tiempo de lectura de datos desde memoria secundaria. Por tanto, el tiempo de acceso a los datos de un índice dependerá fuertemente de tipo de memoria cache empleada y del estado de ésta en cada momento.

Con el objeto de realizar una medida independiente y fiable del rendimiento de las técnicas de indexado consideradas, y teniendo en cuenta las observaciones realizadas anteriormente, en el presente trabajo emplearemos como medida de rendimiento el número de bloques de datos accedidos por la técnica de indexado evaluada. De esta forma, la medida de rendimiento es independiente de cualquier las particularidades del hardware y SO empleado, así como del estado previo del sistema a la hora de realizar las mediciones.

8.4.5.2. Factores influyentes en el rendimiento de los índices

El rendimiento de las técnicas de indexado de datos imprecisos está afectado por una gran multitud de factores. Por una parte, éste se ve afectado por un conjunto de factores físicos y lógicos que afectan a las técnicas clásicas de indexado sobre los que éstos se basan y que ha de ser tenido en cuenta. Aunque alguno de estos factores podrían incrementar el rendimiento del índice si son convenientemente ajustados, éstos son básicamente dependientes del hardware o de casos particulares de configuración, por lo que su estudio no proporcionará ninguna información sobre el rendimiento general, y bajo condiciones generales, de las técnicas de indexado evaluadas. Entre éstos factores, el tamaño de bloque empleado es el factor el más crítico para las técnicas de indexado consideradas.

Por otra parte, existe un conjunto de factores relacionados con la naturaleza de los datos indexados y consultas procesadas que afectarán al rendimiento de los índices. Estos factores no pueden ser normalmente ajustados en sistemas reales y el estudio de su influencia proporcionará una buena medida del rendimiento de los índices evaluados bajo diferentes condiciones de uso.

El principio de indexado, en el que se basan las técnicas evaluadas, determina el conjunto de candidatos de una determinada condición flexible comprobando si el soporte de las distribuciones de posibilidad que representan los elementos indexados contiene elementos comunes con la base de la condición flexible, en el caso de que se aplique una medida de posibilidad, o si el núcleo de las distribuciones de posibilidad que representan los elementos indexados es un subconjunto de la base de la condición flexible, si se aplica una medida de necesidad. Estas operaciones, en el caso de indexado de datos imprecisos numéricos, se equiparan a comprobar si los intervalos que representan dichos conjuntos se solapan o se contienen, según el caso. La longitud de dichos intervalos y el número de elemen-

tos indexados afectará la cantidad de datos de índice que han de ser examinados para resolver una consulta dada y, por tanto, al rendimiento del índice.

La extensión, tanto los intervalos que caracterizan el soporte y el núcleo de las distribuciones de posibilidad que modelan los datos imprecisos indexados, como del intervalo que caracteriza la base de la condición flexible procesada, dependen de la forma que adopta la función característica de las distribuciones de posibilidad o conjuntos difusos, según el caso. Dado que emplearemos funciones características trapezoidales, ésta forma puede ser caracterizada como la longitud del intervalo que representa el soporte y la longitud del intervalo que representa el núcleo. La extensión del soporte supone un límite superior para la extensión de la base de la condición y para el núcleo. La longitud del núcleo determinará la tasa de reducción de la extensión de la base de la condición conforme aumenta el umbral aplicado. Esto, además, evidencia que el umbral aplicado es un también un factor influyente.

Con el objeto de definir de forma general los anteriores descriptores de forma, proponemos el uso de dos grados denominados *longitud relativa del soporte* (LRS) y *longitud relativa del núcleo* (LRN) [9].

La primera de las medidas, la LRS de un conjunto difuso F definido por la función de pertenencia $[\alpha_F, \beta_F, \gamma_F \delta_F]$ se determina como se indica en la Ecuación 8.24, donde D representa el dominio subyacente del conjunto difuso, y $\text{ínf}(D)$ y $\sup(D)$ representan, respectivamente, el ínfimo y el supremo de los valores del dominio.

$$LRS(F) = \frac{\delta_F - \alpha_F}{\sup(D) - \text{ínf}(D)} \tag{8.24}$$

La medida LRN indicará la rapidez de transición desde el soporte al núcleo. Ésta se calcula en base a la extensión relativa del núcleo con respecto al soporte de un conjunto difuso y se define como se indica en la Ecuación 8.25.

$$LRN(F) = 1 - \frac{\gamma_F - \beta_F}{\delta_F - \alpha_F} \tag{8.25}$$

En conclusión, se han identificado los siguientes seis factores influyentes en el rendimiento de los índices evaluados: La cantidad de datos indexados, las medidas LRS y LRN de las distribuciones de posibilidad que modelan los datos, el umbral de cumplimiento de las condiciones flexibles asociadas y las medidas LRS y LRN de la función característica que modela su restricción difusa asociada.

8.4.6. Experimentos

Con el objetivo de medir el rendimiento global así como el rendimiento bajo diferentes condiciones relativas a datos y consultas, el mismo conjunto de experimentos, empleando el mismo conjunto de bases de datos y consultas, ha sido realizado empleando cada uno de los mecanismos de indexado propuestos en el presente trabajo así como el mecanismo de indexado empleado como referencia. El presente apartado está dedicado a describir tales experimentos y las condiciones bajo las cuales éstos han sido realizados.

8.4.6.1. Aislamiento de factores físicos y lógicos

Con el objeto de aislar los experimentos de la influencia de factores físicos y lógicos relacionados exclusivamente con el ajuste de las estructuras de indexado que emplean internamente los métodos propuestos, todas ellas comparten los mismos valores fijos para dichos parámetros. El tamaño de los bloques de datos, y por extensión el de los nodos de los árboles, ha sido fijado a 16 KB, coincidiendo con el tamaño de bloque que emplea la plataforma sobre la que los experiments han sido ejecutados. En lo que respecta a las entradas de los índices, cada puntero tendrá un tamaño asignado de 8 bytes y los valores numéricos (como claves y componentes) serán representados empleado una codificación en punto flotante de 8 bytes.

8.4.6.2. Rendimiento global

Con el objetivo de evaluar el rendimiento global de los métodos considerados se ha realizado un mismo experimento sobre una serie de diferentes bases de datos. El conjunto de datos del experimento está compuesto por 30 bases de datos, cada una de ellas generada aleatoriamente bajo una distribución uniforme dentro del intervalo $[-1.000.000, 1.000.000]$. El tamaño de las bases de datos empleadas se ha fijado en 10.000, 20.000, 40.000, 80.000 y 160.000 elementos, según el caso. Como resultado, seis bases de datos del conjunto de datos del experimento comparten el mismo tamaño. Aprovechando tal circunstancia, se analizará en la misma prueba la influencia del tamaño de la bases de datos indexada. El experimento, en su conjunto, emplea 1.860.000 tuplas indexadas.

El conjunto de consultas de la prueba está compuesto por 10.000 consultas, cada una de ellas compuesta por una única condición flexible. Cada condición flexible empleada está compuesta por una restricción flexible que ha sido generada aleatoriamente empleando el mismo generador aleatorio que en el caso de los datos imprecisos numéricos indexados y un valor de umbral generado aleatoriamente bajo una distribución uniforme en el rango $[0, 1]$.

El experimento descrito, evaluará la eficiencia global de las técnicas consideradas como la media del rendimiento medido al ejecutar el conjunto de consultas del experimento sobre cada una de las bases de datos consideradas.

8.4.6.3. Rendimiento bajo diferentes circunstancias relativas a los datos

Para evaluar el rendimiento de las técnicas propuestas bajo diferentes condiciones relativas a los datos, se ha realizado un conjunto de pruebas similares a las descritas anteriormente para el caso de la evaluación del rendimiento global, una por cada valor fijado para uno de los factores estudiados, sobre un conjunto de datos modificado.

La pruebas realizadas se dividen en dos experimentos. El primero busca determinar la influencia de la medida LRS de los elementos indexados y el segundo evaluará la influencia de la medida LRN de éstos. En cada prueba, el conjunto de datos original ha sido modificado fijando la medidas LRS ó LRN de todos sus elementos indexados a un valor determinado.

En el primero de los experimentos, la medida LRS se ha fijado en cada una de las pruebas a uno de los valores en el rango 0 a 0.9 aplicando incrementos de 0.1 (esto es, valores de la serie $0, 0.1, 0.2, \ldots, 0.9$). El valor 1.0 para la medida

LRS ha sido excluido ya que su aplicación implicará que el soporte de todos los datos generados aleatoriamente sería equivalente a la extensión del dominio subyacente y, por tanto, el mismo para todos los casos. Según la descripción anterior, el presente experimento está compuesto por 10 pruebas aplicadas sobre 30 bases de datos (incluyendo en total 18.600.000 tuplas).

De la misma forma que se ha realizado el anterior experimento, el segundo de ellos sigue la misma metodología pero fijando, en cada una de las pruebas, los valores de la medida LRN. Estos valores se extraerán del rango 0 a 1 aplicando incrementos de 0.1. Visto lo anterior, éste experimento está compuesto por 11 pruebas que se aplican sobre el conjunto de bases de datos modificado (sobre un total de 20.460.000 tuplas).

8.4.6.4. Rendimiento bajo diferentes circunstancias relativas a las consultas

La influencia de los diferentes factores relativos a las consultas ha sido evaluada empleando una metodología similar a la descrita en el apartado anterior para el caso de los factores relativos a los datos.

Para asegurar que la totalidad del espectro de los factores estudiados ha sido considerado, el conjunto de consultas modificado de cada prueba ha sido generado fijando sólo el valor de uno de dichos factores. De esta forma, las restricciones flexibles asociadas a las condiciones flexibles han sido generadas fijando un valor para su medida LRS o LRN dentro del rango $[0, 1]$ aplicando incrementos de 0.1. De igual manera, al estudiar la influencia del umbral de las condiciones flexibles en el rendimiento de las técnicas evaluadas, el umbral asociado a las condiciones flexibles ha sido fijado a un valor dentro del rango $[0, 1]$ aplicando incrementos de 0.1.

Según la descripción anterior, se han generado 33 conjuntos de consultas para la realización de las pruebas que componen el conjunto de experimentos descrito. Cada prueba consistirá en la aplicación de uno de estos conjuntos sobre las mismas 30 bases de datos empleadas en el experimento para la evaluación del rendimiento global de las técnicas propuestas.

El conjunto de experimentos descrito considera un total de 330.000 consultas y 9.900.000 evaluaciones de consultas cuando éstas son aplicadas sobre las 30 bases de datos consideradas.

8.4.7. Resultados

El presente apartado está dedicado a detallar los resultados obtenidos de los experimentos descritos. En primer lugar, se expondrán los resultados relativos al rendimiento global de la técnicas estudiadas y posteriormente los resultados sobre la influencia de los distintos factores considerados sobre datos indexados y consultas. Éstos resultados serán primero expuestos para el caso de la aplicación de medidas de posibilidad (lo que indicará el rendimiento de 2BPT) y posteriormente para el caso de medidas de necesidad (lo que indicará el rendimiento de HBPT).

8.4.7.1. Rendimiento global

Los resultados de los experimentos previos indican una eficiencia media, para el caso procesamiento de consultas flexibles que aplican una medida de posibilidad, de 90.13 bloques leídos por consulta, con una desviación típica de 13.71 bloques, para el método 2BPT y de 85.35 bloques leídos por consulta, con una desviación típica de 29.15 bloques, para el método de referencia GT. La diferencia de rendimiento medio entre métodos indica que 2BPT requiere un 5.6 % más de accesos que GT, aunque se ha de tener en cuenta que la desviación típica de la media para el método GT supera el 34 %. Esta última cantidad está muy por encima de la diferencia existente entre 2BPT y GT y duplica a la desviación registrada para 2BPT (de un 15.2 %). La amplitud de las fluctuaciones en las medidas de rendimiento se explicarán posteriormente una vez estudiada la influencia de los distintos factores relativos al conjunto de datos indexado.

En el caso de procesamiento de consultas flexibles que aplican una medida de necesidad, los resultados de los experimentos arrojan una eficiencia media para HBPT de 21.59 bloques leídos, con una desviación típica de 0.97 bloques, y de 22.50 para GT, con una desviación típica de 0.23 bloques. Estos valores suponen que el rendimiento de HBPT es superior en un 4.2 % al registrado para el método de indexado de referencia.

Se ha de destacar que los valores de medidas y desviaciones típicas indicados en el presente apartado proceden de la media de los valores obtenidos por cada tamaño de base de datos empleado. Dado que los valores de rendimiento se basan en el número de bloques leído y que este número es muy distinto en función del tamaño de la base de datos, no parece lógico obtener un rendimiento medio (y especialmente los valores de la desviación típica) sin discriminar previamente el tamaño de la base de datos empleada.

8.4.7.2. Rendimiento bajo diferentes circunstancias relativas a los datos

Este apartado se expondrán y analizarán los experimentos realizados para determinar la influencia de los distintos factores relativos al conjunto de datos indexado. En primer lugar, se analizará el papel que juega el tamaño de la base de datos y posteriormente los valores de las medidas LRS y LRN de éstos.

Influencia del tamaño de la base de datos.

El primero de los factores considerados es el tamaño de la base de datos indexada. Para el caso en que se aplica una medida de posibilidad, la Figura 8.10 presenta el rendimiento medido de cada uno de los índices. La figura presenta un rendimiento similar de ambos índices, entrecruzándose las gráficas de éstos, con una diferencia máxima entre ellos ligeramente superior al 3 %. Aunque difícil de apreciar en dicha figura, debido a que ésta presenta valores medios, el rendimiento medido para GT fluctúa ligeramente. Estas fluctuaciones explican las diferencias observadas con 2BPT y serán analizadas, en detalle, en el siguiente apartado. La media de rendimiento de 2BPT para el conjunto de pruebas ha sido inferior en un 1.3 % con respecto a la medida para GT.

En lo que respecta al caso en que se aplica una medida de necesidad, el rendimiento de las técnicas evaluadas se muestra en la Figura 8.11. En dicha figura, se observa como HBPT se distancia paulatinamente de GT, llegando a

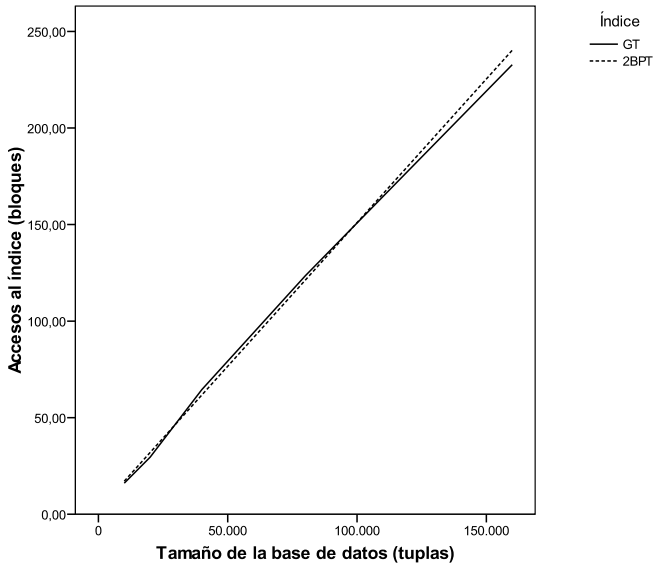

Figura 8.10: Rendimiento de las técnicas para el indexado de datos imprecisos numéricos evaluadas bajo distintos tamaños de base de datos para consultas que aplican una medida de posibilidad

una diferencia máxima del 7 %. Los resultados muestran una ligera superioridad de rendimiento para HBPT debido, presumiblemente, a su tasa de crecimiento sublineal. En media, para la totalidad de las pruebas, el rendimiento de HBPT es superior a la de GT en un 4.2 %.

Se ha de notar sobre las gráficas anteriores, que la tasa de crecimiento de número de bloques de índice leídos se debe a dos factores fundamentales. El primero de ellos es el incremento de número de resultados debido al incremento en el número de elementos indexados. Este factor afecta a las técnicas evaluadas por igual. El segundo factor es el incremento de tamaño de las estructuras de indexado. Este segundo factor afecta de forma diferente a las técnicas evaluadas y es el responsable de las diferencias registradas.

Influencia de la medida LRS de los datos indexados.

Los resultados obtenidos en las pruebas descritas anteriormente para evaluar la influencia de la medida LRS de los datos indexados se muestran, para el caso en que se aplica una medida de posibilidad, en la Figura 8.12. En dicha figura, se puede apreciar la superioridad de GT para los casos en que la medida LRS es baja. Esta diferencia es notable en el caso extremo en que la medida LRS es igual a 0, y por tanto se están indexando datos crisp. En dicho caso, la diferencia entre GT y 2BPT alcanza el 70 %. No obstante, una vez fuera del caso extremo, el rendimiento de 2BPT se asemeja, y en ocasiones supera, al de GT. En dicho rango, la mayor diferencia de rendimiento entre ambos métodos es menor de un 15 %, reduciéndose ésta rápidamente al 10 % en el siguiente nivel de LRS y al 2.2 % en el nivel posterior. La diferencia media de rendimiento para la totalidad de pruebas realizadas es de un 5.3 %.

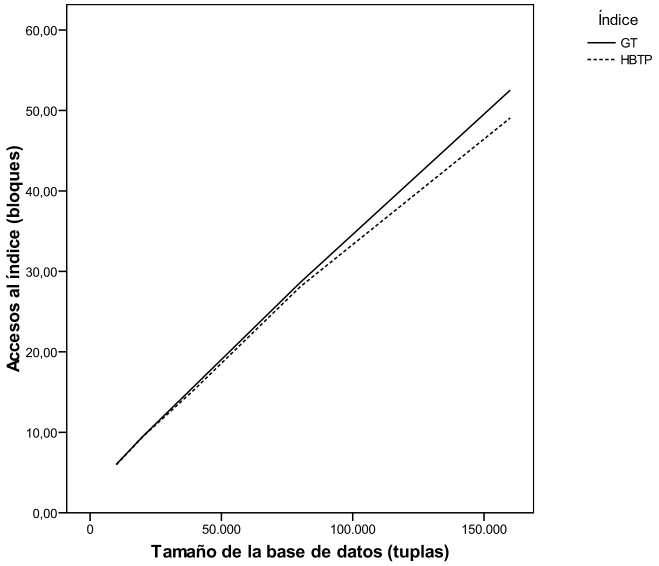

Figura 8.11: Rendimiento de las técnicas para el indexado de datos imprecisos numéricos evaluadas bajo distintos tamaños de base de datos para consultas que aplican una medida de necesidad

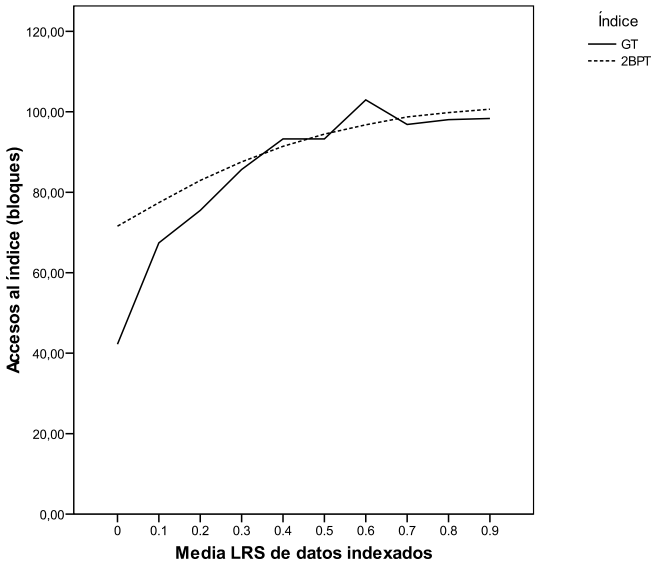

Figura 8.12: Rendimiento de las técnicas para el indexado de datos imprecisos numéricos evaluadas bajo distintos niveles de LRS de los datos indexados para consultas que aplican una medida de posibilidad

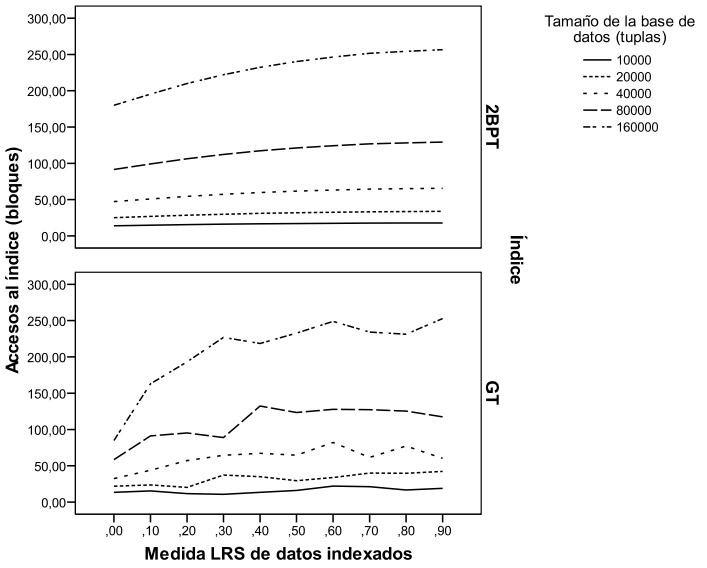

Figura 8.13: Rendimiento de las técnicas para el indexado de datos imprecisos numéricos evaluadas bajo distintos tamaños de base de datos y niveles de LRS de los datos indexados para consultas que aplican una medida de posibilidad

En la anterior figura, se vuelven a observar las fluctuaciones del rendimiento de GT, al igual que en el caso de la Figura 8.10, siendo más evidentes en el presente caso. Esta inestabilidad contrasta, además con la estabilidad observada para el método 2BPT en ambas figuras. De hecho, en el caso de la Figura 8.10 las fluctuaciones del rendimiento de GT son menos aparentes debido a que las gráficas muestran el comportamiento medio bajo diferentes niveles de LRS de los datos indexados. Un análisis más detallado del rendimiento de los métodos bajo distintos niveles de LRS y cantidad de datos indexados, mostrado en la Figura 8.13, revela las citadas fluctuaciones una vez anulado el efecto compensador de la media. En dicha figura, se observa, además, cómo las mencionadas fluctuaciones se hacen más notables con forme el tamaño de la base de datos es mayor. En lo que respecta a 2BPT, se puede observar la estabilidad del rendimiento del método con independencia de las condiciones relativas a los datos indexados.

La diferencia en la estabilidad de estos métodos se debe a la fuerte dependencia con respecto a los factores relativos a los datos de la tasa de uso de bloques de GT, como se muestra en la Figura 8.14. En dicha figura, se muestra la total estabilidad de la tasa de uso de bloques de 2BPT en contraste con las amplias fluctuaciones en dicha tasa de GT. A diferencia de 2BPT, que garantiza un uso mínimo de los bloques del 50 %, GT no garantiza un uso mínimo de los bloques empleados en la estructura del índice, lo que origina la situación descrita. Además de lo anterior, la falta de una tasa de uso mínima de GT puede originar que sus estructuras de índice requieran mayor espacio en memoria secundaria que en el caso de GT. De hecho, en los experimentos realizados, la tasa media de uso de bloques de GT es de 0.52, cuando la de 2BPT es de 0.72.

Nótese que la tendencia ascendente en el número de acceso al índice obser-

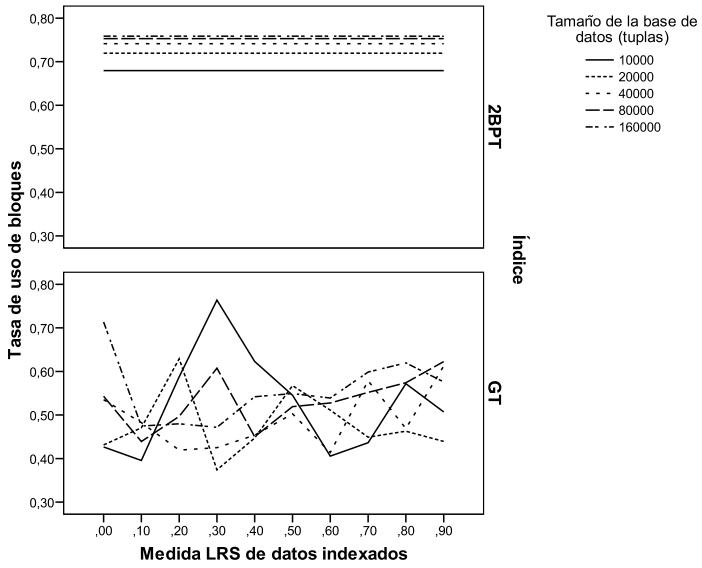

Figura 8.14: Tasa de uso de bloques de las técnicas para el indexado de datos imprecisos numéricos evaluadas bajo distintos tamaños de base de datos y niveles de LRS de los datos indexados para consultas que aplican una medida de posibilidad

vada en la Figura 8.12 para ambos métodos evaluados se debe al incremento del número de resultados de las consultas provocado por la mayor extensión de los soportes de los datos indexados.

En lo que respecta al caso en que se aplican medidas de necesidad, los resultandos de los experimentos para evaluar el rendimiento de las técnicas de indexado consideradas bajo diferentes niveles LRS de los datos indexados se muestran en la Figura 8.15. En dicha figura, se puede apreciar que ambos índices, GT y HBPT, siguen una misma tendencia. Si se obvian las notables fluctuaciones de GT, se puede apreciar que el rendimiento medio de éstos métodos es similar. De hecho, para la totalidad de las pruebas practicas, el rendimiento medio medido para HBPT es un 2 % superior al de GT.

Al igual que en el caso en que se aplican medidas de posibilidad, las fluctuaciones de GT vienen provocadas por la sensibilidad de GT a las condiciones relativas a los datos indexados. La Figura 8.16 muestra la fluctuaciones de la tasa de uso de bloques de GT en función de los distintos niveles de LRS de los datos indexados y la estabilidad de HBPT ante las variaciones en este factor.

Influencia de la medida LRN de los datos indexados.

Los resultados anteriores muestra una significativa influencia de la medida de LRS de los datos indexados en el rendimiento de las técnicas de indexado evaluadas. Dado que la medida LRS sólo influye indirectamente sobre la longitud del núcleo de los datos indexados, es de esperar que el la medida LRN de éstos ejerza una influencia análoga.

Los resultados obtenidos en el experimento dedicado a evaluar la influencia

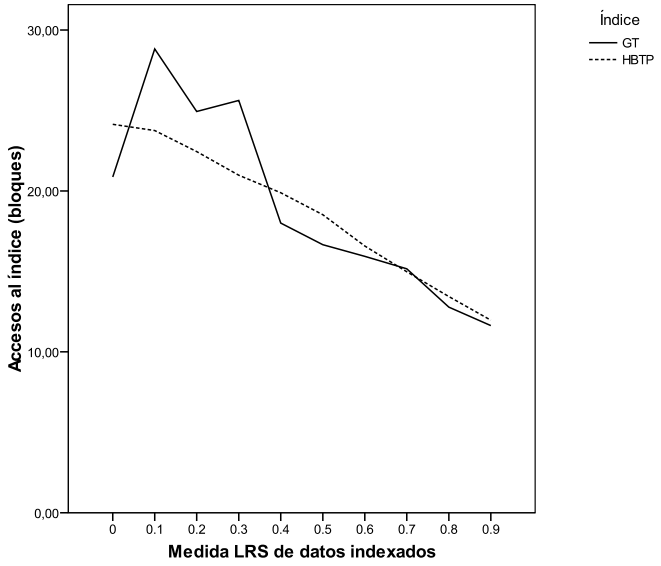

Figura 8.15: Rendimiento de las técnicas para el indexado de datos imprecisos numéricos evaluadas bajo distintos niveles de LRS de los datos indexados para consultas que aplican una medida de necesidad

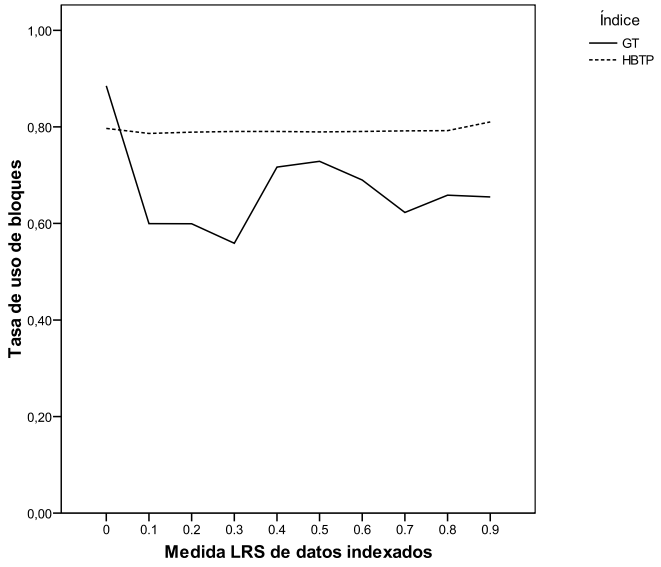

Figura 8.16: Tasa de uso de bloques de las técnicas para el indexado de datos imprecisos numéricos evaluadas bajo distintos niveles de LRS de los datos indexados para consultas que aplican una medida de necesidad

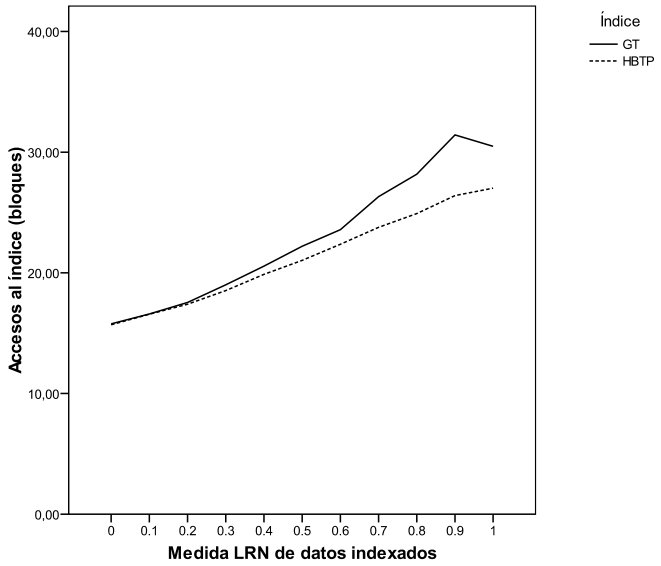

Figura 8.17: Rendimiento de las técnicas para el indexado de datos imprecisos numéricos evaluadas bajo distintos niveles de LRN de los datos indexados para consultas que aplican una medida de necesidad

de la medida LRN de los datos indexados, para el caso en que se aplica una medida de necesidad, confirman tal conjetura y se muestran en la Figura 8.17. En dicha figura, se puede observar cómo el rendimiento de HBPT es igual o, en la mayor parte de los casos, superior al de GT para todo el espectro de niveles de LRN de los datos indexados, siendo el rendimiento medio de la primera técnica superior en un 7.7% al de la segunda. La tendencia descendente de amabas curvas es debida a la reducción del número de resultados de las consultas debido a que los núcleos de las distribuciones de posibilidad que modelan los datos indexados son más amplios.

En la figura anterior, se puede observar de manera notable las fluctuaciones en el rendimiento del índice GT y la estabilidad, por contra, del método HBPT. Al igual que en el caso anterior, en que se estudió la influencia de la medida LRS de los datos indexados, estas fluctuaciones son debidas a la sensibilidad de GT a las características de los datos indexados que provoca grandes fluctuaciones en su tasa de uso de bloques, tal y como se muestra en la Figura 8.18.

Se ha de hacer notar que el presente apartado no expone resultado alguno acerca de la influencia de la medida LRN de los datos indexados para el caso en que se aplica una medida de posibilidad. Esto se debe a que dicha medida no influye de ninguna manera en los valores que caracterizan los datos indexados cuando se aplican este tipo de medidas (los intervalos que caracterizan el soporte de los datos indexados). Por tanto, la influencia de este factor sobre el rendimiento de los mismos es nula.

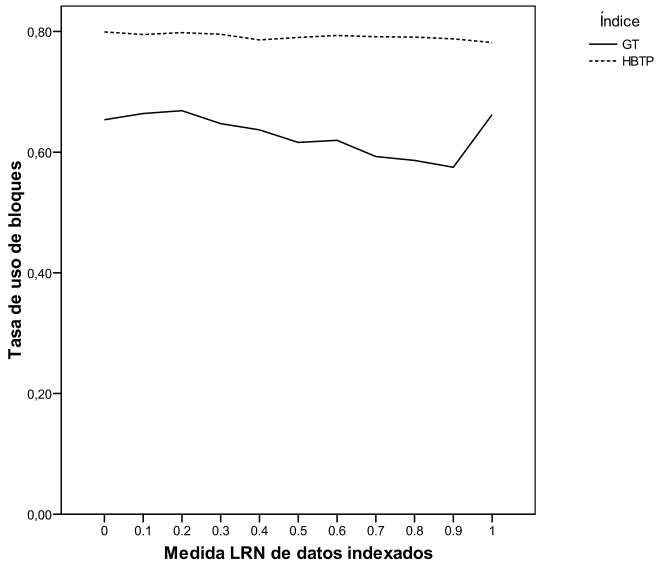

Figura 8.18: Tasa de uso de bloques de las técnicas para el indexado de datos imprecisos numéricos evaluadas bajo distintos niveles de LRN de los datos indexados para consultas que aplican una medida de necesidad

8.4.7.3. Rendimiento bajo diferentes circunstancias relativas a las consultas

El presente apartado está dedicado a exponer los resultados acerca de la influencia de características de las consultas sobre el rendimiento de las técnicas evaluadas. En primer lugar, se estudiará el impacto de la variación de la medida LRS de las consultas. Posteriormente, se analizará el impacto de la variación de la medida LRN de las consultas. Finalmente, se expondrán y analizarán los resultados relativos al rendimiento de las técnicas evaluadas bajo variaciones en el umbral de las consultas empleadas.

Influencia de la medida LRS de las consultas.

La relación entre la medida LRS de las consultas y el rendimiento de las técnicas de indexado evaluadas, para el caso en que se aplica una medida de posibilidad, se muestra en la Figura 8.19. En dicha figura, se pueden apreciar las diferencias de rendimiento entre GT y 2BPT para valores bajos y altos de la medida LRS de las consultas. En el primero de los casos, cuando el valor de la medida LRS de las consultas es bajo, GT presenta un mejor rendimiento que 2BPT. Ésta diferencia es especialmente notable para el caso extremo en que la medida LRS de las consultas es 0 (lo que implica ejecutar consultas crisp), llegando ésta a ser del 45 %. La diferencia de rendimiento entre GT y 2BPT se reduce sensiblemente conforme la medida LRS de las consultas va creciendo (27 % para un nivel de LRS de 0.1, 17 % para un nivel de LRS de 0.2, 10 % para un nivel de LRS de 0.3, etc.) ya que la tasa de crecimiento de la curva de 2BPT es menor que la de GT. Dicho decremento llega hasta tal punto que, para niveles elevados de la medida LRS de las consultas, el rendimiento de 2BPT es

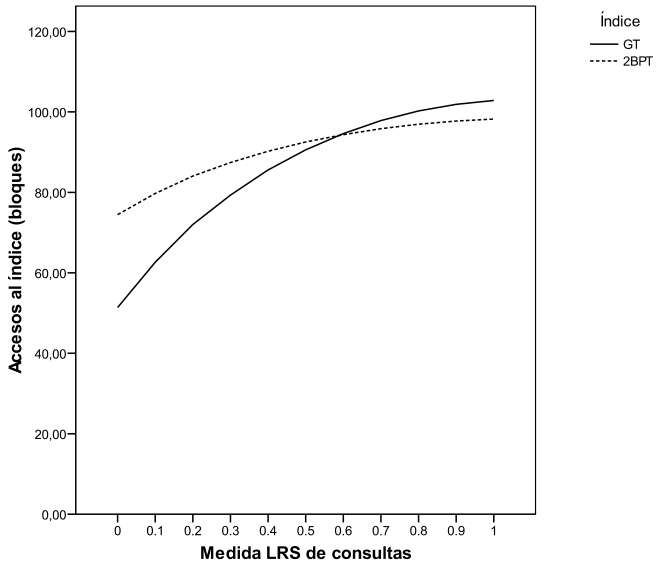

Figura 8.19: Rendimiento de las técnicas para el indexado de datos imprecisos numéricos evaluadas bajo distintos niveles de LRS de las consultas cuando éstas aplican una medida de posibilidad

ligeramente superior al de GT. De hecho, el rendimiento medio de 2BPT para todas las pruebas realizadas es un 5.6 % menor que el del método de referencia.

En lo que respecta al caso en que se aplica una medida de necesidad, la influencia de la medida LRS de las consultas en el rendimiento de las técnicas de indexado evaluadas se muestra en la Figura 8.20. En dicha figura, se puede apreciar que el rendimiento de HBPT es igual o superior, en función del crecimiento de la medida LRS de las consultas, que el registrado para GT. En el caso extremo en que el nivel de dicho factor es 1.0, el rendimiento de HBPT es superior en un 12 % al de GT. En media, para todas las pruebas realizadas, el rendimiento de HBPT es un 7 % superior al de GT.

Obsérvese que la tendencia ascendente en todas las curvas mostradas en las figuras 8.19 y 8.20 se debe al incremento de resultados en las consultas procesadas. Este incremento está causado por el aumento del tamaño del soporte o núcleo, según el caso, debido al incremento del la medida LRS de los conjuntos difusos que modelan las restricciones flexibles aplicadas.

Influencia de la medida LRN de las consultas.

En la Figura 8.21 se muestran, para el caso en que se aplica una medida de posibilidad, los valores de rendimiento registrados para las técnicas de indexado evaluadas ante variaciones en los valores de la medida LRN de las consultas aplicadas. En dicha figura, se puede observar que el rendimiento de GT es superior al de 2BPT. No obstante, el rendimiento de GT es superior al de GT en un 2.3 %, en el caso más favorable para 2BPT (en que la medida LRN es 0), y hasta un 11.3 %, en el caso extremo (en que la medida LRN de las consultas es 1). Para la totalidad de las pruebas, GT posee un rendimiento medio superior

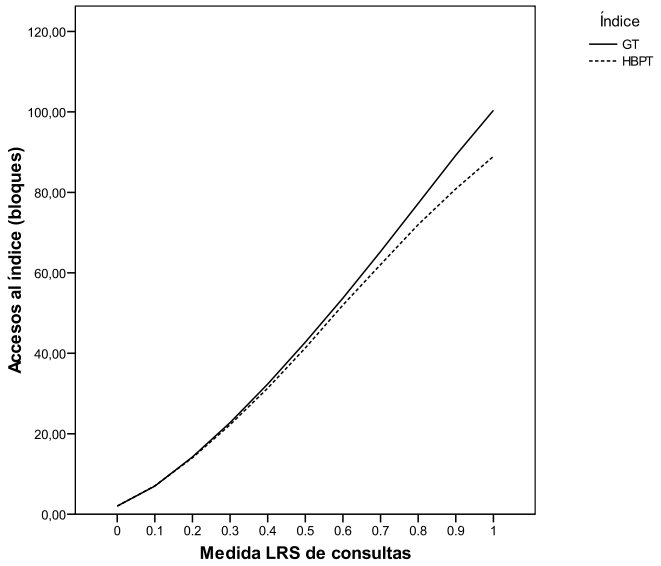

Figura 8.20: Rendimiento de las técnicas para el indexado de datos imprecisos numéricos evaluadas bajo distintos niveles de LRS de las consultas cuando éstas aplican una medida de necesidad

en un 5.6 % con respecto al medido para 2BPT.

La Figura 8.22 muestra la relación entre el valor de la medida LRN de las consultas y el rendimiento de las técnicas de indexado evaluadas para el caso en que se aplica una medida de necesidad. En la citada figura, se puede observar que el rendimiento de HBPT es superior al registrado para GT en todo el espectro de valores del parámetro analizado. La diferencia entre HBPT y el método de referencia es, en el mayor de los casos (cuando el valor de la medida LRN es 0), ligeramente superior al 6 %. Dicha diferencia se reduce hasta el 2.5 % para el caso extremo en que el valor para la medida LRN es 1. Para la totalidad de las pruebas realizadas, el rendimiento medio de HBPT es superior en un 4.1 % con respecto al medido para GT.

Téngase en cuenta que la tendencia descendente de las curvas presentadas en las figuras 8.21 y 8.22 se debe a la disminución del número de resultados de las consultas. Esta disminución es debida a la disminución de la longitud del intervalo que caracterizan la base de las condiciones flexibles aplicadas causada por la disminución de la longitud del intervalo que caracteriza el núcleo de las mismas.

Influencia del valor de umbral de las consultas.

Finalmente, la Figura 8.23 muestra los resultados relativos a la influencia que ejerce el valor de umbral de las consultas sobre el rendimiento de las técnicas de indexado evaluadas para el caso en que se aplica una medida de posibilidad. En dicha figura, se muestra, al igual que ha ocurrido para el caso de los niveles LRN de las consultas, que el rendimiento del método GT es superior al de 2BPT para todo el rango de valores de umbral. No obstante, y al igual que en el citado

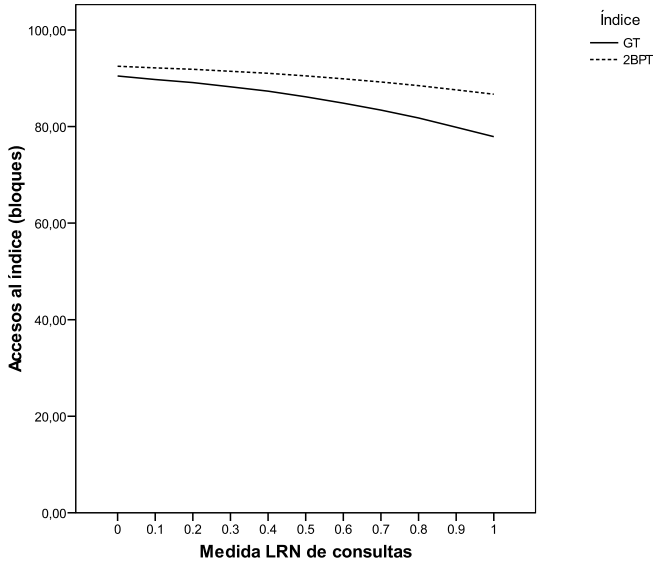

Figura 8.21: Rendimiento de las técnicas para el indexado de datos imprecisos numéricos evaluadas bajo distintos niveles de LRN de las consultas cuando éstas aplican una medida de posibilidad

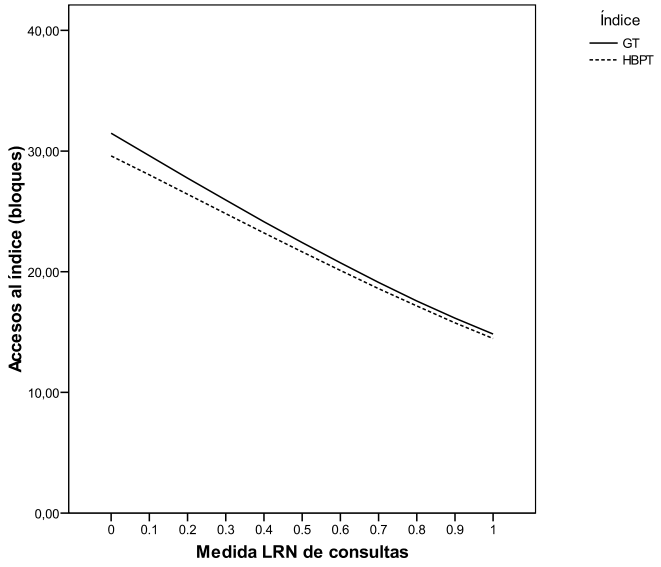

Figura 8.22: Rendimiento de las técnicas para el indexado de datos imprecisos numéricos evaluadas bajo distintos niveles de LRN de las consultas cuando éstas aplican una medida de necesidad

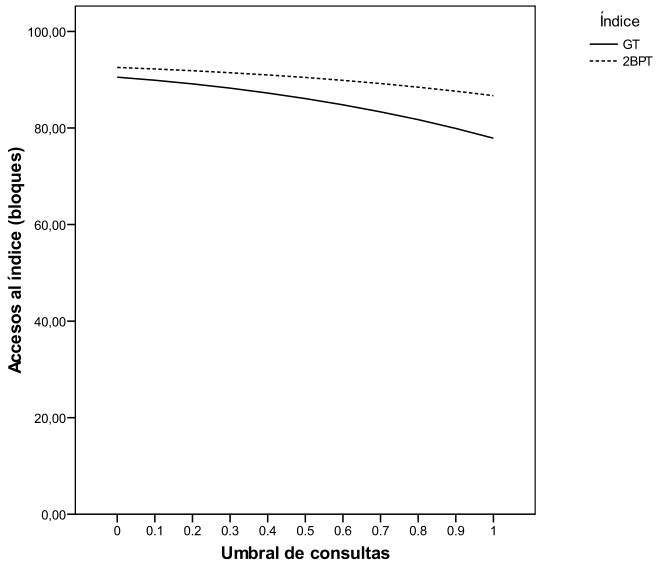

Figura 8.23: Rendimiento de las técnicas para el indexado de datos imprecisos numéricos evaluadas bajo distintos niveles de umbral de las consultas cuando éstas aplican una medida de posibilidad

caso, la diferencia de rendimiento entre los métodos evaluados no es excesiva ya que varía desde un 2.2 %, en el caso más favorable cuando el umbral es 0, hasta el 11.3 % en el caso en que el umbral es 1. El rendimiento medio rendimiento para la totalidad de las pruebas de 2BPT es un 5.6 % inferior al medido para el método re referencia.

En lo que respecta al caso en que se aplica una medida de necesidad, los resultados sobre la influencia del valor de umbral de las consultas sobre el rendimiento de las técnicas de indexado consideradas se muestran en la Figura 8.24. En dicha figura, se muestra un rendimiento de HBPT ligeramente mayor al del método de referencia. Esta diferencia es algo superior al 6.3 % en el caso más favorable, que coincide con que el valor del umbral se establece a 0, y ligeramente superior al 2.5 % en el caso menos favorable cuando el umbral es máximo. El rendimiento medio, para la totalidad de las pruebas, de HBPT es un 4.3 % superior al de GT.

Se ha de notar que los resultados expuestos anteriormente, tanto para el caso en que se aplica una medida de posibilidad como para el que se aplica una medida de necesidad, son prácticamente idénticos a los obtenidos en el estudio de la influencia del nivel de la medida LRS de las consultas (véase figuras 8.21, 8.22, 8.23 y 8.24). Esta coincidencia se debe a que ambos factores afectan de forma similar a la longitud del intervalo que caracteriza la base de las condiciones flexibles aplicadas y, por tanto, generan unas condiciones de consulta similares.

Finalmente, se ha de observar que la tendencia descendente de las curvas mostradas en las figuras 8.23 y 8.24 se debe, al igual que en el caso de los resultados presentados sobre la influencia del nivel de la medida LRS de las consultas, a la reducción del número de resultados provocada por el decreci-

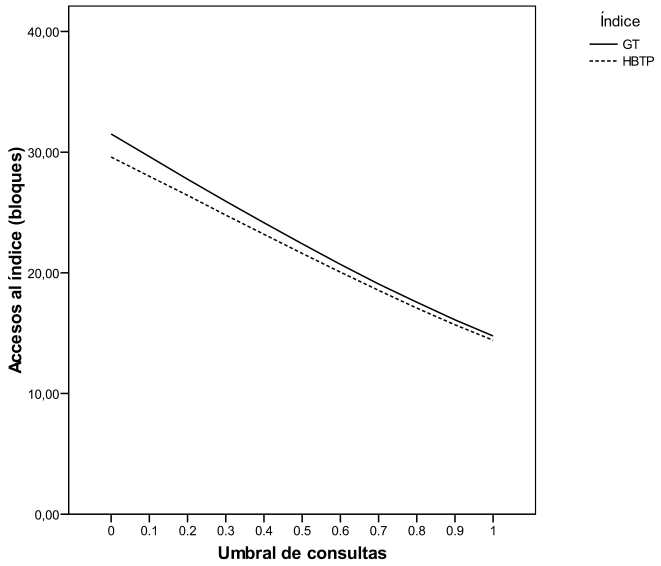

Figura 8.24: Rendimiento de las técnicas para el indexado de datos imprecisos numéricos evaluadas bajo distintos niveles de umbral de las consultas cuando éstas aplican una medida de necesidad

miento en la longitud del intervalo que caracteriza la base de las condiciones flexibles aplicadas.

8.4.8. Conclusiones

Los métodos de indexado de datos numéricos imprecisos, 2BPT y HBPT, han sido evaluados frente a la técnica de indexado de referencia GT.

Para el primero de los casos, la técnica 2BPT para la resolución de consultas flexibles que aplican una medida de posibilidad, se ha medido un rendimiento medio global un 5.6 % inferior al registrado para GT. En lo que respecta al rendimiento bajo distintas condiciones particulares de datos indexados y de consulta realizadas, 2BPT ha mostrado un rendimiento también inferior al de GT. Dicha diferencia de rendimiento es mayor en los casos extremos en los que los datos indexados o las condiciones son *crisp*, no obstante ésta se ve reducida sensiblemente en cuanto los datos o consultas indexados son imprecisos. En media, la diferencia de rendimiento entre 2BPT y el método de referencia es de un 5.6 % en el peor de los casos. Lo resultados han indicado un excelente comportamiento de la técnica 2BPT ante cambios en las condiciones de datos. El rendimiento de 2BPT se mantiene estable ante los citados cambios en contraste con la alta inestabilidad de GT.

El segundo de los casos, en que se ha evaluado la técnica HBPT para la resolución de consultas flexibles que aplican una medida de necesidad, la técnica HBPT ha mostrado en condiciones generales un rendimiento superior al de la técnica de referencia en un 4.2 %. Bajo condiciones particulares de datos indexados y consultas, dicha superioridad se mantiene, siendo en media de un 2 % en el peor de los casos hasta un 7.7 % en el mejor de ellos. Además de lo

anterior, HBPT ha mostrado, al igual que 2BPT, una excelente estabilidad ante cambios en las condiciones particulares de datos, a diferencia de GT que ha mostrado alta inestabilidad en estos casos.

En nuestra opinión, el rendimiento mostrado por los métodos propuestos con respecto al método de referencia, ligeramente inferior para el caso en que se procesan consultas flexibles que aplican una medida de posibilidad y ligeramente superior para el caso en que se aplica una medida de necesidad, junto con la ventaja añadida de que éstos métodos emplean estructuras de indexado subyacentes ya implementadas y altamente optimizas en la mayoría de los SGBD disponibles, hacen de éstos unos buenos candidatos para su uso en SGBDDs, especialmente los desarrollados como extensión de un SGBD clásico.

8.5. Indexado de datos imprecisos escalares

La presente sección propone una adaptación de la técnica de indexado de naturaleza escalar más extendida e implementada en los SGBD actuales, los mapas de bits, para permitir su uso en el indexado de datos imprecisos difusos.

El motivo del empleo de los mapas de bits para esta tarea no es otro que facilitar la incorporación de mecanismos de indexado en SGBDDs desarrollados como extensión de un SGBD clásico en el que esta estructura de indexado ya está presente de forma nativa y su implementación altamente optimizada.

Esta propuesta, una vez descrita, será evaluada con respecto a una técnica de indexado de referencia. De entre las diferentes técnicas que han sido empleadas para indexar datos imprecisos escalares, comentadas en el capítulo dedicado al estado del arte del presente trabajo, las listas invertidas han mostrado, con clara diferencia, un rendimiento mayor al resto [78]. Es por ello, que en el presente trabajo emplearemos éstas como técnica de referencia.

8.5.1. Estructuras de indexado

La presente subsección describe las estructuras de indexado consideradas en el presente trabajo. En primer lugar, se describirá la estructura de indexado empleada como referencia en la evaluación de la propuesta. Apoyándose en esta descripción, ya que ambas técnicas se basan en una estructura similar, se describirá posteriormente la estructura de indexado propuesta.

8.5.1.1. Listas invertidas para indexar datos imprecisos escalares

Una lista invertida (o *inverted file* en su acepción inglesa) es una estructura de indexado diseñada para indexar valores conjuntivos definidos sobre un dominio discreto subyacente. Las listas invertidas están compuestas por un directorio y a una serie de listas de referencias a los registros indexados. El directorio contiene una entrada por cada elemento $d_1, d_2, \ldots, d_m$ del dominio D_a subyacente al asociado al atributo indexado a. Cada una de estas entradas apunta a una lista que contiene referencias a los registros indexados cuyo valor contiene al asociado a la entrada correspondiente. La Figura 8.25 ilustra la estructura de los ficheros invertidos descrita anteriormente. En dicha figura, d_i representa al elemento i-ésimo de dominio asociado al atributo indexado, r_i^j representa a la referencia j-ésima de la lista de referencias asociada al elemento d_i y p^i al número de referencias de dicha lista. Dado que el presente trabajo es la adaptación de

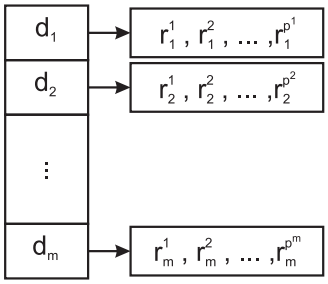

Figura 8.25: Fichero invertido

estas estructuras para indexar datos imprecisos, referimos al lector a [176, 133] para más detalles sobre las listas invertidas *clásicas*.

Compresión de listas de referencias.

En nuestro caso, para incrementar la eficiencia de las listas invertidas, comprimimos las listas de referencias. En lugar codificar directamente en las listas la posición absoluta en la base de datos de los registros referenciados (valor de tipo entero), codificamos la distancia relativa entre los registros a que se refiere una referencia y la siguiente. Para poder implementar este sistema de compresión, las referencias en las listas han de mantenerse ordenadas de forma ascendente ($r_i^j \leq r_i^{j+1}$) y la primera referencia ha de codificarse como la posición absoluta del registro al que apunta. Las distancias entre referencias son números enteros menores, o en el peor caso de igual tamaño, que los correspondientes a las posiciones absolutas de los registros. Aprovechando esta particularidad, éstas puede ser codificadas empleando la codificación Variable-Byte (VB) [98]. Ésta es una codificación de ancho variable que emplea un número variable de bytes para representar enteros. Para ello, VB emplea un bit de cada byte, denominado *bit de continuación*, que indica si el código termina (bit de continuación establecido a 1) o no (bit de continuación establecido a 0) con el presente byte. El resto de bits se emplean para representar el entero codificado.

El siguiente ejemplo ilustra el funcionamiento de VB.

Ejemplo 8.6. *Supongamos que deseamos codificar empleando VB las cantidades enteras* 5 *y* 824.

En un byte disponemos de 7 *bits en el para representar la cantidad (una vez excluido el bit de continuación), empleando un sólo byte podremos representar números enteros inferiores a* 127. *Dado que* 5 *es menor que dicho límite, éste puede ser representado empleando un solo byte. Por tanto, el entero* 5 *puede ser codificado como* **1**0000101, *donde el bit de continuación (indicado en negrita) indica que no serán necesarios más bytes para la codificación, y el resto de bits codifica en binario dicha cantidad.*

En el segundo caso, el entero 824 *supera el límite para la codificación en un solo byte, por lo que serán necesarios más. Concretamente, la representación binaria de dicho entero requiere 10 bits, por lo que podremos representar la cantidad empleando dos bytes (con* 7 *bits hábiles cada uno). Por tanto, el entero* 824 *se puede codificar como* **0**0000110 **1**0111000. *Obsérvese que el primero de los bytes tiene el bit de continuación desactivado, lo que indica que el siguiente byte*

se empleará en la codificación, y que el segundo lo tiene activado para indicar que es el último byte empleado.

Las listas invertidas pueden ser empleadas para indexar datos imprecisos escalares aunque según la medida que apliquen las consultas flexibles a resolver será necesario realizar ciertas modificaciones sobre su estructura básica.

Uso de listas invertidas para resolver consultas que aplican una medida de posibilidad.

En caso de que las consultas a procesar apliquen una medida de posibilidad, el proceso de indexado de datos imprecisos se hará empleando la estructura tradicional de las listas invertidas. Haciendo uso de ésta, se indexarán los conjuntos que representan los soportes de las distribuciones de posibilidad que representan los datos indexados.

La resolución de las consultas se realizará aplicando los principios de indexado expuestos en las ecuaciones 8.7, para el caso general en que el umbral es mayor que cero, y 8.4, para el caso particular en que el umbral es cero. La resolución de las consultas flexibles empleando estos principios es equivalente a la resolución, en términos de conjuntos crisp, de una consulta que busca obtener aquellos registros cuyo valor conjuntivo asociado (el soporte de los elementos indexados) contiene alguno de los elementos presentes en el conjunto empleado como condición (el conjunto que representa la base de la condición flexible). Por simplicidad, a este tipo de consultas se les denominará como *consultas de intersección no vacía.*

Una consulta de intersección no vacía se resolverá simplemente accediendo (mediante el empleo del directorio) a todas las listas de referencias asociadas a los elementos contenidos en el conjunto condición, devolviéndose como resultado el conjunto de dichas referencias.

Uso de listas invertidas para resolver consultas que aplican una medida de necesidad.

Cuando se han de resolver consultas que aplican una medida de necesidad, la estructura del índice (concretamente la de las listas de referencias) tendrá que ser modificada convenientemente.

La resolución de consultas de este tipo se realizará mediante la aplicación de los principios de indexado indicados en la ecuaciones 8.8, cuando se trata de consultas que aplican un umbral superior a cero, y 8.6 en el caso en que el umbral sea cero. Empleando estos principios, la resolución de consultas flexibles se transforma en la resolución de consultas en las que se busca obtener aquellos registros cuyo valor conjuntivo asociado sea un subconjunto del conjunto empleado como condición. A este tipo de consultas se les denominará, por simplicidad, como *consultas subconjuntivas.*

Para resolver consultas flexibles empleando el método descrito anteriormente, los datos imprecisos escalares serán indexados mediante el soporte de las distribuciones de posibilidad que los modelan. Éstos conjuntos se indexan empleando una lista invertida en la que, por cada elemento del conjunto, se insertará en la lista de referencias correspondiente la referencia a su registro asociado. Además de esto, se almacenará por cada referencia la cardinalidad del valor conjuntivo asociado al registro al que apunta. Esta información adicional, es

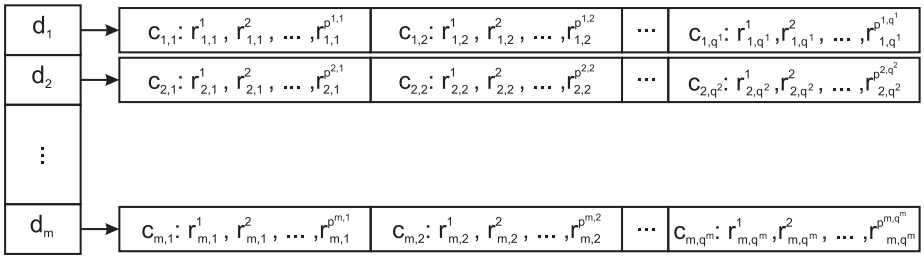

Figura 8.26: Fichero invertido para resolución de consultas subconjuntivas

necesaria para la resolución de consultas subconjuntivas ya que en la listas invertidas, dada una referencia de la lista de referencias, no existe una forma directa de determinar todos los elementos que forman parte de un valor conjuntivo indexado sin acceder directamente al mismo o realizar una búsqueda exhaustiva en el índice. Conociendo la cardinalidad de cada elemento indexado, podemos determinar si existen en su valor conjuntivo asociado elementos no contenidos en el conjunto que se emplea como condición.

Dada la necesidad de almacenar las cardinalidades de los valores indexados, la estructura de indexado deberá modificarse convenientemente para permitir la inclusión de este nuevo dato. Concretamente, modificaremos la estructura de listas de referencias de tal forma que las referencias estén agrupadas según la cardinalidad de los valores conjuntivos asociados a los registros a los que apuntan. Las listas de referencias, por tanto, estarán organizadas como una secuencia de grupos de referencias. La Figura 8.26 ilustra la organización de las listas de referencias que se ha descrito anteriormente. En dicha figura, d_i representa al elemento i-ésimo de dominio asociado al atributo indexado, $c_{i,j}$ la j-ésima cardinalidad de la lista de referencias asociada a d_i, $r^k_{i,j}$ la k-ésima referencia en la agrupación de referencias con cardinalidad $c_{i,j}$, q^i el número de agrupaciones de cardinalidad idéntica presentes en la lista de referencias asociada al elemento d_i y $p^{i,j}$ el número de referencias dentro de la agrupación con cardinalidad $c_{i,j}$ de la lista de referencias asociada al elemento d_i.

Las referencias dentro de cada grupo estarán ordenadas de forma ascendente, en función de la posición en la base de datos del registro al que apuntan ($r^k_{i,j} \leq r^{k+1}_{i,j}$). De esta forma, la secuencia de referencias de una determinada agrupación se podrá comprimir mediante el método descrito anteriormente para el caso general. Así mismo, los grupos dentro de cada lista de referencias estarán ordenados de forma ascendente en función de la cardinalidad ($c_{i,j} < c_{i,j+1}$). Este tipo de ordenación se denomina *impact ordering* [98] en el área de Recuperación de Información (u, originalmente, *Information Retrieval*) e incrementará la eficiencia de la resolución de consultas subconjuntivas.

Empleando la estructura de indexado descrita anteriormente, las consultas subconjuntivas se resuelven en varias etapas. Se recorrerán todas las listas de referencias asociadas a los elementos pertenecientes al conjunto empleado como condición. El recorrido de cada una de ellas se realizará hasta encontrar una agrupación de referencias con una cardinalidad asociada superior a la del conjunto condición, ya que podemos estar seguros de que los valores conjuntivos asociados a los registros apuntados por las referencias de esta agrupación, y las de posteriores agrupaciones, no pueden ser subconjuntos del conjunto empleado

como condición porque poseen más elementos que éste. Dado que la cardinalidad de cada agrupación es el primer dato que encontramos de éstas, no será necesario leer las referencias de la agrupación cuya cardinalidad ha satisfecho el criterio de parada. Durante el recorrido de las listas, se anotarán todos los registros apuntados por la referencias leídas, el número de referencias que apuntan a cada uno de los registros, así como la cardinalidad indicada para cada uno de ellos (mediante la cardinalidad indicada para las agrupaciones a las que pertenecen sus referencias). Finalmente, comprobaremos por cada uno de los registros encontrados si el número de referencias leídas que le apuntan coincide con la cardinalidad del valor asociado a éstos y devolveremos como resultado de la consulta aquellos registros que cumplan esta condición. Podremos descartar los registros que no cumplen la condición, ya que se puede asegurar que el valor conjuntivo asociado a éstos debe poseer un elemento no contenido en el conjunto empleado como condición.

Además del procedimiento de resolución de consultas subconjuntivas descrito anteriormente, existe otro procedimiento que no requiere alterar la estructura original de la lista invertida. Éste se basa en obtener el conjunto de registros que satisface una consulta de intersección no vacía empleando como condición el conjunto complementario al empleado como condición de la consulta original. Por motivos de brevedad, denominaremos a la anterior consulta *consulta complementaria* y a la consulta subconjuntiva, de la que la consulta complementaria deriva, *consulta original*. El conjunto de registros que satisfacen la consulta original se derivará en base al conjunto de resultados de la consulta complementaria. Podemos segura que cada registro que satisface la consulta complementaria posee en su valor conjuntivo, empleado como clave en el índice, algún elemento no incluido en el conjunto condición de la consulta original. Por tanto, es imposible que los valores clave de estos registros sean subconjuntos del la condición de la consulta original. Según lo anterior, podemos concluir que el conjunto de resultados de la consulta original estará formado por todos aquellos registros en la base de datos que no formen parte del resultado de la consulta complementaria. En conclusión, el conjunto de resultados de la consulta original se determinará como el complemento del conjunto de resultados de la consulta complementaria.

8.5.1.2. Mapas de bits comprimidos para indexar datos imprecisos escalares

Los mapas de bits (o *bitmaps* en inglés) son estructuras de indexado ampliamente empleadas para construir índices en los que los valores de clave empleada proceden de un dominio escalar de baja cardinalidad. Un mapa de bits está compuesto por una serie de secuencias de bits, una por cada elemento d_i del dominio D_a (donde D_a es el dominio asociado al atributo a , empleado como atributo clave). En cada una de las secuencias de bits, se dispondrá de un bit por cada registro indexado. Cada bit de la secuencia asociada a un valor d_i indicará si el registro al que éste representa tiene (valor 1) o no (valor 0) a d_i como valor asociado para el atributo a. Los valores del dominio D_a estarán asociados con su secuencia de bits correspondiente mediante un directorio. La Figura 8.27 ilustra la estructura descrita anteriormente.

Gracias a las características de estructura de los mapas de bits, estos pueden ser también empleados para indexar valores conjuntivos definidos sobre un

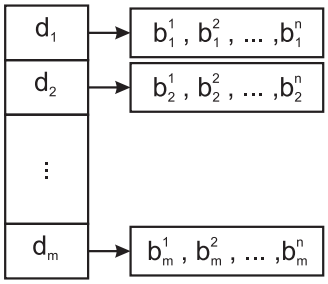

Figura 8.27: Mapa de bits

dominio subyacente de las características anteriormente mencionada. En dicho caso, el bit que representa un determinado registro será de valor 1 para todas las secuencias de bits asociadas a los elementos pertenecientes al valor conjuntivo asociado para el dominio indexado a dicho registro.

Compresión de secuencias de bits.

Es usual, para incrementar la eficiencia de estos índices en grandes bases de datos, que las secuencias de bits se compriman. Existen multitud de métodos para la compresión de cadenas de bits [154], dividiéndose éstos entre los métodos con alineación y los no alineados. La diferencia entre unos y otros estriba en los bloques de datos con los que trabajan. Los bloques de los métodos alineados son de tamaño fijo (normalmente coincidente con la cantidad de bits en un byte o una palabra). En cambio, los bloques de los métodos no alineados son de tamaño variable. Usualmente los métodos no alineados consiguen mejores tasas de compresión pero los algoritmos para el tratamiento de datos no son demasiado rápidos, en cambio los métodos alineados no consiguen las mejores tasas de compresión pero permiten un procesamiento muy rápido de las cadenas comprimidas aprovechando las capacidades de procesamiento en bloque de los actuales procesadores. De entre todos éstos, en este trabajo se empleará BBC [1, 2], un método alineado con bloques del tamaño de un byte. Dado su tamaño de bloque, BBC consigue tasas razonables de compresión y una velocidad de procesamiento muy alta. Se ha de remarcar que este método de compresión es el empleado por el SGBD Oracle [3].

El método de compresión BBC emplea una codificación *run-length* (RLE, del inglés *run-length encoding*) para comprimir secuencias de bits de un mismo valor. Dado el carácter alineado de este método de compresión, estas secuencias deberán tener un tamaño múltiplo del número de bits en un byte. Según la terminología empleada en la descripción original de BBC, las bloques de bits que son comprimidos se denominan *fills* y mientras que los bloques que no pueden ser comprimidos por el método se llaman *tails*. Teniendo en cuenta que el tamaño de los bloques es un byte, se emplearán indistintamente *bloque* y *byte* para referirse a éstos.

Existen dos modos de funcionamiento de BBC. El primero modo sólo comprime secuencias de bits con valor 0, tratando cualquier secuencia que contenga un bit con valor 1 como un bloque *tail*. Debido a esta particularidad, este modo está indicado para secuencias de vectores dispersos. El segundo modo de funcionamiento de BBC comprime tanto secuencias de bits con valor 0 como con

valor 1. Evidentemente, la tasa de compresión de este modo para vectores dispersos es mayor ya que al comprimir cada *fill* hay que dedicar un bit adicional (denominado *fill bit*) para indicar el valor que comparte la secuencia de bits comprimida. Dadas las características, en lo que se refiere a densidad de bits, de las secuencias en los mapas de bits cuando se indexan datos de naturaleza imprecisa, emplearemos en el presente trabajo el segundo modo de BBC para realizar la compresión de los índices.

A la hora de codificar los datos, BBC emplea como unidad de codificación un *fill* seguido de un *tail*. Esta pareja se denomina, según la terminología de BBC, como un *run*. Cada *run* se codificará empleando un sólo byte (que denominaremos *byte cabecera*) seguido, según el caso, de los bytes que componen el *tail* del mismo. La variante que se emplea en el presente trabajo considera cuatro tipos de *run*:

- Tipo 1: El *run* está compuesto por un *fill* de 0 a 3 bytes de longitud y por un *tail* con una longitud de 0 a 15 bytes. Este tipo de *run* codifica en su byte cabecera tanto la longitud del *fill* como el *fill bit*. Así mismo, en la cabecera se codifica la longitud del *tail*, cuyos bloques se representan explícitamente tras ésta.

- Tipo 2: Éste se compone de un *fill* con una longitud de 0 a 3 bytes seguido por un *tail* de 1 byte. En este tipo de *run*, el *tail* ha de ser de una forma particular. Todos los bits en el *tail* han de tener el mismo valor que el *fill bit* del *tail* a excepción de uno de ellos. A esta excepción la denominaremos *odd bit*. Este tipo de *run* se representa exclusivamente empleado el byte cabecera. En éste, se codifica la longitud del *fill*, su *fill bit* y la posición del *odd bit* en el *tail*.

- Tipo 3: Este *run* está formado por un *fill* con una longitud mayor de 3 bytes y un *tail* con una longitud de 0 a 15 bytes. El byte de cabecera codifica el *fill bit* así como la longitud del *tail*. La longitud del *fill* es codificada tras el byte de cabecera empleando la codificación VB, descrita anteriormente como mecanismo de compresión de listas de referencias. Tras los bytes necesarios para la codificación de la longitud del *fill* se representarán explícitamente los bytes que componen el *tail*.

- Tipo 4: Este tipo de *run* se compone de un *fill* de longitud mayor de 3 bytes y de un *tail* formado por un solo byte de las características descritas para el caso del tipo 2. El byte cabecera de este tipo de *run* codifica el *fill bit* y la posición del *odd bit* del *tail*. Tras él, se representa empleando codificación VB la longitud del *fill*.

La estructura de los bytes de cabecera correspondientes a cada uno de los tipos anteriores se corresponde con lo mostrado en la Figura 8.28. En dicha figura, los bits fijos son representados con su valor correspondiente y los bits de valor variable se representa con una equis. Las longitudes y posiciones indicadas en la figura se codifican empleando codificación binario ordinario.

El siguiente ejemplo ilustra la forma en que una secuencia de bits es comprimida empleando BBC.

Ejemplo 8.7. *Supongamos que deseamos comprimir la secuencia de bits mostrada a lo largo de la primera columna de la Tabla 8.5. Nótese que, en dicha*

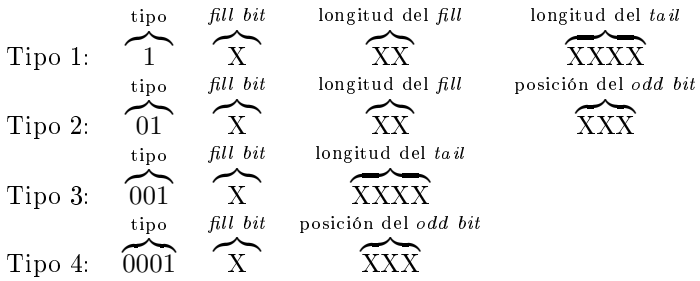

Figura 8.28: Codificación de los diferentes tipos de *runs* de BBC

Secuencia original	Tipo de *run*	Secuencia comprimida
`00 00 00 4F 3C`	1	`B2 4F 3C`
`FF FF 40`	2	`72`
`00 00 00 00 00 00 00 00`	3	`20 08`
`FF FF FF FF FF FF 02`	4	`1D 06`

Tabla 8.5: Ejemplo de compresión de secuencias de bits empleando la BBC

tabla, por motivos de brevedad la secuencia de bits son representas de forma compacta empleando codificación hexadecimal. Esta secuencia de bits está divida en diversas filas, de tal forma que asocia a un run *de la secuencia los fragmentos en cada fila. El tipo al que corresponde cada uno de estos* run *se ha indicado en la segunda columna de la tabla.*

Una vez comprimida empleando BBC, la secuencia de bits mencionada anteriormente se representa por una secuencia de bytes que se corresponde con la mostrada en la tercera columna de la Tabla 8.5. Al igual que la secuencia de bits, la secuencia de bytes se representa empleado codificación hexadecimal. La división de la secuencia de bytes en las diferentes filas de la tabla permite apreciar la codificación de cada uno de los diferentes run *de que se compone la secuencia de bits comprimida.*

A pesar de que la secuencia de bits incluida en este ejemplo es bastante pequeña, el método BBC muestra sus capacidades de compresión reduciendo ésta a algo menos del 35 % de su tamaño original. Para secuencias de bits más largas, el ratio de compresión de BBC puede mejorar significativamente empleando los tipos de run *3 y 4.*

Uso de mapas de bits comprimidos para resolver consultas que aplican una medida de posibilidad.

Los mapas de bits pueden ser empleados para indexar de datos imprecisos si aplicamos los principios de indexado para resolver consultas que aplican una medida de posibilidad indicados en la ecuaciones 8.7 y 8.4, según el umbral de éstas. Empleando este marco, el mapa de bits indexará los conjuntos *crisp* que se corresponden con el soporte de las distribuciones de posibilidad que modelan dichos datos imprecisos. La resolución de consultas que aplican medidas de posibilidad será equivalente a la resolución de consultas *crisp* de intersección no vacía.

Las consultas de intersección no vacía se resuelven aplicando la operación

binaria *OR* sobre las secuencias de bits asociadas a cada uno de los elementos del conjunto empleado como condición. El resultado de esta operación, la agregación disyuntiva de las secuencias mencionadas anteriormente, será una secuencia de bits que indicará, mediante el valor 1, los registros que satisfacen la consulta. El acceso a cada una de las secuencias agregadas se hace de forma directa empleando el directorio del índice. En este punto, se ha de destacar que el método de compresión BBC posee la particularidad de permitir la agregación de secuencias binarias comprimidas sin la necesidad de descomprimir previamente las mismas, lo que redundará en la eficiencia del método de indexado para la resolución de este tipo de consultas. Una vez agregadas las secuencias de bits, las posiciones con valor 1 de la secuencia resultante marcarán aquellos registros que conforman el resultado de la consulta.

Uso de mapas de bits comprimidos para resolver consultas que aplican una medida de necesidad.

En el caso que se deseen resolver consultas que aplican una medida de necesidad empleando un mapa de bits, aplicando los principios de indexado indicados en las ecuaciones 8.8 y 8.6 (según si el umbral es superior a cero o igual a este valor), se indexarán los conjuntos *crisp* que se corresponden con el núcleo de las distribuciones de posibilidad que modelan los datos imprecisos sobre los que se aplica tal consulta. Este mapa de bits tendrá en su directorio una entrada adicional que apuntará a la secuencia de cardinalidades de los valores conjuntivos empleados como clave para los registros indexados según el mismo orden en que éstos se representan en las secuencias de bits.

Empleando el esquema de indexado descrito anteriormente, la resolución de consultas flexibles que aplican una medida de posibilidad será análoga a la resolución, mediante el mapa de bits descrito anteriormente, de una consulta *crisp* subconjuntiva. Las consultas subconjuntivas se resolverán en este tipo de índice de una forma similar a la descrita para el caso en que se emplea una lista invertida. Por cada elemento del conjunto que se emplea como condición, se examinará su secuencia de bits para anotar los registros cuyo valor clave incluye a dicho elemento. Además, por cada registro, se mantendrá un contador del número de veces que éste ha aparecido a lo largo del análisis de cada una de las secuencias de bits consideradas. Terminado el proceso, se examinarán las cardinalidades de los registros anotados indicadas en la entrada especial del índice y se eliminarán aquellos cuya cardinalidad no coincida con el número de apariciones durante el análisis de la secuencias de bits. Filtrados los registros que no cumplan la condición anterior, se devolverá el resto como resultado de la consulta.

Al igual que en el caso de las listas invertidas, existe un método alternativo para la resolución de consultas subconjuntivas. Éste, de la misma forma que el caso mencionado, se basa en la resolución de una consulta de intersección no vacía empleando como condición el conjunto complementario del que hace las veces de condición en la consulta subconjuntiva. Una vez resulta la consulta de intersección no vacía, se devolverán como resultado de la consulta subconjuntiva aquellos registros de la base de datos indexada que no figuren en el resultado de la anterior. Nótese que, en caso de aplicar este método alternativo, no será necesario añadir al mapa de bits la secuencia de cardinalidades de los registros indexados descrita anteriormente. En adelante denominaremos al método

alternativo descrito anteriormente como *método complementario*, por el uso del complemento del valor conjuntivo de la consulta. En contraposición al anterior, denominaremos el método de resolución de consultas descrito en primer lugar como *método directo*.

Dependiendo de la consulta, resultará más eficiente la aplicación del método directo o el complementario. En el caso de los mapas de bits, esta decisión puede tomarse automáticamente añadiendo a la estructura de datos de índice cierta información adicional. Concretamente, si se dispone del número de bloques en que se almacenan cada una de las secuencias de bits comprimidas del índice, se podrá determinar, dada una consulta, el número de bloques que será necesario recuperar para procesar ésta de forma directa o complementaria. Determinadas estas cantidades, se seleccionará el método que requiera menos accesos.

Nótese que esta decisión no puede ser tomada de forma automática para el caso de las listas invertidas. En el caso de los mapas de bits, se puede determinar previamente el número de bloques que se accederán simplemente conociendo las secuencias de bits que se han de recorrer, dado que éstas se recorren íntegramente, y el tamaño de éstas. En cambio, dado que las secuencias de referencias en las listas invertidas no se han de recorrer íntegramente, gracias a su mecanismo de parada en base a la cardinalidad, es imposible conocer, sólo con la información del número de bloques en que se almacena cada lista de referencias, el número de bloques que se han de acceder en cada caso.

8.5.2. Evaluación del rendimiento de la propuesta

La evaluación del rendimiento de la propuesta de uso de un bitmap comprimido para indexar datos imprecisos escalares se realizará de forma similar a la evaluación de rendimiento de las propuestas para datos imprecisos numéricos descrita en la sección 8.4 en su apartado 8.4.5. Por tanto, suscribiendo de nuevo los motivos expuestos en dicho apartado, la medida para evaluar el rendimiento de la propuesta con respecto al método de referencia será el número de bloques accedidos por el índice.

Dado que, en este caso, las dos técnicas evaluadas necesitan de un directorio para el acceso a la información relativa a los registros y que los directorios de ambas técnicas son similares, no se tendrá en cuenta en las medidas de rendimiento el acceso a esta parte del índice.

Con respecto a los factores que afectarán al rendimiento, también se han seguido la consideraciones que el apartado 8.4.5. No obstante, tras observar en la experimentación para la evaluación de las técnicas de indexado de datos imprecisos de numéricos la correlación de la influencia sobre el rendimiento de algunos de los factores los datos de indexados para el caso de la consultas de necesidad (medidas LRS y LRN) y de algunos factores de los factores de consultas (medida LRS, medida LRN y umbral), se simplificará el conjunto de factores observado. Como se indicó en su momento, dicha correlación se origina ya que los factores mencionados determinan de forma indirecta la longitud del núcleo de los datos indexados, para el caso de factores relativos a éstos y consultas que aplican medidas de necesidad, y la longitud de la base de condición, para el caso de los factores relativos a las consultas. Por ello, optaremos en este caso por el estudio de la cardinalidad (en lugar de la longitud de los intervalos dado que en el presente caso las técnicas indexan datos imprecisos escalares) del núcleo de los datos indexados y de la base de la condición, según el caso, en lugar de los

factores que determinan indirectamente a éstos. Para emplear un marco de medida homogéneo independiente de la cardinalidad del dominio subyacente sobre el que se definen los datos imprecisos, la cardinalidad de los soportes, núcleos y bases de las condiciones se medirá de forma relativa con respecto a la del dominio subyacente.

El empleo de un dominio escalar subyacente introduce nuevos factores que no son aplicables en el marco anterior, en el que se usó un dominio subyacente numérico. Además de los factores indicados previamente, se ha de considerar la cardinalidad del dominio subyacente y la frecuencia con que los distintos elementos de éste forman parte de las distribuciones de posibilidad que modelan los datos indexados o los conjuntos difusos empleados como restricciones de las condiciones flexibles. El primero de ellos afectará, junto con el tamaño de la base de datos indexada, al tamaño de los índices y por tanto al rendimiento de éstos. En lo que respecta al segundo, éste afectará a la longitud de las listas de referencias y a la frecuencia con que las listas largas han de ser recorridas, en el caso de las listas invertidas, y a la distribución de los valores de bits en las secuencias de bits, en el caso de los mapas de bits. Estos factores, a su vez, determinarán la efectividad de los diferentes mecanismos para la codificación de referencias de los métodos evaluados y por tanto su estudio dará una información importante sobre el rendimiento de éstos bajo diversas circunstancias.

En resumen, se evaluará el impacto de seis factores sobre el rendimiento de las técnicas de indexado evaluadas. Estos son: el tamaño de la base de datos, la cardinalidad del dominio, la cardinalidad relativa del soporte o núcleo (según la medida que apliquen las consultas) de las distribuciones de posibilidad que modelan los datos indexados, la cardinalidad relativa de la base de las consultas y el sesgo en la distribución de los elementos del dominio subyacente.

8.5.3. Experimentos

La metodología seguida para la realización del conjunto de experimentos destinado a la medición del rendimiento, tanto global como en circunstancias particulares de datos y consultas, de los métodos evaluados es la misma que la descrita en la subsección 8.4.6 para los métodos de indexado de datos imprecisos numéricos.

La presente subsección describirá dichos experimentos y las condiciones en que se han realizado éstos, centrándose en la diferencias que pueda haber con respecto a los descritos en la subsección mencionada anteriormente.

8.5.3.1. Aislamiento de factores físicos y lógicos

De esta forma, los experimentos han sido aislados de factores físicos y lógicos siguiendo la misma metodología. Se ha empleado el mismo tamaño para los datos referentes a una misma propiedad presentes las estructuras de las listas invertidas y bitmaps. En el caso de los datos relativos a la cardinalidad se ha empleado un byte, ya que no es necesario un mayor tamaño, y tanto las secuencias de referencias comprimidas mediante VB, como las secuencias de bits comprimidas por BBC, son representadas mediante el número entero de bytes que sea necesario. El tamaño de bloques de datos ha sido fijado en 4 KB para proporcionar una mayor resolución en la medida del rendimiento dado el tamaño compacto de los índices.

8.5.3.2. Rendimiento global

El rendimiento global ha sido medido aplicando las diferentes técnicas evaluadas sobre un conjunto de 30 bases de datos generadas aleatoriamente sobre las que se han aplicado un conjunto de 1000 consultas. Con objeto de simplificar el proceso de generación aleatoria de consultas, cada una de éstas (concretamente el conjunto que forma la base de la condición) ha sido determinada mediante la extracción aleatoria de un elemento de la base de datos sobre la que va a ser aplicada.

La cardinalidad relativa del soporte o núcleo, según el tipo de consulta, de los elementos de esta base de datos (y, por tanto, de la base de las consultas aplicadas) varía aleatoriamente de forma uniforme en el rango $(0, 1]$. La elección de este rango, concretamente la exclusión de su límite inferior, tiene como objeto evitar la existencia conjuntos vacíos que representen a registros que no satisfarán ninguna consulta y, por tanto, que no serán incluidos en ningún caso en los índices.

En lo que respecta al sesgo en la distribución de los elementos del dominio subyacente en los conjuntos anteriores, éste se ha determinado mediante el valor del parámetro z de la distribución discreta Zipf [120]. En esta distribución, para $z = 0$ obtendremos una distribución normal y, conforme dicho valor es mayor que cero, obtendremos una distribución más sesgada. La elección aleatoria de los elementos que formarán parte de los conjuntos *crisp* que representan el soporte o núcleo de las distribuciones de posibilidad, que a su vez modelan los datos imprecisos indexados, se hará empleando un generador aleatorio que sigue una distribución de Zipf con $z = 0.5$.

En lo relativo al tamaño de la base de datos, determinado por el número de elementos indexados y la cardinalidad del dominio subyacente, para las pruebas de rendimiento global éstas tendrán un tamaño de 100.000 elementos y se empleará un dominio subyacente compuesto por 25 elementos.

8.5.3.3. Rendimiento bajo diferentes circunstancias relativas a los datos

La evaluación del rendimiento de las técnicas de indexado consideradas bajo diferentes condiciones relativas a los datos se realizará conduciendo un conjunto de pruebas similares a la descrita anteriormente para la evaluación el rendimiento global. No obstante, en estas pruebas, las bases de datos originales serán modificadas para ajustar sus características al valor fijado en cada una de ellas para el parámetro estudiado. En definitiva, se seguirá la misma metodología descrita en la subsección 8.4.6 para este tipo de evaluaciones. El conjunto de pruebas realizadas se dividen en cuatro experimentos que se describen a continuación.

El primero de ellos evaluará la influencia del tamaño de la base de datos indexada en el rendimiento de los métodos de indexado empleados. Para ello, el tamaño de las bases de datos variará entre los 10.000, 25.000, 50.000, 100.000, 500.000 y 1.000.000 de elementos. De esta forma, por cada uno de los tamaños considerados, se creará un conjunto de 30 bases de datos sobre las que se aplicará un conjunto de consultas generado aleatoriamente, de las mismas características que el empleado para la medición del rendimiento global.

En segundo lugar, se realizará un experimento para determinar la influencia

de la cardinalidad del dominio subyacente. La cardinalidad de éste variará entre los 5, 10, 25, 50, 100 y 125 elementos. Como en el caso anterior, por cada valor se ejecutará una prueba sobre 30 bases de datos generadas aleatoriamente a las que se aplicarán un conjunto aleatorio de consultas.

El tercer experimento evaluará la influencia de la cardinalidad de los soportes o núcleos, según el tipo de medida empleada por las consultas, de las distribuciones de posibilidad que modelan los datos imprecisos indexados. Este parámetro tomará valores dentro del rango $[0, 1]$, con incrementos de 0.1. En los casos en que la cardinalidad relativa se establece a cero, los conjuntos generados no estarán vacíos. Así, siempre se asegura que éstos poseen al menos un elemento. De esta forma, un valor de cero en este parámetro indicará que los datos indexados son *crisp*.En definitiva, este experimento comprende 11 pruebas que se ejecutarán sobre 30 bases de datos generadas aleatoriamente (330 bases de datos en total) y sobre las que se ejecutarán 10000 consultas en total.

Se ha de notar que, aunque el conjunto de consultas que se aplica se extrae del conjunto de datos, se proporcionan mecanismos de transformación para aislar a este primero de la influencia que pueda ejercer la fijación del factor de cardinalidad relativa de los soportes o núcleos de los datos indexados. Esta transformación supondrá la adición o supresión al valor conjuntivo seleccionado de tantos elementos como sea necesario para transformar su cardinalidad relativa a un valor elegido aleatoriamente y con distribución uniforme en el rango $[0, 1]$. De la misma forma que anteriormente, el valor cero para la cardinalidad relativa de los conjuntos que forman la base de las consultas aplicadas indicará que éstas son *crisp*, ya que el mecanismo de generación de consultas asegura siempre la existencia de al menos un elemento en el conjunto asociado a éstas.

En último lugar, se llevará a cabo un experimento para determinar la influencia del sesgo en la distribución de los elementos del dominio subyacente. Para ello, el factor z del generador aleatorio empleado para la selección de los elementos que formarán parte de los conjuntos *crisp* que representan el soporte o núcleo (según la medida que aplican las consultas) de las distribuciones de posibilidad que modelan los datos imprecisos indexados variará en el rango $[0, 2]$, mediante incrementos de 0.2. Por tanto, el conjunto de pruebas se ejecutará sobre 330 bases de datos y comprenderá la aplicación de 11.000 consultas.

8.5.3.4. Rendimiento bajo diferentes circunstancias relativas a las consultas

De la misma forma que los factores relativos a los datos indexados, la influencia de los diferentes parámetros considerados relativos a las consultas ha sido evaluada empleando la misma metodología descrita en la subsección 8.4.6.

En este caso, el conjunto de pruebas llevadas a cabo corresponden a un único experimento destinado a evaluar la influencia de la cardinalidad relativa de la base de las condiciones flexibles sobre el rendimiento de los métodos de indexado considerados. Para ello, se fijará por cada prueba un valor para la cardinalidad relativa de la base de las consultas aplicadas dentro del rango $[0, 1]$ aplicando incrementos de 0.1. En este caso, se seguirá el mismo mecanismo para la generación del conjunto de consultas descrito para las pruebas destinadas a la evaluación general de rendimiento. No obstante, una vez generado dicho conjunto, el conjunto base de cada consulta será modificado, mediante la adición o supresión de elementos, de tal forma que se asegure que la cardinalidad relativa

de los mismos se corresponde con el valor fijado para la prueba. Como se ha descrito para el caso de los datos, y debido a que el generador de consultas asegura que el conjunto base está compuesto por al menos un elemento, un valor de cero para la cardinalidad relativa de la base de las condiciones flexibles indicará que las consultas aplicadas son *crisp*.

Según se ha descrito anteriormente, el experimento anterior constará de 11 pruebas que supondrán la ejecución de 10.000 consultas sobre cada base de datos de un conjunto de 30 generadas aleatoriamente, lo que hará un total de 330.000 consultas ejecutadas.

8.5.4. Resultados

En esta subsección se detallarán los resultados obtenidos tras la ejecución de los experimentos descritos anteriormente. En primer lugar, se expondrán los resultados obtenidos en la evaluación global del rendimiento de los índices considerados. Seguidamente, se mostrarán y analizarán los resultados obtenidos en el conjunto de experimentos diseñados para la evaluación de la influencia de los distintos factores relativos a los datos indexados considerados sobre el rendimiento de los índices. Finalmente, se realizará un análisis análogo para el caso de los factores relativos a las consultas.

8.5.4.1. Rendimiento global

Los experimentos ejecutados, para el caso en que se procesan consultas que aplican una medida de posibilidad, han obtenido un rendimiento medio de 44.9 bloques por consulta para BBC y 141.5 bloques por consulta para IL. Estos valores indican que el rendimiento medio, en condiciones generales, para BBC es superior en un 215 % al del método de referencia. Los resultados de los experimentos han indicado además que el rendimiento de ambos métodos es muy estable, ya que se ha registrado una variación típica inapreciable para ambos caos (0.01 % para BBC y 0.5 % para IL). No obstante, y a pesar de la baja tasa de la variación en el rendimiento medida para ambos métodos, se ha de apreciar que ambas cantidades varían en un orden de magnitud.

En lo que respecta al procesamiento de consultas que aplican una medida de necesidad, el rendimiento medio obtenido por los resultados de los experimentos es de 40.30 bloques por consulta para el caso de BBC, de 83.16 bloques por consulta para el caso de IL y de 161.72 para la variación complementaria de IL. Según los anteriores valores, el rendimiento de BBC es un 106 % superior al método re referencia IL y un 301 % superior al método IL en su versión complementaria. Los resultandos de los experimentos indican, al igual que en el caso en que se procesan consultas que aplican una medida de posibilidad, que las variaciones típicas en el rendimiento de los métodos estudiados no son significativas. Éstas han sido de un 0.08 % en el caso de BC, del 0.26 % en el caso de IL y de 0.34 % para la versión complementaria de este método. Una vez más se observa que, aunque la tasa de variación es muy baja, existe una diferencia notable entre la registrada para BBC y los métodos de referencia.

8.5.4.2. Rendimiento bajo diferentes circunstancias relativas a los datos

En el presente apartado se mostrarán los resultados obtenidos en el conjunto de experimentos dedicados a la determinación de la influencia de los factores relativos a los datos indexados en el rendimiento de las técnicas de indexado evaluadas. En primer lugar, se analizará la influencia del tamaño del conjunto de datos indexados. Seguidamente, se expondrán los resultandos relativos a la evaluación de la influencia de la cardinalidad del conjunto subyacente sobre el que se definen los datos imprecisos escalares que son indexados. En tercer lugar, se mostrarán los resultados relativos a la influencia de la cardinalidad relativa de los soportes o núcleos, según el caso, de las distribuciones de posibilidad que modelan los valores indexados. En último lugar, se mostrarán los datos obtenidos sobre la influencia del sesgo en la distribución de los elementos del dominio subyacente en los conjuntos anteriores.

Influencia del tamaño de la base de datos.

Los resultados arrojados por los experimentos en lo que se refiere a la influencia de la cantidad de datos indexados se muestra, para el caso en que las consultas aplican una medida de posibilidad en la Figura 8.29. En dicha figura, se puede apreciar como la variación de rendimiento de los métodos evaluados es prácticamente lineal en función del número de elementos indexados. No obstante, la pendiente de éstas rectas no es similar. La diferencia de rendimiento de ambos métodos al inicio de la recta es un 250 % y al finalizara la misma es del 272 %. Sobre las anteriores medidas se ha de mencionar que, en la primera parte de la recta que indica el rendimiento de BBC, éste no varía y la diferencia que existe con el método de referencia se reduce hasta el 61.6 %. Esta salvedad en los datos se debe a que la capacidad de compresión de BBC provoca que las cadenas de bits del índice ocupen un tamaño menor o igual a un bloque (la unidad mínima de asignación en un dispositivo de memoria secundaria), lo que impide que se pueda apreciar el menor tamaño de éstas. El tamaño asignado para las cadenas de bits no superará a un bloque hasta que el número de registros de la base de datos no supere a los 25.000. La diferencia media registrada en el rendimiento de los métodos es de un 263 % empleando el rendimiento el rendimiento de BBC como base. De nuevo, los resultados indican una excepcional estabilidad en el rendimiento de los métodos evaluados.

En lo que respecta al caso en que se procesan consultas que aplican una medida de necesidad, el rendimiento de las técnicas consideradas se muestra gráficamente en la Figura 8.30. Al igual que en la ocasión anterior, se observa un rendimiento prácticamente lineal de los métodos evaluados. Para bases de datos pequeñas, el rendimiento de BBC es superior, respectivamente, en un 127 % y 310 % al registrado para IL y su versión complementaria. Esta comparación, al igual que el caso anterior, excluye las mediciones hechas con pequeñas bases de datos para las que BBC es capaz de comprimir la secuencia de bits en un tamaño menor que la unidad de asignación. Para el mayor tamaño de bases de datos analizado, está diferencia es del 133 % y 354 % respectivamente. Como en casos anteriores, las variaciones en el rendimiento registradas a lo largo del experimento han sido mínimas.

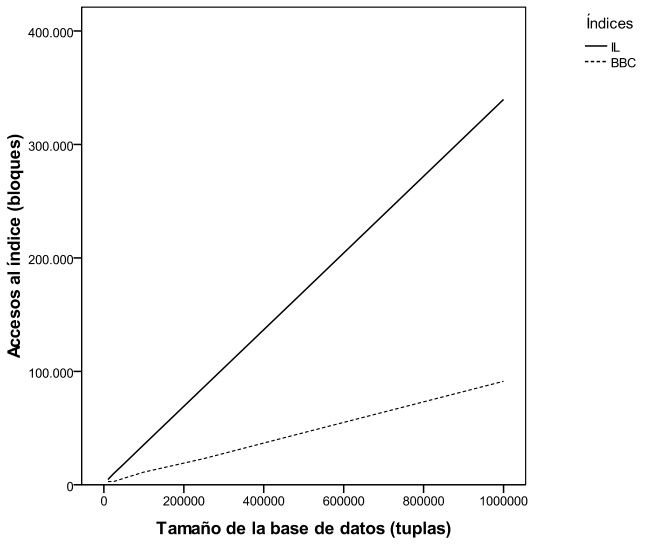

Figura 8.29: Rendimiento de las técnicas para el indexado de datos imprecisos escalares evaluadas bajo distintos tamaños de base de datos para consultas que aplican una medida de posibilidad

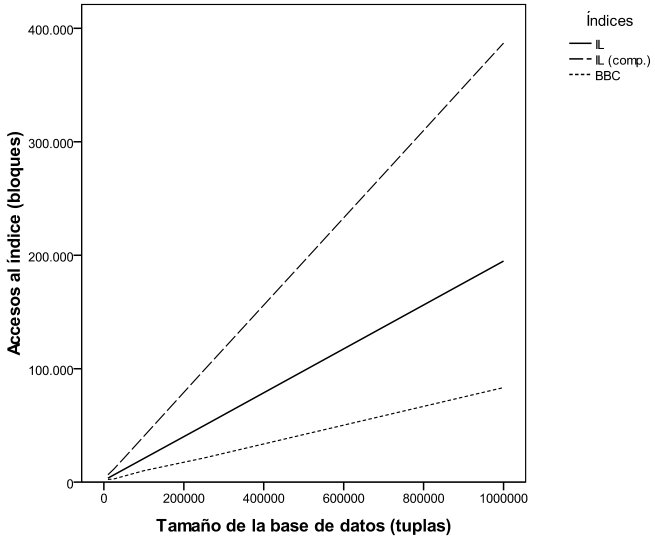

Figura 8.30: Rendimiento de las técnicas para el indexado de datos imprecisos escalares evaluadas bajo distintos tamaños de base de datos para consultas que aplican una medida de necesidad

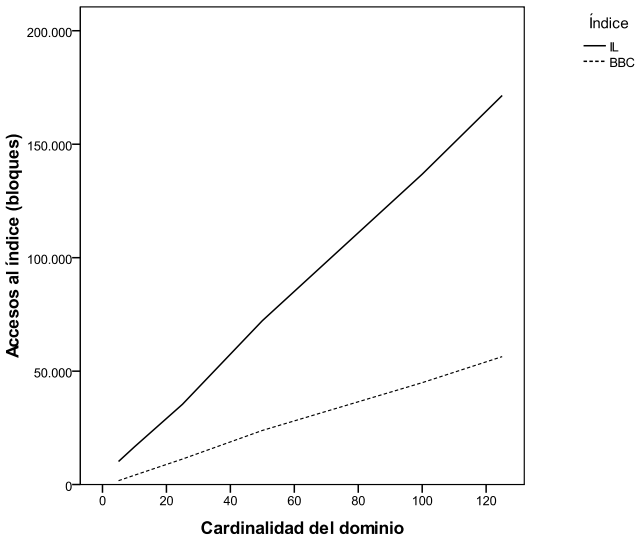

Figura 8.31: Rendimiento de las técnicas para el indexado de datos imprecisos escalares evaluadas bajo distintas cardinalidades del dominio subyacente para consultas que aplican una medida de posibilidad

Influencia de la cardinalidad del dominio subyacente.

Los resultados obtenidos en los experimentos realizados para evaluar el efecto de la cardinalidad del dominio subyacente sobre el rendimiento de las técnicas de indexado consideradas indica que éste ejerce una influencia similar a la medida para el tamaño de la base de datos indexada. La figuras 8.31 y 8.32 indican dicho rendimiento, según las consultas apliquen una medida de posibilidad o necesidad, respectivamente.

En dichas figuras, se puede apreciar una variación prácticamente lineal para los métodos evaluados. En media, el método BBC supera en rendimiento a la técnica IL en un 221 %, en el caso en que las consultas apliquen una medida de posibilidad. Para el caso en que las consultas aplican medidas de necesidad, el rendimiento de BBC es superior en un 230 % al registrado para IL y en un 534 % para el de su variante complementaria.

Influencia de la cardinalidad relativa de los datos indexados.

La cardinalidad relativa de los datos indexados ha mostrado, según los resultados de los experimentos realizados, una influencia significativa en el rendimiento de los métodos evaluados.

Para el caso en que las consultas apliquen una medida de posibilidad, los resultados obtenidos se muestran en la Figura 8.33. En dicha figura, se puede apreciar como el rendimiento de BBC aumenta ligeramente para los casos en que la cardinalidad relativa de los datos es baja o alta. Este comportamiento responde a la posibilidad de que las secuencias de bits contengan secuencias mas largas de ceros o unos que pueden ser representadas con una tasa de compresión más alta por BBC. En cambio, el rendimiento de IL se ve perjudicado por el

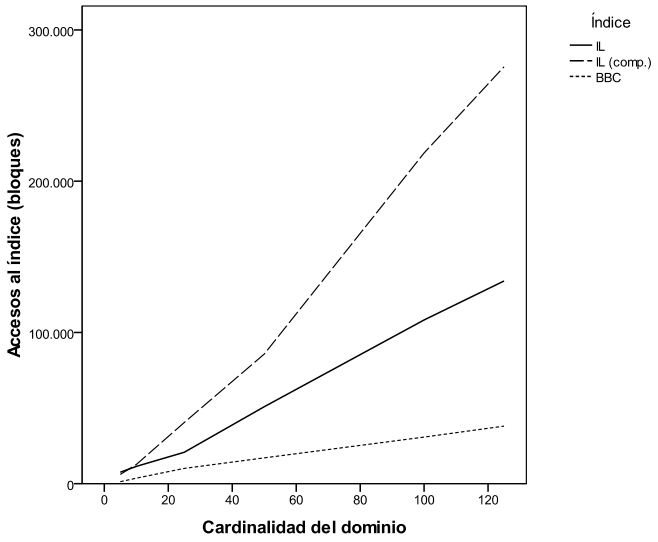

Figura 8.32: Rendimiento de las técnicas para el indexado de datos imprecisos escalares evaluadas bajo distintas cardinalidades del dominio subyacente para consultas que aplican una medida de necesidad

incremento en la cardinalidad relativa de los datos indexados. Dicha disminución de rendimiento responde a un incremento en la longitud de las listas de referencias del índice provocada por la elevación de la cardinalidad de los soportes de las distribuciones de posibilidad que modelan los datos indexados. En cualquiera de los casos el rendimiento de BBC es superior al registrado para la técnica de referencia, variando esta diferencia entre el 44.88 %, en el peor de los casos, y el 790 % para el caso más favorable.

En la Figura 8.34 se muestran los resultados referentes a las pruebas realizadas con consultas que aplican medidas de necesidad. En dicha figura, se puede apreciar cómo el rendimiento de BBC e IL mejora para tasas bajas o altas en la cardinalidad relativa de los datos indexados. El motivo de este comportamiento para el caso de BBC es similar al descrito para el caso en que las consultas aplican medidas de posibilidad. En lo que respecta a IL, este comportamiento está motivado por la disminución de la longitud de las listas de referencias para el caso de una cardinalidad relativa baja de los datos y por el mecanismo de parada, mediante el empleo de las cardinalidades de los elementos indexados, para el caso de una cardinalidad relativa alta. La variante complementaria de IL, por su parte, ve perjudicado su rendimiento con un aumento de la cardinalidad relativa. Este comportamiento se explica por la carencia en esta variante de un mecanismo de parada en base a la cardinalidad de los métodos indexados, lo que obliga a explorar la totalidad de la listas de referencias para resolver una consulta. Las diferencias de rendimiento con respecto a BBC varían, en el caso menos favorable, desde un 17.89 % para IL y un 24.49 % para su variante complementaria. En el caso más favorable, estas diferencias crecen a un 117 % para IL y 736 % para su variante complementaria.

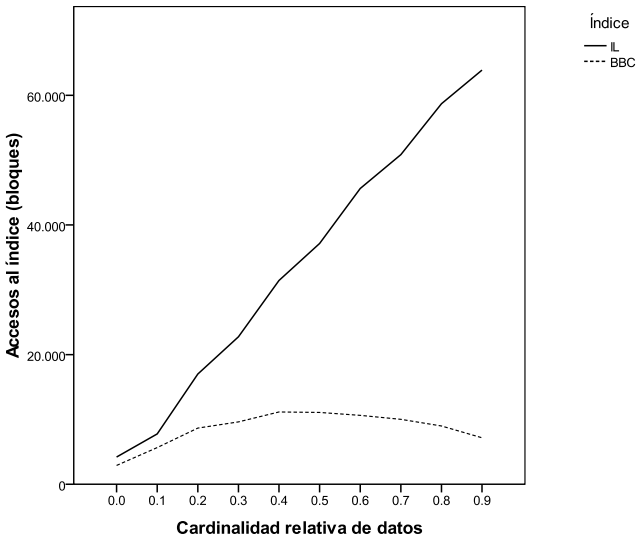

Figura 8.33: Rendimiento de las técnicas para el indexado de datos imprecisos escalares evaluadas bajo distintas cardinalidades relativas de los datos indexados para consultas que aplican una medida de posibilidad

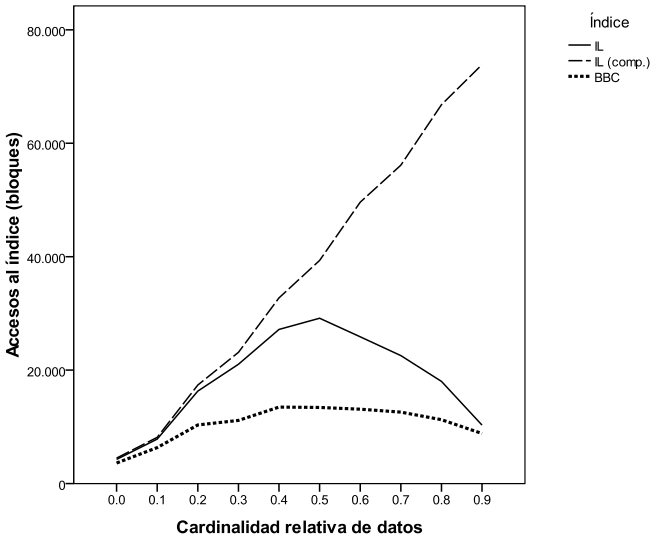

Figura 8.34: Rendimiento de las técnicas para el indexado de datos imprecisos escalares evaluadas bajo distintas cardinalidades relativas de los datos indexados para consultas que aplican una medida de necesidad

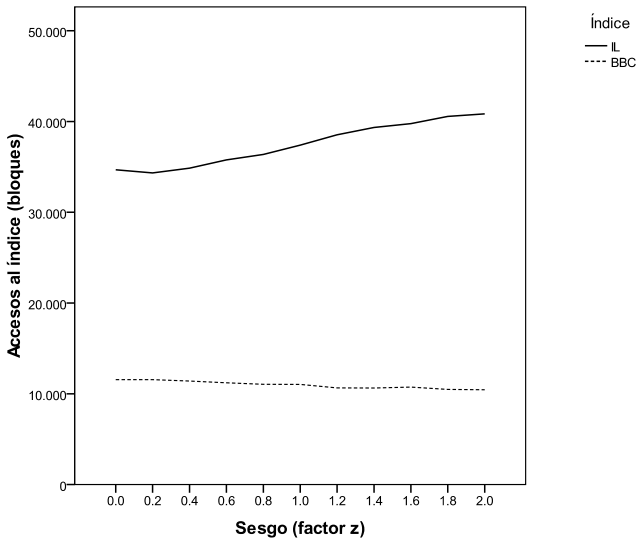

Figura 8.35: Rendimiento de las técnicas para el indexado de datos imprecisos escalares evaluadas bajo distintos grados de sesgo en la distribución de los elementos del dominio subyacente para consultas que aplican una medida de posibilidad

Influencia del sesgo en la distribución de los elementos del dominio subyacente.

En lo que respecta al sesgo en la distribución de los elementos del dominio subyacente, los resultados obtenidos para el caso en que las condiciones aplican una medida de posibilidad se muestran en la Figura 8.35. Aunque en dicha figura no se aprecia claramente debido a la escala, el rendimiento de BBC mejora conforme el sesgo aumenta, llegando esta mejoría a ser del 9.70 %, con respecto a la registrada para una distribución uniforme, en el caso más extremo. Esta mejora es debida al mayor ratio de compresión de BBC gracias al aumento de la frecuencia y longitud de las secuencias de bits con el mismo valor. En cambio, IL sufre una degradación de su rendimiento conforme la distribución de los elementos del dominio subyacente se vuelve más sesgada. Esta degradación llega hasta un 17.36 %, con respecto al rendimiento medido para una distribución normal, para el caso en que el sesgo es máximo. Este decremento en el rendimiento de IL es debido a que, cuanto mayor es el sesgo en la distribución de los elementos del dominio subyacente, los conjuntos indexados contendrán muy frecuentemente los mismos elementos y las consultas incluirán dichos elementos con mucha frecuencia. La alta frecuencia de aparición de un elemento en los conjuntos indexados provoca que la lista de referencias asociada al mismo sea de gran longitud. Este incremento en la longitud de las listas, combinado con que dichos elementos aparecen con mucha frecuencia en las consultas, supone que la resolución requiere el recorrido muy frecuente de listas de referencias cada vez más largas.

Para el caso en que las condiciones aplican una medida de necesidad, los

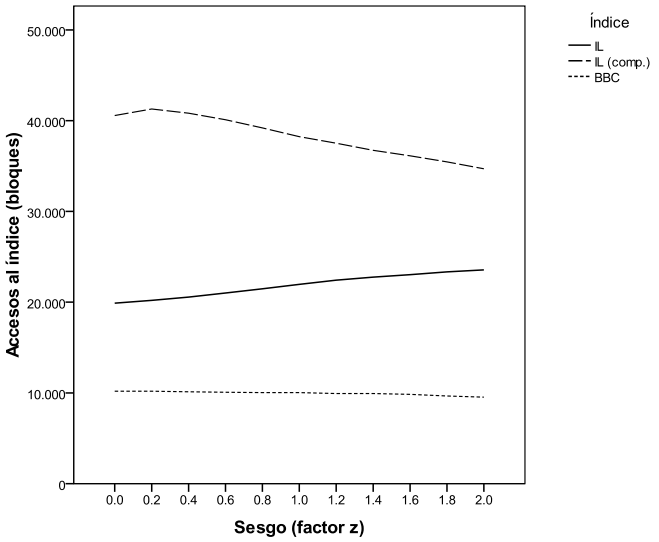

Figura 8.36: Rendimiento de las técnicas para el indexado de datos imprecisos escalares evaluadas bajo distintos grados de sesgo en la distribución de los elementos del dominio subyacente para consultas que aplican una medida de necesidad

resultados, mostrados en la Figura 8.36, son similares a los anteriores para BBC e IL. El primero incrementa su rendimiento según aumenta el sesgo en la distribución de los elementos del dominio subyacente, llegando este incremento a ser de un 6.4 % cuando el sesgo es máximo, con respecto al rendimiento medido para una distribución normal. Para IL, la degradación de rendimiento llega al 17.36 %, para el grado máximo de sesgo considerado, con respecto al rendimiento medido para una distribución normal. Así mismo, el incremento para el caso de BBC y el decremento de rendimiento de IL se deben a las mismas razones expuestas para el caso anterior. En lo que respecta a la variante complementaria de IL, ésta ve incrementado su rendimiento según aumenta el grado de sesgo. En el mejor de los casos, cuando el grado de sesgo es máximo, éste es un 14.18 % superior al registrado para una distribución uniforme. Como se indicó anteriormente, para altos grados de sesgo las listas de referencias de los elementos muy frecuentes crecen en longitud. Por tanto, como efecto colateral, conforme dicho sesgo crece la longitud de la lista de referencias asociada a los elementos poco frecuentes decrece. Dado que la variante complementaria de IL accede a las listas de referencias de los elementos no presentes en el valor conjuntivo que se emplea como condición y teniendo en cuenta que, dado un alto grado de sesgo, las consultas contendrán en su mayoría elementos muy frecuentes, los elementos no presentes en el conjunto condición serán elementos poco frecuentes. Por tanto, la variante complementaria de IL accede, conforme crece el grado de sesgo, a listas de referencias cada vez más cortas, lo que provoca el crecimiento de su rendimiento.

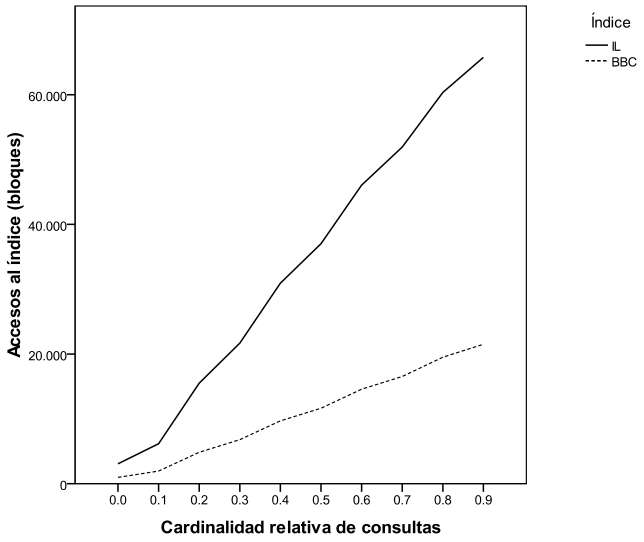

Figura 8.37: Rendimiento de las técnicas para el indexado de datos imprecisos escalares evaluadas bajo distintas cardinalidades relativas de las consultas cuando éstas aplican una medida de posibilidad

Influencia de la cardinalidad relativa de las consultas.

En la Figura 8.37 se muestran los resultados de los experimentos dedicados a la evaluación de la influencia de la cardinalidad relativa de la base de las consultas flexibles en el rendimiento de las técnicas consideradas, para el caso en que las consultas aplican una medida de posibilidad. En dicha figura, se puede observar cómo el rendimiento de ambos métodos evaluados decrece conforme aumenta el valor del parámetro estudiado. Este decrecimiento viene motivado por el aumento del número de elementos presente en la base de las consultas flexibles aplicadas, lo que provoca que, para la resolución de éstas, los métodos evaluados deban acceder a un mayor número de listas de referencias o secuencias de bits, según el caso. La diferencia de rendimiento entre los métodos varía entre un 206 %, en el mejor de los casos y un 220 % en el peor de ellos.

El comportamiento del rendimiento de los métodos evaluados, en lo que respecta al caso en que las consultas aplican una medida de necesidad, se muestra en la Figura 8.38. Como se puede observar en dicha figura, la respuesta de los diferentes métodos considerados es dispar. El rendimiento decrece conforme aumenta el factor estudiado y posteriormente (una vez rebasado el valor 0.4) éste crece. El decremento observado inicialmente viene provocado por el incremento del número de elementos en la base de las consultas flexibles, lo que implica el acceso a un número creciente de secuencias de bits y, por tanto, un decremento en el rendimiento. El incremento de rendimiento observado posteriormente, se debe al procesamiento, a partir de ese punto, de las consultas empleando el método complementario. Dado que según se incrementa el factor estudiado aumenta el número de elementos en la base de la consulta flexible, la aplicación del método complementario de resolución de consultas resulta en una disminución

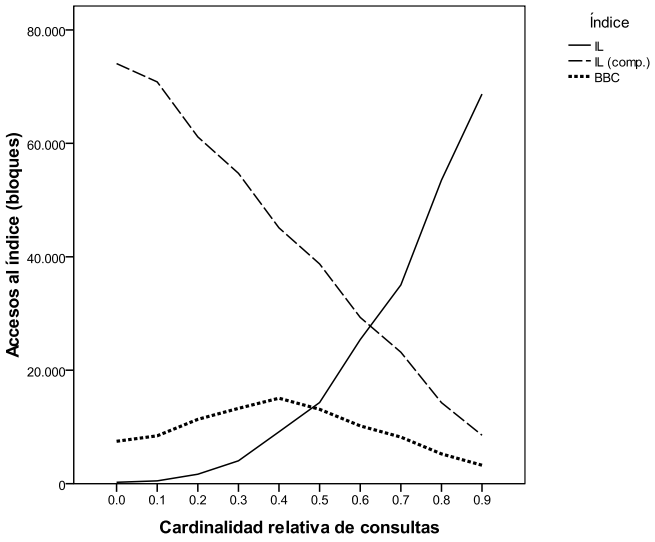

Figura 8.38: Rendimiento de las técnicas para el indexado de datos imprecisos escalares evaluadas bajo distintas cardinalidades relativas de las consultas cuando éstas aplican una medida de necesidad

del número de secuencias de bits que han de ser accedidas. Aunque el rendimiento de IL es muy positivo para valores bajos del parámetro estudiado, los resultados muestran una disminución de su rendimiento conforme la cardinalidad relativa de las consultas aumenta. Esta degradación del rendimiento viene motivada, como en el primer caso de BBC, por un aumento en el número de secuencias de referencias que han de ser accedidas para la resolución de las consultas. En lo que respecta a la variante complementaria de IL, su rendimiento aumenta según se incrementa el factor estudiado. Una vez más, este incremento viene motivado por la técnica de resolución de consultas que aplica, por el que, conforme aumenta el número de elementos en la consulta a resolver, el número de listas de referencias que se deben recorrer decrece.

Como se puede observar, la diferencia de rendimiento de los métodos BBC e IL depende de la cardinalidad relativa de las consultas. Para valores bajos del parámetro el rendimiento de IL es superior al de BBC en una media del 1070 %. Una vez la cardinalidad relativa de las consultas supera el valor 0.4, el rendimiento de BBC es superior al de IL en un 684 %. En lo que respecta a la variante complementaria de IL, el rendimiento de BBC es superior en un rango que varía entre 163 %, en el peor de los casos y 861 % en el caso más positivo, resultando en una diferencia media del 348 %.

Nótese que, según se muestra en la Figura 8.38, el rendimiento de la técnica BBC iguala y, posteriormente, supera al del IL, cuando la cardinalidad relativa de la base de las consultas es igual o superior a un valor cercano a 0.5. Este valor, en que se produce este cambio de tendencia, coincide con la cardinalidad media de los datos indexados, ya que ésta está uniformemente distribuida en el rango [0,1]. La coincidencia de estos valores es un indicio que nos permite

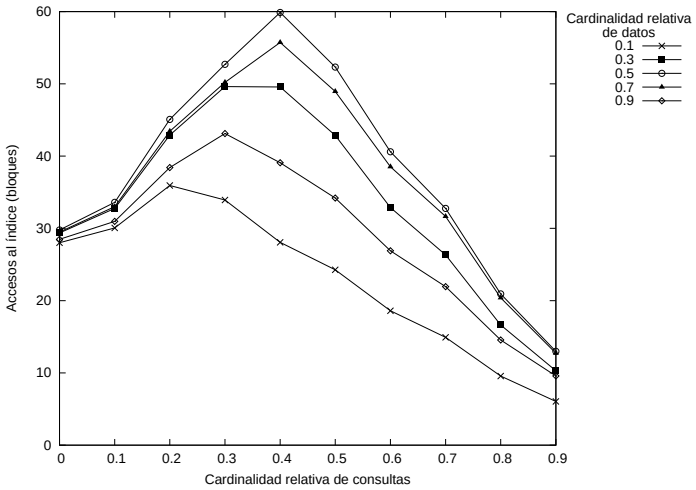

Figura 8.39: Rendimiento de la técnica de indexado BBC evaluada bajo distintas cardinalidades relativas de datos y consultas cuando estas últimas aplican una medida de necesidad

explicar el rendimiento superior de IL para los casos en que la cardinalidad relativa de la base de las consultas en baja. Cuando la cardinalidad relativa de la base de la consulta a resolver por IL es menor que la cardinalidad relativa media de los datos indexados, el mecanismo de parada en base a la cardinalidad de esta técnica permite terminar el recorrido de las listas de referencias de forma temprana. En cambio, cuando la cardinalidad relativa de la base de las consultas es igual o superior a la cardinalidad relativa media de los datos indexados, el fragmento de las listas de referencias que se ha de recorrer es cada vez mayor, lo que incide negativamente en el rendimiento de IL.

El anterior análisis indica que el rendimiento de esta técnica se ve afectado por la relación entre los valores medios de la cardinalidad relativa de consultas y datos. Esta suposición se confirma por los resultados obtenidos tras la ejecución de una serie de pruebas, siguiendo la misma metodología descrita para el resto de experimentos, en las que se ha fijado el valor para estos dos parámetros. En las figuras 8.39, 8.40 y 8.41 podemos observar, respectivamente, los resultados de rendimiento obtenidos en la resolución de consultas que aplican una medida de necesidad bajo distintos escenarios de cardinalidad relativa de datos y consultas para las técnicas BBC, IL y la variante complementaria de IL.

Obsérvese particularmente cómo en la Figura 8.40 se aprecia claramente el efecto de la relación de los citados parámetros sobre el rendimiento de IL descrito anteriormente. La curva del número de accesos al índice en la citada figura se mantiene a niveles bajos, con un crecimiento lineal, hasta que la cardinalidad relativa de las consultas se iguala a la cardinalidad relativa de los datos. Al llegar a este punto, se produce un incremento significativo en el valor de esta curva. Tras el cambio el brusco cambio de nivel en la curva, esta vuelve a crecer de forma lineal conforme se incrementa la cardinalidad relativa de las consultas.

Para el caso de BBC, cuyo rendimiento se puede observar en la Figura 8.39, se aprecia un comportamiento más estable que el observado anteriormente para IL. En dicha figura, se puede apreciar un cambio de tendencia en el rendimiento

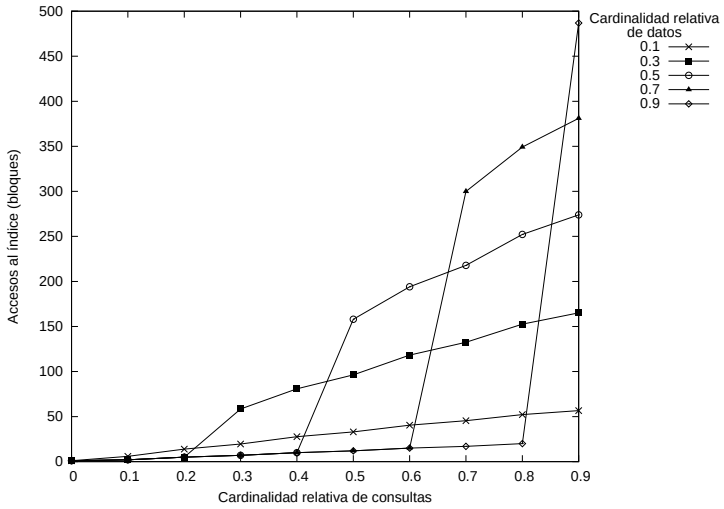

Figura 8.40: Rendimiento de la técnica de indexado IL evaluada bajo distintas cardinalidades relativas de datos y consultas cuando estas últimas aplican una medida de necesidad

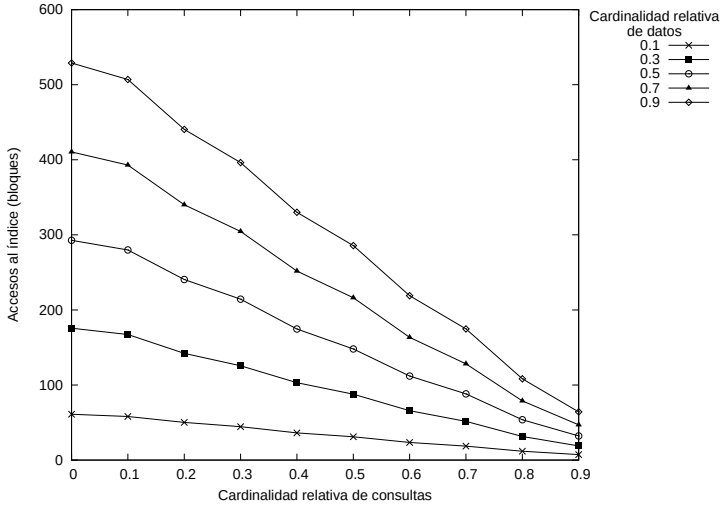

Figura 8.41: Rendimiento de la técnica de indexado IL en su variante complementaria evaluada bajo distintas cardinalidades relativas de datos y consultas cuando estas últimas aplican una medida de necesidad

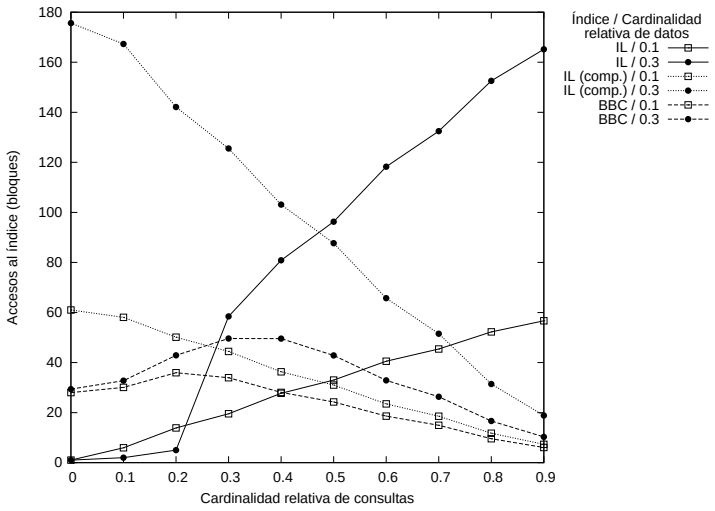

Figura 8.42: Rendimiento de las técnicas para el indexado de datos imprecisos escalares evaluadas a niveles bajos de cardinalidades relativas de datos y consultas cuando estas últimas aplican una medida de necesidad

de BBC cuando el valor de la cardinalidad relativa de las consultas se aproxima ligeramente por debajo al valor de la cardinalidad relativa de datos. Esto indica que, al igual que IL, la combinación de estos dos factores influye de forma clara sobre el rendimiento de la técnica de indexado. No obstante, en el caso de BBC, la llegada al punto crítico significa una mejora en el rendimiento medido. Este cambio de tendencia puede atribuirse la conmutación del mecanismo de resolución de consultas subconjuntivas de directo a complementario. Nótese que el punto en que el rendimiento de BBC cambia su tendencia no se sitúa exactamente cuando la cardinalidad relativa de datos y consultas se iguala. Éste, como se ha comentado anteriormente, está situado cuando el valor de la cardinalidad relativa de las consultas se aproxima ligeramente por debajo al valor de la cardinalidad relativa de datos debido al sobrecoste que supone el acceso a la información relativa a la cardinalidad de los datos indexados para el método directo de resolución de consultas subconjuntivas de BBC.

Para el caso de la variación complementaria de IL, mostrado en la Figura 8.41, no se observa que exista una interacción entre el par de parámetros estudiados que incida en el rendimiento. El rendimiento de la técnica aumenta según lo hace la cardinalidad relativa de las consultas, tal y como se observó anteriormente en este mismo apartado según los resultados mostrados en la Figura 8.38. Independientemente, el rendimiento de la técnica disminuye conforme aumenta la cardinalidad relativa de los datos, resultado que concuerda con los observados en el estudio sobre el impacto de este parámetro cuyos resultados se muestran en la Figura 8.34.

Antes de comparar el rendimiento de estas técnicas, nótese la diferencia de escala de los gráficos mostrados en las figuras 8.39, 8.40 y 8.41. Para facilitar la comparación, en las figuras 8.42 y 8.43 se muestra conjuntamente el rendimiento medido para las técnicas evaluadas respectivamente para casos en que la cardinalidad relativas de datos es baja y alta.

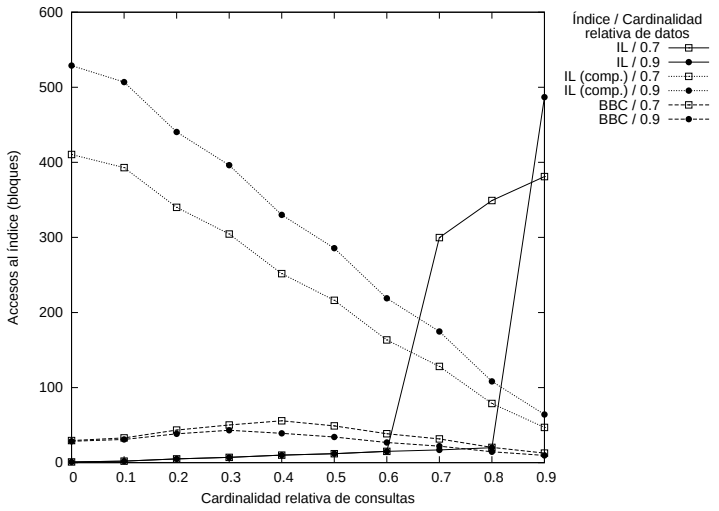

Figura 8.43: Rendimiento de las técnicas para el indexado de datos imprecisos escalares evaluadas a niveles altos de cardinalidades relativas de datos y consultas cuando estas últimas aplican una medida de necesidad

En los casos de muy baja cardinalidad en los datos, valor 0.1, se puede observar cómo el rendimiento de IL es superior al de BBC hasta que la cardinalidad relativa de las consultas es cuatro veces superior. Tras dicho valor, la capacidad de conmutación del método de resolución de las consultas conjuntivas de BBC hace que ésta técnica posea un rendimiento cada vez mejor. En los casos de baja cardinalidad de datos, 0.3, IL pierde su ventaja a valores menores de cardinalidad relativa de datos, siendo el cambio en este caso algo antes de que la cardinalidad relativa de consultas iguale a la de datos. La variante complementaria de IL no ofrece un rendimiento competitivo con el obtenido para BBC en ninguno de los casos.

Para los casos de alta cardinalidad se observa como el rendimiento de IL es muy dispar con respecto al de BBC en función de la relación entre los valores de cardinalidad relativa de datos y consultas. Siempre que la cardinalidad relativa se sitúa por debajo de la de los datos, el rendimiento de IL es superior al de BBC. Una vez superado ese umbral, el rendimiento de IL es extremadamente inferior al de BBC. Esta diferencia observada se agrava según aumenta la cardinalidad relativa de los datos.

Los buenos resultados obtenidos por IL para los escenarios de muy baja cardinalidad de los datos indexados confirman los obtenidos en [79]. No obstante, se ha de notar que en el caso de las bases de datos difusas no se espera encontrar escenarios en que la cardinalidad de datos o consultas sea muy baja. En una base de datos difusa típica la cardinalidad de los dominios escalares subyacentes suele ser baja ya que, en general, éstos suelen ser definidos por expertos. Además, estos dominios suelen tener asociada una relación de similitud entre sus elementos, normalmente definida también por expertos, lo que conlleva el uso de dominios escalares de cardinalidad baja. La baja cardinalidad de los dominios escalares subyacentes, junto con el hecho de que en la base de datos se representan valores imprecisos que contendrán varios elementos en sus distribuciones de posibilidad,

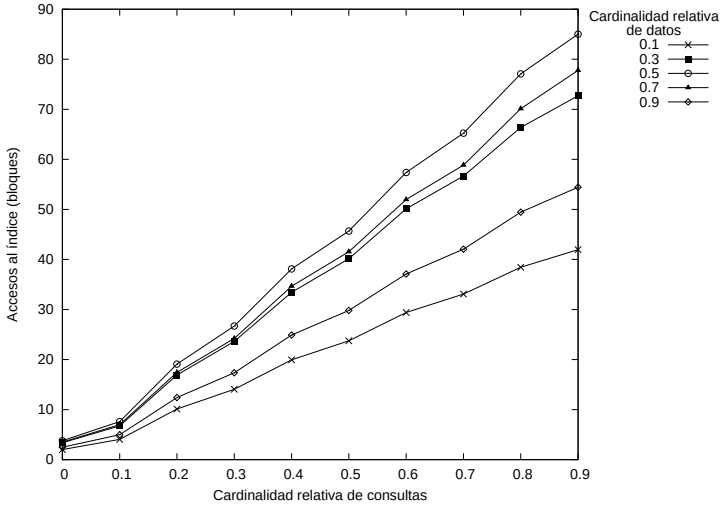

Figura 8.44: Rendimiento de la técnica de indexado BBC evaluada bajo distintas cardinalidades relativas de datos y consultas cuando estas últimas aplican una medida de posibilidad

hace que la cardinalidad relativa media de los datos indexados no se sitúe en valores particularmente bajos. Adicionalmente, se ha de tener en cuenta que en un entorno en que se ejecutan consultas flexibles, para aprovechar la capacidad que éste ofrece a diferencia de los SGBD *crisp*, es habitual que las condiciones empleadas sean algo más imprecisas que los datos indexados. Esta situación, por tanto, haría que la cardinalidad relativa típica de las consultas sea igual o superior a la de los datos.

El mismo estudio, cuyos resultados se muestran en las figuras 8.44 y 8.45 sobre la interacción entre los factores de cardinalidad relativa de datos y consultas se ha llevado a cabo en el caso en que éstas aplican una medida de posibilidad. Los resultandos obtenidos no muestran, como en el caso anterior de la variante complementaria de IL, la existencia de una interacción entre el par de parámetros estudiados que incida en el rendimiento. En la Figura 8.44 se muestra cómo el rendimiento de BBC disminuye según aumenta la cardinalidad relativa de las consultas. En lo que respecta la cardinalidad relativa de los datos, su aumento significa una reducción del rendimiento de BBC hasta que su valor supera 0.5. A partir del citado punto, el rendimiento de BBC aumenta conforme lo hace la cardinalidad relativa de los datos. En el caso de IL, la Figura 8.45 muestra como su rendimiento disminuye conforme aumenta la cardinalidad relativa de datos o consultas.

Este resultados son coherentes con los obtenidos anteriormente, mostrados en las figuras 8.33 y 8.37, para los estudios de la influencia de la cardinalidad relativa de datos y consultas.

8.5.5. Conclusiones

En la presente sección se ha evaluado el rendimiento del método BBC en el indexado de datos imprecisos escalares empleando como referencia la técnica IL

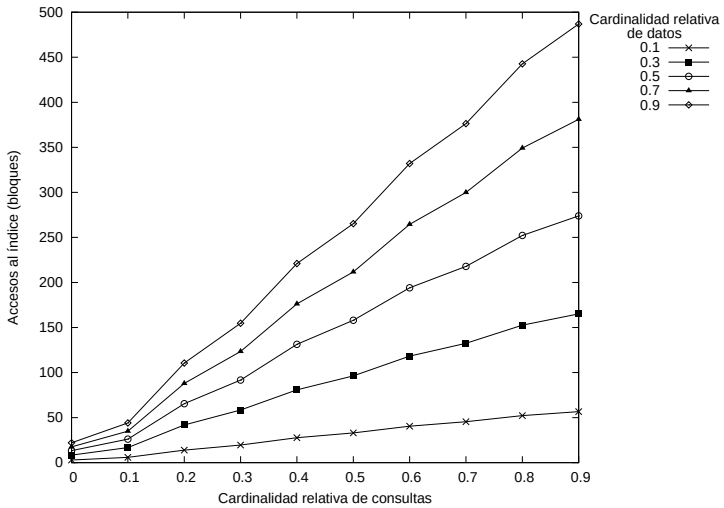

Figura 8.45: Rendimiento de la técnica de indexado BBC evaluada bajo distintas cardinalidades relativas de datos y consultas cuando estas últimas aplican una medida de posibilidad

y su variante complementaria.

Los resultados de los experimentos realizados han mostrado un excelente comportamiento del método BBC en su aplicación sobre una base datos difusa. Las pruebas han indicado un rendimiento medio global de BBC, para el caso en que las consultas aplican una medida de posibilidad,superior en más del doble al registrado para la técnica de referencia. A pesar de que la estabilidad del método de referencia es muy alta, el método propuesto ha mostrado una estabilidad superior en un orden de magnitud a la registrada para IL. Se ha de notar que el rendimiento de BBC ha sido superior al del método de referencia bajo las diferentes condiciones particulares de datos y consultas simuladas en las pruebas realizadas. Así mismo, la influencia de los factores considerados en las diversas pruebas ha sido en general menos negativa que para en método de referencia. Finalmente se ha de notar que, en el caso del aumento en el sesgo en la distribución de los elementos del dominio subyacente, BBC ha mostrado, a diferencia de IL, un incremento de su rendimiento, lo que hace este método más apropiado que el de referencia para situaciones en que la frecuencia de aparición de los elementos del dominio subyacente no es uniforme.

En lo que se refiere al rendimiento de BBC cuando las consultas aplican una medida de necesidad, éste método ha demostrado, al igual que el caso anterior un rendimiento global superior a las técnicas de referencia. Éste es algo más de 1.5 veces superior al de IL y tres veces superior al de su variante complementaria. Igualmente, bajo condiciones particulares de datos y consultas, BBC ha demostrado un rendimiento superior y un menor impacto de los diversos factores en su rendimiento. Cabe destacar, como se ha estudiado en profundidad, la excepción registrada para el caso particular en que la cardinalidad relativa de las consultas se mantiene por debajo de la cardinalidad relativa media de los datos indexados, en cuyo caso IL supera en rendimiento a BBC.

En conclusión, podemos decir que teniendo en cuenta el rendimiento mos-

trado por BBC para indexar datos imprecisos escalares, superando en general a los métodos de referencia, hacen de éste un buen candidato para su uso en un SGBDD. Particularmente, se ha de destacar que, debido a que BBC ya está disponible en SGBDs clásicos con implementación muy optimizada, éste será un buen candidato para su empleo como mecanismo de indexado en SGBDs implementados como extensión de los anteriores.

8.6. Conclusiones

En el presente capítulo se han propuesto diversas técnicas de indexado para datos imprecisos. Concretamente, se han propuesto un par de técnicas para datos imprecisos numéricos, basadas en árboles B^+ y denominadas 2BPT y HBPT, y una técnica para la indexar datos imprecisos de naturaleza escalar, basada mapas de bits, denominada BBC.

En el par de técnicas propuesto para indexar datos imprecisos numéricos, 2BTP está dedicada a la resolución de consultas cuando se aplica una medida de posibilidad y HBPT está dedicada para la resolución de éstas cuando aplican una medida de posibilidad. Estas técnicas han demostrado un rendimiento comparable, o ligeramente inferior en el caso de 2BPT (con un 5.6 % de diferencia media) y superior en el caso de HBPT (con una mejora del 4.2 %), a la técnica empleada como referencia. Además de un rendimiento aceptable, las técnicas han demostrado, a diferencia de la técnica de referencia, una gran estabilidad ante las diversas condiciones específicas de datos y consultas.

El método BBC puede ser empleado, a diferencia de las técnicas anteriores, tanto para la resolución de consultas que aplican una medida de posibilidad o necesidad, siendo necesario en el segundo caso la adición de una entrada más en el directorio del índice. Este método ha demostrado un gran rendimiento, superando entre un 50 % y 100 % (en función de la medida aplicada por las consultas) al mejor rendimiento obtenido por las técnicas de referencia. Además, el rendimiento medido para BBC ha resultado muy estable con independencia de las condiciones particulares de datos y consultas, superando la estabilidad del rendimiento que presentan los métodos de referencia.

El rendimiento y estabilidad de las técnicas propuestas, unido al hecho de que éstas técnicas emplean estructuras de indexado de uso común en la mayoría de los SGBD clásicos, hacen de 2BPT, HBPT y BBC una buenas candidatas para ser empleadas como técnica de indexado en SGBDDs, especialmente cuando éstos se desarrollan como una extensión de un SGBD clásico.

Capítulo 9

Aplicaciones

9.1. Introducción

En el presente capítulo se presentan algunos ejemplos de aplicación que se apoyan en el SGBDORD propuesto en el presente trabajo. En dichos ejemplos, se ilustran las capacidades de representación y manejo de información compleja e imperfecta de dicho SGBDORD, así como las posibilidades que éste ofrece en lo que se refiere a la definición y procesamiento de consultas flexibles. Al mismo tiempo, y de forma recíproca, se mostrarán las ventajas que aporta el uso de un SGBDORD (y, por extensión, el procesamiento natural de información compleja e imperfecta) como parte de los sistemas de aplicación ilustrados.

Los casos de aplicación que abordaremos serán tres: la búsqueda inmobiliaria, un sistema de recuperación de imágenes en base a colores dominantes y un sistema de recuperación de imágenes médicas en base a la descripción de la patología. El caso de la búsqueda inmobiliaria permitirá ilustrar la capacidades de búsqueda flexible que ofrece el sistema, así como los beneficios que ésta capacidad puede reportar a esta clase de aplicaciones. En segundo lugar, el caso de la recuperación de imágenes en base a colores dominantes ilustrará, además de la mencionada capacidad para la búsqueda flexible, las capacidad de representación y manejo de información imperfecta proveniente de descripciones empleando el lenguaje natural (en este caso, la descripción de colores). En último lugar, la aplicación de recuperación de imágenes médicas permitirá ilustrar, además de las características anteriores, la capacidad del sistema para el manejo de información afectada por imprecisión (como, por ejemplo, las mediciones médicas).

9.2. Búsqueda inmobiliaria

En la presente sección, proponemos el uso de un SGBDORD para mejorar los procesos de búsqueda de ofertas comerciales asistidos por computador, centrándonos particularmente en el área de la gestión inmobiliaria.

La integración transparente de aplicaciones orientadas a objetos con SGBDs objeto-relacionales y la disponibilidad para estos SGBDs de una extensión para el manejo de datos difusos, se combinan para facilitar el desarrollo de capacidades mejoradas para la gestión de datos en aplicaciones comerciales. Concre-

tamente, la combinación de estos dos componentes permitirá realizar, de forma sencilla, el almacenamiento y manejo de información imperfecta, así como la consulta de datos utilizando condiciones flexibles.

En este caso particular, se propone utilizar las nuevas capacidades descritas anteriormente para mejorar la interacción entre usuario y aplicación en sistemas de búsqueda de ofertas comerciales, permitiendo a los vendedores expresar dichas ofertar de forma más natural (por ejemplo, empleando descripciones en lenguaje natural que se corresponden con valores imprecisos o inciertos) y posibilitando a los sistemas informáticos el procesamiento de consultas flexibles realizadas por los usuarios. Esta forma de expresar consultas y ofertas hace más natural la interacción con el sistema, emulando el proceso flexible en el que un agente de ventas empareja ofertas y demandas.

9.2.1. El problema de la búsqueda inmobiliaria

El proceso de búsqueda inmobiliaria ha sido elegido como objeto de estudio por la idoneidad de los atributos de los inmuebles para el tratamiento difuso. Los valores asociados a éstos suelen poseer cierto grado de imprecisión o incertidumbre. Esta vaguedad en los datos proviene usualmente de las fuentes de información, en la mayor parte de los casos descripciones empleando lenguaje natural.

En el proceso de búsqueda inmobiliaria se especifica el conjunto de características que deben poseer los inmuebles resultantes pero, por lo general, esas características no siempre están definidas de forma precisa. Un cliente suele tener un conjunto de preferencias, una idea aproximada de lo que está buscando. Esa idea no tiene que ajustarse necesariamente a un valor preciso, puede que esté mejor representada por un rango de valores, un valor aproximado o incluso un límite superior o inferior. Representar esas características de forma difusa permite obtener resultados que verifiquen las preferencias en distintos grados.

La información imperfecta se gestiona de forma intuitiva por los agentes de ventas, que pueden tratar y procesar ésta sin dificultad. En el proceso de búsqueda inmobiliaria, las transacciones se realizan entre dos personas, el cliente y el agente de ventas, pudiendo ambos plantear y atender condiciones flexibles de forma natural.

El problema se plantea cuando uno de estos dos agentes, el agente de ventas en nuestro caso, es reemplazado por un sistema automático. En esa situación, es necesario dotar a dicho sistema de métodos para tratar la información imperfecta de forma similar a como la trata un agente real. En síntesis, se propone una nueva aproximación en la que se permitirá la especificación imperfecta de las características de los inmuebles y acepte condiciones flexibles expresadas por los clientes, con la finalidad de diseñar un sistema capaz de imitar el comportamiento de un agente inmobiliario que pueda interactuar de forma natural con el cliente.

9.2.2. Atributos difusos de inmuebles

Los atributos de inmuebles pueden ser modelados utilizando alguno de los tipos difusos descritos en el Capítulo 7 del presente trabajo. De entre la gran variedad de atributos que definen un inmueble, se han seleccionado aquellos más representativos y que pueden ilustrar mejor el ejemplo que proponemos, a

Tipo Difuso	Atributos
OAFT	Precio
	Superficie
	Habitaciones
	Plantas
	Antigüedad
NOAFT	Clase
	Orientación
	Iluminación
	Vistas
	Conservación
CFC	Extras

Tabla 9.1: Tipos asociados a atributos de inmuebles

Piso	*Casa*	*Dúplex*	*Ático*	**Clase**
0.75	0.3	0.2	0.75	*Apartamento*
	0.3	0.3	0.75	*Piso*
		0.75	0.1	*Casa*
			0.1	*Dúplex*

Tabla 9.2: Relación de proximidad definida sobre el dominio *Clase*

continuación, en la presente subsección. Los atributos seleccionados son los que se muestran en la Tabla 9.1.

Los tipos OAFT son empleados para almacenar datos numéricos imprecisos, representados mediante distribuciones de posibilidad trapezoidales. Por ejemplo, el atributo *Precio* puede tener asociado un valor que represente en rango de precios para un inmueble "entre 100.000 € y 500.000 €" o "hasta 150.000 €". Este ejemplo es igualmente aplicable para los atributos *Superficie* y *Antigüedad*. No será usual que para los atributos *Habitaciones* y *Plantas* se empleen valores imprecisos, ya que éstos atributos son fácilmente cuantificables y la imprecisión necesaria es mínima. Por tanto, en estos casos, se emplearán valores *crisp*, aceptados también como valores de los tipos OAFT (ya que pueden verse como un caso particular de los valores imprecisos). No obstante, dado que el tipo de estos atributos es OAFT, si en algún caso fuese necesario indicar un valor impreciso para los atributos mencionados, se podrá emplear en cualquier caso las capacidades de representación de datos imprecisos del citado tipo.

Los tipos NOAFTs permiten la representación de datos imprecisos definidos mediante una distribución de posibilidad sobre un dominio escalar en el que se ha definido una relación de proximidad. Por ejemplo, el atributo *Clase* tiene asociado un dominio compuesto por los escalares "Apartamento", "Piso", "Casa", "Dúplex" y "Ático". La relación de proximidad, entre pares de elementos, asociada al dominio del atributo *Clase* se muestra en la Tabla 9.2. Cada uno de los restantes atributos definidos sobre este tipo (*Orientación*, *Iluminación*, *Vistas* y *Conservación*) tienen su propio dominio escalar y una relación de proximidad asociada.

Para almacenar los extras de los inmuebles se emplea valor del tipo CFC.

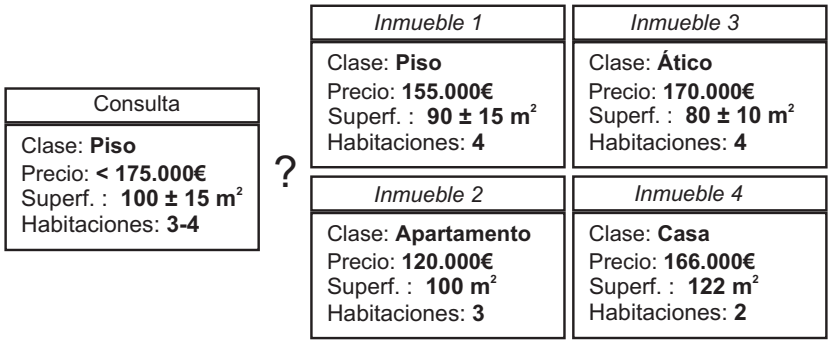

Figura 9.1: Ejemplo de base de datos y búsqueda inmobiliaria

Estos extras podrían ser, por ejemplo, "Jardín", "Pista de tenis", "Chimenea", "Piscina", "Sótano", "Patio", etc. Por simplicidad, cada extra se añade a la CFC con un grado de pertenencia igual a 1, aunque el anterior grado podría emplearse para codificar información cualitativa del extra. Durante la búsqueda cada par de conjuntos de extras es comparado para obtener el grado de semejanza entre colecciones.

Finalmente, el conjunto de datos relativo a un inmueble puede ser agrupado en un tipo compuesto. De esta forma se simplifica su manipulación por parte de las aplicaciones que interactúen con la base de datos y estén programadas en un lenguaje orientado a objetos. Para ello, se empleará un tipo FO. El empleo de este tipo de datos difuso favorecerá, como se verá a continuación, la expresión y aplicación de consultas flexibles sobre el conjunto de inmuebles.

Obviamente, esta propuesta puede extenderse para abarcar un sistema más complejo. Remitimos al lector a [8] para obtener más detalles sobre el esquema propuesto.

9.2.3. Búsqueda inmobiliaria flexible

La presente subsección tiene como objetivo ilustrar la creación de una base de datos conteniendo un conjunto de inmuebles definidos en base a valores imprecisos y mostrar la definición de una consulta incluyendo términos flexibles sobre dicho conjunto. Por motivos de simplicidad, en este ejemplo se utilizará sólo un subconjunto de los atributos mostrados anteriormente, ya que el procedimiento es idéntico para atributos cuyo tipo de datos es el mismo.

9.2.3.1. Representación de inmuebles y consultas

La Figura 9.1 muestra el conjunto de inmuebles y la consulta que consideraremos en el presente ejemplo.

Realizaremos la búsqueda sobre un conjunto de FOs que representan inmuebles construyendo un FO del mismo tipo que represente la consulta. Los valores en el FO consulta asociados a cada uno de sus atributos modelarán las condiciones que han cumplir, por cada atributo, los inmuebles seleccionados. Esta forma de representar las condiciones de las consultas, como atributos de un inmueble definido de forma imprecisa, permite caracterizar la consulta como un objeto

que representa el inmueble que se está buscando. Al representar la consulta como un inmueble, el proceso de búsqueda se reduce a una comparación de objetos difusos, obteniendo los grados de semejanza entre el FO consulta y los FOs que representan los inmuebles almacenados en la base de datos.

9.2.3.2. Definición de datos y consulta

La implementación del ejemplo anterior se realizará empleando el lenguaje de consulta del prototipo de SGBDORD definido en el Capítulo 7.

Antes de comenzar con la definición, recordemos que el método estático *extends(typeName)* es utilizado por los tipos AFT para la creación de nuevos subtipos. Cuando este método es invocado, crea un subtipo (del tipo concreto sobre el que se ha invocado) con el nombre *<typeName>*. De la misma manera, hemos de recordar que el método estático *nearnessDef(memberList,degreeList)* define y almacena las relaciones de proximidad para los elementos miembros del dominio subyacente asociado al tipo. El primero de sus parámetros sirve para indicar los escalares que forman el dominio subyacente al dominio difuso en cuestión. El segundo parámetro es una lista de grados de proximidad entre los distintos elementos del dominio subyacente. El orden de estos grados está definido por el orden en que se declaran los escalares en el primer parámetro. Para el caso del listado que se muestra a continuación, primero se indican en la lista los grados de proximidad entre 'apartamento' y el resto de escalares, luego la de 'piso' con el resto excepto 'apartamento' (ya definida), y se continua con el mismo esquema hasta definir todas las posibles relaciones.

Según lo anterior, el tipo que representará a los inmuebles se define de la forma que se muestra en el siguiente listado.

```
-- Creación de subtipos de OAFT

exec OAFT.extends('Precio');
exec OAFT.extends('Superficie');
exec OAFT.extends('Habitaciones');

-- Creación del subtipo de NOAFT Clase

exec NOAFT.extends('Clase');


-- Definición de la relación de proximidad para el dominio Clase.

exec Clase.nearnessDef(
    ('apartamento','piso','casa','dúplex','ático'),
    (0.75,0.3,0.2,0.75,
          0.3,0.3,0.75,
              0.75,0.1,
                  0.1));
```

```
-- Definición del subtipo de FO Inmueble

create type Inmueble under FO(
    RClase Clase,
    RPrecio Precio,
    RSuperficie Superficie,
    RHabitaciones Habitaciones );
```

Tras definir todos los tipos necesarios para el presente ejemplo, crearemos una tabla denominada *Inmuebles_tab* para almacenar los datos relativos a los inmuebles que conformarán la base datos. La sentencia DDL para la creación de la misma es la que se muestra a continuación.

```
create table Inmuebles_tab of Inmueble;
```

Una vez disponible la tabla, las instancias de tipo *Inmueble* que representan a cada inmueble en la base de datos se insertan en ésta utilizando el método constructor por defecto del citado tipo y los métodos constructores propios de los tipos de sus atributos. Las sentencias DML para realizar las mencionadas inserciones se muestran en el siguiente listado:

```
-- Los constructores de tipos empleados se definen como sigue:
-- OAFT(valor) crea un valor preciso.
-- OAFT(a,b) crea un valor intervalo [a,b].
-- límites inferior y superior.
-- OAFT(a,b,c,d) crea un valor que es una distribución de
-- posibilidad trapezoidal.

insert into Inmuebles_tab values (
    Inmueble(
        Clase('piso'),
        Precio(155000),
        Superficie(75,90,90,105),
        Habitaciones(4)
        ));

insert into Inmuebles_tab values (
    Inmueble(
        Clase('apartamento'),
        Precio(120000),
        Superficie(100),
        Habitaciones(3)
        ));

insert into Inmuebles_tab values (
    Inmueble(
        Clase('ático'),
        Precio(170000),
        Superficie(70,80,80,90),
        Habitaciones(4)
        ));
```

```
insert into Inmuebles_tab values (
    Inmueble(
        Clase('casa'),
        Precio(166000),
        Superficie(122),
        Habitaciones(2)
        ));
```

Una vez almacenadas las instancias de los inmuebles en la base de datos, ya podemos aplicar consultas sobre ella. Según el mecanismo de consulta descrito anteriormente, la definición de cada consulta sobre esta base de datos incluirá el código necesario para la creación de un FO de la clase *Inmueble*. Éste representará el conjunto de condiciones que describen las características de los inmuebles que se desean obtener como resultado de la consulta. A modo de ejemplo, en el siguiente listado se muestra la sentencia SQL representando la consulta descrita la Figura 9.1.

```
SELECT (
    Inmueble(
        Clase('piso'),
        Precio(0,175000),
        Superficie(85,100,115),
        Habitaciones(3,4))
        ).feq(re1)
FROM Inmuebles_tab re1;
```

Como se indicó en el Capítulo 7, el cálculo del grado semejanza entre FOs, por defecto, se realiza mediante la agregación de los grados de semejanza entre cada par de valores de atributos, empleando para ello el operador de agregación V_Q detallado en [99]. En el presente ejemplo emplearemos este mismo procedimiento para comparar las instancias de la clase *Inmueble* que representan los inmuebles y la que representa la consulta.

Según lo anterior, por ejemplo, el grado de semejanza entre la consulta y el Inmueble 1 se calcula como se muestra en la Ecuación 9.1. En este cálculo se han establecido los parámetros del agregador V_Q a los valores $\gamma_Q = 0.2$ y $\mu_D(x) = 1$. Además se ha empleado la función mínimo como t-norma y la función máximo como t-conorma. Finalmente, la comparación entre valores difusos atómicos se ha realizado empleado una medida de posibilidad.

$$\begin{aligned} V_Q(A/D) = 0.2 \cdot max\left(min(1,1), min(1,1), min(1,0.67), min(1,1)\right) + \\ + (1-0.2) \cdot min\left(max(1,0), max(1,0), max(0.67,0), max(1,0)\right) = \\ = 0.2 \cdot 1 + 0.8 \cdot 0.67 = 0.736 \end{aligned} \quad (9.1)$$

La Tabla 9.3 muestra los grados de semejanza obtenidos para la consulta. Nótese que, aunque con ocasión del ejemplo este detalle se ha omitido por simplicidad, será posible aplicar en la consulta un umbral para evitar que aparezcan en el resultado final inmuebles con un grado de semejanza bajo.

Según los grados de semejanza obtenidos en el ejemplo, el inmueble 2 es el que mejor se adapta a la consulta. Obsérvese que, a pesar de que el citado

Inmueble	Grado de semejanza				
	Clase	*Precio*	*Superficie*	*Habitaciones*	V_Q
1	1	1	0.67	1	0.736
2	0.75	1	1	1	0.8
3	0.75	1	0.2	1	0.68
4	0.3	1	0.12	0	0.296

Tabla 9.3: Grados de semejanza calculados

inmueble es el más adecuado de los resultados según la consulta, éste no se ajusta exactamente a la misma.

Se ha de destacar que la consulta de este ejemplo no devolvería resultado alguno empleando un sistema *crisp*, ya que ninguno de los inmuebles presentes en la base de datos verifica de forma exacta la consulta. En el caso del presente ejemplo, si no existe un inmueble con las características que se han especificado en la consulta, se obtienen otras instancias con las características más similares disponibles. Por tanto, empleando un sistema difuso, se experimenta una mejora en el proceso de búsqueda en el sentido en que ésta se flexibiliza para dar cabida a ofertas similares a la demanda del cliente, tal y como haría de forma natural un agente inmobiliario. Esto hace de la búsqueda inmobiliaria difusa una herramienta muy útil, tanto para los clientes, como para los agentes de ventas.

9.2.3.3. InmoSoftWeb

ISW [8] es una aplicación web que implementa una extensión del esquema descrito para el ejemplo anterior. Al ser un sistema en producción permite la introducción de una mayor cantidad de datos sobre un inmueble, así cómo la expresión de condiciones flexibles más complejas. Adicionalmente, posee características de gestión de usuarios, anunciantes, agentes, así como capacidades de notificación automática.

Aunque, como se ha indica anteriormente, las capacidades de ISW son muy amplias, este sistema ofrece al usuario un modo simplificado que permite la definición de consultas sobre el conjunto de atributos que se ha empleado en el ejemplo anterior. Este sistema muestra al usuario un formulario similar al mostrado en la Figura 9.2. En dicha figura, se puede observar la definición de una consulta para la búsqueda de inmuebles que sean *pisos* (o inmuebles de un tipo *bastante* parecido al anterior), con una superficie *entre 85 y 100* metros cuadrados (permitiendo un *margen* de variación de unos *20* metros cuadrados), con un precio *hasta 60000* euros (permitiendo un *margen* de variación de unos *5000* euros) y *estrictamente* con un *mínimo* de dos habitaciones.

Los resultados ofrecidos por el sistema ISW a la consulta descrita anteriormente sobre una base de datos de ejemplo se muestran en la Figura 9.3. En dicha figura, se puede observar cómo los resultados han sido ordenados según su relevancia, medida como un porcentaje, y denominada *ajuste* en dicha figura. Este grado de relevancia es calculado en función al grado de similitud de los inmuebles en la base de datos con respecto a la consulta planteada, trasladándolo al rango [0,100].

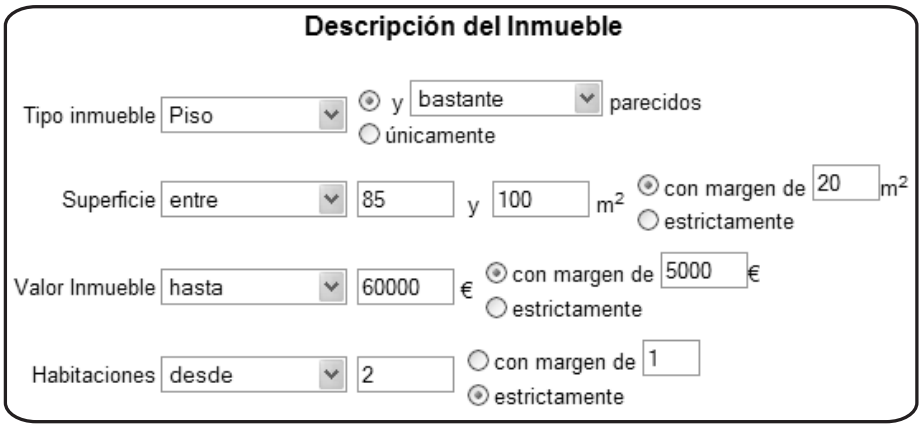

Figura 9.2: Interfaz web para la definición del consultas

Resultados :: Comprar

ID	Ajuste ▼▲	Tipo ▼▲	Precio € ▼▲	Superficie m² ▼▲
51	96%	Atico	109.000 €	109.0 ±9% m²
24	85%	Casa	97.000 €	120.0 - 150.0 ±9% m²
4	80%	Bajo	90.000 €	120.0 - 125.0 ±9% m²
9	78%	Apartamento	125.000 €	70.0 - 80.0 ±9% m²
67	65%	Piso	80.000 €	140.0 ±10% m²
60	64%	Piso	80.000 €	80.0 ±10% m²
41	58%	Atico	70.000 €	80.0 ±10% m²
48	39%	Apartamento	59.000 €	58.0 ±8% m²

Figura 9.3: Interfaz mostrando los resultados

9.3. Recuperación de imágenes en base a su color dominante

En la actualidad, se hace más común el acceso por usuarios no expertos a grandes colecciones de imágenes en entornos donde una respuesta rápida es fundamental (por ejemplo, sitios web). Esta clase de sistemas se caracterizan por el manejo de grandes cantidades de información, por el acceso concurrente de una gran cantidad de usuarios y por la necesidad de que éstos ofrezcan una resolución rápida de consultas. En nuestra opinión, existen tres cuestiones clave que son decisivas para el desarrollo de este tipo de sistemas: capacidad para la descripción natural y automática de las imágenes almacenadas, un mecanismo sencillo y natural para la definición de consultas y un alto rendimiento en la resolución de consultas. Con objeto de ofrecer una solución a las cuestiones anteriormente planteadas, proponemos el uso combinado de un método automático para la descripción de las imágenes junto con un SGBDORD.

Las aproximaciones iniciales a los sistemas de recuperación de imágenes se han basado en descripciones textuales de imágenes. Aunque éste es un mecanismo muy útil, exige que las citadas descripciones sean realizadas con intervención humana. Actualmente, los sistemas de recuperación de imágenes emplean como descriptores ciertas características de las imágenes extraídas de forma automática, como son el color, la textura o la forma. Usualmente, este último tipo de sistemas emplea imágenes de ejemplo o bocetos como base para la definición de consultas, métodos que pueden ser inapropiados para usuarios no expertos.

En la presente propuesta usaremos un método novedoso para la descripción automática de imágenes basada en sus colores dominantes [45, 11]. Este método aprovecha las capacidades de la teoría de conjuntos difusos para tratar con la vaguedad en las descripciones de los colores. Mediante la aplicación de este método, obtendremos una descripción de los colores dominantes en una imagen en forma de *colores difusos* a los que se les asigna un grado de dominancia. Esta aproximación difusa permite la definición de condiciones en las consultas en base a términos lingüísticos, más naturales para los usuarios no expertos.

Los descriptores difusos (y complejos, como veremos más adelante) de las imágenes serán representados en una base de datos aprovechando las capacidades del SGBDORD propuesto en el Capítulo 7 del presente trabajo. Además, este SGBDORD permitirá la ejecución de consultas sobre la base de datos con las que se podrán recuperar, de forma flexible y natural, imágenes en base a sus descriptores.

9.3.1. Descriptores en base a colores difusos dominantes

En términos generales, para asegurar un buen rendimiento del sistema de recuperación de imágenes propuesto, es necesario contar con métodos efectivos y eficientes para la descripción de las imágenes almacenadas. Como se comentó anteriormente, los sistemas actuales enfocan esta tarea empleando características de las imágenes como color, textura o forma [65]. En este contexto, los *colores dominantes* surgen como una potente herramienta para la descripción de los colores representativos de una imagen. La aproximación que se presenta a continuación tiene como propósito la descripción de imágenes mediante sus colores difusos dominantes.

Un color *color difuso* pretende ser una descripción natural de un color o, más precisamente, de una familia de colores que es interpretada e identificada por observadores humanos por un mismo nombre. Para su composición, se emplearán etiquetas lingüísticas que permiten su descripción de forma natural e intuitiva para los usuarios. En contraposición al concepto anterior, denominaremos *color crisp* a un color particular dentro de la gama cromática. En definitiva, un color *crisp* será un color concreto expresado numéricamente dentro de un espacio de color determinado y un color difuso se corresponderá con una distribución de posibilidad definida sobre el citado espacio de color.

El procedimiento para la extracción de colores difusos dominantes se enfoca en la presente propuesta en dos etapas. En primer lugar, se extraerá de cada imagen un conjunto de colores *crisp* dominantes. Posteriormente, el conjunto de colores dominantes obtenidos en la etapa anterior se emplea para obtener un conjunto de colores difusos que describan la imagen. Los siguientes apartados describen éstos procesos.

9.3.1.1. Colores dominantes

El presente apartado presenta una metodología para la extracción de colores *crisp* dominantes para imágenes empleando el espacio de color HSI (del inglés, *Hue-Saturation-Intensity* o, en español, *tono*, *saturación* e *intensidad*). En el espacio perceptual, la primera componente (denominada *tono* y notada como H) representa el tono del color, la segunda componente (notada como S y denominada *saturación*) la cantidad del color y la tercera componente (denominada intensidad y notada como I) la cantidad de luz. Obsérvese que, a pesar de que el espacio de color *Red-Green-Blue* (RGB) es el más empleado para la adquisición de imágenes, es ampliamente conocido que éste no es adecuado para el análisis del color de las imágenes. Particularmente, se ha de notar que los componentes de color en dicho espacio no tienen una interpretación intuitiva dentro de la percepción humana del color [65].

Extracción de colores dominantes *crisp*.

Puede encontrarse en la literatura una gran cantidad de aproximaciones para la extracción de colores dominantes, por ejemplo aquellos basados en el análisis de histograma o técnicas de agrupamiento. Para el presente caso, proponemos la aplicación de una aproximación basada en agrupamiento empleando el algoritmo de Batchelor & Wilkins [81], donde el número de grupos no ha de ser establecido a priori. Al inicio, este método parte de un solo grupo conteniendo todos los puntos de la imagen procesada. Posteriormente, se aplica un proceso iterativo que divide los grupos considerados hasta que un determinado criterio de parada es satisfecho. Dicho criterio de parada está basado en un parámetro $\theta \in [0, 1]$ relacionado con la distancia máxima tolerada para los elementos de un determinado grupo (este parámetro ha sido fijado empíricamente para el presente trabajo al valor $\theta = 0.3$). Para la cuantificación de la distancia entre elementos, proponemos el empleo de la distancia entre colores del espacio HSI [46]. Como resultado del proceso, obtendremos N grupos donde el centroide de cada uno de ellos, determinado como el valor medio, define un color dominante. En adelante, el conjunto de colores dominantes será notado como DCS. Este conjunto se define como se indica en la Ecuación 9.2, donde $dc_k = [h_k, s_k, i_k]$ es un color dominante representado en el espacio HSI.

$$DCS = \{dc_1, dc_2, \ldots, dc_N\} \tag{9.2}$$

Grado de dominancia.

Intuitivamente, podremos decir que un color es dominante en la medida en que éste aparezca frecuentemente en una determinada imagen. Parece natural, por tanto, modelar el conjunto de colores dominantes de una imagen como un conjunto difuso de colores, donde el grado de pertenencia de cada color al citado conjunto difuso indicará el grado de dominancia del mismo. Dichos grados de dominancia podrán ser calculados en base a la frecuencia de aparición de cada color en la imagen, aplicando ciertos umbrales para modelar la forma en que la percepción humana descarta los colores poco frecuentes y acepta como plenamente dominantes (en igual grado) a todos aquellos colores con una frecuencia de aparición muy alta. De esta forma, definiremos la función característica del conjunto difuso de colores dominantes de una imagen de la forma en que se muestra en la Ecuación 9.3. En dicha ecuación, $fr(c)$ representa la frecuencia de aparición del color c en la imagen en consideración, y u_1 y u_2 son dos parámetros, de la forma $0 \leq u_1 < u_2 \leq 1$, empleados para modelar los umbrales en la frecuencia anteriormente mencionados. Estos parámetros han sido fijados empíricamente, para la obtención de los resultados mostrados posteriormente, a los valores $u_1 = 0.05$ y $u_2 = 0.2$.

$$\mu_{dom}(c) = \begin{cases} 0 & fr(c) \leq u_1 \\ \frac{fr(c)-u_1}{u_2-u_1} & u_1 \leq fr(c) \leq u_2 \\ 1 & fr(c) \geq u_2 \end{cases} \tag{9.3}$$

9.3.1.2. Colores difusos dominantes

En el presente apartado se detalla el proceso para la obtención del conjunto de colores difusos dominantes de una imagen a partir del conjunto de colores *crisp* dominantes de ésta.

En la presente aproximación, los colores difusos serán los pertenecientes al espacio de color difuso HSI propuesto en [45, 44]. En dicho espacio, un color HSI difuso $\widetilde{C}$ está definido como una etiqueta lingüística que identifica a un subconjunto difuso del espacio de color HSI $[0, 2\pi] \times [0, 1] \times \{0, \ldots, 255\}$. De esta forma, el espacio de color HSI difuso, notado como $\widetilde{HSI}$, se define como un conjunto de colores HSI difusos que definen una partición del citado espacio de color HSI.

Con objeto de definir y representar un espacio de color HSI difuso, en [45] se propone el uso de subespacios de tono, intensidad y saturación difusos que están particionados, a su vez, por una serie de conjuntos difusos identificados por unas etiquetas lingüísticas que representa diversos tonos, saturaciones e intensidades difusas. En la Figura 9.4 se ilustra dicho particionado.

Según lo anterior, un color HSI difuso $\widetilde{C}$ puede ser definido y representado en la práctica como la tripleta $[\widetilde{C}_H, \widetilde{C}_S, \widetilde{C}_I]$, donde $\widetilde{C}_H$, $\widetilde{C}_S$ y $\widetilde{C}_I$ se corresponden con el tono difuso, saturación difusa e intensidad difusa de $\widetilde{C}$, respectivamente. Remitimos al lector a [45] para más detalles sobre el citado espacio de color difuso.

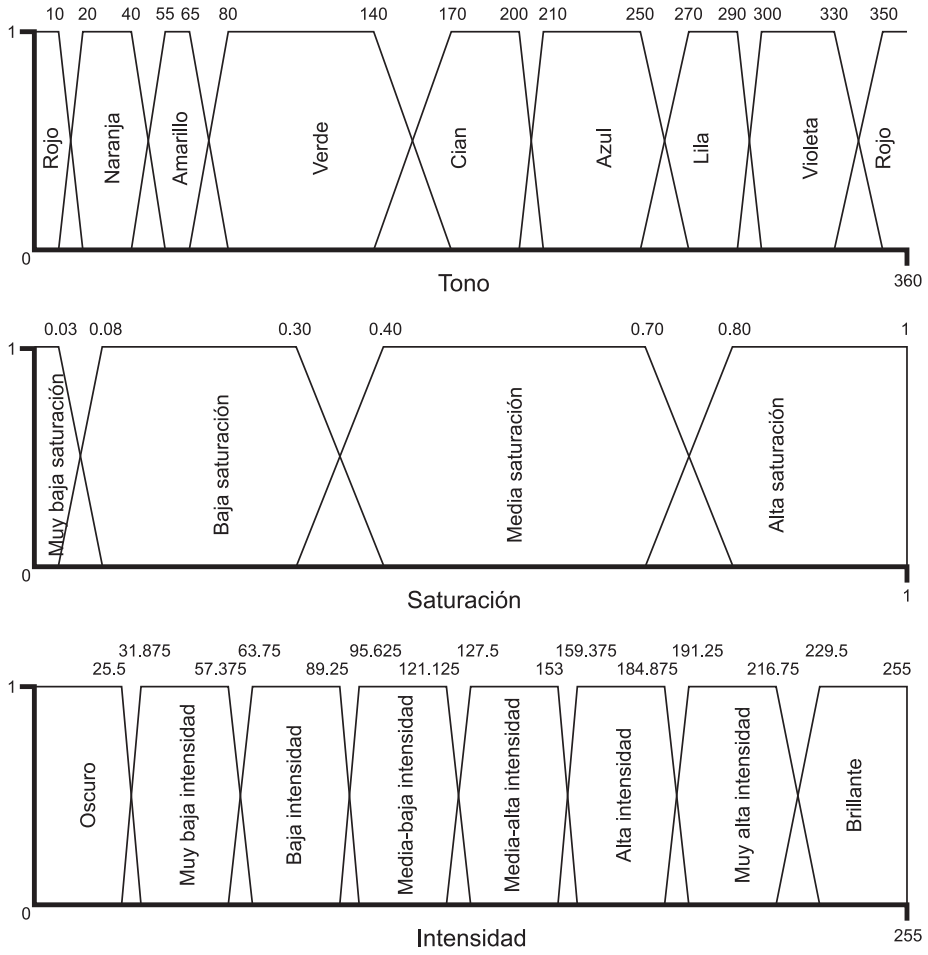

Figura 9.4: Espacio de color HSI difuso

En base al espacio de color HSI difuso, introduciremos el concepto de color difuso dominante en una imagen como se indica a continuación.

Definición 9.1 (Color difuso dominante)**.** *Un color dominante es un color HSI difuso que aparece con frecuencia en la imagen.*

Como el caso de los colores *crisp* dominantes, la anterior definición es imprecisa por naturaleza, ya que *dominante* y, por extensión, *frecuente* son términos imprecisos.

Se pueden emplear muchas aproximaciones para el cálculo de la dominancia de un color difuso en una imagen. La alternativa por la que optamos en la presente propuesta es la obtención del conjunto de colores difusos dominantes desde el conjunto de colores *crisp* dominantes. Consideraremos que un color difuso es dominante en la medida en que éste incluya a un color *crisp* dominante. Este planteamiento lleva a la siguiente definición.

Definición 9.2 (Conjunto de colores difusos dominantes)**.** *Sea el conjunto* $DCS = \{dc_1, \ldots, dc_N\}$ *un conjunto de colores* crisp *dominantes de una imagen, donde* $dc_k = [h_k, s_k, i_k]$.

El subconjunto difuso de colores difusos dominantes de dicha imagen está definido como se indica en la Ecuación 9.4. En dicha ecuación, $\widetilde{DCS}_k$ *se define como se indica en la Ecuación 9.5.*

$$\widetilde{DCS} = \bigcup_{k \in \{1,\ldots,N\}} \widetilde{DCS}_k \tag{9.4}$$

$$\widetilde{DCS}_k = \left\{ \left(\mu_{\widetilde{C}}(dc_k) \otimes \mu_{Dom}(dc_k) \right) / \widetilde{C} | \widetilde{C} \in \widetilde{HSI} \right\} \tag{9.5}$$

Según lo anterior, por cada color *crisp* dominante dc_k, obtendremos un conjunto difuso de colores difusos dominantes determinado como se indica en la Ecuación 9.5. En dicho conjunto difuso, el grado de pertenencia de los colores difusos indica su grado de dominancia. Éste grado, como se indica en la Ecuación 9.5, se calcula como una conjunción entre el grado de dominancia del color *crisp* y el grado de compatibilidad de éste con el color difuso respectivo. Si un determinado color difuso resulta compatible con varios colores *crisp* dominantes, se obtendrán varios grados de dominancia para éste. No obstante, dichos grados serán agregados mediante una t-conorma según se indica en la Ecuación 9.4.

9.3.2. Modelado de los descriptores flexibles de colores dominantes

Como se ha descrito anteriormente, el método propuesto para la descripción automática de imágenes genera, como resultado, un conjunto de colores dominantes por cada imagen. En un sistema de recuperación de imágenes implementado sobre un SGBD, el uso de un tipo definido por el usuario que permita representar de forma compacta dicho descriptor, facilitará el almacenamiento y manejo de dichos descriptores así como la definición y el procesamiento de consultas sobre los mismos. Los tipos de datos del SGBDORD propuesto en el Capítulo 7 proporcionan una buena base para la definición del tipo de datos

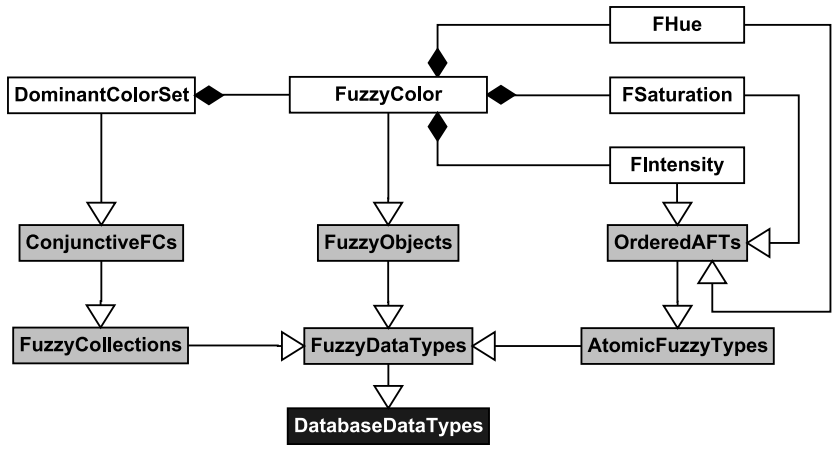

Figura 9.5: Diagrama UML del tipo de datos *DominantColorSet*

que permitirán representar los citados descriptores, ya que éstos incluyen información difusa sobre los colores y grados de dominancia. Además de lo anterior, los operadores para la definición de consultas sobre datos difusos, disponibles en dicho SGBDORD, permitirán la definición de condiciones flexibles sobre los descriptores de las imágenes.

Definiremos un tipo de datos, denominado *DominantColorSet*, para representar en la base de datos los descriptores de colores difusos dominantes propuestos. Este tipo de datos está modelado como se indica en la Figura 9.5. Las clases representadas en la citada figura siguen un código de color en el que los tipos de datos provistos por el propio SGBDORD son representados con un color de fondo gris, los tipos de datos básicos del SGBD objeto-relacional (sobre el que el SGBDORD está construido) se representan con un fondo negro y el resto de tipos definidos de forma auxiliar para definir al tipo *DominantColorSet* se representan con un fondo blanco.

Describiremos a continuación la definición del tipo *DominantColorSet* empleando una aproximación ascendente. Los componentes básicos de un color difuso son las etiquetas lingüísticas representando un tono, saturación e intensidad difusos. Estas etiquetas lingüísticas se corresponden con distribuciones de posibilidad trapezoidales definidas sobre los dominios *crisp* de tono, saturación e intensidad del espacio HSI, tal y como se ha mostrado anteriormente en la Figura 9.4. Dado que todos los dominios subyacentes referidos anteriormente son ordenados, podemos definir un tipo derivado de OAFT por cada uno de estos componentes. De esta forma, declaramos los tipos de datos *FHue*, *FSaturation* y *FIntensity* que servirán para representar a los componentes $\widetilde{H}$, $\widetilde{S}$ e $\widetilde{I}$ de un color difuso, respectivamente. Además de lo anterior, téngase en cuenta que las etiquetas lingüísticas presentes en la Figura 9.4 se definirán para el tipo correspondiente.

Según las definiciones anteriores, un color difuso se representa como una tripleta $[\widetilde{H}, \widetilde{S}, \widetilde{I}]$. Dado que sus componentes se pueden representar empleando etiquetas lingüísticas, un color difuso puede ser visto como una agrupación de tres de estas etiquetas. Un color difuso se presentará en la base de datos como una instancia del tipo *FuzzyColor*. Éste es un subtipo de *FO* y está compuesto

por tres atributos del tipo *FHue*, *FSaturation* y *FIntensity*, respectivamente.

Finalmente, podremos definir, en base a los tipos anteriores, el tipo de datos *DominantColorSet*. Los elementos de este tipo de datos han de representar un conjunto difuso de colores difusos, notado anteriormente como $\widetilde{DCS}$. Como se indicó con anterioridad, el grado de pertenencia de cada color difuso a dicho conjunto representará su grado de dominancia. Dado que $\widetilde{DCS}$ es un conjunto difuso de elementos complejos, que a su vez son difusos, el tipo *DominantColorSet* debe derivarse de FC. En este caso, teniendo en cuenta la semántica conjuntiva de $\widetilde{DCS}$, este tipo será definido como un derivado de CFC, cuyos elementos serán del tipo *FuzzyColor*.

9.3.3. Operadores para la recuperación de imágenes

La presente subsección describe los operadores, aplicados sobre los elementos de los tipos de datos difusos introducidos en la anterior subsección, más significativos en la definición de consultas tipo para la recuperación de imágenes.

9.3.3.1. Operador de inclusión difusa

El operador *FInclusion(A,B)* devuelve el grado en que la relación $A \subseteq B$ se cumple, siendo A y B instancias de un tipo de datos derivado de CFC. En el capítulo 7 se propone el empleado del *Grado de Inclusión Guiado por Semejanza* [99] como método para determinar el citado grado. No obstante, para el ámbito de la aplicación que nos ocupa, éste método no resulta totalmente apropiado. Concretamente, el citado método no permite la inclusión parcial, lo que significa que no se obtendrá un grado de inclusión superior a cero a pesar de que los elementos no incluidos en B sean los menos importantes de A (interpretado el grado de pertenencia a las colecciones como grado de importancia).

Para dotar de la máxima flexibilidad al sistema de recuperación de imágenes propuesto, la inclusión parcial resulta fundamental. Esto implica, de forma práctica, que se tenga en cuenta el grado de dominancia de cada uno de los colores difusos presentes en la imagen y que se actúe, de forma ponderada, en consecuencia. Por esta razón, proponemos emplear, para el caso que nos ocupa, una modificación del *Grado de Inclusión Guiado por Semejanza* que permita tal comportamiento. Inspirados por [74], proponemos una modificación de la Ecuación 7.13 en la Definición 7.5 que implica la sustitución de la agregación por medio de la función mínimo por una agregación de media ponderada, donde los pesos de tal agregación se corresponden con el grado de participación de cada uno de los elementos de A. Formalmente, este método para determinar el grado de inclusión de dos colecciones difusas, que denominaremos *Grado de Inclusión Parcial Guiado por Semejanza*, se define como se indica a continuación.

Definición 9.3 (Grado de inclusión parcial guiado por semejanza)**.** *Sean A y B dos conjuntos difusos definidos, sobre el universo finito de referencia U, por sus funciones características $\mu_A(x)$ y $\mu_B(y)$. Sea S una relación de semejanza definida sobre los elementos del dominio y $\mu_S(x,y)$ su función característica.*

El grado de inclusión parcial guiado por semejanza de A en B se define como se indica en la Ecuación 9.6. En dicha ecuación, $\theta_{A,B,S}(x,y)$ es una función definida como se indica en la Ecuación 7.14.

$$\Theta^*_S(B|A) = \sum_{x \in U} \frac{\mu_A(x)}{|A|} \cdot \max_{y \in U} \theta_{A,B,S}(x,y) \tag{9.6}$$

9.3.3.2. Operador de igualdad difusa

El operador de igualdad difusa, *FEQ(A,B)*, determina el grado de semejanza entre dos valores A y B de un determinado tipo difuso. La forma en que este cálculo se realiza depende del tipo difuso de sus operandos. En el presente apartado introduciremos los mecanismos de cálculo de igualdad difusa que emplearemos en el ámbito del sistema de recuperación de imágenes que nos ocupa.

Operador de igualdad difusa para colecciones difusas conjuntivas.

Si A y B son dos instancias de un tipo derivado de CFC, el grado de semejanza entre éstas se calcula, como se indicó en el Capítulo 7, empleando el Grado de Semejanza Generalizado entre Conjuntos Difusos. Téngase en cuenta que el citado método, que aplica el concepto de doble inclusión, está basado en el *Grado de Inclusión Guiado por Semejanza*. Dado que anteriormente se ha propuesto una modificación para el cálculo del grado de inclusión, el Grado de Inclusión Parcial Guiado por Semejanza, parece lógico que en la determinación del grado de semejanza entre dos CFCs se emplee dicho método. De esta forma, el operador de igualdad difusa resultará más flexible (en lo que se refiere a inclusiones parciales) y, por tanto, el sistema de recuperación se beneficiará de tal flexibilidad.

A pesar del grado de flexibilidad introducido al permitir la inclusión parcial, el operador de igualdad difusa para CFCs no resulta lo suficientemente flexible para los objetivos del sistema de recuperación de imágenes. Tal y como está concebido el Grado de Semejanza Generalizado entre Conjuntos Difusos, se penalizan los casos en que existe una inclusión asimétrica entre los operandos. Esta penalización proviene de la agregación de los grados de inclusión mediante el uso de una t-norma. En el presente caso se propone la agregación de estos grados mediante una media aritmética, lo que permitirá una mayor flexibilidad en las comparaciones.

Operador de igualdad difusa para instancias de colores difusos.

En el caso de este operador, en el ámbito de la presente aplicación, el método por defecto para la comparación de FOs no resulta apropiado, ya que se requiere cierta flexibilidad adicional.

En el marco de la aplicación propuesta, cuando el operador *FEQ(A,B)* es aplicado sobre instancias de un FO, en este caso sobre instancias de la clase *FuzzyColor* (que representan a un color difuso), el grado de similitud entre instancias se calcula según el siguiente método.

Definición 9.4 (Grado de semejanza proporcional)**.** *Sean o_1 y o_2 dos objetos de la clase C, $att(C)$ el conjunto de atributos de los objetos de clase C y $FEQ(a,b)$ el operador de semejanza entre dos instancias a y b de una misa clase. El grado de semejanza entre los objetos o_1 y o_2 se determina según se indica en la Ecuación 9.7.*

$$OR(o_1, o_2) = \frac{1}{|att(C)|} \sum_{a \in att(C)} FEQ(o1.a, o2.a) \tag{9.7}$$

Obsérvese que el método anterior no busca una ponderación entre el cuantificador existencial y el universal, como es el caso de la propuesta empleada en el Capítulo 7. Los anteriores cuantificadores son extremos y no ofrecen cierta graduación entre los mismos. En el caso del grado de semejanza proporcional, se tiene en cuenta el número de atributos cuyos valores resultan compatibles. De esta forma, se obtiene un operador de comparación con un mayor grado de flexibilidad.

9.3.4. Recuperación de imágenes en base a criterios de colores dominantes

El anterior conjunto de tipos de datos y operadores habilitan al SGBDORD para el procesamiento de consultas que incluyan condiciones definidas sobre el conjunto de colores dominantes que describe cada imagen en la base de datos.

Según la aproximación que hemos elegido, una condición basada en colores dominantes es, en definitiva, una condición flexible sobre el conjunto difuso de colores difusos (siendo éstos, a su vez, etiquetas lingüísticas que representan conjuntos difusos). Los operadores de inclusión y semejanza difusa definidos anteriormente, permitirán la definición de consultas para recuperar aquellas imágenes que incluyan entre sus colores dominantes un determinado patrón de color o la búsqueda de aquellas imágenes con descriptores de colores dominantes similares. Todo ello con la flexibilidad que introduce tanto la representación difusa de los colores dominantes como los operadores flexibles definidos anteriormente para este propósito.

Las consultas mencionadas podrán definirse de forma natural, ya que los usuarios podrán especificar los colores difusos que participarán en ellas en base a etiquetas lingüísticas que definirán cada uno de los tres componentes del color. Esta forma natural de definición del color supondrán una ventaja sobre la forma tradicional en que se definen los colores de forma numérica. Además de lo anterior, este mecanismo de definición de los colores permitirá la definición de diferentes condiciones sobre cada uno de los componentes del color, así como la omisión de requisitos sobre una determinada componente. Esta última posibilidad se consigue empleando valores ausentes como valor para las componentes omitidas en la especificación del color difuso empleado como condición.

En los siguientes apartados mostraremos algunos ejemplos de consultas sobre el sistema de recuperación de imágenes descrito, basados en los operadores de inclusión y semejanza definidos anteriormente, y los resultados obtenidos sobre bases de datos de prueba.

9.3.4.1. Consultas basadas en la inclusión de colores dominantes

Supongamos la siguiente petición para el sistema: "Recuperar todas aquellas imágenes que incluyan entre sus colores dominantes el color difuso *rojo, muy saturado, brillante*". Éste es un ejemplo de consulta con una sola condición que emplea el operador de inclusión. Dicha condición se construirá utilizando como primer operado el conjunto difuso que describe el conjunto de colores dominantes

Figura 9.6: Resultados de consultas basadas en inclusión de colores dominantes

de cada imagen en el sistema (siendo éste el atributo *ColorDescriptor* para el presente ejemplo). Como segundo operando, se empleará el conjunto de colores que han de incluir las imágenes seleccionadas entre sus colores dominantes (en este caso, este conjunto está compuesto por un solo elemento, el color difuso [*Rojo*,*Alta saturación*,*Brillante*]). Esta consulta se expresa en SQL, empleando los tipos y operadores definidos anteriormente, de la forma siguiente:

```
SELECT image,cdeg(1) FROM images WHERE
    FCond( FInclusion(
        ColorDescriptor,
        DominantColorSet(
            1.0,
            FuzzyColor(
                FHue('Rojo'),
                FSaturation('Alta saturación'),
                Intensity('Brillante')
    ) ) ) , 1 ) > 0 ORDER BY 2 DESC;
```

La Figura 9.6 muestran en la primera fila los resultados devueltos por la anterior consulta al aplicarla sobre una base de datos que contiene 160 imágenes de banderas. En dicha figura, la primera columna muestra los colores difusos que han de incluir como colores dominantes las imágenes que compongan el resultado. Por cada uno de estos colores difusos, se incluye como referencia una muestra de un color *crisp* completamente compatible con éstos. En la segunda columna se muestran los resultados de la consulta ordenados por relevancia. Además de la citada consulta, en la segunda fila de la citada figura se pueden ver los resultados para una consulta algo más compleja que combina varios criterios.

Otro ejemplo de petición al sistema de recuperación podría ser el siguiente: "Recuperar todas aquellas imágenes que incluyan colores brillantes entre sus colores dominantes". En este caso, se empleará el operador de inclusión para buscar todas aquellas imágenes que contengan el color difuso [*unk*,*unk*,*Brillante*], donde la etiqueta *unk* se corresponde con el valor ausente de tipo desconocido para los dominios del tono y saturación difusos. Nótese que el uso de la etiqueta *unk* evita la definición de condiciones sobre las componentes de tono y saturación del color difuso definido. La consulta anterior, aplicada sobre una base de datos compuesta por 700 imágenes, devuelve los resultados mostrados en la Figura 9.7.

Figura 9.7: Resultados de una consulta de inclusión de colores dominantes empleando sólo la componente de intensidad

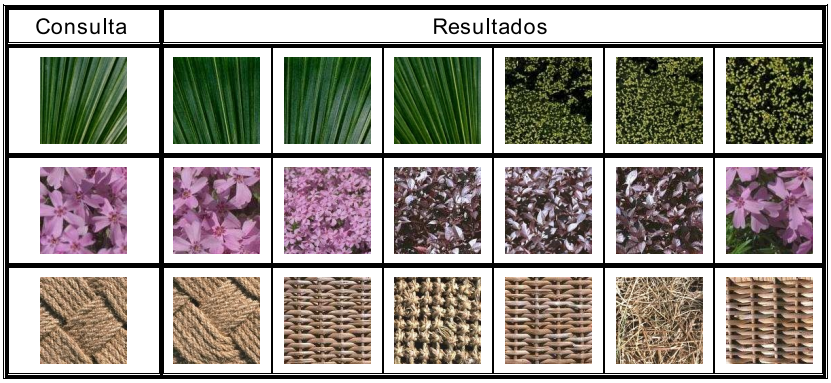

Figura 9.8: Resultados de una consulta basada en la similitud de colores dominantes de una imagen dada

9.3.4.2. Consultas basadas en la similitud de los conjuntos de colores dominantes

Otro tipo interesante de consultas, en este tipo de sistemas, son las que incluyen una condición para la recuperación de las imágenes con un patrón de colores dominantes similar al asociado a una imagen de referencia. Este tipo de condiciones se definen empleado el operador de igualdad difusa para tipos derivados de CFC, el cual determinará la similitud entre los conjuntos difusos de colores dominantes de las imágenes en la base de datos con respecto a los presentes en la imagen referencia. Un ejemplo de consulta de este tipo se expresa en SQL de la forma siguiente:

```
SELECT a.image,cdeg(1) FROM images a, images b WHERE b.id=? AND
    FCond( FEQ(
        a.ColorDescriptor,
        b.ColorDescriptor ),
    1 ) > 0 ORDER BY 2 DESC;
```

Obsérvese que, en la anterior consulta, el carácter *?* se ha empleado como parámetro de sustitución para el identificador de la imagen cuyo patrón de colores dominantes se empleará como condición en la búsqueda.

En la Figura 9.8 se muestran diversos ejemplos de consultas de este tipo y los resultados obtenidos tras su aplicación sobre la base de datos de 700 imágenes citada anteriormente. En dicha figura, se muestra la imagen empleada como referencia en la primera columna, y el resto de columnas muestran los resultados obtenidos ordenados según su relevancia.

9.4. Recuperación de radiografías mediante descriptores de forma

Una aplicación de recuperación, similar a la presentada en la sección anterior, es la recuperación de imágenes médicas en base a sus descriptores [106].

Las imágenes son herramientas fundamentales en el campo de medicina para la diagnosis, estudios clínicos, investigación y docencia. En la actualidad, existen diversas técnicas para la obtención de imágenes de pacientes para apoyar el diagnóstico, como son las radiografías y diversos tipos de tomografías. En esta última categoría se pueden destacar las tomografías obtenidas mediante rayos X y la asistencia de una computadora (denominadas TC), las resonancias magnéticas (denominadas RMI), las tomografías por emisión de positrones (PET) y las ultrasonografías. Las tareas de diagnóstico empleando estas técnicas generan una gran cantidad de imágenes que han de ser archivadas para futuras evaluaciones. Afortunadamente, muchas de estas técnicas producen imágenes digitales, que pueden ser archivadas y manejadas empleando sistemas informáticos de forma más eficiente que mediante sistemas físicos. Se conoce a este tipo de sistemas como PACS (del inglés *Picture Archiving and Communication Systems*, sistema de archivado y transmisión de imágenes). Los PACS son sistemas en red dedicados al almacenamiento, recuperación, distribución y representación de imágenes. Aunque este tipo de sistemas resuelve el problema del almacenamiento de la imágenes, éstos no proporcionan mecanismos para recuperar las mismas en base a su contenido.

Como se indicó en la sección anterior, la situación ideal en sistemas para la recuperación de imágenes en base a su contenido, o en su denominación anglosajona CBIR (*Content Based Image Retrieval*), es que éstos tengan la suficiente capacidad para extraer, de forma automática y sin intervención humana alguna, descriptores del contenido de las imágenes que archivan. Como se demostró en la citada sección, en este tipo de sistemas las aproximaciones empleando lógica difusa ofrecen ventajas en lo que se refiere a flexibilidad de los sistemas automáticos de descripción de imágenes y naturalidad en los sistemas de consulta.

La capacidad de este tipo de sistemas comienza a ser significativa cuando éstos se diseñan para su aplicación en un dominio o área específicos, ya que un conocimiento previo sobre dicho dominio (así como sobre el contenido y estructura de las imágenes) ayuda a la definición y extracción de características relevantes de las imágenes procesadas. Concretamente, la medicina es un área que puede beneficiarse en mayor medida de éstas técnicas. Si enfocamos el desarrollo de un CBIR para el análisis de una determinada patología, quedará claramente definido el tipo de imágenes que trataremos, así como el tipo de características de alto nivel que han de extraerse de las mismas. Existen propuestas para extraer formas vertebrales y espinales de radiografías como, por ejemplo, las que se proponen en [151, 14], aunque sus algoritmos requieren cierta intervención. En [173] se propone una técnica para medir automáticamente el ángulo de Cobb [47], medida que se emplea para evaluar la severidad y evolución de la escoliosis (patología en la que centraremos la propuesta y que describiremos más adelante).

La presente propuesta tiene como objetivo demostrar la utilidad de los SGBDORD en la tarea de representación y recuperación de imágenes médicas a partir de ciertos descriptores difusos éstas. Para ilustrar tal propuesta, se presentará

un ejemplo en el que se recuperarán radiografías anteroposteriores de columnas vertebrales descritas por los ángulos de Cobb medidos sobre ellas.

9.4.1. Escoliosis idiopática y su diagnóstico

Con el objeto de mostrar la utilidad del prototipo de SGBDORD propuesto en el presente trabajo en el área médica, hemos centrado la presente propuesta en el estudio de la representación y recuperación de imágenes médicas empleadas para la valoración de la escoliosis idiopática. En la presente subsección, se describirán brevemente las características más relevantes, en relación con la presente propuesta, de esta patología.

La escoliosis es una deformación tridimensional de la columna vertebral que produce rotación y aplastamiento vertebral, así como curvatura lateral. Se clasifica típicamente como congénita (causada por anomalías en las vértebras desde el nacimiento), idiopática (subclasificada como infantil, juvenil, adolescente o adulta, en función del momento en que se produce) o secundaria de alguna otra condición primaria que la provoca (como, por ejemplo, parálisis cerebral, atrofia muscular espinal o traumas físicos). En función de la severidad y progresión de la deformación, puede resultar necesario tratamiento consistente en la observación continuada, tratamiento ortopédico o cirugía. Aproximadamente del 2 % al 4 % de la población adolescente sufre en algún grado esta patología. Aproximadamente el 2.2 % de estos casos requiere tratamiento.

Para el diagnóstico y tratamiento de la escoliosis es necesario realizar medidas sobre la deformación de la columna vertebral. Aunque mediante un examen físico se puede detectar inicialmente la presencia de tales deformidades, es necesario emplear técnicas radiológicas para realizar un diagnóstico y tratamiento preciso. La técnica más precisa para medir la deformidad es la TC, ya que ésta proporciona una vista tridimensional de la espina dorsal. No obstante, esta técnica resulta onerosa y expone al paciente a grandes dosis de radiación. Teniendo en cuenta que un paciente que sufre escoliosis necesita observación frecuente y tratamiento durante un periodo prolongado, que puede alargarse muchos años, el uso frecuente de esta técnica puede resultar inapropiado. El empleo de radiografías, en lugar de TCs, significa una exposición mucho menor del paciente a radiación. Es por ello que la radiografía anteroposterior (AP) es el método estándar para la evaluación de la severidad y progresión de la escoliosis.

Las radiografías AP proyectan las deformidades de la columna como curvas. El método establecido para la medición de esta curvatura en una radiografía AP es el *Ángulo de Cobb* [47]. El Ángulo de Cobb puede ser medido de forma manual calculando el ángulo entre las líneas dibujadas, respectivamente, a lo largo del borde superior del cuerpo de la vértebra superior que forma parte de la curva y del borde inferior del cuerpo de la vértebra inferior que forma parte de dicha curva, tal como se muestra en la Figura 9.9. Empleando esta medida, cada curva presente en la columna vertebral es caracterizada por cuatro parámetros:

1. El lado de la convexidad de la curva (izquierda o derecha).
2. La vértebra superior que forma parte de la curva.
3. La vértebra inferior que forma parte de la curva.
4. El ángulo de Cobb de la curva.

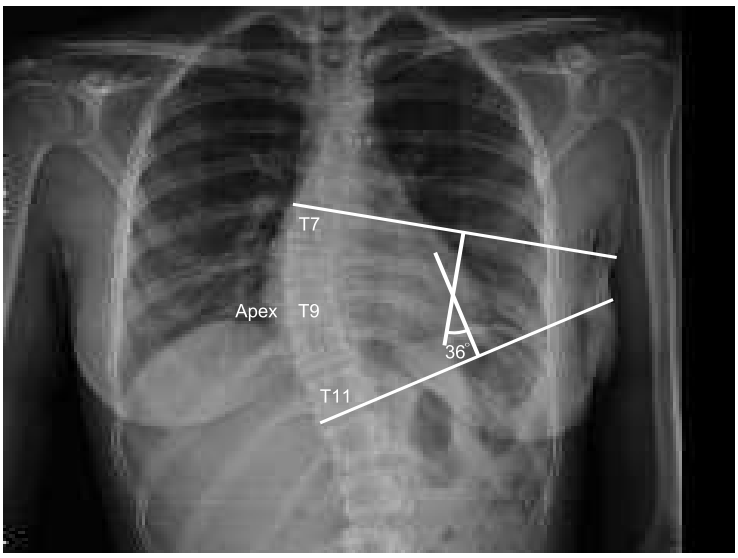

Figura 9.9: Medición del ángulo de Cobb

Además de los parámetros anteriores, resulta importante identificar la vértebra *apex*, vértebra más desviada del eje central de la columna vertebral del paciente. Esta vértebra está normalmente situada en el punto medio de la curva (vértebra T9 en la Figura 9.9).

La medición manual del ángulo de Cobb depende de la experiencia y del juicio personal del especialista. Los errores se producen por la selección de diferentes vértebras finales y por la estimación de diferentes inclinaciones de éstas. Los errores estándar en la medida se establecen entre tres y cinco grados para el mismo observador y de cinco a siete grados para distintos observadores.

Las radiografías AP resultan útiles para el diagnóstico, estudios clínicos y enseñanza de la medicina. Para ayudar en tales propósitos, resulta interesante implementar un sistema para el almacenamiento y recuperación de estas radiografías en base a los parámetros que las caracterizan. Téngase en cuenta que, como se ha mencionado anteriormente, los valores de tales parámetros están afectados por cierta imprecisión y, por tanto, los parámetros de consultas sobre los mismos han de ser flexibles. Por esta razón, los SGBD clásicos no resultan apropiados para este propósito, ya que se requiere un SGBD capaz de almacenar datos imprecisos y procesar consultas flexibles sobre los mismos. Además de lo anterior, se ha de tener en cuenta que la caracterización de las curvas resulta en un valor compuesto, por lo que será necesario un sistema capaz de manejar estructuras complejas y que provea comparadores flexibles sobre éstas. Por estas razones, proponemos el uso de un SGBDORD para esta tarea.

9.4.2. Almacenamiento y recuperación de radiografías en base a sus descriptores difusos

En la presente subsección se describirá la base de datos que almacenará las radiografías y sus descriptores, así como la forma en que se definirán consultas sobre las mismas para su posterior recuperación.

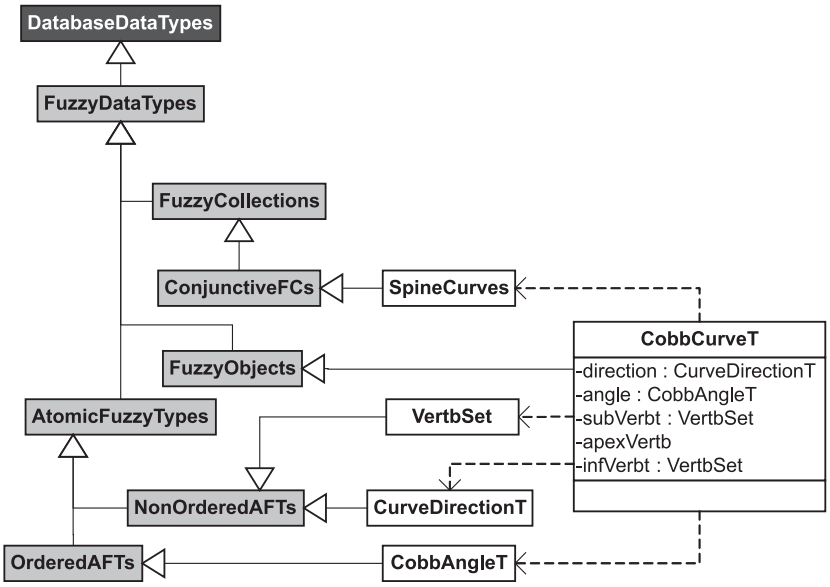

Figura 9.10: Diagrama UML del tipo de datos empleados para el modelado de los descriptores de curvatura en radiografías

9.4.2.1. Base de datos para radiografías y descriptores

El SGBDORD descrito en el presente trabajo es capaz de representar los descriptores propuestos y permite captar la impresión originada por posibles errores en las mediciones.

En la Figura 9.10 se muestra un diagrama UML simplificado que ilustra el modelado de los tipos definidos para la representación de los descriptores de las radiografías. Las clases mostradas en dicha figura siguen el mismo código de color que el empleado en la Figura 9.5.

En la figura anterior, la clase *SpineCurves*, derivada de la clase CFC, es la empleada para modelar la colección de curvas registradas en una determinada columna vertebral. Cada curva se describe por una instancia de la clase *CobbCurveT*. Esta clase, derivada de FO, incluye los siguientes cinco atributos:

1. *Direction*: Almacena el lado de la convexidad de la curva.
2. *Angle*: Empleado para almacenar el grado de curvatura de la curva.
3. *SupVertb*: El valor de este atributo indica cuál es la vértebra superior de la curva.
4. *ApexVertb*: Este atributo es empleado para indicar la vértebra apex de la curva.
5. *InfVertb*: En este atributo se indica cuál es la vértebra inferior de la curva.

A pesar de contener un valor claramente preciso, el atributo *Direction* se ha definido como un tipo NOAFT para poder omitir éste (en realidad, asociar al atributo en la definición del objeto consulta un valor ausente de tipo desconocido) en las definiciones de consultas cuando sea necesario. Los atributos *SupVerb*,

ApexVertb e *InfVertb* son de un mismo tipo, denominado *vertbSetT*. Este tipo, es un tipo NOAFT cuyo dominio subyacente contiene como elementos a las 24 vértebras denominadas ascendentemente, comenzando por la parte inferior de la columna vertebral, como $L5$, $L4$, $L3$, $L2$, $L1$, $T12$, $T11$, $T10$, $T9$, $T8$, $T7$, $T6$, $T5$, $T4$, $T3$, $T2$, $T1$, $C7$, $C6$, $C5$, $C4$, $C3$, $C2$, $C1$. La relación de similitud entre los elementos del dominio anterior se define de la forma que se indica en la Ecuación 9.8. En dicha ecuación, v_i y v_j representan, respectivamente, al i-ésimo y j-ésimo elemento del dominio subyacente de *vertbSetT* siguiendo el orden en que éstas han sido enumeradas anteriormente.

$$s(v_i, v_j) = \begin{cases} 1 & \text{si } |i-j| \leq 1 \\ 0.66 & \text{si } |i-j| = 2 \\ 0.33 & \text{si } |i-j| = 3 \\ 0 & \text{en otro caso} \end{cases} \qquad \forall i, j \in [1, 24] \tag{9.8}$$

El atributo *Angle* es de tipo OAFT con el objeto de que se puedan almacenar en él valores numéricos, ya sean precisos o imprecisos.

Operadores sobre los tipos difusos

Dadas las características particulares de los descriptores de curvatura, las instancias de la clase *CobbCurvT* se compararán según una medida de semejanza propia. Ésta se define como se indica a continuación.

Definición 9.5 (Grado de semejanza parametrizado)**.** *Sean o_1 and o_2 dos objetos de la clase C, $o.a_i$ el valor del i-ésimo atributo del objeto o, $rel(a_i)$ el grado de relevancia del i-ésimo atributo de los objetos de la clase C, n el número de atributos definidos en la citada clase C, m el número mínimo de atributos cuyo grado de semejanza ha de ser superior a 0, y FEQ el operador que devuelve el grado de semejanza entre dos instancias de una misma clase.*

Definiremos el grado de semejanza parametrizado *como se indica en la Ecuación 9.9.*

$$OR(o_1, o_2) = \begin{cases} 0 & \text{si } \exists a \in att(C) \mid \\ & \quad (FEQ(o1.a, o2.a) = 0 \wedge \\ & \quad\ rel(a) = -1) \\ 0 & \text{si } |\{a \mid a \in att(C) \ \wedge \\ & \quad\ FEQ(o1.a, o2.a) > 0\}| < m \\ \dfrac{\sum_{a \in att(C)} FEQ(o1.a, o2.a) \cdot |rel(a)|}{\sum_{a \in att(C)} |rel(a)|} & \text{en otro caso} \end{cases} \tag{9.9}$$

En lo que respecta a los operadores sobre CFCs, se emplearán el Grado de inclusión parcial guiado por semejanza (descrito anteriormente en la Definición 9.3) y el Grado de semejanza generalizado entre conjuntos difusos (cuya descripción podemos encontrar en la Definición 7.6) como métodos para determinar el grado de inclusión y de semejanza de dos colecciones difusas, respectivamente.

Creación de la base de datos

Una vez definida la estructura del descriptor de las radiografías, podemos crear una tabla para almacenar radiografías junto a sus descriptores empleando la sentencia SQL que se muestra a continuación.

```
create table APXRay (
    image# number,
    xray bfile,
    SpineDescription SpineCurves
    );
```

Con el objeto de ilustrar el proceso de recuperación, se han insertado en la tabla anterior 20 radiografías junto con sus descriptores. Cada inserción se ha realizado empleando una sentencia SQL similar a la siguiente:

```
Insert into apxray values (
 313701 ,BFILENAME('APXRays','313701.gif'),
 SpineCurves(
     1,cobbCurvT(
        CurvDirectionT('RIGHT'),
        CobbAngleT(crisp(10.1)),
        VertbSetT('T2'),
        VertbSetT('T5'),
        VertbSetT('T10')
             ),
     1,cobbCurvT(
        CurvDirectionT('LEFT'),
        CobbAngleT(crisp(16.5)),
        VertbSetT('T9'),
        VertbSetT('T12'),
        VertbSetT('L2')
             ),
     1,cobbCurvT(
        CurvDirectionT('RIGHT'),
        CobbAngleT(crisp(12.5)),
        VertbSetT('L2'),
        VertbSetT('L4'),
        VertbSetT('L5')
             )
  )
);
```

9.4.2.2. Ejemplos de consultas

En el presente apartado, ilustraremos las capacidades de recuperación que posee el sistema gracias a los operadores de inclusión y semejanza de CFCs.

Se ha de notar que, para reflejar la imprecisión en la medida de los ángulos de Cobb, los valores *crisp* almacenados en la base de datos han sido transformados en distribuciones de posibilidad trapezoidal. Dichas distribuciones se construyen en base a un núcleo, obtenido como el intervalo resultante de relajar el valor *crisp* un 40 %, y un soporte, modelado a partir del intervalo que resulta cuando se relaja el valor *crisp* un 70 %.

Como primera tarea de recuperación, plantearemos la obtención de todas aquellas radiografías que incluyan, entre las curvas detectadas en la columna, una curva similar a la propuesta como condición. Un ejemplo de este tipo de consulta es "Obtener todas aquellas radiografías que incluyan una curva torácica pronunciada". El patrón de curva torácica propuesto en [119] se describe por los siguiente parámetros aproximados:

1. La convexidad de la curva se sitúa en el lado derecho.
2. La vértebra superior se sitúa entre las vértebras $T4$ y $T8$.
3. La vértebra apex está situada entre las vértebras $T8$ y $T9$.
4. La vértebra inferior se sitúa entre las vértebras $T11$ y $L2$.

Finalmente, definiremos el concepto de *curvatura pronunciada* como una curva con un grado mayor o igual a 25–30 grados.

En función de la descripción anterior, modelaremos la condición de búsqueda mediante la sentencia SQL que se muestra a continuación:

```
SELECT ap.image#,ap.xray,ap.cdeg(1)
FROM apxray ap
WHERE FCOND(
          FINCLUSION(
               ap.spinedescription,
               SpineCurves(1,
                    cobbCurvT(
                         CurvDirectionT('RIGHT'),
                         CobbAngleT(
                              Trapezoid(25,30,120,120)
                         ),
                         VertbSetT('T5'),
                         VertbSetT('T8'),
                         VertbSetT('T12')
                    )
               ) ),1)>0 order by cdeg(1) desc;
```

En la Figura 9.11 se observan las imágenes recuperadas por la anterior consulta de la base de datos de ejemplo. La primera de ellas (según el orden de izquierda a derecha) incluye una curva, destacada por un área rectangular, cuya convexidad está situada a la derecha, su ángulo de Cobb es de 21.8 y cuyas vértebras superior, apex e inferior son $T5$, $T8$ y $T10$, respectivamente. Como se

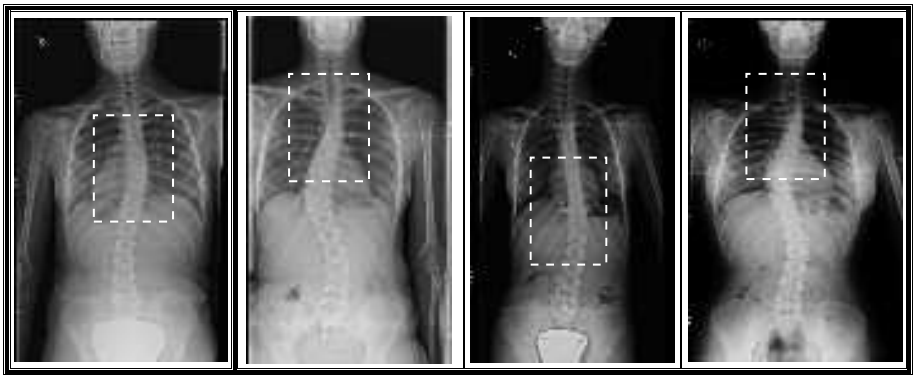

Figura 9.11: Radiografías de pacientes con una curva torácica

puede apreciar, dicha imagen contiene una curva de las características buscadas, como confirma su grado de satisfacción de la condición, 0.91. Las siguientes radiografías contienen imágenes de columnas vertebrales con curvas que, aunque similares a la requerida, no se ajustan completamente al concepto de curva torácica. Obsérvese que el grado de relevancia asociado a éstas, 0.6 para la segunda radiografía de la Figura 9.11 y 0.5 para la tercera y cuarta, decrece en la medida en que las curvas presentes en las imágenes se alejan del patrón empleado como condición.

En otros tipos de estudios puede resultar interesante encontrar, dado un paciente, radiografías de otros pacientes con patrones de curvas similares. En el sistema descrito, esta consulta se basará en el cálculo de similitud de las CFCs que contienen las curvas que caracterizan dichas columnas. Podremos expresar dicha consulta en SQL como se muestra a continuación:

```
SELECT ap1.image#,ap1.xray,ap1.cdeg(1)
FROM apxray ap1, apxray ap2
WHERE ap1.image='q'
     AND FCOND(
          FEQ(ap1.spinedescription, ap2.spinedescription),1
         )>0 order by cdeg(1) desc;
```

En la Figura 9.12 q) se muestra la radiografía cuyo conjunto de curvas servirá como patrón para una consulta de este tipo. Esta consulta, aplicada sobre la base de datos de ejemplo, devuelve como resultados las imágenes a) a e) de la citada figura. Al igual que el caso de la consulta anterior, el grado de relevancia de los resultados (indicado en la misma figura) decrece en la medida en que el conjunto de curvas presentes en las imágenes se asemeja menos al conjunto de curvas que caracteriza la radiografía empleada como condición.

9.5. Conclusiones

En el presente capítulo se han presentado diversos ejemplos de aplicaciones basadas en el uso de un SGBDORD. En dichos ejemplos, se muestra cómo las capacidades de representación y manipulación de datos difusos complejos, así

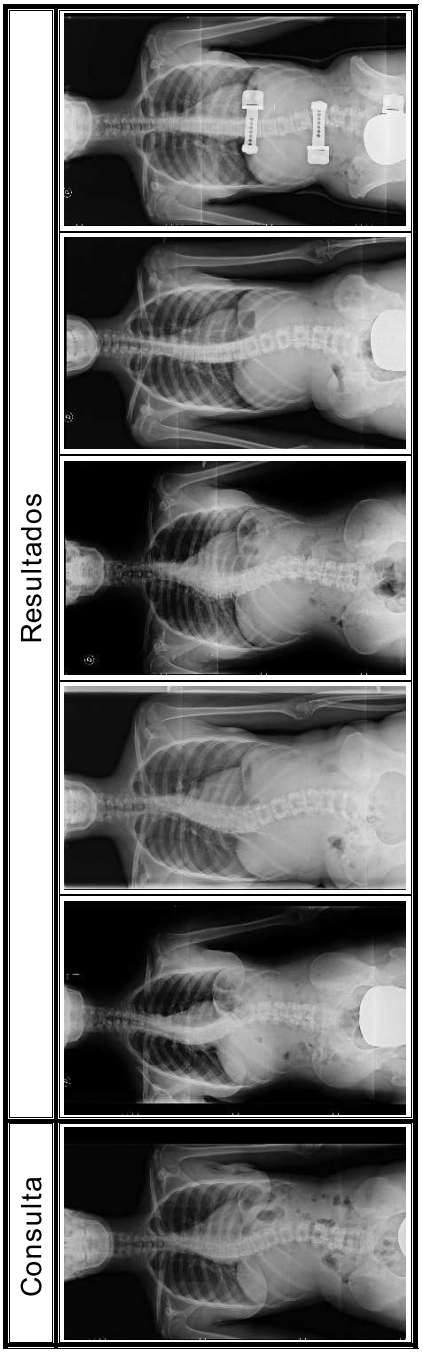

Figura 9.12: Radiografías de pacientes con un mismo patrón de curvas

como la capacidad de procesamiento de consultas de esta clase de sistemas, mejoran los resultados obtenidos por las aplicaciones.

La capacidad de representación y manipulación de datos difusos permite a las aplicaciones el tratamiento, de forma transparente, de información afectada por imprecisión o incertidumbre. Además, al posibilitar el manejo de datos complejos, la representación y modelado de datos para las aplicaciones resulta más sencilla y directa, incluso cuando la estructura compleja se ve afectada a su vez por la incertidumbre. Tales capacidades resultan en una representación natural y directa de los datos procesados, así como de una manipulación adecuada de los mismos.

La posibilidad de tratamiento de consultas flexibles añade un nivel mayor de naturalidad a los sistemas. Éstos, gracias al SGBDORD en la que se apoyan, pueden ofrecer una interfaz de consulta más natural e intuitiva para el usuario. De esta forma, el usuario puede expresar sus requisitos empleando un lenguaje que no le sea extraño y flexibilizar éstos tanto como deseé.

Las características anteriormente mencionadas han proporcionado los siguientes beneficios para las aplicaciones introducidas en el presente capítulo:

- Para el caso de la aplicación de gestión inmobiliaria, el uso de un SGBDORD ha supuesto una mejora para su sistema de consulta. Éste ha permitido expresar de forma natural los requisitos a clientes y, mediante la inclusión de resultados con distintos niveles de relevancia, ha permitido el ajuste de la oferta a la demanda realizada (lo que resulta en un potencial incremento de negocio).

- En el caso del sistema de recuperación de imágenes en base a sus colores dominantes, el uso de un SGBDORD ha permitido la expresión de colores de forma natural. Esta nueva capacidad ayuda a los usuarios a expresar sus consultas con mayor facilidad, sin necesidad de que posean experiencia previa, flexibiliza las condiciones para permitir resultados parciales y permite la obtención de resultados categorizados en función de su relevancia.

- Para el sistema de recuperación de radiografías representando casos de escoliosis, el empleo de un SGBDORD ha permitido en manejo de la imprecisión inherente a las medidas de ángulos de Cobb que caracterizan tales imágenes. Además de lo anterior, el sistema se beneficia de una capacidades de expresión de consultas de forma natural, así como de una categorización de los resultados en función de su relevancia.

Capítulo 10

Conclusiones y líneas futuras

El presente trabajo ha introducido una serie de propuestas destinadas al establecimiento de una base sólida para la creación de SGBDs objeto-relacionales difusos. Este tipo de SGBD se caracteriza por proporcionar capacidades complejas de modelado, similares a las disponibles el los SGBDs objeto-relaciones *crisp* actuales, junto con la posibilidad de representar y manipular variantes imperfectas de los tipos de datos que éstos soportan.

Esta propuesta proviene del análisis de los diferentes SGBDD disponibles actualmente y las tendencias de la industria de desarrollo de sistemas de gestión de bases de datos. Como resultado, se ha propuesto un modelo base de datos que incluye las características anteriormente mencionadas, así como un álgebra como herramienta formal para la manipulación de los datos presentes en las bases de datos que siguen este modelo.

Más allá del ámbito teórico, en el presente trabajo se ha desarrollado un marco para la creación de sistemas capaces de tratar información imperfecta, basándose en la extensión de un SGBD objeto-relacionales *crisp*. Esta estrategia permite que el esfuerzo de implementación, para crear el sistema de gestión de bases de datos difusas, sea mucho menor y heredar todas las características del sistema anfitrión, como pueden ser el rendimiento del mismo, y sus características de valor añadido (distribución de bases de datos, clustering, tratamiento de datos espaciales, etc.). Gracias a estas características, el SGBDD propuesto será apropiado para ir más allá del ámbito experimental y formar parte de entornos en producción.

Además de lo anterior, este trabajo de investigación ha supuesto también la implementación de un prototipo, de la citada propuesta, sobre un SGBD actual, concretamente Oracle. Este prototipo ha demostrado la viabilidad real de la propuesta de extensión, y la utilidad de la misma para el tratamiento de información imperfecta.

Con objeto de incrementar el rendimiento de los SGBDDs, uno de los aspectos que dificultan su implantación en entornos de producción, se han propuesto una serie de técnicas de indexado para datos imprecisos. Estas técnicas, 2BPT, HBPT y BBC, han demostrado un rendimiento comparable, o incluso superior en algunos casos, al de las técnicas de referencia disponibles en la actualidad. Además del rendimiento de las mismas, se ha de destacar su gran estabilidad ante diversas variaciones en la naturaleza de datos y consultas. Esta estabilidad ha resultado especialmente superior con respecto a la técnica de referencia para

el caso de 2BPT y HBPT y ha superado en un orden de magnitud a la técnica de referencia en el caso de BBC. Adicionalmente al buen comportamiento en lo que se refiere al rendimiento, se ha de destacar que, en línea con las propuestas de este trabajo, las diferentes técnicas de indexado propuestas están basadas en las estructuras de indexado clásicas empleadas en los SGBD objeto-relaciones *crisp* disponibles en la actualidad. Este hecho, facilitará la implantación de dichas técnicas en SGBD objeto-relaciones difusos desarrollados como extensión de SGBD objeto-relacionales *crisp*.

Finalmente, se han detallado diferentes aplicaciones en diversos ámbitos del prototipo de SGBDORD propuesto en el presente trabajo. Estas sirven de muestra para comprobar la utilidad de tales sistemas en la gestión de información compleja, especialmente cuando ésta está afectada por cierto grado de imprecisión o incertidumbre.

Las líneas futuras, que determinarán la forma en las que evolucione el trabajo presentado, son, entre otras, las siguientes:

- Estudio de las propiedades del álgebra propuesta en el presente trabajo. El estudio de estas propiedades permitirá el desarrollo de sistemas de optimización de consultas, lo que redundará en un mayor rendimiento de los SGBDDs basados en el modelo propuesto.
- Incorporación de nuevos tipos de datos imperfectos a la propuesta de implementación. Esto permitirá modelar, con mayor facilidad, ciertos tipos de datos de dominios concretos, como pueden ser por ejemplo, los derivados del área de procesamiento de imágenes empleando lógica difusa.
- Consideración de nuevos operadores para la manipulación de los datos imperfectos, lo que permitirá la incorporación de nuevas capacidades de consulta para el presente modelo.
- Dotar al prototipo de SGBDORD de capacidades para que las t-normas, t-conormas, negaciones fuertes, implicaciones, agregadores y otras funciones aplicadas por los operadores, puedan ser adaptadas por parte del usuario. Esta incorporación supondría una ventaja significativa ya que permitiría adaptar el modelo a las necesidades de problemas concretos.
- Estudiar, definir, e implementar mecanismos de indexado eficientes para mejorar el rendimiento del procesamiento de consultas sobre colecciones difusas y otro tipo de datos complejos. La incorporación de estos mecanismos mejoraría el rendimiento para las consultas de este tipo en las implementaciones de SGBDDs y, por tanto, incrementaría la aplicabilidad de los mismos en problemas reales con grandes volúmenes de datos complejos.
- Definir técnicas de indexado directas, no basadas en los principios de indexado de Bosc. La disponibilidad de este tipo de técnicas permitiría aumentar aún más el rendimiento de los SGBDDs, ya que reduciría el número de falsos positivos que introducen los filtros actuales basados en los mencionados principios de indexado.
- Ampliar el prototipo actual para incorporar en el mismo la totalidad de las propuestas actuales y las futuras provenientes de las anteriores líneas futuras.

Concluding remarks and future works

This work has introduced several proposals leading to set up a solid base for the building of a object-relational fuzzy DBMS. This sort of DBMS is distinguished by their complex modeling capabilities, similar to the ones of current crisp object-relational DBMS, together with the possibility of managing imperfect variants of the data types supported by this class of crisp systems.

This proposal is originated from the analysis of different DBMS currently available and from the trends of DBMS development industry. As result, we propose the development of a new database modelo whose data model includes all the above features and, also, an algebra as a formal tool for managing the data in a data base following this model.

Beyond the theoretic framework, this work has elaborated a practical framework to develop systems able to deal with imperfect information, as an extension of current crisp object-relational DBMS. This strategy makes the implementation effort, of the development of a fuzzy DBMS, smaller and inherits all host system features, as its performance and its added value features (distributed databases, clustering, spacial data management, etc.). Thanks to the mentioned properties and features, the proposed fuzzy DBMS would be suitable to exceed the experimental scope and take part in a production environment.

Additionally, this research work has also involved the development of a prototype, of the above proposal, into a current DBMS (Oracle). This prototype has shown the actual viability of the extension proposal and its ability for dealing with imperfect information.

With the purpose of increasing the fuzzy DBMS performance, one of the main obstacle that prevent this kind of systems in production environments, we have developed several indexing techniques for imprecise data. This techniques, 2BPT, HBPT and BBC, has shown comparable, or even greater in some cases, performance than the measured for state-of-the-art reference techniques. Moreover, regarding to the performance of the proposed techniques, we have to highlight its good stability under different data and queries scenarios. The mentioned stability is especially greater than the measured for the reference technique for 2BPT and HBPT and is over an order of magnitude for BBC. In addition the good behavior of the proposals performance, we have to emphasize that, following this work proposal, the different proposed indexing techniques are based on classical indexing structures that are used on current crisp object-relational DBMS. This will ease the introduction of these techniques in fuzzy object-relational DBMS developed as and extension of crisp ones.

Finally, we have depicted different applications for fuzzy object-relational DBMSs in different areas. These applications help to show that these systems are useful for managing complex data, especially when it is pervaded with imprecision or uncertainty.

Future works, that would determine the way this proposal evolves, are, among others, the following:

- To study the properties of the algebra proposed in this work. The study of these properties would allow the development of query optimization systems, which would result in a greater performance of the fuzzy DBMSs based on the proposed model.
- To include new imprecise data types to the implementation proposal. This would allow to model, more easily, some particular data types as, for instance, data types from image processing area.
- To consider new operator for imperfect data manipulation, which would allow to include new query capabilities for the proposed model.
- To provide the implementation model with capabilities for user-defined t-norms, t-conorms, strong negations, implications, agregators and other functions. This feature would provide a significant advantage as it would allow the model to be adapted to particular needs.
- To study, define and implement efficient indexing techniques to improve processing performance of queries on fuzzy collections and other complex data types. The inclusion of these techniques would increase the load capacity of fuzzy DBMS implementations for this query types and, therefore, would increase their applicability in real world problems with high volumen of complex data.
- To define direct indexing techniques, not based on Bosc the indexing principles. The availability of this kind of techniques would allow to increase even more the performance of fuzzy DBMS, as this would reduce then number of false positive introduced by current filters based on the aforementioned indexing principles.
- To expand the current prototype to include the entire set of current proposals and future one resulting from the above future works.

Acrónimos

AP	Anteroposterior (*Tipo de radigrafía*)
API	Application programming interface (*Interfaz de programación de aplicaciones*)
CAD	Computer Aided Design (*Diseño asistido por computadora*)
CAM	Computer Aided Manufacturing (*Fabricación asistida por computadora*)
CASE	Computer Aided Software Engineering (*Ingeniería del software asistida por computadora*)
CBIR	Content based image retrieval (*Recuperación de imágenes en base a su contenido*)
CFC	Conjunctive fuzzy collection
CLR	Common language runtime
DBMS	Database management system (SGBD en Castellano)
DCL	Data control language
DDL	Data definition language
DFC	Disjunctive fuzzy collection
DFSQL	Deductive FSQL (ver FSQL)
DML	Data manipulation language
FEQ	Fuzzy equal
FIRST	Fuzzy Interface for Relational Systems (*Interfaz difusa para sistemas relacionales*)
FMB	Fuzzy Meta-knowledge Base (*Base de metaconocimiento difuso*)
FNP	Forma normal de particiones
FNPE	Forma normal de particiones extendida
FO	Fuzzy object

FoodBi	Fuzzy Object Oriented DataBase Interface (*Interfaz de base de datos orientada a objetos difusos*)
FSQL	Fuzzy SQL (ver SQL)
GEFRED	Generalized Model of Fuzzy Relational Databases (*Modelo generalizado para bases de datos difusas*)
GIS	Geographic Information System (*Sistema de información geográfica*)
HSI	Hue-Saturation-Intensity (*Espacio de color Tono-Saturación-Intensidad*)
ISBL	Information systems base language
ISW	InmoSoftWeb
JRT	Java routines and types
LRN	Longitud relativa del núcleo
LRS	Longitud relativa del soporte
MRCD	Multirrelación compleja difusa
NF^2	Non First Normal Form (*Modelo de relaciones anidadas*)
NFEQ	Necessarily fuzzy equal
NOAFT	Non ordered atomic fuzzy type
OAFT	Ordered atomic fuzzy type
OLB	Object language bindings
ODMG	Object Data Management Group ó, tras 1998, Object Database Management Group
PACS	Picture archiving and Communication Systems (*Sistema de archivado y transmisión de imágenes*)
RGB	Red-Blue-Green (*Espacio de color Rojo-Azul-Verde*)
RLE	Run-length encoding (*Tipo de codificación*)
RMI	Resonancia magnética por imágenes
SEQUEL	Structured English Query Language (*Lenguaje Inglés de consulta estructurado*)
SGBD	Sistema de gestión de bases de datos
SGBDD	Sistema de gestión de bases de datos difusas
SGBDORD	Sistema de gestión de bases de datos objeto-relacional difuso
SO	Sistema operativo

SQL	Structured query Language (*Lenguaje de consulta estructurado*)
TC	Tomografía computerizada
UDT	User defined type (*Tipo definido por el usuario*)
VB	Variable-byte (*Tipo de codificación*)
XML	Extensible Markup Language (*Lenguaje extensible de marcas*)

Bibliografía

[1] G. Antoshenkov. Byte aligned data compression (nov. 1994).

[2] G. Antoshenkov. Byte-aligned bitmap compression. En "Data Compression Conference, 1995. DCC '95. Proceedings", pág. 476 (mar. 1995).

[3] G. Antoshenkov y M. Ziauddin. Query processing and optimization in Oracle Rdb. *The VLDB Journal* **5**(4), 229–237 (dic. 1996).

[4] M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier y S. Zdonik. The object-oriented database manifesto. En "Proceedings of the First International Conference on Deductive and Object-Oriented Databases", págs. 223–240, Kyoto, Japan (dic. 1989).

[5] J. Baldwin. Knowledge engineering using a fuzzy relational inference language. En "Proc. IFAC Symp. on Fuzzy Information Knowledge Representation and Decission Analisys", págs. 12–21 (1983).

[6] F. Bancilhon, G. Barbedette, V. Benzaken, C. Delobel, S. Gamerman, C. Lecluse, P. Pfeffer, P. Richard y F. Velez. The design and implementation of O_2, an object-oriented database system. En K. Dittrich, editor, "Advances in Object-Oriented Database Systems", nº 334 en Lecture Notes in Computer Science, págs. 1–22. Springer-Verlag (1988).

[7] D. Barbara, H. Garcia-Molina y D. Porter. The management of probabilistic data. *IEEE Transactions on Knowledge and Data Engineering* **04**(5), 487–502 (1992).

[8] C. D. Barranco, J. Campaña, J. M. Medina y O. Pons. Immosoftweb: a web based fuzzy application for real estate management. En J. F. et al., editor, "Advances in Web Intelligence", vol. 3034 de "Lecture Notes in Computer Science", págs. 196–206. Springer Berlin / Heidelberg (2004).

[9] C. D. Barranco, J. R. Campaña y J. M. Medina. A B^+-tree based indexing technique for fuzzy numerical data. *Fuzzy Sets and Systems* **159**(12), 1431–1449 (2008).

[10] C. D. Barranco, J. R. Campaña y J. M. Medina. Towards a fuzzy object-relational database model. En J. Galindo, editor, "Handbook of Research on Fuzzy Information Processing in Databases", cap. 17, págs. 431–461. Information Science Reference (2008).

[11] C. D. Barranco, J. M. Medina, J. Chamorro-Martínez y J. M. Soto-Hidalgo. Using a fuzzy object-relational database for colour image retrieval. En H. L. Larsen, G. Pasi, D. Ortiz-Arroyo, T. Andreasen y H. Christiansen, editores, "Flexible Query Answering Systems: 7th International Conference", vol. 4027 de "Lecture Notes in Artificial Intelligence", págs. 307–318. Milan (Italy) (June 2006).

[12] R. Bayer y E. M. McCreight. Organization and maintenance of large ordered indexes. *Acta Informatica* **1**(3), 173–189 (1972).

[13] R. Bellman y M. Giertz. On the analytic formalism of the theroy of fuzzy sets. *Information Sciences* **5**, 149–156 (1973).

[14] M. Benjelloun y S. Mahmoudi. Spine localization in x-ray images using interest point detection. *Journal of Digital Imaging. Published online, doi:10.1007/s10278-007-9099-3* (2008).

[15] I. Blanco. "Deducción en Bases de Datos Relacionales Difusas". Tesis Doctoral, Departamento de Ciencias de la Computación e Inteligencia Artificial, Universidad de Granada, España (2001).

[16] I. Blanco, J. Cubero, O. Pons y M. Vila. An implementation for fuzzy deductive relational databases. En "Recent Issues on Fuzzy Databases", Studies in Fuzziness and Soft Computing Series, págs. 183–207. Physica-Verlag (2000).

[17] W. D. Blizard. Real-values multisets and fuzzy sets. *Fuzzy Sets Syst.* **33**(1), 77–97 (1989).

[18] W. D. Blizard. A theory of shadows (an informal discussion of negative membership). *Journal of the Western regional Chapter of the Alternative Natural Philosophy association* **1**(3), 7–9 (1989).

[19] W. D. Blizard. Correction to: "The development of multiset theory". *Modern Logic* **2**(2), 219 (1991).

[20] W. D. Blizard. The development of multiset theory. *Modern Logic* **1**(4), 319–352 (1991).

[21] W. D. Blizard. Dedekind multisets and function shells. *Theoretical Computer Science* **110**(1), 79–98 (1993).

[22] W. D. Blizard. Updates and corrections to: "The development of multiset theory". *Modern Logic* **7**(3–4), 434 (1997).

[23] G. Bordogna, D. Lucarella y G. Pasi. A fuzzy object oriented data model. En "Proceedings of FUZZ-IEEE", págs. 313–317 (1994).

[24] G. Bordogna, G. Pasi y D. Lucarella. A fuzzy object oriented data model for managing vague and uncertain information. *International Journal of Intelligent Systems* (1999).

[25] P. Bosc, D. Dubois, O. Pivert y H. Prade. Flexible queries in relational databases: The example of the division operator. *Theroretical Computer Science* **171**, 281–302 (1997).

[26] P. Bosc y M. Galibourg. Indexing principles for a fuzzy data base. *Information Systems* **14**(6), 493–499 (1989).

[27] P. Bosc, M. Galibourg y G. Hamon. Fuzzy querying with SQL: Extensions and implementation aspects. *Fuzzy Sets and Systems* **28**, 333–349 (1998).

[28] P. Bosc, D. Kraft y F. Petry. Fuzzy sets in database and information systems: Status and opportunities. *Fuzzy Sets and Systems* **156**(3), 418–426 (2005). 40th Anniversary of Fuzzy Sets.

[29] P. Bosc y O. Pivert. Fuzzy querying in conventional databases. En "Fuzzy logic for the management of uncertainty", cap. Fuzzy querying in conventional databases, págs. 645–671. John Wiley & Sons, Inc. (1992).

[30] P. Bosc y O. Pivert. SQLf: A relational database language for fuzzy querying. *IEEE Transactions on Fuzzy Systems* **3**, 1–17 (1995).

[31] P. Bosc y H. Prade. An introduction to the fuzzy set and possibility theory-based treatment of flexible queries and uncertain or imprecise databases. En "Uncertainty Management in Information Systems", págs. 285–324. (1996).

[32] P. Bosc y D. Rocacher. About Z_f, the set of fuzzy relative integers, and the definition of fuzzy bags on Z_f. En T. Bilgiç, B. De Baets y O. Kaynak, editores, "Fuzzy Sets and Systems – IFSA 2003", vol. 2715 de "Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)", págs. 25–39. Springer Berlin / Heidelberg (2003).

[33] B. Boss y S. Helmer. Index structures for efficiently accessing fuzzy data including cost models and measurements. *Fuzzy Sets and Systems* **108**(1), 11–37 (nov. 1999).

[34] B. Buckles y F. Petry. "Fuzzy Information and Decision Processes", vol. 2, cap. Fuzzy Databases and Their Applications, págs. 361–371. North-Holland, Amsterdam (1982).

[35] B. Buckles y F. Petry. A fuzzy representation of data for relational databases. *Fuzzy Sets ans Systems* **7**, 213–226 (1982).

[36] B. Buckles, F. Petry y H. Sachar. A domain calculus for fuzzy relational databases. *Fuzzy Sets ans Systems* **29**, 327–340 (1989).

[37] R. Carrasco. "Lenguajes e Interfaces de Alto Nivel para Data Mining con Aplicación Práctica a Entornos Financieros". Tesis Doctoral, Departamento de Ciencias de la Computación e Inteligencia Artificial, Universidad de Granada, España (2003).

[38] R. Carrasco, M. Vila y J. Galindo. FSQL: a flexible query language for data mining. En "Enterprise Information Systems IV", págs. 68–74. Kluwer Academic Publisher, Dordrech, Holanda (2003).

[39] R. Cattell y D. Barry. "The Object Data Standard: ODMG 3.0". Morgan Kaufmann Publishers Inc. (2000).

[40] K. Chakrabarty. Notion of fuzzy IC-Bags. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* **12**(3), 327–346 (2004).

[41] K. Chakrabarty, R. Biswas y S. Nanda. Fuzzy shadows. *Fuzzy Sets Syst.* **101**(3), 413–421 (1999).

[42] D. Chamberlin y R. Boyce. SEQUEL: A structured english query language. En "Proc. 1974 ACM SIGMOD Workshop on Data Description, Access, and Control", Ann Arbor, Michigan (May 1974).

[43] D. e. a. Chamberlin. SEQUEL/2: A unified approach to data definition, manipulation and control. *IBM J. R&D.* **2**(6) (November 1976).

[44] J. Chamorro-Martínez, J. Medina, C. Barranco, E. Galán-Perales y J. Soto-Hidalgo. An approach to image retrieval on fuzzy object-relational database using dominant color descriptors. En "Proceedings of 4th Conference of the European Socienty for Fuzzy Logic and Technology", págs. 676–684 (2005).

[45] J. Chamorro-Martínez, J. Medina, C. Barranco, E. Galán-Perales y J. Soto-Hidalgo. Retrieving images in fuzzy object-relational databases using dominant color descriptors. *Fuzzy Sets and Systems* **158**(3), 312–324 (February 2007).

[46] J. Chamorro-Martínez, D. Sánchez, B. Prados-Suarez y E. Galán-Perales. Fuzzy connectivity measures for path-based image segmentation. En "Proceedings of IEEE International Conference on Fuzzy Systems", págs. 218–223 (2005).

[47] J. Cobb. Outline for the study of scoliosis. *Am Acad Orthop Surg Inst Course Lect* **5**, 261–275 (1948).

[48] E. Codd. A relational model of data for large shared data banks. *Communications of the ACM* **13**(6), 337–38 (June 1970).

[49] E. Codd. Relational completeness of data base sublanguages. En "Data Base Systems", vol. 6 de "Courant Computer Science Symposia Series", págs. 65–98. Pretince Hall, Englewood Clifs, New Jersey (1972).

[50] E. Codd. Extending the database relational model to capture more meaning. *ACM Transactions on Database Systems* **4**, 262–296 (1979).

[51] E. Codd. Missing information (applicable and inapplicable) in relational databases. *ACM SIGMOD Record* **15**(4) (1986).

[52] E. Codd. More comentary on missing information in relational databases. *ACM SIGMOD Record* **16**(1) (1987).

[53] E. Codd. "Relational model for Database Management, Version 2". Reading Mass. Addison-Wesley (1990).

[54] D. Comer. Ubiquitous B-tree. *ACM Comput. Surv.* **11**(2), 121–137 (1979).

[55] J. C. Cubero, N. Marín, J. M. Medina, O. Pons y M. A. Vila. Fuzzy object management in an object-relational framework. En "Proceedings of X Intl. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU)", págs. 1767–1774 (2004).

[56] L. Cuevas, N. Marín, O. Pons y M. Vila. pg4DB: A fuzzy object-relational system. *Fuzzy Sets and Systems* **159**(12), 1500 – 1514 (2008). Advances in Intelligent Databases and Information Systems.

[57] L. Cuevas Rodríguez. "Modelo difuso de bases de datos objeto-relacional: propuesta de implementación en software libre". Tesis Doctoral, Departamento de Ciencias de la Computación e inteligencia artificial, Universidad de Granada (2009).

[58] C. Date. Null values in database management. En C. Date, editor, "Relational Database: Selected Writings". Addison Wesley (1986).

[59] C. Date. "Relational database writings 1989–1991". Addison Wesley (1992).

[60] C. Date. "Introducción a los sistemas de bases de datos". Pearson Educación (2001).

[61] C. Date y H. Darwen. "Foundation for Objet/Relacional Databases. The Third Manifesto". Addison Wesley Longman (1998).

[62] G. de Tré y R. de Caluwe. A generalised object-oriented database model. En G. Bordogna y G. Pasi, editores, "Recent Issues on Fuzzy Databases", págs. 155–182. Physica-Verlag, Heidelberg, Germany (2000).

[63] G. de Tré y R. de Caluwe. Level-2 fuzzy sets and their usefulness in object-oriented database modelling. *Fuzzy Sets and Systems* **140**(1), 29 – 49 (2003). Fuzzy Databases.

[64] R. Dedekind. "Was sind und was sollen die Zahlen?" (1888).

[65] A. Del Bimbo. "Visual information retrieval". Morgan Kaufmann Publishers (2001).

[66] M. Delgado, M.-J. Martín-Bautista, D. Sánchez y M.-A. Vila. On a characterization of fuzzy bags. En T. Bilgiç, B. De Baets y O. Kaynak, editores, "Fuzzy Sets and Systems – IFSA 2003", vol. 2715 de "Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)", págs. 84–92. Springer Berlin / Heidelberg (2003).

[67] M. Delgado, M.-J. Martín-Bautista, D. Sánchez y M.-A. Vila. An extended characterization of fuzzy bags. *International Journal of Intelligent Systems* **24**(6), 706–721 (2009).

[68] D. Dubois y H. Prade. Fuzzy number: An overview. En J. Bezdek, editor, "Analisys of Fuzzy Information". Crc Press, Bocan Ratón, Florida (1985).

[69] S. Fukami, M. Umano, M. Mizimoto y H. Tanaka. Fuzzy databases retrieval and manipulation language. *IEICE Technical Reports* **78**(233), 65–72 (1979).

[70] J. Galindo. "Tratamiento de la Imprecisión en Bases de Datos Relacionales: Extensión del Modelo y Adaptación de los SGBD Actuales". Tesis Doctoral, Departamento de Ciencias de la Computación e Inteligencia Artificial, Universidad de Granada, España (1999).

[71] J. Galindo, A. Urrutia y M. Piattini. "Fuzzy Databases: Modeling, Design and Implementation". Idea Group Publishing, Hershey, PA, USA (2006).

[72] R. George, B. Buckles y F. Petry. Modelling class hierarchies in the fuzzy object-oriented data model. *Fuzzy Sets and Systems* **60**, 259–272 (1993).

[73] C. Giardina. Fuzzy databases and fuzzy relational associative processors. Informe t'ecnico, Stevens Institute of Technology, Hoboken, N.J. (1979).

[74] J. Goguen. The logic of inexeact concepts. *Synthese* **9**(3–4), 325–373 (1969).

[75] J. Grant. Incomplete information in a relational daytabase. *Fundamenta Informaticae* **3**, 363–378 (1980).

[76] P. W. P. J. Grefen y R. A. d. By. A multi-set extended relational algebra - a formal approach to a practical issue. En "Proceedings of the Tenth International Conference on Data Engineering", págs. 80–88, Washington, DC, USA (1994). IEEE Computer Society.

[77] P. A. V. Hall, P. Hitchcock y S. J. P. Todd. An algebra of relations for machine computation. En "Conference Record of the Second ACM Symposium on Principles of Programming Languages", págs. 225–232, Palo Alto, California (January 1975).

[78] S. Helmer. Evaluating different approaches for indexing fuzzy sets. *Fuzzy Sets and Systems* **140**(1), 167–182 (2003).

[79] S. Helmer y G. Moerkotte. A performance study of four index structures for set-valued attributes of low cardinality. *VLDB Journal* **12**(3), 244–261 (2003).

[80] G. Hulin. On restructuring nested relations in partitioned normal form. En "VLDB '90: Proceedings of the 16th International Conference on Very Large Data Bases", págs. 626–637, San Francisco, CA, USA (1990). Morgan Kaufmann Publishers Inc.

[81] A. Jain y R. Dubes. "Algorith for clustering data". Pretince Hall (1988).

[82] S. Khoshafian. "Object-Oriented Databases". John Wiley and Sons, New York (1993).

[83] K.-S. Kim y S. Miyamoto. Application of fuzzy multisets to fuzzy database systems. En "Fuzzy Systems Symposium, 1996. 'Soft Computing in Intelligent Systems and Information Processing'.", págs. 115–120 (1996).

[84] G. Klir y T. Folger. "Fuzzy Sets, Uncertainty, and Information". Prentice-Hall International, London (1988).

[85] G. Klir y B. Yuan. "Fuzzy Sets and Fuzzy Logic: Theory and Applications". Prentice-Hall, Englewood Cliffs, New Jersey (1995).

[86] D. Knuth. "The art of computer programming", vol. 2, Seminumerical Algorithms. Addison-Wesley (1981).

[87] D. H. Kraft y F. E. Petry. Fuzzy information systems: managing uncertainty in databases and information retrieval systems. *Fuzzy Sets and Systems* **90**(2), 183–191 (1997). Fuzzy Sets: Where Do We Stand? Where Do We Go?

[88] A. Kumar. G-tree: a new data structure for organizing multidimensional data. *IEEE Transactions on Knowledge and Data Engineering* **6**(2), 341–347 (1994).

[89] M. Lacroix y A. Pirotte. Domain-oriented relational languages. En "Proc. 3rd International Conference on Very Large Data Bases" (October 1997).

[90] G. Lamperti, M. Melchiori y M. Zanella. On multisets in database systems. En C. S. Calude, G. Păun y G. R. A. Salomaa, editores, "Multiset Processing", vol. 2235 de "Lecture Notes in Computer Science", págs. 147–215. Springer Berlin / Heidelberg (2001).

[91] J. Lawder y P. King. Using space-filling curves for multi-dimensional indexing. En "Advances in Databases 17th British National Conference on Databases, BNCOD 17 Exeter, UK, July 3–5, 2000 Proceedings", vol. 1832 de "Lecture Notes in Computer Science", págs. 20–35 (2000).

[92] M. Levene y G. Loizou. The nested relation type model: an application of domain theory to databases. *Comput. J.* **33**(1), 19–30 (1990).

[93] M. Levene y G. Loizou. Correction to null values in nested relational databases by M.A. Roth, H.F. Korth, and A. Silberschatz. *Acta Inf.* **28**(6), 603–605 (1991).

[94] D. Li y D. Liu. "A Fuzzy Prolog Database System". John Wiley & Sons Inc. (1990).

[95] C. Liu, A. Ouksel, P. Sistla, J. Wu, C. Yu y N. Rishe. Performance evaluation of g-tree and its application in fuzzy databases. En "CIKM '96: Proceedings of the fifth international conference on Information and knowledge management", págs. 235–242, New York, NY, USA (1996). ACM Press.

[96] H.-C. Liu y J. X. Yu. Algebraic equivalences of nested relational operators. *Inf. Syst.* **30**(3), 167–204 (2005).

[97] Z. M. Ma. "Flexible Databases Supporting Imprecision and Uncertainty", vol. 203 de "Studies in Fuzziness and Soft Computing", cap. Modeling Fuzzy Information in the EER and Nested Relational Database Models, págs. 123–146. Springer Berlin / Heidelberg (2006).

[98] C. Manning, P. Raghavan y H. Schütze. "Introduction to Information Retrieval". Cambridge University Press (2008).

[99] M. Marín, J. Medina, O. Pons, D. Sánchez y M. Vila. Complex object comparison in a fuzzy context. *Inf. and Software Technology* **45(7)**, 431–444 (2003).

[100] N. Marín. "Estudio de la Vaguedad en los Sistemas de Bases de Datos Orientados a Objetos: Tipos Difusos y sus Aplicaciones". Tesis Doctoral, Departamento de Ciencias de la Computación e Inteligencia Artificial, Universidad de Granada, España (2001).

[101] N. Marín, O. Pons y M. A. Vila. Fuzzy types: A new concept of type for managing vague structure. *International Journal of Intelligent Systems* **15**(11), 1061–1085 (2000).

[102] N. Marín, O. Pons y M. A. Vila. A strategy for adding fuzzy types to an object-oriented database system. *International Journal of Intelligent Systems* **16**(7), 863–880 (2001).

[103] J. Medina. "Bases de Datos Relacionales Difusas: Modelo Teórico y Aspectos de su Implementación". Tesis Doctoral, Departamento de Ciencias de la Computación e Inteligencia Artificial, Universidad de Granada, España (1994).

[104] J. Medina, J. Galindo, F. Berzal y J. Serrano. Using object relational features to build a fuzzy database server. En "Proceedings of 9th Int.l Conf. of information processing and management of uncertainty in knowledge-based systems (IPMU 2002)" (2002).

[105] J. Medina, O. Pons y M. Vila. GEFRED. a generalized model of fuzzy relational databases. *Information Sciences* **76**(1–2), 87–109 (1994).

[106] J. M. Medina, S. Jaime-Castillo, C. D. Barranco y J. R. Campaña. Flexible retrieval of x-ray images based on shape descriptors using a fuzzy object-relational database. En J. P. Carvalho, D. Dubois, U. Kaymak y J. M. da Costa Sousa, editores, "Proc. of IFSA/EUSFLAT Conf.", págs. 903–908 (2009).

[107] J. Melton. "Advanced SQL:1999 Understanding Object-Relational and Other Advanced Features". Morgan Kaufmann (2003).

[108] J. Melton y A. Simon. "SQL:1999 Understanding Relational Language Components". Morgan Kaufmann (2002).

[109] S. Miyamoto. Fuzzy multisets and their generalizations. En G. Goos, J. Hartmanis y J. van Leeuwen, editores, "Multiset Processing", vol. 2235 de "Lecture Notes in Computer Science", págs. 225–235. Springer Berlin / Heidelberg (2001).

[110] S. Miyamoto. Generalizations of multisets and rough approximations. *International Journal of Intelligent Systems* **19**(7), 639–652 (2004).

[111] S. Miyamoto. Multisets and fuzzy multisets as a framework of information systems. En V. Torra y Y. Narukawa, editores, "Modeling Decisions for Artificial Intelligence", vol. 3131 de "Lecture Notes in Computer Science", págs. 1–28. Springer Berlin / Heidelberg (2004).

[112] W. Y. Mok. A comparative study of various nested normal forms. *IEEE Trans. on Knowl. and Data Eng.* **14**(2), 369–385 (2002).

[113] N. Mouaddib. Fuzzy identification in fuzzy databases: The nuanced relational division. *International Journal of Intelligent Systems* **9**, 461–473 (1994).

[114] R. Murthy, S. Sundara, N. Agarwal, Y. Hu, T. Chorma y J. Srinivasan. Supporting ancillary values from user defined functions in oracle. En "Data Engineering, 2003. Proceedings. 19th International Conference on", págs. 151–162 (2003).

[115] H. Nakajima, T. Sogoh y M. Arao. Fuzzy database language and library - fuzzy extension to SQL. En "Second International Conference on Fuzzy Systems, FUZZ-IEEE'93", págs. 477–482 (1993).

[116] Z. Pawlak. Rough sets. *International Journal of Computer and Information Sciences* **11**, 341–356 (1982).

[117] Z. Pawlak. "Rough Sets: Theoretical Aspects of Reasoning about Data". Kluwer Academic Publishers, Norweel, MA (1991).

[118] F. E. Petry y P. Bosc. "Fuzzy databases: principles and applications". International Series in Intelligent Technologies. Kluwer Academic Publishers (1996).

[119] I. Ponseti y B. Friedman. Prognosis in idiopathic scoliosis. *J Bone Joint Surgery* **32**(2), 381–395 (1950).

[120] V. Poosala. Zipf's law. Informe t'ecnico, University of Wisconsin Madison (1995).

[121] H. Prade. Lipski's approach to incomplete information data bases restated and generalized in the setting of zadeh's possibility theory. *Information Systems* **9**(1), 27–42 (1984).

[122] H. Prade y C. Testemale. Generalizaing database relational algebra for the treatment of incomplete/uncertain information and vague queries. *Information Sciences* **34**, 113–143 (1984).

[123] H. Prade y C. Testemale. "Analysis of Fuzzy Information", vol. 3, cap. Representation of Soft Constraints and Fuzzy Attribute Values by means of Possibility Distributions in Databases. CRC Press (1987).

[124] H. Prade y C. Testemale. Fuzzy relational databases: Representational issues and reduction using similarity measures. *Journal of american Society for Information Sciences* **38**(2), 118–126 (1987).

[125] P. Reisner, R. Boyce y D. Chamberlin. Human factors evaluation of two data base query languages: SQUARE and SEQUEL. En "Proc. NCC 44", Anaheim, California (May 1975). AFIPS Press.

[126] D. Rocacher. On fuzzy bags and their application to flexible querying. *Fuzzy Sets and Systems* **140**(1), 93 – 110 (2003). Fuzzy Databases.

[127] D. Rocacher y P. Bosc. The set of fuzzy rational numbers and flexible querying. *Fuzzy Sets and Systems* **155**(3), 317 – 339 (2005).

[128] J. Rossazza, D. Dubois y H. Prade. A hierarchical model of fuzzy classes. En "Fuzzy and Uncertain Object-Oriented Databases: Concepts and Models", vol. 13 de "Advances in Fuzzy Systems - Applications and Theory", págs. 21–61. (1998).

[129] M. A. Roth, H. F. Korth y A. Silberschatz. Extended algebra and calculus for nested relational databases. *ACM Trans. Database Syst.* **13**(4), 389–417 (1988).

[130] M. A. Roth, H. F. Korth y A. Silberschatz. Null values in nested relational databases. *Acta Inf.* **26**(7), 615–642 (1989).

[131] M. A. Roth, H. F. Korth y A. Silberschatz. Addendum to null values in nested relational databases. *Acta Inf.* **28**(6), 607–610 (1991).

[132] E. Ruspini. Imprecision and uncertainty in the entity-relationship model. En H. Prade y C. Negiota, editores, "Fuzzy Logic and Knowledge Engineering", págs. 18–28. Verlag TUV Reheiland (1986).

[133] R. Sacks-Davis y J. Zobel. "Indexing Techniques for Advanced Database Systems". Kluwer Academic Publishers (1997).

[134] H. Sagan. "Space–filling curves". Springer-Verlag (1994).

[135] E. Sanchez. Solution of fuzzy equations with extended operations. *Fuzzy Sets and Systems* **12**(3), 237 – 248 (1984).

[136] H. J. Schek y M. H. Scholl. The relational model with relation-valued attributes. *Inf. Syst.* **11**(2), 137–147 (1986).

[137] B. Scheweizer y A. Sklar. "Probabilistic Metric Spaces". North-Holland (1983).

[138] A. Syropoulos. Mathematics of multisets. En G. Goos, J. Hartmanis y J. van Leeuwen, editores, "Multiset Processing", vol. 2235 de "Lecture Notes in Computer Science", págs. 347–358. Springer Berlin / Heidelberg (2001).

[139] V. Tahami. A conceptual framework for fuzzy query processing - a step towards very intelligent databases systems. *Information Processing and Management* **13**, 289–303 (1997).

[140] The PostgreSQL Global Development Group. PostgreSQL 8.5devel documentation, appendix D. SQL conformance, D.2. unsupported features. http://developer.postgresql.org/pgdocs/postgres/unsupported-features-sql-standard.html (1996–2009).

[141] S. J. P. Todd. The peterlee relational test vehicle – a system overview. *IBM Systems Journal* **15**(4), 285–308 (1976).

[142] E. Trillas. Sobre funciones de negación en la teoría de conjuntos difusos. *Stochastica* **3**(1), 47–59 (1979).

[143] E. Trillas, E. Alsina y J. Terricabras. "Introducción a la lógica borrosa". Ariel Matemática (1995).

[144] J. Ullman. "Principles of Database Systems". Computer Science Press (1982).

[145] M. Umano. FREEDOM-O: A fuzzy database system. En M. Gupta y E. Sanchez, editores, "Fuzzy Information and Decision Processes", Pub. Comp., págs. 339–347. North-Holland, Amsterdam (1982).

[146] M. Umano y S. Fukami. Fuzzy relational algebra for possibility-distribution-fuzzy-relational model of fuzzy data. *Journal of Intelligent Information Systems* **3**, 7–28 (1994).

[147] M. Umano, S. Fukami, M. Mizumoto y K. Tanaka. Retrieval processing fom fuzzy databases. Informe t'ecnico, IEICE, Japan (1980).

[148] N. Van Gyseghem y R. de Caluwe. Fuzzy object-oriented databases: Some behavioral issues. En "Proceedings of EUFIT", págs. 361–364 (1994).

[149] N. Van Gyseghem y R. de Caluwe. Imprecision and uncertainty in the UFO database model. *Journal of the American Society for Information Science* **49**, 236–252 (1998).

[150] R. Vanderberghe y R. de Caluwe. An entity-relationship approach to the modeling of vagueness in databases. En "Proceedings of ECSQAU - Symbolic and Quantitative Approaches to Uncertainty", págs. 338–343 (1991).

[151] B. Verdonck, R. Nijlunsing, F. A. Gerritsen, J. Cheung, D. J. Wever, A. Veldhuizen, S. Devillers y S. Makram-Ebeid. Computer assisted quantitative analysis of deformities of the human spine. *Lecture Notes on Computer Sciences* **1496**, 822–831 (1998).

[152] E. Wong. A statistical approach to incomplete information in database systems. *ACM Trans. Database Syst.* **7**(3), 470–488 (1982).

[153] M. Wong y K. Leung. A fuzzy database-query language. *Information Systems* **15**, 583–590 (1990).

[154] K. Wu, E. J. Otoo y A. Shoshani. Compressing bitmap indexes for faster search operations. En "SSDBM '02: Proceedings of the 14th International Conference on Scientific and Statistical Database Management", págs. 99–108, Washington, DC, USA (2002). IEEE Computer Society.

[155] R. Yager. On the theory of bags. *International Journal of General Systems* **13**, 23–37 (1986).

[156] R. R. YAGER. Cardinality of fuzzy sets via bags. *Mathematical Modelling* **9**(6), 441 – 446 (1987).

[157] A. YAZICI, D. AKSOY Y R. GEORGE. The similarity-based fuzzy object oriented data model. En "Proceedings of IPMU", vol. 3, págs. 1177–1182 (1996).

[158] A. YAZICI Y D. CIBICELI. An index structure for fuzzy databases. En "Proceedings of the Fifth IEEE International Conference on Fuzzy Systems", vol. 2, págs. 1375–1381 vol.2 (1996).

[159] A. YAZICI Y D. CIBICELI. An access structure for similarity-based fuzzy databases. *Information Sciences* **115**(1-4), 137–163 (abr. 1999).

[160] A. YAZICI, R. GEORGE Y D. AKSOY. Design and implementation issues in the fuzzy object-oriented data model. *Journal of Information Sciences* **108**, 241–260 (1998).

[161] A. YAZICI, C. INCE Y M. KOYUNCU. An indexing technique for similarity-based fuzzy object-oriented data model. En H. CHRISTIANSEN, M.-S. HACID, T. ANDREASEN Y H. LARSEN, editores, "Flexible Query Answering Systems", vol. 3055 de "Lecture Notes in Computer Science", págs. 334–347. Springer (2004).

[162] A. YAZICI Y M. KOYUNCU. Fuzzy object-oriented database modeling coupled with fuzzy logic. *Fuzzy Sets and Systems* **89**, 1–26 (1997).

[163] L. ZADEH. Fuzzy sets. *Information and Control* **8**, 338–353 (1965).

[164] L. ZADEH. Similarity relations and fuzzy orderings. *Information Sciences* **3**, 177–200 (1971).

[165] L. ZADEH. The concept of a linguistic variable and its application to approximate reasoning - I. *Information Sciences* **8**, 199–248 (1975).

[166] L. ZADEH. The concept of a linguistic variable and its application to approximate reasoning - II. *Information Sciences* **8**, 301–357 (1975).

[167] L. ZADEH. The concept of a linguistic variable and its application to approximate reasoning - III. *Information Sciences* **9**, 43–80 (1975).

[168] L. ZADEH. A computational approach to fuzzy quantifiers in natural languages. *Computers and Mathematics* **9(1)**, 149–184 (1983).

[169] L. A. ZADEH. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems* **1**(1), 3–28 (ene. 1978).

[170] L. A. ZADEH. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems* **100**(Supplement 1), 9–34 (1999).

[171] M. ZEMANKOVA Y A. KANDEL. "Fuzzy Relational Databases – A Key to Expert Systems". TÜV Rheinland (1984).

[172] M. ZEMANKOVA Y A. KANDEL. Implementing imprecision in information systems. *Information Sciences* **37**, 107–141 (1985).

[173] J. Zhang, E. Lou, L. H. Le, D. L. Hill, J. V. Raso y Y. Wang. Automatic Cobb measurement of scoliosis based on fuzzy hough transform with vertebral shape prior. *Journal of Digital Imaging. Published on-line, doi: 10.1007/s10278-008-9127-y* (2008).

[174] R. Zicari. Incomplete information in object-oriented databases. *SIGMOD Rec.* **19**(3), 5–16 (1990).

[175] A. Zivieli y P. Chen. Entity-relationship modeling and fuzzy databases. En "Proceedings of the Second International Conference on Data Engineering - IEEE", págs. 18–28 (1986).

[176] J. Zobel, A. Moffat y R. Sacks-Davis. An efficient indexing technique for full text databases. En "VLDB '92: Proceedings of the 18th International Conference on Very Large Data Bases", págs. 352–362, San Francisco, CA, USA (1992). Morgan Kaufmann Publishers Inc.

Índice de tablas

Índice de figuras

University of Granada

Department of Computer Science
and Artificial Intelligence

Thesis Abstract

Fuzzy Object-Relational Databases: Model, Architecture and Applications

Author

Carlos D. Barranco Gonzlez

Advisor

Juan Miguel Medina Rodrguez

September, 2009

Contents

Chapter 1

Introduction

This work is the abstract of the thesis titled Bases de Datos Objeto-Relacionales Difusas: Modelo, Arquitectura y Aplicaciones (Fuzzy Object-Relational Databases: Model, Architecture and Applications) written by Carlos D. Barranco under Prof. Juan M. Medina supervision. The thesis is ascribed to the Ph.D. program Diseno, Análisis y Aplicaciones de Sistemas Inteligentes (Design, Analysis and Applications of Intelligent Systems) at the Department of Computer Science and Artificial Intelligence of the University of Granada (Spain).

The thesis main topics are:

1. To propose a fuzzy database model incorporating the characteristics, specially data constructs, of current object-relational databases.
2. To design an object-relational fuzzy database management system and to implement it as an extension of a current object-relational database management system.
3. To propose indexing techniques to improve the performance of query processing in an object-relational fuzzy database management system.
4. To study some applications of the proposal to prove its validity and utility.

This abstract is conceived as a compilation of the most important published papers of the author on the thesis main topics. It is organized as follows. Chapter 2 discusses about the proposition of an object-relational fuzzy database model, its implementation and an example application. Chapters 3 and 4 discuss about the proposals on fuzzy numerical data indexing techniques. Chapter 3 proposes an indexing technique that improves the processing performance of possibility measured flexible conditions, whereas Chapter 4 proposes a novel fuzzy data indexing technique that improves the processing performance of necessity measured flexible conditions. Finally, Chapter 5 summarizes the concluding remarks of this work.

Chapter 2

Towards a fuzzy object-relational database model

2.1 Abstract

This chapter introduces a fuzzy object-relational database model including fuzzy extensions of the basic object-relational databases constructs, the user-defined data types and the collection types. The fuzzy extensions of these constructs focus on two main flexible aspects, a way to flexibly compare complex data types and an extension of collection types allowing partial membership of its elements. Collection operators are also adapted to consider flexibly comparable domains for its elements. Such a fuzzy object-relational database model, and its implementation in a fuzzy object-relational database management system, provides an easy and e ective way to manage a great amount of complex fuzzy data in object-relational databases for emerging fuzzy applications. As a sample of the proposal advantages, an application for dominant color based image retrieval, which is built on an object-relational database management system implementing the proposed fuzzy database model, is introduced.

2.2 Introduction

The introduction of the fuzzy set theory by Prof. Zadeh [46] has provided the database community with a very useful tool for representing imprecise, uncertain and inapplicable data. This approach eases and makes flexible the way in which real world data can be represented and managed in databases. Fuzzy databases are databases able to represent and retrieve fuzzy data and, some of them, also able to process flexible queries including weakly defined conditions. Fuzzy databases are a very convenient data storage and retrieval system for

Original chapter published in: Barranco, C.D., Campaña, J.R., & Medina, J.M. (2008). Towards a Fuzzy Object-Relational Database Model. In Galindo, J. (Ed.), Handbook of Research on Fuzzy Information Processing in Databases, Vol. II, pp. 435-461. Hershey, PA, USA: Information Science Reference.

dealing with classical and non classical problems in which real world data, user perception and/or natural language concepts and descriptions are involved.

Research in the field of fuzzy databases has led to a significant number of fuzzy database models. The aim of these models was always to extend current, widespread, and accepted database models in order to make them suitable for fuzzy data storage and retrieval. Fuzzy models have evolved along with the conventional, i.e. crisp, database models to answer data processing needs. During the apogee of the relational model for databases, several fuzzy relational database models appeared. When the object database model appeared for solving the some of the lacks in the relational one, the fuzzy database research focused on fuzzy object databases.

In recent years, a new kind of databases, object-relational databases, is progressively breaking through in the database mainstream. This kind of database was born in order to solve the relational model lacks by enriching it with some features of object databases. Object-relational databases aim to merge the good qualities of both, the well-known relational model and the object oriented database paradigm, while neutralizing their drawbacks. Currently, object-relational databases are well accepted by database professionals and manufactures, and their most important concepts are gradually being incorporated in recent SQL standards.

As object-relational databases are gradually conquering the database market, it seems natural to work on their extension to allow fuzzy data storage and retrieval. In fact, this kind of databases is very suitable for such an extension, as one of its most important features is its extensibility.

This chapter proposes a model that extends object-relational databases for fuzzy data representation and querying. This model makes the database able to represent a number of new types of fuzzy data that are derived from the extension of the basic object-relational databases constructs, the user defined complex data types and the multi-valued attribute. Additionally, the model also supports all fuzzy data types that were considered in earlier models.

The chapter is organized as follows. Firstly, the chapter includes an introduction on object-relational databases. Secondly, a brief background on fuzzy databases is also presented. Afterwards, the proposed fuzzy object-relational database model is depicted. Next, the basis of an implementation of a Fuzzy Object Relational Database Management System (FORDBMS), which is based on the proposed model, is described. Then, an application of the introduced FORDBMS and some examples of its queries are given. Finally, some concluding remarks and future research directions are proposed.

2.3 Object-Relational databases

Even though nowadays the relational model is the most commonly used model in database theory and practice, as computer applications started to manage large amounts of complex data, it was noticed that this database model was not very suitable for managing complex data.

The relational model faces special difficulties to manage complex data resulting from the composition of other data elements, which is very common in computer aided design, geographical information systems and multimedia applications. A relational database stores the data elements of an entity in a relation,

and relates a complex data element with its components by foreign keys. When the application requires retrieving a complex data element, a number of join predicates has to be performed to gather all its atomic components, which lead to severe performance reductions.

The aim to seamlessly represent and manage complex data in databases led the database community to propose the concept of Object Oriented DBMS (OODBMS). An OODBMS represents data elements as objects. The objects are uniquely identified by an object identifier, or *oid* in short, that substitutes the primary key concept of the relational model. Data is modelled using classes of objects and following the object oriented principles, encapsulation, inheritance and polymorphism. Moreover, the procedures related to the manipulation of the data, which are named *methods* in object oriented context, are bound to the data objects and stored in the database, which meant a great paradigm shift in data modelling conception.

Despite the fact that the OODBMS model has more powerful modelling capabilities than the relational model and it is more appropriate to represent complex data, this kind of DBMS suffer from low performance due to complex or sometimes practically impossible query optimization, and their inability to support large scale systems. Moreover, there was a lack of a formal model and standard Data Definition Language (DDL) and Data Manipulation Language (DML) until the publication by the Object Data Management Group (ODMG) of the ODMG report 1.0 in 1993. The latest revision of the ODMG report is the release 3.0 [11], which was published shortly before the group was disbanded.

OODBMSs were not very well received by the DBMS market due to two main reasons. First of all, they were unsuccessful for technical reasons. While trying to solve the problems observed in complex data handling in relational DBMSs they failed in the same point, the low performance. Secondly, they were not accepted for economical reasons. The transition from the existing systems relying on Relational DBMSs (RDBMS) to OODBMS required new investments for training of personnel (programmers had to learn new DDL and DML languages that were proprietary in most cases), for redefining the database schemas, and for testing and tuning again the database software.

In the mid-90s, a new kind of DBMSs, Object-Relational DBMS (ORDBMS) [37], was proposed. This new concept was conceived to gather the benefits of relational and object oriented DBMS, without suffering from the drawbacks of these models. The aim of the ORDBMS proposal was to enhance RDBMSs to allow a richer data type support, so that complex data types along with their methods could be seamlessly managed. All of this is accomplished without losing desirable features of the traditional RDBMSs and maintaining the compatibility with legacy systems.

2.3.1 User-Defined data types

The ORDBMS concept was first partially integrated into SQL standards in the SQL:1999 [16] standard revision. This standard mandates that a table column can be a *User-Defined Data Type* (UDT). A UDT is a structure of named fields, along with a set of associated methods. The data type of a field can be any of the built-in data types or a UDT. This makes possible very complex UDT definitions. The standard includes some object oriented features, as inheritance and polymorphism, to ease the definition of UDTs and to empower their usage

flexibility.

For instance, a UDT for gathering data describing current weather conditions could be defined. This UDT, which can be named WEATHER, can be composed of the following fields:

- Issuer: This field contains the name of the organization that publishes the weather conditions as a VARCHAR data type.
- Temperature: This field contains, as a value of NUMBER data type, the current temperature.
- Cloudiness: This field describes the current cloud coverage as a value of the set $\{clear, sunny, cloudy, overcast\}$.
- Precipitation: The field value describes the current precipitation conditions as a value of the UDT PRECIPITATION, or a null vale in case of no precipitation. The UDT PRECIPITATION is composed by the following fields:

  Intensity: This field describes the intensity of the phenomenon as a value of the set $\{light, moderate, heavy\}$.

  Kind: This field describes the kind of precipitation. The value for this field is taken from the set $\{rain, hail, snow\}$.

2.3.2 Collection data types

A *collection data type* is a user defined data type able to hold multiple values of another type, which is called *base* type. SQL:1999 introduced the first collection type, an array of values where each element is accessed by a numerical index. This data type construct is very limited as each array type has a fixed maximum cardinality limit, and the standard does not allow the definition of an array data type of another array data type.

The collection types are generally known as *nested tables* but the arrays do not behave like real tables. Later on, a new revision of SQL standard was published as SQL:2003 [17]. This revision maintains UDTs and arrays, and includes a new complex data type called *multiset*. This data type is able to contain an unordered collection of values, all of them of the same data type, including UDTs. It is possible to define a multiset of multisets, and a multiset does not have a fixed cardinality limit. This new type construct fits to the nested table concept, so it has become the collection type of reference. This relegates arrays to very particular applications.

For instance, a collection type of the WEATHER UDT defined earlier can be used to store in a field the current weather conditions of a city issued by di erent organizations. An example table with a column of this type is shown in Table 2.1.

2.4 Background on fuzzy databases

A *fuzzy database* [33, 22] can be defined as a database containing *imperfect data* which is generally modelled as fuzzy sets. The term imperfect data encompasses data that are uncertain, imprecise, vague, or inapplicable.

City (VARCHAR)	WeatherConditions (MULTISET OF WHEATHER)				
	Issuer	Temperature	Cloudiness	Precipitation	
				Intensity	Kind
London	MetO ce	$10°C$	overcast	heavy	rain
	acme	$8°C$	overcast	null	
Madrid	INM	$18°C$	cloudy	null	
	WMO	$18°C$	cloudy	light	rain
	acme	$22°C$	sunny	null	
Paris	MeteoFrance	$3°C$	cloudy	light	snow

Table 2.1: Weather conditions of world cities

The work [8] includes an excellent discussion about the meaning of the characteristics of imperfect data. We refer the reader to this work for more details and references. The following, which is based on and extends the previous discussion, is a brief definition of the characteristics of imperfect data:

- Uncertain data: Data that is not totally trustworthy, for which there is available an estimation of its reliability. For instance, if we ask the age of *Prudence* to a neighbor, a co-worker, a friend and her mother, and each one reply a different value, we can assign a reliability degree according to the strength of the relationship between *Prudence* and the asked person.
- Imprecise data: When the data is not available in its maximum finest granularity, we have some an approximation but not a precise value. In this case, it could be available a set or a range of values, among which the actual precise value is unknown. For instance, we know that *Prudence* is between 25 and 28 years old, but we can not give an exact age for her.
- Vague data: This kind of data is defined by a gradual predicate. Usually, vague data corresponds to linguistic terms of natural language. For instance, we could know that *Prudence* is middle-aged. The linguistic term middle-aged corresponds to a gradual predicate that is completely incompatible for age values below 30 and above 60, and completely compatible for ages between 35 and 55. The predicate is partially compatible for ages between 30 and 35, and ages between 55 and 60, where the compatibility degree gradually ascends in the former case, and gradually descends in the latter case.
- Inapplicable data: There may be some entities for which a piece of data relating to one of its properties can not be acquired due to a lack of the property. For instance, if *Prudence* is not married the data related to his spouse is inapplicable.

Fuzzy logic and fuzzy set theory have proven their great ability for modelling this kind of data.

Even though the great majority of proposals for modelling and managing imperfect data take advantage of the fuzzy paradigm, there are some early non-fuzzy proposals. The proposal in [13], which introduces the idea of null values for representing unknown or inapplicable data, is probably the best known approach. Another widely known non-fuzzy approaches make use of statistical inference [42] and probability distributions [1].

2.4.1 Fuzzy relational databases

Most of the initial fuzzy database proposals aim to extend the relational model in order to make it suitable for handling imperfect data. These approaches take advantage of the convenient modelling capabilities of fuzzy sets and logic in several ways by generalizing different aspects of the relational model to make them more flexible:

- Relations including partially belonging tuples: This is the basic fuzzy database model. In this model the set of tuples of a relation is replaced by a fuzzy set. Each tuple has a membership degree to its relation, so

the tuple can partially belong to it. This idea has been adopted in [31]. If a flexible relational operator is included in a query on vague data, the query result could contain tuples partially satisfying the query condition. The degree to which a tuple satisfies the query conditions can be naturally used to measure its membership to the result set. This approach has been included in [9, 34, 40, 49]. An extended relational algebra for operating with this kind of fuzzy relations is proposed in [40]. In [7] relation oriented operators for nested queries are extended so they can be employed on fuzzy relations.

- Imprecise attribute values: The proposal [9] introduces a way to model imprecise attribute values. Every attribute value could be a subset of the attribute domain. The subset has a disjunctive meaning, thus only one of the subset members is the actual value for the attribute.

- Vague attribute values: Once again, the previous proposal makes possible modelling vague data by taking advantage of fuzzy numbers to represent vague numerical data. Furthermore, possibility distributions defined on basic attributes are used to model arbitrary vague and imprecise data in the works [38, 34, 40]. The first one also studies the definition of flexible queries using vague predicates in a relational database. The representation of unknown attribute values is made possible by means of a possibility distribution where every domain element is totally possible. This possibility distribution is defined as shown in eq. 2.1 , where D is the attribute domain.

$$\{1/d : d \in D\} \tag{2.1}$$

- Inapplicable attribute values: The proposals [34, 40] also include approaches for inapplicable data handling. The former introduces a special value, denoted as e, which is added to each attribute domain. The e value represents the inapplicability of the attribute and can be part of possibility distributions representing an attribute value. The inclusion of this special value in possibility distributions makes possible the definition of attribute values that are partially inapplicable. An instance of such a possibility distribution is the one shown in eq. 2.2. The representation of total ignorance about the applicability of an attribute and the value of the attribute (in case of applicability) is also possible. This case is described by the possibility distribution shown in eq. 2.3. The latter proposal studies deeper these extreme cases and introduces the special values *unknown* (unknown attribute value), $undefined$ (inapplicable attribute) and *null* (total ignorance) to represent them.

$$\{0.5/e, 0.7/v_1, 0.3/v_2, ...\}, v_i \in D \tag{2.2}$$

$$\{1/d : d \in D \cup \{e\}\} \tag{2.3}$$

- Flexible equivalence relations: In

- Buckles1982 the idea of making use of similarity relations [47] in order to soften the classical equivalence relations in databases is introduced. Every query must specify a similarity threshold which determines the indistinguishability of domain members for this query. A similar proposal, but based on proximity relations was introduced in [35]. When a database includes imprecise and vague attribute values that are modelled using possibility distributions, the equivalence operator is substituted by possibility and necessity measure based equality operators as proposed in [34].

- Flexible relational operators: The proposal [49] introduces the idea of fuzzy relational operators. These fuzzy extensions of the classical relational operators make flexible the order relation between possibility distributions of numbers and scalars. Later proposals, as [30], adapt the concept of fuzzy relational operators for fuzzy numbers. These new adapted operators soften the classical order relations of numbers, where the result can be only true or false values, by returning a degree in which the compared fuzzy numbers are related.

2.4.2 Generalized model for fuzzy relational databases

The work [30] proposes a fuzzy relational database model, *A Generalized Model of Fuzzy Relational Databases* (GEFRED), which aims to gather the main di erent aspects of the relational model that has been made flexible in previous proposals. The integration of these proposals in the same framework results in a generalized model with a wide ability to represent and handle fuzzy data based on a fuzzy extended relational database model.

The model introduces the concept of *Generalized Fuzzy Domain*. A generalized fuzzy domain D_G is defined as shown in eq. 2.4, where D is the basic domain which is being generalized, $P(D)$ is the fuzzy powerset composed of all possibility distributions that can be defined on D, and $NULL$ is a special value meaning total ignorance in the sense of [40] that has been described previously.

$$D_G \subset P(D) \cup \{NULL\} \tag{2.4}$$

A wide variety of fuzzy data, and of course non-fuzzy data, can be represented by using a generalized fuzzy domain. Particularly, any arbitrary possibility distribution can be represented, and hence imprecise, vague, unknown and undefined values can be handled. For each generalized fuzzy domain, a set of linguistic labels, each one representing a fixed possibility distribution, can be defined to ease data representation. Every fuzzy generalized domain have at least the linguistic labels $UNKNOWN$ defined as the possibility distribution $\{1/d : d \in D\}$ for representing unknown values, and $UNDEFINED$ corresponding to the possibility distribution $\{0/d : d \in D\}$ which is used to represent inapplicable values. The special value $NULL$ can be informally described for illustrative purposes by the hypothetical possibility distribution $\{1/UNKNOWN, 1/UNDEFINED\}$. This possibility distribution indicates that any case, the value is unknown and the value is not applicable, is totally possible.

A *Generalized Fuzzy Relation* can be built on top of generalized fuzzy domains. The model defines a generalized fuzzy relation as $R_{FG} = (\mathcal{H}, \mathcal{B})$, where $\mathcal{H}$ is the head and $\mathcal{B}$ the body of the generalized relation.

The head of a relation is defined as eq. 2.5 shows, where, each attribute A_{G_j} has an associated fuzzy domain D_{G_j} $(j = 1, 2, \ldots, n)$, and $C_{A_{G_j}}$ is a compatibility attribute whose domain is $[0, 1]$.

$$\mathcal{H} = \left\{ \left(A_{G_1} : D_{G_1}[, C_{A_{G_1}}] \right), \left(A_{G_2} : D_{G_2}[, C_{A_{G_2}}] \right), \ldots, \left(A_{G_n} : D_{G_n}[, C_{A_{G_n}}] \right) \right\} \tag{2.5}$$

The body of a relation is defined as eq. 2.6 shows, where $i = (1, 2, \ldots, m)$, m is the number of tuples belonging to the relation, $d_{ij} \in D_{G_j}$ is the value of the j-th attribute of the i-th tuple, and c_{ij} is the compatibility degree of the j-th attribute of the i-th tuple.

$$\mathcal{B} = \left\{ \left(A_{G_1} : d_{i1}[, c_{A_{i1}}] \right), \left(A_{G_2} : d_{i2}[, c_{A_{i2}}] \right), ..., \left(A_{G_n} : d_{in}[, c_{A_{in}}] \right) \right\} \tag{2.6}$$

The square brackets in the previous formulae denote an optional element. The compatibility degrees are optional elements because they are not allowed in the base relations of the database, as the model does not consider partial membership for the tuples of base relations. The compatibility degrees are exclusively used for relations representing the result of a query, where their value represents the compatibility of the attribute value with respect to the query conditions.

In order to manipulate the data in a fuzzy database, the model defines the Generalized Fuzzy Relational Algebra with union, intersection, di erence, Cartesian product, projection, and join extended operators for generalized fuzzy relations. The selection and join operations base their criteria on fuzzy compatibility measures for fuzzy data.

Further work on the GEFRED model has addressed the fuzzy domain calculus [19] and the division operator on fuzzy relations [20].

2.4.3 Object oriented fuzzy databases

With the appearance of the first object oriented databases in the early 90s, research on fuzzy databases was focused on the incorporation of fuzziness into all of their concepts. Fuzziness has been considered for attribute values, for the behavior (methods) of objects, for the structure definition (i.e. set of attributes) of classes and objects, for the relationships between objects and their classes, and for the inheritance relationships between classes as well. This chapter only focuses on aspects related to the flexible comparison of objects, so we refer the reader to the book [10] for further study of object oriented fuzzy databases. This is an excellent compilation of early work on fuzzy object oriented databases.

2.5 A fuzzy object-relational database model

The previous aims in the fuzzy database research field were the extension of well studied and widespread database models of the moment, i.e. based on the relational and object oriented database models, in order to make them able to represent and handle fuzzy data. This work, as a natural continuation of the aims in fuzzy database research, proposes an extension of Object-Relational Databases. This proposal takes advantage of the extension mechanisms of ORDBMSs

to make them able to represent, store, and retrieve fuzzy data, even of complex data types.

Even though object-relational databases adopt an object oriented model for data type definition and integrate it into the relational framework, the purpose of this work is not to incorporate fuzziness in the definition and hierarchy of data types. Most fuzzy object oriented model proposals aim to soften the relationships between objects and classes, and subclasses and superclasses. The proposal of this chapter, in contrast, aims to provide a way to flexibly compare complex fuzzy data.

This section proposes a model to create fuzzy extensions of the basic data type constructs of ORDBMSs, the UDTs and the collection types, in order to make flexible their equivalence relations, and to allow a seamlessly integration and representation of fuzzy data in the object relational framework. For the chapter purposes, the proposed model is named SDSDM as an acronym of Soft Data Server Database Model.

2.5.1 Flexibly comparable types

As it has been stated previously, the axis of the proposal is the aim of making flexible the equivalence relations of complex data elements. This section proposes a fuzzy extension of UDTs which allows a flexible comparison of their objects.

In the previous section, which is devoted to a brief fuzzy database background, the idea of substituting an equivalence relation by a similarity relation [9] or by a proximity relation [35] has been introduced. For chapter purposes, a proximity or similarity relation substituting an equivalence relation is named a *flexible equivalence relation*.

The application of this idea requires the definition a flexible equivalence relation for each domain whose elements are going to be flexibly compared. These flexible equivalence relations are modelled by a fuzzy relation, which is typically described by a membership function.

Relational databases and their fuzzy extensions do not provide a way to attach a user defined flexible equivalence relation to a domain. To fill this gap, the implementations of the early fuzzy extensions stored the definition of these relations as discrete membership functions in tables of the data dictionary (or meta-database). This way, users are unable to define any arbitrary flexible equivalence relation, specially a continuous one or a flexible equivalence relation for a domain with high cardinality.

As the underlying database model of this proposal is an object-relational one, it provides a seamless and natural way to attach the definition of the flexible equivalence relation of a domain (UDTs in object-relational context) along with the structure and behavior definition of the domain. SDSDM proposes that the flexible equivalence relation of each flexibly comparable UDT should be specified as part of its behavior.

We define a *Flexibly Comparable Type* (FCT) as a UDT which encloses, in its own definition, its flexible equivalence relation specification as one of its methods. In other words, a FCT is a class whose objects have a common method whose implementation defines the membership function of the flexible equivalence relation associated with the UDT. Such a method is named *feq* (Fuzzy Equal) and its functional definition is shown in eq. 2.7, where D is the

	overcast	cloudy	sunny
clear	0	0	0.75
sunny	0	0.25	
cloudy	0.75		

Table 2.2: Flexible equivalence relation on the cloudiness domain

domain defined as a FCT, and μ_{ER_D} is the membership function of the flexible equivalence relation defined for the FCT.

$$feq(a,b) = \mu_{ER_D}(a,b); a,b \in D \tag{2.7}$$

The domains (UDTs and built-in types) of a FORDBMS implementing the proposed model are divided into two separate groups, those implementing the *feq* method, and those not implementing it. The domains of the former type are named FTC types, and the domains of the latter group are named Non FTC (NFTC) types.

The strength of this proposal is the natural way in which flexible equivalence relations are bound to their domains, at database model level, and the freedom in which these equivalence relations can be defined, virtually any flexible equivalence relation which could be implemented as a method.

A exibly comparable type for a scalar discrete domain

One of the most immediate and simple FCT is the one designed for modelling a scalar domain on which a flexible equivalence relation is defined.

Each object of this FCT is able to represent one scalar, or less formally a *label*, of the domain. The implementation of the *feq* method of the class returns a value corresponding to the application of the membership function of the flexible equivalence relation defined on the domain, which has been specified by the FCT designer.

For instance, a FCT of this kind is useful to allow the flexible comparison of *cloudiness* attribute domain values, defined in the previous example. This domain is defined as $\{clear, sunny, cloudy, overcast\}$. A flexible equivalence relation, which in this case is a proximity relation, for this domain could be the one shown in Table 2.2.

The designer of the FCT can implement it to allow each object to represent one of the scalars of the domain, and implement the *feq* method for returning a value according to the previously defined flexible equivalence relation.

A exibly comparable type for fuzzy numbers

Another good example of a possible FCT is the one designed to represent and flexibly compare fuzzy numbers.

This FCT can be designed to model a domain whose elements are fuzzy numbers that are defined as trapezoidal possibility distributions. A trapezoidal possibility distribution is noted $[\alpha, \beta, \gamma, \delta]$ and it is defined by the membership function shown in eq. 2.8.

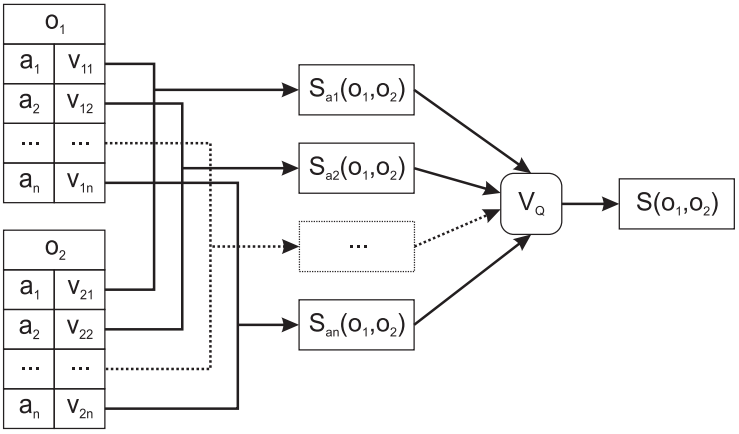

Figure 2.1: Sketch of the process to calculate the resemblance of two objects as proposed in CORM

$$\mu_{[\alpha,\beta,\gamma,\delta]}(x) = \begin{cases} 0 & x \leq \alpha \text{ or } x \geq \delta \\ \frac{x-\alpha}{\beta-\alpha} & \alpha < x < \beta \\ 1 & \beta \leq x \leq \gamma \\ \frac{\delta-x}{\delta-\gamma} & \gamma < x < \delta \end{cases} , \alpha \leq \beta \leq \gamma \leq \delta \tag{2.8}$$

The proposed FCT includes an attribute of the numerical built-in type for each of the four values defining a trapezoidal possibility distribution.

The resemblance of two fuzzy numbers could be calculated by means of their possibility measure. Therefore, the *feq* method of this FCT can be implemented as shown in eq. 2.9, where a and b are two fuzzy numbers, and $\otimes$ a t-norm.

$$feq(a,b) = \sup_x(\mu_a(x) \otimes \mu_b(x)) \tag{2.9}$$

A generic flexible equivalence relation for user defined complex flexibly comparable types

Even though a flexible equivalence relation of a user defined FCT can be any of the designed by the user to meet particular requirements, in this section a generic purpose flexible equivalence relation for user defined complex FCT is introduced.

The proposed flexible equivalence relation is a fuzzy resemblance measure for complex objects, whose attributes either are of FCT or NFCT types. For the sake of simplicity, the proposed flexible equivalence relation is named *Complex Object Resemblance Measure* (CORM).

The original idea of CORM was first proposed in [29] for OODBMS context, and later adapted to the object-relational paradigm in [15].

The way CORM determines the resemblance between two objects of the same type is sketched in Fig. 2.1, where o_1 and o_2 are objects of the same FCT, $a_1, a_2, \ldots, a_n$ are attributes of these objects, and v_{ij} is the value of the attribute a_j for the object o_i. This procedure is divided into the following steps:

1. A resemblance degree between the pair of values of each attribute of the compared object is calculated. For the i-th attribute, the function $S_{ai}(o_1, o_2)$, which is detailed later, is used.

2. The resemblance degrees of each pair of attribute values are aggregated. For this purpose, the V_Q aggregator, which is described later, is employed.

Attribute resemblance measure One of the two main components of CORM is the function to calculate the resemblance degree between a pair of values, of the same attribute, from two objects of the same FCT. This resemblance measure is based on the flexible equivalence relation for the attributes whose type is a FCT, and in the classical built-in equivalence for the attributes whose type is a NFCT.

This resemblance measure is described by the function shown in eq. 2.10, where o_1 and o_2 are two objects of the same class, $o_1.a_i$ and $o_2.a_i$ are the values of the i-th attribute of the objects o_1 and o_2, and C_{a_i} is the class or type of the i-th attribute of these objects, $feq_{C_{a_i}}$ is the membership function of the flexible equivalence relation for the type C_{a_i}, $\delta(a, b)$ and is the Kronecker delta function defined as eq. 2.11 shows.

$$S_{a_i}(o_1, o_2) = \begin{cases} feq_{C_{a_i}}(o_1.a_i, o_2.a_i) & \text{if } C_{a_i} \text{ is a FCT} \\ \delta(o_1.a_i, o_2.a_i) & \text{if } C_{a_i} \text{ is a NFCT} \end{cases} \tag{2.10}$$

$$\delta(a, b) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{if } a \neq b \end{cases} \tag{2.11}$$

Attribute resemblance aggregation Once the resemblance degree between the pair of values of both compared objects is calculated for each attribute, these degrees are aggregated to produce a global resemblance value for the objects.

In CORM, this aggregation is calculated by obtaining the degree of truth of the vague sentence *"Most of the important attributes of the class present similar values in both objects"*. This vague sentence involves linguistic quantifiers [48].

CORM allows the FCT designer to indicate a degree of relevance for each attribute, so the resemblance measure can be adapted to the type specificity. This makes CORM to focus more on the relevant attributes of the type.

The CORM proposal is based on the Vilas' approach [41] to calculate the degree of truth of the previous sentence. This approach is based on the concept of *coherent family of quantifiers* to interpret the fuzzy quantifier *"Most"*. In this approach, this quantifier is interpreted as a weighted combination of the degrees of truth of the existential, $\exists$, and universal, $\forall$, quantifiers.

The degree of truth of the existential quantifier corresponds to degree of truth of the sentence *"There exists an important attribute which presents similar values in both objects"*. This degree is calculated as shown in eq. 2.12. In this equation A is the set of the attributes of the compared objects data type. P is the fuzzy set of relevant attributes whose membership function is defined as eq. 2.13 shows, where p_i is the relevance weight of the attribute given by the type designer. Finally, R is the fuzzy set of the attributes whose values for both objects resemble. Equation 2.14 shows the membership function of R.

$$\exists a \in A, a \in P \wedge a \in R = \sup_{a \in A}(\mu_P(a) \otimes \mu_R(a)) \tag{2.12}$$

$$\mu_P(a_i) = p_i \ , a_i \in A \tag{2.13}$$

$$\mu_R(a) = S_a(o_1, o_2), a \in A \tag{2.14}$$

Likewise, the degree of truth of the universal quantified sentence *"All important attributes present similar values in both objects"* is calculated as shown in eq. 2.15, where the implication has been substituted as shown in eq. 2.16, and $\oplus$ represents a t-conorm.

$$\forall a \in A, a \in P \rightarrow a \in R = \inf_{a \in A}((1 - \mu_P(a)) \oplus \mu_R(a)) \tag{2.15}$$

$$P \rightarrow R = \neg P \vee R \tag{2.16}$$

As it has been stated before, the Vilas's approach combines the previous degrees of truth to obtain the degree of truth of the fuzzy quantifier $Most$. This degree of truth is calculated by the function defined in eq. 2.17, where, γ_{Most} is a factor in the interval $[0, 1]$ for modelling the fuzzy quantifier $Most$. The value of this factor for the existential quantifier, $\gamma_{\exists}$, is 1, and for the universal quantifier, $\gamma_{\forall}$, is 0.

$$\begin{array}{c} V_{Most}(P/R) = \gamma_{Most} \sup_{a \in A}(\mu_P(a) \otimes \mu_R(a)) + \\ (1 - \gamma_{Most}) \inf_{a \in A}(\mu_P(a) \oplus (1 - \mu_R(a))) \end{array} \tag{2.17}$$

The result of the previous function corresponds to the resemblance degree of a pair of compared objects. More formally, the resemblance degree of two compared objects is defined as eq. 2.18 shows.

$$S(o_1, o_2) = V_{Most}(P/R) \tag{2.18}$$

2.5.2 Fuzzy collections

Previously, it has been stated that the proposed SDSDM model aims to create fuzzy extensions of the basic data type constructs of ORDBMSs. In the previous section, a fuzzy extension of UDTs has been proposed, so the objects of the extended UDTs could be flexibly compared. In this section a fuzzy extension of collection types is proposed. The proposed extension aims to add flexibility to the collection types by allowing partial membership of their elements.

If a collection is extended to allow partial membership, there is not a direct transposition from it to a fuzzy set as it might seem at first glance. A fuzzy set, in its classical sense, is defined on a classical domain, where the classical equality can be applied. This is not the general case of the fuzzy sets defined in the SDSDM model. The base type of a fuzzy extension of a collection type can be a FCT, where the classical equality is substituted by a flexible equivalence relation. Because of the previous reason, we consider an extension of a collection type as a fuzzy set whose associated operators are modified to take into account the possible flexibility of the equivalence relation of the base type of the collection. Such an extension is named a *Fuzzy Collection*.

Basic operators on fuzzy collections

A useful concept for the redefinition of the basic fuzzy collections operators is the similarity driven extension of a fuzzy collection. We define a similarity driven extension of a fuzzy collection A as a fuzzy set A^S containing the elements of the original fuzzy collection A and the domain elements which are similar to the elements of A in any degree greater than zero. Formally, A^S is defined as shown in eq. 2.19, where, S is the flexible equivalence relation defined for the base type of the fuzzy collection A.

$$d \in A^S \iff \exists d' \in A, (d, d') \in S \tag{2.19}$$

The membership function of a similarity driven extension of a fuzzy collection is defined in eq. 2.20, where, D is the underlying domain (i.e. the base type) on which A is defined.

$$\mu_{A^S}(d) = \sup_{d' \in D} \mu_A(d') \otimes \mu_S(d, d'), d \in D \tag{2.20}$$

In case of a fuzzy collection whose elements are of a NFCT, the flexible equivalence relation is defined as the classical identity. In this case a fuzzy collection is similar to a fuzzy set in the classical sense. Thus, a fuzzy set is a particular case of a fuzzy collection.

With the help of the previous concept, the intersection of two fuzzy collections is a fuzzy collection defined as shown in eq. 2.21. This operation is noted $\cap_S$ in order to distinguish it from the fuzzy set intersection. This definition can be reformulated in a set oriented view as eq. 2.22 shows.

$$d \in A \cap_S B \iff d \in A \wedge d \in B^S \vee d \in B \wedge d \in A^S \tag{2.21}$$

$$A \cap_S B = A \cap B^S \cup B \cap A^S \tag{2.22}$$

The previous definition takes into account the partial equivalence of FCT objects by correlating the elements of each intersected set with the similarity driven extension of its counterpart. The corresponding membership function is shown in eq. 2.23.

$$\mu_{A \cap_S B}(d) = \mu_A(d) \otimes \mu_{B^S}(d) \oplus \mu_B(d) \otimes \mu_{A^S}(d), d \in D \tag{2.23}$$

The union and complement operations of a fuzzy collection remain as they are defined for classical fuzzy sets. These operators do not have to correlate the members of their operands (i.e. fuzzy collections), and therefore the flexible equivalence relation is not needed to perform them.

The membership function of the fuzzy collection resulting from the union of two fuzzy collections A and B is shown in eq. 2.24.

$$\mu_{A \cup B}(d) = \mu_A(d) \oplus \mu_B(d), d \in D \tag{2.24}$$

The same way, the membership function of the complement of a fuzzy collection A is the one shown in eq. 2.25.

$$\mu_{\overline{A}}(d) = 1 - \mu_A(d), d \in D \tag{2.25}$$

Fuzzy collection inclusion

One of the most common relational operators used in queries in an ORDBMS when they involve a collection type field is the inclusion operator. This operator, in its classical version, returns a true value when all the elements of the left operand are elements of the right operand, and a false value otherwise. Likewise, the classical inclusion operator for fuzzy sets returns a Boolean value indicating whether the fuzzy set corresponding to the left operand is included in the fuzzy set corresponding to the right operand. Beyond the classical conception of the inclusion operator for fuzzy sets, there is a large variety of proposals which makes flexible this operator by defining a degree of inclusion instead of a Boolean value. One of the reference works on making flexible the inclusion operator for fuzzy set is [36].

A direct adoption for fuzzy collections of the classical fuzzy version of the inclusion operator is not coherent. As it has been argued in the previous subsection, an equivalent operator for fuzzy collections should take into account the peculiarities of the allowed domains in the SDSDM model, as the model allows the definition of domains with a flexible equivalence relation substituting the classical equality.

In order to calculate the degree to which a fuzzy collection A is included in a fuzzy collection B, the SDSDM model makes use of the *Resemblance Driven Inclusion Degree*. This degree is originally proposed in [29] and later adopted for its usage in a FORDBMS context in [15].

If the original function is adapted to the notation proposed in this work by means of the similarity driven extension of a fuzzy set, the resemblance driven inclusion degree of a fuzzy collection A in a fuzzy collection B is calculated taking into account the reasoning shown in eq. 2.26. This operator is noted as $\subseteq_S$ in the rest of the paper.

$$A \subseteq_S B \iff \forall d, d \in A \rightarrow d \in B^S \tag{2.26}$$

The functional expression to calculate the degree of truth of the previous reasoning is shown in eq. 2.27, where I is a fuzzy implication operator.

$$\Theta_S(B|A) = \inf_{d \in D} I(\mu_A(d), \mu_{B^S}(d)) \tag{2.27}$$

A resemblance measure for fuzzy collections

Another possible relational operator for collection types is the equality operator. This operator is applied in the classical context to check if two collections contain the same elements. If so, a true value is returned, otherwise a false value is returned. There is also a version of this operator for fuzzy sets. This version continues returning a true value if the two compared fuzzy sets contain the same elements and a false value otherwise.

Once again, a direct adoption for fuzzy collections of the classical fuzzy sets equality operator is not possible. As stated before, this operator was conceived for fuzzy sets defined on an underlying domain where the classical equality holds. Thus, a fuzzy collection equality operator that takes into account the flexible equivalence relation defined on the base type of the collection is needed.

Besides, the equality operator must take into account the semantics of the fuzzy collections. Like fuzzy sets, the semantics of the membership of an element

into a fuzzy collection a ects the way the resemblance degree is determined. On the one hand, the basic set oriented view of a fuzzy set traditionally calls for a conjunctive semantics. If the value of an attribute is a fuzzy set with conjunctive semantics, all the fuzzy set elements are meant to be actual values of the values of the attribute. On the other hand, fuzzy sets can be used to model exclusive expressions (for instance, possibility distributions) where the semantics is clearly disjunctive. An attribute, whose value is a possibility distribution represented as a fuzzy set with disjunctive semantics, is meant to actually have only one value. The actual value of the field is unknown, but it must be an element of the possibility distribution.

A resemblance measure for fuzzy collections with conjunctive semantics The SDSDM model adopts the *Generalized Resemblance Degree between Fuzzy Sets* proposed in [29], which once more was adapted to the FORDBMS context in [15], for measuring the resemblance of two fuzzy collections with conjunctive semantics.

This operator is devised relying on the concept of double inclusion. Two sets A and B are equal if, and only if, A is included in B and B is included in A, or more formally as shown in eq. 2.28.

$$A = B \iff A \subseteq B \wedge B \subseteq A \tag{2.28}$$

The previous expression makes use of the classical inclusion operator. When A and B are two fuzzy collections the *Resemblance Driven Inclusion Degree* depicted in the previous section is applied. Thus, the flexible equivalence relation defined for the base types of the fuzzy collections is taken into account. This resemblance measure for fuzzy collections with conjunctive semantics, noted as $\stackrel{\wedge}{=}_S$, is formally defined as eq. 2.29 shows.

$$A \stackrel{\wedge}{=}_S B \iff A \subseteq_S B \wedge B \subseteq_S A \tag{2.29}$$

The degree of truth for the previous expression is calculated as shown in eq. 2.30.

$$\beth_S(A, B) = \Theta_S(B|A) \otimes \Theta_S(A|B) \tag{2.30}$$

A resemblance measure for fuzzy collections with disjunctive semantics In order to determine the resemblance of two fuzzy collections with disjunctive semantics, the SDSDM model adopts the classical possibility compatibility measure of possibility distributions. It is adapted so the flexible equivalence relation of a FCT base type is taken into account.

The classical possibility compatibility measure of two possibility distributions A and B corresponds to the truth value of the expression shown in eq. 2.31.

$$\exists d, d \in A \wedge d \in B \iff \exists d, d \in A \cap B \iff A \cap B \neq \emptyset \tag{2.31}$$

As the intersection operator for fuzzy sets has been defined previously, the second expression in the previous sentence is used to adapt the classical possibility measure to the fuzzy collections peculiarities. The resemblance measure for

fuzzy collections with disjunctive meaning, noted as $\overset{\vee}{=}_S$, is defined as eq. 2.32 shows.

$$A \overset{\vee}{=}_S B \iff \exists d \in A \cap_S B \tag{2.32}$$

Therefore, the functional definition of the resemblance measure results as shown in eq. 2.33.

$$\Pi(B|A) = \sup_{d \in D} \mu_{A \cap_S B}(d) \tag{2.33}$$

Fuzzy collections as exibly comparable types In the previous subsections, a way to calculate the resemblance of a fuzzy collection, either with conjunctive or disjunctive semantics, is introduced. These measures provide a way to flexibly compare fuzzy collections depending on its semantics.

The only requirement of a database type for being considered a FCT, is the availability of a way to calculate the degree to which two values of the type are equivalent. If the previously introduced resemblance measures are used for this purpose, a fuzzy collection type can be considered a FCT, and therefore it can be nested in other complex FCT types as any other FCT.

2.5.3 User defined linguistic labels

The usage of linguistic labels is a common practice in fuzzy database querying. Actually, linguistic labels mean for fuzzy databases what is commonly known as symbolic constants. Linguistic labels are an easy way to specify common domain values in queries. They result especially convenient for those values of FCT which are large fuzzy collections or complex objects, as they mean a shortcut for these values. In this case, the usage of a linguistic label means an increment of query clarity because it avoids complex constant expression, and of course a saving of query writing e ort.

In the SDSDM model, a user defined set of linguistic labels can be attached to every FTC. The relation between a FCT and its linguistic labels, together with the value which represents the linguistic label, is maintained in a table of the data dictionary (or meta-database).

Special linguistic labels

In the SDSDM model, every FCT has a predefined set of special linguistic labels. These linguistic labels represent special values which are used to model the ignorance of a field value, the $UNKNOWN$ label, the inapplicability of a field, the $UNDEFINED$ label, and the ignorance about the applicability of the field and its value if the field were applicable, the $NULL$ label.

These special linguistic labels are directly taken from the GEFRED model [30]. The way the special linguistic labels are defined in the SDSDM model di er slightly from GEFRED model. In the latter, the special linguistic labels are defined as possibility distributions, which are the GEFRED model basic constructs. As SDSDM includes some more type constructs such as conjunctive fuzzy collections and flexible comparable types that are not possibility distribution oriented, the special linguistic labels are defined in terms of their resemblance values when they are compared to other domain values.

When the resemblance of the linguistic label $UNKNOWN$ to another domain value is computed, the expression shown in eq. 2.34 is applied.

$$feq(d, UNKNOWN) = 1, \forall d \in D \tag{2.34}$$

If the special linguistic label $UNDEFINED$ is compared to another domain value or the label $UNKNOWN$, the resemblance degree is computed by applying the expression shown in eq. 2.35.

$$feq(d, UNDEFINED) = 0, \forall d \in D \cap \{UNKNOWN\} \tag{2.35}$$

When the special linguistic label $NULL$ is involved in a resemblance comparison with another domain value or the special linguistic labels $UNKNOWN$ or $UNDEFINED$, the resemblance degree is computed as shown in eq. 2.36, where *null* is the native null value of object-relational databases.

$$\begin{array}{l} feq(d, NULL) = null, \\ \forall d \in D \cap \{UNKNOWN, UNDEFINED\} \end{array} \tag{2.36}$$

The reader should not confuse the previous *null* value, which is named native null for the sake of clearness, with the special linguistic label $NULL$. The former is a value of the three-valued logic used in relational and object relational databases, where null means ignorance of the truth value of a proposition. This is used by the SDSDM model to indicate the ignorance about a resemblance degree value, because it is impossible to be determined with the current data in the database. The latter, the special linguistic label $NULL$, as stated earlier, represents the case of total ignorance about the applicability of a field, and the ignorance of its value if it were applicable.

2.5.4 Expression capabilities of SDSDM

With the previously defined elements, the SDSDM model reaches the expressive capabilities of the preceding fuzzy database models.

The GEFRED model stands out for gathering in one model the contributions of the most important previous proposals. As a result, GEFRED is able to manage all kinds of fuzzy and non-fuzzy data which are supported by most of the previous models. These kinds of fuzzy and non-fuzzy data are:

1. A single scalar. This kind of values is represented as a discrete possibility distribution including one element. The GEFRED model supports the definition of a fuzzy equivalence relation between the elements of the scalar domain.

2. A single number. This kind of values is represented as a built-in numerical type or as a discrete possibility distribution with only one element.

3. Possibility distributions on scalar domains. This kind of values is represented as discrete fuzzy sets with disjunctive semantics. The GEFRED model supports the definition of a fuzzy equivalence relation between the scalars of the domain on which the possibility distribution is defined.

4. Possibility distributions on numerical domains. This kind of values is represented as discrete fuzzy sets with disjunctive semantics. This kind of data includes the particular case of fuzzy numbers.

5. Special values representing the ignorance about a field value. This kind of values is represented by the special linguistic label $UNKNOWN$.

6. Special values representing the inapplicability of a field. This kind of values is represented by the special linguistic label $UNDEFINED$.

7. Special values representing the ignorance about the field applicability and about its field value if it were applicable. This kind of values is represented by the special linguistic label $NULL$.

The SDSDM model is able to represent the previous kinds of values. These are represented by means of the following constructs:

1. A single scalar. This kind of values is represented by using the previously described FCT for discrete scalar domains. As GEFRED, the SDSDM model supports the definition of a fuzzy equivalence relation between the scalars of the domain.

2. A single number. In SDSDM model this kind of values is represented as the ORDBMS built-in numerical type.

3. Possibility distributions on scalar domains. This kind of values is represented in SDSDM model as a combination of FCTs. The values are represented as a fuzzy collection with disjunctive semantics of values of the previously defined FCT that models the scalar domains.

4. Possibility distributions of numerical domains. In SDSDM model, this kind of values is represented, as the previous kind, as a combination of data types. The values are represented as fuzzy collections with disjunctive semantics whose elements are values of the host ORDBMS built-in type for numbers. Fuzzy numbers are represented by the previously described FCT for fuzzy numbers.

5. Special values representing the ignorance about a field value. The special linguistic label $UNKNOWN$ is also available in SDSDM model for symbolizing the ignorance of the attribute or field value.

6. Special values representing the inapplicability of a field. The special linguistic value $UNDEFINED$ is inherited by the SDSDM model from the GEFRED model in order to symbolize this kind of values.

7. Special values representing the ignorance about the field applicability and about its value if it were applicable. Once again, the SDSDM model adopts the special linguistic label $NULL$ from the GEFRED model to symbolize this kind of values.

In addition to the previous kinds of values, the SDSDM model is able to represent the following:

8. Possibility distributions on any kind of FCT and NFCT domains. Fuzzy collections with disjunctive semantics are not limited to numerical or scalar values. In SDSDM model it is possible to define a fuzzy collection with disjunctive semantics on every domain of the database, which includes every FCT or NFCT type.

9. Fuzzy sets on any kind of FCT and NFCT domains. The SDSDM model supports the representation of fuzzy sets defined on every kind of database domain, particularly including FCT and NFCT types. This kind of data is represented by fuzzy collections with conjunctive semantics.

10. Complex data types including FCT and NFCT attribute values. As an object-relational database model, SDSDM model allows the definition by the user of complex data types as a structure of fields, which can include user defined methods to encapsulate its behavior. The SDSDM model includes an additional and optional feature to enable the flexible comparison of complex data elements by making use of a flexible equivalence relation. This flexible equivalence relation is totally user definable as it is defined as a special method of the type, so the user can programmatically specify the relation. Of course, SDSDM model also allows the definition and usage on each model construct of classical complex data types that are not flexibly comparable.

2.6 Soft Data Server, a fuzzy object relational database management system

The previously depicted model is implemented in an experimental prototype named *Soft Data Server* (SDS) [15]. SDS is an extension of a well-known and widespread commercial ORDBMS. It creates a FORDBMS on the underlying ORDBMS.

SDS extends the host ORDBMS by taking advantage of the extension mechanisms included in the latest SQL standards, SQL:1999 and SQL:2003, as the underlying ORDBMS is compliant with useful parts of these standards for SDS purposes.

SDS mainly defines a group of UDTs which holds the representation and manipulation details of fuzzy data in the database, and helps the user to create his own FCTs.

These UDTs, and their supertype/subtype relation are depicted in fig. 2.2. The figure includes a pair of abstract data types that do not correspond to real database types, but help to clarify the SDS type structure. The mentioned abstract data types are *DatabaseDataTypes*, which model a root type for every database data type, and *BuiltInTypes*, which is the common ancestor for the built-in data types of the ORDBMS. In the figure these abstract data types have a dark background in order to di erentiate them from non-abstract UDTs

The UDTs for fuzzy data representation and manipulation included in SDS are the following:

- *FlexiblyComparableTypes*: This UDT is the common ancestor for all the UDTs included in SDS. Its main purpose is to encapsulate all the common and compulsory behavior for FCTs. One of these compulsory methods is an abstract method for flexible comparison, which is redefined in each subtype in order to implement its corresponding flexible equivalence relation. This redefinition is particularly important in user defined FCTs, so the user attaches to the data type an especially designed flexible equivalence relation. Another common behavior encapsulated in this UDT is the

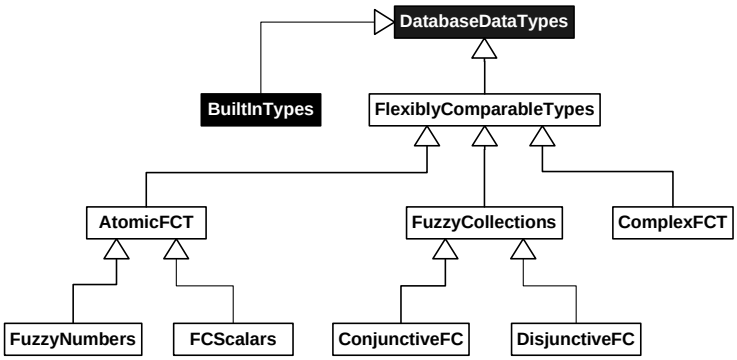

Figure 2.2: SDS Data types

set of methods to allow the definition, update and deletion of linguistic labels for the type.

- *AtomicFCT*: This UDT act as the common ancestor for those SDS UDTs designed to represent non complex or set oriented data, namely *atomic* fuzzy data, as fuzzy numbers and scalars.

- *FuzzyNumbers*: This UDT is designed to represent and manage fuzzy numbers in SDS. This UDT is a FCT as it implements the method *feq*. It encapsulates the flexible equivalence relation proposed in the subsection devoted to describe the FCT for fuzzy numbers. Additionally, this type includes methods implementing fuzzy relational comparators for fuzzy numbers. These fuzzy relational comparators are similar to those included in FSQL Server [21, 22], where are named *fuzzy comparators*.

- *FCScalars*: This UDT is designed as a common ancestor for the FCT representing discrete scalar domains where a flexible equivalence relation is defined. The type encloses helper methods for the definition and removal of FCTs representing flexibly comparable scalar domains along with their associated flexible equivalence relations.

- *FuzzyCollections*: This UDT is the common ancestor of every type included in SDS which represents an extension of the collection types of the ORDBMSs.

- *ConjunctiveFC*: This UDT gathers all the necessary functionality related to fuzzy collections with conjunctive semantics. This functionality includes helper methods for the definition and deletion of fuzzy collections of base type determined by the user. In addition to these DDL methods, the type includes a default implementation for the *feq* method, which implements the flexible equivalence relation. This implementation is made according to the flexible equivalence relation for fuzzy collections with conjunctive semantics described in subsection devoted to this kind of data.

- *DisjunctiveFC*: This UDT is analogous to the previously described UDT, but it is conceived for being the common ancestor of fuzzy collections with disjunctive semantics. As the previous UDT, this data type includes

helper methods for the definition and deletion of fuzzy collections whose elements are of a user determined base type. Likewise, the type includes a default implementation for the feq method that is based on the functional specification of the flexible equivalence relation described in the SDSDM model for this kind of types.

- *ComplexFCT*: This UDT is a common ancestor for every user defined FCT designed to represent and manage complex data organized as a structure of fields. It includes a set of methods that encapsulate the user defined behavior for the data type. This supertype includes a specific implementation of the feq method that is based on the generic flexible equivalence relation for flexibly comparable complex data types described in a previous section.

2.7 A fuzzy object-relational database application on image retrieval

This section introduces a sample application of a FDBMS implementing the SDSDM model. This example focuses on querying an image database by dominant color criteria, where each dominant color is described by linguistic labels. In this example the previously proposed data types are used to model a complex data type for representing the dominant color of an image, and the use of flexibly equivalence relations for retrieving images with a similar set of dominant colors.

2.7.1 Dominant fuzzy color descriptors

The base of this example has been previously published in [12, 2]. In these papers an algorithm for extracting dominant fuzzy colors from an image is described. For the example purpose, the only relevant detail is that the output of the algorithm is a fuzzy set representing the dominant fuzzy colors of an image. We refer the reader for particular details of this algorithm to the previous references.

A fuzzy color is described as a composition of three linguistic labels which describe the hue, saturation and illumination components of the HSI color space. A simplified set of these linguistic labels is illustrated by fig. 2.3. In this figure, each label is associated with its corresponding trapezoidal possibility distribution defined on the domain of a color component.

Each image is related to a fuzzy set of dominant fuzzy colors, where the membership degree of each dominant fuzzy color is equivalent to its degree of dominance in the image.

An example of the fuzzy set of dominant fuzzy colors of an image could be the one shown in eq. 2.37.

$$\begin{array}{l}\{0.7/(red, lowSaturation, bright),\\ 0.5/(blue, highSaturation, VeryHighIllumination),\\ 0.3/(yellow, mediumSaturation, HighIllumination)\}\end{array} \tag{2.37}$$

At first glance, an evident advantage for the user can be noticed. Traditional image databases work with quantities, so the user experiences more difficulties to understand these values, and to write queries. This approach makes flexible

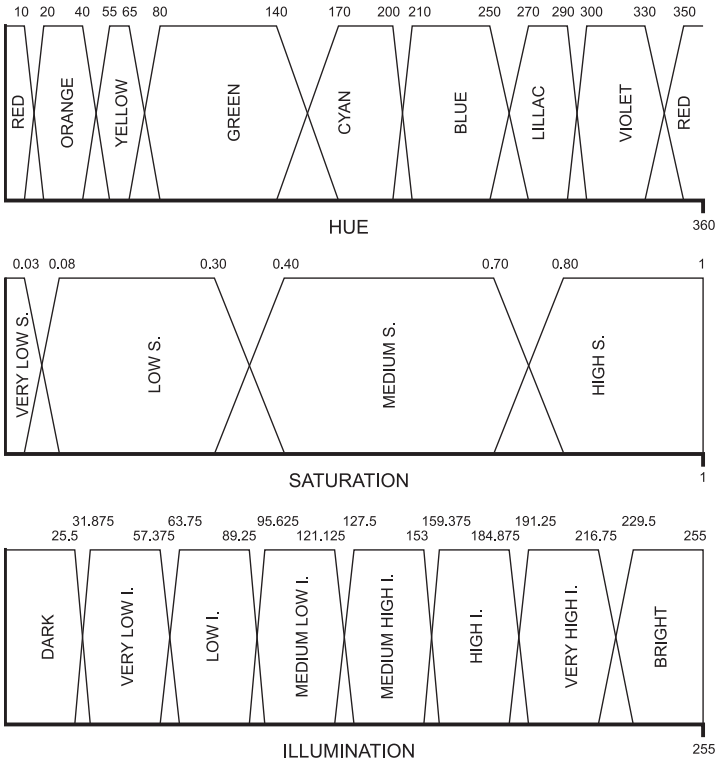

Figure 2.3: Linguistic labels for HSI color components

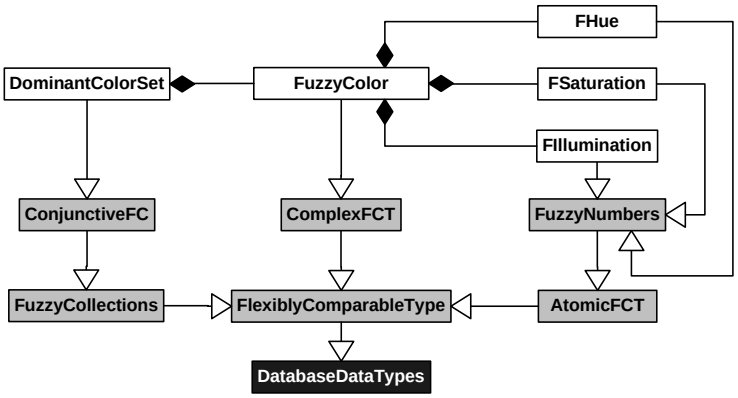

Figure 2.4: UML Diagram for DominantColorSet data type

the color comparison and definition and, at the same time, makes the query definition and the results more easily understandable for users.

2.7.2 Flexible comparable types for representing dominant fuzzy color descriptors

Previously, a descriptor for the set of dominant colors of an image is described. When this data is handled by an image retrieval system supported by a DBMS, a UDT for representing this complex piece of data can ease the storage, handling and querying of these image descriptors.

As these descriptors are a complex combination of fuzzy data constructs, a FORDBMS based on the SDSDM model would be very convenient for storing and querying this kind of data.

The data type *DominantColorSet* is the FCT which models the proposed image domain fuzzy color descriptor. This data type is modelled as shown in fig. 2.4.

Let us describe the definition the *DominantColorSet* data type by a bottom-up approach. A fuzzy color is basically composed of linguistic labels which represent its hue, saturation and illumination components. These linguistic labels, in turn, correspond to trapezoidal possibility distributions defined on crisp numerical hue, saturation and illumination domains, as shown in fig. 2.3. Each component of the fuzzy color could be modelled as a FCT based on the *FuzzyNumber* data type of SDS. Thus, the FCT *FHue*, *FSaturation* and *FIllumination* data types are defined in order to model the domain of the components of a fuzzy color. On these data types, the linguistic labels, shown in fig. 2.3, are defined on their corresponding data type in order to make them valid constant values.

As it was described previously, a fuzzy color is a composition of three linguistic labels, each one representing a fuzzy value of the hue, saturation and illumination components. In the database a fuzzy color is represented as a complex object. The data type *FuzzyColor*, which is a subtype of the *ComplexFCT* SDS data type, is a complex data type composed by three fields whose values are respectively of the *FHue*, *FSaturation*, and *FIllumination* data types.

Finally, the *DominantColorSet* data type is defined as a fuzzy collection with conjunctive semantics whose elements are of the *FuzzyColor* FCT. The

membership degree of each element of the fuzzy collection corresponds to the degree of dominance of the represented fuzzy color.

2.7.3 Flexible operators for dominant color based retrieval

Once the dominant color descriptors are stored in the fuzzy database along with their corresponding images, it is necessary to provide the user with a way of defining queries to retrieve the images using the color descriptors as criteria.

As the dominant color descriptors of images are fuzzy collections of fuzzy colors, the previously defined operators for fuzzy collections are useful.

The previously defined resemblance driven inclusion is a flexible way of retrieving from a database images that include a set of dominant fuzzy colors. For instance, the user could ask the system for images including *high saturated bright red*, and *medium saturated high illuminated blue* as dominant colors. If the inclusion operator is used to define a condition on the set of dominant colors in a query, the result is composed by images including the set of fuzzy dominant colors defined in the condition. Besides, as the proposed inclusion operator is based on resemblance measures, the results also contain images including a similar set of fuzzy dominant colors. Each returned image is related to a fulfillment degree, which is as high as the set of dominant fuzzy colors of the image are more similar to the set defined in the condition. This fulfillment degree makes possible the ordering of the result by their similarity to the query conditions.

Another useful element for query definition in the proposed image database is the previously defined fuzzy collection resemblance measure, which is modelled as the fuzzy collection resemblance operator. If this operator is used in queries, the user can retrieve images with a dominant fuzzy color descriptor similar to a given one. This kind of queries is useful to retrieve from an image bank those images with a similar set of dominant colors of a given sample image.

Flexible equivalence relation for fuzzy color instances

As it has been described previously, fuzzy color instances are values of the *FuzzyColor* data type. This data type is a subtype of the *ComplexFCT* data type and therefore its default flexible equivalence relation is the previously proposed generalized resemblance measure for flexible comparable types.

During some tests with users, it was found that the resemblance degree between two fuzzy colors computed using this default flexible equivalence relation results to be very strict.

As the SDSDM model is designed with openness in mind, the deliberated inheritance relation between the user defined FCT for representing fuzzy colors (*FuzzyColor*) and its ancestor (*ComplexFCT*) makes possible a redefinition of the flexible comparable relation of domain elements, so it can be adapted to the user needs and to the domain specificities.

When the *FuzzyColor* data type is defined, its flexible equivalence relation is defined to match the expression shown in eq. 2.38, where o_1 and o_2 are two objects of the *FuzzyColor* data type , $o_1.a_i$ and $o_2.a_i$ are the values of the i-th attribute of the objects o_1 and o_2, and feq is the membership function of the flexible equivalence relation of each corresponding attribute type.

$$feq(o_1, o_2) = \frac{1}{n} \sum_{i \in \mathcal{A}}^{n} feq(o1.a_i, o2.a_i) \tag{2.38}$$

2.7.4 Image retrieval by a dominant fuzzy color criteria

A dominant fuzzy color based image retrieval can be easily performed by taking advantage of the previously defined FCTs, operators and the fuzzy color resemblance measure.

The previous elements make the FORDBMS able to answer queries including conditions defined in the set of dominant colors. SDS takes advantage of the latest SQL standards so a flexible query for this server could be expressed as a standard SQL in which UDT methods and user defined operators are used to allow the access to flexible query features of the FORDBMS.

A dominant color based condition is, in fact, a flexible condition defined on the fuzzy collection of fuzzy colors describing the dominant colors of each image in the database. As it has been stated before, the inclusion and resemblance operators of fuzzy collections can be used to create this kind of conditions. Each user defined fuzzy color constant used in a query can be defined by using the linguistic labels previously defined for the FCTs that represent the HSI color components.

A requirement on a color component could be omitted by using the special linguistic label $UNKNOWN$, which fully resembles to any value.

An example of a condition using the fuzzy inclusion operator is *"Retrieve all images including bright very high saturated red"*. This condition is defined using the fuzzy collection inclusion operator between the image descriptor and a user defined fuzzy collection constant of fuzzy colors. A query applying this condition is expressed in SQL using the previously defined data types and operators as the following sentence:

```
SELECT image,cdeg(1) FROM images WHERE FCond( FInclusion(
  ColorDescriptor,
  DominantColorSet(1.0, FuzzyColor(
        FHue('red'),FSaturation('veryhighsat'),
            FIllumination('bright')
        ) ) ) , 1 ) > 0 ORDER BY 2 DESC;
```

where *cdeg* is an *ancillary operator* [32] that returns the fulfillment degree of the flexible condition marked by the same numerical value in the *where* clause. $FCond$ is a function that encloses flexible conditions and returns their fulfillment degree. An ancillary operator is a special operator that makes possible the transfer of the result of an operator in the *where* section of a query to the select section. In this case, $FCond$ has the value 1 as its last argument to link it to the *cdeg* operator at the *select* section that receives the same value as argument.

Figure 2.5 shows the results of the previous query and some more complex query examples combining several color inclusion conditions of this type of queries applied on a database of 160 flag images. In this figure, the first column shows the fuzzy colors which must be included in the resulting images, and for each fuzzy color a sample crisp color which fits it. The second column shows the top 5 most relevant images.

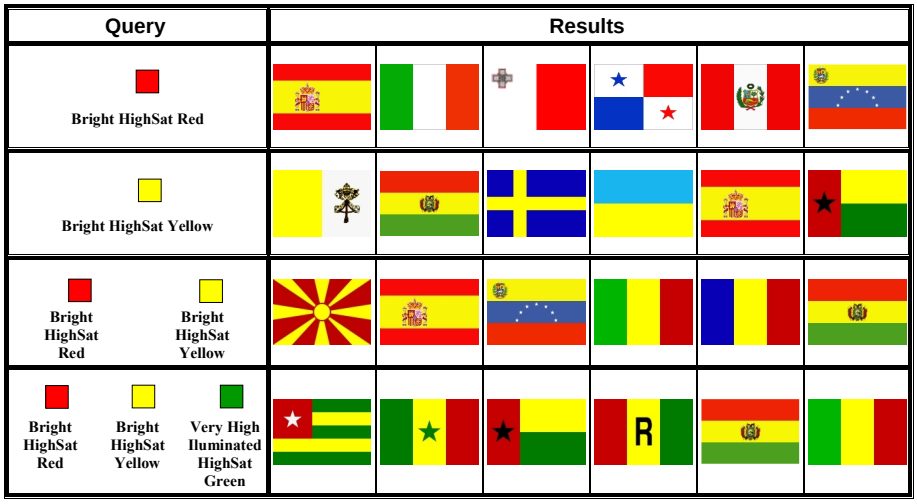

Figure 2.5: Color inclusion query results

Another example of this kind of condition is the requirement *"Retrieve all the images including bright colors"*. In this case, the inclusion operator must ensure the inclusion of the fuzzy color whose hue is $UNKNOWN$, its saturation is $UNKNOWN$, and its illumination is *bright*. The label $UNKNOWN$ must be used to avoid the definition of the condition for the hue and saturation components. The previously defined query, applied on a database of about 700 color images, obtains the results shown in fig. 2.6. In the same figure the results of a similar query for retrieving images with dark dominant colors is included.

Finally, an example of a query using the resemblance operator to retrieve the set of images of a database with a dominant color pattern similar to the one associated with a sample image is the following:

```
SELECT a.image,cdeg(1) FROM images a, images b WHERE b.id=# AND
FCond( FEQ( a.ColorDescriptor, b.ColorDescriptor ),1 ) > 0
ORDER BY 2 DESC;
```

where # is a number constant corresponding to the numerical identifier (i.e. primary key) of the sample image.

The results of several examples of this kind of queries, applied on a database with about 700 color images, are shown in fig. 2.7. In each example, the first column shows the sample image, and the second column shows the set of images ordered by relevance with a dominant color pattern similar to the image sample. The image sample is excluded from the results for the sake of brevity, as it is obviously a perfect match.

2.8 Conclusions and future research directions

At the beginning of this chapter, the current trends in the database world and in the fuzzy database research field have been discussed.

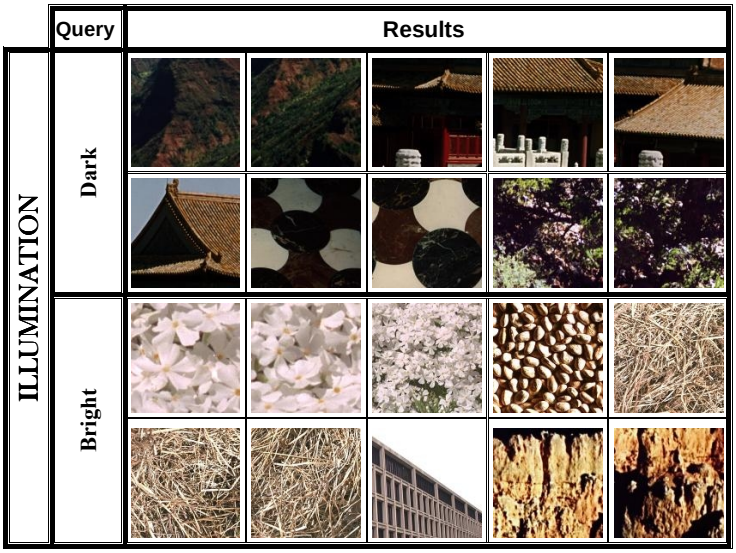

Figure 2.6: Color inclusion query results for the illumination component

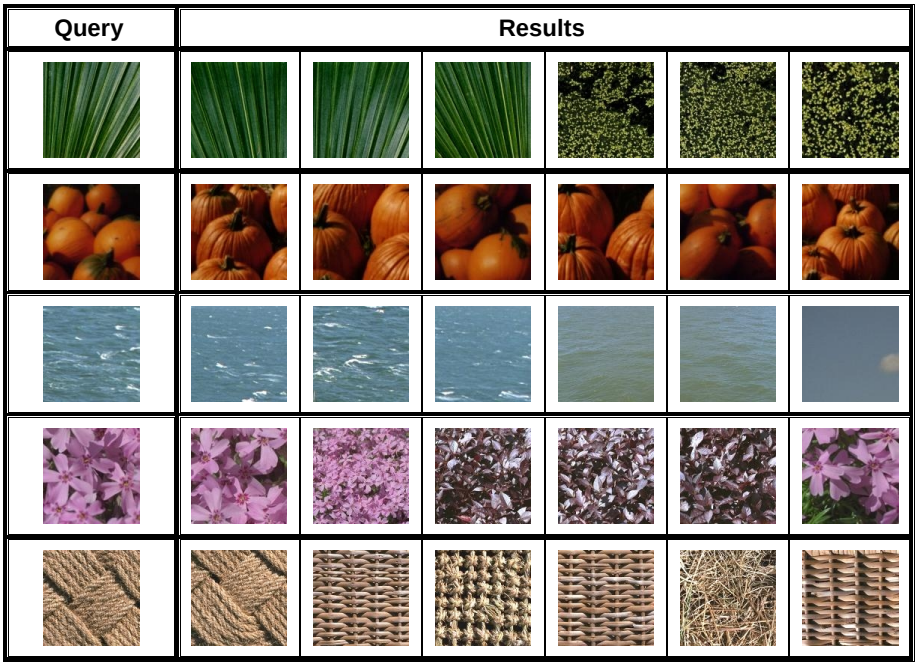

Figure 2.7: Image resemblance query results

On the one hand, the most recent answer of crisp DBMS practitioners and manufacturers to the problem of complex data management is the object-relational database model. Object-relational databases are being well accepted by the database market due to their combination of the good features of the relational and object-oriented database models. As a result of its success and acceptance, the object-relational paradigm is gradually being incorporated in recent SQL standards.

On the other hand, the aim of the fuzzy database models proposed in the literature has been to extend current database models in order to make them able to represent and retrieve fuzzy data. As a result, the relational model and the object-oriented database model, have been extended in several ways.

In this chapter we propose the SDSDM model. This is an extension of the object-relational model to make it able to represent fuzzy data. The model keeps the modelling power of previous fuzzy relational models, as it is able to represent the kind of fuzzy data that can be represented in previous proposals. Moreover, the SDSDM model is able to represent new kinds of fuzzy data resulting from the extension of the basic data constructs of ORDBMS, the UDTs and the collection data types. These new fuzzy data types make the SDSDM model able to represent complex data composed of fuzzy and crisp values, and fuzzy collections of data with conjunctive and disjunctive semantics. The model is designed with openness in mind, so the user can customize each default aspect of the model.

Additionally, the chapter has introduced the basis of an implementation of the SDSDM model in a FORDBMS. The proposed data types can be seamlessly integrated in an ORDBMS by making use of its native extension mechanisms, i.e. UDTs. As the fuzzy data representation features are integrated as native extensions, there is no need for a special bridge, which overlies the underlying DBMS, to process the specificities of the fuzzy data. Actually, every component for fuzzy data handling is fully integrated in the host ORDBMS and the query language extension for fuzzy querying is fully SQL compliant. A sample application of the resulting FORDBMS and its query language has been presented.

One of the main weaknesses of fuzzy databases is that the flexible conditions in queries hugely increment the number of candidate results. This leads to a performance reduction of query processing, which makes fuzzy databases not actually competitive, in contrast with crisp systems. Future research will focus on providing indexing mechanisms for fuzzy data in order to increment the performance of FORDBMS query processing. Additionally, the optimization of fuzzy queries should also be addressed, as it could contribute to an additional increase of fuzzy query processing performance.

Chapter 3

A B$^+$-tree based indexing technique for fuzzy numerical data

3.1 Abstract

This paper proposes an indexing technique for fuzzy numerical data which increases the performance of query processing when the query involves an atomic possibility measured flexible condition. The proposal is based on a classical indexing mechanism for numerical crisp data, B$^+$-tree, which is implemented in most commercial database management systems (DBMS). This makes the proposed technique a good candidate for integration in a fuzzy DBMS when it is developed as an extension of a crisp DBMS. The e ciency of the proposal is contrasted with another indexing method for similar data and queries, G-tree, which is specifically designed to index multidimensional data. Results show that the proposal performance is similar to and more stable than the measured for G-tree when used for indexing fuzzy numbers.

3.2 Introduction

Fuzzy set theory is a paradigm shift which helps to resolve classical, and non-classical, problems in a more convenient way than crisp systems by softening set boundaries. The database world has taken advantage of fuzzy set theory by using it as a way to manage imprecise, uncertain and inapplicable data (called *fuzzy data* in this framework) and to model and process flexible queries. As a result of this trend, a significant number of proposals on fuzzy database models [18, 39, 34, 49, 30, 10] and flexible querying [7] have been published. There have also been various implementations of fuzzy DBMS (FDBMS) [24, 21], in which imprecise, uncertain and inapplicable data can be managed and/or flexible queries can be processed.

Original paper published in: Barranco, C. D., Campaña, J. R., and Medina, J. M. 2008. A B+-tree based indexing technique for fuzzy numerical data. Fuzzy Sets Syst. 159, 12 (Jun. 2008)

The emergence of fuzzy databases means that a new tool is available for developing novel applications which would process very large data sets pervaded with imprecision and vagueness using fuzzy methods. As these novel applications prove their potential as prototypes, they are integrated into real-world environments. In this kind of environment, high performance, scalability and availability are required for applications, and subsequently for each of their components, particularly for their underlying FDBMS.

Unfortunately, performance is the Achilles heel of existing FDBMS implementations. Many authors point out that FDBMS should not be developed from scratch as building an e cient database engine is a costly task. In this case, the resulting FDBMS (most of them research prototypes) would not o er the same high performance, scalability and availability as their crisp counterparts. If so, this shortcoming would undermine the benefits provided by the fuzzy approach.

An FDBMS can, however, be built on a crisp database engine. This approach allows advantage to be taken of the qualities of the underlying crisp DBMS with no costly implementation. One solution following this idea has been o ered by a recent proposal for a fuzzy object-relational DBMS (FORDBMS) model [15]. This proposal takes advantage of the extension mechanisms of recent commercial object-relational DBMS (ORDBMS) in order to build user-defined data types to seamlessly represent, store, query and manage fuzzy data by modeling them as native database objects. The resulting FORDBMS following this approach combines the high performance, scalability and availability of the underlying crisp DBMS with the fuzzy data handling power of FDBMS.

In addition to an e cient database engine, indexing mechanisms are the key for high performance in databases. Although a great deal of research has been carried out into fuzzy database models, not so much work has been done into indexing mechanisms for e ciently accessing fuzzy data. Moreover, the existing proposals of indexing mechanisms for fuzzy data require specific data structures and algorithms which cannot be easily incorporated into an ORDBMS using its extension mechanisms.

This paper proposes an indexing technique for fuzzy numerical data to improve query processing when a particular kind of flexible condition, known as the possibility measured atomic flexible condition and defined in the following section, is involved. The data structure and search algorithm of the proposed indexing mechanism is based on classical indexing structures for numerical crisp data, i.e. B^+-trees. This underlying indexing technique is not specifically designed to index fuzzy numerical data but it is a simple and well-optimized technique which is available in virtually every current DBMS. This near-to-general availability of B^+-tree indexing methods would result in a reduction of implementation, integration and optimization e orts, which in some cases do not exist at all, to incorporate the proposed indexing technique to an FDBMS extending a crisp DBMS. Although its performance might be lower than more complicated specifically designed fuzzy numerical data indexing techniques, we do not consider the di erence in performance to be significant and therefore the e ort reduction would make up for this.

The chapter is organized as follows. The concept of fuzzy numerical data and possibility measured flexible conditions in the context of the paper is described in Section 3.3. Section 3.4 briefly introduces related work on fuzzy data indexing. Section 3.5 outlines the indexing principle on which the proposed and studied indexing techniques rely. Section 3.6 describes the studied indexing techniques.

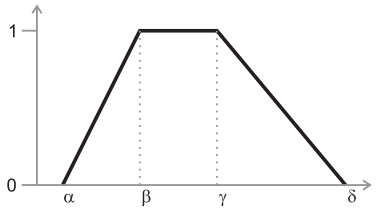

Figure 3.1: Trapezoidal characteristic functions

Section 3.7 presents the measures and procedures for evaluating the performance of the studied indexing techniques. Section 3.8 analyzes the performance results. Finally, Section 3.9 contains the concluding remarks and explores future lines of research.

3.3 Fuzzy data and flexible queries

As mentioned previously, this paper focuses on the proposal of an indexing method and its associated data structure for *fuzzy numerical data* to improve the performance of query processing when a *possibility measured atomic flexible condition* is involved. This section explains these concepts in the context of this paper.

3.3.1 Fuzzy numerical data

For the purposes of the paper, a fuzzy numerical value is an imprecise and/or vague numerical value. This kind of value is stored in a fuzzy database when the precise value of a numerical attribute is not known. Examples of this kind of data for an *age* attribute are about five or between 20 and 30 . A value of this kind of data is modeled by a convex possibility distribution defined on an underlying domain D in which a linear order relation is defined. The function that characterizes the possibility distribution maps every element d of D to the possibility degree of the fact that d is the actual value of the attribute.

This kind of fuzzy data can be modeled using any possibility distribution characterized by a convex function $f : D \rightarrow [0, 1]$, and the considered indexing techniques are valid for any of them. Trapezoidal functions will, however, be used exclusively in the examples in this paper for the sake of simplicity and the strong expressive capacity of this kind of characteristic function. A trapezoidal function $t_{[\alpha,\beta,\gamma,\delta]}$ on the underlying domain D is defined as shown in Eq. 3.1. In the rest of the paper it is noted as $[\alpha, \beta, \gamma, \delta]$ for the sake of simplicity. A graphical representation of this kind of function is shown in Fig. 3.1.

$$t_{[\alpha,\beta,\gamma,\delta]}(d) = \begin{cases} 0 & d \leq \alpha \text{ or } d \geq \delta \\ \frac{d-\alpha}{\beta-\alpha} & \alpha < d < \beta \\ 1 & \beta \leq d \leq \gamma \\ \frac{\delta-d}{\delta-\gamma} & \gamma < d < \delta \end{cases} , \alpha, \beta, \gamma, \delta, d \in D, \alpha \leq \beta \leq \gamma \leq \delta \quad (3.1)$$

Trapezoidal possibility distributions are well suited for representing several kinds of fuzzy and crisp numerical data such as:

- Fuzzy intervals: when the trapezoidal possibility distribution is defined as $[p-m, p, q, q+m]$, where $[p, q]$ is a crisp interval, and m is the margin defining the looseness of the approximation.
- Approximate values: this is a particular case of the earlier type when the possibility distribution is defined as $[n-m, n, n, n+m]$, where n is the value which is approximated, and m is the margin for expressing the looseness of the approximation.
- Crisp intervals: a particular case of fuzzy intervals is when some imprecision about the value can be represented as a strict interval. In this case, the trapezoidal possibility distribution is defined as $[a, a, b, b]$, where a and b are the lower and upper bounds of the interval.
- Crisp numbers: a particular case of all the previous kinds is when the possibility distribution is defined as $[n, n, n, n]$, where n is the precise number.

3.3.2 Atomic flexible conditions

In the context of this paper, an atomic flexible condition is a combination of a mono-attribute *fuzzy condition* and a threshold.

A *fuzzy condition* is a gradual restriction imposed on the values of an attribute. This restriction is modeled as a fuzzy set of acceptable values which must be characterized by a convex membership function. Some instances of this kind of condition on an age attribute are around 30 and in his/her 20s .

Naturally, a fuzzy condition could be partially satisfied. When a fuzzy condition is applied on an attribute of a table containing fuzzy numerical values (including crisp values as a particular case), it results in two fuzzy sets of rows: those *possibly* satisfying the condition and those *necessarily* satisfying it. The membership degree of the table rows to these two fuzzy sets are the *possibility* and the *necessity* degrees, respectively, and they indicate the extent to which the fuzzy condition is *possibly* and *necessarily* satisfied.

The membership function of the fuzzy set of rows possibly and necessarily, respectively, satisfying the fuzzy condition C on the attribute A is defined as in Eq. 3.2, and Eq. 3.3, respectively. For these equations, $D(A)$ is the underlying domain associated to the fuzzy attribute A, $\Pi_{A(r)}$ is the possibility distribution which describes the fuzzy value of the attribute A for the row r, and μ_C is the membership function defining the fuzzy condition C.

$$\Pi(C/r) = \sup_{d \in D(A)} (\Pi_{A(r)}(d) \wedge \mu_C(d)) \tag{3.2}$$

$$N(C/r) = \inf_{d \in D(A)} ([1 - \Pi_{A(r)}(d)] \vee \mu_C(d)) \tag{3.3}$$

This paper only focuses on possibility measured conditions since necessity measured conditions, which can take advantage of a more e cient indexing schema, warrant specific work. Nevertheless, since the possibility degree is always greater than or equal to the necessity degree [25], the proposed indexing schema can be used as a preselection mechanism for the processing of queries including necessity measured atomic flexible conditions. In the rest of the paper, the possibility degree is therefore called the fulfilment degree of the condition.

A fuzzy condition is typically used in conjunction with a crisp relational comparator for setting a threshold (i.e. a minimum) for its fulfilment degree. This minimum specifies the degree of flexibility in which the fuzzy condition is applied. The threshold varies from 1 (no flexibility) to 0 (maximum flexibility). The typical expression for applying a threshold T is $\Pi(C/r) \geq T$, except when the threshold is 0. In the latter case the expression $\Pi(C/r) > T$ is applied. The combination of a mono-attribute fuzzy condition with a threshold results in a crisp Boolean condition. This combination is called an atomic flexible condition for the purposes of this paper and is notated as $\langle C, T \rangle$.

3.3.3 Queries involving flexible conditions

The performance of the evaluation process of a query involving an atomic flexible condition can be improved by using an index when the fuzzy restriction is a constant fuzzy numerical value. An example of this kind of query is the following:

```
SELECT * FROM table WHERE FEQ(att,[a,b,c,d])>=T
```

The previous query is based on the syntax of Soft Data Server (SDS) [15], a prototype of an FORDBMS. It includes the fuzzy comparator `FEQ` which is based on the described possibility measure. When applied to two attributes or values of a fuzzy numerical data type, the operator returns a numerical result in the interval $[0, 1]$. This value indicates the extent to which the values of the two attributes are possibly equal.

This query returns a table containing all the rows from the table `table` where its value for the fuzzy numerical attribute `att` is possibly equal to the fuzzy constant restriction, which is represented as a trapezoidal fuzzy set $[\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}]$, with a fulfillment degree greater than or equal to the threshold value `T`.

3.4 Related work

As mentioned previously, not much research has been carried out into the increase of performance of fuzzy databases by means of indexing techniques. Included in this work is the seminal paper [5] by Bosc et al. which highlights the need for specific indexing techniques for fuzzy databases. This paper proposes two indexing principles for flexible querying using possibility and necessity measures on fuzzy attributes modeled as possibility distributions. These indexing principles, which are described in the next section, are the basis of later work on indexing this kind of fuzzy data, which includes the indexing techniques discussed in this paper.

In [6, 33], the same author proposes an indexing method to improve the performance of flexible query processing on crisp databases. This consists in creating one index structure for each fuzzy predicate associated to a crisp attribute. Each index structure (one for each fuzzy predicate) associates each possible satisfaction degree of the fuzzy predicate to the rows which satisfy the predicate at this degree. This indexing technique results in a costly technique due to the high number of necessary index structures, which raises the storage and maintenance costs. In addition, the indexing technique is only applicable when the number of potential flexible restrictions which can be used to build

flexible conditions of queries is finite and low. Fuzzy restrictions must be linguistic labels, and so it is not possible to query using arbitrarily defined fuzzy restrictions and the attribute domains cannot be fuzzy.

At a later stage, Yazici et al. proposed an indexing technique for fuzzy databases based on a crisp multidimensional indexing technique [43, 44]. This proposal is applicable on heterogeneous domains (i.e. domains including crisp and fuzzy values) and needs only one indexing structure. The experimental results of these publications reveal that the proposed indexing technique outperforms the previous one proposed in [6, 33]. The technique has been enhanced and adapted to work in fuzzy object-oriented databases in [45]. This technique is only applicable when the domain of the indexed attribute contains a finite number of fuzzy values defined as linguistic labels that are also used as the only available fuzzy restrictions.

Finally, [28] proposes an indexing technique based on a crisp multidimensional indexing technique suitable for indexing arbitrary fuzzy values defined on an ordered underlying domain as convex possibility distributions. This technique is valid for the kind of fuzzy data and flexible conditions on which this paper focuses and it is described in detail in a later section of this paper.

3.5 Indexing principle

The proposed indexing scheme and the fuzzy index to which it is compared are based on the indexing principles proposed in [5]. This work proposes an indexing principle for possibility and necessity measured flexible conditions. This section briefly describes the principle for possibility measured flexible conditions and includes some discussion about it.

3.5.1 Preselection mechanism

The indexing principle considered in the paper is useful for preselecting those rows from the queried table which potentially satisfy an atomic flexible condition (in other words, the principle allows those rows which definitely do not satisfy the condition to be filtered out). To accomplish this goal, the indexing principle follows the next reasoning.

When an atomic flexible condition based on a possibility measure is included in a query in its least restrictive case, when the threshold T is set to zero, the only rows which are excluded from the result are obviously those with a fulfillment degree which is equal to zero. Recalling Eq. 3.2, it can be concluded that the fulfillment degree of a possibility measured condition is greater than zero for a row r if and only if there is at least a possible element of the possibility distribution $\Pi_{A(r)}$ modeling the attribute value $A(r)$ which also belongs to the fuzzy set modeling the condition restriction. This conclusion can be reformulated in a set-oriented expression as Eq. 3.4 shows.

$$\Pi(C/r) > 0 \iff \exists d \in \Pi_{A(r)}, d \in C \iff \Pi_{A(r)} \cap C \neq \emptyset \iff supp(\Pi_{A(r)}) \cap supp(C) \neq \emptyset \tag{3.4}$$

In the previous equation, $supp(S)$ represents the support of the fuzzy set S, which is defined as Eq. 3.5 shows, where D is the underlying domain on which S is defined and μ_S is the membership function of the fuzzy set S.

$$supp(S) = \{d \in D, \mu_S(d) > 0\} \tag{3.5}$$

If the threshold of the atomic flexible condition is greater than zero, the reasoning of Eq. 3.4 can be reformulated as Eq. 3.6 shows. In this equation, $supp_\lambda(S)$ represents a λ-cut of a fuzzy set S, defined as in Eq. 3.7, where D is the underlying domain on which S is defined and μ_S is the membership function of the fuzzy set S.

$$\begin{array}{l}\Pi(C/r) \geq T \iff \exists d \in \Pi_{A(r)}, d \in C, \Pi_{A(r)}(d) \geq T, \mu_C(d) \geq T \iff \\ supp_T(\Pi_{A(r)}) \cap supp_T(C) \neq \emptyset\end{array} \tag{3.6}$$

$$supp_\lambda(S) = \{d \in D, \mu_S(d) \geq \lambda\} \tag{3.7}$$

The conclusion of Eq. 3.4 could be a very e ective indexing principle when the threshold of a flexible condition is zero, as every row satisfying it must be included in the set of rows satisfying the flexible condition. To apply this principle, it is only necessary to keep the support of the data in the restricted attribute in a fast access data structure.

However, the conclusion of Eq. 3.6 is not an e ective indexing principle for flexible conditions with a threshold greater than zero. Although every row satisfying it satisfies the flexible condition, the application of the indexing principle requires quick access to every possible λ-cut of the data in the restricted attribute. Nevertheless, it is possible to derive from Eq. 3.6 a weakened indexing principle, as Eq. 3.8 shows, which only requires the support of the fuzzy numerical data to be indexed.

$$supp_T(\Pi_{A(r)}) \cap supp_T(C) \neq \emptyset \Rightarrow supp(\Pi_{A(r)}) \cap supp_T(C) \neq \emptyset \tag{3.8}$$

It is worth mentioning that the previous weakened principle is a necessary but not a su cient condition, and therefore each row satisfying a flexible condition satisfies it, but not every row satisfying it satisfies the flexible condition. For this reason, the weakened principle can only be used to preselect the candidate rows for the query result by filtering out those rows which definitely do not satisfy the necessary condition. After obtaining the preselection set of rows, it is necessary to check whether each preselected row satisfies the flexible condition in order to exclude possible false positives from the final result.

In our opinion, the previous conclusions can be generalized and reformulated creating only one preselection criterion as shown in Eq. 3.9 and Eq. 3.10. In these equations, the preselection function *ps* is defined as shown in Eq. 3.11, for which *base* is defined in Eq. 3.12. Henceforth, the interval resulting from Eq. 3.12 when applied to a flexible condition $\langle C, T\rangle$ is called the *base* of the flexible condition.

$$\Pi(C/r) > 0 \iff ps(C/r, 0) \tag{3.9}$$

$$\Pi(C/r) \geq T \Rightarrow ps(C/r, T), 0 < T \leq 1 \tag{3.10}$$

$$ps(C/r, T) \iff supp(\Pi_{A(r)}) \cap base(\langle C, T\rangle) \neq \emptyset, 0 \leq T \leq 1 \tag{3.11}$$

$$base(\langle C,T\rangle) = \begin{cases} supp(C) & ,T = 0 \\ supp_T(C) & ,0 < T \leq 1 \end{cases} \tag{3.12}$$

When the fuzzy data and the fuzzy conditions are modeled using convex possibility distributions defined over a linearly ordered underlying domain, it can be assumed that the support and generally every λ-cut is an interval. With this in mind, when the preselection criterion is applied on fuzzy numerical data, it can be reformulated as a conjunction of range conditions as shown in Eq. 3.13. For this equation, $\inf(S)$ and $\sup(S)$ correspond, respectively, to the infimum and the supremum of a crisp set S.

$$\begin{array}{l} ps(C/r,T) \iff \sup(supp(\Pi_{A(r)})) \geq \inf(base(\langle C,T\rangle)) \wedge \\ \inf(supp(\Pi_{A(r)})) \leq \sup(base(\langle C,T\rangle)) \end{array} \tag{3.13}$$

The previous indexing principle was proposed in Bosc's seminal work [5] which relies on three assumptions:

1. The underlying domains are provided with a linear order relation.
2. Each attribute value is represented by a possibility distribution whose support is a closed interval.
3. The restriction of each fuzzy condition is modeled by a fuzzy set whose support is a closed interval.

If our basic example is considered, when the possibility distributions and fuzzy restrictions are modeled by trapezoidal membership functions, it is obvious that the last two assumptions are not applicable because the support for a trapezoidal fuzzy set is an open interval.

In our opinion, these assumptions can be avoided if the previous indexing principle and preselection criteria are substituted by weakened ones which are derived from the original. This derivation is shown in Eqs. 3.14, 3.15 and 3.16, where $\Pi_{A(r)}$ is the trapezoidal possibility distribution $[\alpha_{A(r)}, \beta_{A(r)}, \gamma_{A(r)}, \delta_{A(r)}]$ and $base(\langle C,T\rangle)$ corresponds to the open interval $(l_{base(\langle C,T\rangle)}, u_{base(\langle C,T\rangle)})$ or the closed interval $[l_{base(\langle C,T\rangle)}, u_{base(\langle C,T\rangle)}]$, where $l_{base(\langle C,T\rangle)}$ and $u_{base(\langle C,T\rangle)}$ are, respectively, the infimum and the supremum of $base(\langle C,T\rangle)$.

$$\Pi(C/r) > 0 \Rightarrow ps'(C/r,0) \tag{3.14}$$

$$\Pi(C/r) \geq T \Rightarrow ps'(C/r,T), 0 < T \leq 1 \tag{3.15}$$

$$ps'(C/r,T) \iff \delta_{A(r)} \geq l_{base(\langle C,T\rangle)} \wedge \alpha_{A(r)} \leq u_{base(\langle C,T\rangle)} \tag{3.16}$$

It should be noted that the proposed preselection criterion slightly increases the occurrence of false positives in some extreme cases when $\delta_{A(r)} = l_{base(\langle C,T\rangle)}$ or $\alpha_{A(r)} = u_{base(\langle C,T\rangle)}$ if these values are not part of the support of the trapezoidal distributions $A(r)$ and C. This is a collateral e ect of including the boundaries of open intervals to transform them into closed ones in order to avoid the restrictive assumptions.

record	r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	r11	r12
key	5	7	8	3	10	1	6	4	15	2	11	9

Table 3.1: Sample data

3.6 Indexing techniques

This paper compares two indexing techniques for fuzzy numerical data based on the previous indexing principle. The main di erence between these schemes is the kind of indexing structure for crisp data on which they are based. The first approach, the one proposed in this paper, takes advantage of a combination of two classical B^+-trees, whereas the second approach uses a multidimensional indexing structure. The following subsections describe both approaches.

3.6.1 Fuzzy indexing using B^+-trees

The initial idea of this proposal was depicted in [2]. This paper reformulates it in order to make it simple to evaluate its performance by the indexing technique described in the previous section. This fuzzy indexing technique takes advantage of classical B^+-tree indexing structures for indexing the intervals representing the support of the indexed fuzzy numerical data. For the rest of the paper this technique is called 2BPT.

B^+-trees

A B^+-tree [14] is one of the variations of B-trees [4]. This kind of data structure is a type of balanced tree used to maintain a collection of sorted records by a key value in a way that allows for e cient insertion, retrieval and removal.

A B-tree stores the indexed keys and a pointer to its associated record in every tree node, so that when the tree is traversed in-order, the records can be retrieved in the order determined by their key values. To maintain the balance of the tree, each node is allowed to store a minimum number of entries d, and a maximum number of entries $2d$. This policy ensures a minimum node usage of 50%. The value d is named the *order* of the tree [14]. A B^+-tree, in contrast, stores the indexed keys and a pointer to its associated record only in the leaf nodes. The internal (i.e. non-leaf) nodes only store key values, and act as a fast index access to the entries in leaf nodes. Additionally, each leaf node is linked to the next leaf node. Because all the entries are stored in the leaf nodes and these nodes are linked, the performance of sequential processing of the indexed data of B^+-trees is higher than of B-trees, especially when the tree is stored in secondary memory.

An instance of a B^+-tree of order 1 indexing the example records included in Table 3.1 is graphically represented in Fig. 3.2, where the stripped entries of the nodes are empty.

It can be seen that each entry in a node has two associated pointers on its left and right sides. For the internal nodes, the pointer on the left points to the subtree indexing the records with a key value which is less than the entry value. Respectively, the pointer on the right points to the subtree indexing the records with an associated key value which is greater than or equal to the entry value. The leaf nodes make use of the left and right pointer of an entry in a di erent

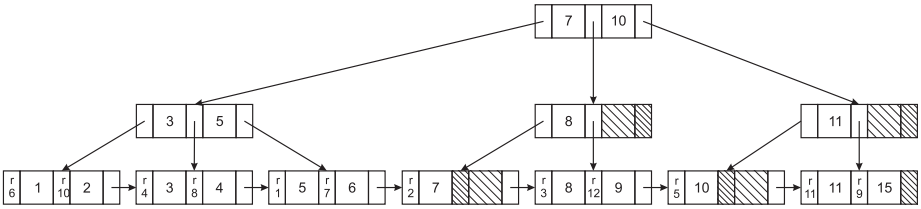

Figure 3.2: A sample B$^+$-tree

Figure 3.3: B$^+$-tree node structure

way: the left pointer points to the record associated with the key value of the entry, whereas the farthest right pointer (the only one not associated as a left pointer to an entry) points to the next leaf node in the tree.

We refer the reader to [14] for details about the insertion and deletion algorithms of B$^+$-trees, which are responsible for maintaining tree balance.

The search algorithm for a record with an associated key k in a B$^+$-tree is straightforward. Starting from the root node, we look for the leaf node which should contain k. To determine the path to the leaf node, we follow the pointer p_i between two consecutive entries k_{i-1} and k_i fulfilling $k_{i-1} \leq k < k_1$, p_0 if $k < k_0$ or p_{2d+1} if $k \geq k_{2d}$. Once we have found the corresponding leaf node, we look for an entry $k_i = k$. If it exists, the record associated with k is pointed by p_i, otherwise there is no record associated to k. In the previous description, keys and pointers are numerated as the scheme in Fig. 3.3 shows.

B$^+$-trees are very appropriate for increasing the performance of range search. The algorithm for searching all records whose keys are in the range $[k, K]$ is also quite straightforward. We search for the associated leaf node of k as if it were a single key search. Once the leaf node has been found, we return all pointers p_i fulfilling $k \leq k_i \leq K$. If the last key of the leaf node fulfills $k_{2d} < K$, we obtain the next leaf using the pointer p_{2d+1} and repeat the described procedure until we find an entry k_i such that $k_i >= K$.

B$^+$-trees applied to fuzzy indexing

In the 2BPT proposal, an interval is indexed by means of two B$^+$-tree structures. One B$^+$-tree indexes the values of the lower bound of the indexed intervals, whereas the other B$^+$-tree indexes the upper bound values.

The preselection row set for a flexible condition $\langle C, T \rangle$ is calculated by following these steps:

1. Apply the range condition $\sup(supp(\Pi_{A(r)})) \geq \inf(base(\langle C, T \rangle))$ from the preselection criterion of Eq. 3.13 (or $\delta_{A(r)} \geq l_{base(\langle C,T \rangle)}$ from Eq. 3.16 when working with trapezoidal possibility distributions and/or fuzzy sets) using the B$^+$-tree which indexes the upper bound values of data support. The set of rows satisfying this condition is named $U_{\langle C,T \rangle}$.

rowID	att	rowID	att
1	[8,10,10,21]	6	[10,12,12,14]
2	[42,46,47,51]	7	[26,28,30,32]
3	[1,9,15,19]	8	[51,57,60,60]
4	[41,50,55,57]	9	[18,22,27,37]
5	[38,46,47,47]	10	[25,35,39,45]

Table 3.2: Sample data table

rowID	$[\alpha_{att}, \delta_{att}]$	rowID	$[\alpha_{att}, \delta_{att}]$
1	[8,21]	6	[10,14]
2	[42,51]	7	[26,32]
3	[1,19]	8	[51,60]
4	[41,57]	9	[18,37]
5	[38,47]	10	[25,45]

Table 3.3: α_{att} and δ_{att} values for attribute *att*

2. Apply the range condition $\inf(supp(\Pi_{A(r)})) \leq \sup(base(\langle C, T\rangle))$ extracted from the preselection criterion of Eq. 3.13 (or $\alpha_{A(r)} \leq u_{base(\langle C,T\rangle)}$ from Eq. 3.16 when working with trapezoidal possibility distributions and/or fuzzy sets). This range condition will be processed by using the B^+-tree which indexes the lower bound values of data support. The resulting set of rows is $L_{\langle C,T\rangle}$.

3. The preselection row set $PS_{\langle C,T\rangle}$ comprises those rows which satisfy the two previous range conditions. This set is trivially calculated as $PS_{\langle C,T\rangle} = U_{\langle C,T\rangle} \cap L_{\langle C,T\rangle}$.

Example of exible query processing

This example shows how the result set of a flexible query is calculated using the previously described indexing technique.

Example data and indexes An example data table, whose data is shown in Table 3.2, is defined. This example table consists of two columns: *rowID* and *att*. The *rowID* column is the primary key of the table and stores integer values. The *att* attribute is a column of a fuzzy numerical type that stores fuzzy numbers defined by trapezoidal possibility distributions.

In order to show how the proposed indexing mechanism works, a 2BTP index is defined on the attribute *att* of the example table. Table 3.3 shows the support of the fuzzy values used to build the index structures. The two B^+-trees of order 2 (this order is arbitrarily chosen only for example purposes) indexing the supremum and infimum values of the support are shown in Figs. 3.4 and 3.5, where the pointers of leaf nodes points to the rows whose *rowID* corresponds to the value enclosed in parenthesis.

Query processing The example query, for which the steps of the query processing mechanism will be depicted, is as follows:

```
SELECT rowID,att FROM table WHERE FEQ(att,[20,25,35,40])>=0.3
```

This query includes the condition `FEQ(att,[20,25,35,40])>=0.3` which is defined on attribute *att* and whose threshold value is 0.3. When the query processor detects that there is an index defined on attribute *att*, it starts the query processing mechanism of the proposed indexing method by following these steps:

1. Search ranges calculation: the $l_{base(\langle C,T\rangle)}$ and $u_{base(\langle C,T\rangle)}$ values are extracted from the second argument of the fuzzy condition operator. The second condition operator (the restriction) is the trapezoidal fuzzy set `[20,25,35,40]`, therefore $l_{base(\langle C,T\rangle)} = 20$ and $u_{base(\langle C,T\rangle)} = 40$.

2. $L_{\langle C,T\rangle}$ calculation: $L_{\langle C,T\rangle}$ is quickly calculated by using the B$^+$-tree indexing the lower bound of the support, obtaining all rows which satisfy the condition $\alpha_{att} \leq 40$. In this example, the $L_{\langle C,T\rangle}$ set comprises the rows $\{3, 1, 6, 9, 10, 7, 5\}$.

3. $U_{\langle C,T\rangle}$ calculation: the $U_{\langle C,T\rangle}$ row set is calculated by means of range search on the B$^+$-tree indexing the upper bound of the support. This set contains all rows satisfying the $\delta_{att} \geq 20$ condition. The calculated $U_{\langle C,T\rangle}$ is the row set $\{7, 9, 10, 5, 2, 4, 8\}$.

4. Preselection row set calculation: this set is calculated as the intersection of $L_{\langle C,T\rangle}$ and $U_{\langle C,T\rangle}$. In this example, the preselection row set $PS_{\langle C,T\rangle}$ is $\{9, 10, 7, 5\}$

5. Query result calculation: the preselection set is filtered in order to select only those rows whose fulfilment degree satisfies the query threshold. Each row is retrieved and the condition is applied to calculate its fulfillment degree. The fulfilment degree for each row in the previously calculated preselection set is shown in Table 3.4. Finally, the query result consists of the rows $\{7, 9, 10\}$ (row 5 is excluded because its fulfilment degree is below the threshold).

3.6.2 Fuzzy indexing using a G-tree

The work [28] proposes that a G-tree [26] be used to index the lower and upper bounds of the interval representing the support of the possibility distributions which models the indexed data. This section introduces the basis of G-trees and their application for fuzzy indexing.

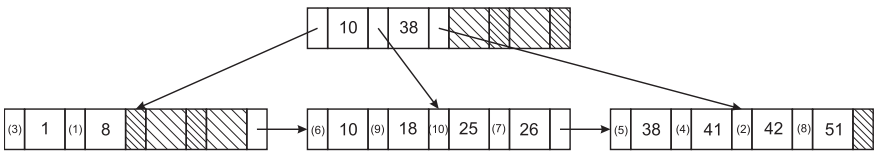

Figure 3.4: B$^+$-tree index ($d = 2$) for the infimum values of the support of *att*

rowID	$FEQ(att, [20, 25, 35, 40])$
5	0.1538
7	1
9	1
10	1

Table 3.4: Fulfilment degree for rows in the preselection set

G-trees

A G-tree is a combination of a B^+-tree and a grid file for indexing multidimensional data points. This index structure supports single point queries as well as range queries.

G-trees organize data into partitions, each corresponding to a fixed size data bucket. Initially, the D-dimensional data space is divided into two halves along the first dimension. Each data point inserted in the G-tree corresponds to one of the partitions and is stored in its associated data bucket. When a data bucket overflows, its associated partition is split into two halves along the $d+1$ dimension, where d is the dimension along which the parent partition was split. This division procedure is repeated, if necessary, until there is no overflow. Each resulting subpartition is associated with a new data bucket and the points of the original partition are stored in the bucket of their associated subpartition.

A partition is uniquely identified by a binary string. This string is created by appending the identifier of the parent of the partition and a 0, if the partition is the lower child of its parent, or 1, if it is the upper child of its parent.

Figure 3.6 shows the described partitioning and identification schema for bi-dimensional data and a maximum data bucket of 2 data points. Figure 3.6a represents the initial partitions with 3 data points. Figures 3.6b, 3.6c and 3.6d, respectively, show the resulting partitions after a split due to an overflow in the partition 0, 01 and 011.

Each time a data point is inserted, deleted or retrieved, the same procedure must be repeated:

1. Determine the partition to which the data point belongs.
2. Retrieve the associated data bucket of the partition, and insert, delete or retrieve the data point.

The first step is carried out by a simple geometric operation which does not require any data to be stored and can be quickly calculated. However, the second step requires an associative data structure. G-trees are in fact data structures designed for fast determination of the associated data bucket given a

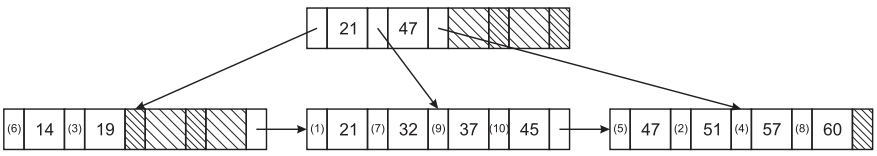

Figure 3.5: B^+-tree index ($d = 2$) for the supremum values of the support of *att*

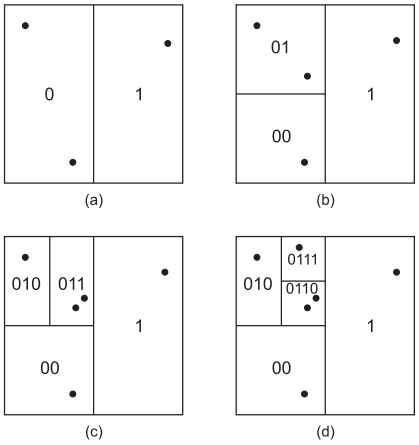

Figure 3.6: G-tree partitioning identification schema

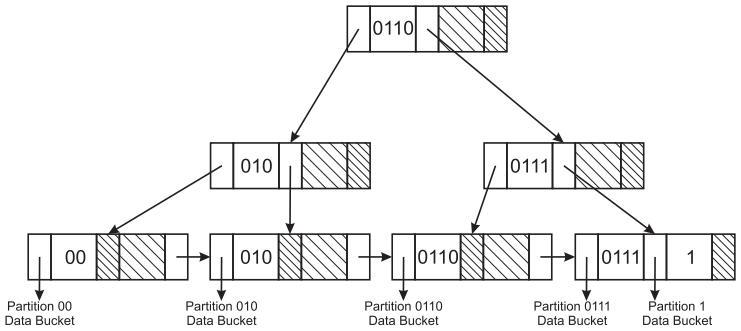

Figure 3.7: G-Tree for access to partitions buckets

partition identifier. They work like a B^+-tree where the keys are the partition identifiers. To enable the indexing of partition identifiers in a G-tree, an order criterion must be set for them. This criterion is defined in Eq. 3.17, where the partitions p_1 and p_2 have, respectively, an identifier of b_1 and b_2 bits long, $b = min(b_1, b_2)$, and $MSB(p, b)$ means for the b most significant bits of the identifier of partition p.

$$p_1 > p_2 \iff MSB(p_1, b) > MSB(p_2, b) \tag{3.17}$$

Figure 3.7 is a graphical representation of the G-tree used for indexing the data points (in fact for associating the partitions with their respective data buckets) represented in Fig. 3.6d.

We refer the reader to [26] for details on the insertion and deletion algorithm of G-trees.

G-trees are very suitable for processing range queries. This kind of query in the D-dimensional context is aimed at finding all data points inside a given hyper-rectangle q. This kind of query is resolved following these steps:

1. Determine the partitions p_l and p_u respectively associated with the lower point l and the upper point u in q.

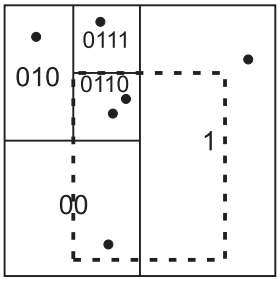

Figure 3.8: Range query

2. Determine the set of partitions $P = \{p_i : p_l \leq p_i \leq p_u\}$ using a range search similar to the one described for B^+-trees in the G-tree structure.

3. For every $p_i \in P$:

   (a) If the partition p_i is inside q, then every data point belonging to the partition belongs to the query result.

   (b) If p_i overlaps q, each data point belonging to p_i must be checked to be inside q. Obviously, only the data points inside q are part of the query result.

   (c) If the partition p_i is completely outside q, then it is discarded.

An instance of a range query (graphically represented by a dotted rectangle) on the G-tree indexing the data points included in Fig. 3.6d is shown in Fig. 3.8. The partitions p_l and p_u associated to the lower and upper point of the range query are $p_l = 00$ and $p_u = 1$. The set of partitions $P = \{p_i : p_l \leq p_i \leq p_u\}$, calculated by conducting a range query on the G-tree represented in Fig. 3.7, is $P = \{00, 010, 0110, 0111, 1\}$. Partitions 010 and 0111 are excluded from further processing as they are outside the range query. The data points on partition 0110 are included in the query results as they are inside the range query. The data points of partitions 00 and 1 are checked as they overlap the range query. The only data point in partition 00 is included in the query result because it is inside the range query, and the point in partition 1 is excluded as it is outside the range.

G-trees applied to fuzzy indexing

Following the approach proposed in [28], henceforth denoted GT, an interval (the support of an indexed fuzzy value) is mapped to a two-dimensional data point. For each interval $[l, u]$, a point (l, u) is inserted in the indexing structure. The preselection row set of a flexible condition $\langle C, T\rangle$, whose base is the interval $[l_{base(\langle C,T\rangle)}, u_{base(\langle C,T\rangle)}]$, is calculated by retrieving all entries in the index satisfying the range query $m \leq x \leq u_{base(\langle C,T\rangle)}, l_{base(\langle C,T\rangle)} \leq y \leq M$, where m and M are the minimum and maximum values of the underlying domain on which the fuzzy attribute is defined. This range condition is a translation of the preselection criteria of Eq. 3.13 (or Eq. 3.16 when working with trapezoidal possibility distributions) to this multidimensional mapping scheme.

3.7 Performance evaluation

It can clearly be seen that 2BPT doubles the overhead due to directory access (i.e. browsing non-leaf tree nodes) because it makes use of two trees. This should result in lower performance in contrast with GT, which uses only one tree. However, 2BPT does not su er from the performance degradation of GT due to low bucket usage caused by its partitioning method, which is extremely sensitive to data distribution. The stability of bucket usage of 2BTP may neutralize its overhead in the directory access. The counteraction of these factors might enable the proposed fuzzy indexing technique to perform in a similar way to the G-tree based technique.

In order to assess the signi cance of the di erence between 2BPT and GT performance, a quantitative performance evaluation must be carried out. This section describes the index performance measure used, the in uential factors on index performance which must be taken into account, and the experiment aimed to evaluate performance.

3.7.1 Performance measurement

The performance of an index may be measured as the saving in time which results when a query is processed using it, in contrast to the case when the query is processed using sequential access. Obviously, this time saving is inversely related to the time required to apply the indexing mechanism, which can be divided into the necessary time to access index data and the time to process this data.

Since the data of a large index is usually stored in secondary memory, the time needed to access this data is much greater than the time to process it. Besides, the computing power evolves faster than the secondary memory device bandwidth, and the rst depends much more strongly on the hardware features than the second. Therefore, the processing time is usually neglected in the index performance measurement context.

The time required to access the data is clearly dependent on the amount of data accessed. Since secondary storage devices are block-based, the data block is their information transfer unit instead of bytes or bits. Moreover, a DBMS is usually run on an operating system (OS), and its disk access mechanism must be taken into account. In order to increment disk performance, the OS usually includes cache techniques to reduce data time access, and implements a block access mechanism, the size of which is a multiple of the block size of the disk device.

Taking into account the previous remarks, in order to achieve an independent performance measure of the indexing techniques, a performance measure based on the number of data blocks accessed by the indexing technique would be appropriate. Any hardware or OS dependent techniques to reduce the access time are ignored because the amount of data, not the time taken to access it, is taken into account. Such performance measures should also take into account the cardinality of the query results as the processing of a high cardinality query would result in more index data accesses than the process of a low cardinality one.

A good performance measure candidate which meets the previous consideration would be the *index efficiency*. This performance measure is calculated as

shown in Eq. 3.18, where d is the minimum number of data blocks in which the result set can be fitted, and i is the number of blocks of index data accessed by the indexing technique. The measure considers the minimum number of data blocks which fits the query result, not the real number which is dependent on the physical layout of the data due to DBMS and OS policies. This measure therefore corresponds to the worst performance case.

$$eff = \frac{d}{d+i} \tag{3.18}$$

3.7.2 Influential factors for index performance

There are a large number of factors which would a ect the performance of the indexing techniques for fuzzy data. On one hand, there is the set of physical and logical factors related to the classical indexing techniques on which these indexes for fuzzy data are based. Although these factors could increase index e ciency when tuned, they are basically hardware or particular case dependent. Therefore, their study does not provide a good insight into the general performance of the considered indexing techniques under general conditions. On the other hand, there is a set of factors related to the indexed data and the processed queries which would a ect the index performance.

The employed indexing principle, on which both studied techniques are based, calculates the preselection set as the intersection of two intervals: the fuzzy data support and the flexible condition base. For this reason, as the extent of those intervals and the amount of indexed data increases, calculating their intersection becomes more complex.

In order to measure the extent of the support of a fuzzy set (or possibility distribution) on an ordered underlying domain, the formula shown in Eq. 3.19 is proposed, where $u_{supp(S)}$ and $l_{supp(S)}$ are the upper and lower bounds of the interval representing the support of the fuzzy set S, and D_M and D_m are the maximum and minimum value of the domain D on which the fuzzy set S is defined. For the purposes of this paper, the result in this formula would be called the *imprecision degree* of the fuzzy set S.

$$impr(S) = \frac{u_{supp(S)} - l_{supp(S)}}{D_M - D_m} \tag{3.19}$$

Although the imprecision factor of the fuzzy set modeling flexible queries is also an influential factor, it is not the only one related to the extent of the base of a flexible query. This extent is reduced as the threshold of the flexible query is increased. The speed of this reduction is determined by the sharpness of the transition of the possibility distribution from values of low possibility to high possibility values.

A possible way to measure the sharpness of fuzzy sets (or possibility distributions) on an ordered underlying domain is the one defined in Eq. 3.20, where $u_{supp(S)}$ and $l_{supp(S)}$ are the upper and lower bounds of the interval representing the support of the fuzzy set S and, $u_{core(S)}$ and $l_{core(S)}$ are the upper and lower bounds of the interval representing the core of the fuzzy set S, which is defined in Eq. 3.21. For the purposes of this paper, this value will be called the *fuzziness degree* of the fuzzy set S.

$$fuzz(S) = \frac{u_{core(S)} - l_{core(S)}}{u_{supp(S)} - l_{supp(S)}} \tag{3.20}$$

$$core(S) = \{d \in D, \mu_S(d) = 1\} \tag{3.21}$$

To sum up, the following five influential factors on the fuzzy index performance have been identified: the amount of indexed data, the imprecision of the fuzzy data, and the imprecision, the fuzziness and the threshold of flexible queries.

3.7.3 Experiments

In order to asses the global performance of the compared fuzzy data indexing techniques and their particular performance under di erent data and query scenarios, the same experiment using the same randomly generated data and queries has been conducted on both.

Di erent data scenarios have been taken into account by conducting the experiment on di erent data sets. In order to evaluate the influence of the amount of indexed data, the cardinality of data sets has been fixed to 6,250, 12,500, 25,000, 50,000 and 100,000 elements. The data of each is randomly generated using a uniformly distributed random generator within the interval $[-1000000, 1000000]$ ensuring that each data element has the same fixed imprecision and fuzziness degree for all the set. For each cardinality, 110 data sets have been evaluated, each with a fixed imprecision degree ranging from 0 to 0.9, and a fixed fuzziness degree ranging from 0 to 1 applying a 0.1 increment. The imprecision degree 1 is ignored because it means that all the randomly generated data are the same. A total of 550 data sets comprising 21,312,500 random fuzzy data elements in total are considered in this experiment.

Likewise, di erent flexible query scenarios have been considered. Each data set has been queried using randomly generated atomic flexible conditions. In order to ensure that the influential parameter spectrum has been equably considered, the fuzzy sets modeling the restrictions of the applied flexible conditions are generated ensuring a fixed imprecision and fuzziness degree ranging from 0 to 1 by applying an increment of 0.1. Similarly, the spectrum of threshold values is explored by fixing threshold values ranging from 0 to 1 and an increase of 0.1. For each fixed combination of imprecision, fuzziness degree and threshold, 20 atomic fuzzy conditions are randomly generated. A total of 26,620 randomly generated queries are applied to each data set, which results in an evaluation of 14,641,000 queries.

In order to isolate the experiment from physical and logical factors related to the underlying indexing techniques, both share the same fixed values. The bucket/node size has been fixed to 16kB which is the block size of the OS of the platform on which the experiments were run. For the index entries, a size of 8 bytes has been assigned to bucket/node pointers and partition identifiers, and an 8-byte floating point data type is used to represent numerical values. Each data element of the queried data set is considered to have an 8-byte primary key and a fuzzy numerical attribute represented by a trapezoidal possibility distribution of 32 bytes in length (4 floating point numerical values of 8 bytes). This configuration of the data set minimizes the row size which generates worse case performance measures.

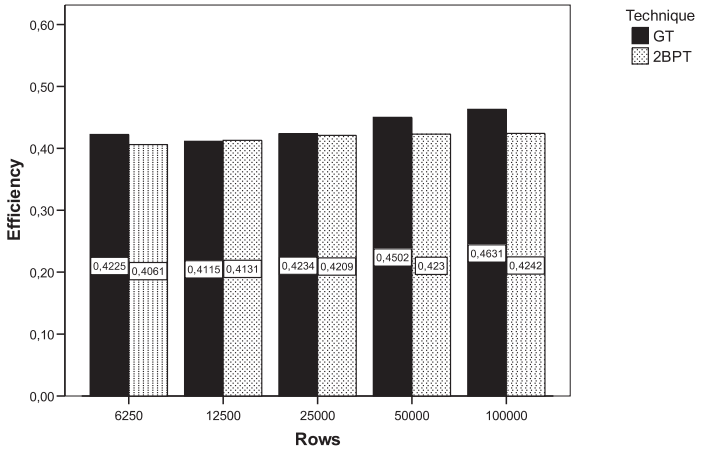

Figure 3.9: Comparison of e ciency under di erent database sizes

3.8 Results

The results from the previously described experiment yield a global approximation of the e ciency of the indexing techniques of 0.42 with a 0.9 standard deviation for 2BPT, and 0.43 with a 0.6 standard deviation for GT. This means an approximate 2% di erence.

The results have been analyzed in greater depth in order to assess the importance of the previously depicted influential factors. The first considered factor is the database size which results in bigger index structures and may result in an increase in the cardinality of the results. Figure 3.9 shows the mean e ciency of the compared techniques for di erent database sizes. The figure shows that even though the GT is slightly more e cient than the 2BPT, the 2BPT technique is quite similar. The fluctuation in the e ciency of GT is directly related to the fluctuation of block usage in its indexing data structure as will be shown below.

Another crucial factor for the index e ciency is the imprecision degree of the indexed data. Figure 3.10 shows the mean e ciency measured for di erent imprecision degrees of the considered data sets. In it, it can be seen that 2BPT has a similar (and in some cases greater) e ciency than GT for data sets comprising highly imprecise data. The e ciency is much di erent for data sets which consist of very imprecise data, especially when data sets are composed exclusively of crisp data, which is a very extreme case of a fuzzy database. Once again, a fluctuation in the e ciency of GT may be observed.

Figure 3.11 shows the aforementioned imprecision fluctuation of GT, contrasted with the stability of 2BPT. The block usage of GT is strongly data dependent, as this indexing technique does not ensure a minimum block usage value. In contrast, 2BPT ensures a minimum block usage of 0,5, which results in an average of 0.73 block usage, whereas an average of 0.52 block usage is measured for GT. Figure 3.12 shows the fluctuation of block usage in GT, which relates to the fluctuation in e ciency, in contrast with the stability of 2BPT block usage and hence e ciency.

With regards to the set of influential query factors, Fig. 3.13 shows the

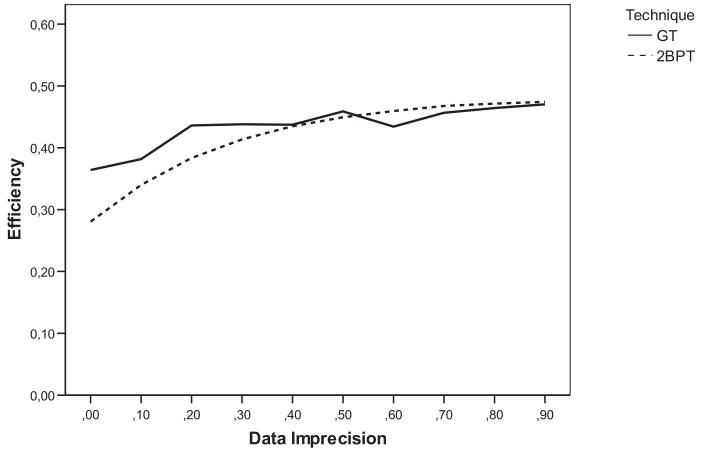

Figure 3.10: Comparison of e ciency under di erent data imprecision degrees

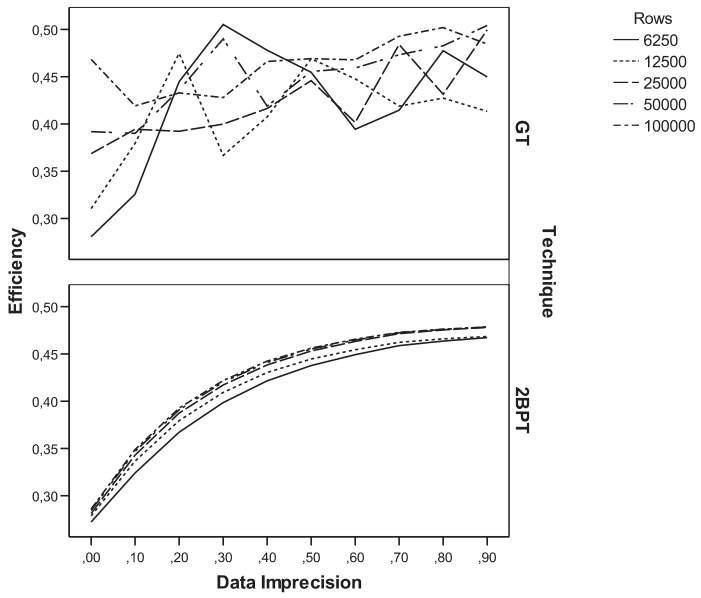

Figure 3.11: Comparison of e ciency stability under di erent database sizes and data imprecision degrees

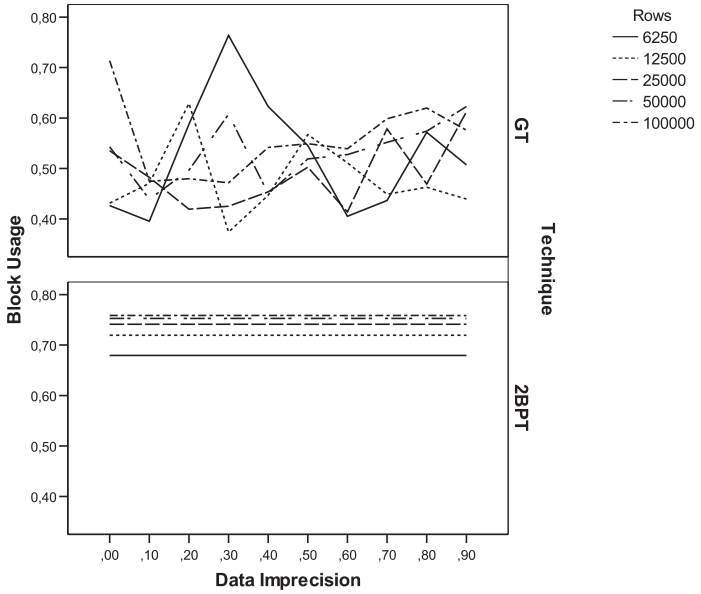

Figure 3.12: Comparison of block usage stability under di erent database sizes and data imprecision degrees

relation between query imprecision and index e ciency. It can be seen that 2BPT is more a ected by this factor than GT. Nevertheless, 2BPT shows a similar e ciency except for very low imprecision queries, and particularly for crisp ones.

Query fuzziness is another considered influential factor on index e ciency. Figure 3.14 shows the comparison of the e ciency of GT and 2BPT under di erent query imprecision conditions. A non-significant influence of this factor for 2BPT and an inappreciable influence for GT can be observed. Similarly, the query threshold has a slight influence on the e ciency of 2BPT, and practically no e ect on the e ciency of GT, as can be observed in Fig. 3.15.

3.9 Conclusion and outlook

In this paper, a new indexing technique, 2BPT, has been proposed. This improves query processing performance when a possibility measured atomic flexible condition is used to query fuzzy numerical data.

The e ciency of 2BPT has been contrasted with the e ciency of GT, which is, to the best of our knowledge, the only indexing technique suitable for handling the same studied data and query types. This comparison has been made under di erent data and query circumstances. Experimental results reveal a similar mean performance for both techniques. Moreover, 2BPT has proved to be a more stable indexing method than GT, which present instability issues relating to data distribution.

2BPT is based on a classical, well-known and spread indexing technique, B^+-tree, which is implemented and highly optimized in most, if not all, of the commonly used DBMS. GT, meanwhile, requires a specific data structure and

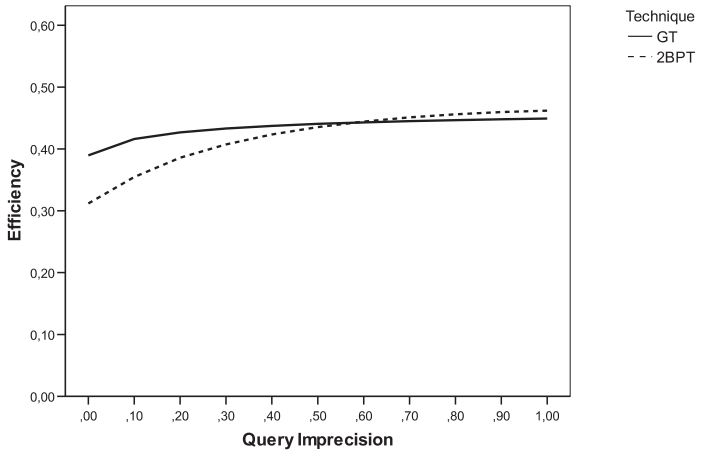

Figure 3.13: Comparison of e ciency under di erent query imprecision degrees

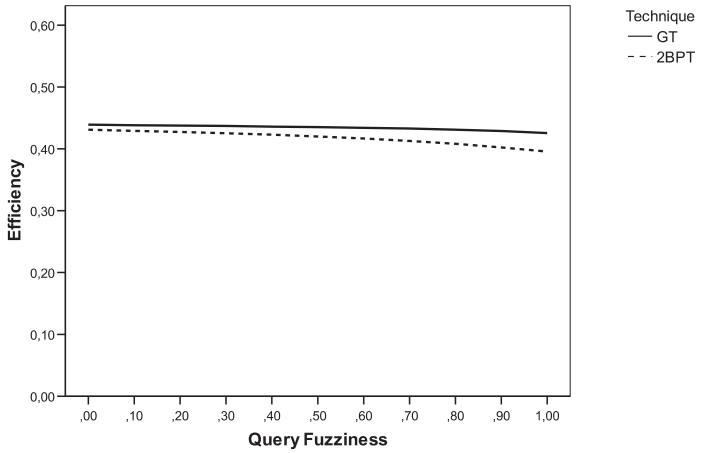

Figure 3.14: Comparison of e ciency under di erent query fuzziness degrees

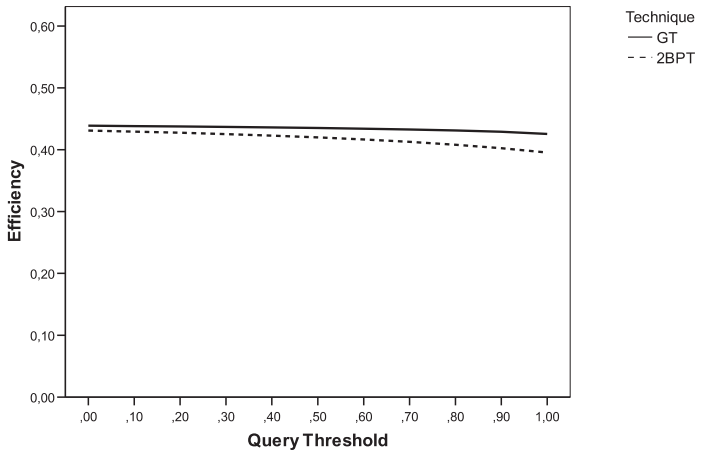

Figure 3.15: Comparison of e ciency under di erent query thresholds

algorithm implementation.

In our opinion, the slight di erence in e ciency and the enormous di erence in implementation e ort of GT in contrast with 2BPT makes the latter a good candidate for use in FDBMS built as an extension of a classical DBMS where the B^+-tree indexing technique is available. 2BPT o ers a practically immediate, or low implementation cost, indexing mechanism for fuzzy numerical data with a practically similar index e ciency.

Future work would focus on providing and studying indexing mechanisms (with low implementation cost if possible) for other necessity measured conditions and other imprecise, uncertain and inapplicable data types, such as scalar fuzzy data, fuzzy objects and fuzzy collections.

Chapter 4

A B^+-tree based indexing technique for necessity measured flexible conditions on fuzzy numerical data

4.1 Abstract

The paper proposes an indexing technique for fuzzy numerical data which increases the performance of query processing when the query involves an atomic necessity measured flexible condition. The proposal is based on a classical indexing technique for numerical crisp data, B^+-tree. Its e ciency is contrasted with another indexing method for similar data and queries, G-tree, which is specifically designed to index multidimensional data. Results show that the proposal performance is similar to and more stable than the reference technique.

4.2 Introduction

Database world has taken advantage of fuzzy set theory by using it as a way to manage imprecise, uncertain and inapplicable data (generally called *fuzzy data*) and to model and process flexible queries. As a result of this trend, there is a significant number of proposals on fuzzy database models [18, 39, 34, 49, 30, 10], flexible querying [7] and implementations of fuzzy database management systems (FDBMS) [24, 21].

Unfortunately, up to now FDBMSs are not generally integrated into real-world environments as the existing implementations do not provide the required performance. In fact, it is not the fault of implementations. The flexibility of fuzzy databases result in a increment of query results and makes the classical indexing techniques, which are the key for high performance in databases,

Extension of the original paper: Barranco, C. D., Campaña, J. R., and Medina, J. M. 2008. A B^+-tree based indexing technique for necessity measured flexible conditions on fuzzy numerical data. In Proceedings of the twelfth International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU) (Jun. 2008)

inapplicable. Although a great deal of research has been carried out into fuzzy database models, not so much work has been done into indexing mechanisms for e ciently accessing fuzzy data.

This paper proposes an indexing technique for fuzzy numerical data to improve query processing when a particular kind of mono-attribute flexible condition based on as the necessity measure is involved. The data structure and search algorithm of the proposed indexing mechanism is based on classical indexing structures for numerical crisp data, i.e. B^+-trees. This underlying indexing technique is not specifically designed to index fuzzy numerical data but it is a simple and well-optimized technique which is available in virtually every current DBMS. This near-to-general availability of B^+-tree indexing methods would result in a reduction of implementation, integration and optimization e orts to incorporate the proposed indexing technique to an FDBMS built on a crisp DBMS.

The paper is organized as follows. The concept of fuzzy numerical data and necessity measured flexible conditions in the context of the paper are described in Section 4.3. Section 4.4 briefly introduces related work on fuzzy data indexing and outlines the indexing principle on which the proposed and studied indexing techniques rely. Section 4.5 describes the proposed and considered indexing techniques. Section 4.6 presents the measures and procedures for evaluating the performance of the studied indexing techniques. Section 4.7 analyzes the performance results. Finally, Section 4.8 contains the concluding remarks and future works.

4.3 Basic concepts

Previously, the proposed indexing technique is described as a technique to improve the performance of the processing of queries on *fuzzy numerical data* when a *necessity measured atomic flexible condition* is involved. This section defines these concepts in the context of the paper.

A *fuzzy numerical value*, for the purposes of this paper, is a convex possibility distribution on an underlying domain in which a linear order relation is defined.

Also for the purposes of this paper, a *fuzzy condition* is a restriction imposed on the values of an attribute which contains a fuzzy numerical value for each row. This restriction is specified as a fuzzy numerical value to which the restricted attribute value must possibly or necessarily be compatible. This paper only focuses on necessity measured conditions since possibility measured conditions has been previously studied [3].

The necessity degree is called the fulfillment degree of the condition in the rest of the paper. This degree is computed as shown in eq. 4.1, where $D(A)$ is the underlying domain associated to the fuzzy attribute A, $\Pi_{A(r)}$ is the possibility distribution which describes the fuzzy value of the attribute A for the row r, μ_C is the membership function defining the fuzzy condition C and $\vee$ means for a t-conorm.

$$N(C/r) = \inf_{d \in D(A)} [(1 - \Pi_{A(r)}(d)) \vee \mu_C(d)] \tag{4.1}$$

A fuzzy condition is combined with a crisp relational comparator to set a threshold for its fulfillment degree. This threshold specifies the degree of

flexibility in which the fuzzy condition is applied, from 1 (no flexibility) to 0 (maximum flexibility). The typical expression for applying a threshold T is $N(C/r) \geq T$, except when the threshold is 0. In the latter case, $N(C/r) > 0$ is applied. The combination of a fuzzy condition with a threshold is called an *atomic flexible condition* for the rest of this paper and it is notated $\langle C, T \rangle$.

An example of a query including a necessity measured atomic flexible condition is the following:

```
SELECT * FROM tbl WHERE
NFEQ(f,[a,b,c,d])>=T
```

The previous query is expressed using the SDS [15] query language. It returns a table containing the rows from `tbl` table whose value for the attribute `f` is necessarily equal to the fuzzy numerical constant value represented as a trapezoidal possibility distribution [a, b, c, d], with a fulfillment degree which is greater than or equal to the threshold value `T`.

4.4 Related work

The works on fuzzy data indexing started, to the best of our knowledge, with the seminal paper [5]. It exposes the need for specific indexing techniques for fuzzy databases, and proposes two indexing principles for flexible querying using possibility and necessity measures on fuzzy attributes described as possibility distributions. For the sake of brevity this section only describes the indexing principle for necessity measured fuzzy conditions, on which this paper is focused. The reader is referred to [5, 3] for details on possibility measured ones.

The indexing principle allows the rows which do not satisfy an atomic flexible condition to be filtered out of a table. For necessity measured flexible conditions the article [5] proposes the one shown in Eq. 4.2, where, given a fuzzy set F, $S(F)$ means for the support of F and $S_\lambda(F)$ means for the λ-cut of F. It can be applied by indexing the cores of the indexed data.

$$N(C/r) > \lambda \rightarrow S_1(A(r)) \subset S_\lambda(C) \tag{4.2}$$

Since the publication of Bosc's seminal paper, some fuzzy data indexing techniques are proposed. The techniques [6, 33] and [43, 44, 45] are not based in the previous indexing principles and are designed for applications where the number of potential flexible conditions which can be used to build queries is finite and low. Whereas, [23] takes advantage of Bosc's indexing principles and supports any query but it is designed for low cardinality fuzzy data types. The limitations of the previous techniques make them unsuitable for indexing fuzzy numerical values.

The paper [28] proposes an indexing technique for convex possibility distributions defined on an ordered domain that relies on a crisp multidimensional indexing method. This technique is suitable for numerical fuzzy data indexing, on which this paper focuses, and it is described in the next section.

4.5 Considered and proposed indexing techniques

This paper contrasts the performance of two fuzzy data indexing techniques based on the previously described indexing principle. Both tackle the problem of indexing the core of fuzzy data, which in fact is a closed interval, by mapping it to a point in a bidimensional space. This way, for each interval $[l, u]$ a point (l, u) is inserted in the indexing structure.

Given a necessity measured flexible condition $\langle C, T\rangle$, its preselection row set is obtained by retrieving all entries (x, y) in the index satisfying the range query $l_{S_T(C)} \leq x \leq u_{S_T(C)}, l_{S_T(C)} \leq y \leq u_{S_T(C)}$, where $l_{S_T(C)}$ and $u_{S_T(C)}$ are the lower and upper bounds of the interval corresponding with $S_T(C)$. For the rest of the paper $S_T(C)$ is called the *base* of the condition flexible condition $\langle C, T\rangle$. This range condition is a translation of the preselection criteria of eq. 4.2 to the described two-dimensional mapping scheme.

4.5.1 Fuzzy data indexing with a G-tree

Based in the previous proposal, [28] makes use of a G-tree [26] for indexing the bidimensional points representing the core of the possibility distributions modeling the indexed fuzzy data. A G-tree is a combination of a B^+-tree and a grid file for indexing multidimensional data points. This index structure supports single point queries as well as range queries. For the rest of the paper, this technique is called GT for the sake of conciseness.

4.5.2 Fuzzy data indexing with a B^+-tree

This paper proposes a technique that takes advantage of classical B^+-tree indexing structures [4] for indexing bidimensional points representing the core of the indexed fuzzy numerical data.

B^+-trees are indexing structures designed for one-dimensional data. In order to make them suitable to index multidimensional data some additional work is required. One possible solution is to reduce multidimensional data to a one-dimensional counterpart, and then use a B^+-tree to index it. This reduction can be made with the help of a space-filling curve [27], a kind of curve that induces a linear order for multidimensional spaces. Among the variety of space-filling curves, in this paper Hilbert curve is chosen. The decision is motivated by its good performance contrasted with other popular possibilities.

Processing a multidimensional range query using the combination of a B^+-tree and a space filling-curve is an iterative process that comprises several one-dimensional range queries on the B^+-tree. Each iteration means a one-dimensional range query for each segment of the space-filling curve that cross the multidimensional query region. Figure 4.1 illustrates this process, where the center of each square forming the gray grid represents one point in a two-dimensional space, the order induced by the Hilbert curve is represented by a continuos black line and the two-dimensional range query is represented by a dotted rectangle. In the figure it can be seen that the Hilbert curve enters and leaves the query area several times. In order to find the indexed points included in the query area it is necessary to perform a one-dimensional range query on the B^+-tree for each curve segment inside the query region. For instance, for

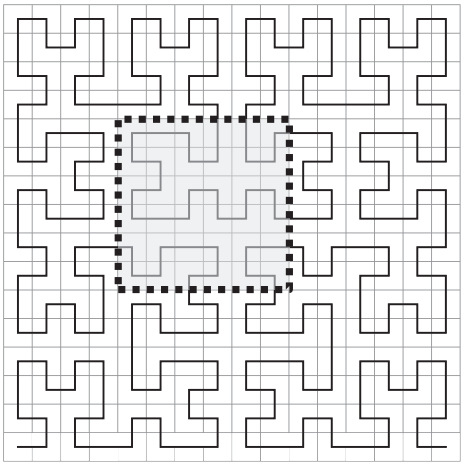

Figure 4.1: Number of query results and database size correlation

the case of the query region shown in the figure it is necessary to perform 4 one-dimensional range queries on the B^+-tree.

If the described method to solve multidimensional queries is applied directly it is unpredictably inefficient, as the number of one-dimensional queries necessary to perform varies for each multi-dimensional range query and so the index efficiency. This decreases as the number of necessary one-dimensional queries increases. This efficiency decrement is caused because each time a one-dimensional query is performed most of the non-leaf nodes of the B^+-tree must be read again. In order to solve this problem we propose to make use of a non-leaf node cache that will maintain only one node for each non-leaf level of the index. Taking into account that one-dimensional queries are applied following the order induced by the Hilbert curve, there is no need for a bigger cache. For instance, a 10,000,000 elements database of the same characteristics of the ones used in the experiments described later only requieres to maintain 2 non-leaf nodes in the worst case. Throughout the remainder of this paper this proposed indexing technique is called HBPT.

4.6 Performance evaluation

As it is remarked previously, HBPT is based on an one-dimensional indexing scheme, so this should result in lower performance in contrast with GT, which uses an specific multidimensional scheme. However, HBPT does not suffer from the performance degradation of GT due to low bucket usage caused by its partitioning method, which is extremely sensitive to data distribution. In fact, the stability of bucket usage of HBPT may neutralize the disadvantages due to its non multidimensional specific indexing scheme. This might enable the proposed fuzzy indexing technique to perform as well as GT.

In order to assess HBPT and GT performance, a quantitative performance evaluation has been carried out. This section describes the index performance measure used in this evaluation, the influential factors on index performance which has been taken into account, and the experiment designed to evaluate

performance.

4.6.1 Performance measurement

In order to measure the e ciency of the studied indexes independently of hardware and OS dependent factors [3] the index e ciency measure in equation 4.3 is used, where d is the minimum number of data blocks in which the result set can be fitted, and i is the number of blocks of index data accessed by the indexing technique (i.e. the index overhead).

$$eff = \frac{d}{d+i} \tag{4.3}$$

Index performance is measured as the saving in time which results when a query is processed using it, in contrast to the case when the query is processed using sequential access. This time saving is inversely related to the time required to apply the indexing mechanism, which can be divided into the necessary time to access index data and the time to process this data.

The data of a large index is usually stored in secondary memory, thus the time needed to access this data is much greater than the time to process it. Besides, the computing power evolves faster than the secondary memory device bandwidth. Therefore, the processing time is commonly neglected in the context of index performance measurement and only the data access time is taken into account.

The time required to access data is clearly dependent on the amount of data accessed. Secondary storage devices are block-based, so the data block is their information transfer unit. The time required to access the data is composed by the time to read the data blocks and the required time to locate these data blocks. The time to read the data blocks and the number of data blocks accesses are proportional measures, but the time to locate the data blocks strongly depends on the distribution of the blocks in the storage device media and on the device technology to locate them. Moreover, a DBMS is run on an operating system (OS), and its disk access mechanism must be taken into account. OSs usually include cache techniques to reduce data time access so the time required to access the data strongly depends on the cache schemes and the cache status in a given moment.

In order to achieve an independent performance measure of the indexing techniques, taking into account the previous remarks, a performance measure based on the number of data blocks accessed by the indexing technique would be appropriate. Any hardware or OS dependent techniques to reduce the access time are ignored because the amount of data, instead of the access time, is taken into account.

Such performance measures should also take into account the cardinality of the query results as the processing of a high cardinality query would result in more index data accesses than the process of a low cardinality one.

The measure considers the minimum number of data blocks which fits the query result, not the real number which is dependent on the physical layout of the data due to DBMS, OS, cache and file system policies. Thus, this measure corresponds to the worst performance case.

4.6.2 Influential factors for index performance

Performance of indexing techniques for fuzzy data are a ected by a large number of factors. On one hand, a set of physical and logical factors related to the classical indexing techniques on which these indexes for fuzzy data are based must be taken into account. Even though the mentioned factors could increase index e - ciency when tuned, they are basically hardware or particular case dependent, so their study does not provide a good insight into the general performance of the considered indexing techniques under general conditions. Among these factors, the index block size is the most critical for the considered indexing techniques. On the other hand, there is a set of factors related to the indexed data and the processed queries which would a ect the index performance. These factors can not be tuned and would provide a good measure of index performance under di erent usage conditions.

The indexing principle on which both studied techniques are based calculates the preselection set as the set of fuzzy data elements whose core is contained inside the flexible condition base. The extent of these intervals and the amount of indexed data would a ect to the number of results for a query. Therefore, this would relativize the index overhead and a ect the index e ciency.

Both, the extent of fuzzy data core and the flexible condition base, are dependant of the shape of the fuzzy set modeling the fuzzy data element and flexible condition. This shape can be described as the extent of the support of the fuzzy set and the sharpness of its transition from its support to its core. The extent of the support means an upper bound for the flexible condition base and also for the fuzzy data core. The sharpness means the speed of reduction of the extent of the base of a flexible condition with respect to its threshold. This highlights that flexible condition threshold is also an influential factor. Additionally, the sharpness combined with the extent of the fuzzy set support determine the extent of the fuzzy set core. In order to generally describe these shape descriptors [3] *imprecision* and *fuzziness* degrees are proposed.

To sum up, the following six influential factors on the fuzzy index performance have been identified: the amount of indexed data, the imprecision and fuzziness of the fuzzy data and flexible queries, and the threshold of flexible queries.

4.6.3 Experiments

The same experiments using the same set of databases and queries have been conducted on both fuzzy data indexing techniques in order to asses their global performance and their particular performance under di erent data and query scenarios.

In order to isolate the experiment from physical and logical factors related to the underlying indexing techniques, both share the same fixed values chosen to generate worst case performance measures.

A test have been conducted on di erent randomly generated databases to evaluate the global performance of the evaluated indexes. The test data set is composed by 30 databases, each one randomly generated using a uniformly distributed random generator within the interval $[-1,000,000\ ,\ 1,000,000]$. The size of these databases has been fixed to 10,000, 20,000, 40,000, 80,000 and 160,000 elements.

The test query set is composed by 10,000 queries. Each flexible condition of the query test set has also been randomly generated using a uniformly distributed random generator within the interval $[-1,000,000\ ,\ 1,000,000]$.

In order to evaluate the index e ciency under di erent data and query scenarios, the same previous test have been conducted several times (once for a fixed value of each influential factor) on a modified test data set.

The original test data set has been modified by fixing the imprecision (in a first test) and fuzziness (in a second test) degree of the database elements. This way data elements are randomly generated except for the fixed factor.

For the first test, the imprecision degree has been fixed to values ranging from 0 to 0.9 by applying a 0.1 increment (i.e. 0, 0.1, 0.2, ..., 0.9). The imprecision degree 1 is ignored because it means that all the randomly generated data supports are the same as the extent of their support is equivalent to the extent of the underlying domain. A total of 10 test data sets composed by 30 databases (a total 18,600,000 randomly generated data elements) are considered in this test.

The same way, the fuzziness degree has been fixed to values ranging from 0 to 1 by applying a 0.1 increment for the second test. A total of 11 test data sets composed (a total 20,460,000 randomly generated data elements) are considered in this test.

In order to evaluate the influence of di erent query scenarios on index efficiency, the same global e ciency test has been carried out several times by applying each time a modified test query set.

In order to ensure that the influential parameter spectrum has been equably considered, the modified test query set has been generated by fixing the value of an influential factor, one factor for each modified test query set. This way, all the fuzzy sets modeling the restrictions of the applied flexible conditions of a test query set are randomly generated ensuring a fixed imprecision or fuzziness degree ranging from 0 to 1 by applying an increment of 0.1. Similarly, the spectrum of threshold values is explored by fixing threshold values ranging from 0 to 1 and an increase of 0.1.

As a result, 33 modified test query sets are randomly generated. Each query test is then applied on the randomly generated test data set to measure the average e ciency of each index.

The described tests consider a total of 330,000 randomly generated queries and 9,900,000 query evaluations when they are applied to the 30 databases included in the test data set.

4.7 Experiment results

Results from the previously described experiment yield an average e ciency of the indexing techniques of 0.45 with a 0.04 standard deviation for HBPT, and 0.44 with a 0.04 standard deviation for GT. The minimum di erence, which is within the standard deviation range, practically means a similar performance of both indexing techniques.

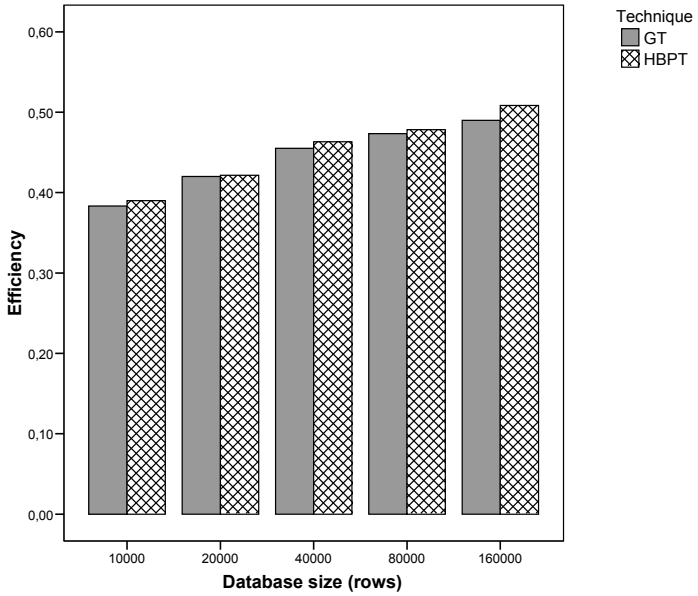

Figure 4.2: Comparison of e ciency under di erent database sizes

4.7.1 Influence of data related factors

The first considered influential factor is the database size which results in bigger index structures and may result in an increase in the cardinality of the results. Figure 4.2 shows the average e ciency of the compared techniques for di erent database sizes. The figures shows a similar performance for both indexing techniques, where HBPT slightly over-performs GT.

At first glance, it can be concluded that the larger the database size the greater the performance. This conclusion is counterintuitive as the larger the database size the greater the index overhead due to directory reads. Actually, a deep study reveals that the performance increment shown in fig. 4.2 is caused by an increase of the numbers of query results. The number of query results is directly related with the database size as the proposed test generates the test data set by randomly selecting values inside $[-1,000,000\ ,\ 1,000,000]$ whatever the database size is. This makes that the larger the database the greater the number of data elements inside a given interval (i.e. the greater the data density).

In conclusion, the results show that data density, not the database size, is an influential factor on index e ciency of both evaluated indexing methods. This is explained by the fact that a high data density means a greater number of query results that significatively reduces the impact of index overhead.

The results obtained by the previously described test for evaluating the influence of data imprecision on index e ciency are shown in fig. 4.3. In it, it can be seen that both indexing methods have a similar tendency, but also it can be noticed that HBPT is more stable than BT under data imprecision changes.

From the observed tendency, it can be concluded that the greater the data imprecision the smaller the e ciency. This is an expected behavior as the greater the support of data elements, the greater can be its core. Large cores of data

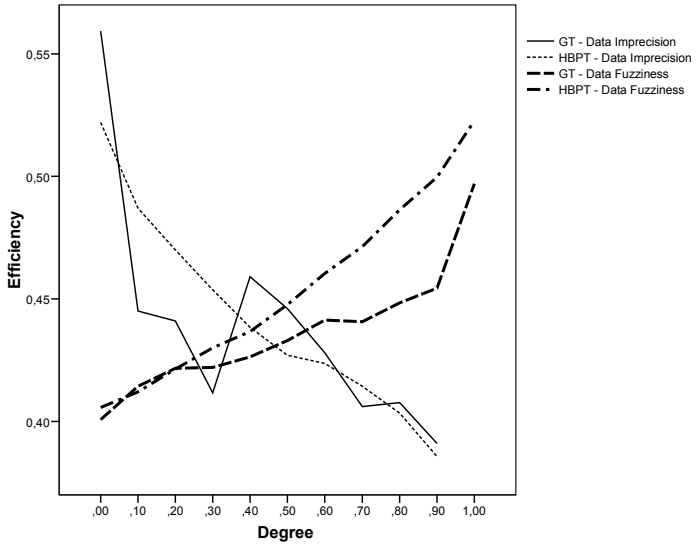

Figure 4.3: Comparison of e ciency under di erent data imprecision and fuzziness degrees

elements reduce the number of query results, as it is less probable given a query to found data elements whose core is contained in the condition base. Finally, less query results means a greater impact of index overhead that results in a smaller index e ciency.

The aforementioned e ciency fluctuation of GT is directly caused by the fluctuation of its block usage. The block usage of GT is strongly data distribution dependent, as this indexing technique does not ensure a minimum block usage value. In contrast, HBPT ensures a minimum block usage of 0.5. Figure 4.4 shows the fluctuation of block usage in GT in contrast with the stability of HBPT block usage.

As previous results show a significative influence of the imprecision degree of the data, and taking into account the imprecision degree of data only influences indirectly on the extent of the core of data elements, it is expected that fuzziness degree would be also significantly influential.

The results shown in fig. 4.3 confirm this conjecture. Results show for both indexing techniques that the greater the fuzziness degree, the greater the e ciency. This is explained by the increment of query results related to an increment of the fuzziness degree, which causes smaller cores for data elements and thus increment the number of data elements whose core is contained is the condition base of a given query.

One more time, a fluctuation of GT e ciency can be observed. This fluctuation is the result of a fluctuation of the block usage of GT, shown in fig. 4.4, that makes evident its sensibility to data distribution in contrast with HBPT stability.

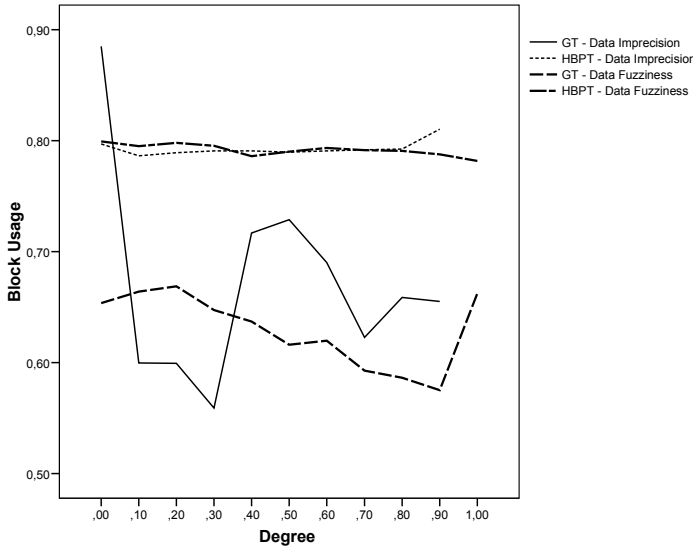

Figure 4.4: Block usage under di erent data imprecision and fuzziness degrees

4.7.2 Influence of query related factors

With regards to the set of influential query factors, fig. 4.5 shows the relation between query imprecision and index e ciency. It can be seen that both indexing techniques are a ected by this factor, as it is the lower bound of condition base extent, and so drastically determine the average number of query results. The e ciency is particularly low for the extreme case of a query imprecision of 0, which means a crisp condition. It can also be observed that the e ciency of GT does not grow at the rate of HBPT e ciency when query imprecision is high.

Query fuzziness is another considered influential factor on index e ciency. Figure 4.5 shows the comparison of the e ciency of GT and HBPT under different query fuzziness conditions. As the fuzziness degree indirectly determines the extent of the core of conditions, and given that the extent of the core of conditions means an upper bound for its condition base, the higher the imprecision the smaller the number of query results, and so the smaller the e ciency. Additionally, in the figure it can be seen that this factor is more influential for GT than for HBPT, specially for low fuzziness degrees where the di erence of e ciency is larger.

Finally, the observed results and influence of query threshold is the same as the influence of fuzziness degree (Not included in fig. 4.5 for the sake of clarity). In fact, both factors indirectly determine the average number of query results by reducing the average extent of the base of conditions.

4.8 Concluding remarks and future works

In this paper, a new indexing technique, HBPT, has been proposed and evaluated. It improves query processing performance when a necessity mono-attribute flexible condition is used to query fuzzy numerical data. The e ciency of HBPT

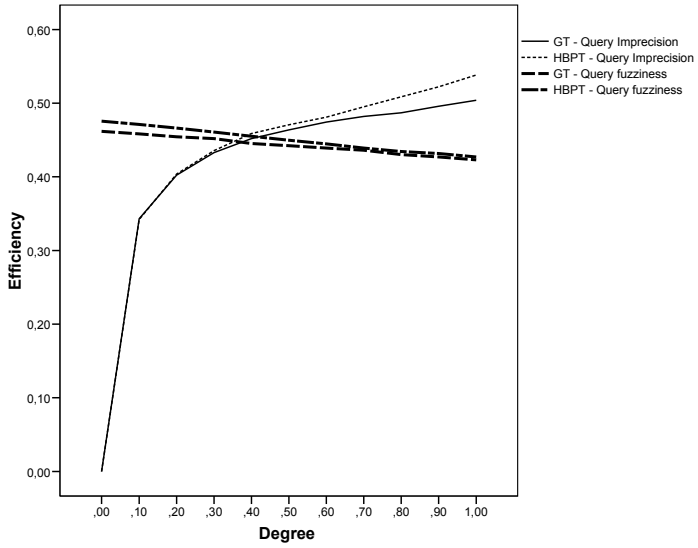

Figure 4.5: Comparison of e ciency under di erent query imprecision and fuzziness degrees

has been contrasted with the e ciency of GT, which is, to the best of our knowledge, the only indexing technique suitable for handling the same studied data and query types.

Experimental results reveal a similar average e ciency for both GT and HBPT. HBPT has proved to be a more stable indexing method than GT, which present instability issues relating to data distribution. It makes GT more affected by data related factors. Moreover, HBPT is less a ected by the studied query related factors.

In our opinion, the similar e ciency of both methods, the greater stability of HBPT and the di erence in implementation e ort of GT in contrast with HBPT makes the latter a good candidate for use in FDBMS, specially for these built as an extension of a classical DBMS where the B^+-tree indexing technique is available.

Future work will focus on providing and studying indexing mechanisms (with low implementation cost if possible) for other imprecise, uncertain and inapplicable data types, such as scalar fuzzy data, fuzzy objects and fuzzy collections.

Chapter 5

Conclusions

This work has proposed an object-relational fuzzy database model, an architecture and an implementation, in the form of database management system and as an extension of a current ORDBMS. Additionally, a dominant color image retrieval application have been presented in order to prove the utility and validity of the proposal. Finally, several indexing techniques have been proposed to improve the performance of flexible query processing on numerical fuzzy data in these kind of database management systems.

The proposed fuzzy object-relational model keeps the modeling power of previous fuzzy relational models and it is able to manage new kinds of fuzzy data resulting from the extension of the basic data constructs of ORDBMS. This makes model able to represent complex data composed of fuzzy and crisp values, and fuzzy collections of data with conjunctive and disjunctive semantics. Additionally, an implementation has been proposed. Its data types can be seamlessly integrated in an ORDBMS by making use of its native extension mechanisms.

In order to improve query processing performance when a possibility measured atomic flexible condition is used to query fuzzy numerical data the indexing technique 2BPT has been proposed. The technique has proved to be a more stable indexing method than its competitor.

Finally, a new indexing technique, HBPT, has been proposed to improve query processing performance when a necessity mono-attribute flexible condition is used to query fuzzy numerical data. HBPT has proved to be a more stable indexing method than its competitor and it is less a ected by the studied query related factors.

Bibliography

[1] D. Barbara, H. Garcia-Molina, and D. Porter. The management of probabilistic data. *IEEE Transactions on Knowledge and Data Engineering*, 04(5):487 502, 1992.

[2] C. D. Barranco, J. R. Campana, and J. M. Medina. On an indexing mechanism for imprecise numerical data for fuzzy object relational database management systems. In *Proceedings of 11th Int.l Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU)*, volume 2, pages 2205 2212, 2006.

[3] C. D. Barranco, J. R. Campana, and J. M. Medina. A B+-tree based indexing technique for fuzzy numerical data. *Fuzzy Sets and Systems*, 159(12):1431 1449, 2008.

[4] R. Bayer and E. M. McCreight. Organization and maintenance of large ordered indexes. *Acta Informatica*, 1(3):173 189, 1972.

[5] P. Bosc and M. Galibourg. Indexing principles for a fuzzy data base. *Information Systems*, 14(6):493 499, 1989.

[6] P. Bosc and O. Pivert. Fuzzy querying in conventional databases. In *Fuzzy logic for the management of uncertainty*, chapter Fuzzy querying in conventional databases, pages 645 671. John Wiley & Sons, Inc., 1992.

[7] P. Bosc and O. Pivert. SQLf: a relational database language for fuzzy querying. *IEEE Transactions on Fuzzy Systems*, 3(1):1 17, 1995.

[8] P. Bosc and H. Prade. An introduction to the fuzzy set and possibility theory-based treatment of flexible queries and uncertain or imprecise databases. In *Uncertainty Management in Information Systems*, pages 285 324. 1996.

[9] B. Buckles and F. Petry. A fuzzy model for relational databases. *Fuzzy Sets and Systems*, 7:213 226, 1982.

[10] R. Caluwe de, editor. *Fuzzy and Uncertain Object-oriented Databases (Concepts and Models)*, volume 13 of *Advances in Fuzzy Systems - Applications and Theory*. World Scientific, 1997.

[11] R. Cattell and D. Barry. *The Object Data Standard: ODMG 3.0*. Morgan Kaufmann Publishers Inc., 2000.

[12] J. Chamorro-Martnez, J. Medina, C. Barranco, E. Galán-Perales, and J. Soto-Hidalgo. Retrieving images in fuzzy object-relational databases using dominant color descriptors. *Fuzzy Sets and Systems*, 158(3):312 324, February 2007.

[13] E. F. Codd. Extending the database relational model to capture more meaning. *ACM Trans. Database Syst.*, 4(4):397 434, 1979.

[14] D. Comer. Ubiquitous b-tree. *ACM Comput. Surv.*, 11(2):121 137, 1979.

[15] J. C. Cubero, N. Marín, J. M. Medina, O. Pons, and M. A. Vila. Fuzzy object management in an object-relational framework. In *Proceedings of X Intl. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU)*, pages 1767 1774, 2004.

[16] A. Eisenberg and J. Melton. Sql: 1999, formerly known as sql3. *SIGMOD Rec.*, 28(1):131 138, 1999.

[17] A. Eisenberg, J. Melton, K. Kulkarni, J.-E. Michels, and F. Zemke. Sql:2003 has been published. *SIGMOD Rec.*, 33(1):119 126, 2004.

[18] S. Fukami, M. Umano, M. Muzimoto, and H. Tanaka. Fuzzy database retrieval and manipulation language. *IEICE Technical Reports*, 78(233):65 72, 1979.

[19] J. Galindo, J. Medina, and M. Aranda. Querying fuzzy relational databases through fuzzy domain calculus. *International Journal of Intelligent Systems*, 14:375411, 1999.

[20] J. Galindo, J. Medina, J. Cubero, and M. Garca. Relaxing the universal quantifier of the division in fuzzy relational databases. *International Journal of Intelligent Systems*, 16(6):713 742, 2001.

[21] J. Galindo, J. Medina, O. Pons, and J. Cubero. A server for fuzzy sql queries. In T. Andreasen, H. Christiansen, and H. Larsen, editors, *Flexible Query Answering Systems*, volume 1495 of *Lecture Notes in Artificial Intelligence*, pages 164 174. Springer, 1998.

[22] J. Galindo, A. Urrutia, and M. Piattini. *Fuzzy Databases: Modeling, Design and Implementation*. Idea Group Publishing, Hershey, PA, USA, 2006.

[23] S. Helmer. Evaluating di erent approaches for indexing fuzzy sets. *Fuzzy Sets and Systems*, 140(1):167 182, 2003.

[24] J. Kacprzyk and S. Zadrozny. FQUERY for access: fuzzy querying for a windows-based DBMS. In P. Bosc and J. Kacprzyk, editors, *Fuzziness in Database Management Systems*, pages 415 433. Physica-Verlag, 1995.

[25] G. J. Klir. On fuzzy-set interpretation of possibility theory. *Fuzzy Sets and Systems*, 108(3):263 273, Dec. 1999.

[26] A. Kumar. G-tree: a new data structure for organizing multidimensional data. *IEEE Transactions on Knowledge and Data Engineering*, 6(2):341 347, 1994.

[27] J. Lawder and P. King. Using space-filling curves for multi-dimensional indexing. In *Advances in Databases 17th British National Conference on Databases, BNCOD 17 Exeter, UK, July 3–5, 2000 Proceedings*, volume 1832 of *Lecture Notes in Computer Science*, pages 20 35, 2000.

[28] C. Liu, A. Ouksel, P. Sistla, J. Wu, C. Yu, and N. Rishe. Performance evaluation of g-tree and its application in fuzzy databases. In *CIKM '96: Proceedings of the fifth international conference on Information and knowledge management*, pages 235 242, New York, NY, USA, 1996. ACM Press.

[29] M. Marín, J. Medina, O. Pons, D. Sánchez, and M. Vila. Complex object comparison in a fuzzy context. *Inf. and Software Technology*, 45(7):431 444, 2003.

[30] J. M. Medina, O. Pons, and M. A. Vila. GEFRED: a generalized model of fuzzy relational databases. *Inf. Sci. Inf. Comput. Sci.*, 76(1-2):87 109, 1994.

[31] N. Mouaddib. Fuzzy identification in fuzzy databases-the nuanced relational division. In *Uncertainty Modeling and Analysis, 1993. Proceedings., Second International Symposium on*, pages 455 462, 1993.

[32] R. Murthy, S. Sundara, N. Agarwal, Y. Hu, T. Chorma, and J. Srinivasan. Supporting ancillary values from user defined functions in oracle. In *Data Engineering, 2003. Proceedings. 19th International Conference on*, pages 151 162, 2003.

[33] F. E. Petry and P. Bosc. *Fuzzy databases: principles and applications*. International Series in Intelligent Technologies. Kluwer Academic Publishers, 1996.

[34] H. Prade and C. Testemale. Generalizing database relational algebra for the treatment of incomplete or uncertain information and vague queries. *Information Sciences*, 34:115 143., 1984.

[35] S. Shenoi and A. Melton. Proximity relations in the fuzzy relational database model. *Fuzzy Sets and Systems*, 31(3):285 296, July 1989.

[36] D. Sinha and E. R. Dougherty. Fuzzification of set inclusion: Theory and applications. *Fuzzy Sets and Systems*, 55(1):15 42, Apr. 1993.

[37] M. Stonebraker and D. Moore. *Object-Relational DBMSs: The Next Great Wave*. Morgan Kaufmann, 1996.

[38] V. Tahani. A conceptual framework for fuzzy query processing - a step toward very intelligent database systems. *Information Processing & Management*, 13(5):289 303, 1977.

[39] M. Umano. Freedom-O: A fuzzy database system. In M. Gupta and E. Sanchez, editors, *Fuzzy Information and Decision Processes*. North-Holand Pub. Comp., 1982.

[40] M. Umano and S. Fukami. Fuzzy relational algebra for possibility-distribution-fuzzy-relational model of fuzzy data. *Journal of Intelligent Information Systems*, V3(1):7 27, Feb. 1994.

[41] M. Vila, J. Cubero, J. Medina, and O. Pons. The generalized selection: an alternative way for the quotient operations in fuzzy relational databases. In B. Bouchon-Meunier, R. Yager, and L. Zadeh, editors, *Fuzzy Logic and Soft Computing*, pages 214 250. World Scientific, 1995.

[42] E. Wong. A statistical approach to incomplete information in database systems. *ACM Trans. Database Syst.*, 7(3):470 488, 1982.

[43] A. Yazici and D. Cibiceli. An index structure for fuzzy databases. In *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems*, volume 2, pages 1375 1381 vol.2, 1996.

[44] A. Yazici and D. Cibiceli. An access structure for similarity-based fuzzy databases. *Information Sciences*, 115(1-4):137 163, Apr. 1999.

[45] A. Yazici, C. Ince, and M. Koyuncu. An indexing technique for similarity-based fuzzy object-oriented data model. In H. Christiansen, M.-S. Hacid, T. Andreasen, and H. Larsen, editors, *Flexible Query Answering Systems*, volume 3055 of *Lecture Notes in Computer Science*, pages 334 347. Springer, 2004.

[46] L. Zadeh. Fuzzy sets. *Information and Control*, 8:338 353, 1965.

[47] L. Zadeh. Similarity relations and fuzzy orderings. *Information Sciences*, 3(2):177 200, Apr. 1971.

[48] L. Zadeh. A computational approach to fuzzy quantifiers in natural languages. *Computers and Mathematics*, 9(1):149 184, 1983.

[49] M. Zemankova and A. Kandel. *Fuzzy Relational Databases – A Key to Expert Systems*. TV Rheinland, 1984.