UNIVERSIDAD DE GRANADA
E.T.S. DE INGENIERÍA INFORMÁTICA

Departamento de
Ciencias de la Computación
e Inteligencia Artificial

TESIS DOCTORAL

Sistema de Gestión de Bases de Datos Relacionales Difusas Multipropósito. Una Ontología para la Representación del Conocimiento Difuso

Carmen Martínez Cruz

Granada, noviembre de 2008

Editor: Editorial de la Universidad de Granada
Autor: Carmen Martínez Cruz
D.L.: GR. 2741-2008
ISBN: 978-84-691-8242-0

Sistema de Gestión de Bases de Datos Relacionales Difusas Multipropósito. Una Ontología para la Representación del Conocimiento Difuso

memoria para optar al grado de

Doctor en Informática

EL DOCTORANDO

Carmen Martínez Cruz

DIRECTORES

Ignacio J. Blanco Medina

María Amparo Vila Miranda

Granada, 17 de Noviembre de 2008

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
E INTELIGENCIA ARTIFICIAL

E.T.S. de INGENIERÍA INFORMÁTICA UNIVERSIDAD DE GRANADA

Agradecimientos

En primer lugar me gustaría agradecer a Mª Amparo Vila por la apuesta que hizo al traerme de la Universidad de Almería para comenzar este periplo granadino. Gracias a ella y a sus brillantes ideas este trabajo se ha podido llevar adelante y muchos más continuarán esta línea de investigación abierta. Es una excelente directora y soy muy afortunada por haberla tenido.

A Nacho fundamentalmente le tengo que agradecer el haber encontrado además de un director de tesis, un gran amigo. Gracias a él, me plantee venir a Granada y nunca me sentí sola. Has sido generoso en todo, gracias.

A mis padres y mi hermana Eva que siempre me han apoyado y han sufrido conmigo estos largos años que ha durado la elaboración de este trabajo. Aquí por fin está el resultado de mi esfuerzo, gracias por el vuestro que ha sido mucho mayor que el mío.

A mis adorados Raúl y Miguel, vosotros me habéis apoyado desde el momento cero en que llegue a Granada, porque siempre habéis estado ahí con una palabra de aliento o un buen consejo, no lo olvidaré. Y a Yolanda, Cristina y Amador, que por extensión tanto me han aguantado y con más mérito porque no conocen cómo es esto.

A mis amigos del alma, Belén, Carlos Molina y Jesús Alcalá, me habéis dado la inercia para trabajar, la sabiduría para adaptarme y la energía para continuar. Vosotros sabéis mejor que nadie lo que me ha costado todo. Mil gracias por vuestro ánimo y apoyo. ¡Viva Mecenas!

A mis amigos de la Universidad de Jaén: Antonio Rueda, Carlos Porcel y Francisco de Asís, por haber aguantado conversaciones monotemáticas eternas acerca de mi tesis, por aconsejarme, ayudarme y animarme en este último año que tanta falta me ha hecho. Tengo mucha suerte de teneros entre mis filas. A Josema, por toda la ayuda que me ha proporcionado. Este trabajo sólo contiene una mínima parte de tu enorme talento. Además, quiero agradecer al resto de compañeros del Dpto. de Informática de la Universidad de Jaén, por apoyarme y hacerme la vida más fácil, en especial a Samu.

A María José, Olga, Juanmi, Juan Carlos, Nico, Fernando, Miguel, Carmen y Jesús Campaña, compañeros del grupo de investigación IDBIS que siempre han sido amables y serviciales conmigo, haciéndole sentir parte del grupo desde el primer día. Sois un gran ejemplo a seguir y me habéis permitido conocer que significa ser un buen investigador. Y a Dani especialmente, por la tranquilidad que me ha transmitido en estos últimos tiempos, has sido un gran

descubrimiento. Gracias.

También agradezco al Dpto. de C.C.I.A. de la Universidad de Granada, por el marco inmejorable que me ha proporcionado para empezar mi trabajo investigador.

No quiero dejar de mencionar a mis ex-compañeros de la Universidad de Almería, ellos -Samuel, Alfonso, Joaquín, Jorge y Paco- me motivaron para comenzar este camino y no lo he olvidado.

Finalmente, a mi querida prima May y a mis amigas de fuera de este ambiente universitario que son las que me hacen desconectar de este mundo de ceros y unos, recobrar la cordura y pasar muy buenos ratos: María Jesús, Rosa y Pilar, gracias por estar ahí.

Tendría que agradecer a algunas personas más que han pasado por mi vida colaborando de alguna forma a que yo realizara este trabajo. Aunque no os mencione aquí, os llevo en el corazón.

"Sólo cerrando las puertas detrás de uno

se abren ventanas hacia el porvenir"

Françoise Sagan.

A mi familia.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El Entorno y las Bases de Datos

Hoy en día encontramos una creciente necesidad de representar información, manejarla, explotarla y compartirla. Esta necesidad se debe a la enorme capacidad de acceso a la información que tenemos gracias a las múltiples fuentes que nos la proporcionan, desde los tradicionales Sistemas de Gestión de Bases de Datos (SGBDs) hasta las más actuales relacionadas con la Web (Internet, Web 2.0, la Web Semántica y la Web 3.0). Además dicha información se encuentra representada en formatos muy diversos, dependiendo del contenido de la misma y del soporte en el que se halla representada y almacenada.

Sin embargo aunque las nuevas tecnologías están facilitando el acceso a la información gracias a las clasificaciones y búsquedas por contenidos semánticos (la web semántica, ontologías, anotaciones, etc.), no se debe olvidar que los sistemas que por excelencia mejor gestionan la información son los SGBDs [Cod07] dado que han sido diseñados específicamente para ello. Estos SGBDs han desarrollado durante los siglos XX y XXI un gran número de avances a la hora de representar información de muy diversa índole. Se han desarrollado Sistemas de Representación de Bases de Datos Relacionales, Orientadas a Objetos o mixtas, Lógicas y/o Deductivas, para Minería de datos, de Grandes Volúmenes, Transaccionales, Multimedia, Temporales, Difusas, etc., todas ellas encaminadas a gestionar de manera eficiente cualquier tipo de información.

El Modelo Relacional, desde que fue presentado por Codd en [Cod70], se consideró como el modelo que más eficientemente representaba la información estructurada [Cod07], y se extendió tan rápidamente que hoy en día, a pesar de las múltiples propuestas que han surgido para representar

la información, sigue siendo uno de los más utilizados. Debido a ello, se han realizado un gran número de extensiones al Modelo Relacional, entre la que destaca la integración de la Lógica Difusa al modelo con objeto de representar valores imprecisos y flexibles [Bos88, Med94b, Gal99].

Sobre dicha extensión se han propuesto otras basadas, por ejemplo, en la representación de reglas lógicas sobre una Base de Datos Relacional Difusa (BDRD) [Bal84, Bla01] y la posibilidad de realizar deducciones usando mecanismos de inferencia clásicos extendidos.

Otro enfoque para la integración de más elementos al Modelo Relacional es el de la denominada *Segunda Generación de Minería de Datos (DBMining)*, que propone un Sistema de Minería de Bases de Datos y Descubrimiento de Información (*KDDB System*). Esta nueva tendencia, planteada por Imielinski [Imi96] fusiona dos disciplinas, los procesos de minería de datos con las Bases de Datos. Basado en esto, Carrasco et al. [Car03a] propone la integración de operaciones de Minería de Datos (DM) con una extensión difusa sobre los modelos relacionales de bases de datos.

En este trabajo de tesis se propone una *Arquitectura Unificada Multipropósito* donde se combinen la gestión de minería de datos [Car03a] con la representación de información imprecisa [Med94b, Gal99] y lógica [Bla01] para aumentar el potencial de las consultas sobre una base de datos. Con esta arquitectura se permitirá aumentar la escalabilidad del sistema y la capacidad operativa del modelo relacional, de forma que se puedan ejecutar consultas complejas mediante la combinación de las operaciones implementadas en el sistema, como puede ser la combinación de procesos de minería de datos sobre relaciones definidas sobre reglas lógicas.

No obstante no todo son ventajas en la unificación de las diferentes arquitecturas. La complejidad que puede adquirir el sistema para manipular toda esta información hace muy costosa su puesta en marcha. Dicha complejidad se origina debido a la gran cantidad de estructuras que se necesitan para gestionar los diferentes tipos de información y el gran coste que supone su comprensión para su posterior uso. Esta problemática hace plantearse la viabilidad de llevar este sistema a cabo o bien realizar algún proceso de ingeniería que permita la simplificación del sistema.

Otro inconveniente que surge a la hora de extender los SGBDR con otro tipo de representaciones es el de la estrecha dependencia que tiene cualquier nueva implementación con el SGBD en la que se esté realizan-

do. Hay que tener en cuenta que los SGBD además de hacer una representación particular del lenguaje SQL dependiendo de las estructuras que implementan, disponen de mecanismos de extensión propios en los que algunos incluso incorporan capacidades funcionales para poder ejecutar sus programas (como Oracle © con el PL/SQL y PostgreSQL © con el PL/pgSQL). Otros en cambio permiten ejecutar programas en lenguajes de programación genéricos como JAVA o C. Dicha situación obliga a los implementadores a evaluar si es conveniente realizar una extensión personalizada para cada sistema (en tanto que se ganaría en eficiencia) o utilizar una común (se ganaría en independencia del sistema). En este trabajo, también se propone una arquitectura unificada capaz gestionar diferentes SGBDs sean cuales sean sus particularidades. De esta forma, la elección sobre la implementación de la extensión al SGBD elegida no será determinante para que el sistema no pueda ser integrado con el resto y su información no sea definida o manipulada de forma ajena a dichas particularidades.

Las Ontologías

Por otro lado, nos encontramos con que los esquemas de Bases de Datos no se consideran información útil en la Web. En un entorno en el que las páginas Web son indexadas semánticamente gracias a que su contenido se representa mediante ontologías o anotaciones, las páginas que representan información consultada *ad hoc* sobre una Base de Datos (se trata de formularios o interfaces *back-end* que realizan consultas contra bases de datos bajo demanda), se quedan fuera de esta clasificación. Con este mismo problema también se encuentran otro tipo de BBDD, también accesibles al público a través de la web, carentes de un entorno específico para su acceso. La información de éstas BBDD estaría disponible a través del uso de aplicaciones genéricas, como el ISQLPlus © [Ora07]. La información de los esquemas de dichas BBDD (normalmente descritos en lenguajes como SQL o UML) serían muy útiles especialmente en casos como el segundo, dado que éstos datos no pueden ser integrados en entornos como la Web Semántica.

Actualmente las ontologías se han convertido en el sistema más usado de representación del conocimiento. Las ontologías [Her02] se están empleando en todo tipo de aplicaciones informáticas en las que es necesario definir concretamente el conjunto de entidades relevantes en un campo de aplicación determinado, así como las interacciones entre las mismas. Algunas ontologías se crean con el mero objetivo de alcanzar una com-

prensión del Universo del Discurso pertinente, ya que su creación impone una especificación muy detallada. Otras en cambio son creadas para un propósito general que está orientado a la construcción de una base de conocimiento que contenga el conocimiento humano necesario para hacer inferencias.

La aparición de la Web Semántica, como entorno que permite consultar información web a partir del contenido semántico que las páginas web contengan, ha contribuido al éxito de las ontologías, que han sido utilizadas como mecanismo preferido (que no el único) para representar dicha semántica. Las ontologías por consiguiente pueden ser representadas utilizando lenguajes computacionalmente comprensibles a través de la web, permitiendo así que la información que en ellas se halla definida sea más universal.

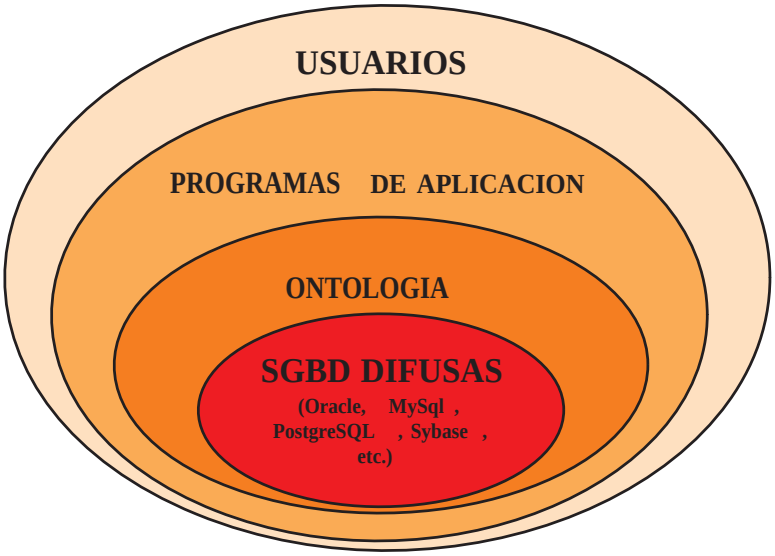


Figura 1.1: Relación de la Ontología con el Entorno

Esto nos lleva a plantear la representación de la *Arquitectura del SGBDRD Multipropósito* en forma de ontología como solución al problema de la complejidad. Dicha ontología permitirá la generalización y estructuración de los elementos que componen dicho *Servidor Multipropósito*, siendo a su vez independiente de las particularidades del SGBDR sobre el que estuviera desarrollada, y lo suficientemente clara y genérica para su manipulación y posible ampliación. La ontología se plantea así como

una capa abstracta que generaliza los conceptos representados a más bajo nivel por el SGBD, tal y como muestra la figura 1.1. El usuario final tendrá acceso a la ontología bien directamente, o bien a través de alguna herramienta desarrollada para esta tarea. Esta opción es la más deseable, dado que los lenguajes utilizados para la representación de ontologías no suelen ser fácilmente interpretables por humanos.

La Solución: una *Ontología para la Representación del Conocimiento Difuso*

Se plantea así el desarrollo una ontología para la representación del conocimiento difuso, entendiendo por este, aquel conocimiento impreciso representado mediante lógica difusa. Dicha ontología contendrá la definición de las estructuras y relaciones que permiten definir información imprecisa sobre un SGBDR (Sistema de Gestión de Bases de Datos Relacional) genérico, esto es, al margen de las particularidades que pueda tener un SGBDR concreto. La ontología actuará como interfaz entre el usuario y la base de datos haciendo transparente para el usuario la estructura de BD que permite almacenar la información difusa (información imprecisa representada mediante lógica difusa). De esta forma, únicamente se mostrará la representación que hace la ontología de la información del SGBDRD (Sistema de Gestión de Bases de Datos Relacional Difuso), tal y como podemos ver en la figura 1.2. De igual manera el usuario final podrá seguir accediendo a la información difusa tal y como ha ido haciéndolo, directamente sobre el SGBD Extendido. La única diferencia es que con esta propuesta se incrementan las posibilidades de comunicación con la misma.

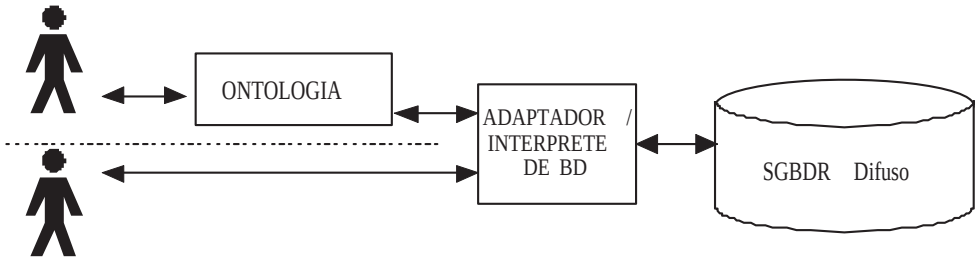


Figura 1.2: Interacción entre el Usuario y el SGBD

La *Ontología para la Representación del Conocimiento Difuso*, tal y

como se ha denominado a este primer prototipo, intentará unificar toda la representación del conocimiento difuso en un solo entorno, haciendo esta información portable a cualquier otro medio de representación, principalmente, a SGBDRs heterogéneos.

Además con esta representación se pretende dotar al usuario final de un mecanismo de definición y manipulación de de datos que facilite la gestión de información difusa sobre el cualquier SGBDRD. Por otro lado la operación de consulta también se definirá para guiar a usuario en la elaboración de la misma. En la figura 1.3 se muestra como un usuario puede utilizar una herramienta de consulta que opere contra el SGBDRD (Sistema de Gestión de Bases de Datos Relacionales Difusos) directamente o bien, realizar el proceso de consulta mediante el uso de la ontología. La diferencia entre ambas reside en que la primera interfaz es totalmente dependiente del SGBD contra la que la realiza y la segunda se ayuda de la ontología para generar la consulta, y no a partir de los datos que se hayan definidos en el SGBDRD.

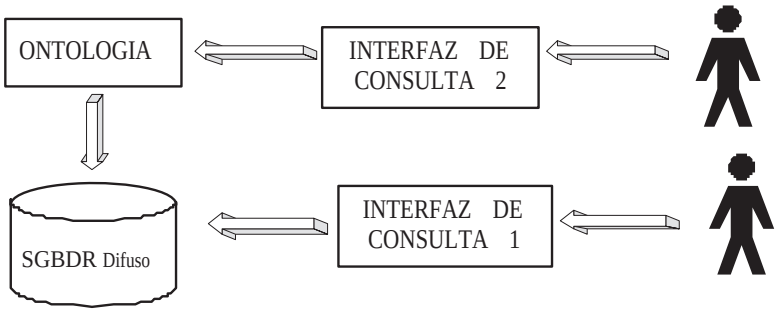


Figura 1.3: Interacción entre el Usuario y el SGBD para realizar las operaciones proporcionadas por la ontología

Objetivos

Los objetivos de este trabajo de tesis son los siguientes:

- Plantear una Arquitectura de un SGBDR genérica que permita combinar diferentes extensiones al modelo relacional, concretamente, para permitir representar información imprecisa, lógica y de DM en un único sistema.

- Definir una Ontología para representar el conocimiento difuso representado en un SGBDRD.
- Proponer una arquitectura del sistema que permita combinar información entre varios SGBDRD heterogéneos - cada uno con su propia representación de datos y su propio lenguaje en el caso que permitan capacidades funcionales.
- Aislar al usuario de las particularidades de representación de los SGBD en los que desee almacenar información.
- Facilitar al usuario la definición de información imprecisa mediante mecanismos o interfaces intuitivas.
- Permitir a los usuarios elaborar consultas en FSQL(Fuzzy SQL), extensión al SQL, para permitir manipular datos difusos sin tener cuenta las particularidades de dicho lenguaje, las particularidades de representación de los datos difusos o de los propios SGBDs
- Permitir la comunicación simultánea entre diferentes SGBDRD heterogéneos para definir esquemas o información difusa.
- Permitir la generación de las clases de catálogo de forma genérica, sin tener en cuenta las particularidades de los sistemas dónde se representen.
- Incorporar los esquemas de BD (Difusas) a la Web Semántica. Estudiar la relación de la ontología que representa el conocimiento difuso de un SGBDRD con el resto de estructuras que forman la web semántica.

Contenidos

El contenido de esta tesis esta estructurado de la siguiente manera:

- En el capítulo 2 se hace un repaso de algunas de las principales aplicaciones de las ontologías (descritas en detalle en el Anexo A). Concretamente se estudia la relación de las ontologías con la Web y con los Sistemas de Gestión de Bases de Datos, haciendo un estudio profundo de las diferentes propuestas que existen sobre esta ultima relación. Además se exponen las bases sobre las que se ha fundamentado la elaboración de la *Ontología para la Representación del Conocimiento Difuso* propuesta en esta tesis.

- En el capítulo 3 se describen brevemente los modelos de bases de datos extendidos que permiten manipular información difusa (expuestos con detalle en el Anexo B). Además se propone la Arquitectura de *Servidor Multipropósito* que combina dichas extensiones en un único modelo que permite mezclar todas las operaciones desarrolladas que utilizan información difusa multipropósito.

- En el capítulo 4 se describe la *Ontología para la Representación del Conocimiento Difuso*. Dicha ontología se plantea como una forma de solucionar el problema surgido en el proceso de unificación de las extensiones al modelo de bases de datos relacionales expuestas en el capítulo 3.

- En el capítulo 5 se establece la arquitectura del sistema para que la ontología establezca una comunicación con un SGBDR heterogéneo y permitir así la definición de datos y su posterior manipulación. Dicha arquitectura desemboca en el desarrollo de una interfaz de usuario, que se encuentra descrita también en en éste capítulo.

- El capítulo 6 termina con las conclusiones del trabajo realizado y las investigaciones futuras que se plantean a continuación del mismo.

Al final se incorporan varios anexos:

- En el Anexo A se muestra un estudio detallado del concepto de Ontologías. Este estudio hace un repaso por la definición de Ingeniería Ontológica, la clasificación de las ontologías, de las herramientas existentes para su explotación, de los lenguajes utilizados para su representación y de las diferentes operaciones que se llevan a cabo con ellas.

- En el Anexo B se muestra un amplio de resumen de las extensiones a los SGBD clásicos para la representación y manipulación de información imprecisa, lógica y de algunas técnicas de minería de datos. Este resumen va desde el planteamiento del modelo teórico hasta la descripción de las bases de metaconocimiento que permiten la puesta en marcha de estos sistemas en el modelo relacional.

- En el Anexo C se muestra la estructura de datos del ejemplo expuesto a lo largo de este trabajo de tesis.

Capítulo 2

Ontologías y Bases de Datos Difusas

2.1. Introducción

En este capítulo se hace un repaso por algunas de las principales aplicaciones de las ontologías en el campo de la representación de la información en la actualidad.

Por un lado, se describe el concepto de Web Semántica, analizando el impacto que tienen las ontologías sobre dicha Web. Se destaca la presencia de las Bases de Datos en la Web y las diferentes técnicas para el acceso a los datos de las Bases de Datos Difusas (BDD) que existen, para incorporarlas a la Web Semántica. Además se hace un repaso por el resto de tecnologías que están apareciendo en la Web y que también permiten representar información con cierta semántica.

Por otro lado, se trata la relación que existe entre el concepto de base de datos y el de ontología revisando las diferentes tendencias a la hora de considerar la representación llevada a cabo por una base de datos como si fuese una ontología. Se analizarán las diferentes representaciones de BBDD usando ontologías y el uso actual que tienen dichas representaciones.

Por último en este capítulo se expondrán dos ontologías, una desarrollada por Pardede et al. [Par05], que clasifica los tipos de datos predefinidos en el modelo relacional, y la ontología propuesta por Calero et al. [Cal05], que modela el ANSI SQL2003 [fSIIT03]. Estas dos ontologías establecerán los cimientos sobre los que se desarrollará la *Ontología de Representación del Conocimiento Difuso* base de esta tesis y expuesta en

detalle en el capítulo 4.

2.2. La Web Semántica

A pesar de que la Web ha traído con ella nuevas oportunidades para intercambiar, compartir, publicar y consultar información en la sociedad, también presenta sus limitaciones y desventajas. Por un lado, conforme ha ido creciendo su extensión y aumentando el número de páginas e información accesible a través de la misma, han ido cambiando las necesidades, haciéndose cada vez más importante disponer de buscadores o mecanismos de clasificación o acceso que permitan la obtención de información exacta, cada vez más cercana a la pregunta, para evitar así las cantidades ingentes de páginas como resultado. Otro problema de la Web se encuentra en la carencia de semántica en las respuestas obtenidas tras una consulta, obteniendo así respuestas imprecisas o erróneas [Lau04]. Además, la Web necesita representar información que pueda ser procesada computacionalmente [BL01]. Para ello requiere nuevas tecnologías que estructuren la información disponible como XML, XML-S, RDF(S), etc. Por contra, HTML la presenta de manera desorganizada y carente de significado. Sin embargo, en Web clásica, tal y como la conocemos, resumiendo lo dicho anteriormente:

- El contenido no puede ser determinado.
- Las consultas semánticas no pueden realizarse, puesto que las páginas Web no pueden ser interpretadas, y
- Los agentes inteligentes no pueden obtener información significativa.

La Web Semántica se propone como solución a todos estos problemas, y como muchos investigadores afirman [BL01, Gob03], será la tecnología capaz de hacer los contenidos de la Web comprensibles por humanos y procesables computacionalmente. Mas formalmente, la Web Semántica se puede definir como *el resultado de extender la Web estándar con lenguajes, información y recursos que nos permitan extraer información acerca del significado de los contenidos de la Web automáticamente* (Berners-Lee et al. [BL01]).

Estos contenidos se encuentran en diferentes formatos, por ejemplo en forma de documentos web, esquemas semiestructurados, o datos dinámicos [Hen02]. En la Web Semántica se extiende cada fuente de informa-

ción con una representación estructurada de su semántica. Existen varias aproximaciones para incluir esta semántica, la más utilizada, como se indica en [Fin05], son las ontologías, aunque también se ha propuesto el uso de anotaciones [She05].

Tal y como describimos en detalle en el Anexo A, una ontología es una descripción formal del dominio del discurso para un problema concreto, y la intención de la misma es ser compartida entre diferentes usuarios o aplicaciones. Una de sus ventajas es que puede ser expresada en un lenguaje (la mayoría en lógica descriptiva o de primer orden) de tal forma que pueda utilizarse para razonar [GP03b, Noy04, Sta04].

Por tanto esta primera aproximación para incorporar semántica a los contenidos de la Web consistirá en la incorporación de la ontología a la página web cuyo contenido esté describiendo, bien dentro del código de la web, bien adjuntándolo al resto de los archivos donde se encuentre la misma [Fin05]. Sin embargo McCool en [McC06] descubre ciertos problemas con esta solución (véase figura 2.1):

- La complejidad que adquiere la Web Semántica.
- La baja participación de los usuarios.
- El hecho de que en la actualidad exista un número muy escaso de aplicaciones.
- La complejidad de los lenguajes de descripción de ontologías.

La segunda solución presenta anotaciones acerca del contenido de la página Web y del vocabulario. Esta solución [McC06] reduce la complejidad de la Web Semántica, permite obtener más rápidos resultados en las consultas, y permite una mayor participación de los usuarios y desarrolladores. Sin embargo también tiene sus desventajas, no es tan expresiva como las ontologías a la hora de representar la semántica de una fuente de información (véase sección 2.2.2).

De cualquier manera, la Web Semántica se mantiene como alternativa a la Web clásica y permite que toda la información que en ella se encuentra pueda ser consultada y accedida.

2.2.1. Bases de Datos en la Web Semántica

Una parte importante de la información en la Web, la podemos encontrar en forma de documentos de texto (Word, PDF, txt,...), páginas HTML, documentos XML, páginas Web dinámicas, contenidos FLASH,

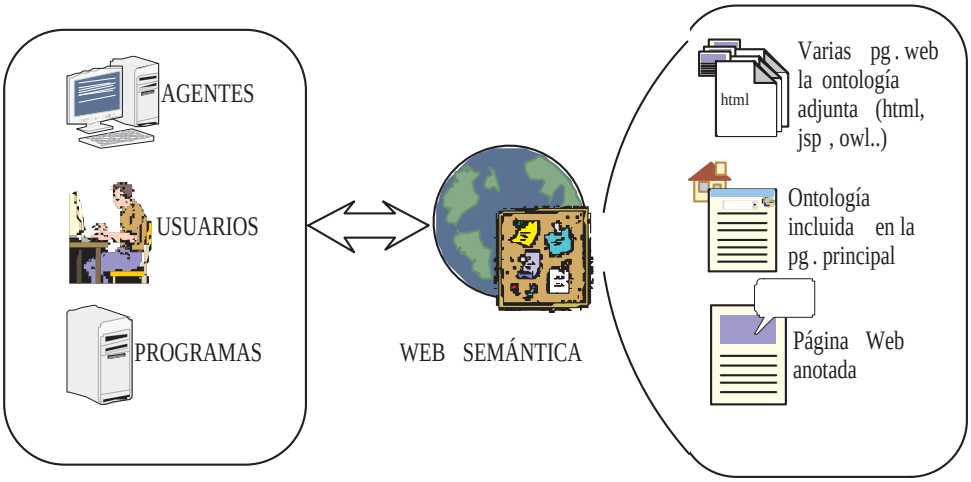


Figura 2.1: La Web Semántica, usuarios, y acceso a la información

librerías, ejecutables, interfaces de programas, bases de datos, formularios, etc. Incluso podemos encontrar simples datos (registros de datos, tuplas) o metadatos, podemos acceder a bases de datos, o inferir conocimiento a partir de estas (véase la figura 2.2), pero nosotros necesitamos definir las tecnologías que permitan acceder a toda esta información de la manera y formato que se requiera en cada uno de los casos.

Una página dinámica es un tipo de contenido Web que se genera mediante la consulta de una base de datos. Para generar un página dinámica, suelen utilizarse tecnologías como JSP, ASP, o PHP, para lanzar las consultas sobre dichas bases de datos. En estas páginas, la semántica no puede ser introducida, dado que se trata de un interfaz (*front-end*) para la base de datos. Sin embargo, podrían describirse semánticamente mediante los contenidos de la base de datos a la que ellas acceden [Jur07].

Existen incluso otros tipos de páginas web, que son más complejas aún para ser definidas de manera semántica, estas son por ejemplo, las interfaces Web genéricas para consultar bases de datos. Un ejemplo de este tipo de páginas es el ISQLPlus (Oracle ©) [Ora07] o el WebinTool [Hu96] o incluso, aquellas desarrolladas con paquetes de acceso a bases de datos como LISBDB [Eri07]. Estas páginas permiten acceder a la información de bases de datos, pero no pueden ser semánticamente indexadas porque sus contenidos no se conocen hasta que no se accede a una u otra base de datos.

En el contexto de la Web Semántica, y como caso particular de lo explicado en la anteriormente, sería interesante poder almacenar la semántica de dichos datos y emplear dicha semántica para su acceso. Por ejemplo si tratáramos de buscar registros de datos sobre una BD concreta o dominio particular usando ISQLPLUS(Oracle ©) [Ora07], el tener acceso a los esquemas de bases de datos que definen la información contenida en la misma resultaría muy útil. También sería interesante obtener entre los resultados de una búsqueda: referencias a bases de datos existentes que contengan entre sus datos información que cumpla el criterio de búsqueda, referencias a aplicaciones cliente existentes o formularios (una vez que los tenemos semánticamente declarados), etc. Entonces serán los usuarios los que deciden que resultados son los que necesitan.

Hay propuestas que intentan generar o extraer información de bases de datos relacionales, a partir de estas páginas en HTML, o de las páginas dinámicas. Por ejemplo Astrova [Ast04] construye estos esquemas de BD relacionales utilizando *wrappers* (programas para extraer dicha información). Otro autor que también utiliza *wrappers* para incorporar la semántica de las BDR a la Web es Champin et al. [Cha07] que propone un herramienta para construir en un lenguaje de la Web Semántica la información de la BDR.

Sin embargo, en las BBDD la dificultad para acceder a ellas está en función del tipo de información que representen. Si se trata de BBDD Extendidas (como una BD Difusa) entonces su representación no forma parte del estándar ANSI [fSIIT99], lo cual implicaría que se necesitara hacer pública la información acerca de los metadatos que representan la dicha información especial (por ejemplo la imprecisa), para garantizar el acceso correcto a los datos almacenados en la BD, tanto por parte de los usuarios como de los agentes que lo intenten [Bla05a, Bla05b].

Para realizar la tarea de publicación de contenido de una BD en la Web Semántica, existen numerosas aproximaciones. Una de las más utilizadas consiste en la representación formal del modelo relacional en forma de ontología, la cual actúa de interfaz entre el usuario y la BD real. Algunos de los autores que han llevado a cabo esta propuesta son LaBorda et al. [PdL05], Calero et al. [Cal06] o Blanco et al. [Bla05a] que presentan cada uno una interpretación del ANSI SQL como solución para que las bases de datos sean visibles a través de la Web (véase para mayor detalle sección 2.3.2). Esta interfaz mantendría separada la representación de los datos de su almacenamiento, y simplificaría la definición necesaria para acceder a la misma. En el caso de las BD Extendidas esta característica

es fundamental, dado que proporcionaría una definición pública de la estructura especial que tenga la información que representen, haciéndolas mas accesibles y comprensibles al usuario o agente final. La ontología resultante define las metaclases que definen la estructura de la información (el catálogo del sistema) que proporcionan la interfaz de acceso a los datos adecuada. Esta ontología entonces podría integrarse con el resto de las estructuras que se encuentran en la Web Semántica, comentadas con anterioridad. No obstante dicha representación será descrita con más detalle en los capítulos sucesivos.

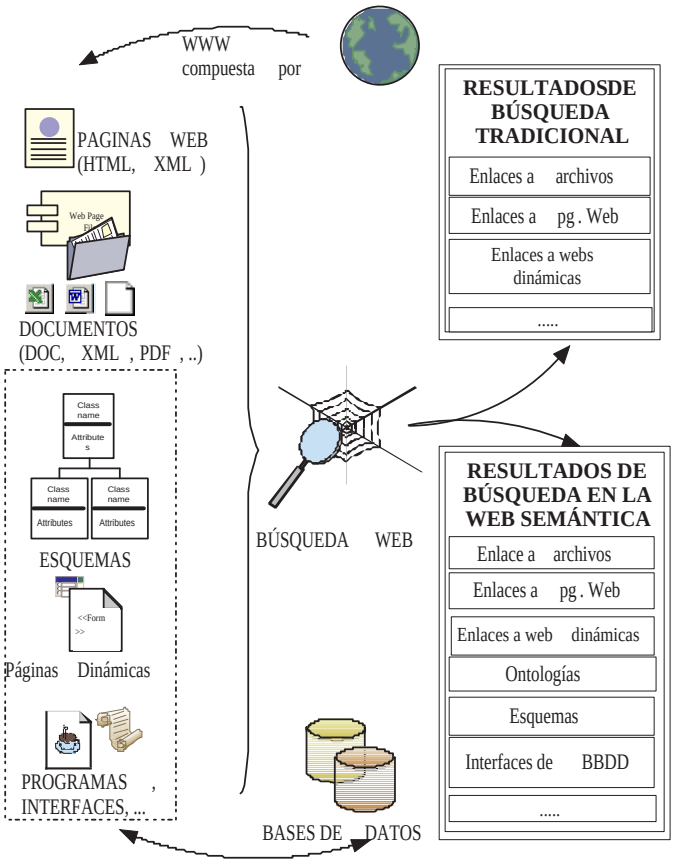


Figura 2.2: Comparación de documentos obtenidos de la web normal y de la web semántica

2.2.2. Hacia donde va la Web

Además de todo lo expuesto acerca de la Web Semántica, y dado que se está hablando de nuevas tecnologías, no se puede obviar la nueva tendencia de la Web que está en contraste con la Web Semántica, la Web 2.0. Esta tecnología que ha surgido en paralelo con la Web Semántica, está centralizada en los servicios que se presentan en la Web, al contrario que la Web Semántica que se centra en el contenido de la información. Es decir, la Web 2.0 aparece para satisfacer al usuario en las nuevas necesidades de comunicación que van surgiendo, como la mayor interacción entre los usuarios, el mayor uso de las redes sociales (*blogs, ebay, del.icious, wikipedia, youtube, myspace, etc.*), es decir, que demandan un mayor servicio. Las *folksonomias* (sería la traducción en español de *folksonomy*) se corresponderan así a la manera que tiene la Web 2.0 para etiquetar la información y así categorizar el contenido. Esta práctica es más sencilla que la generación de ontologías [Bre07] y evita tener expertos generando la semántica de una web concreta [Sha06a], puesto que son clasificaciones de contenido mediante etiquetas (*tags*) que van surgiendo de un trabajo colaborativo de los usuarios de la red. No obstante, los servicios que aporta la Web 2.0 no son incompatibles con la propuesta que da la Web Semántica, todo lo contrario, se complementan tal y como Ankolekar et al. en [Ank07] o Berners-Lee et al. en [BL06] proponen.

Por último, se debe señalar que existen un gran número de detractores de la Web Semántica, por los problemas expuestos anteriormente resumidos por McCool [McC05]. De entre estos problemas destaca de manera significativa el escaso número de aplicaciones de Web Semántica que están en uso actualmente, son conocidas o populares. Es un hecho que algunos autores, defensores de la Web Semántica, vaticinan que a la Web Semántica para su éxito total le quedan unos 5 años de desarrollos [Car07]. Otros, en cambio, califican este movimiento de "vaporware", es decir algo que desvía esfuerzos hacia un objetivo irreal. De cualquier manera, se considera que la Web Semántica se consolidaría como la Web 3.0. Sin embargo, cabe preguntarse si bien esta "nueva web" conseguirá triunfar, si aunará esfuerzos con la filosofía de la Web 2.0, si coexistirán las dos tecnologías en paralelo, o bien si tras estos cinco años, desaparecerá como otros tantos intentos de nuevas tecnologías, que a pesar de ser grandes ideas no han conseguido cuajar en la industria.

2.3. Ontologías versus Bases de Datos

El concepto de ontología en el campo de la ciencia de la computación se ha incorporado con mayor fuerza en las áreas de la inteligencia artificial y de las bases de datos. Concretamente el área de las bases de datos siempre ha tratado de modelar la información del mundo real, pero presenta una serie de restricciones, impuestas por el modelo de datos que se elija, para proporcionar la mayor eficiencia en el acceso y manipulación de la información.

De hecho, un gran número de estudios en el campo de las bases de datos han sido recuperados [Mee01a] para ser utilizados en desarrollos muy parecidos en el campo de las ontologías. Hay que observar que en el campo de las ontologías, existen los problemas de heterogeneidad de información, búsqueda de correspondencias, la combinación de esquemas, alineamiento, traducción, mezcla, etc. que tienen mucho parecido, por no decir que son idénticos a los problemas que surgen entre los esquemas de Bases de Datos [Ma05, Hai05, Men01, Sta04].

Sin embargo, existe un gran debate abierto en cuanto a la consideración de una base de datos (el esquema y su estado) como una ontología. Una ontología esta considerada como una representación de la realidad basándose fundamentalmente en el modelado de la semántica que representa. Para llevar esto a cabo, utiliza clases, propiedades, instancias, relaciones de agregación, generalización..., y sobre todo restricciones, que en la mayoría de los casos están representadas mediante lenguajes lógicos (lógica descriptiva o de primer orden), para poder añadir esa semántica al modelo. De esta manera una ontología no basa su representación en cómo la información será almacenada computacionalmente y por tanto es independiente del aspecto físico de su implementación [Bre04]. De acuerdo con esto, un esquema de base de datos podría verse como una ontología, puesto que también representan los hechos de mundo real, tienen su propia estructura de representación, e incluso restricciones que dan semántica al modelo. Es más, existen correspondencias directas entre una ontología y una base de datos, como por ejemplo que una clase puede ser una tabla, o que una propiedad se corresponde a un atributo, además de que también pueden modelarse relaciones de agregación, generalización, restricciones, etc. [Unc04] Sin embargo su comparación no es tan trivial.

A continuación veremos las diferencias que existen entre la representación de conceptos utilizando Ontologías y BBDD. Se analizarán sus

ventajas e inconvenientes, comparándolas a través de los siguientes puntos de vista:

- La semántica o conceptualización de la información que representan.
- El mecanismo de representación de los datos (tuplas/instancias) en las mismas.
- El modelo utilizado para la representación de la información.
- La eficiencia de uso.

Con respecto a la conceptualización de la información

Los teóricos de las ontologías afirman que una base de datos, dada la estructura de representación de su información y su finalidad, se correspondería con una ontología de *peso ligero* (*light weight ontology*), es decir, no sería una ontología propiamente dicha [GP03b, Bre04, Noy04], puesto que carece en su definición, entre otras cosas, de los axiomas que permiten realizar inferencias. Se considera que los esquemas de bases de datos están normalmente destinados a satisfacer los requisitos de una aplicación, mientras que las ontologías son fruto de un trabajo consensuado y deben ser compartidas entre toda la comunidad [Bre04, Unc04]. Además destacan que las ontologías no necesitan distinguir obligatoriamente entre los tipos de datos primitivos y complejos, que sus propiedades tienen mucha más semántica y que no necesitan realizar una normalización (gracias a todo esto se facilita la unión y compartición de las mismas).

En el área de las bases de datos, por contra, se considera que el modelado conceptual en el que está basada su representación [Spy02, Cul03, Mee01b, Rui06] proporciona una descripción mas rica semánticamente que la que da la lógica descriptiva en la que están basadas la mayoría de las descripciones de ontologías. No obstante, existen en la actualidad representaciones conceptuales de ontologías, como las que están basadas en *frames*, que son mucho mas intuitivas. Sin embargo, las ontologías requieren un nivel más alto de expresividad que el que le pueden dar los modelos conceptuales. Otros autores, como Jean et al. [Jea06] consideran que los modelos conceptuales no cumplen el criterio de la representación de un conocimiento consensuado y la capacidad de ser referenciada la información que se representa. Mylopoulos [Myl07] considera que una ontología no es un modelo conceptual, puesto que un ontología se considera reutilizable y un modelo conceptual lo es en menor grado.

De cualquier manera, las desventajas que puedan verse en cuanto a la representación de la realidad utilizando una base de datos, por su carencia a la hora de expresar axiomas para dar aún más semántica a la realidad que están representando, pueden verse resueltas por la incorporación de información lógica, gracias al uso de una Base de Datos Lógico-Deductiva. Es más, existen hoy en día un gran número de Sistemas de Bases de Datos dedicados al tratamiento de información de muy diverso tipo, desde BD Temporales, Espaciales, de Minería de Datos, Multimedia, Transaccionales, etc. Esto es más de lo que puede decirse del modelado de información en ontologías, que hoy en día esta carente de la posibilidad de asociar especificaciones temporales o espaciales [Cul03] a las mismas.

Con respecto a los datos que se representan

La representación de la realidad que hace una ontología, mezcla la información del esquema con los datos o instancias que este esquema pueda tener. De hecho, no es necesario que la ontología tenga datos. En cambio, en una base de datos hay una clarísima separación entre esquema y datos. La información del esquema se encuentra almacenada como tuplas en el diccionario de datos, de esta forma también puede ser consultada al igual que los datos en sí [Cul03, Unc04]. Gracias a la carencia de esta separación en las ontologías, es posible la definición de clases que puedan ser a la vez instancias, flexibilidad que jamás permitiría un modelo de bases de datos relacional (un modelo orientado a objetos puro si que lo permitiría). Sin embargo esta flexibilidad también traerá consecuencias a las ontologías, puesto que no se podrá hacer deducciones sobre aquellas ontologías que utilicen dicha funcionalidad.

En cuanto a la incorporación de las instancias, una ontología no sigue ningún tipo de patrón ni regla [Cul03]. Es más, en una ontología la definición de una nueva instancia no requiere que se cumplan las restricciones impuestas a la misma, son añadidas sin más. Contrariamente, el modelado de una base de datos requiere el cumplimiento de todos los requerimientos definidos sobre la misma para asegurar la integridad de la información (algunos autores consideran que el principal motivo de las BBDD es la integridad de los datos [Unc04]). De hecho, una tupla, en una BD Relacional, no podría ser incluida en la BD si no cumple todas las restricciones impuestas a la misma, incluyendo las restricciones semánticas del modelo (*CHECK Constraints*) además de las propias del Modelo Relacional (claves primarias, ajenas, nulos, etc.). Este hecho proporciona al Sistema de Bases de Datos la presunción de mundo cerrado

(aunque para sus detractores [Unc04] supondría una pérdida semántica en la representación de la información importante). Para solventar este problema en las ontologías, se ha de recurrir a los razonadores, que son los que determinarán qué instancias son las que pertenecen o no a la ontología donde están definidas. Por tanto, cada vez que se desee realizar dicha comprobación, habrá que lanzar el razonador sobre la ontología, cosa que en Bases de Datos no es necesario, dado que la información es consistente desde el principio.

Evidentemente los razonadores de ontologías sirven además para poder obtener nueva información además de la que ya se encuentra representada. Su analogía en el campo de las bases de datos se encontraría en los lenguajes de consulta como el SQL. La principal diferencia está en que los razonadores pueden encontrar esta información sobre las ontologías, independientemente de que tengan instancias definidas en ellas. Al contrario que las bases de datos, que únicamente obtendrán nueva información sobre la que ya está almacenada en ella, y por supuesto, sería imposible, obtener información mixta, es decir, de datos y de partes del esquema a la vez [Unc04]. Por supuesto, el razonamiento taxonómico es otra de las partes fundamentales de las ontologías.

Con respecto a la técnica de modelado

También habrá que tener en cuenta las diferencias que existen en el modelo de base de datos utilizado para su representación. El modelado lógico de una BD no representa la misma semántica que el modelado conceptual de la misma. La serie de limitaciones, dependiendo del modelo que se trate, proporcionará mayores o menores pérdidas semánticas a la representación. Así por ejemplo, un esquema entidad-relación extendido, o un diagrama UML, presenta una información como las relaciones es-un que se pierden al transformarse en el modelo relacional. Sin embargo la mayoría de estas transformaciones hacen perder muy poca semántica al modelo lógico que representan.

En las ontologías, estas pérdidas también ocurren dependiendo del lenguaje de representación utilizado. Así no será lo mismo representar una ontología mediante KIF, RDF, OWL (en cada una de sus modalidades), LOOM o utilizando cualquier herramienta de generación de ontologías, que haga su propia representación de la información. Además, cada lenguaje propone sus propias restricciones, provocando pérdidas semánticas a la hora de realizar traducciones entre las mismas. Esta desventaja no ocurre con las Bases de Datos y concretamente en el Mo-

delo Objeto-Relacional que utiliza siempre el mismo lenguaje estándar para su representación, el ANSI SQL [fSIIT99]. De cualquier modo es conveniente destacar que OWL esta tomando cada vez más fuerza en el campo de la representación de ontologías gracias a la aparición de la Web Semántica, pudiendo llegar a convertirse en estándar en un futuro no muy lejano.

Con respecto a la eficiencia en la representación

En cuanto a la ventaja de las bases de datos dada la gestión tan eficiente de la información que hacen, ha provocado que esta tecnología esté presente en el entorno de las ontologías, ya que no es lógico tener ingentes cantidades de datos almacenadas en forma de archivo de texto en formato OWL, o RDF. Es decir, las instancias donde se encuentra la información almacenada, deberían estar almacenadas en un entorno de bases de datos y ser entonces la ontología la que quedaría como una envoltura que permite acceder a esta información. En lugar de acceder al esquema de la BD, será la ontología la que proporcione la información para poder formular la consulta. Siguiendo esta idea existen numerosas propuestas que permiten acceder a la información que se encuentra en una BD (principalmente relacional debido a su hegemonía en el área de las BBDD), a través de una ontología de dominio. A continuación se presenta un gran número de estas propuestas, clasificadas en función de cómo es utilizada la ontología para representar o acceder a dicha información.

2.3.1. Comunicación entre Bases de Datos y Ontologías

La comunicación entre Bases de Datos y Ontologías sólo es posible si los esquemas de BD coinciden de alguna manera con las ontologías que representan dicha información. Para ello han surgido distintas propuestas que, según [Vys06] pueden categorizarse en:

Generar Descripciones de Ontologías y Esquemas de Bases de Datos utilizando la misma Técnica de Modelado

Se trata de propuestas generadas en esquemas conceptuales que pueden ser válidos tanto para bases de datos como para la definición de ontologías. De esta forma una ontología, podría ser traducida en cualquier otro lenguaje. Este es el caso de [Bro06], que usa UML para definir ontologías, pero es el menos frecuente, dado que la mayoría de las repre-

sentaciones de ontologías utilizan un modelo de representación específico para ellas, lo mismo que las bases de datos.

Generar el Esquema de Bases de Datos a partir de Ontologías

En este caso, una ontología ya existente sería la encargada de generar el esquema de bases de datos. Esta opción provoca, según algunos autores una gran pérdida semántica, dado que gran parte de la información inherente en la ontología se pierde en la traducción. Existen propuestas que llevan a cabo este proceso, como la que se encuentra en [Vys06] o en [Gal05b], en las que se implementan procedimientos que automáticamente generan esquemas de bases de datos a partir de una ontología en OWL. Existen otras propuestas como la de El-Ghalayini et al. en [EG06] que propone la obtención de un modelo conceptual a partir de una ontología.

Extraer o Representar la Descripción de la Base de Datos en forma de Ontología

Este último caso, consistente en generar una ontología a partir del esquema de bases de datos, es el más desarrollado en la comunidad, de hecho existen un gran número de propuestas de muy diversa índole llevándolo a cabo. Este proceso también se conoce como ingeniería inversa de bases de datos relacionales a ontologías [Ast04]. En él se permitirá el acceso a la información, esto es, a las instancias almacenadas en las bases de datos, a través de la ontología. Según Astrova [Ast05], existen 3 aproximaciones a la hora de hacer esta *Ingeniería Inversa*:

- *La basada en un análisis del esquema relacional.* Stojanovid et al. [Sto02] construyen reglas para mapear los constructores en el modelo de BD con su equivalentes constructores en la ontología. También Juric y Sckocir [Jur07] proponen unas tablas de mapeo para generar la ontología en OWL a partir de un esquema relacional, sin embargo esta propuesta luego se enriquece semánticamente con otras fuentes de información. Incluso en la propuesta de Champin [Cha07] se modela formalmente el modelo relacional, estableciendo las correspondencias con OWL para poder implementar una aplicación que obtenga las ontologías. El programa DataGenie [Gen07] desarrollado por Gennari et al. establece mapeos entre la ontología y la BD. Existen propuestas que incluso diseñan un lenguaje propio para declarar estos mapeos, como el Web-PDDL [Dou06], o el

lenguaje declarativo R20 de Barrasa et al. [Bar03] que representa las correspondencias entre el modelo de BD relacional y una ontología basado en la propuesta previa D2R MAP de Bizer [Biz03]. Por otro lado, hay otras propuestas que suben un nivel de abstracción, centrándose en el modelo conceptual de la base de datos relacional (el entidad relación), para obtener la ontología de dominio [Jea06] correspondiente en OWL. En [Upa05] y [Xu04] se propone incluso una herramienta propia y reglas de mapeo para obtener la ontología en OWL. A este nivel también se proponen lenguajes para establecer las correspondencias entre los dos modelos. Así a partir de Diagramas E/R o UML, se propone un nuevo lenguaje para la definición de la ontología, llamado el DLR-DB de Lubyte y Tessaris [Lub07].

- *Los basados en un análisis de los datos.* Construyen la ontología basada en un análisis del esquema relacional (también se analizan los datos para añadir semántica). Este es el caso de Astrova [Ast04] y Tijerino et al. [Tij05], este último, analiza los contenidos de las tablas para obtener una ontología. A través de los contenidos, podemos descubrir relaciones entre datos y restricciones y, a partir de ellas, descubrir correspondencias con otras ontologías ya realizadas o bien desarrollar una nueva propia. Otra aproximación que construye una ontología en OWL de manera semiautomática que se corresponde con el contenido de una BD relacional a partir del análisis de formularios en HTML la propuesta por Benslimane et al. [Ben06].

- *Las basadas en un análisis de las consultas de los usuarios.* Como la de Kashyap [Kas99], que construye una ontología basada en un análisis del esquema relacional. La ontología se refina con las consultas de los usuarios. Esta ontología no crea axiomas.

Sin embargo, algunos autores destacan los problemas que tiene este proceso de ingeniería inversa [Jur07]. La construcción de una ontología basada en un análisis del esquema relacional, afirman, puede estar limitado por la compleción de la información de entrada y la corrección de la misma. Dicha afirmación se basa en problemas como la falta de información en los esquemas, la creación de esquemas no normalizados, las pérdidas al traducir de los esquemas conceptuales a los relacionales, o el uso de nombres inapropiados en la representación de la información. No obstante, estos problemas no dejan de ser puntuales, y no suponen la mayor parte de las representaciones de bases de datos relacionales que son susceptibles de ser convertidas en ontologías.

Existen otras alternativas en la comunicación entre BD Relacionales y Ontologías, estas consisten en la utilización por parte de esta última del modelo relacional para poder "ser almacenadas". Estas propuestas olvidan la conceptualización o realidad que la ontología representa y que debería analizarse y modelarse a la hora de representar información en una BD, dado que el interés del uso de la BD únicamente subyace en la representación de la meta información que constituye la ontología. Es decir, la base de datos, representa la metaontología: clases, propiedades, instancias, restricciones, etc, y su tarea será la gestión eficiente de todas las ontologías que en ella se encuentren almacenadas. Estas propuestas reciben el nombre de modelos OBDB (*Ontologies Based DataBases*) y se definen *como los modelos de bases de datos que permiten almacenar la ontología y los datos en un modelo de datos común y único* [Jea06]. Jean et al. [Jea06] propone un modelo que separa la definición de la ontología y la de las instancias. La propuesta de Roldán-García et al. [Rol05] propone una herramienta para almacenar ontologías en OWL en un BD relacional, mediante el uso de archivos XML como archivos de configuración que podrán ser almacenados en cualquier RDBMS. Pan et al. [Pan03] almacena ontologías en un RDBMS de Access, creando una tabla para cada clase, o propiedad. La jerarquía de clases se almacena en el sistema utilizando vistas. Otro tipo de propuestas las representa como por ejemplo *Sesame* [Bro02, Kam07], se proponen una arquitectura desarrollada para un almacenamiento y consulta eficiente de datos en RDF y sobre todo independiente de cualquier sistema de almacenamiento. Este modelo propone una API (en Java), que permite acceder a los procedimientos que gestionan la información de la ontología. Hay otras muchas propuestas como esta, como la que proporciona JENA [Pro07] permitiendo la interpretación también de ontologías OWL.

2.3.2. Ontologías como Representación de Modelos de Bases de Datos

Otra tendencia que existe es la de generar ontologías que describen la conceptualización de una base de datos. De esta forma, la información que representan estas ontologías son metadatos, y desde este punto de vista, pueden ser consideradas estas ontologías como *Ontologías de Alto Nivel*. A partir de la definición de la ontología de alto nivel, la definición de los esquemas como instancias de la ontología será un paso sencillo, poniendo de esta manera a disposición del usuario la perspectiva de una base de datos como si se tratara de una ontología. Existen también un

gran numero de propuestas sobre esta idea: LaBorda et al. [PdL05] propone Relational.OWL, una ontología muy básica que describe el modelo de bases de datos relacional con el fin de poder compartir información entre bases de datos heterogéneas, utilizando OWL-Full para representar dicha ontología. En la misma linea que el anterior, Kupfer et al. [Kup06] presenta una ontología en OWL-Lite que denomina *ontología abstracta de bases de datos*, la cual permite representar un esquema de bases de datos, mediante la instanciación de dicha ontología. Trinh et al. [Tri06] también propone una ontología llamada OWL-RDBO que representa en OWL los elementos básicos de una BD relacional y las relaciones semánticas entre ellos. Calero et al. [Cal05, Cal06] también representa el modelo relacional de bases de datos de la manera más completa hasta la fecha, dado que describe íntegramente en el ANSI SQL 2003 [fSIIT03]. Sin embargo, utilizan una representación en UML para describirlo, junto con descripciones en OCL para completar su definición. En cuanto a propuestas software concretas existe *Ontobase* [Yab07], una herramienta que representa los contenidos de una base de datos automáticamente, a través de la herramienta de representación de ontologías de *Protégé*.

En definitiva, podemos considerar entonces una Base de Datos como una ontología, puesto que todas las carencias que pudiera presentar son solventables de una u otra manera. Esta propuesta, permite la definición de una BD a base de la instanciación de la ontología de alto nivel que representa la información del modelo de bases de datos relacional. Si se trata dicha información tal como es por naturaleza, esto es, como metadatos, entonces la instanciación de ciertas clases de dicha ontología deberá dar lugar a unas nuevas clases, que representan la información del esquema de la BD que es en ultima instancia lo que se desea representar. Por ejemplo, cualquier propuesta de ontología que represente el modelo relacional de bases de datos, contará con una definición de *Tabla* no como clase, sino como metaclase, dado que una tabla es la representación de la estructura de la información. De esta forma, la instancia de la clase *Tabla*, por ejemplo, con el concepto de *Personas* daría lugar a una nueva clase, la clase *Personas*, que sería en ultima instancia, la que albergaría los datos o información al respecto de la realidad que representa a través de sus instancias (se correspondería con las tuplas en el modelo relacional). En la figura 2.3 se puede visualizar este ejemplo, donde el fondo de color destaca la relación entre las metaclases y las instancias del esquema de BD generadas.

Las principales motivaciones que llevan al modelado del modelo de

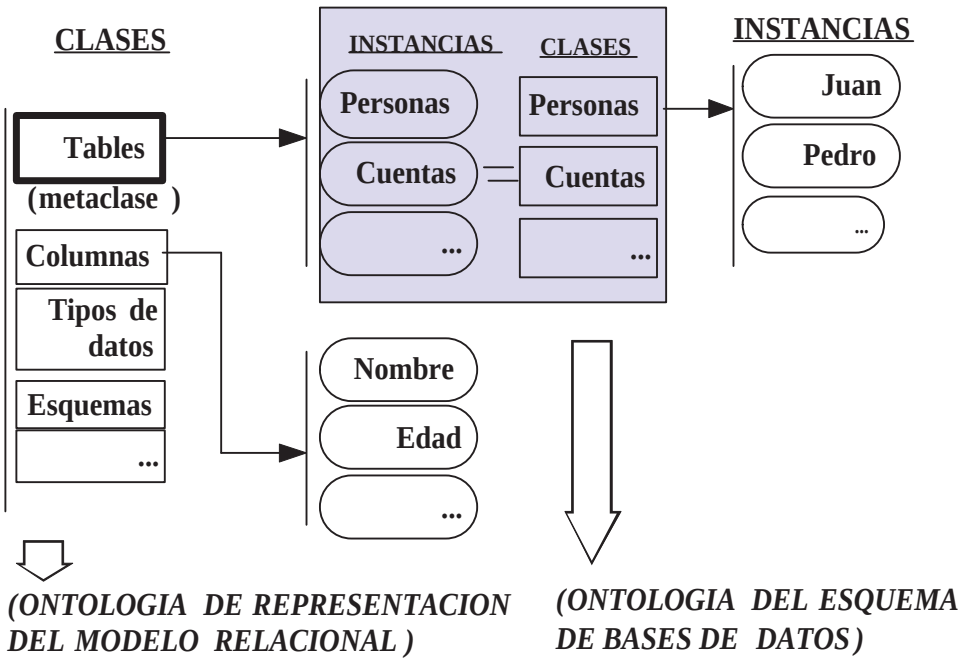


Figura 2.3: Ejemplo de relación entre las Metaclases y las Instancias en la representación mediante ontologías de un esquema de BD

bases de datos como una ontología son las siguientes:

- Simplifica la visión de una base de datos, dado que presentan el modelo alejado de una implementación concreta.
- Aporta otra posibilidad de acceso a la información almacenada en una base de datos, además de la que da el propio RDBMS o las aplicaciones que la utilicen.
- Hace visible la estructura de la información de una base de datos, a través de una representación estándar como puede ser OWL o RDF. Esto puede resultar útil en algunos entornos como la Web Semántica, donde el acceso al contenido semántico de las bases de datos es escaso.
- Permite incluir en la Web Semántica la información de los esquemas, anotar páginas web dinámicas o sistemas de acceso a BDs.
- Es fácil mantener actualizados los cambios que se producen en la estructura de una base de datos, puesto que la generación de la ontología es automática.
- Permite la comunicación y compartición de información entre bases de datos heterogéneas, dado que la representación de la información es independiente de cualquier RDBMS.
- Permite el establecimiento de relaciones entre diferentes modelos de representación de datos además de esquemas relacionales, esto es: orientados a objetos, ontologías, estructuras XML, esquemas RDFS, etc.
- Permite la gestión homogénea de bases de datos distribuidas.
- Enriquece la representación de la información, permitiendo a la ontología generada de Bases de datos relacionarse con otro tipo de ontologías de dominio, más ricas en semántica (a través de técnicas de mapeo o alineamiento), para mejorar así la calidad de la información representada.
- Permiten representar tipos de datos complejos o diferentes tipos de datos de manera sencilla para el usuario, cuya representación y gestión a través de un RDBMS sería mucho mas costoso para su interpretación y manipulación por parte de usuario (dado la estructura de representación de datos que tienen estos sistemas).

2.3.3. Integración de Información

En la actualidad nos encontramos con un gran número de ontologías que describen cualquier concepto de la realidad. Los contenidos de las ontologías varían desde la descripción de un simple dominio, hasta la descripción de tareas o metadatos. A su vez, existen ontologías que representan la misma realidad o parte de ella. Es más incluso existen otros mecanismos de representación de información alternativos a las ontologías también accesibles y utilizados por los usuarios. Ante toda esta cantidad de información se impone la necesidad de acceder a la misma, de forma unificada y transparente, para lo que requiere la utilización de mecanismos de integración de estructuras de información.

Se han desarrollado un gran número de sistemas para permitir integrar una amplia variedad de datos provenientes de muy diversas fuentes. La integración de Ontologías, descrita en [Cho06, Ham04, Noy04], es una de las operaciones más estudiadas, desarrolladas e implementadas dado el gran número de representaciones de este tipo que existen. Sin embargo el campo de las ontologías no es el único que necesita utilizar sistemas de integración de información. Desde que la Web Semántica permite el acceso a fuentes de información de muy diversa índole (dicha información se encuentra representada en diversos formatos, incluso en diferentes lenguajes) la necesidad de sistemas de integración cada vez más sofisticados se hace más acuciante. Algunos ejemplos de los diferentes tipos de esquemas que nos podemos encontrar son: esquemas XML, ontologías (RDF, OWL), esquemas relacionales (SQL), esquemas orientados a objetos (UML), folksonomias, tesauros, etc.

El proceso de integración de información no es simple. George [Geo05] resume los diferentes tipos de heterogeneidad que nos podemos encontrar en los esquemas y las dimensiones de integración, que pueden establecerse en tres:

- Integración del Sistema, representa la heterogeneidad en la plataforma donde se representa la información,
- Integración del Esquema, representa la heterogeneidad entre esquemas. En [Geo05] se identifican cinco tareas en este proceso: a) preintegración, que es cuando se traduce el esquema en forma de modelo de datos, b) comparación, que es cuando se identifican los conflictos semánticos, c) ajuste: hace los conflictos compatibles para combinarlos mediante una representación similar, d) combinación: integra los esquemas, e) reestructuración: refina el esquema

- Integración Semántica, resuelve las diferencias entre la representación de los datos conceptuales mediante la determinación de equivalencias entre los constructores del esquema.

Aunque la mayoría de las aproximaciones para integrar información están basadas en técnicas de integración de esquemas que provienen de las disciplinas de bases de datos, existen ciertas diferencias entre las ontologías y las bases de datos, como hemos visto en la sección anterior y destacan Kalfoglou y Schorlemmer [Kal03]. Las dos aproximaciones más comunes en el proceso de integración de esquemas son: *Vista Local* y *Vista Global* [Gog05]. La aproximación de *Vista Global* consiste en establecer una representación de dominio genérica (un esquema global) donde los esquemas locales se mapeen con el global (esta técnica es la más utilizada en BBDD [Apa05]). La aproximación de *Vista Local* implica establecer correspondencias directas entre los diferentes esquemas locales.

Existen varias propuestas para establecer la correspondencia entre esquemas y ontologías. Por ejemplo, MAPONTO [An04] es una herramienta que utiliza lógica para establecer mapeos entre ontologías y BDs, COMA++ [Aum05] herramienta que resuelve los problemas de correspondencias entre los esquemas y las ontologías escritas en diferentes lenguajes como SQL, W3C XSD u OWL. GLUE [Doa02] u *Ontomap* [Gal05a] son otros ejemplos de herramientas usadas para la búsqueda de correspondencias entre esquemas automáticas.

En este trabajo, se intentará establecer un marco idóneo para integrar esquemas de bases de datos difusas con el resto de estructuras heterogéneas que pueden obtenerse a través de la Web Semántica. Se han identificado dos dimensiones en este marco:

- La integración del sistema, que requerirá la integración de esquemas a partir de SGBDs diferentes como Oracle ©, MySQL ©, PostgreSQL ©, etc. Cada sistema tendrá sus propias características que deberán ser analizadas para que esta integración pueda llevarse a cabo.

- La integración del esquema que permite integrar esquemas heterogéneos. Estos esquemas pueden representarse utilizando diferentes lenguajes como SQL, XML u OWL. Esta tarea requiere que sean resueltos algunos conflictos, como: conflictos de tipo de datos, de escalado de datos, de pérdida de datos, etc. [Hai05]

Dadas las características específicas de la última dimensión, la integración semántica será estudiada una vez que las dos dimensiones previas

sean desarrolladas. Cualquiera de las dos aproximaciones, *Vista Global* o *Vista Local*, pueden ser válidas para establecer correspondencias entre diferentes esquemas. Sin embargo, dada la naturaleza de la información con la que se trabaja, lo más usual es que se utilice mayoritariamente la de Vista Local.

La representación de esquemas de BD Difusas en forma de ontología puede establecer un marco idóneo donde cualquier esquema local pueda establecer la correspondencia con dicho esquema que permite la representación de información difusa. Dicha representación de esquemas será detallada en los capítulos siguientes.

2.4. Ontologías Previas

En este apartado se presentan aquellas dos ontologías en las que está basado este trabajo de investigación. A partir de ellas y tal y como marcan las metodologías de generación de ontologías (realizando una operación de mezcla y de alineamiento de ontologías) se formará una nueva, objeto de esta tesis.

2.4.1. Ontología de Tipos de Datos

Los tipos de datos, también denominados tipos base, tipos primitivos o tipos built-in, han sido descritos por todos los lenguajes de programación, o sistemas de representación de datos, que requieren el almacenamiento o manipulación de los mismos. A partir de ellos, se formarán otros tipos de datos más complejos y con mayor capacidad expresiva.

Sin embargo si requerimos la representación de los tipos de datos base, en teoría, simplemente tendríamos que irnos a cualquier especificación de lenguaje que los utilizara, y obtendríamos un listado con los nombres que tienen asignados y sus características principales. El problema viene dado porque cada representación difiere en los tipos de datos que implementa, incluso algunos nombres son diferentes, como es el caso de lenguajes de programación como C o Pascal, o SGBDs como Oracle © o MySQL ©.

El ANSI SQL hace una distinción de los tipos de datos predefinidos, describiendo cada uno de ellos de manera ligada a la representación de datos que se hace en el Modelo Relacional [fSIIT99, fSIIT03]. Pardede et al. [Par05] toma la ultima revisión del SQL , el ANSI SQL 2003 (SQL4) y hace una clasificación de los tipos de datos predefinidos tal y como se puede ver en la figura 2.4.

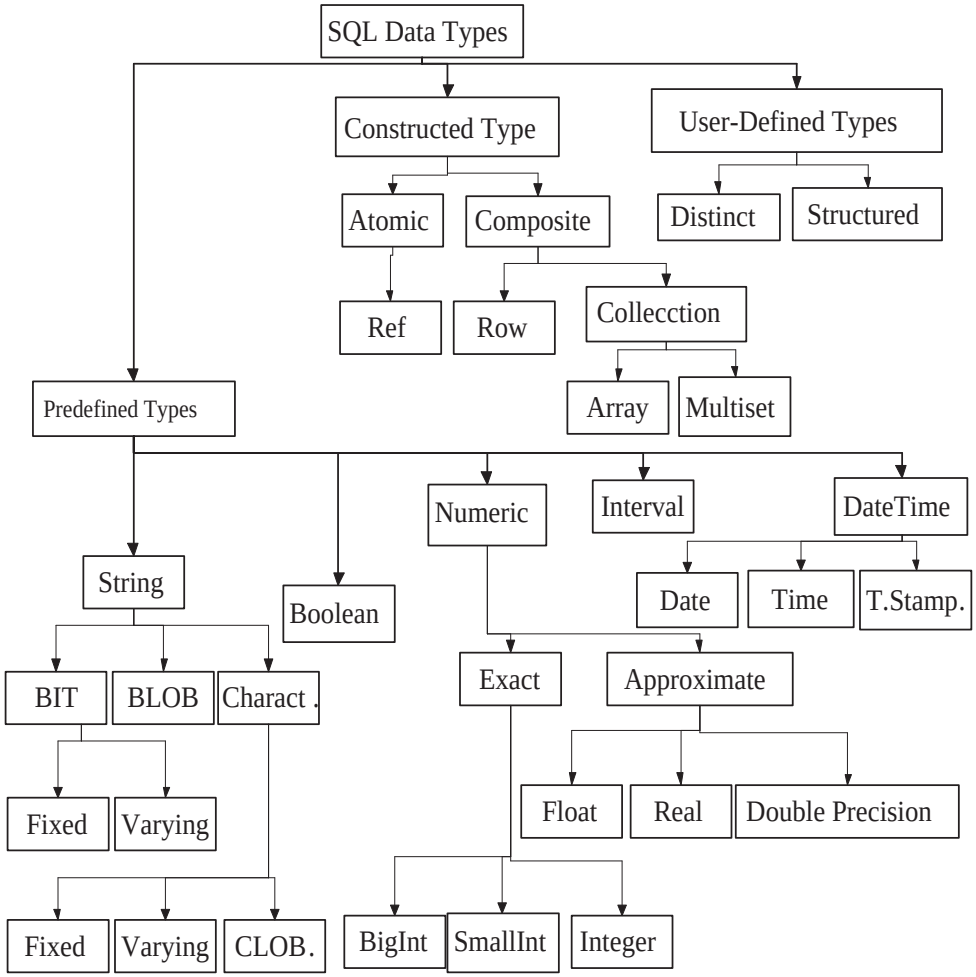


Figura 2.4: Clasificación de los tipos de datos expresados en SQL4 dada por Pardede [Par05]

Esta clasificación será utilizada en este trabajo de investigación como parte de la ontología desarrollada para formar una ontología de representación del conocimiento difuso.

2.4.2. Ontología de Descripción del SQL2003

Siguiendo con el Modelo de Representación de Datos Relacional, y dado que el ANSI 2003 [fSIIT03] propone la última revisión del estándar, Calero et al. en [Cal05, Cal06] propone una ontología, que revisa la especificación de este estándar, utilizando el lenguaje de representación UML y añadiendo las restricciones necesarias que plantea el modelo utilizando el lenguaje de restricciones OCL. El diagrama de clases en UML que se presenta en la figura 2.5 muestra la parte de la ontología de Calero et al. [Cal05, Cal06] utilizada en este trabajo de investigación:

La ontología de Calero et al. [Cal05, Cal06] describe todas las estructuras objeto relacionales que presenta el SQL4, aunque no detalla los tipos base predefinidos y sin embargo, si que lo hace con los tipos de datos complejos y el resto de estructuras que representa el Modelo Objeto-Relacional.

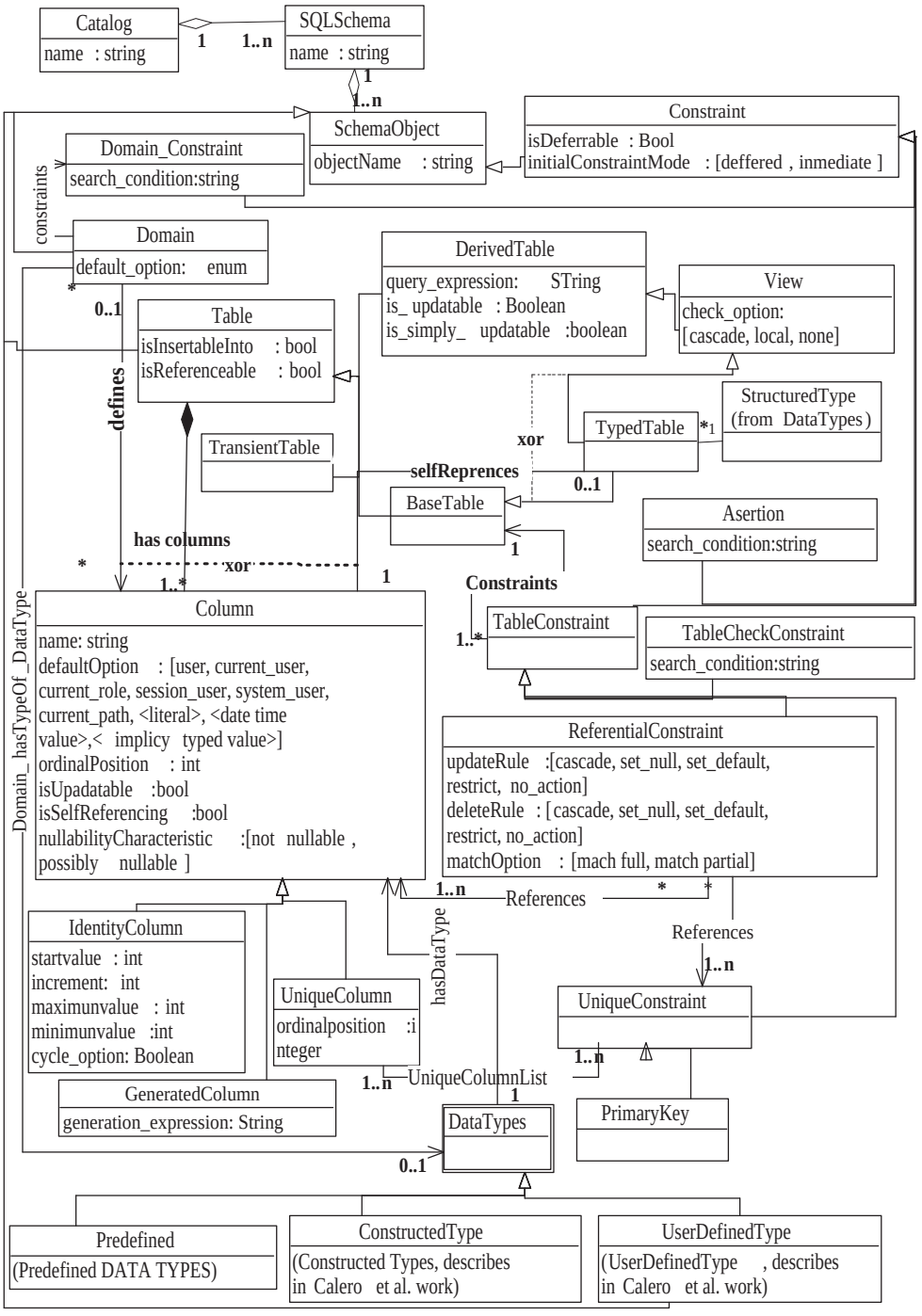


Figura 2.5: Parte de la Ontología en UML dada por Calero et al. [Cal06] del SQL4

Capítulo 3

El problema de la Representación de Datos Heterogéneos en Bases de Datos Difusas. Arquitectura de un SGBDR Multipropósito

3.1. Introducción

En este capítulo se hace un repaso por las diferentes extensiones al modelo de bases de datos relacional que se han propuesto para representar información difusa. Dicho repaso culmina con la descripción en profundidad de la extensión difusa al modelo de base de datos relacional desarrollado por Medina et al. [Med95], denominado *GEFRED*. También se describen a continuación las extensiones realizadas a *GEFRED* que permiten manipular estructuras lógicas para realizar deducciones por un lado, y operaciones de minería de datos sobre un SGBD Difuso por otro. Estas tres extensiones al modelo de bases de datos relacional clásico, forman la arquitectura básica sobre la que se fundamenta este trabajo de investigación.

Cada una de las extensiones descritas presentará en primer lugar el modelo teórico sobre el que está basada la arquitectura propuesta. A continuación se presentan los datos que puede representar el modelo y la estructura necesaria para que dichos datos puedan ser representados (extensión del catálogo del sistema). Y por último, la extensión al lenguaje

de consulta SQL que permitirá gestionar la información almacenada en el SGBD.

Es importante destacar que la arquitectura propuesta, *FIRST* (basada en *GEFRED*, que permite la representación de información imprecisa), sirve como base para la implementación de las otras dos arquitecturas desarrolladas, que incluirán, entre sus funcionalidades particulares, el manejo de datos difusos.

Finalmente, se propone como nueva aportación una última extensión al SGBD que permita la combinación de las tareas de gestión de minería de datos y representación de información imprecisa y lógica (las tres extensiones al SGBD anteriores) a la vez, para aumentar así, el potencial de las consultas. Esta propuesta define la arquitectura de un *Servidor Unificado Multipropósito* que permite resolver consultas complejas sobre una base de datos que soporta todos los tipos de datos descritos anteriormente. Para ello, extiende la Base de Metaconocimiento de tal forma que sea lo suficientemente genérica para permitir la inclusión de las nuevas arquitecturas y, por tanto, aumentar la escalabilidad del sistema. Con el *Servidor Multipropósito* propuesto, tratamos de solucionar el problema de incompatibilidades que existen entre las extensiones anteriormente descritas. Además, se presenta un ejemplo de cómo funcionaría esta propuesta, utilizando una consulta compleja, y qué ventajas e inconvenientes presenta.

3.2. Representación de Información Imprecisa en el Modelo Relacional

Se han propuesto muchas extensiones al modelo relacional de bases de datos desde que Zadeh ([Zad65]) introdujera el concepto de conjunto difuso, que permitió representar datos difusos. Existen en la literatura actual, numerosas recopilaciones donde se resumen las diferentes extensiones difusas realizadas al modelo relacional, entre las que destacamos los libros de Ma [Ma05, Ma06], y los trabajos de Chen [Che99], Petri [Pet96], Medina et al. [Med94b] o Galindo et al. [Gal06].

3.2.1. Antecedentes del Modelo Relacional Difuso

Para la representación y el tratamiento de información imprecisa en el ámbito de las Bases de Datos Relacionales, se han presentado varios modelos a lo largo de estos años. Entre ellos, destacan:

- Aproximaciones que no emplean la lógica difusa, y que se basan en el modelo original de Codd [Cod79, Cod86, Cod87, Cod90].

- Aproximaciones que usan distribuciones de posibilidad para representar la información difusa a nivel de tuplas, como la de Raju and Majumdar [Raj88]. Este modelo también se ha denominado *Modelo Básico de Bases de Datos*.

- Aproximaciones que utilizan las relaciones de similitud para representar la información difusa, son aquellos desarrollados por Buckles y Petri [Buc82b, Buc82a], Shenoi y Melton [She89] y Rundensteiner et al. [Run89].

- Aproximaciones que usan distribuciones de posibilidad para representar la información difusa a nivel de atributo. Algunas de estas son las de Prade and Testemale [Pra84b, Pra84a, Pra87b, Pra87a], Umano y Fukami [Fuk79, Uma80, Uma82b, Uma82a, Uma94] o Zemankova y Kaendel [Zem84, Zem85].

- Aproximaciones mixtas que combinan diferentes técnicas para representar la información imprecisa y conseguir representar el máximo de información posible. Estas aproximaciones se basan en la propuesta de un modelo difuso que combina distribuciones de posibilidad y relaciones de similitud a la vez, como la *Base de Datos Difusa Extendida Basada en Posibilidad* propuesta en Ma et al. [Ma00], Rundensteiner et al. [Run89] y Chen et al. [Che92], o la extensión hecha por Medina et al. en [Med94b, Med94a] denominada *GEFRED*.

El Modelo propuesto por Medina et al. en [Med94b, Med94a] se describe con mayor detalle en los apartados siguientes, dado que ha sido utilizado como base de este trabajo de investigación.

3.3. Extensiones al Modelo Relacional para Representar Información Imprecisa

El modelo GEFRED establece las bases de la representación de datos difusos en el modelo relacional. A partir del mismo, otras extensiones realizadas ya incluirán la gestión de datos difusos como una parte más del sistema. Esta fórmula la utilizan dos extensiones concretas: una que

permite representar información lógico-deductiva y otra que realiza operaciones de minería de datos (búsqueda de reglas de asociación, operaciones de clustering, etc.) ambas utilizando datos difusos.

A continuación se presentan muy brevemente dichas extensiones, pudiéndose consultar con más detalle en el Anexo B o en las fuentes bibliográficas referenciadas.

3.3.1. Modelo Generalizado para Bases de Datos Relacionales Difusas (GEFRED)

El modelo GEFRED de Medina et al. [Med94a, Med94b, Med95] surge como una integración de algunas tendencias (véanse trabajos de Prade y Testemale [Pra84b, Pra84a, Pra87b, Pra87a], Umano y Fukami [Fuk79, Uma80, Uma82b, Uma82a, Uma94], Bucles y Petri [Buc82b, Buc82a, Buc84, Buc89], Zemankova y Kaendel [Zem84, Zem85]) para resolver el problema de la representación y consulta de información imprecisa en el seno del modelo relacional.

Dicho modelo define formalmente una Base de Datos Relacional Difusa (BDD) a través de las definiciones de los siguientes conceptos:

- *Dominio Difuso Generalizado*: se trata de una extensión del concepto de dominio relacional que amplía el rango de valores que un atributo puede tomar. Entre algunos de estos valores se encuentran: el valor nulo, el valor no aplicable, el valor desconocido, un conjunto de asignaciones escalares o numéricas posibles, distribuciones de posibilidad construidas sobre dominios escalares o numéricos, etc.
- *Relación Difusa Generalizada*: define una relación incluyendo el concepto de *Dominio Difuso Generalizado*.
- *Comparadores Difusos Generalizados*: extienden el concepto de comparador para incluir las comparaciones entre valores que existen en el *Dominio Difuso Generalizado*.
- Operaciones de BBDD: proyección y selección difusa.

Las definiciones formales de estos conceptos están detalladas en la sección B.1.1.

Arquitectura FIRST

A partir de esta definición formal se propone una representación concreta de la información imprecisa, la cual se ha denominado FIRST (des-

crita en detalle en la sección B.1.2). Esta representación plantea una estructura de los datos difusos definidos en el *Dominio Difuso Generalizado*, discriminando entre datos imprecisos sobre un referencial:

- *Ordenado*: para ello se establece un mecanismo para representar las distribuciones de posibilidad utilizando aproximaciones a las mismas a través de representaciones trapezoidales (véase figura B.1 del Anexo B) y etiquetas lingüísticas.
- *No ordenado*: son datos sobre los que se definirá una relación de semejanza para representar su dominio subyacente. Las distribuciones de posibilidad en este tipo de dato definen asignando un grado de pertenencia de cada valor al conjunto de valores del atributo. La figura B.4 del Anexo B muestra los valores que puede tomar dicha representación.
- Además se permite representar los valores especiales *Null, Unknown y Undefined*.

Resumiendo, en FIRST se definen explícitamente tres tipos de atributos para representar el *Dominio Generalizado Difuso*:

- *Tipo Difuso 1*: representa datos almacenados de forma precisa que pueden ser consultados de forma imprecisa. Los tipos utilizados son los tipos base propios del SGBDR que se utilice.
- *Tipo Difuso 2*: representa datos imprecisos pertenecientes a un dominio difuso construido sobre un referencial ordenado y que pueden ser consultados de forma imprecisa. Para ello se necesita una representación especial de estos datos, la cual, utiliza estructuras que combinan los tipos de datos base proporcionados por el SGBDR. En la tabla B.1 del Anexo B se muestra la estructura necesaria que han de seguir las representaciones de valores: *Null, Undefined, Unknown*, etiquetas lingüísticas, valores intervalares, aproximados o triangulares, trapezoidales o clásicos.
- *Tipo Difuso 3*: representa datos imprecisos pertenecientes a un dominio difuso construido sobre un referencial discreto no ordenado, sobre el que se define una relación de similitud y que pueden ser consultados de forma imprecisa. Para ello, se representan las estructuras de datos: *Null, Undefined, Unknown*, valores simples, y distribuciones de posibilidad descritas en detalle en la tabla B.2 del Anexo B.

Además de la representación de la información, se llevó a cabo la implementación de una serie de comparadores difusos para gestionar este tipo de información y se añadió el concepto de *Grado de Cumplimiento de una Condición o Umbral*, para completar la operación de selección.

FMB

Para poder llevar a cabo la representación de la información imprecisa, tal y como describe FIRST, en un SGBDR concreto, se propone la creación de la Base de Metaconocimiento Difuso (FMB). La FMB esta formada por las relaciones donde se incluye toda la información acerca de la estructura de los dominios y los valores que puede tomar cada atributo difuso. Estas relaciones, descritas con detalle en la sección B.1.3, se encuentran brevemente descritas a continuación:

- *Fuzzy_Col_List*: contiene los atributos difusos definidos en la BD.
- *Fuzzy_Object_List*: contiene todos los objetos difusos de la BD (por ejemplo, todas las etiquetas definidas en la BDD).
- *Fuzzy_Label_Def*: contiene las distribuciones de posibilidad trapezoidales asociadas a etiquetas lingüísticas.
- *Fuzzy_Approx_Much*: contiene los parámetros usados para la comparación de valores difusos contenidos en columnas de los *Tipos Difusos 1 y 2*.
- *Fuzzy_Nearness_Def*: contiene la relación de semejanza entre cada par de valores de un dominio de TD 3.
- *Fuzzy_Compatible_Col*: contiene aquellos *Tipo Difuso 3* que comparten dominio.
- *Fuzzy_Qualifiers_Def*: contiene el umbral mínimo de satisfacción para cada cualificador definido sobre una etiqueta lingüística.

Cada una de estas relaciones contiene una serie de atributos y restricciones que determinan su funcionamiento. En la figura B.6 se puede observar el comportamiento de las mismas de modo gráfico.

FSQL

El lenguaje FSQL (Fuzzy SQL) aparece junto con la arquitectura FIRST, para extender el lenguaje que permite gestionar la información imprecisa en un SGBD que soporta dicha arquitectura. Este lenguaje incluye las extensiones del DDL y el DML como se describe en el apartado B.1.4. Además en la tabla B.3 se encuentra una referencia a todas las instrucciones extendidas que aporta este lenguaje.

Toda la arquitectura FIRST se implementó en un SGBDR concreto, Oracle © utilizando el lenguaje de programación incrustado PL/SQL, permitiendo así que la definición de los operadores y el intérprete del lenguaje FSQL (Fuzzy SQL) fuera una parte más del sistema de representación de datos.

3.3.2. Representación de Información Lógica sobre BDD

Las Bases de Datos Relacionales Lógico Deductivas permiten extraer información a partir de los datos que se encuentran en una BD cualquiera o representar información lógica. Esta funcionalidad se lleva a cabo a través del uso de relaciones especiales (extensivas e intensivas), reglas lógicas y de motores lógicos (Prolog, Datalog, etc.) que permiten la deducción de información.

El tratamiento de la información difusa en una BD Lógica requiere, en primer lugar la representación de dicha información difusa en un SGBD. Para ello que se utiliza GEFRED como modelo de datos difuso. A continuación se extiende GEFRED para representar algunos de los conceptos fundamentales del modelo lógico-deductivo, apareciendo así, FREDDI [Pon96, Med97]. Dicha extensión, que se encuentra detallada en la sección B.2, se describe brevemente a continuación:

- *Relación Extensiva Difusa*, es una relación Difusa Generalizada desde el punto de vista del modelo GEFRED (definición formal B.7 localizada en la sección B.2).
- *Relación Intensiva Difusa*, que consta de una cabecera que describe una Relación Difusa Generalizada, pero el cuerpo sería un conjunto de reglas orientadas a la deducción con datos difusos, que permiten el cálculo de la instancia de la relación (véase definición B.8).
- *Regla Generalizada con Grado de Acoplamiento*, sería definida para poder generar la instancia de las relaciones intensivas difusas. Su

definición esta descrita en el apartado B.2, y se corresponde con la B.6.

Arquitectura FREDDI Extendida

Al igual que pasaba en GEFRED con FIRST, FREDDI [Pon96, Med97] se propone como arquitectura donde se unifica el sistema de consulta deductivo con el sistema de consulta difuso ambos construidos sobre un SGBDR.

FREDDI propone las siguientes estructuras (descritas con más detalle en B.2.2) para representar la información lógico-deductiva en un SGBDR:

- *Relación Intensiva*: es una relación normal pero su instancia se calcula en función de los predicados que intervienen en el cuerpo de reglas cuando se consulta, o bien es una relación temporal construida en el momento de resolver la consulta.
- *Reglas Lógicas*: se representan en función de sus predicados y variables, almacenándose en orden y con el grado especificado.
- *Motor de Inferencia*: será un módulo, bien interno al SGBDR (si este lo permite) o bien externo, implementado en un lenguaje de programación lógico.

Las *Relaciones Extensivas* carecen de representación especial dado que se corresponderían con las *Relaciones Difusas Generalizadas* anteriormente descritas en FIRST.

Base de Metaconocimiento Deductivo. Base de Reglas (RB)

La representación de la información deductiva en una Base de Datos Difusa necesitará estar descrita por dos bases de metaconocimiento:

- *FMB*, anteriormente descrita, representa la información difusa.
- *RB o Base de Reglas*, proporciona la representación de las relaciones intensivas y las *Reglas Generalizadas con Grado de Acoplamiento Difuso*.

La *Base de Reglas* está compuesta por un conjunto de relaciones que se describen con detalle en la sección B.2.3. Sus funciones, atributos, y restricciones se ilustran en la figura B.7 del Anexo B y se listan a continuación de forma muy resumida:

- *Intensional_Table_Description*: almacena los predicados intensivos.
- *Rule_Description*: describe cada una de las reglas como una secuencia de predicados extensivos e intensivos y comparaciones concatenados con el operador de conjunción.
- *Predicate_Description*: describe el orden de las variables en cada uno de los predicados.
- *Comparision_Description*: describe las condiciones, tipo especial de predicados, que sólo poseen dos variables y su tipo es uno de los siguientes: $=$, $\neq$, $\leq$, $<$, $\geq$, $>$, *FEQ*, *FGT*, *FGEQ*, *FLT*, *FLEQ*, *MGT*, *MLT*, *NFEQ*, *NFGT*, *NFGEQ*, *NFLT*, *NFLEQ*, *NMGT* y *NMLT*.

La arquitectura FREDDI Extendida es la que permite flexibilizar la representación de las reglas difusas y aumentar el número de comparadores difusos tal y como se ve en la figura B.7del Anexo B.

DFSQL

Al igual que ocurre con FSQL, el DFSQL (Deductive FSQL) es el lenguaje de consulta extendido que añade a los predicados descritos en FSQL aquellos que permiten realizar operaciones deductivas. Se añaden así sentencias de definición de datos, como reglas lógicas o relaciones intensivas, y se modifican sentencias de manipulación de datos como la SELECT para realizar consultas deductivas. En la sección B.2.4 del Anexo B se puede encontrar un resumen de las mismas y referencias a una información mas detallada de este lenguaje.

3.3.3. Ampliación de GEFRED para la Minería de Datos

Antes de realizar tareas de minería de datos, se requiere resolver el problema de gestionar información, cualquiera que sea su forma. Carrasco et al. [Car03a, Car03b] propone la implementación de un modelo de BDRD sobre un SGBDR en el que el tratamiento difuso de la diversidad de dominios susceptibles de ser tratados por un sistema de minería de datos sea resuelto. Para ello se extiende GEFRED, y a continuación la arquitectura o interfaz (FIRST) que permite su representación en el SGBDR. Una vez representada la información, las operaciones de minería de datos se describen a través de una nueva extensión a la arquitectura que se ha denominado DmFIRST y que será descrita más adelante.

GEFRED*

Para la gestión de la información difusa, se utilizará la propuesta de GEFRED, sin embargo, dadas las características del modelo de Minería de Datos se necesita su redefinición para permitir que el concepto de dominio difuso tenga un sentido más universal es decir, no restringido a un dominio concreto, y permitir representar tipos de datos complejos (formados por más de un atributo clásico). Esta redefinición, que se ha denominado GEFRED* (véase sección B.3 del Anexo B para mayor detalle), viene dada ante la necesidad de realizar tareas de minería de datos sobre una BDD que requiere operar con tipos de datos más complejos que los presentados hasta el momento.

Se redefinen entonces los conceptos del modelo teórico GEFRED (sección B.3) para gestionar un nuevo concepto de dominio: el *Dominio Difuso Generalizado Complejo* (Definición B.11, sección B.3). En este dominio se describe cómo cualquier atributo definido sobre el mismo podrá tomar cualquier valor simple, excluyente o distribución de posibilidad.

También se encontraran nuevas definiciones para:

- *Relación Difusa Generalizada Compleja*, Definición B.12.
- *Comparador Difuso Generalizo Complejo*, Definición B.14.
- *Proyección Difusa Generalizada Compleja*, Definición B.15.
- *Selección Difusa Generalizada Compleja*, Definición B.16.

Todas estas, se diferencian de las anteriores descritas en GEFRED en el nuevo dominio sobre el que sus datos son definidos.

FIRST*

Carrasco et al. [Car03a, Car03b] también proponen FIRST* como una interfaz que proporciona el acceso a múltiples tipos de datos, definidos en el modelo GEFRED*, con el objeto de realizar tareas de minería de datos sobre un SGBDR. Este interfaz se encuentra descrito en el apartado B.3.2 y extiende el modelo FIRST anteriormente descrito.

Entre las extensiones que realiza destaca:

- La inclusión del *Tipo Difuso 4* representa a la serie de atributos clásicos que determinan un *Dominio Difuso Generalizado Complejo* y por tanto pueden ser consultados de forma imprecisa. Es un

supertipo, estaría formado por los atributos de datos y si es necesario, por los atributos de metadatos (que describen el significado los datos representados en los atributos de datos).

- Esta arquitectura implementa los *Comparadores Difusos Generalizados Complejos*, los cuales confieren al usuario la posibilidad de definir sus propios comparadores difusos en función del *Tipo Difuso 4* definido.

No obstante, esta propuesta no excluye el resto de estructuras descritas en FIRST, por lo que seguirán existiendo los *Tipos Difusos 1, 2 y 3* y el resto de estructuras anteriormente definidas.

FMB*

Tal y como ocurría con la FMB, la FMB* permite describir la información sobre la estructura de los dominios y los valores que puede tomar cualquier elemento descrito en GEFRED*. Dado que GEFRED* extiende GEFRED, en la FMB* se incluyen todas las estructuras que ya formaban parte de la base de Metaconocimiento FMB, añadiendo o modificando aquellas que posibilitan la definición y tratamiento del *Tipo Difuso 4* (véase con más detalle en sección B.3.3 y en figura B.8). Concretamente:

- *Fuzzy_Col_List*: se modifica para contemplar el *Tipo Difuso 4*.
- *Fuzzy_Object_List*: se modifica para almacenar los objetos relacionados con el *Tipo Difuso 4*.
- *DmFSQL_Col_Col*: lista de aquellos atributos de la tabla de la base de datos que forman parte de un dominio difuso generalizado complejo.
- *DmFSQL_Label_Definition*: contiene información sobre las etiquetas lingüísticas definidas para los tipos difusos 4.
- *DMFSQL_Functions*: define la referencia de las funciones tanto que implementan a los distintos comparadores difusos de los atributos difusos de tipo 4, como las funciones de representación de los mismos.
- *DmFSQL_Functions_Col*: contiene la definición para cada atributo *difuso tipo 4*.

- *DmFSQL_Col_Par*: contiene la información de los parámetros adicionales para construir las llamadas a funciones que implica cada tipo difuso 4 respecto a cada comparador.

Minería de Datos en FIRST*: DMFIRST

En Carrasco et al. [Car03a, Car03b] también se propone la implementación de una interfaz que permita utilizar FIRST* como base a la aplicación de distintas técnicas de Minería de Datos en el marco del modelo de BDRD ya implementado. Esta interfaz se denomina DMFIRST y permite realizar las operaciones de clustering, caracterización, clasificación difusa y búsqueda de dependencias difusas entre atributos (para más detalle véase sección B.3.4).

DMFMB

Dado que las operaciones de minería de datos sobre una BDR son complejas se propone definir un nuevo objeto denominado *proyecto* en el cual se proporcionen los parámetros necesarios para realizar una operación de estas características (desde condiciones iniciales, resultados intermedios, y finales). Este nuevo elemento estará descrito en la Base de Metaconocimiento Difuso para la Minería de Datos, denominada DMFMB (descrita en detalle en la sección B.3.5) y engloba las siguientes relaciones (véase figura B.9):

- *DmFSQL_Project*: contiene la información general sobre los proyectos de Minería de Datos.

- *DmFSQL_Col_List*: contiene la información sobre las distintas columnas requeridas en el proceso de Minería de datos.

DMFSQL

Para poder realizar tareas de minería de datos sobre este sistema, se ha extendido el FSQL con un conjunto de sentencias de definición de datos, para crear proyectos de MD (Minería de Datos), y modificado sentencias de manipulación de datos para lanzar consultas de MD. Esta extensión del lenguaje se ha denominado DMFSQL (Data Mining FSQL) y se encuentra descrita en la sección B.3.6.

3.4. Unificación de las Arquitecturas

3.4.1. Visión General del Problema de Unificación

Como se expuso en el apartado anterior, se encuentran desarrolladas tres arquitecturas de bases de datos que permiten gestionar datos y realizar operaciones de muy diversa índole. Resumiendo estas arquitecturas son:

- FIRST, que implementa un SBD que permite almacenar imprecisión en la información,
- FREDDI*, extiende el SGBD para almacenar datos para realizar deducciones a partir de la definición de reglas lógicas, que también pueden ser difusas, y
- FIRST* y DMFIRST que mediante un nuevo tipo de datos, permite la realización de ciertas operaciones de DM dentro de un SGBD.

Nótese que a partir del desarrollo de la arquitectura FIRST, fueron desarrolladas las otras dos, aunque siempre desde el punto de vista de cubrir las necesidades que se requerían para la puesta en funcionamiento de cada sistema en particular. De esta forma se dio lugar a soluciones "ad hoc" que nada tenían que ver entre si, excepto por el hecho de que todas trabajaban con información imprecisa, y que podrían reutilizar y compartir la funcionalidad que proporcionaba la arquitectura FIRST.

No obstante, una vez en funcionamiento todas estas arquitecturas independientes sobre un mismo SGBD, se nos plantean las siguientes preguntas:

- ¿son compatibles los datos que utilizan las diferentes arquitecturas dado que se apoyan sobre la misma implementación que permite la gestión de información imprecisa?,
- ¿se pueden utilizar reglas lógicas para almacenar cualquiera de los procesos de minería de datos gestionados en DMFSQL?,
- ¿Una relación intensiva podría ser consultada por un proceso de minería de datos?.

Como respuesta a estas preguntas, se retoma la reflexión de que cada arquitectura fue desarrollada sin tener en cuenta nada más que aquello que fuera necesario para resolver los objetivos específicos del problema, con lo que se deduce que no existe ningún mecanismo para la combinación de las mismas. Esto es, las operaciones y estructuras definidas por DFSQL son incompatibles con las definidas en DMFSQL.

En este apartado se plantea la infraestructura de un servidor unificado que integra las características de las arquitecturas anteriormente definidas y que permite combinar sus funcionalidades. Para ello, se estudia la viabilidad de la puesta en marcha de dicha unificación, planteando las ventajas e inconvenientes en el desarrollo del sistema [Bla04, Bla05a].

3.4.2. Sistema Actual

En la Figura 3.1 se muestra la arquitectura del sistema actual donde coexisten en un mismo SGBD las tres arquitecturas anteriormente expuestas. Como se puede observar, hay una interfaz de usuario por cada uno de los clientes que puede relacionarse con el sistema:

- El Cliente SQL es el cliente por defecto del SGBD. Accede directamente al Ejecutor de Consultas del SGBD para obtener la respuesta.

- El Cliente FSQL es aquel que permite realizar consultas flexibles al sistema, usando datos clásicos o difusos. Accede a la arquitectura FSQL.

- El Cliente DFSQL, permite consultar al sistema utilizando estructuras lógicas. Accede a los motores deductivos implementados para hacer inferencias y permite combinar aspectos lógicos con estructuras difusas y clásicas.

- El Cliente DmFSQL permite realizar operaciones de minería de datos, difusas o no, definiendo nuevos tipos de datos y extendiendo el FSQL anterior.

Cada interfaz esta conectada con su correspondiente arquitectura. Las arquitecturas comparten el mismo analizador léxico y sintáctico, pero no semántico. El motivo es que la extensión del analizador léxico es muy simple, puesto que consiste en añadir a la lista de tokens permitidos los

necesarios para reconocer los nuevos comandos. El analizador semántico por el contrario, dependerá del significado de cada expresión y por tanto su análisis será realizado de forma particular en cada una de las arquitecturas. Cada módulo se encargará de traducir la consulta en una o varias sentencias en SQL. El acceso a la Base de Datos es común y se realiza a través del Ejecutor de Consultas.

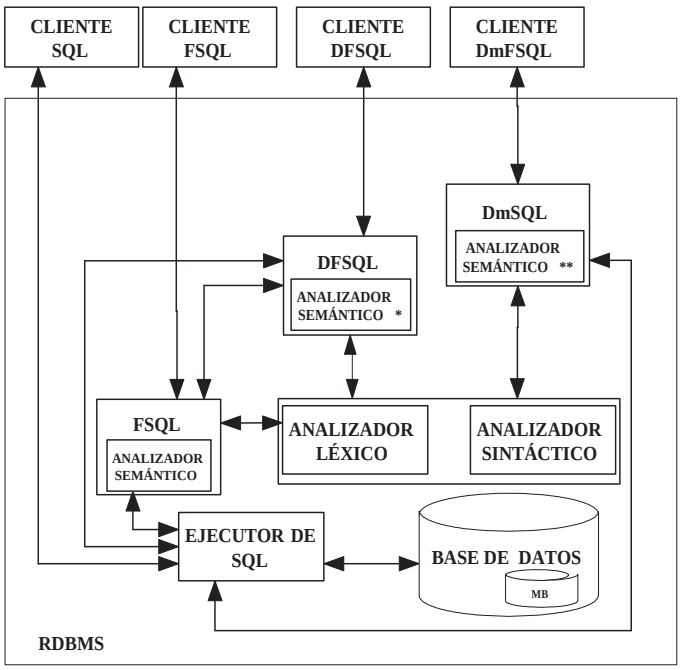


Figura 3.1: Arquitectura de los Servidores Independientes

3.4.3. Arquitectura de un Servidor Multipropósito Unificado

El problema surge cuando se pretenden combinar las distintas tareas que hace cada arquitectura por separado. Así pues, por ejemplo, si en algún momento se quisiera tener almacenada en la base de reglas una que nos mostrase el resultado de haber calculado una dependencia funcional difusa sobre una relación, habría que introducirla a mano, y aún así, su existencia en el sistema nos resultaría extremadamente inútil, dado que no existen operaciones para explotar esta funcionalidad.

Es necesario plantearse cómo extender los diferentes servidores ya

implementados para permitir la combinación de operaciones entre sí y una gestión de datos conjunta. Además, el sistema deberá mantenerse lo suficientemente estructurado para que la incorporación de nuevas operaciones (como puedan ser nuevas tareas de minería de datos, gestión de nuevos tipos de datos como por ejemplo el tiempo, etc.) sea sencilla o cuando menos, posible. Es decir, se plantea un nuevo trabajo de ingeniería inversa, consistente en redefinir y aunar cada una de las arquitecturas planteadas, dejando las definiciones de datos, y operaciones abiertas a nuevas incorporaciones, generando así un único sistema escalable y completo.

De esta forma, y utilizando las arquitecturas anteriormente descritas, se propone la infraestructura de un servidor unificado que integra las funcionalidades de cada una de las arquitecturas permitiendo combinarlas entre sí (véase [Bla04, Bla05a]). Esta integración es capaz de procesar diferentes tipos de consultas en una misma sentencia. Por ejemplo consultas que permitan deducir con datos difusos y utilizando resultados de un proceso de minería de datos.

A continuación se describen los cambios que permitirán la unificación del sistema:

- Combinación y extensión de las diferentes Bases de Metaconocimiento.
- Arquitectura unificada que especifica la secuencia de procesamiento de una consulta. Se genera un servidor creado específicamente para decidir los módulos implicados en dicha ejecución y el orden de participación de los mismos.

3.4.3.1. Base de Metaconocimiento (MB)

Se ha denominado Base de Metaconocimiento (MB) al conjunto de relaciones del catálogo que almacenan la definición de los objetos, tipos de datos, etiquetas, dominios, etc. utilizados por las diferentes arquitecturas y por el servidor unificado. Está formada por los siguientes subcatálogos:

- FMB: representa tipos de datos difusos, dominios difusos, etiquetas difusas, etc.
- RB: almacena predicados intensivos y su definición descrita mediante reglas lógicas.

- FMB*: define un nuevo tipo de datos capaz de representar texto, XML, objetos, relaciones, etc. y las operaciones que pueden ser aplicadas a este nuevo tipo de datos.

- DMFMB: almacena información acerca de las operaciones de clustering, clasificación, y búsqueda de dependencias funcionales sobre datos clásicos o difusos.

Sobre esta nueva estructura relacional se hace necesaria una extensión, dado que las relaciones del catálogo de cada arquitectura son invisibles entre sí, particularidad que elimina cualquier posibilidad de realizar operaciones conjuntas.

En las figuras 3.2 y 3.3 y en [Bla04, Bla05a] se muestra como se relacionan las diferentes estructuras del catálogo a partir de la definición de dos nuevas relaciones. Las dos nuevas relaciones permitirán la compartición de información entre las arquitecturas, y su descripción se detalla a continuación:

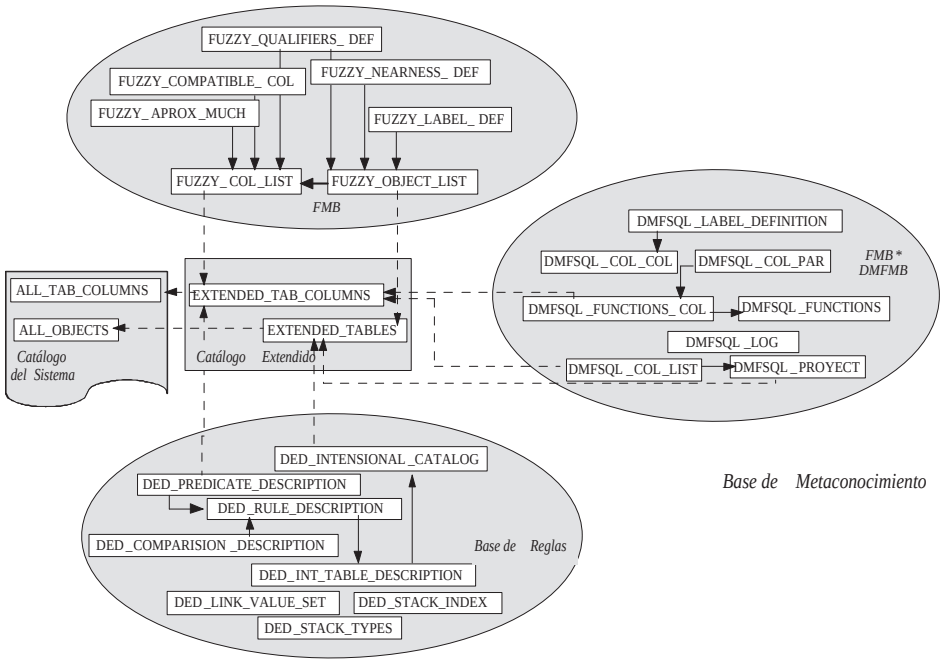


Figura 3.2: Base de Metaconocimiento (MB)

- *Extended_Tables*: almacena las relaciones (clásicas o extendidas) definidas en la base de datos que pueden ser usadas en consultas difusas, deductivas o de minería de datos. Aquellas relaciones almacenadas en *All_Objects*[1] (solo aquellas que hagan referencia a tablas) están incluidas en esta relación (conexión (5) de la figura 3.3). De esta forma, esta relación es una especialización de *All_Objects* dado que todas las relaciones incluidas en ella tienen alguna característica especial de las mencionadas previamente. La tabla 3.1 muestra los atributos de esta relación y los valores que puede tomar. *OBJ#* representa el identificador de la tabla, *TYPE* indica si la tabla es intensiva (no contiene datos) o extensiva y *Orig* da información acerca del tipo de datos que dicha tabla contiene o a partir de donde se ha formado.

Tabla 3.1: Relación *Extended_Tables*

OBJ#	**Type**	**Orig**
	0 (Extensiva) 1 (Intensiva)	0 (Datos Clásicos) 1 (Datos Difusos) 2 (Descripción de la Regla) 3 (Datos de DM) 4 (Descripciones de DM)

- *Extended_Tab_Columns*: proporciona información acerca de todos los atributos (tanto clásicos como extendidos) a los que puede acceder el usuario. Esto incluye algunos atributos almacenados en *All_Tab_Columns*[2] (conexión (1) de la figura 3.3) y una descripción de éstas. Como en la relación anterior, *Extended_Tab_Columns* es una especialización de *All_Tab_Columns* puesto que las tuplas que esta relación puede referenciar pueden ser atributos difusos descritos en FIRST, atributos intensivos descritos en FREDDI, atributos usados en procesos de minería de datos, o atributos que pueden almacenar información temporal o resultados de procesos de minería de

[1]Tabla que hace referencia a todas las tablas del sistema. Esta notación corresponde únicamente a la tabla del catálogo del SGBD de Oracle © para acceder a todos los objetos del sistema. Otros SGBDs utilizan otros nombres para referenciar esta tabla.

[2]Tabla que hace referencia a todas las columnas del sistema. Esta notación corresponde únicamente a la tabla del catálogo del SGBD de Oracle © para acceder a todas las columnas del sistema. Otros SGBDs utilizan otros nombres para referenciar esta tabla.

datos. El atributo TYPE de esta relación (véase tabla 3.2) almacena el tipo de datos que el atributo referenciado puede contener: una regla, un dato difuso, etc., mientras que OBJ# y COL# identifican de forma única el atributo en el SGBD.

Tabla 3.2: Relación *Extended_Tab_Columns*

OBJ#	COL#	Type
		0 (Columna Difusa) 1 (Columna Lógica) 2 (Columna de DM)

Como ya se ha comentado, estas nuevas relaciones se corresponden con las relaciones especificas del catálogo del sistema SGBD utilizadas para contener información acerca de todas las columnas y tablas definidas en la base de datos. En esta propuesta las vistas específicas del SGBDR: *All_Tab_Columns* y *All_Objects* de Oracle ©, han sido usadas a modo de ejemplo para referenciar a los contenidos de tablas y atributos de los SGBDs.

Las conexiones establecidas entre las diferentes arquitecturas y estas dos nuevas relaciones (mostradas en la figura 3.3) son:

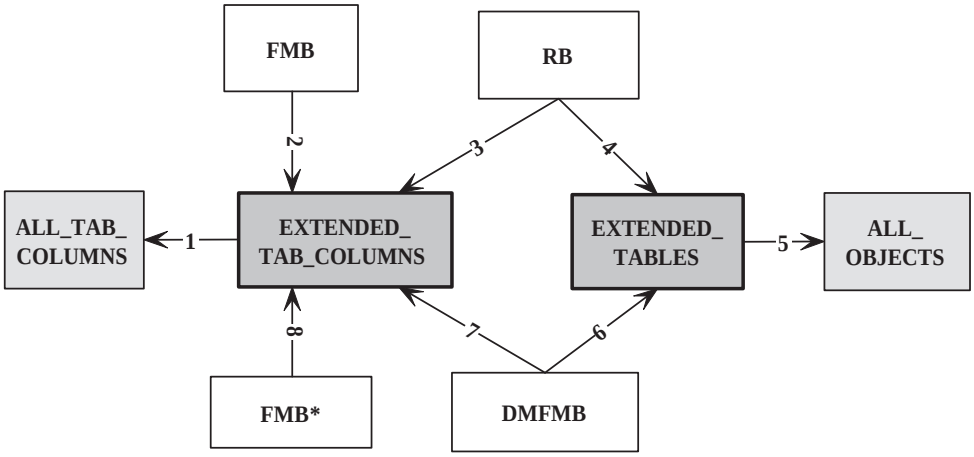


Figura 3.3: Base de Metaconocimiento (MB) con las tablas del catálogo de Oracle ©

- Las conexiones 2 y 8 permiten que se relacionen el FMB y el FMB* con *Extended_Tab_Columns* ya que éstas amplían las definiciones de atributos y sus dominios.
- La conexión 4 permite que la RB se relacione con *Extended_Tables* ya que FREDDI incorpora nuevas relaciones al sistema.
- La conexión 3 relaciona RB con *Extended_Tab_Columns* porque esta extensión debe disponer de atributos a partir de otras relaciones.
- La conexión 7 permite que DMFMB se relacione con *Extended_Tab-_Columns* porque las operaciones de minería de datos se aplican sobre cualquier tipo de atributos.
- La conexión 6 permite que DMFMB se relacione con *Extended_Tables* porque los resultados de sus operaciones tienen que ser almacenadas como nuevas relaciones en la base de datos.

La inclusión de estas tablas hará el sistema escalable en la forma en que permiten una sencilla extensión de la Base de Metaconocimiento (MB).

3.4.3.2. Arquitectura del Servidor Multipropósito Unificado

En la figura 3.4 se muestra una propuesta de arquitectura unificada que permite que todo el flujo de información pase a través de un único cliente. El cliente se encarga de recoger todas las consultas por parte del usuario y enviarlas a un servidor unificado de consultas que será capaz de identificar el tipo de relaciones implicadas en cada una. El servidor se encarga de analizar la consulta enviándola al módulo correspondiente para obtener la solución. Una vez que la consulta ha sido analizada, el servidor controlará la ejecución de todos los módulos que permitan traducir las partes de las que esté compuesta la consulta, e integrará sus respuestas.

Además habrá otro modulo dentro del servidor, el *Planificador de Estrategias de Consulta*, que planificará el orden en el que las consultas deberán ser ejecutadas de forma que aumente la eficiencia del servidor. La estrategia seguida por este planificador consiste en analizar la consulta compleja (consulta que implica diferentes módulos para su resolución) y determinar el orden de ejecución de cada una de las subconsultas incluidas en la sentencia compleja.

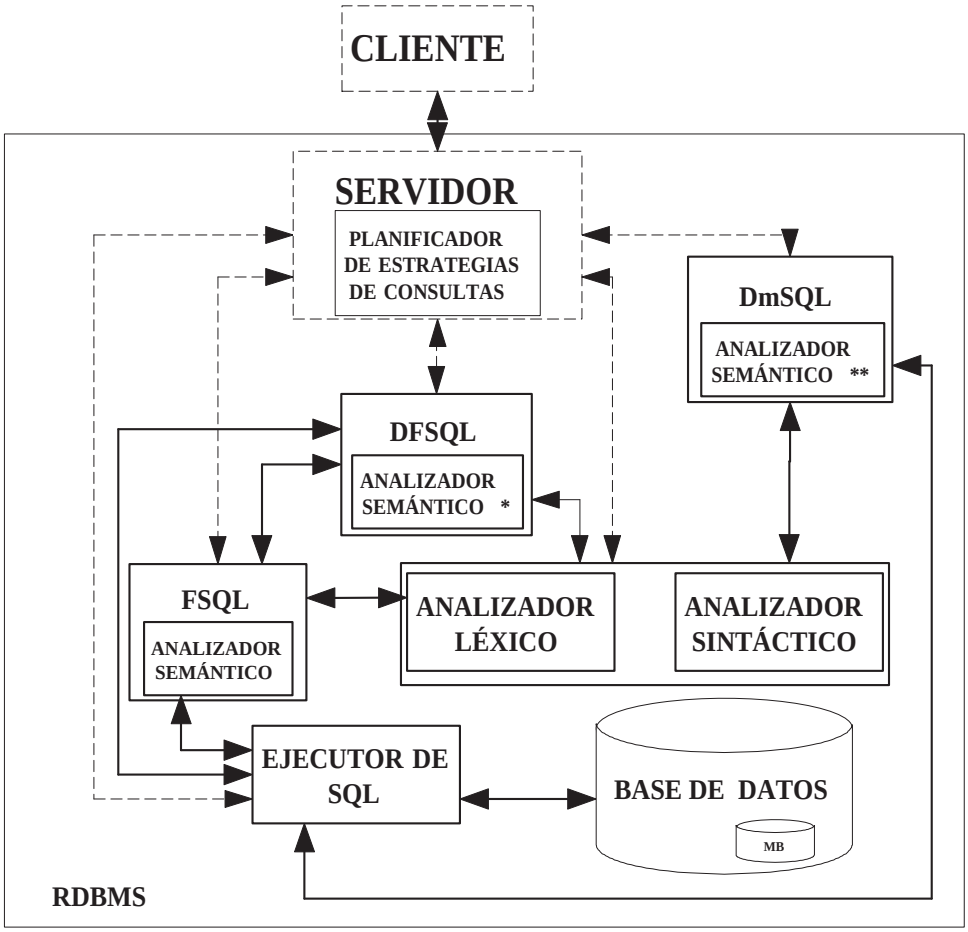


Figura 3.4: Servidor Multipropósito

Las modificaciones propuestas por esta arquitectura están señaladas en la figura 3.4 con líneas discontinuas. El proceso de resolución de una consulta puede resumirse de la siguiente manera:

1. El cliente envía la consulta al servidor.

2. La consulta se analiza por el servidor utilizando los analizadores léxicos y sintácticos para determinar los módulos implicados en su resolución.

3. La sentencia se divide, si es necesario, y enviada al módulo corres-

pondiente. El *Planificador de Estrategias de Consulta* planifica la ejecución de las diferentes subconsultas).

4. Cada modulo analiza semánticamente su parte asignada de la consulta y la traduce a una sentencia en SQL.

5. La parte procesada de la consulta original se devuelve al servidor que integra todas las traducciones proporcionas por cada uno de los módulos implicados construyendo así una única consulta en SQL que será enviada al Ejecutor de consultas SQL, y

6. El servidor formatea el conjunto de tuplas resultantes proporcionadas por el Ejecutor de consultas SQL antes de enviarlas al Cliente.

Como se muestra en la figura 3.4, tanto el servidor como todos los módulos integrados en la arquitectura necesitan consultar la MB.

3.4.4. Ejemplo de Resolución de una Consulta Compleja

La integración de las arquitecturas previamente descritas permite la combinación de diferentes tipos de consultas y el almacenamiento de los resultados de las mismas en forma de relaciones, reglas lógicas, datos calculados, etc. que cualquier otro proceso podría usar con posterioridad.

Este apartado muestra cómo puede relacionarse una operación de minería de datos con la gestión de reglas lógicas difusas. En concreto, este ejemplo muestra cómo una dependencia funcional difusa encontrada mediante un proceso de minería de datos, puede generar una *Regla Generalizada Difusa con Grado de Acoplamiento* y almacenarla en la base de datos.

Dado que se dispone de una *Base de Datos Difusos de Suelos* descrita en el Anexo C, utilizada a lo largo de este trabajo de tesis para ejemplificar todas las aportaciones realizadas en el mismo, incluyendo esta primera de unificación de servidores, se plantea el hecho de buscar, si existe, alguna relación entre los datos que componen esta BDD. En principio se va a tratar de buscar la existencia de dos dependencias funcionales difusas:

La primera dependencia funcional tratará de describir si hay algún tipo de relación entre la *Precipitación Media* que tiene el emplazamiento del terreno particular y la temperatura media que registra dicho emplazamiento. Ambas características que se encuentran descritas en la tabla C.2 Anexo C son difusas, la *Precipitación Media* y la *Temperatura Media*

son atributos de carácter difuso pero basado en un referencial numérico ordenado (*Tipos Difusos 2*). Los valores de su dominio son etiquetas lingüísticas descritas en el Anexo C, tablas C.8 y C.9.

La segunda dependencia trata de descubrir si entre la vegetación que caracteriza un suelo y el tipo de estructura que tenga dicho suelo existe una relación. Esta búsqueda versará sobre datos localizados en la tabla C.3. Los atributos *Vegetación* y *Tipo de Estructura*, a partir de ahora *tipo_est*, son campos de *Tipo Difuso 3* descritos a través de las relaciones de similitud establecidas entre sus valores de dominio que podemos encontrar en las tablas C.27, y C.38 y C.39 respectivamente.

Para la búsqueda de las Dependencias Funcionales Difusas (DFD) planteadas se ve necesaria la utilización de técnicas de minería de datos que permitan analizar las relaciones *Localización* y *Estructura*(tablas C.2 y C.3), concretamente los atributos *Tmedia* y *Pmedia*, por un lado y *Vegetación* y *Tipo de Estructura* por otro. Una vez conocido si se cumplen dichas hipótesis, esto es, la existencia de las DFD que estamos buscando, éstas podrán ser almacenadas en la base de datos como *reglas lógicas con grado de acoplamiento* de forma que el conocimiento extraído no se pierda sino que se almacene y vaya verificándose con las nuevas inserciones sin necesidad de ser recalculado.

Para buscar la DFD en primer lugar es necesario tener la información almacenada en la base de datos y más específicamente, dado el caso que nos ocupa, conocer cómo esta información esta almacenada en la Base de Metaconocimiento (MB) anteriormente descrita.

La figura 3.5 muestra, de manera resumida, la sucesión de acciones en cuanto a creación de tablas o inserción de tuplas en la MB para que sea posible la ejecución de la consulta compleja que se ha planteado. La relación *Localización* mostrada en la tabla C.2 ha sido almacenada en la base de datos utilizando la estructura especial para los datos difusos (descrita en la tablas B.1 y B.2 del Anexo B). Dicha representación almacenada en la base de metaconocimiento se encuentra descrita en la tabla 3.3.

Además de la relación y las tuplas en la base de datos, debe haber constancia de la estructura de la información difusa que se halla en el sistema. Las fases 2 a 6 de la figura 3.5 muestran todas las relaciones implicadas en el almacenamiento de esta información en la FMB. El atributo *F_TYPE* de la tabla *Fuzzy_Col_List* (tabla 3.8) especifica el tipo de dato difuso del atributo almacenado, concretamente: *PMedia y TMedia* son *Tipos de Datos Difuso 2* mientras que *Vegetación y Tipo_es*

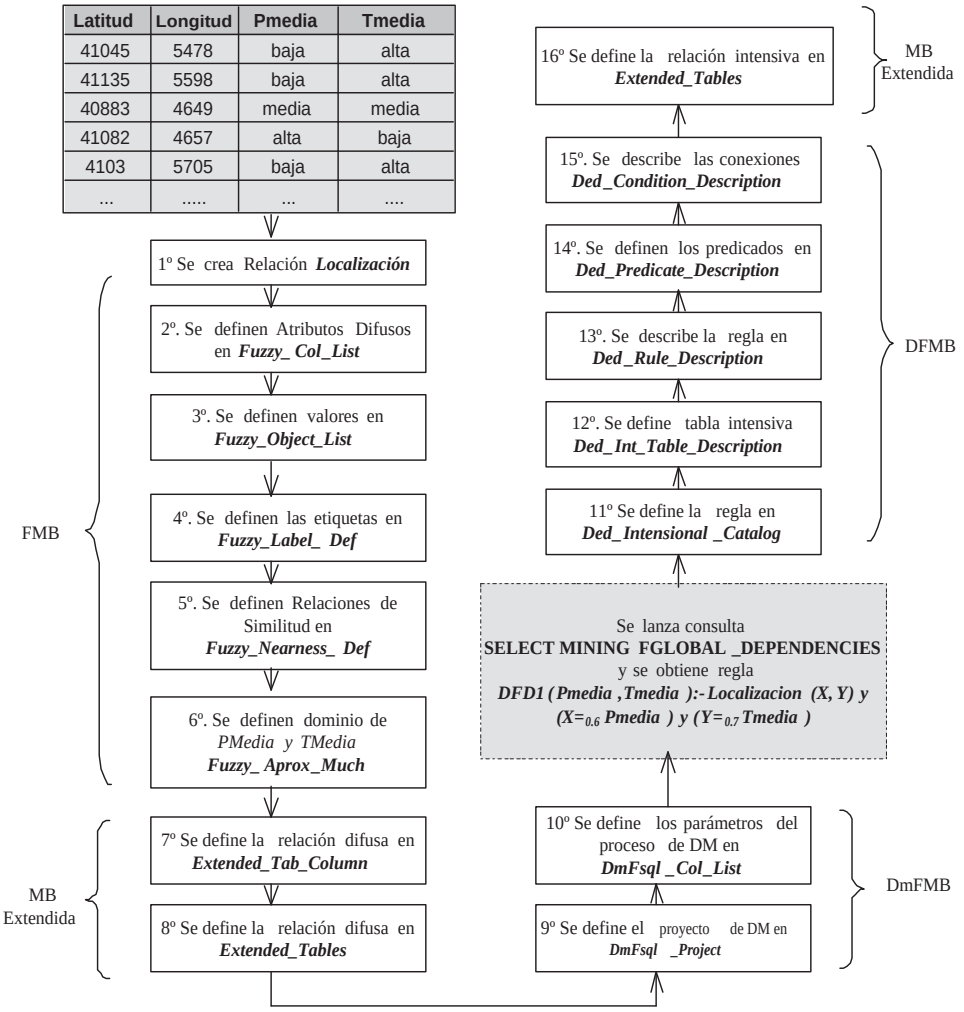


Figura 3.5: Resumen de las acciones ocurridas en la MB en una consulta compleja. Ejemplo DFD1

es un *Tipo de Datos Difuso 3*. La relación de similitud entre los valores de *Vegetación y Tipo_es* están almacenados en la Base de Metaconocimiento en la relación *Fuzzy_Nearness_Def* (descripción en tabla 3.6).

Las etiquetas lingüísticas utilizadas, como la calificación *Alta, o Ba-*

ja están descritas en la relación *Fuzzy_Label_Def* (descripción en tabla 3.7) de la MB. La relación *Fuzzy_Object_List* (descripción en tabla 3.5) almacena las etiquetas utilizadas en el atributo *Vegetación y Tipo_es* y todas las etiquetas que pueden usarse para describir el valor del atributo *PMedia y TMedia*. Además, esta tabla establece un identificador único para cada etiqueta, evitando así cualquier confusión.

Todas las relaciones mostradas anteriormente están referidas únicamente a la parte de representación de información difusa, correspondiente al módulo FMB de la MB. Sin embargo, estos datos deberán ser definidos en las nuevas relaciones de la arquitectura unificada para que puedan ser visibles a todos los sistemas incluidos en el SGBD. De esta forma la relación *Localización y Estructura* y los atributos que la componen serán definidos también en las tablas *Extended_Tables* y *Extended_Tab_Columns* de la MB, correspondientes a los pasos 7 y 8 de la figura 3.5. La relación *Extended_Tab_Columns* (descripción detallada en la tabla 3.10) contiene una referencia a todos los atributos usados difusos en este ejemplo y al tipo de datos que representan (datos difusos). La relación *Extended_Tables* (detalle en tabla 3.11) mantiene una descripción de las relaciones usadas en el ejemplo: hasta ahora únicamente las tablas extensivas recién definidas, *Localización y Estructura*. Los signos '-' de la tabla 3.11 simbolizan que el valor no es relevante en la relación y por tanto no se necesita rellenar este campo.

Una vez definida la estructura sobre la que se va a operar, se puede iniciar el proceso de definición de datos para llevar a cabo una operación de DM. Un nuevo *proyecto* de DM debe definirse sobre la base de datos (especificación más detallada en el apartado B.3.2). Este *proyecto* genera un conjunto de nuevas tuplas en las relaciones correspondientes a la DMFMB de la MB (pasos 9 y 10 de la figura 3.5). La sentencia que permite definir este proyecto tiene la siguiente forma (véase referencia a la sintaxis completa en [Car03a, Car03b]):

```
CREATE_MINING PROJECT Localizacion_PRJ
  ON TABLE Localizacion
  WITH COLUMNS FOR
   FGLOBAL_DEPENDENCIES '('      {
    ANTECEDENT Pmedia FCOMP_GLOBAL_DEPENDENCIES
                              FEQ THOLD_ANT 0.6
    CONSEQUENT Tmedia FCOMP_GLOBAL_DEPENDENCIES
                              FEQ THOLD_CON 0.7}
```

donde *Localizacion_PRJ* es el identificador del proyecto de DM, que mantiene toda la información necesaria para llevar a cabo el proceso de DM, como parámetros, tablas temporales, etc. En concreto el proceso de búsqueda de dependencias funcionales difusas necesita conocer el tipo de dependencias difusas que se han de buscar, el grado de acoplamiento de cada atributo, etc. La definición del proyecto es el primer paso para comenzar el proceso de DM. La estructura para definir el proyecto en la DmFMB está descrita con detalle en las tablas 3.12 y 3.13. La tabla 3.12 almacena una especificación general acerca de la dependencia funcional difusa propuesta y la tabla 3.13 almacena información acerca de cada una de las columnas que forman parte del proceso de búsqueda.

La dependencia funcional difusa buscada se ha denominado DFD1 y se describe con la siguiente expresión:

0.6 - 0.7 DFD1 PMedia $_{FEQ*FEQ} \longrightarrow$ Tmedia with confidence c and support s

El objetivo de esta dependencia funcional consiste, obviamente, en encontrar si la presión en la localización de un suelo influye en la temperatura media, donde el grado de acoplamiento para *Pmedia* es de 0.6 y para *Tmedia* 0.7. La siguiente sentencia DML permite ejecutar en en servidor de minería de datos la búsqueda de la *DFD* planteada:

```
SELECT_MINING FGLOBAL_DEPENDENCIES Localizacion_PRJ
  USING T_NORM THOLD_ANT_CON
```

Esta consulta estará formada, en ultima instancia, por un conjunto de sentencias en FSQL que tendrán una estructura similar a la de la siguiente (que se corresponde a la última sentencia que permite ejecutar esta operación):

```
SELECT COUNT(*) FROM Localizacion A1, Localizacion A2
WHERE(A1.NAME<>A2.NAME) AND
   (A1.Pmedia FEQ A2.Pmedia
   THOLD 0.6)   AND NOT
   (A1.Tmedia NFEQ A2.Tmedia
   THOLD 0.7)
```

El soporte y la confianza de la DDF se han calculado con una sentencia similar a la anterior, contando el número de apariciones del antecedente y consecuente. Si el soporte y la confianza son lo suficientemente altos,

entonces la DFD será aceptada y automáticamente almacenada en la base de datos como una regla lógica. En cambio, si el soporte o la confianza no son lo suficientemente buenos, entonces o bien la dependencia funcional difusa se rechaza o bien se disminuyen los umbrales de cumplimiento.

En el caso en que si se acepte la dependencia, la estructura de la regla lógica sería la siguiente:

DFD1(Pmedia,Tmedia) :- Localizacion(X,Y) $\wedge$
(X $=_{0,6}$ Pmedia)$\wedge$ (Y $=_{0,7}$ Tmedia)

Una vez conocida la estructura de la regla, se procede a su almacenamiento en la base de datos. esto se lleva a cabo a través de su definición en la RB de MB (pasos del 11 al 15 de la figura 3.5). La sentencia en DFSQL que permite generar esta regla es (véase referencia a la sintaxis completa en [Bla01, Bla00b]):

```
CREATE INTENSIONAL TABLE DFD1
   (Pmedia FTYPE2 (2,3) NUMBER (3,2)
   Tmedia FTYPE3 );

CREATE RULE FOR DFD1 (Pmedia, Tmedia) AS
   Localizacion (X SOURCE Pmedia, Y SOURCE Tmedia) AND
   ( X FEQ Pmedia THOLD 0.6) AND (Y FEQ Tmedia THOLD 0.7)
```

donde *Create Intensional Table* inserta una nueva tupla en la tabla 3.14 y crea una nueva relación *DFD1* en la base de datos, por supuesto, sin tuplas ya que se trata de una relación intensiva. Este proceso también incluiría la inserción de una tupla en la tabla 3.11 especificando que el tipo de relación almacenada es intensiva ($Tab\_type = 1$) (paso 16 de la figura 3.5). La sentencia *Create Rule* almacena la estructura de la regla en las relaciones de la RB: 3.15, 3.16, 3.17 y 3.18. Estas cuatro relaciones permiten describir íntegramente la estructura de una *Regla Generalizada Difusa con Grado de Acoplamiento*: los predicados y variables que la conforman, tal y como se puede ver en la sección B.2.

Una vez que la regla se ha definido en la base de datos, cada nueva inserción de una tupla en la tabla *Localizacion* provocará que el sistema compruebe si dicha tupla cumple o no la regla DFD1, es decir, si

sobrepasa el umbral establecido para cada atributo (antecedente y consecuente). Si lo cumple la nueva tupla validará la regla y será insertada en la BD incrementando la confianza y soporte de la regla.

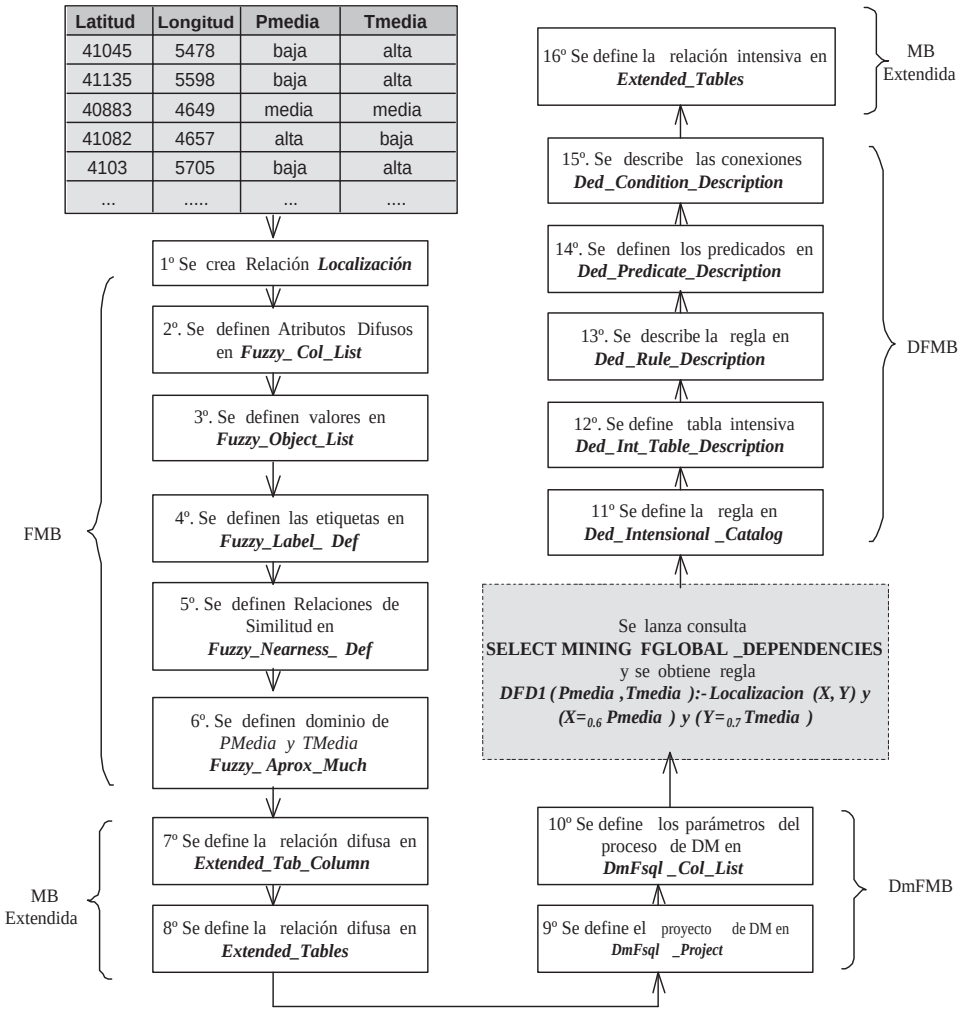


Figura 3.6: Resumen de las acciones ocurridas en la MB en una consulta compleja. Ejemplo DFD2

La misma secuencia de operaciones se ha de seguir para calcular la

segunda dependencia funcional difusa planteada, que podemos ver en la figura 3.6. En las tablas de la MB descritas, se encuentra especificada cada una de las entradas correspondientes a la creación de dicha dependencia, denominada DFD2.

Este ejemplo demuestra que una vez unificada la base de datos, las operaciones de cada extensión pueden ser combinadas, haciéndola así mas operativa. Al igual que ocurre con este ejemplo otro tipo de operaciones pueden realizarse combinando las funcionalidades de los sistemas integrados. En el apartado siguiente se resumen brevemente las operaciones que se pueden realizar sobre esta nueva arquitectura.

3.4.5. Ventajas e Inconvenientes del Servidor Unificado

Una vez diseñada la arquitectura unificada aparecen un buen número de nuevas funcionalidades en el sistema. En general las ventajas del sistema son las siguientes:

- El incremento del número de operaciones y tipos de datos que un RDBMS difuso (FRDBMS) puede gestionar. Estas operaciones incluyen:
  - Realizar deducciones sobre estructuras resultantes de un proceso de minería de datos.
  - Realizar operaciones de minería de datos sobre estructuras lógicas (como relaciones intensivas).
  - Almacenar resultados de operaciones de minería de datos utilizando estructuras lógicas.
- La conversión de esta arquitectura en una más escalable, para que el sistema pueda incrementar el número de operaciones y tipos de datos.
- La capacidad de mantener un lenguaje de consulta unificado.

La posibilidad de utilizar datos difusos está presente en cada operación del sistema, dado que todas las arquitecturas que forman parte de él han sido desarrolladas con dicha funcionalidad.

Con objeto de implementar esta arquitectura, el *Planificador de Estrategias de Consulta* deberá ser implementado integramente, mientras que las arquitecturas iniciales FIRST, FREDDI y DMFIRST, ya desarrolladas están funcionando actualmente.

Por contra, algunos de los inconvenientes que plantea esta arquitectura unificada son:

- *Posible disminución del rendimiento*, dado que hay un gran número de operaciones a realizar y aumenta la complejidad de las consultas.
- *Aumento en la complejidad del sistema.* El sistema debe gestionar un gran número de estructuras, por ejemplo las del catálogo y otras muchas operaciones y procedimientos.
- *Complejidad del desarrollo.* Es muy costoso el proceso de estudio del sistema actual para incorporar nuevos procesos o simplemente la definición de datos.
- *Dependencia del SGBD utilizado.* Se necesita una implementación diferente por cada SGBD utilizado, aunque el planteamiento teórico para el desarrollo del sistema sea el mismo que el planteado en este trabajo.

Dichos inconvenientes nos hacen plantearnos la puesta en marcha de este sistema multipropósito. La generación de una base de metaconocimiento más compleja aún que la que ya existía, al añadir dos nuevas relaciones, es un hecho nada deseable. Como tampoco lo es la dependencia que se crea del SGBD de Oracle©. Además, la solución propuesta puede parecer en un principio una solución temporal al problema puesto que la inclusión de nuevos tipos de datos u operaciones aumentará, lógicamente, la base de metaconocimiento, convirtiéndose en una tarea aún mas tediosa la comprensión de la misma, pudiendo provocar que se vuelvan a generar soluciones parciales, independientes del sistema global.

Como solución a este problema se propone rediseñar esta nueva arquitectura global de tal forma que sea posible la comunicación del usuario con la información sin la necesidad de emplear muchos recursos en ello, dejándola además, abierta a nuevas incorporaciones. El diseño de dicha arquitectura puede realizarse haciendo uso de las nuevas tecnologías que permiten modelar los metadatos que estructuran información, de manera abstracta e independiente del sistema sobre el que se vaya a desarrollar. Proponemos de esta manera, como solución a todos estos inconvenientes, modelar esta arquitectura mediante el uso de ontologías, utilizando las mismas para servir de interfaz entre el SGBD y el usuario. El estudio de esta propuesta será el centro de los siguientes capítulos de esta tesis.

Tabla 3.3: Relación Localización

Latitud	Longitud	TmediaT	Tmedia1	Tmedia2	Tmedia3	Tmedia4	...
41045	5478	4	0	NULL	NULL	NULL	...
41135	5598	4	0	NULL	NULL	NULL	...
4103	5705	4	0	NULL	NULL	NULL	...
41082	5675	4	0	NULL	NULL	NULL	...
40963	5636	4	0	NULL	NULL	NULL	...
41049	5578	4	0	NULL	NULL	NULL	...
...	...	...	...	...	...	...	...

...	PmediaT	Pmedia1	Pmedia2	Pmedia3	Pmedia4	...
...	4	2	NULL	NULL	NULL	...
...	4	2	NULL	NULL	NULL	...
...	4	2	NULL	NULL	NULL	...
...	4	2	NULL	NULL	NULL	...
...	4	2	NULL	NULL	NULL	...
...	4	2	NULL	NULL	NULL	...
...	...	...	...	...	...	...

Tabla 3.4: Relación Estructura

Latitud	Longitud	VegetacionT	VegetacionP1	Vegetacion1	...
41045	5478	3	1	4	...
41135	5598	3	1	4	...
4103	5705	3	1	4	...
41082	5675	3	1	4	...
40963	5636	3	1	2	...
41049	5578	3	1	4	...
...	...	...	...	...	...

...	Tipo_esT	Tipo_esP1	Tipo_es1	...
...	3	1	5	...
...	3	1	1	...
...	3	1	7	...
...	3	1	7	...
...	3	1	1	...
...	3	1	8	...
...	...	...	...	...

Tabla 3.5: Relación Fuzzy_Object_List de la BD de Suelos

OBJ#	COL#	FUZZY_ID	FUZZY_NAME	FUZZY_TYPE
Localizacion	TmediaT	0	'baja'	0
Localizacion	TmediaT	1	'media'	0
Localizacion	TmediaT	2	álta'	0
Localizacion	PmediaT	0	'baja'	0
Localizacion	PmediaT	1	'media'	0
Localizacion	PmediaT	2	álta'	0
Estructura	Vegetacion	0	'1'	1
Estructura	Vegetacion	1	'2'	1
Estructura	Vegetacion	2	'3'	1
Estructura	Vegetacion	3	'4'	1
Estructura	Vegetacion	4	'5'	1
Estructura	Vegetacion	5	'6'	1
Estructura	Vegetacion	6	'7'	1
Estructura	Tipo_es	0	'1'	1
Estructura	Tipo_es	1	'2'	1
Estructura	Tipo_es	2	'3'	1
Estructura	Tipo_es	3	'4'	1
Estructura	Tipo_es	4	'5'	1
Estructura	Tipo_es	5	'6'	1
Estructura	Tipo_es	6	'7'	1
Estructura	Tipo_es	7	'8'	1
Estructura	Tipo_es	8	'9'	1
...	...	...	...	...

Tabla 3.6: Relación Fuzzy_Nearness_Def de la BD de Suelos

OBJ#	COL#	FUZZY_ID1	FUZZY_ID2	DEGREE
Estructura	Tipo_es	0	1	0.4
Estructura	Tipo_es	0	2	0.4
Localizacion	Tipo_es	0	3	0.4
Localizacion	Orientacion	0	4	0.4
Localizacion	Orientacion	0	5	0.4
Localizacion	Orientacion	0	6	0.4
Localizacion	Orientacion	1	2	0.4
Localizacion	Orientacion	1	3	0.4
Localizacion	Orientacion	1	4	0.4
Localizacion	Orientacion	1	5	0.4
Localizacion	Orientacion	1	6	0.4
Localizacion	Orientacion	2	3	0.4
Localizacion	Orientacion	2	4	0.4
Localizacion	Orientacion	2	5	0.4
Localizacion	Orientacion	2	6	0.4
Localizacion	Orientacion	3	4	0.4
Localizacion	Orientacion	3	5	0.4
Localizacion	Orientacion	3	6	0.4
Localizacion	Orientacion	4	5	0.4
Localizacion	Orientacion	4	6	0.4
Localizacion	Orientacion	5	6	0.4
...	...	...	...	...

Tabla 3.7: Relación Fuzzy_Label_Def en la BD de Suelos

OBJ#	COL#	FUZZY_ID	ALFA	BETA	GAMMA	DELTA
Localizacion	TmediaT	0	0	0	6.5	8.5
Localizacion	TmediaT	1	8.5	10.5	12.5	14.7
Localizacion	TmediaT	2	14.7	16.9	21.0	21.0
Localizacion	PmediaT	0	183	183	315	490
Localizacion	PmediaT	1	490	664	731	818
Localizacion	PmediaT	2	818	905	1287	1287
...	...	...	...	...	...	...

Tabla 3.8: Relación Fuzzy_Col_List en la BD de Suelos

OBJ#	COL#	F_TYPE	LEN	COM
Localizacion	TmediaT	2	NULL	"Localizacion.Tmedia"
Localizacion	PmediaT	2	1	"Localizacion.Pmedia"
Estructura	Vegetacion	3	NULL	"Estructura.Vegetacion"
Estructura	Tipo_es	3	1	"Estructura.Tipo_es"
...	...	...	...	...

Tabla 3.9: Relación Fuzzy_Aprox_Much en la BD de Suelos

OBJ#	COL#	MARGEN	MUCH
Localizacion	TmediaT	4	10
Localizacion	PmediaT	50	300

Tabla 3.10: Relación Extended_Tab_Column en la BD de Suelos

OBJ#	COL#	COL_TYPE
Localizacion	TmediaT	0
Localizacion	TOrientacion	0
Estructura	Vegetacion	0
Estructura	Tipo_es	0

Tabla 3.11: Relación Extended_Tables en la BD de Suelos

OBJ#	TAB_TYPE	ORIG
Localizacion	0	1
Estructura	0	1
DFD1	1	-
DFD2	1	-

Tabla 3.12: Relación DmFsql_Project en la BD de Suelos

PROJECT_NAME	OWNER	OBJ#	STATUS-_FGD	THOLD-_ANT_FGD
Localizacion_PRJ	OWNER	Localizacion	-	0.6
Localizacion_PRJ	OWNER	Estructura	-	0.8

THOLD_CON_FGD	CONFIDENCE_FGD	SUPPORT_FGD	...
0.7	c	s	...
0.8	c	s	...

Tabla 3.13: Relación DmFsql_Col_List en la BD de Suelos

PROJECT_NAME	COL-_TYPE	COL#	FUZZY_-COMP_FGK	THOLD-_FGD
Localizacion_PRJ	A	TmediaT	FEQ	-
Localizacion_PRJ	Q	PmediaT	FEQ	-
Localizacion_PRJ	A	Vegetacion	FEQ	-
Localizacion_PRJ	Q	Tipo_es	FEQ	-

Tabla 3.14: Relación Ded_Intensional_Catalog de la Bd de Suelos

ID_PRED	MARCADO	NVARS
DFD1	1	2
DFD2	1	2

Tabla 3.15: Relación Ded_Int_Table_Description de la BD de Suelos

Table_ID	Rule_Id
DFD1	1
DFD2	1

Tabla 3.16: Relación Ded_Rule_Description de la BD de Suelos

Table_ID	Rule_Id	Pred_Id	Occ_Number	Negated	Type
DFD1	1	2	1	0	0
DFD1	1	-	2	0	2
DFD1	1	-	3	0	2
DFD2	1	2	1	0	0
DFD2	1	-	2	0	2
DFD2	1	-	3	0	2

Tabla 3.17: Relación Ded_Predicate_Description de la BD de Suelos

Table-_ID	Rule-_Id	Pred-_Id	Occ-_Number	Var-_Id	Col-_Id	Source_Col
DFD1	1	2	1	3	1	TmediaT
DFD1	1	2	1	4	2	PmediaT
DFD2	1	2	1	3	1	VegetacionT
DFD2	1	2	1	4	2	Tipo_esT

Tabla 3.18: Relación Ded_Condition_Description de la BD de Suelos

Table-_ID	Rule-_Id	Pred-_Id	Occ-_Number	Var-_Id1	Var-_Id2	ComOp	Thold
DFD1	1	-	2	3	1	6(FEQ)	0.6
DFD1	1	-	3	4	2	6(FEQ)	0.7
DFD2	1	-	2	3	1	6(FEQ)	0.8
DFD2	1	-	3	4	2	6(FEQ)	0.8

Capítulo 4

Ontología para la Representación del Conocimiento Difuso (FKRO)

4.1. Introducción

Tal y como se expuso en el apartado 3.4.5, la arquitectura que permite combinar las operaciones de manejo de datos difusos, estructuras lógicas y tareas de minería de datos en un único sistema presenta algunos inconvenientes. Uno de los más destacados consiste en la complejidad que el *Servidor Multipropósito Unificado* tiene a la hora de gestionar la información (definir estructuras, relaciones, procesos en el sistema) o ampliar el sistema incluyendo nuevos tipos de datos u operaciones. Dicha complejidad repercute directamente en el aumento de recursos para la comprensión del funcionamiento del sistema y su consiguiente explotación. Por otro lado, a pesar de que el planteamiento de las arquitecturas se ha tratado de hacer independiente de la plataforma del SGBD en la que se haya implementado, es realmente difícil desvincular completamente el sistema de la misma puesto que el catálogo de datos y el tipo de datos con los que se trabaja requieren su presencia.

Dado que existen metodologías para la representación del conocimiento que permiten mantener la información de un dominio lo suficientemente estructurada y clasificada para permitir la independencia de los datos con respecto del sistema de información en que sonn representados, se plantea la definición de una Ontología para representar la información asociada al *Servidor Multipropósito* expuesto. Dicha ontología será una

meta-ontología o una *Ontología Representacional* (véase definición en Anexo A), puesto que conceptualiza los formalismos para representación del conocimiento difuso, deductivo, etc.

Una ontología con el objetivo antes mencionado permitirá la definición de la Base de Metaconocimiento en forma de conceptos exclusivamente (usando por ejemplo una jerarquía de clases), en lugar de como se encuentra planteada actualmente, como un conjunto de relaciones y atributos del catálogo de un SGBD en particular. Dicha definición además nos facilitará el plantear de forma abstracta el tipo de datos que se utilizan para almacenar información de muy diversos tipos (difusa, deductiva, de minería de datos, etc.) en la base de datos y las restricciones propias que pueda imponer un SGBD determinado a la hora de definir las relaciones en el mismo. Finalmente esta propuesta de ontología también será susceptible de posteriores extensiones de forma inmediata a otros sistemas de representación de bases de datos difusos como pueden ser el orientado a objetos, almacenes de datos, etc.

La figura 4.1 contextualiza el lugar de la ontología en nuestra propuesta de arquitectura de un SGBD multipropósito extendido. Dicha ontología actúa como interfaz entre el usuario y el SGBD, proporcionando así una alternativa al tipo de acceso a la información ordinario (esto es, del usuario a los datos a través de SGBD directamente).

En este trabajo de tesis se define la primera versión de ésta ontología que es la correspondiente a la parte de la arquitectura unificada de la figura 3.4 que permite representar y gestionar información difusa. Dicha ontología permitirá definir de forma clara todas las entidades necesarias para que el almacenamiento y manipulación de la información difusa sea independiente del contexto en que se incluya.

La ontología para la representación de información imprecisa en el Modelo Relacional que se propone se encuentra dividida en dos subontologías que definen la información del esquema de modo distinto:

- **Sub-Ontología para la Representación del Catálogo**. Esta ontología, representa la información del catálogo del sistema, y su instanciación permite la definición completa de un esquema difuso o clásico utilizando el estándar SQL 2003 [Cal06, fSIIT03] extendido con la propuesta dada en FIRST [Med95] para la manipulación de datos difusos. Se trata de una meta-ontología, puesto que contiene metaclases, que permiten a posteriori, definir los datos o tuplas que esté representando dicho esquema.

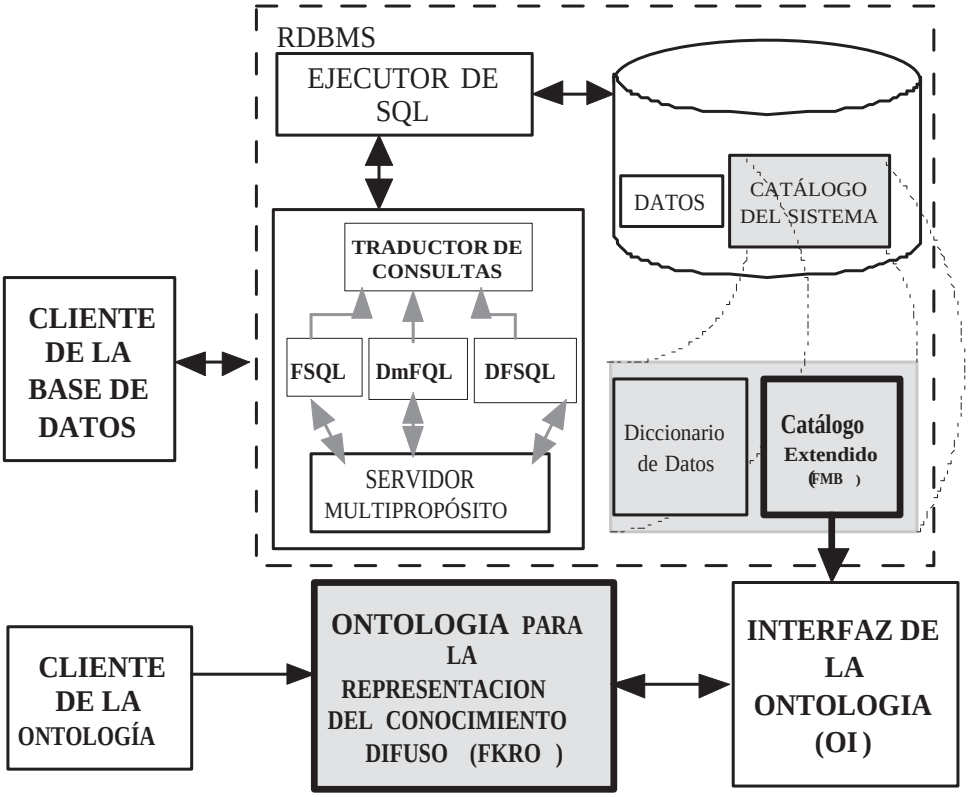


Figura 4.1: Relación de la Ontología con el *Servidor Multipropósito Unificado*

- **Sub-Ontología para la Representación del Esquema de BD Difusas**. Esta ontología, será generada a partir de la subontología anterior y representa un esquema de BDD (Bases de Datos Difusas) sobre un dominio concreto. Su objetivo es el de aportar la posibilidad de instanciar los datos o tuplas que dicho esquema pueda contener.

La ontología en global se denominará *Ontología para la Representación del Conocimiento Difuso (FKRO, Fuzzy Knowledge Representation Ontology*) y establece las bases, la representación de la información imprecisa, para la representación a posteriori del resto de la información de la arquitectura que forma el *Servidor Multipropósito* descrito en el apartado 3.4.

Dicha ontología será descrita utilizando el modelado de datos UML

para su desarrollo. Una vez descrita, su representación final será implementada utilizando un lenguaje de definición de ontologías basado en Web, OWL [Bec, Ant03]. La elección de este lenguaje de representación viene motivada porque permite que los datos sean comprensibles en este entorno web, además de que se trata de un lenguaje cada vez más aceptado y estandarizado que esta produciendo un gran número de aplicaciones nuevas con las que interactuar. La descripción de las diferentes ontologías descritas en este capítulo se incluyen en el CD que acompaña a este trabajo.

4.2. Ontología para la Representación del Conocimiento Difuso

4.2.1. descripción

La *Ontología para la Representación del Conocimiento Difuso* [Bla08b] define la información difusa que se encuentra representada en una BD Relacional Difusa. Esta ontología actúa como una interfaz entre el SGBDD y el usuario y/o los programas de aplicación, de tal forma que hace transparente el modo en que la información esta representada en el SGBDD (mediante el diccionario de datos).

La estructura del Modelo Relacional Extendido para la manipulación de información imprecisa se encuentra representada en la *Sub-Ontología del Catálogo Extendido*, y por tanto la definición de una BDD será realizada a través de la instanciación de dicha sub-ontología, olvidando así representaciones particulares que los SGBDs pueden hacer. De esta forma se define el esquema de la BDD. Sin embargo, cualquier interacción con dicho esquema para la manipulación de los datos de la BDD, requerirá de su conversión explícita a forma de ontología. El proceso de conversión consiste en generar a partir de las instancias de la *Sub-Ontología del Catálogo Extendido* una nueva ontología, denominada de forma genérica, *Sub-Ontología para la Representación del Esquema de BDD*. Dicho proceso puede ser automatizado y su resultado, la *Sub-Ontología para la Representación del Esquema de BDD* correspondiente, permitirá la manipulación de los datos difusos también de forma transparente al SGBDS en el que se encuentren almacenados.

La *Ontología para la Representación del Conocimiento Difuso*, engloba ambas sub-ontologías descritas, que tratan de representar la misma información, un esquema relacional de BDD, aunque la manera de rep-

resentarlo difiere en gran medida:

- Mediante la instanciación de la *Sub-Ontología del Catálogo Extendido* se permite la definición completa del esquema de datos difusos, especificando todas las restricciones propias del Modelo Relacional y de la extensión teórica para la representación de datos difuso propuesto en GEFRED.
- Mediante la generación de la *Sub-Ontología para la Representación del Esquema de BDD* se permite la compartición de esquemas de datos difusos con el entorno, y por supuesto la definición de datos (tuplas) sobre la misma.

Ambas representaciones son dependientes, es decir, una (la del *Esquema*) es generada a partir de la otra (la instanciación del *Catálogo*) y por tanto, deben coexistir las dos al menos en el momento de su generación.

A su vez, ambas están vinculadas puesto que comparten las clases necesarias para representar la información (véase sección 4.4.2 para conocer los detalles) y por tanto ambas requieren de la importación de la *Sub-Ontología del Catálogo Extendido*. Asimismo, los datos del dominio relativos a etiquetas y relaciones de similitud, deben existir también en ambas definiciones, y en mayor medida sobre la *Sub-Ontología para la Representación del Esquema de BDD* dado que es dónde se han de definir los valores de las tuplas.

4.2.2. Ejemplo

En la figura 4.2 se muestra gráficamente en qué consiste la *Ontología de Representación de Conocimiento Difuso* en el Modelo Relacional. Como puede observarse, el núcleo fundamental, la *Sub-Ontología del Catálogo Extendido* permitirá representar cualquier BDD (su esquema) mediante su instanciación. En el ejemplo se plantean cuatro BDD, la de la *Clínica Veterinaria*, otra de *Suelos*, *Características de Diamantes*, o *Liga de Baloncesto*.

Una vez instanciadas el proceso de Generación de la *Sub-Ontología para la Representación del Esquema de BDD* sería único para cada BDD. Cada uno de los esquemas descritos generará una ontología de su esquema propio, siguiendo los pasos establecidos en la sección 4.4. Estas cuatro *Sub-Ontologías para la Representación del Esquema de BDD*, al instanciarlas permitirán la inserción de datos (tuplas) referentes a los datos que se han modelado en sus esquemas correspondientes.

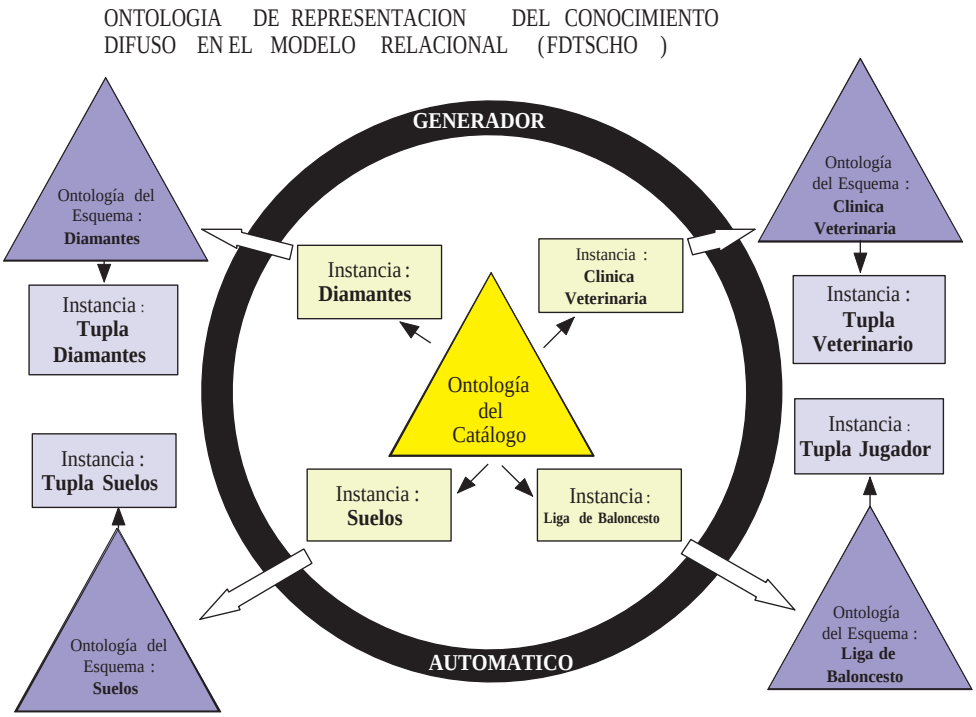


Figura 4.2: Ejemplos de vínculos entre las sub-ontologías del Esquema y Catálogo para cuatro BDD

4.3. Sub-Ontología para la Representación del Catálogo Extendido

4.3.1. Justificación de la Sub-Ontología

En la representación de la información que existe en una base de datos, una parte fundamental de la misma es el esquema de la base de datos, que describe la estructura de la información siguiendo un modelo de representación determinado. Dado que el Modelo Relacional es el más extendido y usado en el entorno de las Bases de Datos, será éste el elegido para ser representado por la ontología.

Puesto que se intenta estar lo más desvinculado de cualquier implementación concreta de un SGBDR, la representación del estándar ANSI SQL2003 [fSIIT03] nos parece la propuesta más razonable. En cuanto a la representación de información imprecisa, muchas propuestas han extendi-

do al Modelo Relacional en los últimos años, tal y como se ha descrito en el capítulo 3. El modelo FIRST propuesto por Vila et al. [Med95, Gal99] se plantea en este trabajo dada su completitud en la descripción de este tipo de información.

Un esquema de bases de datos difusas o clásicas, no deja de ser un mecanismo de descripción de la información que hay almacenada en la BD en forma de tuplas en las relaciones o tablas. En este caso, dicha descripción o esquema simplemente establecerá las restricciones mediante las cuales los datos, en última instancia, estarán almacenados en la BD. Por tanto, cuando se habla de esquema de base de datos se debe hablar de *metadatos*, puesto que los esquemas son la representación de la estructura de los datos finales.

Sin embargo la definición de *metadatos* o esquemas en un Sistema de Gestión de Bases de Datos cualquiera supone la generación automática de las estructuras necesarias para almacenar la información. Por ejemplo, cada vez que se define una relación como *Jugadores* automáticamente se genera en el SGBD una tabla *Jugadores* con todas las características y restricciones descritas en DDL (lenguaje de definición de datos) de SQL, tal y como se muestra a continuación:

```
CREATE TABLE Jugadores (
  Nombre VARCHAR2(60) NOT NULL PK,
  Equipo   VARCHAR2(30) NOT NULL REFERENCES TEAM(TName),
  Altura NUMBER(4,2) NOT NULL,
  ColorPelo FTYPE3 (2) NOT UNDEFINED,
  FechaNac DATE CHECK (>=1980)
  CONSTRAINT minHeight CHECK Altura BETWEEN 1.70 AND 2.50)
```

Al realizar esta definición de los metadatos o esquema de una BD realmente se están instanciando las tablas del catálogo o diccionario de datos de un SGBDR. En dicho catálogo únicamente se almacena la información que esta sentencia proporciona, no existen datos referentes a ningún jugador, únicamente datos referentes a las características de la relación.

La inserción de un nuevo registro en la tabla *Jugadores* implica la creación de dicha relación (proceso automático en un SGBD al ser incluida su información en el Catálogo) y la instanciación de la misma.

Esto no ocurre de la misma manera cuando se representa información utilizando ontologías. Una ontología, tal y como está definida, esta compuesta por un conjunto de clases, propiedades y restricciones que per-

miten describir una realidad concreta. La instanciación de dicha ontología nos permite definir la información que queremos representar basándonos en la misma. Es decir, que si nuestra ontología trata de representar el estándar ANSI extendido con representación de datos difusos, su instanciación nos permite definir esquemas tal y como hemos visto en el ejemplo anterior con la tabla *Jugadores*. Pero únicamente hasta ahí podremos llegar.

Es un hecho que se pueden instanciar cuantos elementos se deseen de una clase, sin embargo una instancia no puede ser instanciada de nuevo para definir mas conceptos asociados con la realidad que representa. Consecuentemente, no podríamos actuar de la misma forma que actuaría un SGBD con las estructuras descritas usando DDL (Lenguaje de Definición de Datos). Siguiendo con el ejemplo anterior, si definimos en la ontología, la clase *Tabla*, e instanciamos la misma con la relación *Jugador*, no podríamos volver a instanciar tabla *Jugador* para almacenar el registro *(Juan López, Juventud, 1.99, Castaño, 10/7/1985)*.

Es por esta razón que la ontología se considera una *meta-ontología* o una *Ontología Representacional*, puesto que para solucionar este problema se propone la inclusión de metaclases en la definición de la *Ontología del Esquema* difuso. Estas metaclases permitirán que la generación de instancias sobre las mismas generen a su vez nuevas clases que, ya sí, podrán ser instanciadas para incluir los datos o tuplas (véase la figura 2.3 del capítulo anterior que ilustra dicha explicación).

A continuación se describe la *Sub-Ontología para la Representación del Catálogo Extendido*, a partir de ahora denominada *Ontología del Catálogo* por motivos de claridad. Se describirá cómo se ha construido, las clases que la integran, las metaclases definidas en la misma, y un ejemplo de utilización.

4.3.2. Metodología de Desarrollo

Para representar el esquema de una base de datos, debemos plantearnos cómo la información de este esquema está estructurada. Para realizar este proceso se ha tomado como analogía el catálogo del sistema dado por cualquier SGBD, que es el mecanismo que utilizan dichas aplicaciones para poder representar esta información. Sin embargo, cada SGBD define su propio catálogo de forma única, por lo que no tendría ningún sentido elegir alguna representación concreta dada por un SGBD comercial.

Siguiendo con esta analogía, la propuesta consiste en representar un catálogo que permita mantener un registro de toda la estructura de una

base de datos relacional desvinculándose de cualquier implementación concreta. Por tanto, la representación del estándar ANSI SQL2003 es la propuesta más razonable. Calero et al. [Cal05, Cal06] describe a modo de ontología dicha representación (ver sección 2.4.2) utilizando el lenguaje de modelado UML.

La propuesta de Calero carece de la descripción de los tipos de datos predefinidos, que proporciona el estándar. Dichos tipos de datos, también se encuentran desglosados a modo de jerarquía de conceptos en el trabajo de Pardede et al. [Par05], tal y como se ha descrito en la sección 2.4.1.

Como ya se definió anteriormente, una ontología debe representar el conocimiento compartido y consensuado por un conjunto de expertos acerca de una parcela concreta de la realidad. Partiendo de esta base, y dada la amplia difusión de las ontologías en la actualidad, se estima conveniente no comenzar a realizar una ontología desde cero, sino reutilizar ontologías que se encuentran ya desarrolladas. Una vez identificadas la ontologías adecuadas, se procederá a realizar sobre ellas diversos procesos que refinamiento y soporte, para dar lugar a la ontología final que contiene toda la información que se desea representar [Cor06].

A continuación se detalla el proceso para realizar la conceptualización de la ontología que en este trabajo se propone. Este proceso está dividido en varias etapas que se exponen a continuación y que pueden verse en la figura 4.3:

1. Recorte de la ontología de Calero et al. [Cal06, Cal05].

2. Recorte y extensión de la ontología de Pardede et al. [Par05] con los tipos de datos difusos definidos en *GEFRED*.

3. Mezcla de la ontología de Calero et al. [Cal06, Cal05]. recortada con la de Pardede et al. [Par05] extendida.

4. Extensión de la ontología resultante con la extensión al Modelo Relacional para definir la información imprecisa definida por el modelo denominado *GEFRED* y su implementación, denominada *FIRST* (véase sección B.1.1 del Anexo B).

Paso 1. Recorte de la Ontología de Calero et al. [Cal06]

Tal y como describe Calero et al. [Cal06], el ANSI SQL 2003 representa una extensión a las versiones anteriores ANSI SQL en tanto que

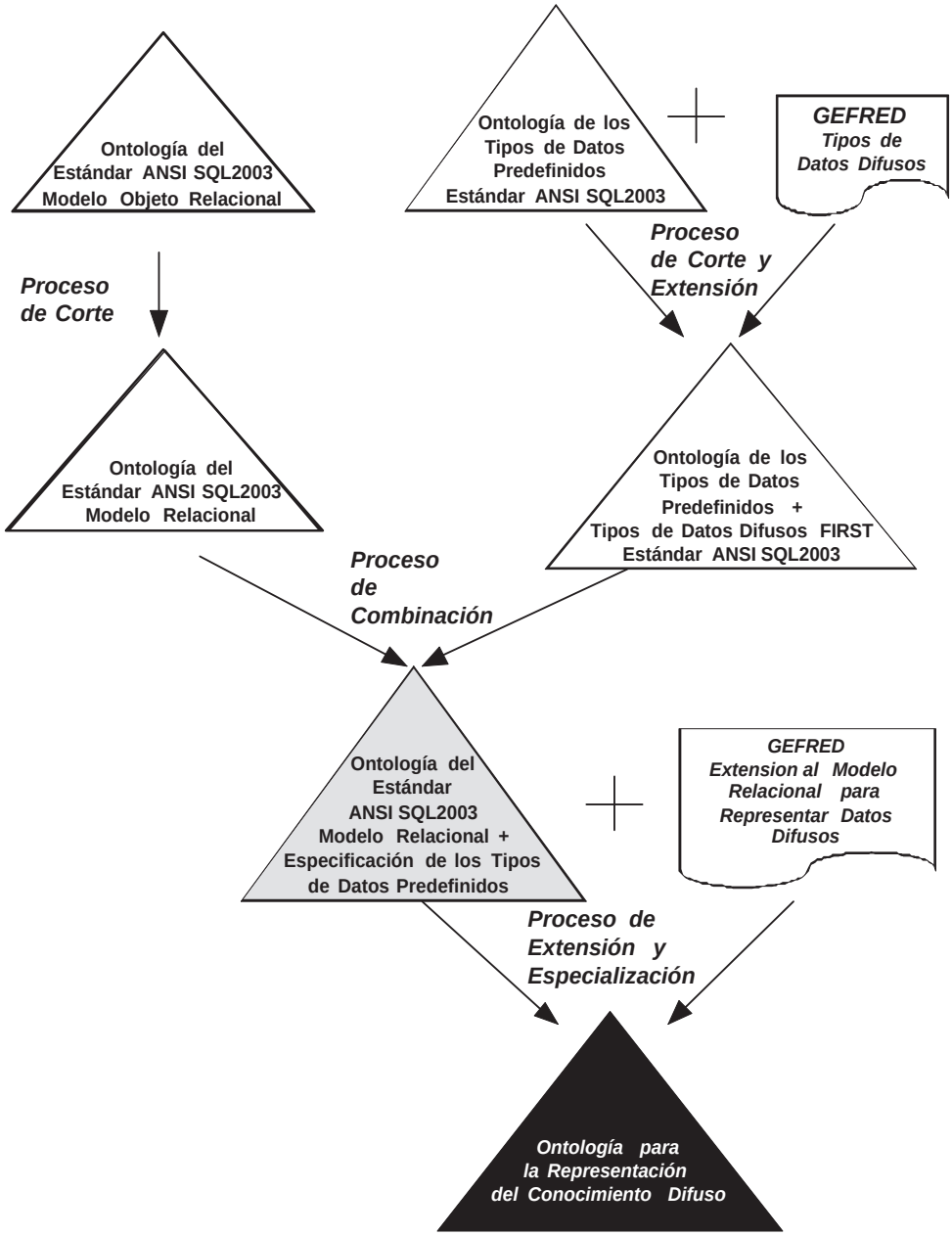


Figura 4.3: Proceso de Desarrollo de la *Ontología del Esquema*

añade al Modelo Relacional la manipulación de objetos y algunos elementos nuevos de lenguaje Web XML a lo que ya existía (véase [Eis04]).

El modelo que la *Ontología del Catálogo* trata de representar, se centra únicamente en el Modelo Relacional, y por lo tanto, la parte orientada a objetos no es un requisito en su representación. De esta manera la ontología de Calero et al. [Cal06] es recortada dejando únicamente las estructuras necesarias para representar el Modelo Relacional tal y como puede verse en la figura 4.4. Dichas estructuras se describen brevemente en la tabla 4.1 (también se encuentran descritas con mayor detalle en [Cal06]). También se ha eliminado parte de la representación del Modelo Relacional que resulta irrelevante para probar la manipulación de la representación imprecisa. Entre estos recortes se encuentran los tipos de datos compuestos y otras estructuras que complican el modelo en demasía con respecto a su grado de utilización como por ejemplo la representación de las claves candidatas o las columnas generadas por una expresión (muy útiles en tareas de minería de datos). Este recorte se ha realizado fundamentalmente para simplificar la representación de la ontología en este primer prototipo. Sin embargo la inclusión de dichas clases será algo inmediato en las fases de extensión de este trabajo.

Paso 2. Recorte y Extensión de la Ontología de Pardede et al. [Par05]

La clasificación de Pardede et al. [Par05] desglosa todos los tipos de datos base que el estándar ANSI ha ido declarando a lo largo de todas sus versiones. A este conjunto de datos se propone añadir aquellos que permiten representar datos difusos sobre una BD y que están definidos en el capítulo B.1.1. Los tipos de datos difusos que aparecerán en esta extensión son (véase sección B.1.2 para una descripción más detallada):

- *Tipo Difuso 1 o TD1.* Representa datos cuyos valores se basan en un referencial ordenado, pero que pueden ser consultados de forma difusa.
- *Tipo Difuso 2 o TD2.* Representa datos cuyos valores se basan en un referencial ordenado, pero pueden ser almacenados y/o consultados siguiendo una distribución de posibilidad utilizada para representar datos difusos.
- *Tipo Difuso 3 o TD3.* Representa datos cuyos valores se basan en un referencial discreto y en las relaciones de semejanza descritas explícitamente sobre este referencial.

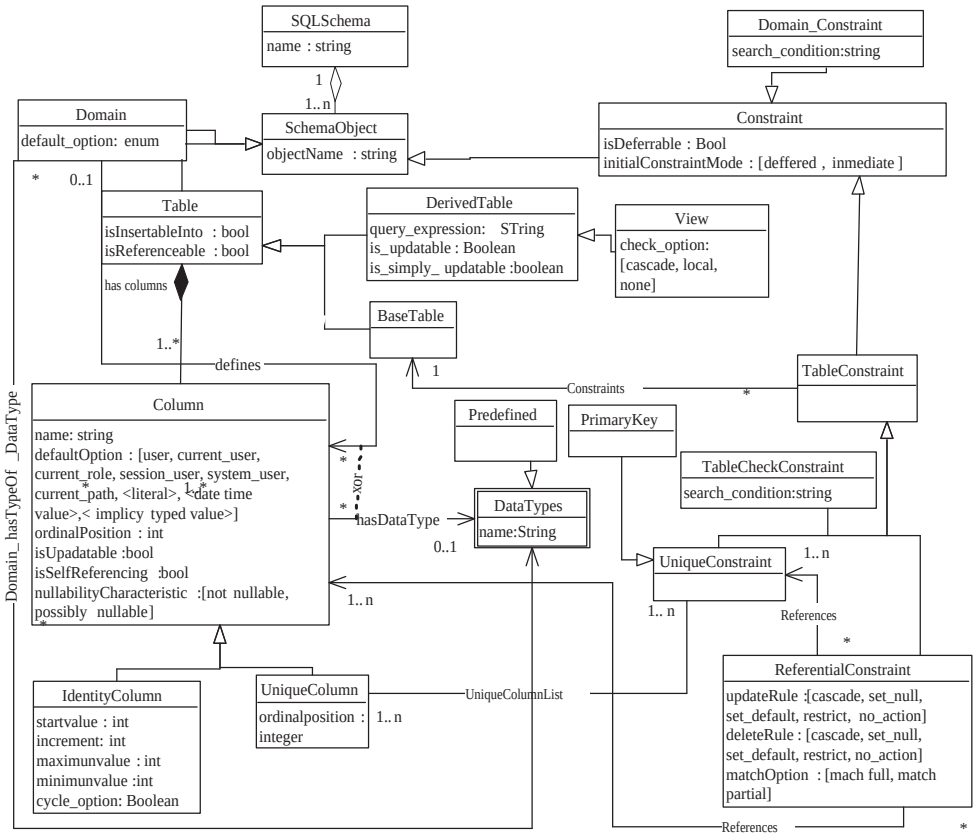


Figura 4.4: Ontología en UML del SQL4 de Calero et al. [Cal06] recortada

Tabla 4.1: descripción Breve de las Clases de la Ontología Recortada de Calero et al.

Clase	Superclase	descripción	Atributos
SQLSchema		Representa los esquemas de BD	
Schema-Object	SQLSchema	Es cualquier objeto del esquema	objectName
Table	Schema-Object	Es un objeto Relacion	isInsertableInto, isReferenceable
Derived-Table	Table	El resultado de una consulta	queryExpresion, isUpdatable, isSimplyUpdatable
BaseTable	Table	Tabla compuesta por atributos	
View		Tabla virtual basada en una consulta	CheckOption
Domain	Schema-Object	Conjunto de valores de un atributo agrupado, compuesto de un tipo de datos y restricciones	defaultOption
DataType		Conjunto de valores representables	
Predefined	DataType	Tipos de Datos Base	
Constraint	Schema-Object	Restricciones sobre la BD	isDeferrable, initialConstraintMode
Table-Constraint	Constraint	Restricciones sobre Tablas	
Domain-Constraint	Constraint	Restricciones sobre Dominios	searchCondition
TableCheck-Constraint	Table-Constraint	Restricciones de Negocio (Check Constraint) sobre Tablas	searchCondition
Unique-Constraint	Table-Constraint	Restriccion de Clave Única	
Referential-Constraint	Table-Constraint	Restriccion Referencial	updateRule, setDefault, deleteRule, matchOption
PrimaryKey	Unique-Constraint	Representa la clave primaria	
Column		Columna o atributo	name, defaultOption, ordinalPosition, isUpdatable, isSelfReferencing, nullabilityCharacteristic
Identity-Column	Column	Columna que automáticamente actúa como una secuencia	startVal, increment, maximunVal, minimunVal, cycleOp
Unique-Column	Column	Columna que contiene valores únicos	ordinalPos

Así pues la clasificación de Pardede et al. [Par05] quedaría representada tal y como se indica en la figura 4.5. En la misma, apartado A, se describe explícitamente dónde entraría la extensión difusa en la clasificación. En el apartado B se describen con mas detalle los tipos de datos difusos y su integración con el resto de la clasificación de Pardede et al. [Par05]. Como podemos observar, sólo aquellos datos difusos que se basan en un referencial numérico se relacionan con el resto de la clasificación (Tipo Difuso 1 y 2), es decir definen explícitamente el referencial en el que se basan.

Tal y como ocurre en el paso 1 con la clasificación de Calero et al. [Cal06] se propone recortar la clasificación de Pardede et al. [Par05] excluyendo los tipos de datos complejos y dejando únicamente los tipos base, también llamados predefinidos (para así poder mezclar ambas ontologías en el paso siguiente). Por otro lado, en esta clasificación también se propone un nuevo recorte que excluye los tipos de datos menos comunes, es decir, mantenemos únicamente aquellos que son más susceptibles de ser implementados en la mayoría de los SGBDs del mercado en este primer prototipo de ontología, tal y como se ve en la figura 4.6.

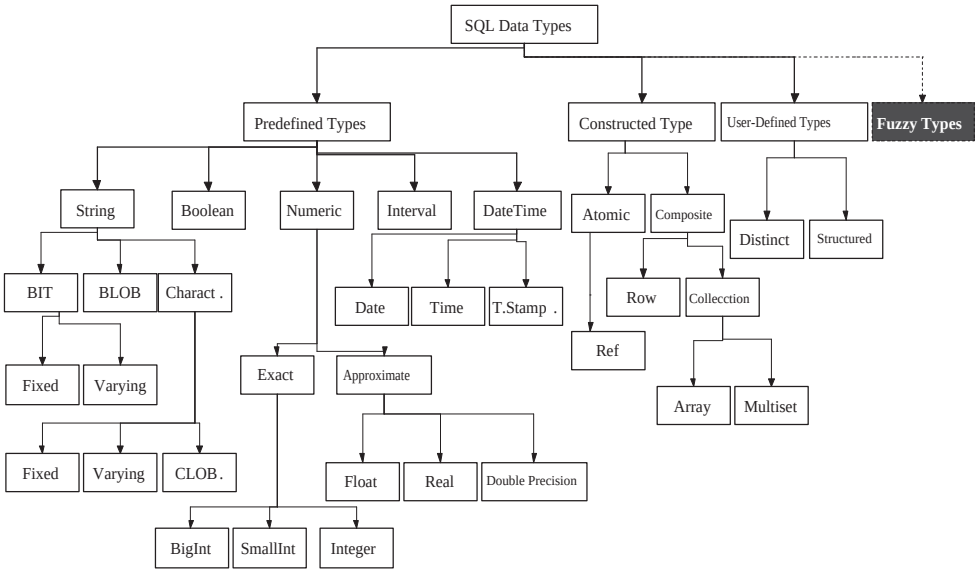
Uniendo las diferentes modificaciones a la clasificación de Pardede et al. [Par05] la clasificación quedaría tal y como se describe en la figura 4.7.

La clase *FDTOrder* es una clase abstracta que agrupa a los tipos de datos difusos que representan valores sobre un referencial ordenado numérico. Ambos tipos de datos (*TD1* y *TD2*) deberán ser definidos por el referencial numérico sobre el que se basan, a través de la relación *hasDataType* y por los atributos *margen* y *much* que se usan para la comparación de valores dentro del dominio difuso. Ambos valores tienen la cardinalidad establecida a 1, puesto que por cada tipo de dato sólo tienen un valor asociado al mismo.

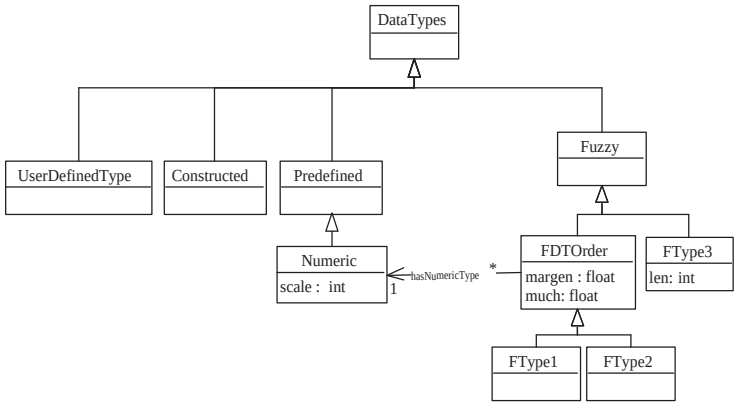
Paso 3. Proceso de Mezcla

Una vez que disponemos de las diferentes ontologías fuente a partir de las cuales se va a generar la nueva, el siguiente proceso es el de unificar dichas ontologías usando las partes que tengan en común para formar una ontología mayor y más especializada. De esta forma procederemos a identificar las partes comunes que ambas ontologías tienen:

- La clase *DataTypes* aparece en la Ontología Recortada y Extendida de Pardede et al. [Par05] mostrada en la figura 4.7, de la cual



A) Taxonomía de Pardede & Wenny Rahayu , de los tipos de datos del SQL4 junto con la inclusión de los tipos de datos difusos .



B) Tipos de Datos Difusos relacionados con la taxonomía general de Pardede et al.

Figura 4.5: Extensión de la ontología de Pardede et al. [Par05] con los datos difusos

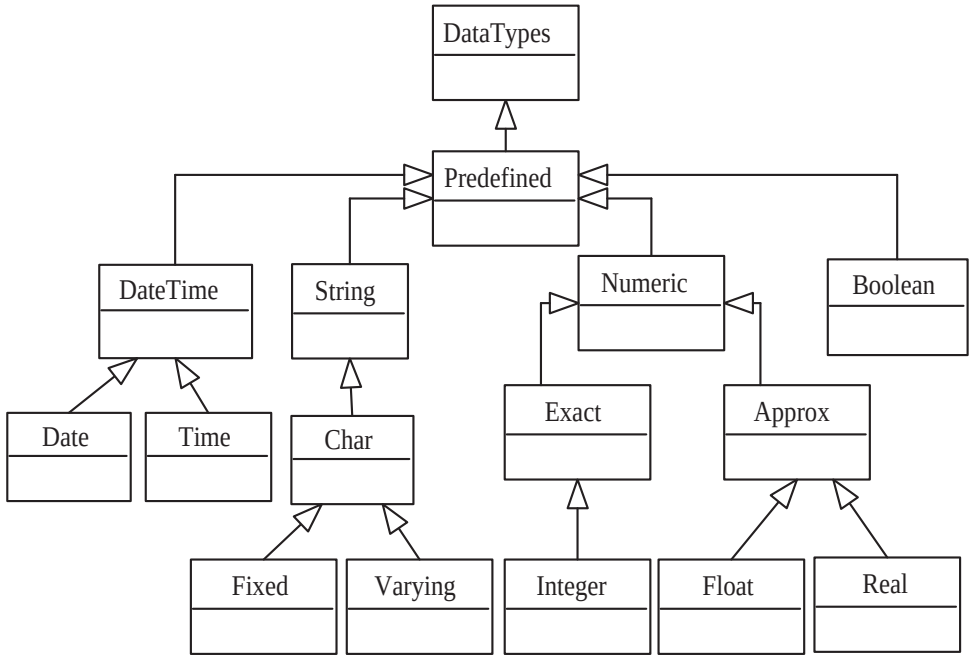


Figura 4.6: Clasificación de Pardede et al. [Par05] recortada

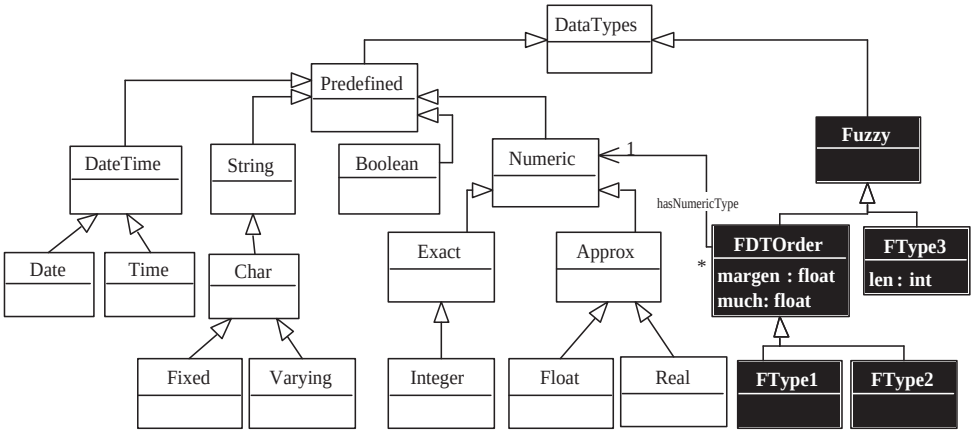


Figura 4.7: Clasificación de Pardede et al. [Par05] recortada con la inclusión de datos difusos

depende *Predefined* y *Fuzzy* que representan los tipos de datos que se van a usar en esta ontología.

- La clase *DataTypes* aparece en la Ontología de Recortada de Calero et al. [Cal06] tal y como se puede ver en la figura 4.4, con la subclase *Predefined* dependiente de ella. Sin embargo dicha clase, carece de la especificación jerárquica de los elementos que la componen. Además la clase *DataTypes* cuenta con el atributo *Name*, que representa el nombre de del tipo de datos.

Dado que las clases especificadas anteriormente coinciden en significado y nombre, la unificación de las ontologías es inmediata. Este proceso de mezcla incorpora todos los elementos que aparecen en una u otra ontología, de tal forma que se forma una nueva lo más completa posible tal y como vemos en la figura [1] 4.8.

Paso 4. Extensión o de Definición de Datos Difusos

Este proceso consiste en extender la ontología generada a través de los procesos de recorte y mezcla anteriores, para poder representar información imprecisa en el Modelo Relacional representado por dicha ontología (véase figura 4.8).

La extensión de la ontología consta de varias fases listadas a continuación y descritas con detalle en las subsecciones siguientes:

- Redefinición del concepto *Columna* para poder englobar aquellas que contienen datos difusos.
- Extensión del número de restricciones que pueden definirse en el Modelo Relacional para englobar aquellas que tengan que ver con datos difusos.
- Definición de metaclases necesarias para formar la ontología de datos.
- Definición del concepto de *Dominio Difuso*, junto con la definición de etiquetas lingüísticas, valores discretos y relaciones de similitud.
- Definición de aquellas estructuras que permiten el almacenamiento de valores difusos (distribución posibilidad, estructuras trapezoidales, triangulares, intervalares, etc.).

[1] Todas las clases que aparecen en la figura 4.8 se encuentran descritas en los apartados anteriores.

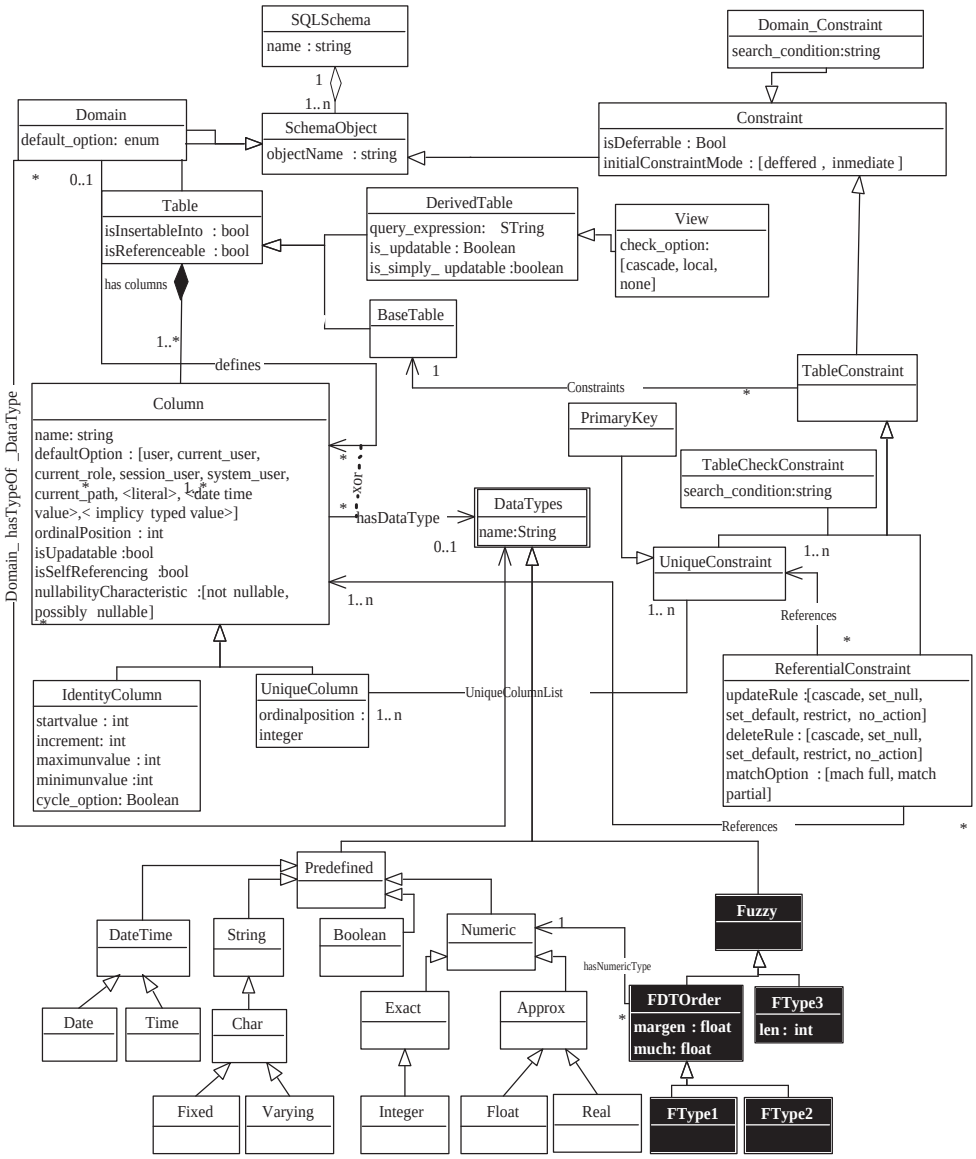


Figura 4.8: Ontología de Calero et al. [Cal06] y Pardede et al. [Par05] del SQL4 y Tipos de Datos Difusos mezclada

4.3.3. descripción de la Ontología del Catálogo Extendido

El catálogo de un SGBD nos permite almacenar los metadatos, es decir, la información que representa la estructura de la información que la BD va a permitir almacenar. Si a este catálogo que ya tenemos propuesto como ontología en la figura 4.8, le añadimos la extensión descrita en FIRST (véase sección B.1.2) que presenta las estructuras para representar información imprecisa, obtendremos la que hemos denominado *Ontología del Catálogo Extendido*.

A continuación se presentan de manera desglosada cada uno de los cambios que son requeridos en la presente ontología para obtener dicha *Ontología del Catálogo Extendido*.

4.3.3.1. Columnas Difusas

En la representación de la ontología del estándar SQL4, se debe modificar el concepto de Columna para que sea especializado en dos categorías (*Columna_Base* y *Columna_Difusa*), quedando las clases relacionadas con ella de la siguiente manera (véase figura 4.9 para ilustrar las definiciones siguientes):

- *Columna*: es la clase raíz, representa todas las columnas de la BD. Sin embargo esta clase será considerada como abstracta, y únicamente sus subclases deberán ser instanciadas. Los atributos de dicha clase son los mismos que los definidos anteriormente, a excepción de la *nullabilityCharacteristic* (que representa si la clase puede contener valores nulos), que aparecerá como atributo en la subclase *Columna_Base*.

- *Columna_Base*: s aquella que representa las columnas clásicas en una BD. Seguirá con las mismas relaciones que en la anterior definición lo hacía la clase *Columna*, de hecho es este tipo de columna la que va a permitir mantener las diferentes restricciones de integridad que exige el Modelo Relacional. El tipo de dato vendrá determinado por la relación *hasDataTypes* con los tipos de datos predefinidos (no olvidemos que el resto de tipos de datos compuestos los hemos excluido de esta propuesta) o bien por la relación *defines* que relaciona una columna con un dominio. Aunque no es la más utilizada en la definición de datos, se ha optado por representar esta relación dado que muchos SGBDs la utilizan implícitamente en muchas definiciones (en cualquier caso dicha relación se describirá con más de-

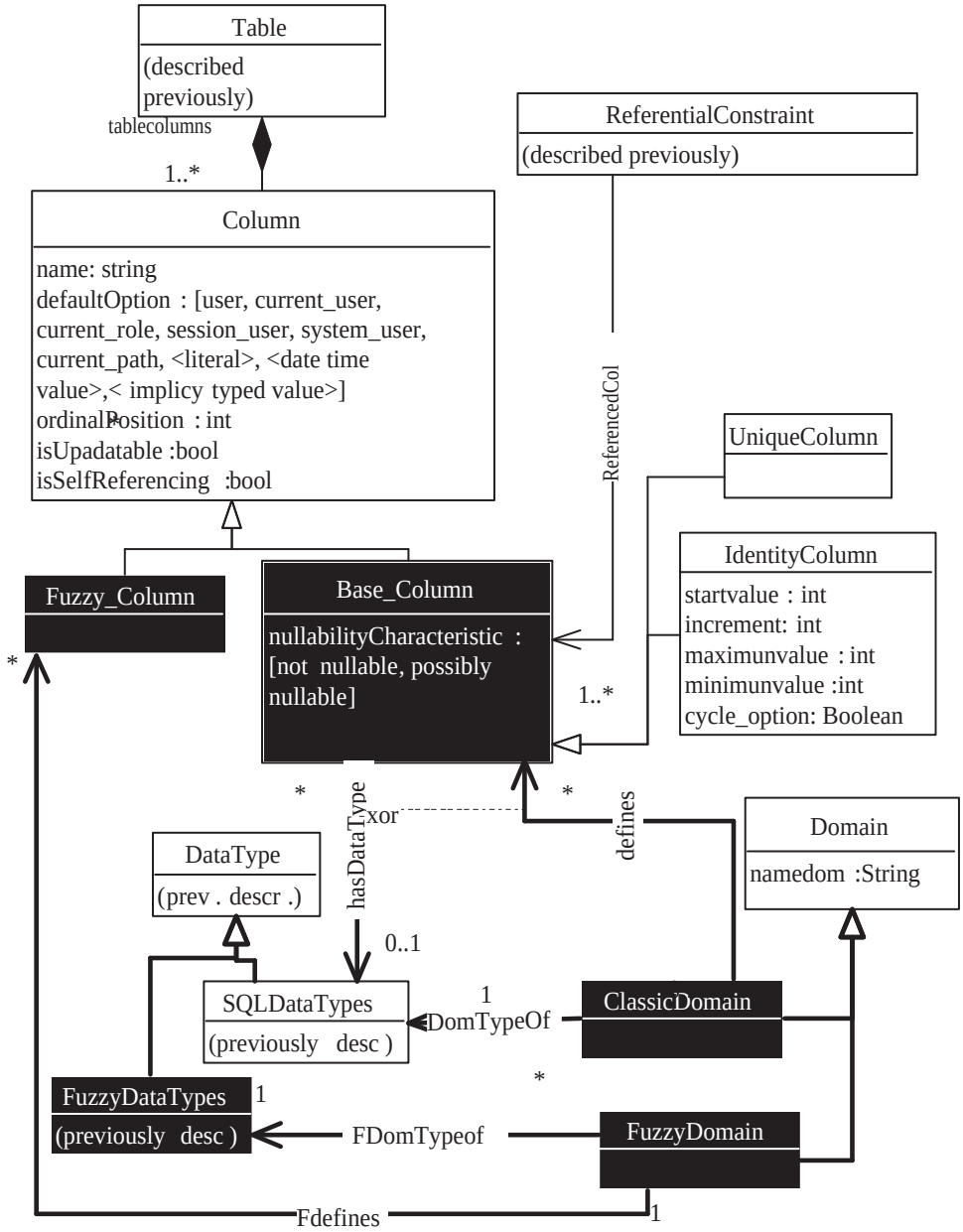


Figura 4.9: Especialización de la clase *Columna*

talle en el apartado siguiente). La definición del tipo de dato, sea directamente, o sea a través de un dominio, sólo puede ser una, tal y como especifica la relación *xor* en el diagrama. En cuanto a los atributos, esta clase es una especialización del concepto Columna, contiene todos los de dicha superclase e incluye *nullabilityCharacteristic*, completando así la lista de atributos que antes representaba *Columna* en la ontología del ANSI SQL4 descrita en la sección anterior.

- *Columna_Difusa*: clase que representa todas aquellos atributos que pueden contener datos difusos o bien ser consultados utilizándolos. Una columna difusa por definición sólo podrá contener datos o bien difusos o bien que sean susceptibles de ser consultados de forma difusa, por tanto, el tipo de datos será representado por la relación *Fdefines*. Esta relación garantiza que una columna difusa esté vinculada con un dominio difuso únicamente, y no en sentido inverso. Es por esta razón que no existe un vínculo directo a los tipos de datos difusos, ya que cuando se define un tipo de dato difuso no suele tratarse de un único valor sino que suele tratarse de un dominio que viene acompañado por definiciones previas de etiquetas lingüísticas o relaciones de similitud entre valores discretos, usados en el proceso de definición de datos o en consultas. En cualquier caso, esta relación será descrita con más detalle en el apartado de dominios difusos siguiente. En cuanto a atributos concretos, esta relación no tendrá más que los heredados por la superclase. No existen restricciones definidas sobre columnas difusas porque dichas restricciones se establecen sobre los dominios difusos, como ya veremos en el subapartado de dominios y restricciones difusas siguiente.

4.3.3.2. Dominios Difusos

Tal y como se describió brevemente en el párrafo anterior correspondiente a las columnas, un dominio representa el conjunto de valores y restricciones que un atributo puede contener. Un gran número de SGBDs, dependiendo del tipo de dato que se este definiendo crea dominios en lugar de establecer un vínculo a un tipo de dato predefinido, pero de forma invisible al usuario.

Cuando se trata de gestionar información imprecisa la presencia de dominios es imprescindible, dado que normalmente a la definición de un tipo de dato difuso siempre acompaña un conjunto de valores además

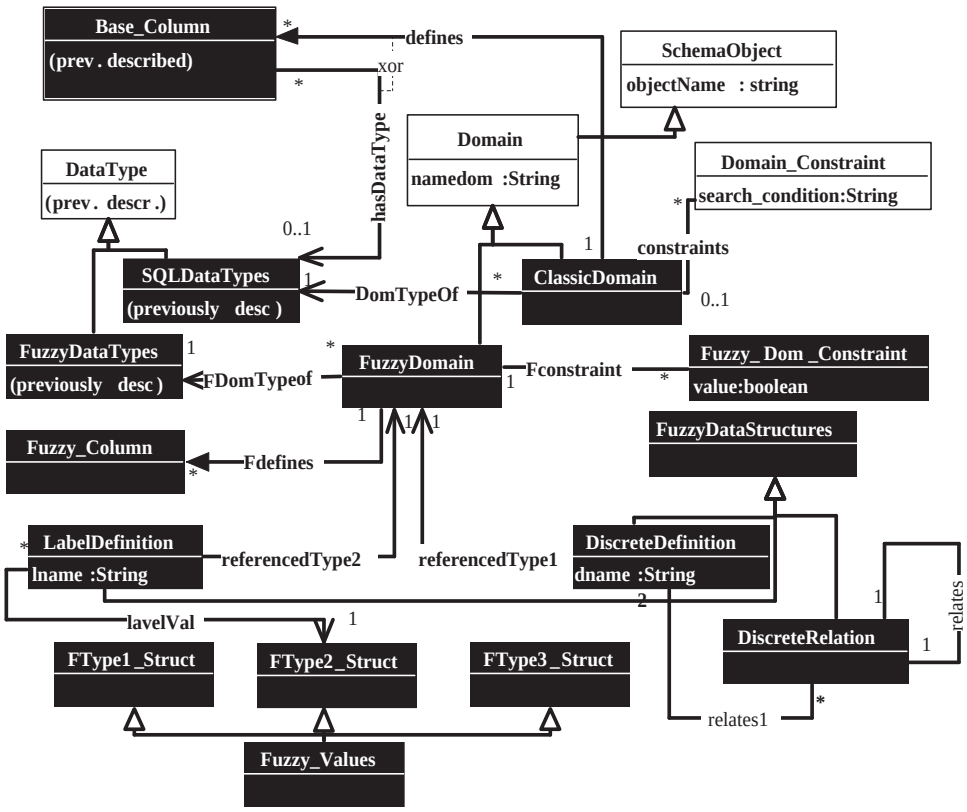


Figura 4.10: descripción de la clase *Dominios*

de los datos numéricos predefinidos (en el caso que el tipo de dato los permita). Estos valores suelen venir dados por etiquetas lingüísticas ligadas a distribuciones de posibilidad concretas. Por ejemplo: un atributo que describa la *altura* de una *persona*, tendrá representado mediante etiquetas lingüísticas los valores: *muy_bajo, bajo, mediano, alto, muy_alto*. Si bien se tratara de un tipo de dato cuyos valores son discretos, estos también deberán ser definidos previamente, por ejemplo, el *color de pelo* de una *persona*, vendrá definido por las etiquetas: *rubio, moreno, castaño y pelirrojo* y las relaciones entre dichas etiquetas. Y no solámente se utilizan dichas definiciones para insertar información en la BD sino también para consultarla.

Las clases que describen los dominios difusos están representadas en la figura 4.10 y se describen a continuación:

- *Domain*: se trata de una clase abstracta que agrupa los diferentes tipos de dominios que ahora existen en la ontología, el domino básico (definido por el ANSI SQL) y el dominio difuso. Consta de un atributo, que identifica el nombre del dominio del que se trata. Por último cabe destacar que *Domain* es subclase de *Schema_Object*, ya que es otro de los elementos que forman parte de un esquema de bases de datos.

- *Classic_Domain*: se trata del dominio clásico, y se corresponde con la clase *Domain* de la anterior ontología. Este dominio esta formado por la relación que establece su vínculo con las diferentes columnas que pueden usar dicha definición de dominio (*defines*), la relación que establece el tipo de dato por el que está formado el dominio (*domTypeOf*) que tiene que definirse explícitamente, y aquella que establece las restricciones de integridad de datos que puede definirse sobre el mismo, llamada *constraints* (estas restricciones se suelen corresponder con la sentencia *Check*).

- *Fuzzy_Domain*: sta clase representa aquellos dominios que involucran datos difusos en su definición. La única forma de definir un tipo de dato difuso y vincularlo a una columna difusa es a través de la relación *Fdefines*. El tipo de datos difuso asociado al dominio se establece mediante la relación *FDomTypeOf*. A su vez, los dominios difusos también tienen restricciones asociadas a los mismos, el número de restricciones asociadas a un dominio queda determinado por *Fconstraint*. Por último, asociadas con los dominios también se encuentran las definiciones de etiquetas lingüísticas y valores discretos, sin embargo estas relaciones se describirán en en subapartado correspondiente. En la tabla 4.2 se muestran las restricciones definidas sobre las propiedades relacionadas con dicha clase y su descripción mas detallada.

Tal y como se puede observar, la definición de los tipos de datos en los valores clásicos, puede hacerse bien a través de dominios o bien a través de un vínculo directo con el tipo de datos en cuestión. En datos difusos únicamente se podrá realizar a través de dominios, siendo entonces dicha clase la que contendrá las referencias a las columnas que están relacionadas con el mismo. Gracias a esta particularidad, las columnas difusas pueden compartir dominios y ahorrarse así definiciones repetitivas de los mismos.

atributo	Restricción	Valor	descripción
Fdefines	cardinalidad	multiple	Varias columnas pueden representarse por el mismo dominio difuso
FDomTypeOf	cardinalidad	1	A un dominio le corresponde sólo un tipo de datos difuso
Fconstraint	cardinalidad	múltiple	Sobre un dominio se pueden relacionar múltiples restricciones difusas

Tabla 4.2: Restricciones de los atributos de *Fuzzy_Domain*

4.3.3.3. Restricciones Difusas

Los tipos de datos difusos definidos en FIRST (véase sección B.1.2 del Anexo B para más detalle), también pueden incluir restricciones en su definición. Dichas restricciones consisten en la especificación de qué tipos de valores (estructuras de datos a usar) pueden o no contener los atributos definidos.

Para realizar esta especificación se han creado las siguientes clases (pueden verse en la figura 4.11):

- *Fuzzy_Dom_Constraint*: clase que agrupa todas las restricciones difusas. Es una clase abstracta, que tiene como atributo a *value*, que es de tipo booleano y especifica si la restricción está activada o no. La restricción de cardinalidad que existe sobre este valor esta establecida a 1.
- *Label_Constr*: significa que la definición que incluya esta restricción (si el atributo value esta a verdadero) no permitirá la inclusión de valores que sean etiquetas lingüísticas.
- *Crisp_Constr*: si el atributo value esta a verdadero, implicará que no se permiten valores Crisp (numéricos comunes) en el atributo que contenga dicha restricción.
- *Interval_Constr*: incluir esta restricción implica no permitir el uso de valores intervalares.
- *Trapezoid_Constr*: no se permitirán valores trapezoidales en aquellos atributos donde está restricción tenga un valor de verdadero.
- *Appr_Constr*: los valores aproximados a un número concreto no serán permitidos en aquellos atributos en los que se defina esta restricción.

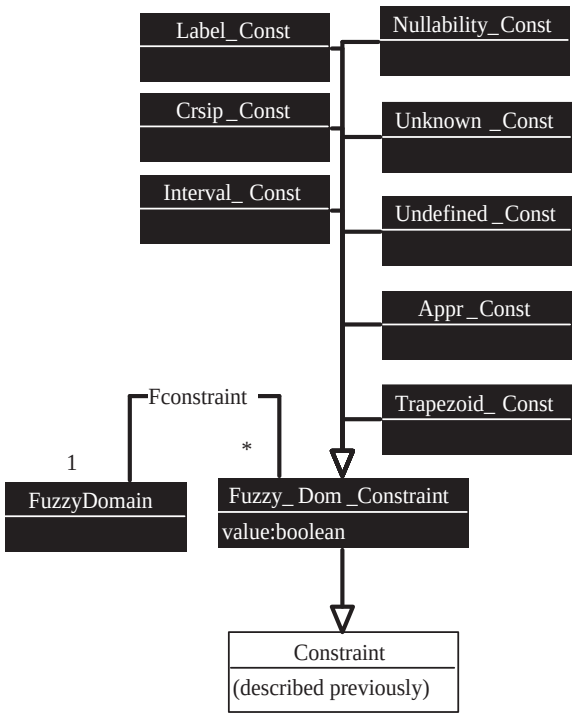


Figura 4.11: descripción de las Restricciones Difusas

- *Nullability_Constr*: indica que no se permiten valores nulos en este dominio.
- *Unknown_Constr*: indica que no se permiten valores desconocidos (*unknown*) en este dominio.
- *Undefined_Constr*: indica que no se permiten valores indefinidos (*undefined*) en este dominio.

Existen otras dos restricciones en la definición del modelo, son *ONLY LABEL* y *ONLY LABEL OR UNKNOWN*. Estas dos restricciones se forman generando explícitamente las restricciones necesarias para que se cumplan las anteriores, es decir generar una instancia de todas las restricciones a excepción de la de etiquetas lingüísticas, y/o de valores de *unknown*.

En la ontología la definición de las restricciones difusas se realiza a través de la definición de un dominio usando la relación *FConstraint* y no, tal y como se podría plantear a priori, sobre las columnas. A pesar de que utilizando el lenguaje FSQL [Gal99] las restricciones se vinculan a las columnas, esta decisión se ha tomado debido al hecho de que los tipos de datos difusos se vinculan a las columnas a través de dominios, con lo que resulta más lógico que sea este dominio el que albergue dichas restricciones.

4.3.3.4. Valores Discretos y Relaciones de Similitud

La representación de valores discretos, nos permite establecer etiquetas que no tienen significado basado en un referencial ordenado, sino que su significado vendrá dado por la relación que se establece entre las diferentes etiquetas asociadas al mismo dominio. Dicho significado será establecido mediante la asignación de un valor de similitud (establecido entre 0 y 1) para cada par de valores.

Para representar estas etiquetas y las relaciones, se han propuesto dos nuevas clases en la ontología (tal y como podemos ver en la figura 4.10):

- *Fuzzy_Data_Structures*: clase abstracta que engloba las estructuras difusas concretas relacionadas con etiquetas como las etiquetas lingüísticas y los valores discretos.
- *Discrete_Definition*: clase que permite representar las etiquetas que tienen un valor semántico asociado, pero no puede ser cuantificado

sin una representación explícita de su valor a través de relaciones de similitud con otras etiquetas. El valor de esta etiqueta viene representado mediante el atributo de esta clase denominado *dname*. La relación *referencedType1* permite establecer con qué dominios está relacionada la etiqueta que está definida.

- *Discrete_Relation*: esta clase representa como los valores discretos, definidos como instancias de la clase *Discrete_Definition*, son relacionados dos a dos a través un valor numérico (entre 0 y 1) establecido con el atributo *similarity*. El atributo *relates* es el que permitirá establecer que dos etiquetas discretas están relacionadas con un valor dado.

Las restricciones sobre las propiedades de objetos de estas clases se describen en la tabla 4.3.

Estas clases representan la base del tipo difuso 3 ya que las instancias de las mismas definen todos los valores que este tipo de dato puede representar.

Tabla 4.3: Restricciones de los atributos de *Discrete_Relation* y *Discrete_Definition*

Atributo	**Restricción**	**Valor**	**descripción**
ReferencedType1	cardinalidad	1	Una definición de discreto, sólo puede estar asociada con un dominio difuso
Relates	cardinalidad	2	Una relación difusa tiene dos definiciones de discretos relacionadas
dname	cardinalidad	1	Nombre del discreto

4.3.3.5. Etiquetas lingüísticas

En la figura 4.10 podemos ver cómo se ha representado la existencia de etiquetas lingüísticas que representan valores basados en distribuciones de posibilidad (basadas en un referencial ordenado).

Las clase *Label_Definition* es la que representa todas las etiquetas mediante su instanciación. Se les da un nombre a través del atributo *lname*, se vinculan a dominios difusos mediante la relación *referencedType2* y representan un valor mediante una referencia a una distribución de posibilidad definida a través de la relación *labelVal* (que conecta dicha etiqueta con la clase *Ftype2_Struct* que a continuación será descrita). Las restricciones sobre esta clase se describen en la tabla 4.4.

Por ultimo dicha clase es subclase de *Fuzzy_Data_Structures*, que es una clase abstracta que engloba las estructuras difusas definidas en FIRST (véase sección B.1.2 del Anexo para más detalle).

Atributo	**Restricción**	**Valor**	**descripción**
ReferencedType2	cardinalidad	1	Una definición de etiqueta, sólo puede estar asociada con un dominio difuso
labelVal	cardinalidad	1	Una etiqueta únicamente puede tener asociada una distr. posibilidad
lname	cardinalidad	1	Nombre de la etiqueta

Tabla 4.4: Restricciones de los atributos de *Label_Definition*

4.3.3.6. Representaciones de las Estructuras de los Tipos de Datos Difusos

Los valores que los tipos de datos difusos representan deben de ser definidos siguiendo una estructura concreta. Los tipos de datos difusos basados en un referencial ordenado podrían representar datos clásicos, para lo cual pueden usarse los tipos de datos base, o bien ser distribuciones de posibilidad, pero en este último caso las estructuras que permitirán almacenar dichos valores deben ser generadas explícitamente. Dichas estructuras están descritas con detalle en FIRST (más detalle en la sección B.1.2 del Anexo B).

La figura 4.12 muestra las estructuras de datos que los tipos de datos difusos pueden soportar para almacenar sus valores. Dichas estructuras están representadas por las siguientes clases:

- *FuzzyValues*: clase abstracta que agrupa aquellas estructuras de representación que se necesitan para definir datos que no son los predefinidos y representan datos difusos basados en un referencial ordenado.
- *FType1_Struct*: clase abstracta que agrupa aquellos estructuras que únicamente pueden usar los valores que han sido definidos como de *Tipo Difuso 1*.
- *Null*: clase que representa un valor nulo. Esta clase es subclase de *FType1_Struct, FType2_Struct o FType3_Struct*.
- *Unknown*: clase que representa un valor unknown o desconocido. Esta clase es subclase de *FType2_Struct o FType3_Struct*.

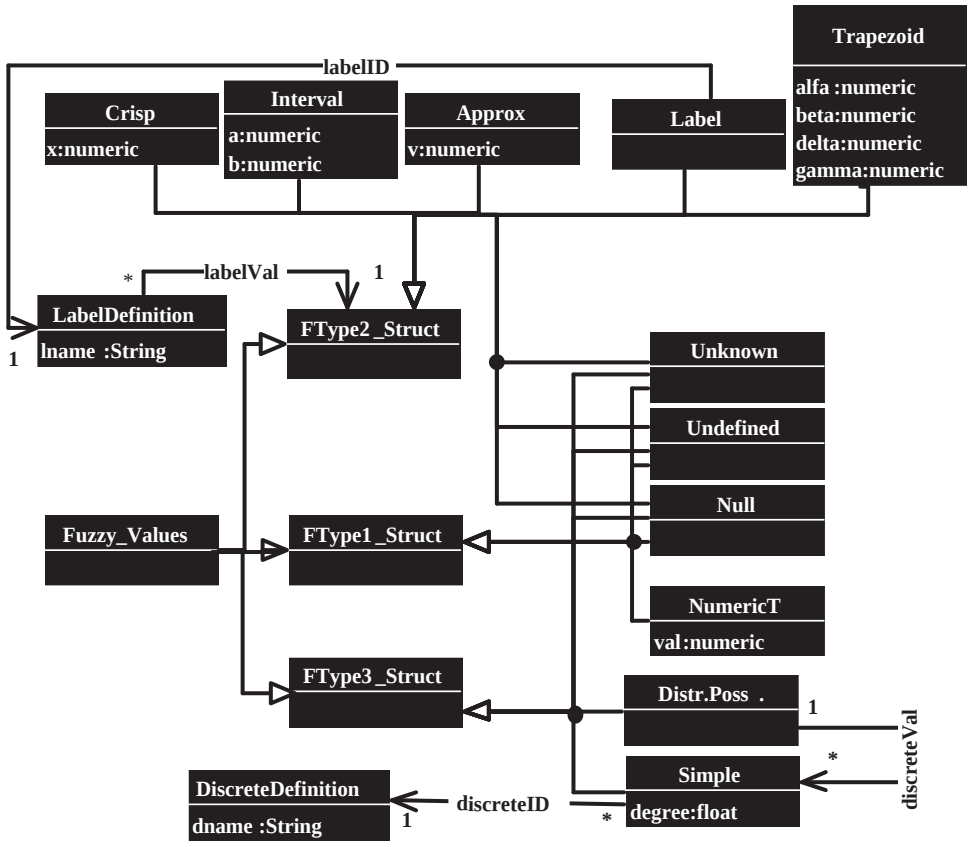


Figura 4.12: descripción de las estructuras para los TD Difusos

- *Undefined*: clase que representa un valor indefinido. Esta clase es subclase de *FType2_Struct o FType3_Struct.*
- *NumericT*: clase que representa un valor clásico numérico. Para ello esta clase cuenta con el atributo *val* que es de tipo numérico. En este atributo se almacena el valor numérico deseado mediante su instanciación.
- *FType2_Struct*: clase abstracta que agrupa aquellos estructuras que únicamente pueden usar los valores que han sido definidos como de *Tipo Difuso 2.*
- *Crisp*: clase que representa un valor clásico numérico. Para ello esta clase cuenta con el atributo x que es de tipo numérico. En este atributo se almacena el valor numérico deseado mediante su instanciación.
- *Approx*: clase que representa un valor aproximado a un número concreto. Tal y como se describe en FIRST, se trata de una distribución de posibilidad aproximada. Para ello esta clase cuenta con el atributo v que es de tipo numérico en el que se almacena el valor deseado mediante su instanciación.
- *Interval*: clase que representa un valor intervalar. Tal y como se describe en FIRST, se trata de una distribución de posibilidad intervalar. Para ello esta clase cuenta con los atributos a y b que definen los límites del intervalo que se usan.
- *Trapezoid*: clase que representa un valor trapezoidal. Tal y como se describe en FIRST, se trata de una distribución de posibilidad aproximada. Para ello esta clase cuenta con los atributos *alfa, beta, delta y gamma* que definen los limites del trapecio.
- *Label*: clase que representa a una etiqueta lingüística. Dicha clase hace referencia a la clase *Label_Definition* a través de la relación *labelID.*
- *FType3_Struct*: clase abstracta que agrupa aquellos estructuras que únicamente pueden usar los tipos que han sido definidos como de *Tipo Difuso 3.*
- *Simple*: clase que representa un valor discreto, con un cierto grado de certeza dado por el atributo *degree.* Dicho atributo sólo será usado cuando la clase *Simple* sea parte de una *DistrPoss* (clase descrita

a continuación). En el resto de los casos su valor carece de interés pues se interpreta como 1. El valor de este atributo en el primer caso hace referencia a la definición previa de valores discretos en la clase *discrete_Definition* usando el atributo *discreteID*.

- *DistrPoss*: clase que representa un conjunto de valores simples (instancias de la clase *Simple*) con un valor de certeza en cada uno de ellos (matizados por *degree* y descritos previamente), a través de la relación *discreteVal*.

En la tabla 4.5 se muestran las restricciones relacionadas con estas estructuras.

Atributo	**Restricción**	**Valor**	**descripción**
labelID	cardinalidad	1	La estructura de un valor label, tiene asociada una definición de etiqueta (instancia de *Label_Definition*)
discreteID	cardinalidad	1	La estructura de un valor *Simple*, tiene asociada una definición de valor discreto (instancia de *Discrete_Definition*)
discreteVal	cardinalidad	multiple	Una distr. posibilidad, tiene asociada uno o varios valores de instancias de valores *Simples*

Tabla 4.5: Restricciones de las estructuras de datos que representan valores difusos

4.3.3.7. Definición del Esquema. Metaclases.

Una vez planteada la *Ontología del Catálogo* (podemos verla en la figura 4.13), nos queda especificar cómo un esquema que representa datos difusos puede ser definido en la misma.

Metaclases

La *Ontología del Catálogo* describe la estructura de la información que puede ser representada en una BD relacional extendida con información imprecisa. Por tanto, al representar la estructura de la cualquier información trabajamos con metadatos.

En ontologías, los metadatos son aquellos que permiten especificar como la realidad está representada. En este caso, serían metadatos la definición de *clase, propiedad, restricción* y cualquier otro elemento que

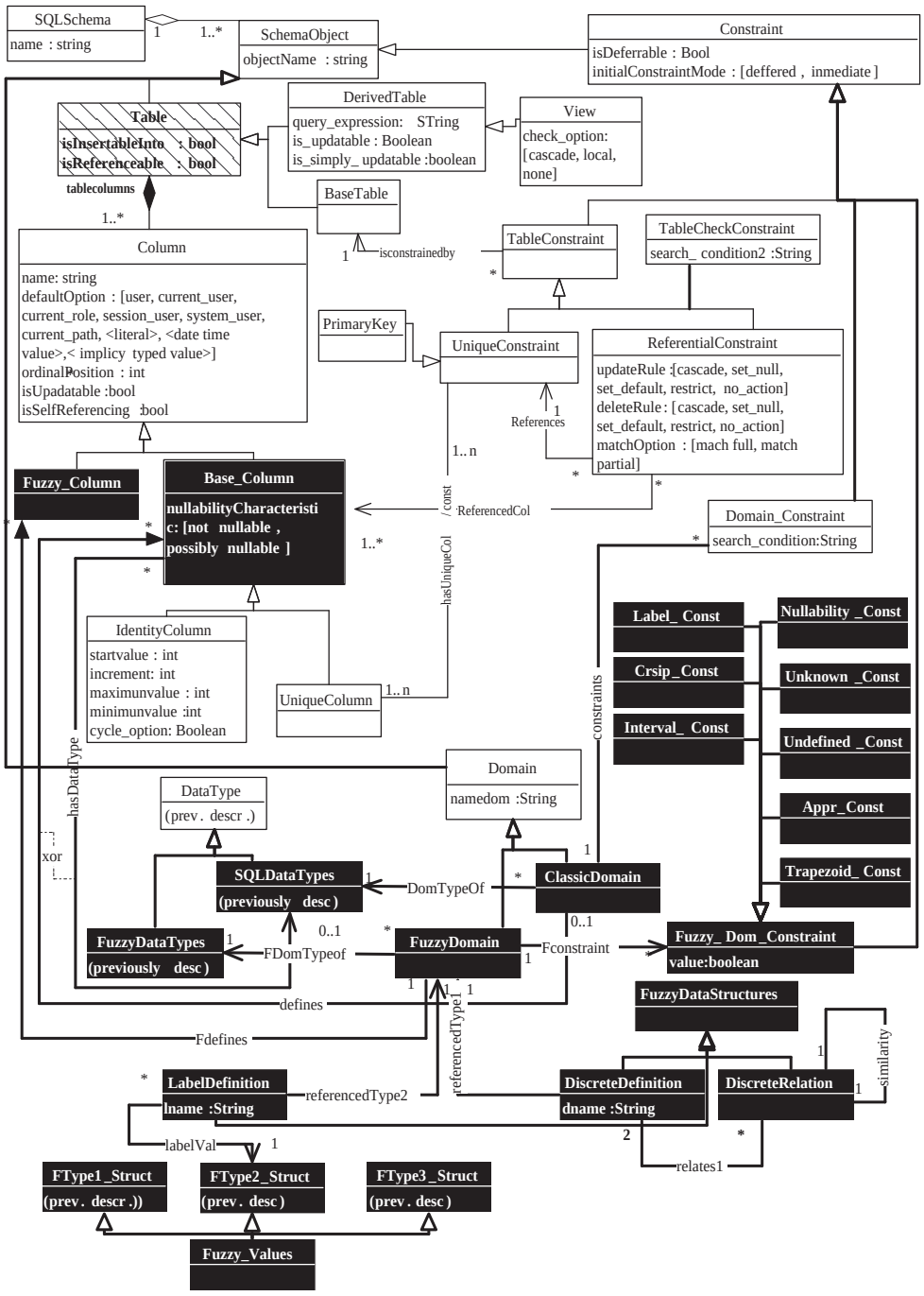


Figura 4.13: descripción de *Ontología del Catálogo*

nos permitiera representar algo en la misma (este es el caso de las *Ontologías Representacionales*).

En el caso de la *Ontología del Catálogo* del Modelo Relacional se cuenta con la definición básica de una *Relación o Tabla* como elemento fundamental para representar la información. Además de describir la estructura y particularidades de una relación esta deberá ser por sí misma un elemento que permita la inserción de datos. Por ejemplo y al contrario que ocurre en un SGBD del Modelo Relacional que al representar una relación o tabla en el catálogo automáticamente obtendríamos relaciones como *personas, departamentos, piezas, proyectos, etc.*, esto no resultaría posible hacerlo en una ontología. En ella ocurre que al representar estas mismas relaciones en el catálogo definido hasta ahora, se obtendrían únicamente las instancias de las clases de catálogo (de la clase *Tabla*, *Columna_Base*, *Columna_Difusa*, etc.). Por tanto, insertar datos en dichas tablas sería imposible dado que *no se puede instanciar una instancia*, que es lo que dichas tablas son en la *Ontología del Catálogo* tal y como está representada.

En la figura 4.13 vemos como la clase *Tablas* está sombreada para especificar que dicha clase es una *Metaclase*. Esta decisión se ha tomado para poder representar dichas relaciones (tablas), como lo que verdaderamente son, una representación de alto nivel de una forma de organizar los datos. Por consiguiente la definición de tabla (relación) deberá ser tratada como una metaclase y su instancia genera una nueva clase. Siguiendo con el ejemplo anterior, todas aquellas tablas que habíamos definido como instancias: *personas, departamentos, piezas, proyectos, etc.* serían ahora a su vez clases de la ontología, y por tanto podrían ser instanciadas, para contener datos. En el apartado 2.3.2 se introduce el concepto de cómo las metaclases, una vez que son instanciadas, generan a la vez instancias y clases, y cómo estas mismas clases pueden volver a generar una nueva ontología dado que representan una realidad diferente a la del catálogo. En este caso la realidad que representan dichas instancias será la del propio esquema que se esta representando, por ejemplo, el de la gestión de una biblioteca, la gestión de una BD multimedia, etc.

Importación

Cuando se desea representar una ontología cualquiera mediante un lenguaje concreto, como por ejemplo OWL, se ha de recurrir a las ontologías representacionales que definen los conceptos sobre los que se representa la información. En el caso de OWL se requieren las ontologías

descritas en la tabla 4.6 para poder utilizar la representación de datos que éste lenguaje propone.

En el caso de una BDD se requiere la presencia de la *Ontología del Catálogo* para su correcta definición. Sobre dicha ontología, se procederá a definir instancias que representan la BDD deseada. Sin embargo, la *Ontología del Catálogo* es una estructura estática, es decir, representa una realidad y como tal no debe ser modificada, simplemente debe ser utilizada para representar otros conceptos. De esta forma, y al igual que ocurre con las descripciones de la tabla 4.6, la *Ontología del Catálogo* no se utiliza directamente para generar el nuevo esquema de BDD, puesto que no debe permitirse su modificación. El proceso consistirá en generar una nueva ontología basada en OWL, donde se importa la *Ontología del Catálogo*, cuyos elementos estarán accesibles y serán instanciables, pero no podrán modificarse, y no formarán parte de la nueva definición. La nueva ontología estará entonces definida por un conjunto de instancias y clases (dado que existen metaclases en la ontología) que representan un esquema de BDD relacional.

Ontología	**URL**	**descripción**
xsd	http://www.w3.org/2001/XMLSchema#	Tipos de datos base y elementos XML
rdfs	http://www.w3.org/2000/01/rdf-schema#	Elementos del esquema de RDFS
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	Sintaxis básica de RDF
owl	http://www.w3.org/2002/07/owl#	Características propias del OWL

Tabla 4.6: Ontologías importadas en OWL

La *Ontología del Catálogo* se ha denominado *fdtscho*, y se encuentra disponible en *http://personales.ya.com/fkrowl/fdtscho/fdtscho.owl*.

También podemos ver el código descrito en el CD que se adjunta a este trabajo de tesis.

4.3.4. Ejemplos

A continuación se mostrará un breve ejemplo de como un esquema de BDD puede ser definido en la *Ontología del Catálogo* mediante la instanciación del mismo. Vamos a plantear el esquema de BDD que se

muestra en figura 4.14 y cuyo esquema lógico en FSQL[2] [Bla03b] se expone a continuación:

```
CATS (  CatID       INTEGER PK,
        CatName     STRING (20),
        Age         FTYPE2 (1,2) FLOAT (1) NOT UNKNOWN,
        Weigh       FTYPE1 (0.4,2.0) FLOAT (2),
        Character   FTYPE3 (3) NOT NULL,
        hasBreed    (BREED.BreedName) )
BREED(  BreedName   STRING (100) PK,
        CharacterB  FTYPE3 (3))
VISIT(  Date        Date   PK,
        Price       FLOAT (2),
        Cat         (CATS.CatID) PK)
TREATMENT (
        illness     STRING (200)
        kind        FTYPE3 (2)
        Date        (VISIT.Date)  PK,
        Cat         (VISIT.CatID) PK)
MEDICINE (
        MedicineName STRING (100) PK,
        dose FTYPE2  (0.5,3) FLOAT (2) )
PRESCRIBE (
        Medicine    (MEDICINE.MedicineName) PK,
        Date        (TREATMENT.Date)  PK,
        Cat         (TREATMENT.CatID) PK)

PERIODICAL_TREATMENT (
        Date        (TREATMENT.Date)  PK,
        Cat         (TREATMENT.CatID) PK,
        duration    INTEGER,
        period      INTEGER)

SPORADIC_TREATMENT (
        Date        (TREATMENT.Date)  PK,
        Cat         (TREATMENT.CatID) PK,
        rule        STRING (200) )
```

[2]Delante de cada una de las tablas descritas debería escribirse la sentencia CREATE TABLE que se ha omitido para clarificar la descripción

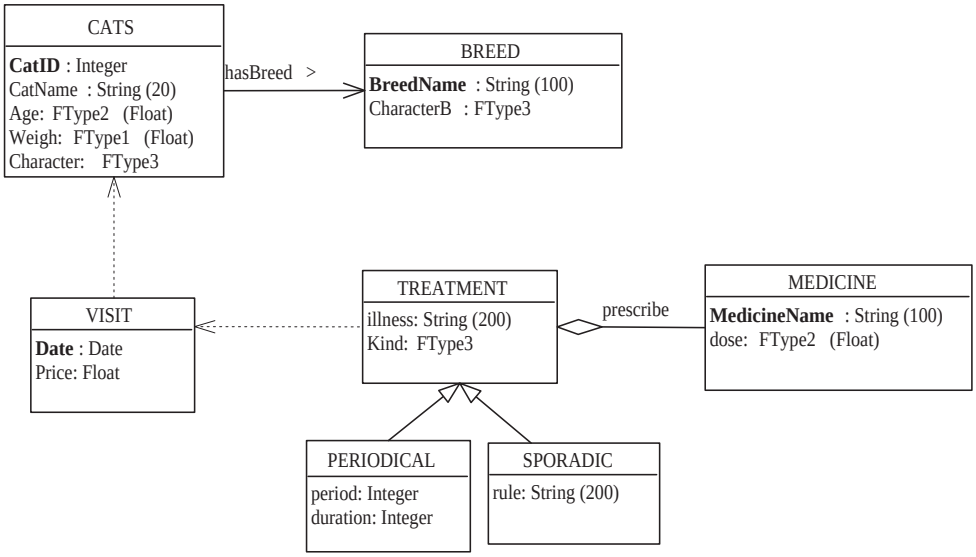


Figura 4.14: Ejemplo en UML de una BD de Clínica Veterinaria

Además de los datos comunes del esquema, quedan por definir en detalle de los dominios difusos que forman esta BDD, siendo:

- Etiquetas lingüísticas utilizadas en el dominio definido para el atributo *Age* de la tabla *Cat*. La descripción de dichas etiquetas, nombre y tipo de estructura utilizada y valor dado a las mismas se muestra en la tabla 4.7.

Nombre Etiqueta	Tipo de Valor	Valor
Cachorro(puppy)	Intervalar	0-1
Joven (young)	Trapezoidal	1-1,4-4-5
Adulto (adult)	Trapezoidal	2-3-6-10
Mediana-Edad (middle-aged)	Aproximado	5
Viejo (old)	Trapezoidal	7-8-13-15

Tabla 4.7: descripción de los valores de las etiquetas lingüísticas relacionadas con el dominio del atributo *Age* de la tabla *Cat*

4.3.4.1. Ejemplo 1: Clínica Veterinaria

- Valores Discretos relacionados con el atributo *Character* de la tabla Cat y su relaciones de similitud. Los valores discretos definidos son: *Agresivo (Aggresive), Tranquilo (Calm), Inquieto (eager), Indiferente (Indiferent), Cariñoso (Loving*). La relación de similitud establecida entre estos valores se describe en la tabla 4.8.

	Indiferente	Tranquilo	Cariñoso	Inquieto	Agresivo
Indiferente	1	0.8	0 .3	0.4	0.1
Tranquilo		1	0.5	0	0
Cariñoso			1	0.2	0.1
Inquieto				1	0.5
Agresivo					1

Tabla 4.8: Relaciones de Similitud del Atributo *Character*

Una vez descritos los valores del esquema de bases de datos difusas que se desea definir, se procede a la instanciación del mismo. Concretamente se presentan en este ejemplo las instancias generadas para la definición completa de las tablas *Cat* y *Breed* en la tabla 4.9, ya que el resto de las tablas tienen una definición similar, y por motivos de claridad es preferible su exclusión del documento.

Cada una de estas instancias queda definida por los valores que adquieren las propiedades de las clases instanciadas. En la tabla 4.10 se listan las propiedades de objeto más representativas de las instancias de *Cats* y *Breed* descritas anteriormente en la tabla 4.9. En la tabla 4.11 se describen las propiedades de tipo de datos básicos, que definen los valores finales del esquema.

Por último hay que destacar, que todas aquellas instancias de la clase *Table* (o como subclase *Base_Table*) se convierten a su vez en clases también. Así pues tendríamos generadas en el ejemplo descrito una clase: *Cats, Visit, Breed, Medicine, Treatment, Sporadic_Treatment, Periodical_Treatment y Prescribe.*

4.3.4.2. Ejemplo 2: BD Suelos

En este ejemplo se propone la utilización de una BD Difusa Real, como la de *Suelos* descrita en el Anexo C y cuyo esquema se encuentra definido en el apartado C.1.2. Dicha definición supone la instanciación de la *Ontología del Catálogo*, concretamente de todas aquellas clases que permitan la definición

Tabla 4.9: Instanciación de la *Ontología del Catálogo* del ejemplo de la Clínica Veterinaria

rdf:ID	**Instancia de**	**rdfs:comment**
Cats	BaseTable	Representa Información de los gatos
Breed	BaseTable	Especies de Gatos
CatID	UniqueColumn	Identificador de gatos
CatName	BaseColumn	Nombre de gatos
Age	FuzzyColumn	Edad de gatos (años)
Weigh	FuzzyColumn	Peso de gatos (gramos)
Character	FuzzyColumn	Cáracter de gatos(cariñoso,amigable,arisco)
CatBreed	BaseColumn	Nombre de gatos
BreedName	UniqueColumn	Nombre de la especie
CharacterB	FuzzyColumn	Carácter de la especie
FDom_Age	FuzzyDomain	Dominio para tipo dato difuso Age
FDom_Character	FuzzyDomain	Dominio para Character
FDom_Weight	FuzzyDomain	Dominio difuso relacionado con Weight
DTCatID	Integer	Tipo de dato de *CatID*
DTCatName	Varying	Tipo de dato de *CatName*
DomAge_FDT	FType2	Tipo de dato difuso de *FDom_Age*
DomAge_DT	Float	Tipo de dato de *FDom_Age*
DomWeigh_FDT	FType1	Tipo de dato difuso de *FDom_Weigh*
DomWeigh_DT	Float	Tipo de dato de *FDom_Weigh*
DomCharacter_FDT	FType3	Tipo de dato difuso de *FDom_Character*
DTBreedName	Varying	Tipo de dato de *BreedName*
PrimaryKeyCat	PrimaryKey	Restricción de Clave Primaria para Cat
PrimaryKeyBreed	PrimaryKey	Restricción de Clave Primaria para Breed
FK_Cat_Breed	Referential-Constraint	Restricción de clave ajena para atributo Breed de la relacion Cats
LD_MiddleAgedCat	Label_Definition	Etiqueta del dominio *FDom_Age*
DT_LV_-MiddleAgedCat	Approx	Valor asignado a la etiqueta *LD_MiddleAgedCat*
LD_PupyCat	Label_Definition	Etiqueta del dominio *FDom_Age*
DT_LV_PuPyCat	Intervalar	Valor asignado a la etiqueta *LD_PupyCat*
LD_AdultCat[3]	Label_Definition	Etiqueta del dominio *FDom_Age*
DT_LV_AdultCat[4]	Trapezoid	Valor asignado a la etiqueta *LD_AdultCat*
agressiveCat	Discrete_Definition	Valor discreto relacionada con el dominio *FDom_CatCharacter*
calmCat[5]	Discrete_Definition	Valor discreto relacionado con el dominio *FDom_CatCharacter*
discreterelations_18[6]	Discrete_Relation	Relación entre *calmCat* e *indiferentCat*
NullabilityConst_15	NullabilityConst	Restricción de Valor atrib. *Character*
UnknownConst_18	UnknownConst	Restr. valor Desconocido en atrib. *Age*

Tabla 4.10: Propiedades de Objeto en la *Ontología del Catálogo* del ejemplo de la Clínica Veterinaria

De	**Atributo**	rdf:resource
Cats	*tableColumns*	CatName, CatID, Age, Weigh, Character,BreedName
Breed	*tableColumns*	BreedName, CharacterB
CatID	*hasDataType*	DTCatID
CatID	*hasUniqueConst*	PrimaryKeyCats
CatName	*hasDataType*	DTCatName
CatBreed	*hasDataType*	DTBreedName
BreedName	*hasDataType*	DTBreedName
FDom_CatAge	*FDomTypeOf*	FDTCatAgeDom
FDom_CatAge	*FDefines*	Age
FDom_CatAge	*FConstraints2*	UnknownConst_18
FDTCatAgeDom	*hasNumericType*	DTCatAgeDom
FDom_CatWeigh	*FDomTypeOf*	FDTCatWeighDom
FDom_CatWeigh	*FDefines*	Weigh
FDTCatWeighDom	*hasNumericType*	DTCatWeightDom
FDom_CatCharacter	*FDomTypeOf*	FDTCatCharacterDom
FDom_CatCharacter	*FDefines*	Character, CharacterB
FDom_CatCharacter	*FConstraints2*	NullabilityConst_15
PrimaryKeyCats	*isConstrainedBy*	Cats
PrimaryKeyCats	*hasUniqueCol*	CatID
PrimaryKeyBreed	*isConstrainedBy*	Breed
PrimaryKeyBreed	*hasUniqueCol*	BreedName
FK_Cats_Breed	*referencedCol*	CatBreed
FK_Cats_Breed	*references*	PrimaryKeyBreed
LD_adultCat	*labelVal*	DT_LV_AdultCat
LD_adultCat	*referencedType*	FDom_CatAge
LD_adultCat	*labelVal*	Approx_10
LD_adultCat[7]	*referencedType*	FDom_CatAge
agressiveCat[8]	*referencedType3*	FDom_CatCharacter
Discrete_Relations_18[9]	*relates*	CalmCat, IndiferentCat

Tabla 4.11: Propiedades de Tipo de datos en la *Ontología del Catálogo* del ejemplo de la Clínica Veterinaria

De	**Atributo**	**rdf:datatype**	**value**
Cats	*ObjectName*	xsd:String	Cats
Cats	*isReferenceable*	xsd:bool	true
Breed	*ObjectName*	xsd:String	Breed
CatID [10]	*nameCol*	xsd:String	ID
FDomCatAge[11]	*ObjectName*	xsd:String	FDom_CatAge
FDTCatName	*lenghStr*	xsd:int	20
FDTAgeDom	*margen*	xsd:float	1
FDTAgeDom	*much*	xsd:int	2
DTAgeDom	*precision*	xsd:int	1
FDTWeighDom	*margen*	xsd:float	0.4
FDTWeighDom	*much*	xsd:int	2
DTWeighDom	*precision*	xsd:int	2
DTCharacterDom	*len*	xsd:int	3
DTBreedName	*lenghStr*	xsd:int	100
DT_LV_AdultCat	*alfa*	xsd:float	2
DT_LV_AdultCat	*beta*	xsd:float	3
DT_LV_AdultCat	*delta*	xsd:float	6
DT_LV_AdultCat	*gamma*	xsd:float	10
Approx_10[12]	*v*	xsd:float	5
Discrete_Relations_18[13]	*similarity*	xsd:float	0.8

del esquema que describe dicha *BDD de Suelos*. A continuación se refleja qué clases han de ser instanciadas, de forma breve, y se ejemplifica dicha descripción mediante la exposición de parte de las instancias que se han generado (dado que es inviable mostrar todas las instancias debido a su gran número). La definición completa de la *BDD de Suelos* en la *Ontología del Catálogo* se adjunta en un CD a este trabajo de tesis.

- Se instanciarán las clases relativas a la definición de relaciones (Base_Table) y atributos (BaseColumn, FuzzyColumn, UniqueColumn). En este ejemplo se expone la instanciación de la relación *Localización* y los atributos: *Latitud*, *Longitud*, *fisiografia*, *tmedia*, *codigo_es*. También se definirá la relación *Estructura* y los atributos: *codigo_es* y *grado_es*.
- Clases relativas a las restricciones de Clave primaria (*PrimaryKey*). Al definir las tablas se definen sus claves primarias, en el ejemplo que estamos poniendo, *Latitud y Longitud* en *Localización* y *codigo_es* en *Estructura*.
- Clases relativas a las restricciones de clave ajena (instancias de *Referential_Constraint*). En el ejemplo que nos ocupa, se asociaría *codigo_es* de *Localización* con la clave primaria de la tabla estructura, esto es, *codigo_es* de *Estructura*.
- Clases relativas a la definición de etiquetas lingüísticas relacionadas con los Tipos Difusos 2.
- Clases relativas a la definición de los valores discretos relacionados con los Tipos Difusos 3, y las relaciones de similitud que definen a cada uno de estos valores discretos.
- Clases relativas a los dominios difusos.
- Clases relativas a la definición de restricciones difusas.

Tabla 4.12: Instancias de la *Ontología del Catálogo* del ejemplo de la BDD Suelos

rdf:ID	**Instancia de**	**rdfs:comment**
Localizacion	BaseTable	Definición de la relación *Localización*
Estructura	BaseTable	Definición de la relación *Estructura*
Latitud	Unique-Column	Identificador de Localización, coordenadas de latitud
Longitud	Unique-Column	Identificador de Localización, coordenadas de longitud
fisiongrafia	FuzzyColumn	Fisiografía del suelo en esta localización
tmedia	FuzzyColumn	Media de la temperatura
codigo_es_loc	BaseColumn	Código de la estructura de suelos que hay en esta localización
codigo_es	Unique-Column	Identificador numérico de la estructura
grado_es	FuzzyColumn	Tipo de estructura
FDom_tmedia	FuzzyDomain	Dominio para tipo dato difuso *tmedia*
FDom_fisiografia	FuzzyDomain	Dominio para *fisiografía*
FDom_grado_es	FuzzyDomain	Dominio difuso relacionado con *grado_es*
DTLatitud	Numeric	Tipo de dato de *Latitud*
DTLongitud	Numeric	Tipo de dato de *Longitud*
FDT2_dom_tmedia	FType2	Tipo de dato difuso de *tmedia*
DT_dom_FDT2	Float	Tipo de dato asociado a los FDT2 puesto que la mayoría son Float(2)
FDT3_dom_fisiografia	FType3	Tipo de dato difuso de *fisiografía*
DTcodigo_es	Numeric	Tipo de dato de *codigo_es*
FDT3_dom_grado_es	FType3	Tipo de dato difuso de *grado_es*
PKLocalizacion	PrimaryKey	Restricción de Clave Primaria para *Localización*
PKEstructura	PrimaryKey	Restricción de Clave Primaria para *Estructura*
FK_codigo_es_localizacion	Referential-Constraint	Restricción de clave ajena para atributo *codigo_es_loc* de la relacion *Localización* hacia la relacion *Estructura*
LD_baja_tmedia	Label_Definition	Etiqueta relacionada con el dominio *FDom_tmedia*. Significa temperatura media baja.
T_baja_tmedia	Trapezoid	Valor asignado a la etiqueta *LD_baja_tmedia*
LD_media_tmedia	Label_Definition	Etiqueta relacionada con el dominio *FDom_tmedia*. Significa temperatura media media.

Tabla 4.12: Instancias de la *Ontología del Catálogo* del ejemplo de la BDD Suelos

rdf:ID	**Instancia de**	**rdfs:comment**
T_media_tmedia	Trapezoid	Valor asignado a la etiqueta *LD_media_tmedia*
LD_alta_tmedia	Label_Definition	Etiqueta relacionada con el dominio *FDom_tmedia*. Significa temperatura media alta.
T_alta_tmedia	Trapezoid	Valor asignado a la etiqueta *LD_alta_tmedia*
DD_llano-_fisiograf	Discrete-_Definition	Discreto del dominio *FDom_fisiografia* con valor *Llano*
DD_fladera-_fisiograf	Discrete-_Definition	Discreto del dominio *FDom_fisiografia* con valor *FondoLadera*
DD_ladera-_fisiograf	Discrete-_Definition	Discreto del dominio *FDom_fisiografia* con valor *Ladera*
DD_cima-_fisiograf	Discrete-_Definition	Discreto del dominio *FDom_fisiografia* con valor *Cima*
DD_meseta-_fisiograf	Discrete-_Definition	Discreto del dominio *FDom_fisiografia* con valor *Meseta*
DD_VWeak-_fisiograf	Discrete-_Definition	Discreto del dominio *FDom_grado_es* con valor *VeryWeak*
DD_Weak-_fisiograf	Discrete-_Definition	Discreto del dominio *FDom_grado_es* con valor *Weak*
DD_Moderate-_fisiograf	Discrete-_Definition	Discreto del dominio *FDom_grado_es* con valor *Moderate*
DD_Strong-_fisiograf	Discrete-_Definition	Discreto del dominio *FDom_grado_es* con valor *Strong*
discrete-_relations_35	Discrete-_Relation	Relación entre *Llano* y *FondoLadera*
discrete-_relations_36	Discrete-_Relation	Relación entre *Llano* y *Ladera*
discrete-_relations_37	Discrete-_Relation	Relación entre *Llano* y *Cima*
discrete-_relations_38	Discrete-_Relation	Relación entre *Llano* y *Meseta*
discrete-_relations_39	Discrete-_Relation	Relación entre *FLad* y *Ladera*
discrete-_relations_40	Discrete-_Relation	Relación entre *FLad* y *Cima*
discrete-_relations_41	Discrete-_Relation	Relación entre *FLad* y *Meseta*
discrete-_relations_42	Discrete-_Relation	Relación entre *Ladera* y *Cima*
discrete-_relations_43	Discrete-_Relation	Relación entre *Ladera* y *Meseta*
discrete-_relations_44	Discrete-_Relation	Relación entre *Cima* y *Meseta*

Tabla 4.12: Instancias de la *Ontología del Catálogo* del ejemplo de la BDD Suelos

rdf:ID	**Instancia de**	**rdfs:comment**
discrete-_relations_76	Discrete-_Relation	Relación entre *Very Weak* y *Weak*
discrete-_relations_77	Discrete-_Relation	Relación entre *Very Weak* y *Moderate*
discrete-_relations_78	Discrete-_Relation	Relación entre *Very Weak* y *Strong*
discrete-_relations_79	Discrete-_Relation	Relación entre *Weak* y *Moderate*
discrete-_relations_80	Discrete-_Relation	Relación entre *Weak* y *Strong*
discrete-_relations_81	Discrete-_Relation	Relación entre *Moderate* y *Strong*
UndefinedConst-_138	Undefined-Const	Restr. de valor indefinido para *FDom_tmedia*
UnknownConst-_137	Unknown-Const	Restr. valor desconocido para *FDom_tmedia*
UnknownConst-_52	Unknown-Const	Restr. valor desconocido para *FDom_fisiograf*
UndefinedConst-_51	Undefined-Const	Restr. de valor indefinido para *FDom_fisiograf*
NullabilityConst-_49	Nullabi-lityConst	Restr. de valor nulo para *FDom_fisiograf*
TrapezoidConst-_50	Trapezoid-Const	Restr. de valor trapezoidal para *FDom_fisiograf*
IntervalConst-_48	IntervalConst	Restr. de valor intervalar para *FDom_fisiograf*
CrispConst_47	CrispConst	Restr. de valor crisp para *FDom_fisiograf*
ApproxConst_46	ApproxConst	Restr. de valor approximado para *FDom_fisiograf*

Tabla 4.13: Propiedades de Objeto en la *Ontología del Catálogo* del ejemplo de la BDD de Suelos

De	**Atributo**	rdf:resource
Localizacion	*tableColumns*	Latitud, Longitud, fisiografia, tmedia, codigo_es
Latitud	*hasUniqueConst*	PKLocalizacion
Latitud	*hasDataType*	DTLatitud
Longitud	*hasUniqueConst*	PKLocalizacion
Longitud	*hasDataType*	DTLatitud
Codigo_es_loc	*hasDataType*	DTCodigo_es
FDom_fisiograf	*FDomTypeOf*	PKEstructura
FDom_fisiograf	*FDefines*	FDT3_Dom_Fisiografia

Tabla 4.13: Propiedades de Objeto en la *Ontología del Catálogo* del ejemplo de la BDD de Suelos

De	**Atributo**	rdf:resource
FDom_fisiograf	*FConstraints2*	ApproxConst_46, CrispConst_47, IntervalConst_48, Trapezoid-Const_50, NullabilityConst_49, UndefindedConst_51, Unknown-Const_52
ApproxConst_46	*FConstraints*	FDom_fisiograf
CrispConst_47	*FConstraints*	FDom_fisiograf
IntervalConst_48	*FConstraints*	FDom_fisiograf
NullabilityConst_49	*FConstraints*	FDom_fisiograf
TrapezoidConst_50	*FConstraints*	FDom_fisiograf
UnknownConst_51	*FConstraints*	FDom_fisiograf
UndefinedConst_52	*FConstraints*	FDom_fisiograf
FDom_tmedia	*FDomTypeOf*	
FDom_tmedia	*FDefines*	tmedia
FDom_tmedia	*FConstraints2*	UndefindedConst_138, Unknown-Const_139
FDT2_Dom_tmedia	*hasNumericType*	DT_Dom_FDT2
PKLocalizacion	*isConstrainedBy*	Localizacion
PKLocalizacion	*ishasUniqueCol*	Latitud,Longitud
FK_codigo_e-_localizacion	*referencedCol*	codigo_es_loc
FK_codigo_e-_localizacion	*references*	PKEstructura
FK_codigo_e-_localizacion	*isConstrainedBy*	Localizacion
Estructura	*tableColumns*	codigo_es, grado_es
Codigo_es	*hasUniqueConst*	PKEstructura
FDom_grado_es	*FDomTypeOf*	FDT3_grado_es
FDom_grado_es	*FDefines*	grado_es
PKEstructura	*isConstrainedBy*	Estructura
PKEstructura	*ishasUniqueCol*	codigo_es
LD_baja_tmedia	*labelVal*	T_Baja_tmedia
LD_baja_tmedia	*referencedType*	FDom_tmedia
LD_media_tmedia	*labelVal*	T_media_tmedia
LD_media_tmedia	*referencedType*	FDom_tmedia
LD_alta_tmedia	*labelVal*	T_alta_tmedia
LD_alta_tmedia	*referencedType*	FDom_tmedia
DD_llano_fisiograf	*referencedType*	FDom_fisiograf
DD_FLad_fisiograf	*referencedType*	FDom_fisiograf
DD_ladera_fisiograf	*referencedType*	FDom_fisiograf
DD_cima_fisiograf	*referencedType*	FDom_fisiograf
DD_meseta_fisiograf	*referencedType*	FDom_fisiograf
DD_VWeak_fisiograf	*referencedType*	FDom_grado_es
DD_Weak_fisiograf	*referencedType*	FDom_grado_es

Tabla 4.13: Propiedades de Objeto en la *Ontología del Catálogo* del ejemplo de la BDD de Suelos

De	**Atributo**	rdf:resource
DD_Mederate-_fisiograf	*referencedType*	FDom_grado_es
DD_Strong_fisiograf	*referencedType*	FDom_grado_es
Discrete_Relations_35	*relates1*	DD_llano_fisiograf
Discrete_Relations_35	*relates2*	DD_FLad_fisiograf
Discrete_Relations_36	*relates1*	DD_llano_fisiograf
Discrete_Relations_36	*relates2*	DD_ladera_fisiograf
Discrete_Relations_37	*relates1*	DD_llano_fisiograf
Discrete_Relations_37	*relates2*	DD_cima_fisiograf
Discrete_Relations_38	*relates1*	DD_llano_fisiograf
Discrete_Relations_38	*relates2*	DD_meseta_fisiograf
Discrete_Relations_..	*relates1*	...
Discrete_Relations_..	*relates2*	...

Tabla 4.14: Propiedades de Tipo de datos en la *Ontología del Catálogo* del ejemplo de la BDD de Suelos

De	**Atributo**	**rdf:datatype**	**value**
Localizacion	*ObjectName*	xsd:String	Localizacion
Estructura	*ObjectName*	xsd:String	Estructura
latitud	*NameCol*	xsd:String	latitud
longitud	*NameCol*	xsd:String	longitud
tmedia	*NameCol*	xsd:String	tmedia
fisiografia	*NameCol*	xsd:String	fisiografia
codigo_es_loc	*NameCol*	xsd:String	codigoEs
codigo_es	*NameCol*	xsd:String	codigoEsLoc
grado_es	*NameCol*	xsd:String	gradoEs
FDT2_dom_tmedia	*margin*	xsd:float	4.0
FDT2_dom_tmedia	*much*	xsd:float	10.0
DT_dom_FDT2	*precision*	xsd:float	2.0
FDT3_dom-_fisiografia	*len*	xsd:int	1
FDT3_dom-_grado_es	*len*	xsd:int	1
PKLocalizacion	*ObjectName*	xsd:String	PKLocalizacion
PKEstructura	*ObjectName*	xsd:String	PKEstructura
FK_codigo_es-_localizacion	*ObjectName*	xsd:String	FK_codigo_es-_localizacion
LD_baja_tmedia	*lname*	xsd:String	baja
T_baja_tmedia	*alfa*	xsd:float	0
T_baja_tmedia	*beta*	xsd:float	0
T_baja_tmedia	*delta*	xsd:float	6.5
T_baja_tmedia	*gamma*	xsd:float	8.5

Tabla 4.14: Propiedades de Tipo de datos en la *Ontología del Catálogo* del ejemplo de la BDD de Suelos

De	**Atributo**	**rdf:datatype**	**value**
LD_media_tmedia	*lname*	xsd:String	media
T_media_tmedia	*alfa*	xsd:float	8.5
T_media_tmedia	*beta*	xsd:float	10.5
T_media_tmedia	*delta*	xsd:float	12.5
T_media_tmedia	*gamma*	xsd:float	14.5
LD_alta_tmedia	*lname*	xsd:String	baja
T_alta_tmedia	*alfa*	xsd:float	14.5
T_alta_tmedia	*beta*	xsd:float	16.5
T_alta_tmedia	*delta*	xsd:float	21
T_alta_tmedia	*gamma*	xsd:float	21
DD_llano_fisiograf	*dname*	xsd:String	llano
DD_fladera-_fisiograf	*dname*	xsd:String	FondoLadera
DD_ladera-_fisiograf	*dname*	xsd:String	Ladera
DD_cima_fisiograf	*dname*	xsd:String	Cima
DD_meseta-_fisiograf	*dname*	xsd:String	Meseta
DD_VWeak-_fisiograf	*dname*	xsd:String	VeryWeak
DD_Weak_fisiograf	*dname*	xsd:String	Weak
DD_Moderate-_fisiograf	*dname*	xsd:String	Moderate
DD_Strong-_fisiograf	*dname*	xsd:String	Strong
discrete-_relations_35	*similarity*	xsd:float	0.5
discrete-_relations_36	*similarity*	xsd:float	0.2
discrete-_relations_37	*similarity*	xsd:float	0.2
discrete-_relations_38	*similarity*	xsd:float	0.2
discrete-_relations_39	*similarity*	xsd:float	0.2
discrete-_relations_40	*similarity*	xsd:float	0.2
discrete-_relations_41	*similarity*	xsd:float	0.2
discrete-_relations_42	*similarity*	xsd:float	0.2
discrete-_relations_43	*similarity*	xsd:float	0.2

Tabla 4.14: Propiedades de Tipo de datos en la *Ontología del Catálogo* del ejemplo de la BDD de Suelos

De	Atributo	rdf:datatype	value
discrete-_relations_44	*similarity*	xsd:float	0.5
discrete-_relations_76	*similarity*	xsd:float	0.3
discrete-_relations_77	*similarity*	xsd:float	0.3
discrete-_relations_78	*similarity*	xsd:float	0.3
discrete-_relations_79	*similarity*	xsd:float	0.3
discrete-_relations_80	*similarity*	xsd:float	0.3
discrete-_relations_81	*similarity*	xsd:float	0.3
UndefinedConst-_138	*value*	xsd:bool	true
UnknownConst-_137	*value*	xsd:bool	true
UnknownConst_52	*value*	xsd:bool	true
UndefinedConst_51	*value*	xsd:bool	true
NullabilityConst-_49	*value*	xsd:bool	true
TrapezoidConst_50	*value*	xsd:bool	true
IntervalConst_48	*value*	xsd:bool	true
CrispConst_47	*value*	xsd:bool	true
ApproxConst_46	*value*	xsd:bool	true

Como vemos, este ejemplo permite representar un caso real de BDD que se utiliza en la actualidad. En este ejemplo predominan los *Tipos Difusos 2*, con etiquetas lingüísticas asociadas a su dominio y definidas mediante representaciones trapezoidales y los Tipos Difusos 3, con valores que utilizan una única etiqueta para describir su valor (len=1). Este tipo de representación es el más utilizado dado que es el que los usuarios encuentran más sencillo para describir la realidad.

4.4. Sub-Ontología del Esquema de Datos Difusos

4.4.1. Justificación de la Sub-Ontología

Tal y como se describió anteriormente, la *Ontología del Catálogo* lo único que permite es describir los esquemas completamente a modo de instancias, al igual que actúa el diccionario de datos en los SGBDs. Sin embargo, no

disponemos de dichos esquemas definidos en forma de clases, como una ontología única que representa la realidad.

Al definir metaclases en la *Ontología del Catálogo* estamos esbozando una segunda ontología, que pretende representar dicho esquema descrito. Sin embargo el generar estas clases simplemente no implica obtener la ontología necesaria que permite describir la realidad que el esquema está representando según el Modelo Relacional. Y por tanto, tampoco permite que dicha ontología sea instanciada para poder definir la información que exista acerca de los datos representados por ese esquema (en el entorno de las BDD, se correspondería con las tuplas).

Nos vemos en la necesidad, por tanto, de definir una nueva ontología a partir del esquema descrito mediante la instanciación de la *Ontología del Catálogo*. A esta ontología la denominaremos a partir de ahora, *Ontología del Esquema* en lugar de *Sub-Ontología del Esquema* de Datos Difusos, por razones de claridad.

Sin embargo, cabe plantearse la funcionalidad de esta *Ontología del Esquema* que por un lado, representa la realidad de un dominio particular siguiendo los criterios de representación del Modelo Relacional y por otro, permite insertar valores en la misma a modo de instancias, tal y como ocurre en el Modelo Relacional con las tuplas.

En cuanto a la representación del esquema como ontología aporta utilidad en tanto que su publicación en Web, permitirá el acceso a la estructura de información por parte de usuarios autorizados, o bien ayudará a la compartición de esquemas de muy diversa índole (como se planteó en el apartado 2.2.1).

La incorporación de datos en la ontología como instancias (al igual que tuplas en el Modelo Relacional) proporciona una gran utilidad en tanto que facilita al usuario la definición de la información, puesto que lo aisla de las particularidades del sistema de almacenamiento o del lenguaje utilizado para su definición. En cuanto al hecho de usar una ontología como medio de gestión de grandes cantidades de información no se considera el medio más adecuado para hacerlo al contrario de lo que ocurre con un SGBD, puesto que éste último hace una gestión mas eficiente de la información que alberga.

Otra aportación de esta propuesta es la generación de consultas (formulación de las mismas) a través del entorno intuitivo que la *Ontología del Esquema* facilita.

De cualquier modo, la *Ontología del Esquema* representa el conocimiento que hay en una BDD de forma accesible a los usuarios y la generación de la misma es, como se expondrá a continuación, automática gracias a la definición previa realizada sobre la *Ontología del Catálogo*.

4.4.2. Generación o Conversiones

La *Ontología del Esquema* puede ser generada a partir de la definición del mismo en forma de instancia de la *Ontología del Catálogo*. De esta forma, se comienza dicha ontología a partir de las clases generadas por las metaclases (concretamente la metaclase *Tables*) y se realizan las siguientes acciones para obtener la *Ontología del Esquema* completa.

Se genera para cada tabla una propiedad por cada uno de los atributos definidos en ella (atributo *tableColumn*). El nombre de la propiedad se corresponde con el nombre de la columna asociada a la tabla. Además, cada una de estas propiedades tendrá cardinalidad con valor de 1, un valor atómico indivisible, para cumplir con la normalización (*Forma Normal 1*) que define el Modelo Relacional. Como rango estas propiedades tienen la estructura de datos correspondiente al tipo de datos para el que ha sido definido en la *Ontología del Catálogo*, de tal forma que:

- Si se trata de una columna difusa: Se genera una propiedad de objeto donde el rango se fija dependiendo del tipo de dato que tenga el dominio al que este ligado el atributo. Así pues este rango se fija siguiendo la tabla de correspondencias 4.15. Sin embargo, esta definición de rangos tiene una excepción consistente determinada por la existencia de restricciones difusas sobre el dominio. Así pues si sobre el dominio difuso correspondiente existe alguna restricción difusa, entonces el rango esta limitado únicamente a aquellas clases en las que se cumpla dicha restricción, y no a su superclase (véase la figura 4.11 y la sección 4.3.3.3 para conocer con más detalle dichas restricciones).

Tipo de Dato Difuso	Estructura de Dato Difusa
FType1	FType1_Struct
FType2	FType2_Struct
FType3	FType3_Struct

Tabla 4.15: Correspondencia de los tipos de datos difusos con las estructuras de datos difusas en la ontología

- Si se trata de una Columna Clásica: Se genera una propiedad de tipo de datos, donde se asocia a cada tipo de dato definido en la *Ontología del Catálogo* un tipo de dato definido en la ontología representacional. Los tipos de datos definidos en la ontología representacional son los definidos por el estándar XML en el caso de usar lenguajes de representación de ontologías basados en Web, como OWL o RDF, en otro caso, se vincula

el rango al tipo de datos base que tenga definido el propio lenguaje escogido.

T.D. Predefinido	Correspondencia con XML
Boolean	http://www.w3.org/2001/XMLSchema#Boolean
Varying	http://www.w3.org/2001/XMLSchema#string
Float	http://www.w3.org/2001/XMLSchema#float
Integer	http://www.w3.org/2001/XMLSchema#int
Date	http://www.w3.org/2001/XMLSchema#date
TStamp	http://www.w3.org/2001/XMLSchema#datetime
Time	http://www.w3.org/2001/XMLSchema#time
...	...

Tabla 4.16: Correspondencia de algunos de los tipos de datos predefinidos en la *Ontología del Catálogo* con los tipos de datos base definidos en XML

El uso de las estructuras de datos difusos que se plantean (véase tabla 4.15) requiere la utilización de la *Ontología del Catálogo*, dado que estos datos están definidos en la misma. Esta ontología es recomendable que sea importada, y usada de esta forma que no pueda ser modificada (dada su naturaleza como ontología representacional). Estas estructuras se encuentran definidas en esta ontología dado que se utilizan para representar la información imprecisa necesaria para definir los dominios de los atributos difusos (como las etiquetas lingüísticas).

Otro vínculo que existe entre la *Ontología de Catálogo* y la del *Esquema* reside en la definición de los dominios difusos del esquema a representar. Una vez definido el esquema como instancia de la *Ontología del Catálogo*, existen valores definidos a priori asociados con los dominios difusos, como etiquetas lingüísticas y relaciones de similitud entre valores discretos, que a pesar de estar definidos como instancias en esta *Ontología del Catálogo* deben estar disponibles para poder ser utilizados en la *Ontología del Esquema* recién definida.

Existen dos alternativas para resolver esta situación y así permitir el uso de los datos del dominio. Ambas alternativas contemplan el hecho de que las instancias de la *Ontología del Catálogo* se encuentran definidas en un nuevo archivo que tiene importada esta ontología. Y que la *Ontología del Esquema* se puede generar automáticamente a partir de dichas instancias.

- La primera consiste en definir en el mismo archivo donde estaban las instancias de Catálogo, la nueva *Ontología del Esquema*. Este es el caso

más habitual y tiene como ventaja que no sería necesario hacer ningún cambio sobre los dominios puesto que ya se encontrarían disponibles.

- La segunda consiste en utilizar un nuevo archivo para definir la *Ontología del Esquema*, eliminando así el archivo de instancias generado a partir de la *Ontología del Catálogo*. En dicho caso habría que importar los valores definidos en los diferentes dominios difusos.

La generación de la *Ontología del Esquema* también conlleva algunas desventajas. Existen restricciones del esquema, que se encuentran definidas (como instancias de la *Ontología del Catálogo*) que no pueden tenerse en cuenta en la *Ontología del Esquema*. Estas restricciones vienen dadas por la naturaleza propia de la ontología que estamos representando, tanto por los conceptos representados, como por el lenguaje de ontologías utilizado, OWL-Full (descrito en la sección A.2.3) que confiere una flexibilidad total para representar cualquier tipo de información. Esta flexibilidad que es necesaria para la representación de Metadatos, implica la misma flexibilidad a la hora de definir cualquier información sobre la *Ontología del Catálogo* esté permitido o no, como vemos a continuación:

- Es imposible comprobar si existe una coincidencia en la definición de una restricción de clave ajena, entre los atributos referenciados y los atributos que referencian (este inconveniente no es especialmente relevante dada la imposibilidad de definir claves ajenas, impedimento descrito en el punto siguiente).

- No es posible restringir los valores que pueden tomar los atributos que son claves ajenas. Es decir, se permitirá la inserción de cualquier valor en el campo a pesar de que se trate por definición de una clave ajena que hace referencia a otro atributo de otra relación.

- No se puede controlar que la definición de valores discretos en un atributo *Distr.poss*, coincida con el parámetro máximo *len* que determina el número máximo de valores que permite definir el dominio.

- Tampoco es posible que restricciones simples sobre los tipos de datos predefinidos (como definir una cadena de 20 caracteres, o un flotante con 3 decimales) se lleven a cabo. Esto se debe a que se utilizan tipos de datos genéricos de OWL sobre los que no existe ninguna limitación.

La solución a estas posibles violaciones de restricciones de integridad o de negocio serán tratadas en una fase posterior en la que ya intervenga el SGBD.

4.4.3. Ejemplos

4.4.3.1. Ejemplo 1: Clínica Veterinaria

Este ejemplo trata de mostrar el proceso de generación de una *Ontología del Esquema* a partir de las instancias de la *Ontología del Catálogo* y cómo dicha *Ontología del Esquema* permitiría definir información de tuplas, mediante su instanciación. Para ello se propone la utilización de la ontología descrita en el ejemplo por la figura 4.14 de la sección anterior 4.3.4.

Siguiendo el proceso de conversión descrito anteriormente, la *Ontología del Esquema de la BDD de la Clínica Veterinaria* que parte con las clases generadas por la metaclase *Tabla*, tendría la descripción en UML mostrada en la figura 4.15. Como puede observarse dependiendo del tipo de datos de los atributos, se establecen nuevas relaciones en la descripción del Esquema. Así, las clases van definiendo los atributos en forma de propiedades de objeto o de tipo de datos como se detalla a continuación:

- Todos los atributos clásicos se quedan como propiedades de tipo de dato en el diagrama UML aparecen como atributos de clase normales.
- Todos los atributos difusos sin ninguna restricción definida sobre ellos, se vinculan directamente a la estructura correspondiente tal y como se establece en la tabla 4.15. Por ejemplo la relación *Weight*.
- Todos los atributos difusos que contengan alguna restricción sobre el dominio difuso, restringen el ámbito de las estructuras difusas con las que están relacionados para cumplir dicha restricción. Este es el caso del atributo *Character* y *CharacterB* de la Tabla *Cats* y *Breed* respectivamente, y *Age* de la Tabla *Cats*. *Character* y *CharacterB*, al compartir dominio comparten la misma propiedad en la que sólo permitirán los valores de *FType3_Struct* a excepción de *Null*. En cambio *Age* no permite valores *Unknown*, por tanto todas las estructuras de *FType2_Struct* son permitidas a excepción de *Unknown*.

La instanciación de la ontología a partir de aquí es inmediata. Se podrán definir nuevas tuplas sobre el esquema de la BDD como las expresadas a continuación en lenguaje FSQL obteniendo los siguientes resultados:

```
INSERT INTO CATS VALUES(  1, "Kitty", $young, 2.3, $eager,
"siames")

INSERT INTO CATS VALUES(  2, "Garfield", $3, 1.9,
{$indiferent(0.9), $calmado(0.6)}, angora)
```

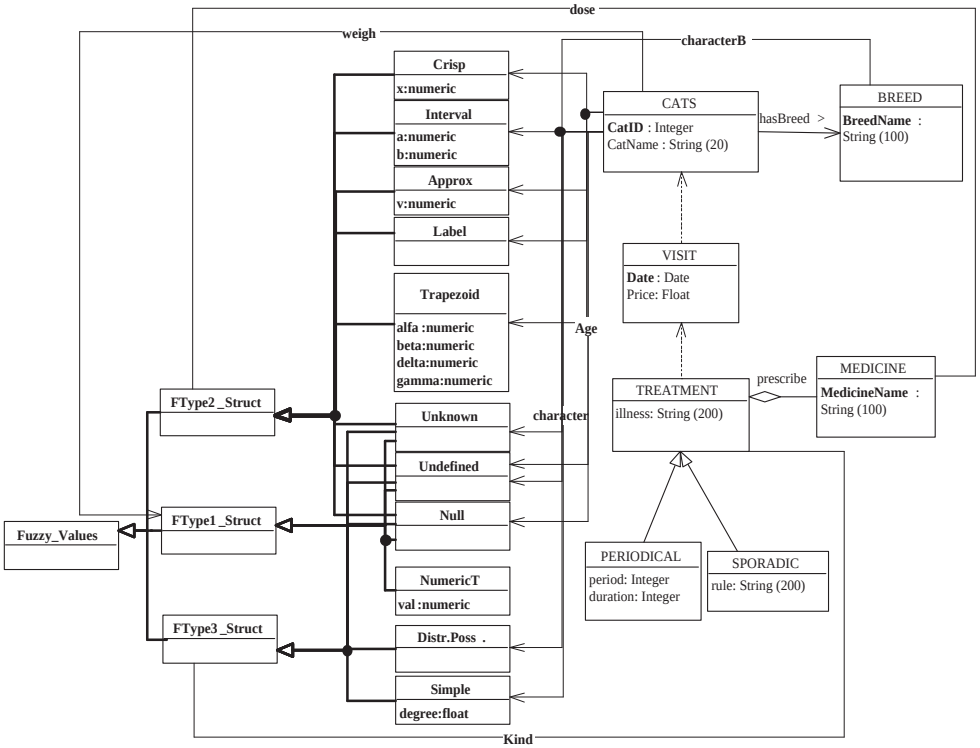


Figura 4.15: Ejemplo de una Clínica Veterinaria generada como una *Ontología del Esquema*

```
INSERT INTO BREED VALUES(  siames, {$inquieto (0.9), $jugeton
(0.8), $agresivo (0.5)})

INSERT INTO BREED VALUES(angora, $calmado)
```

Como podemos ver, con estas muestras, la inserción de datos implica la instanciación de las clases: *Cats* y *Breed* y de todas las clases relacionadas con ellas, tal y como se muestra en la figura 4.15. En la tabla 4.17 se listan todas las instancias generadas para definir estas tuplas, y en las tablas 4.18 y 4.19 se listan los valores y referencias de los atributos que componen los valores de dichas tuplas. Concretamente en la tabla 4.18 se describen los atributos que son de objeto, es decir, que hacen referencia a instancias de otras clases, de esta manera sabremos con qué instancia estarán vinculadas los valores *Age, Weight*, o *Character*. En cuanto a la tabla 4.19, se describen los valores concretos, como los referentes a *CatName* o *BreedName*.

Tabla 4.17: Instanciación de la Clínica Gatos definida en como Ontología de Esquema

rdf:ID	**Instancia de**	**rdfs:comment**
Cats1	Cats	1º tupla del ejemplo: instancia de *Cats*
Cats2	Cats	2º tupla del ejemplo: instancia de *Cats*
Breed3	Breed	3º tupla del ejemplo: instancia de *Breed*
Breed4	Breed	4º tupla del ejemplo: instancia de *Breed*
Label5	Label	define la referencia al valor *young*
NumericT6	NumericT	define el valor FTD1: 2.3
Simple7	Simple	define valor Simple *eager*
Approx8	Approx	define el valor Approximadamente 3
NumericT9	NumericT	define valor 1.9
DistrPoss10	DistrPoss	define valor indiferent(0.9),calmado(0.6)
Simple11	Simple	define el valor Simple *indiferente*
Simple12	Simple	define el valor Simple *calmado*
DistrPoss13	DistrPoss	define valor *inquieto*(0.9),*jugueton*(0.8)
Simple14	Simple	define el valor Simple *inquieto*
Simple15	Simple	define el valor Simple *jugueton*
Simple16	Simple	define el valor Simple *calmado*

4.4.3.2. Ejemplo 2: BDD Suelos

Siguiendo con el ejemplo de la BDD real de *Suelos*, la generación de la *Ontología del Esquema* de esta BDD a partir de las instancias definidas anteriormente sobre la *Ontología del Catálogo* ha generado una ontología que

Tabla 4.18: Propiedades de Objeto en la *Ontología del Esquema* del la Clinica Veterinaria

De	**Atributo**	rdf:resource
Cats1	*Age*	Label5
Cats1	*Weigh*	NumericT6
Cats1	*Character*	Simple7
Cats2	*Age*	Approx8
Cats2	*Weigh*	NumericT9
Cats2	*Character*	DistrPoss10
DistrPoss10	*discreteVal*	Simple11
DistrPoss10	*discreteVal*	Simple12
Breed3	*CharacterB*	DistrPoss13
Breed4	*CharacterB*	Simple16
DistrPoss13	*discreteVal*	Simple14
DistrPoss13	*discreteVal*	Simple15
Label5	*labelID*	LD_YoungCat

Tabla 4.19: Propiedades de tipos de dato en la *Ontología del Esquema* de la Clinica Veterinaria

De	**Atributo**	**rdf:datatype**	**value**
Cats1	*CatID*	xsd:String	1
Cats1	*CatName*	xsd:String	Kitty
Cats2	*CatId*	xsd:String	2
Cats2	*CatName*	xsd:String	Garlfield
Breed3	*BreedName*	xsd:String	angora
Breed4	*BreedName*	xsd:String	siames
NumericT6	*val*	xsd:Float	2.3
NumericT9	*val*	xsd:Float	1.9
Approx	*v*	xsd:Float	3
Simple7	*degree*	xsd:Float	-
Simple11	*degree*	xsd:Float	0.9
Simple12	*degree*	xsd:Float	0.6
Simple15	*degree*	xsd:Float	0.8
Simple16	*degree*	xsd:Float	0.5
Simple17	*degree*	xsd:Float	-

puede ser instanciada para almacenar/gestionar las tuplas de dicha BDD. Algunas de estas tuplas se encuentran definidas en el Anexo C, concretamente en la sección C.2.

La *Ontología del Esquema de la BD de Suelos* ha generado una clase por cada una de las relaciones definidas. Las propiedades de las mismas se han creado en función del tipo de dato que se trate, esto es, propiedades de tipo de datos, en el caso de que el tipo de dato sea predefinido y propiedades de objeto en el caso de que el tipo de datos se defina utilizando un dominio, como en el caso de los difusos. El resto de características, como restricciones también se han tenido en cuenta en la definición de la misma, siempre y cuando puedan ser aplicables, como por ejemplo, las restricciones difusas y las restricciones de cardinalidad.

En la tabla 4.20 se muestran todas las instancias generadas junto con una descripción de qué representan. Los valores concretos de las tuplas serán representados en las tablas de propiedades, que se dividen en dos: las propiedades de objeto, que hacen referencia a instancias que ya han sido definidas y que pueden verse en la tabla 4.21. Y las propiedades de tipo de dato, que representan valores concretos y que pueden verse en la tabla 4.22. Los valores representados en estas tablas se corresponden con las 2 primeras tuplas descritas en el apartado C.2 del Anexo C. Además, para seguir con la definición del esquema de la BDD de Suelos descrito en el apartado 4.3.4.2, sólo se visualizarán en estas tablas los valores correspondientes a los atributos definidos, esto es, para la tabla Localización: latitud, longitud, tmedia, fisiografía, codigo_es_loc y para la tabla Estructura: codigo_es y grado_es.

Tabla 4.20: Instanciación de la BDD Suelos definida en como Ontología de Esquema

rdf:ID	**Instancia de**	**rdfs:comment**
Localizacion1	Localizacion	1º tupla del ejemplo: instancia de *Localizacion*
Localizacion2	Localizacion	2º tupla del ejemplo: instancia de *Localizacion*
Estructura3	Estructura	3º tupla del ejemplo: instancia de *Estructura*
Estructura4	Estructura	4º tupla del ejemplo: instancia de *Estructura*

Este ejemplo es similar al anterior (*BDD Clínica Veterinaria*), pero menos rico en variedad de datos a definir. Los valores utilizados en una BD real consisten mayoritariamente en etiquetas lingüísticas descritas dado que son más fáciles de utilizar para el usuario final. El resto de datos son en datos numéricos de tipo flotante o entero.

Tabla 4.21: Propiedades de Objeto en la *Ontología del Esquema* de la BDD Suelos

De	**Atributo**	rdf:resource
Localizacion1	*fisiografia*	DD_ladera_fisiograf
Localizacion1	*tmedia*	DD_baja_tmedia
Localizacion2	*fisiografia*	DD_ladera_fisiograf
Localizacion2	*tmedia*	DD_baja_tmedia
Estructura3	*grado_es*	DD_Weak_grado_es
Estructura4	*grado_es*	DD_Weak_grado_es

Tabla 4.22: Propiedades de tipos de dato en la *Ontología del Esquema* de la BDD Suelos

De	**Atributo**	**rdf:datatype**	**value**
Estructura3	*codigo_es*	xsd:float	1
Estructura4	*codigo_es*	xsd:float	2
Localizacion1	*latitud*	xsd:float	41045
Localizacion1	*longitud*	xsd:float	5478
Localizacion1	*codigo_es_loc*	xsd:float	1
Localizacion2	*lontitud*	xsd:float	41135
Localizacion2	*longitud*	xsd:float	5598
Localizacion2	*codigo_es_loc*	xsd:float	2

4.5. Conclusiones

Se ha conseguido en este capítulo describir la *Ontología para la Representación del Conocimiento Difuso*. Dicha ontología, cuyo objetivo es representar la información de una Base de Datos Relacional Difusa, está compuesta por dos sub-ontologías de diferente tipo que representan la misma información. Una primera, se basa en la instanciación de la *Ontología del Catálogo*, ontología representacional que describe con exactitud la estructura del catálogo del Modelo Relacional. Una segunda, denominada de manera genérica *Ontología del Esquema*, representa en forma de ontología (no de instancias) la BDD que se pretende definir. Dicha ontología podrá ser instanciada y por lo tanto definir tuplas sobre ella.

Gracias a esta definición podremos aislar la representación de una BDD del modelo de representación dónde se almacene, a la vez que se le proporcionan como valor añadido todas las características que la representación ontológica aporta, como puede ser su presencia en la Web Semántica. A continuación se describen con detalle las ventajas e inconvenientes que presenta dicha propuesta.

4.5.1. Ventajas e Inconvenientes

Ventajas

La *Ontología de Representación del Conocimiento Difuso* plantea las siguientes ventajas, que se agruparan atendiendo a dos criterios:

Con respecto a la naturaleza de la información:

- Claridad en la Información. La información imprecisa y el Modelo Relacional quedan representadas a través de los conceptos fundamentales, atendiendo en particular a la generalidad de dichos conceptos y a la simplicidad de los mismos. Se trata de evitar que la definición de la información sea dependiente de una representación en particular que complica la comprensión de los datos.

- Independencia del SGBD. La representación de la ontología evita la relación directa con el SGBD con el que se esté trabajando. Este hecho, permite que las particularidades de cada SGBD se obvien en la representación de una BDD, y se dejen para las tareas de traducción o comunicación. De esta forma la definición de una BDD en la ontología siempre se realizará de la misma manera sea cual sea el recipiente de dicha información.

- Extensibilidad. Una representación genérica del modelo relacional difuso permite que la extensión del mismo, para representar otros tipos de datos más complejos o operaciones, sea más sencilla. La extensión sería realizada sobre la ontología (la capa abstracta) dejando los complejos detalles de la extensión sobre los SGBD ocultos para los usuarios finales.

- Normalización. Los datos definidos sobre la ontología siempre son definidos siguiendo el patrón definido en la *Ontología del Catálogo*. Por tanto la definición del esquema de BDD siempre tiene la misma representación. De esta forma ayudará a que cualquier aplicación que requiera el uso de esta información sólo necesite conocer los detalles de dicha *Ontología del Catálogo*.

- Automatización de la Conversión. Al diseñar el modelo relacional difuso utilizando una ontología con metadatos definidos en la misma, se establece el vinculo entre el esquema y la información del diccionario de datos. El esquema de BDD definido sobre la *Ontología del Catálogo* permite una conversión directa a la generación de una *Ontología propia del Esquema*. Dada la naturaleza de la información que se encuentra normalizada y la relación que existe entre ambas definiciones (ontologías) el proceso exacto para poder realizar dicha traducción esta claramente determinado y puede ser fácilmente automatizado.

Con respecto a la interacción con el entorno:

- Estandarización. Con la *Ontología del Catálogo* se obtiene una plantilla accesible y pública sobre la que se definen los datos de un esquema difuso. Cualquier esquema difuso definido utilizándola sería accesible por cualquier usuario/programa y compartiría la misma representación que otro esquema de las mismas características (sea cual sea el lugar donde esté almacenado).

- Automatización. Se puede automatizar la interacción con el SGBD, es decir, establecer una vía de comunicación entre la ontología y cualquier SGBD y así poder intercambiar información.

- Publicación de Datos Difusos. La ontología de un Esquema de BDD relacionales en Web, permite que los usuarios tengan información difusa accesible desde cualquier mecanismo de consulta que lo permita.

- Publicación de Esquemas de BDD en Web. La publicación de la ontología de cualquier Esquema basado en BDD confiere semántica a la información que no puede ser anotada semánticamente por cualquier otro medio,

dado que la información que una BD contiene no se encuentra incluida en los archivos web.

- BD Heterogéneas. La disponibilidad de una *Ontología del Catálogo* genérica sobre el Modelo Relacional (con datos difusos) permite la compartición de información entre BBDD de muy diversa índole, puesto que todos los SGBDs comparten la misma definición del Catálogo descrito en dicha ontología.

- Compartición y otras Operaciones con el Entorno. Se presenta otra nueva forma de representar información utilizando esquemas de BDD mediante una ontología. Este esquema puede ser compartido y usado por otras representaciones que definan la misma realidad o complementaria. Existen (tal y como se describió en el capítulo anterior) otros tipos de representación de esquemas como ontologías que no están basadas en modelos relacionales, Esquemas de BD, Esquemas XMLs, Esquemas en RDF, folksonomías, jerarquías de conceptos, etc. Toda esta información necesita mecanismos para permitir su interacción, pero la publicación de cómo la información esta estructurada en dichos esquemas, permite que estos mecanismos sean fácilmente definibles.

Inconvenientes

Esta propuesta también presenta algunas desventajas, tal y como se ve a continuación:

- Lenguaje inteligible: Por un lado, la representación de la ontología en un entorno de frames, permite la interpretación de la misma de forma intuitiva en la que los conceptos son claramente distinguibles. Sin embargo, tal y como ocurre en este caso, si se utiliza un lenguaje de representación web, la interpretación de la misma se hace tediosa e incluso imposible dada la ingente cantidad de etiquetas que hacen falta para representar todos los conceptos.

- Necesidad de una aplicación para interaccionar con la ontología: Dada la naturaleza del lenguaje de representación OWL, ininteligible para el usuario, se hace necesaria una herramienta que permita visualizar y/o editar dicha ontología de forma intuitiva para el usuario.

- Aplicación para definir /consultar. El proceso de consulta o definición de esquemas requiere de alguna aplicación específica que permita realizar dicho proceso de manera guiada al usuario. La gestión o definición de

los datos del esquema en una ontología requieren una aplicación para permitir tratar la información difusa de forma correcta.

- Necesidad de una aplicación para la conversión automática. Si se desea realizar la automatización para la generación de la *Ontología del Esquema* a partir del esquema definido como instancias de la *Ontología del Catálogo*, es necesaria la elaboración explícita de los procesos que permitan generarla.

- Comunicación con los SGBD compleja. La ontología no almacena los datos como finalidad, se trataría de un interfaz que comunica con el SGBD donde los datos estarían almacenados. La comunicación deberá contemplar las particularidades de cada SGBDs. Así cualquier interacción con el SGBD tendrá que tener un módulo que interprete las características del mismo para poder interaccionar con él. Estas particularidades serán ampliamente detalladas en el capítulo siguiente.

- Dependencia del programa de definición de ontologías. Dado que es complicado definir de manera manual la ontología, por lo tedioso del lenguaje, la generación de la *Ontología del Esquema* requiere el uso de librerías y convenciones (JENA [Pro07]) para generar el código en OWL automáticamente. Esta característica vuelve dependiente a la ontología de las particularidades de la librería escogida o lenguaje utilizado.

Capítulo 5

Arquitectura del Sistema y Aplicaciones

5.1. Introducción

En este capítulo se realizará una descripción de la arquitectura necesaria para desarrollar y explotar la *Ontología de Representación del Conocimiento Difuso* descrita en el capítulo anterior.

Tal y como se definió anteriormente la *Ontología de Representación del Conocimiento Difuso* esta dividida en dos sub-ontologías: la *Ontología del Catálogo* y la *Ontología del Esquema* que permiten representar la información difusa almacenada en una BDD Relacional, a la vez que tenerla disponible en forma de ontología. Para que dicha representación de información se lleve a cabo se requiere la ejecución de dos procesos bien diferenciados: por un lado, se necesita establecer un sistema que permita la representación de la *Ontología del Esquema*, y de alguna manera facilitar al usuario la definición/manipulación de los datos difusos de manera amigable e intuitiva. Por otro lado, dicha *Ontología del Esquema* una vez que se ha desarrollado, se comunica con el SGBDD correspondiente. Dicha comunicación no es trivial, dado que los SGBDDR no comparten la misma forma de representación de datos, ni soportan el mismo tipo de lenguaje (cada uno realiza una representación del SQL distinta) ni tienen las mismas capacidades funcionales.

Para llevar a cabo estos procesos de representación y manipulación de información difusa a través del uso de una ontología y su posterior comunicación con un SGBD real se describe la *Arquitectura del Sistema*. En ella se presenta el flujo de información de la Ontología al SGBD especificando los módulos más representativos que constituyen el sistema y todos los casos que pueden darse en dicha comunicación.

Se presentarán también, de manera razonada, las diferentes decisiones que se han tomado para llevar a cabo la implementación de dicha arquitectura, desde herramientas que permitan generar una ontología, hasta las particularidades de los diferentes SGBD que permiten representar información imprecisa.

Además se muestra la aplicación desarrollada, describiendo qué funcionalidad aporta y qué herramientas y tecnologías han sido utilizadas en la implementación de la misma.

Por último, en este capítulo se realizará un repaso por las diferentes casos de uso que la arquitectura presenta a la hora de manipular la información imprecisa representada en una BDR destacando, cuáles se han llevado a cabo y cuáles serán fruto de trabajos posteriores a éste.

5.2. Arquitectura del Sistema

La arquitectura que se propone permite trabajar con información imprecisa desde el momento en que un usuario desea representar/manipular información hasta la representación de estos mismos datos en un SGBD Relacional Difuso cualquiera, sean cuales sean sus características. En la figura 5.1, se muestra un esquema genérico de la Arquitectura del Sistema.

Esta arquitectura puede dividirse en dos fases dados los diferentes problemas a resolver: la primera fase conduce a la obtención de la *Ontología del Esquema*, y se describirá en el siguiente apartado como *Arquitectura de Comunicación con la Ontología*. La segunda fase describirá los diferentes mecanismos de conexión con los SGBDs, y se describirá como *Arquitectura de Comunicación con la BD*.

5.2.1. Arquitectura de Comunicación con la Ontología

Tal y como se muestra en la figura 5.1, esta arquitectura permite generar la *Ontología del Esquema*. Para ello, se requiere la utilización de los siguientes módulos:

Interfaz de Usuario Este módulo consiste en un entorno amigable que permite a un usuario novel la generación una BD Difusa y la manipulación de los datos almacenados en la misma (esto incluye las BD clásicas también). Además dicha interfaz debe gestionar tanto la información relativa a los metadatos que describen una BDD como los contenidos en ella.

Generador de OWL Este módulo está destinado a generar el código en OWL necesario para la definición y manipulación de esquemas difusos utilizando la *Ontología de Representación del Conocimiento Difuso*.

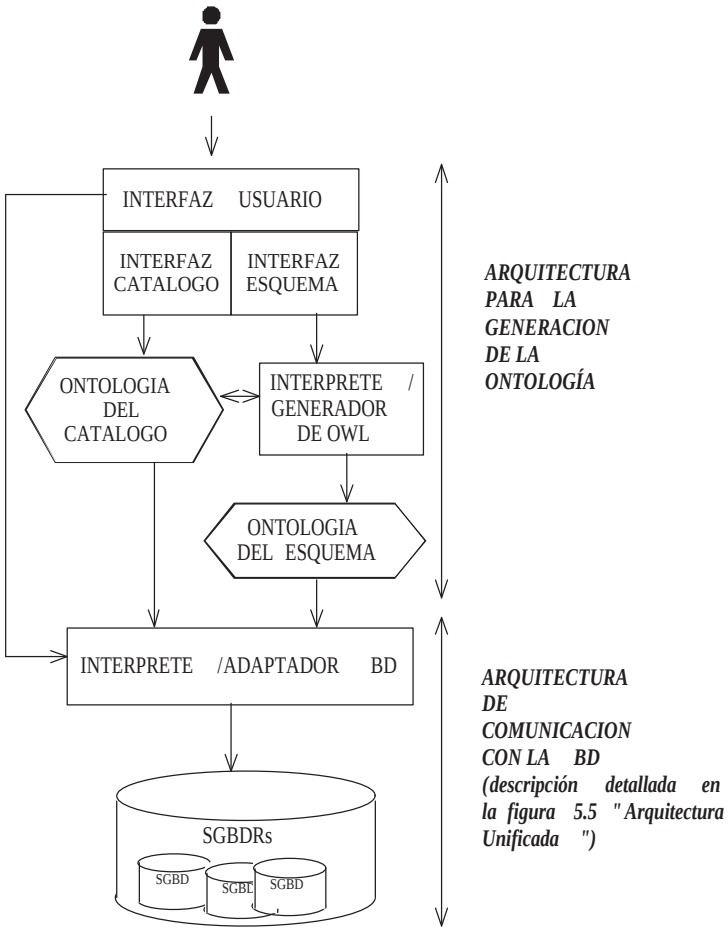


Figura 5.1: Arquitectura del Sistema General

Para ello debe proporcionar los procedimientos para leer la *Ontología del Catálogo* y permitir su instanciación y generar la *Ontología del Esquema* derivada de la misma.

Ontología del Catálogo Este módulo representa a la *Ontología del Catálogo* (descrita en el capítulo anterior) cuyo uso es imprescindible para la generación de la *Ontología del Esquema* en código OWL. Todas las representaciones de BDRD en forma de ontología requieren la incorporación/importación de esta (meta)ontología para poder definir y manipular las estructuras de una BDRD.

Ontología del Esquema Ontología en OWL que describe un esquema de BD Difusas.

Los dos primeros módulos requieren de la utilización de herramientas software, que permitan llevar a cabo la funcionalidad que describen. A continuación se detallan cada uno de estos módulos.

5.2.1.1. Interfaz de Usuario

Uno de los principales motivos que llevan al desarrollo de la *Ontología para la Representación del Conocimiento Difuso* se basa en la dificultad para definir información imprecisa en un SGBD Extendido. Esta dificultad, que se incrementa conforme las extensiones al modelo se van incorporando, ha inducido a la generación de una ontología que mantenga la definición de dicha información al margen de cualquier representación concreta de un SGBD.

De esta forma, la generación de una interfaz de usuario es un elemento fundamental en la arquitectura. Las tareas más básicas que dicha interfaz debe aportar son:

- Permitir la definición de un Esquema de BDD a partir de la instanciación de la *Ontología del Catálogo* descrita en esta tesis.
- Conectar con un SGBDRD cualquiera para incorporar la información descrita en forma de ontología. La conexión podría realizarse con varios SGBDRs simultáneamente.
- Mantener al usuario al margen de los detalles propios de la definición de datos difusos, para lo que se requiere que la interfaz sea lo más intuitiva posible.
- Contemplar la opción de manipular las estructuras del catálogo para los casos de SGBD que carezcan de las estructuras para la representación

de datos difusos. Esta opción debe incluir la posibilidad de extender el sistema en caso necesario.

Como en todos los sistemas software, las decisiones de desarrollo se toman en función de la disponibilidad y métodos de acceso que se desea que la herramienta proporcione, las dos principales alternativas son:

- Utilizar una herramienta local de desarrollo propio, basada en tecnología Web o cualquier otra plataforma (dependiendo del lenguaje usado para su desarrollo) que permita la consulta y manipulación de la ontología.
- Usar *Herramientas de Gestión de Ontologías* que permitan la edición de la ontología y su instanciación. Esta opción se contempla dado que existen herramientas que permiten su extensión para incorporar nuevas funcionalidades.

Dicha interfaz de usuario se divide en dos, dependiendo del tipo de usuario que va a tener acceso a la misma, y de la información que dichas interfaces están encargadas de gestionar y que a continuación se detalla.

Interfaz de Catálogo

La *Interfaz del Catálogo* esta destinada a ser utilizada por los administradores del SGBD que se deben de encargar de definir las estructuras necesarias en el catálogo del sistema para que la definición de datos difusos sobre la BD sea posible. De esta forma, dicha interfaz debe permitir las funciones de:

- Generar tablas del catálogo que permitan la definición de datos difusos.
- Identificar las particularidades de cada SGBD para incorporar dichas tablas del catálogo en el espacio más adecuado, dependiendo del sistema que se trate.
- Establecer permisos sobre dichas tablas para que los usuarios puedan acceder a ellas.
- Dependiendo de las particularidades del SGBD en cuestión, incorporar la funciones de gestión de datos difusos necesarias para el manejo de los mismos (operaciones de comparación, interpretación de consultas, etc.). Estas funciones pueden estar incrustadas en el sistema, o bien ser ajenas al mismo, como veremos más adelante.

Para llevar a cabo estas funciones es requisito, en la mayoría de los SGBDs, tener privilegios de administrador, dado que se trata de tablas a incorporar al catálogo del sistema, y que interaccionan con el resto del mismo. No obstante esto depende del SGBDs en el que se esté trabajando.

Esta interfaz sólo se utiliza en el proceso de definición o extensión de un SGBDR común, para incluirle la funcionalidad de gestionar información imprecisa. Por lo tanto, será utilizado en una sola ocasión para cada sistema que se instale.

Interfaz de Esquema

La interfaz del esquema, será la mas utilizada, pues es la encargada de permitir al usuario realizar las siguientes funciones:

- Permitir definir el Esquema de BD Difusa o Clásica de manera intuitiva para el usuario final. Esto puede llevarse a cabo a través de asistentes o formularios sencillos.
- Permitir la visualización del Esquema de BD Difusas de manera intuitiva, sin necesidad de que el usuario final tenga que acceder a la BDD para hacerlo, ni de conocer la sintaxis del código OWL.
- Permitir la generación de la *Ontología del Esquema* de manera automática, es decir, que la misma herramienta una vez que tiene definido el esquema en forma de instancias de la *Ontología del Catálogo* autogenera las clases y relaciones necesarias para que dicho esquema sea a su vez instanciable.
- Permitir la conexión con SGBD heterogéneas, de las que conoce sus particularidades, para así poder realizar la definición de las estructuras definidas en OWL.
- Permitir la definición de datos sobre la *Ontología del Esquema* generada, es decir, la definición de las tuplas.
- Permitir la generación de consultas de datos difusos en FSQL a partir de la ontología.
- Permitir la comunicación simultánea con SGBDRDs para realizar cualquier tarea de definición o manipulación a la vez, sin tener en cuenta las particularidades de cada sistema.

La implementación final de estas funciones se describe en los apartados 5.3.3 y 5.3.4 de este capítulo.

5.2.1.2. Generador de OWL

Tal y como se viene expresando en los capítulos anteriores, el uso del lenguaje de representación de ontologías de OWL(véase sección A.2.3 del A) conlleva tantas ventajas como inconvenientes. Entre los inconvenientes más destacables se encuentra el gran coste de desarrollar una ontología en OWL de forma manual, dada la naturaleza misma del lenguaje que siendo simple, es muy tedioso a su vez en la representación de cualquier concepto debido al gran número de etiquetas que necesita para ello. De esta forma, una representación manual de una ontología en OWL sería un error no sólo por el esfuerzo en realizar esta tarea, sino por la alta probabilidad de cometer un error en la misma, hecho que impediría cualquier manejo automático de la ontología por cualquier aplicación a posteriori.

De esta forma, se estima necesaria la utilización de herramientas que interpreten ontologías representadas en OWL y que a su vez permitan la definición de nuevos conceptos en dicho lenguaje. Existen dos alternativas para trabajar con OWL:

- Utilizar librerías como *JENA*, *Sesame* u *OWLAPI* (opciones descritas en el Anexo A). Esta opción permite incorporar a cualquier programa de gestión de datos propio, operaciones de manipulación de código *OWL* mediante la utilización de los métodos incluidos en dichas librerías.

- Utilizar programas de gestión de ontologías *per se*. Estos programas proporcionan toda la funcionalidad necesaria para definir ontologías de forma gráfica (en su mayoría) o cuanto menos intuitiva. Además la mayoría permiten una gestión integral de las mismas, opción más que deseable. Una de las aplicaciones más populares que ofrecen dicha funcionalidad es *Protégé*, también existen otras como *OntoEdit*, *OntoStudio*, etc. (véase sección A.2.3.3 para conocer las características principales y referencias a estas herramientas).

5.2.2. Arquitectura de Comunicación con la BD

Los SGBD Relacionales actuales a pesar de representar el mismo modelo de datos, realizan dicha representación de modos muy diferentes. Existen diferencias tanto en la definición del estándar de SQL que implementa cada sistema (véase [Gen06]), hasta en la forma para gestionar y representar la información en el mismo, hecho más que lógico dado que cada sistema tiene definido su propio formato de catálogo para almacenar los metadatos. Además, dependiendo del SGBD de que se trate, incorpora mecanismos de comunicación y explotación del sistema con diferente grado de eficiencia, por ejemplo algunos

incorporan un lenguaje de programación incrustado, opción muy deseable en términos de eficiencia y rapidez, mientras que otros simplemente permiten la comunicación con lenguajes o programas ajenos al sistema, opción mucho más lenta e ineficiente.

De esta forma, dependiendo del tipo de SGBD Relacional (SGBDR) en el que se quiera implantar la base de datos representada por la ontología, es necesario interponer una serie de fases que traduzcan las acciones recogidas de forma implícita en la ontología a las sentencias propias de un SGBDR. Dichas fases se encuentran descritas en [Bla08a] y a continuación:

5.2.2.1. Extensión del SGBDR para Incorporar el FSQL

La extensión difusa *FSQL* [Bla03b] del lenguaje relacional SQL permite la creación y manejo de estructuras relacionales capaces de contener atributos con dominios difusos.

Un sistema con estas características incorpora sus propias estructuras de catálogo para la representación de los dominios difusos y para la creación de relaciones con datos difusos. Además, incorpora el conjunto de funciones necesarias para gestionar dichas sentencias FSQL. Por ello, parece que carece de sentido la implantación de la *Ontología para la Representación del Conocimiento Difuso* dado que la definición de datos difusos en este caso se podría realizar utilizando directamente las sentencias FSQL, sin entrar en detalles de representación.

Sin embargo, para implantar las estructuras relacionales contenidas en la *Ontología del Esquema* de datos difusos es necesario generar las sentencias FSQL que creen dichas estructuras en la base de datos, empleando un módulo de traducción como el observado en la figura 5.2. Con esta arquitectura, traducimos la ontología al lenguaje FSQL y la sentencia traducida se envía a la base de datos para su procesamiento (que incluye traducción a SQL con capacidades funcionales) y ejecución.

Existen por tanto dos alternativas diferentes en la extensión de los SGBDR necesaria para representar dicha información imprecisa y que determinarán el modo de interpretación del FSQL. Estas configuraciones son directamente dependientes de la capacidad del SGBD para incorporar capacidades funcionales (es decir, que carece de un lenguaje de programación que le permite la creación de bloques de sentencias para la operación con datos) como se describe a continuación.

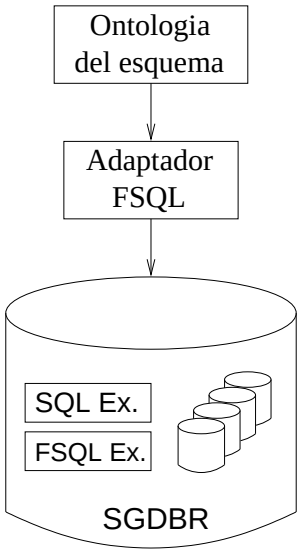


Figura 5.2: Arquitectura de integración con un SGBDR con capacidades FSQL

5.2.2.2. SGBDR con Capacidades Funcionales

La implantación de un esquema de BDD en un SGBDR que carezca de la implementación *FSQL* pasa por la creación de las estructuras de catálogo relacional extendido de forma difusa, que permitirán almacenar la información referente a la *ontología para la representación de datos difusos*. Incorporando las relaciones de dicho catálogo, almacenamos la información referente a los dominios y datos difusos contenidos en la base de datos, haciendo posible el acceso a la información desde las funciones y procedimientos que se creen a tal efecto pero sin depender de la ontología.

Sin embargo, toda la funcionalidad que incorpora la implementación del lenguaje *FSQL*, vista en el apartado 5.2.2.1, tendrá que ser proporcionada en este caso por una serie de bloques funcionales implementados en el lenguaje procedimental que proporcione la implementación relacional dada. SGBDR como Oracle© o PostgreSQL© incorporan estas capacidades mediante la definición de sus lenguajes procedimentales Oracle® PL/SQL y PG/PLSQL, respectivos.

Si bien proponemos un paso intermedio para la representación en *FSQL* de toda operación realizada a través de la ontología, en el caso de este tipo de sistemas, será necesaria la traducción de la sentencia al lenguaje SQL (el cual incluirá llamadas a funciones que resuelvan los problemas difusos) puesto

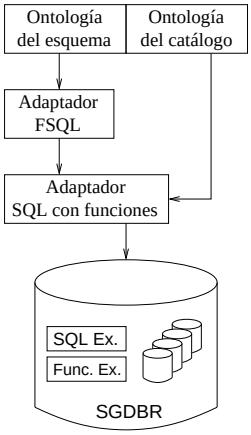


Figura 5.3: Arquitectura de integración con un SGBDR con capacidades funcionales

que el sistema carece de la capacidad de compilar y ejecutar sentencias *FSQL*. El procesamiento de la sentencia FSQL tiene que ser realizado a nivel del *Adaptador a SQL con funciones*, mostrado en la figura 5.3, el cual traduce la sentencia FSQL a una sentencia SQL con llamadas a funciones que tendrán que estar almacenadas en la base de datos para su llamada cuando se ejecute la sentencia SQL traducida. Estas funciones insertadas en el SGBD son las encargadas de realizar operaciones con los datos difusos que en ellas se encuentran representados.

5.2.2.3. SGBDR sin Capacidades Funcionales

En este caso, se implanta el esquema de base de datos contenido en un SGBDR que no incorpora capacidades funcionales, por lo tanto, la base de datos actúa como mero recipiente de datos. De este modo, todas las fases del tratamiento de la sentencia *FSQL* (traducción y procesamiento) tendrán que ser implementadas en un lenguaje de programación externo al propio SGBDR.

Este es el caso de SGBDR como MySQL®, en el que se puede proveer el tratamiento de la sentencia *FSQL* mediante una serie de elementos funcionales programados en Java. Lo cual ralentizaría la ejecución de cualquier operación que requiriera de esta funcionalidad.

En un sistema de este tipo, el elemento denominado *Módulo funcional*, que se muestra en la figura 5.4, tendrá que realizar la interpretación de la consulta FSQL y, mediante diversas consultas SQL a la base de datos, obtener conjuntos de datos relacionales (no difusos) a los que aplicará funciones para

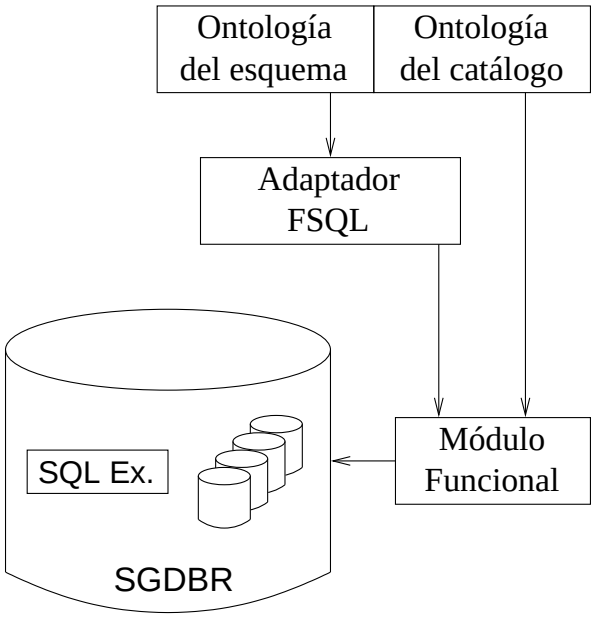


Figura 5.4: Arquitectura de integración con un SGBDR sin capacidades funcionales

proporcionar la semántica difusa a estos datos y poder operar con ellos.

5.2.2.4. Arquitectura Unificada

Reuniendo todas las posibilidades de SGBDR vistas en los apartados anteriores, la arquitectura unificada quedaría como se muestra en la figura 5.5.

En dicha arquitectura, la ontología actuaría como interfaz abstracta para la definición de datos difusos o clásicos, independiente de cualquier SGBD que actúe como recipiente de la información. Dicha ontología proporcionaría tanto la definición de los datos de la BDD, como cualquier petición de manipulación sobre los mismos, sin tener en cuenta las particularidades que cada SGBDD tiene en su representación.

Además, en la arquitectura separamos por niveles todos los aspectos relativos al lenguaje de consulta de datos difusos en el que nos basamos (como primer paso para la operación sobre la base de datos) de los aspectos específicos relacionados con la implementación concreta. Así pues nos encontramos con:

Primera Fase de Adaptación al Lenguaje Se utiliza el *Adaptador FSQL*,

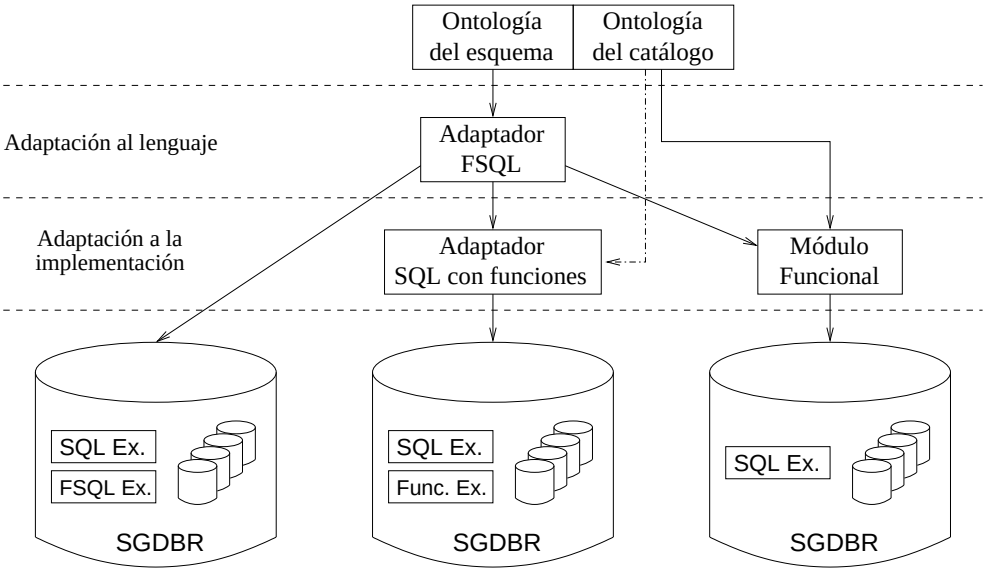


Figura 5.5: Arquitectura Integrada

que como se ha especificado anteriormente, permite generar la sentencia de definición o manipulación de datos en este lenguaje.

Segunda Fase de Adaptación a la Implementación Entra a formar parte las particularidades del SGBD con el que se esté trabajando. Por tanto la sentencia lanzada contra la BDD (en FSQL) será resuelta en función de la decisión tomada para extender el SGBDR con capacidad de tratamiento de datos difusos sobre el que ha sido lanzada.

En la actualidad, la primera fase de adaptación al lenguaje se ha desarrollado tanto a través de las herramientas generadas en esta tesis para la gestión de la ontología, como a través de la utilización de otras herramientas como el *FuzzyQuery2+* [Bla02b]. La segunda fase de adaptación a la implementación se encuentra íntegramente desarrollada y en perfecto funcionamiento para el SGBDR Oracle ©. Esta función permite generar cualquier sentencia correcta en FSQL y posteriormente procesarla automáticamente sobre dicho sistema. Esto es posible gracias a que Oracle tiene capacidades funcionales (PL/SQL) y una implementación completa, usando dichas capacidades, de las operaciones para analizar y ejecutar una sentencia FSQL.

Como trabajos futuros se propondrá la implementación de las operaciones para analizar y ejecutar una sentencia en FSQL utilizando un lenguaje

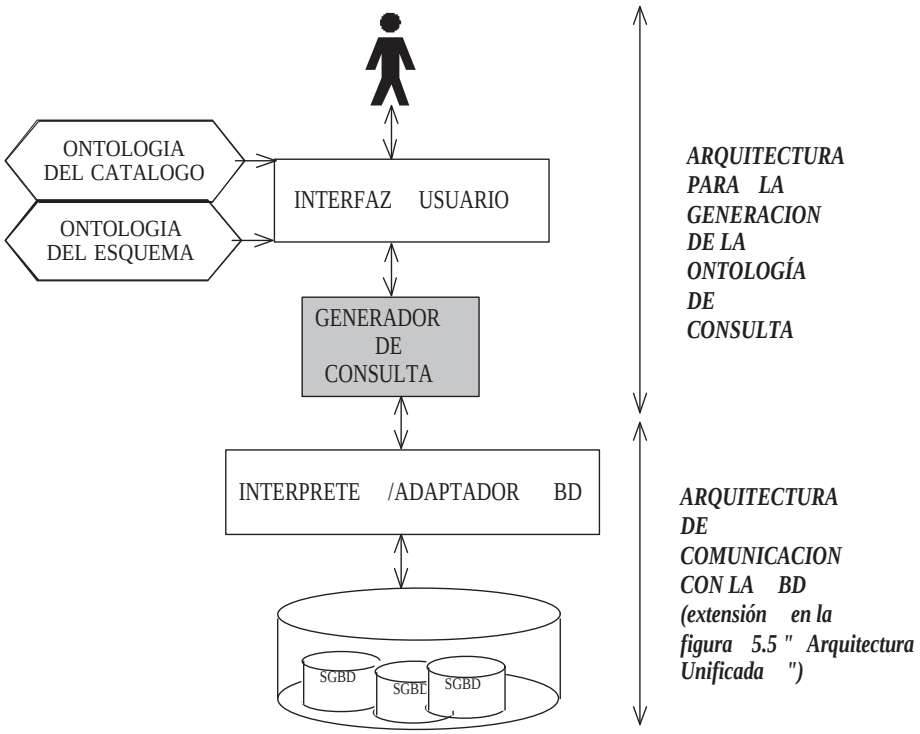


Figura 5.6: Arquitectura de Consulta

de programación genérico, que esté aceptado por la mayoría de SGBDR, y fundamentalmente por aquellos que no dispongan de capacidades funcionales.

5.2.3. Arquitectura de Consulta

La arquitectura del sistema planteada en el apartado anterior, está diseñada para la definición de la *Ontología del Esquema* y su posterior comunicación con el SGBDRD correspondiente para su correcta generación. Incluso también es válida en la definición de datos (tuplas) sobre dicha *Ontología del Esquema*, a través de la instanciación del mismo.

Sin embargo, el flujo de información que hay en la misma es unidireccional, es decir, no permite comunicación con el usuario final una vez definido el mismo.

La Arquitectura de Consulta debe establecer las bases para permitir al usuario definir una consulta relacionada con el Esquema sobre el que quiere consultar, de forma transparente al SGBDRD consultado, y devolver al mismo

los resultados. Es por este motivo que se hace necesaria una modificación a la Arquitectura del Sistema previamente propuesta, en el que el flujo de información sea bidireccional.

La parte de la *Arquitectura de Comunicación con la BD* es idéntica a la descrita en el apartado 5.2.2 anterior, a excepción de que las comunicaciones en esta arquitectura, son también bidireccionales.

Por otro lado, con respecto a la *Arquitectura de Consulta* se hace necesario la definición de un nuevo elemento en la misma, el *Generador de Consulta*. Este Generador conduce a la generación del código en FSQL requerido para poder formar la consulta a partir de los datos descritos en las *Ontologías del Esquema y el Catálogo*.

La *Ontología del Esquema* es necesaria en tanto que proporciona la estructura de información que puede ser consultada. Además, se encuentra definida en la misma el dominio de cada uno de los atributos que comprenden el esquema. El conocimiento de dichos dominios por parte del usuario para realizar la consulta es imprescindible cuando se trata de atributos difusos, dado que muchos de los valores permitidos se deben hallar previamente definidos en la ontología. Este sería el caso de los atributos de *Tipo Difuso 3*, cuyos valores únicamente serán aquellos definidos y relacionados entre ellos explícitamente. Un ejemplo de este uso sería el de acceder a etiquetas previamente definidas sobre un atributo en la BDD como *Alto, Bajo, Rubio o Moreno* para poder realizar una operación de comparación.

Por otro lado, la *Ontología del Catálogo* nos permite acceder a las estructuras de datos, concretamente las estructuras difusas, necesarias para poder definir los valores sobre los que se van a realizar las comparaciones en dichas consultas. Por ejemplo, nos va a permitir definir un valor *Trapezoidal* o *Aproximado* para ser comparado con los que se hallan almacenados en la BDD.

La interfaz de usuario debe proporcionar un entorno intuitivo en el que el usuario sea capaz de definir una consulta en términos difusos sin necesidad de conocer el lenguaje FSQL. La implementación de dicha interfaz será planteada de igual manera que se ha descrito para la interfaz de usuario del apartado anterior 5.2.1.1. La funcionalidad cubierta por la herramienta de consulta y la puesta en marcha mediante el desarrollo de una aplicación se describe con detalle en el apartado 5.3.4.3.

5.3. descripción del Sistema Implementado

5.3.1. Propuestas

La utilización de la ontología propuesta en este trabajo de tesis se llevará a cabo a través del desarrollo de un entorno que permita la definición y mani-

pulación de esquemas de BDRD y su explotación mediante consultas. Dichas implementaciones se describen a continuación:

- Un *Entorno Web* que permite la gestión de la *Ontología del Catálogo* y su comunicación con diversos SGBDD. A su vez, este mismo entorno implementa las funciones de gestión de esquemas de BDD (generación y consulta), a través del uso de ontologías. Esta propuesta está descrita en la sección 5.3.3.

- La *Extensión de una Herramienta Integrada de Gestión de Ontologías*: *Protégé*, para la generación de Esquemas de BDD basados en la *Ontología del Catálogo*, definición de datos sobre la *Ontología del Esquema* y para la generación de consultas sobre la BDD. Esta funcionalidad está descrita en el apartado 5.3.4.1 para esquemas, 5.3.4.2 para inserciones y 5.3.4.3 para consultas.

5.3.2. Bases de Datos Utilizadas

Las bases de datos son un elemento imprescindible en este trabajo de tesis dado que el motivo principal para el desarrollo de la ontología es permitir la definición de Esquemas de Bases de Datos Difusas en cualquier SGBDR Difuso, con independencia de su implementación física y particularidades de uso.

Por tanto, en la arquitectura del sistema (sección 5.2.2) cuando se hace referencia a las bases de datos difusas, se ha de tener en cuenta las particularidades del SGBD para que pueda ser extendido y así manejar datos difusos. La arquitectura se divide en tres casuísticas de conexión con el SGBD, que son directamente dependientes de la funcionalidad aportada por dichos sistemas.

En la implementación de este trabajo, se propone la utilización de un SGBD que sea representativo para cada una las diferentes arquitecturas de extensión presentadas. Así pues, los SGBDR utilizados son:

Oracle © Este sistema representa a un SGBDR con capacidades funcionales pero que puede tener implantado o no el FSQL en su totalidad (referente a la figura 5.2 y 5.3). De esta manera dispone entre su conjunto de funciones de todas aquellas necesarias para procesar una sentencia en dicho lenguaje y devolver los resultados al usuario de forma legible. Evidentemente también su catálogo del sistema se encuentra extendido para poder llevar a cabo toda esta funcionalidad.

PostgreSQL © Este sistema representa a un SGBDR con capacidades funcionales igual que ocurre con Oracle ©. Su lenguaje de programación

incrustado PL/pgSQL permite la inclusión de funciones dentro de su sistema. Sin embargo, no se encuentra implementada ninguna funcionalidad de gestión de datos difusos en este lenguaje en la actualidad. Así pues, sobre PG se puede desarrollar tanto una implementación especial para FSQL incrustada en el mismo, o bien una aplicación genérica a través de un módulo ajeno a dicho sistema, tal y como se describe en la sección 5.2.2.3.

MySQL © Este sistema representa a un SGBD sin ninguna capacidad funcional. Por tanto, este sistema únicamente puede incorporar las estructuras del catálogo, pero la ejecución de sentencias, tanto de definición como de manejo de datos difusos, debe llevarse a cabo en un modulo externo. Este será el que se ocupe de lanzar las consultas a la base de datos en un lenguaje comprensible para la misma, o sea, en SQL, de obtener los resultados, realizar las comparaciones difusas si es pertinente y formatear los datos de salida para mostrarlos al usuario (arquitectura correspondiente con la figura 5.4 y sección 5.2.2.3). Dicho sistema requiere la implementación de su funcionalidad utilizando lenguajes de programación externos que sean capaces de comunicarse con la misma, como es el JAVA, que es el lenguaje utilizado en este trabajo.

Es obvio que un SGBD con capacidades funcionales, proporciona un mayor rendimiento en la gestión de información imprecisa puesto que todas las funciones u operaciones son ejecutadas en el mismo entorno y por tanto con mayor rapidez. Sin embargo estos sistemas con capacidades funcionales requieren una implementación propia para la gestión del FSQL, y por tanto un esfuerzo muy significativo para poder ponerlo en marcha.

Por otro lado la utilización de un lenguaje de programación genérico como el Java para crear un entorno en el que las funciones sean independientes del SGBD evita la necesidad de adaptar las funciones de gestión del lenguaje FSQL a cada sistema particular, pero provoca una gran caída del rendimiento, dado que los datos deben portarse de un lugar a otro continuamente para ser procesados.

5.3.3. Entorno Web

A través de un entorno web (véase figura 5.7) se ha pretendido que el usuario sea capaz de realizar las funciones básicas que proponemos en este trabajo de tesis, esto es, la definición y manipulación de esquemas difusos sobre un SGBD Extendido. Esta herramienta se desarrolla con el objetivo de permitir comunicar al usuario con los diferentes SGBD sin necesidad de instalar en su computadora más que un simple navegador web.

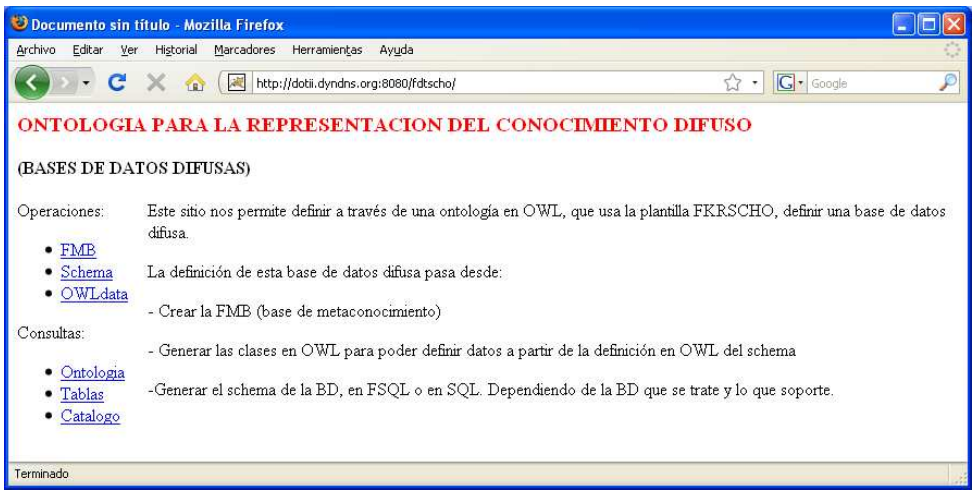


Figura 5.7: Imagen de la aplicación web para gestionar esquemas.

A continuación se describe brevemente la funcionalidad propuesta en el entorno:

- Generación de un esquema de BDD sobre un SGBD con capacidad de almacenamiento de difusos (es necesario tener definida la FMB en el SGBDR correspondiente). Será requisito que dicho esquema se encuentre previamente descrito como instancias de la *Ontología del Catálogo* en OWL. Los pasos para realizar esta operación pueden verse en la figura 5.8 y consisten en proporcionar al sistema el archivo en OWL y a continuación rellenar los campos de un formulario para conectar con la BD deseada (en este caso MySQL ©). A continuación se obtendría un informe del estado en el que se ha realizado la operación. En la imagen 5.9 puede observarse el resultado de realizar la operación sobre un SGBDR Oracle © y el resultado de la operación sobre el SGBDR a través de la herramienta *sqlplus*.

- Generación del script o conjunto de sentencias de definición del esquema de BDD deseado en lenguaje FSQL o SQL para su almacenamiento en forma de archivo. Esta opción se corresponde al botón *Ver Código* de la figura 5.8. El resultado de dicha ejecución tendrá el resultado mostrado en la figura 5.10 si la opción seleccionada es mostrar el código FSQL.

- Generación de la FMB sobre un SGBDs que no tengan esta extensión ya incorporada. Esta opción consiste en la generación automática de las es-

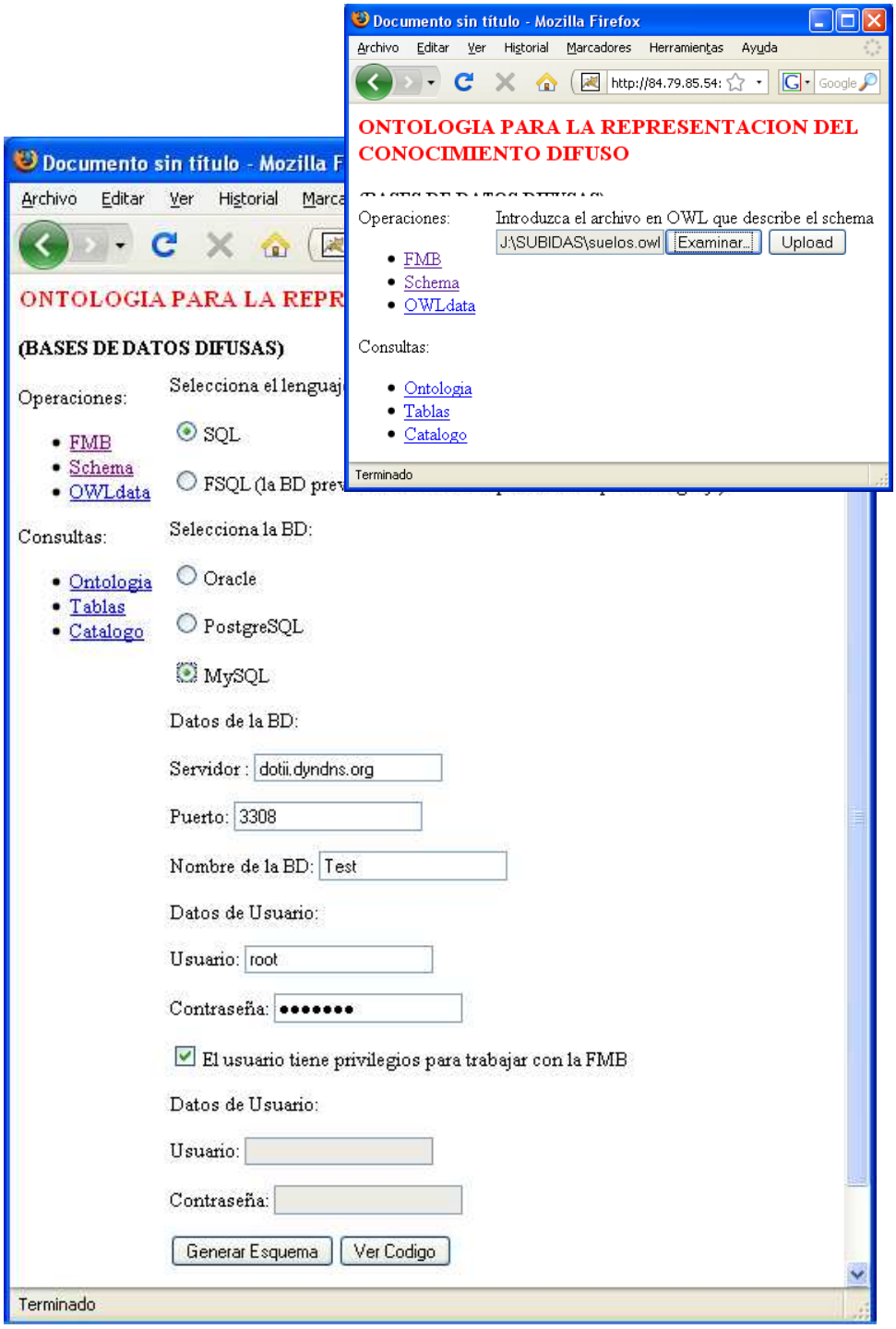


Figura 5.8: Imagen de la aplicación web para gestionar esquemas. Formulario de conexión para generar un esquema dado en OWL en una BD MySQL

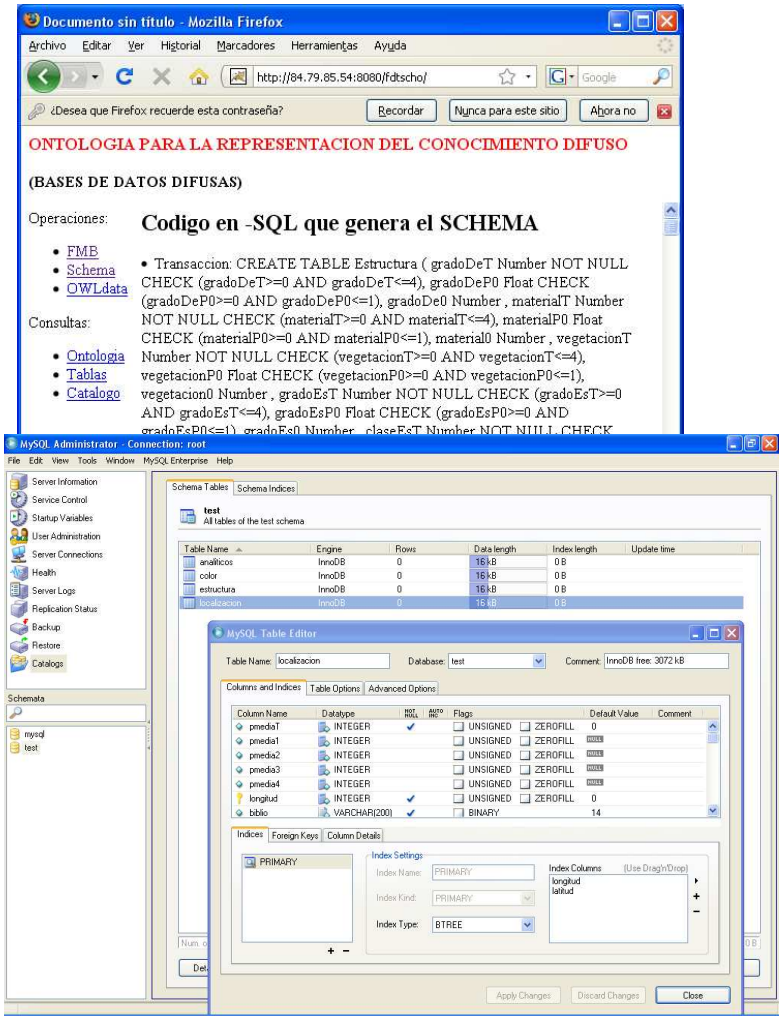


Figura 5.9: Imagen de la aplicación web para gestionar esquemas. Formulario de resultado tras ejecutar la generación de un Esquema de BD en MySQL ©

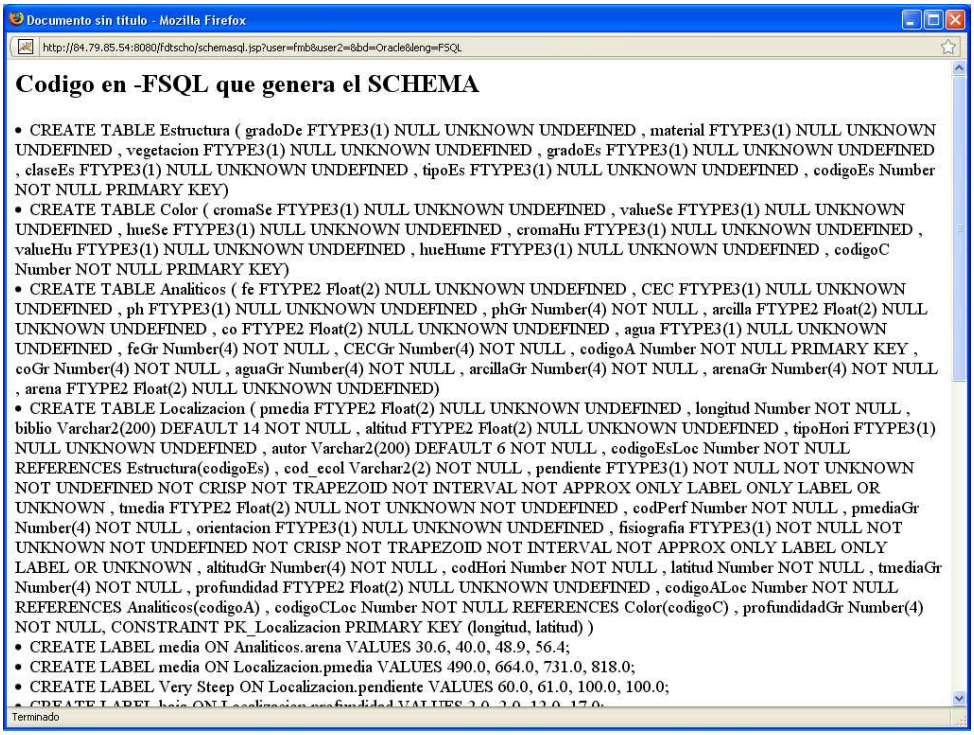


Figura 5.10: Imagen de la aplicación web para gestionar esquemas. Script de generación de esquemas en FSQL.

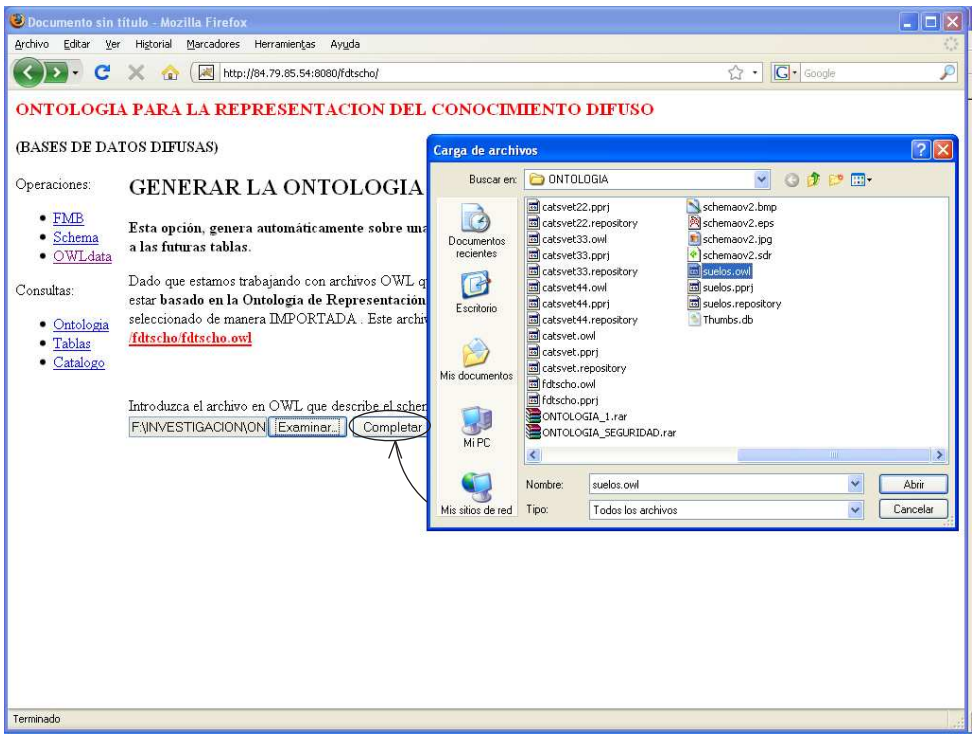


Figura 5.11: Selección del archivo de la ontología a generar

tructuras del catálogo para permitir almacenar datos difusos. El proceso de definición de esta estructura es similar al descrito anteriormente para definir un esquema, dado que habrá que insertar los datos en un formulario para establecer la conexión con el SGBDR correspondiente. En este caso, sin embargo, habrá que introducir las claves de administrador, puesto que se van a utilizar elementos del catálogo. Esta opción permite además la generación del script que genera dicha FMB en SQL.

- Generación de la *Ontología del Esquema* final, a partir de la definición inicial del esquema de BDD a través de la instanciación de la *Ontología del Catálogo*. Este proceso devolverá como resultado una nueva ontología que puede ser instanciada para insertar la información relativa a las tuplas. El proceso consistirá en la selección en primer lugar de la ontología que se desea generar, como se puede ver en la figura 5.11 y a continuación se obtiene la ontología resultante y un resumen de la conversión, como se puede ver en la figura 5.12.

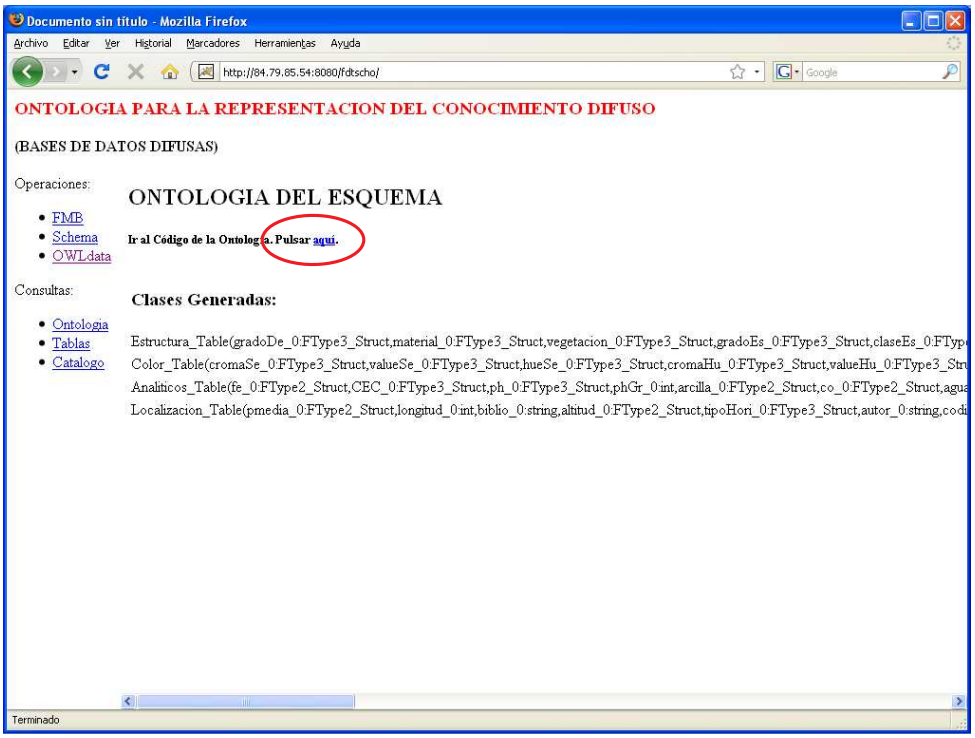


Figura 5.12: Resultado de la ontología generada

- Visualización de la *Ontología del Esquema* de forma intuitiva para el usuario, que no debe de necesitar entender ni SQL, ni OWL para conocer el conjunto de elementos que se encuentran definidos en la ontología. Esta opción permite ver de manera conjunta todos los elementos más representativos del esquema, dada la *Ontología del Catálogo* en la que está basada.

- Visualización del contenido de las tablas de la FMB.

- Visualización de las tablas de la BDD incluyendo la descripción de los elementos que componen a las mismas.

Cabe destacar que este entorno Web permite la conexión con los SGBDs descritos en la sección 5.3.2, definidos por su representatividad (Oracle ©, PostgreSQL © y MySQL ©). La extensión a otros sistemas de gestión, como puedan ser MS Access ©, MS SQLServer ©, SyBase ©, Paradox ©, etc. es inmediata dado que se han desarrollado las interfaces necesarias para facilitar la misma.

Por otro lado, esta herramienta no aporta ningún mecanismo para definir la ontología. No obstante, esto no supone ninguna desventaja en tanto que se podrá utilizar cualquier entorno de definición o manipulación de ontologías en OWL, que como se describe en el Anexo A, sección A.2.3.3 existen en gran número y aportan muy diversa funcionalidad.

Todos los módulos/procedimientos que gestionan la generación de ontologías y la traducción de los elementos de la ontología a la BD se han desarrollado utilizando el lenguaje de programación JAVA. Concretamente, la gestión de las ontologías se ha llevado a cabo a gracias al uso de las librerías para la gestión de OWL de JENA. A su vez, la herramienta web ha utilizado la tecnología JSP para poder llamar a los procedimientos que realizan las tareas anteriormente descritas.

5.3.4. Extensión de la Herramienta de Desarrollo de Ontologías: *Protégé*

Anteriormente se ha visto como podemos establecer la comunicación entre la *Ontología de Representación de Conocimiento Difuso* y diferentes SGBDRDs a través de un entorno Web. Sin embargo esta opción presenta la desventaja de que no aporta ningún mecanismo para definir la ontología. Además, sigue existiendo el problema de la complejidad en la definición de esquemas difusos, aunque en menor medida ya que ahora dichos esquemas son definidos a través de una ontología y no directamente sobre un SGBDRD concreto. Por ello se hace necesario el desarrollo de una herramienta que permita al usuario

definir y manipular de forma intuitiva la información difusa, además de poder tener acceso a ella del mismo modo.

Se propone así la extensión de una herramienta de gestión integral de ontologías como *Protégé*. La elección de esta herramienta concreta viene dada por los siguientes criterios:

- Permite utilizar dicho entorno para definir una ontología completamente sin necesidad de incluir ninguna extensión.
- Permite utilizar metadatos.
- Permite la definición y utilización de ontologías importadas y hace una gestión eficiente de las mismas.
- Tiene una interfaz visual cómoda e intuitiva.
- Está ámpliamente extendida, probada y aceptada entre la comunidad científica.
- Hace una representación del OWL genérica, comprensible y portable a otros entornos de gestión.
- Permite extender su entorno con otras implementaciones que hacen uso de la representación de ontologías que proponen. La extensión del entorno puede realizarse de muy diversas formas, todas bien documentadas, entre la que destacamos la utilización de extensiones (*plug-ins*) que es la utilizada en este trabajo.

Por todas estas razones se ha propuesto la triple extensión de la herramienta dado que ha de realizar tres tareas bien diferenciadas:

- La de definición y manipulación de la información de esquemas de bases de datos difusas y su correspondiente conexión y exportación a los SGBDRDs seleccionados.
- La de inserción de datos difusos (o clásicos) sobre la ontología y su posterior definición sobre los SGBDRDs seleccionados.
- La de ayuda a la generación de consultas difusas en FSQL a través del uso de la ontología.

La extensión de este entorno se ha realizado utilizando el lenguaje de programación JAVA, que es en el que está desarrollada la herramienta de *Protégé*.

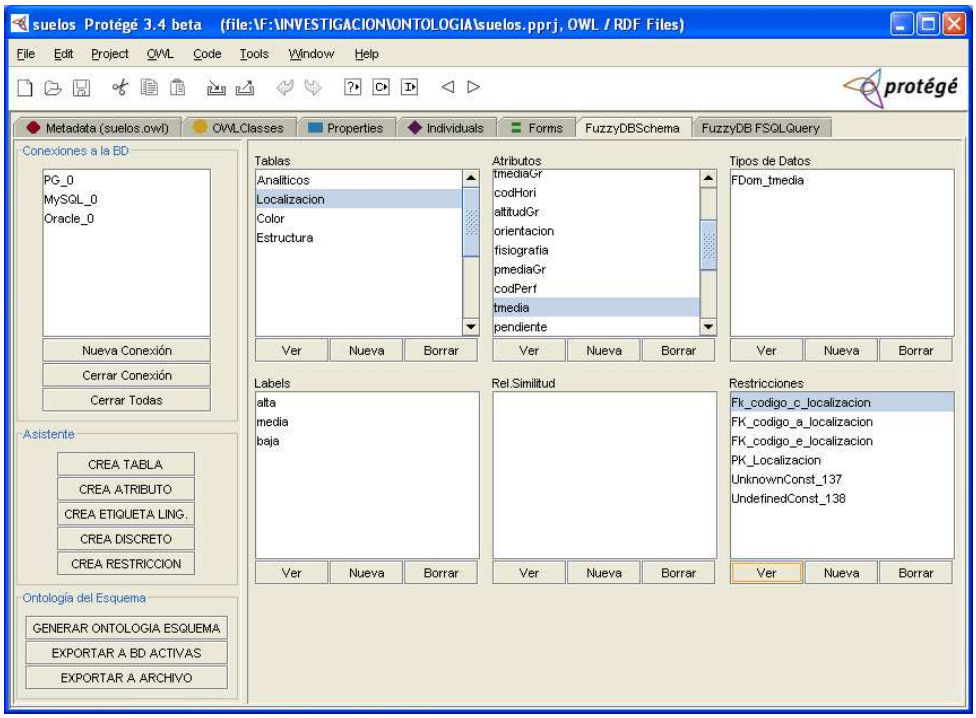


Figura 5.13: Imagen de la aplicación para gestionar esquemas añadida a la herramienta *Protégé*

5.3.4.1. Gestión de Esquemas de Datos Difusos en *Protégé*

Esta aplicación incluida en la herramienta *Protégé* tiene el aspecto visual de una pestaña más en la misma, tal y como podemos ver en la figura 5.13, y aporta la siguiente funcionalidad:

- Gestionar y visualizar los elementos más representativos de un Esquema de Base de Datos difusas: Tablas, Atributos, Tipos de Datos, Dominios Difusos, Restricciones, Etiquetas Lingüísticas (bajo un referencial ordenado o no ordenado). Estos elementos son visualizados a través de cuadros de texto cuyo contenido varía dinámicamente dependiendo de la selección de la estructura de datos que se realice. Estos cuadros, además de permitirnos visualizar las características de cada uno de los elementos seleccionados, nos permiten añadir nuevos elementos o eliminarlos. Esta opción es la correspondiente parte central de la figura 5.13.

- Gestionar la conexión con los SGBDRD y permitir el mantenimiento de

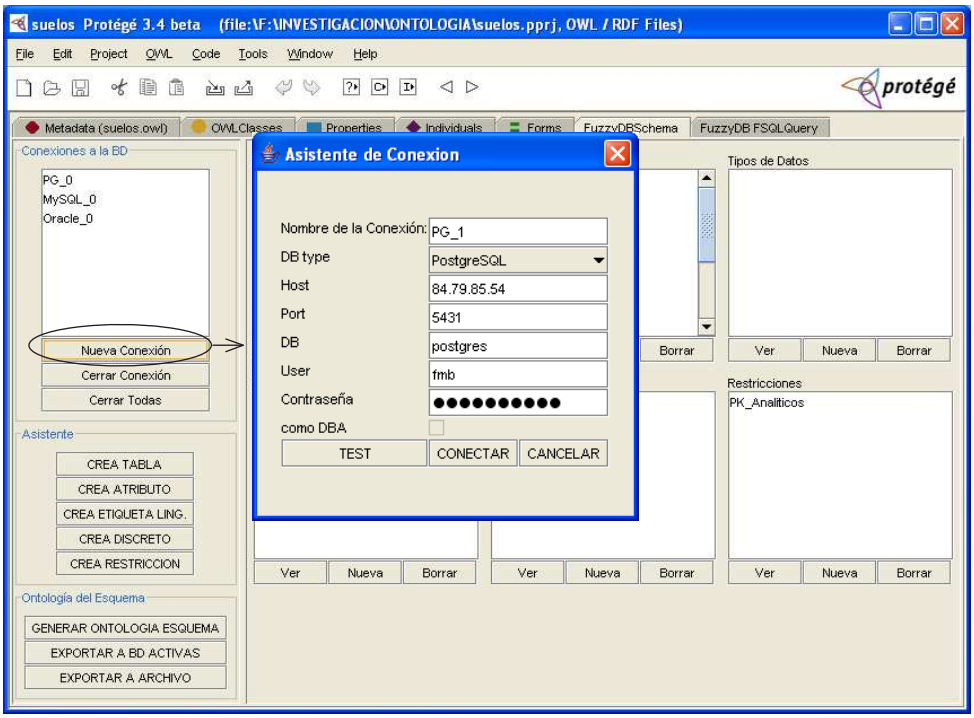


Figura 5.14: Imagen de la aplicación para gestionar las conexiones con el esquema en *Protégé*

dichas conexiones de forma simultánea para poder realizar operaciones en paralelo (opción correspondiente a la zona superior izquierda de la figura 5.14).

- Exportar la *Ontología del Esquema* a los SGBDRDs cuyas conexiones se encuentren establecidas. Este proceso será realizado de manera simultánea con todos los SGBDRDs que se encuentren abiertas. Esta opción se encuentra localizada en la zona inferior izquierda de la figura 5.13.

- Exportar el código fuente o script correspondiente a la creación del esquema definido a través de la ontología a un archivo. Dicho script puede obtenerse en FSQL o SQL dependiendo de la preferencia del usuario. Además dicho script contendrá las particularidades del SGBDR seleccionado. Esta opción se encuentra localizada en la zona inferior izquierda de la figura 5.13.

- Generar la *Ontología del Esquema* a partir de su definición previa sobre la *Ontología del Catálogo* (esta función también se aportaba de igual modo en el entorno web). Esta opción sería llamada a partir del botón situado en la zona inferior izquierda de la figura 5.13.

- Asistir al usuario en la generación de los elementos más representativos de un Esquema de BDD, mediante la utilización de asistentes guiados. Se encontraran asistentes para generar: tablas, atributos, etiquetas lingüísticas y relaciones de similitud, restricciones y dominios. Las llamadas a los asistentes se harían a través de los botones situados en la zona central izquierda de la figura 5.15.

5.3.4.2. Definición de Datos Difusos en *Protégé*

En la herramienta *Protégé* se ha incorporado una nueva pestaña, tal y como podemos ver en la figura 5.16, que permite la definición de datos imprecisos a través del uso de una *Ontología del Esquema* concreta. Dicha extensión a la herramienta dispone de las siguientes opciones:

- Visualizar el contenido de cada una de las relaciones o tablas de la *Ontología del Esquema* en forma de tabla dinámica.

- Cargar todas las instancias definidas en la *Ontología del Esquema,* a la tabla dinámica para facilitar su visualización.

- Permitir la inserción de nuevas tuplas a la ontología a través de un entorno tabular donde la información pueda ser definida de forma rápida. Para facilitar dicha tarea se describen, a través de una leyenda las características de definición de cada tipo de dato debajo de la tabla de inserción.

- Facilitar al usuario el acceso a los datos del dominio difuso. Dicha funcionalidad consiste básicamente en proporcionar de manera dinámica las etiquetas lingüísticas asociadas a los atributos difusos de tipo 2 o 3, en caso que las requieran, para definir los valores que componen la tupla.

- Gestionar la conexión los SGBDRD y permitir el mantenimiento de dichas conexiones de forma simultánea para poder realizar operaciones en paralelo. Dicha interfaz se encuentra localizada en la zona izquierda de la pantalla y es similar a la presentada en la figura 5.14).

- Exportar los datos definidos en *Ontología del Esquema* a los SGBDRDs cuyas conexiones se encuentren establecidas. Este proceso será realizado

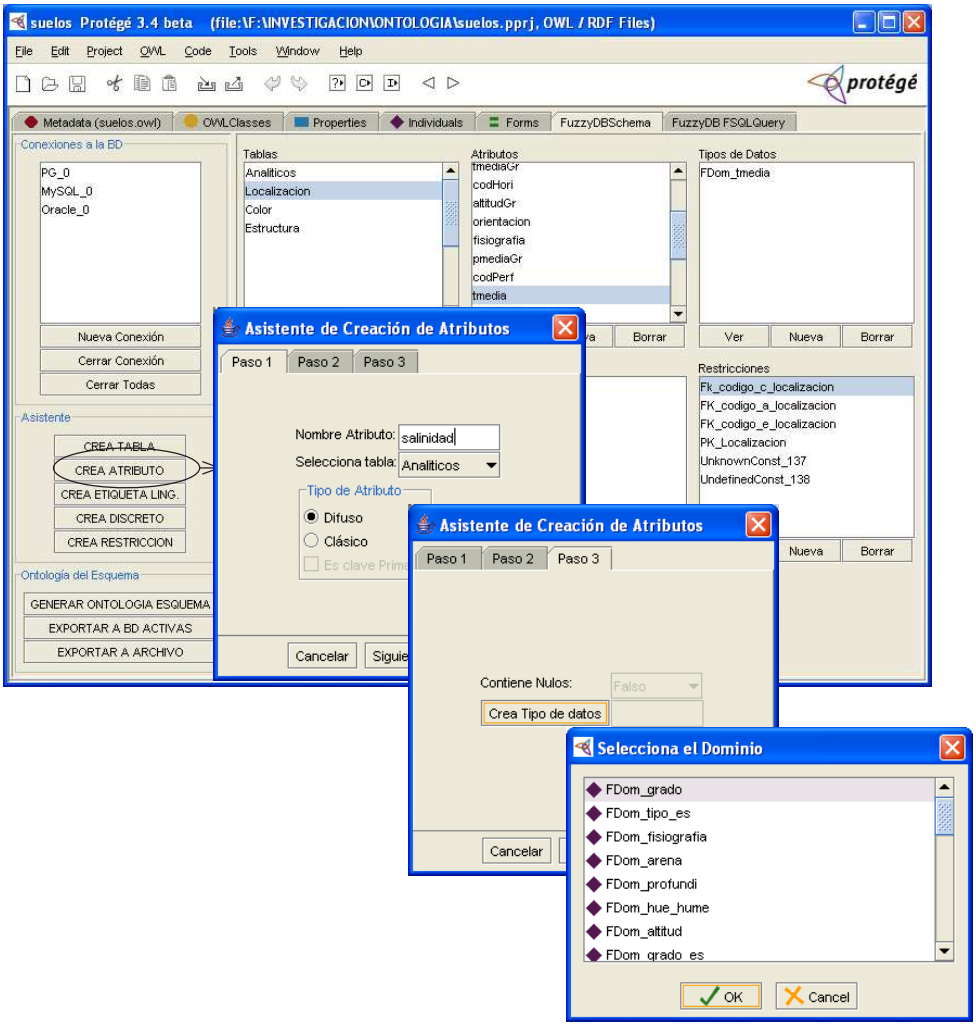


Figura 5.15: Imagen de la aplicación para abrir el asistente para la generación de un atributo en *Protégé*

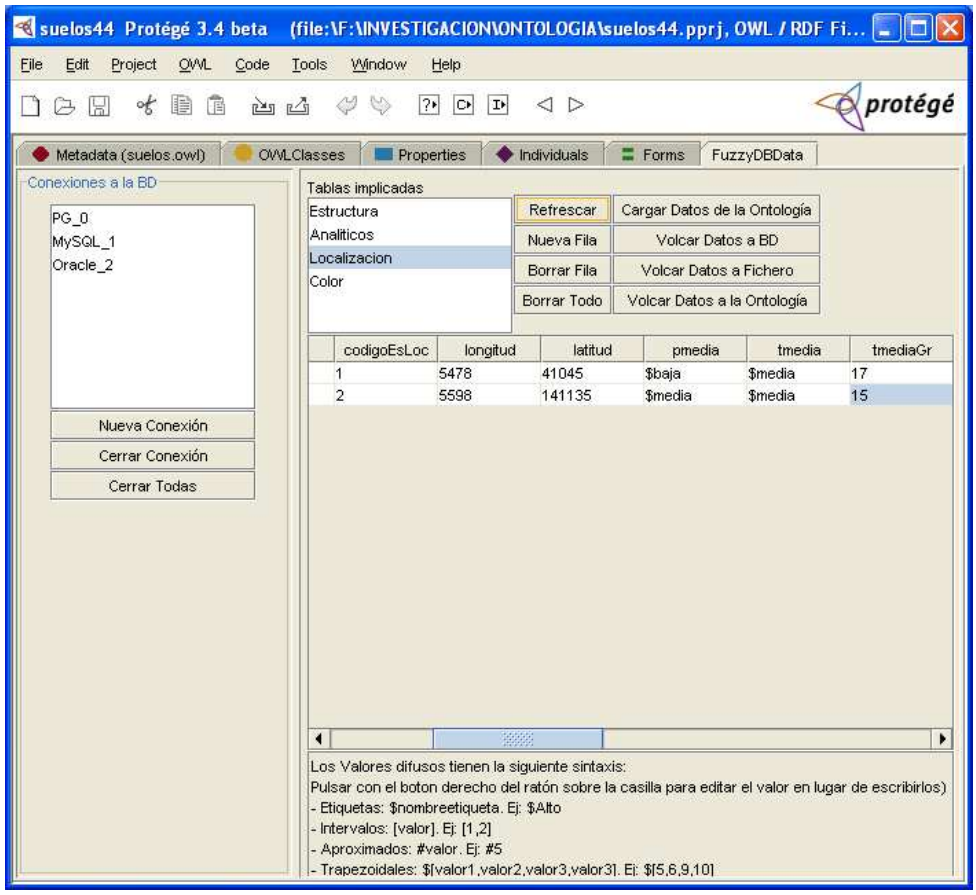


Figura 5.16: Imagen de la aplicación para gestionar esquemas añadida a la herramienta *Protégé*

de manera simultánea con todos los SGBDRDs que se encuentren abiertas.

- Exportar el código fuente o script correspondiente a la creación de las tuplas definidas a través de la ontología. Dicho script puede obtenerse en FSQL o SQL dependiendo de la preferencia del usuario. Además dicho script contendrá las particularidades del SGBDR seleccionado.

5.3.4.3. Generación de Consultas en FSQL en *Protégé*

La interfaz de consulta de Bases de Datos Difusas desarrollada a través del entorno de *Protégé* permite desarrollar una consulta sobre datos difusos en lenguaje FSQL a través de un entorno guiado e intuitivo. Para ello proporciona al usuario la información necesaria para realizar la consulta en función de los datos que dispone gracias la ontología que describe el esquema de BDD sobre el que se desea consultar.

Esta extensión de *Protégé* tiene un desarrollo similar a las anteriores, tal y como podemos ver en la figura 5.17, y aporta la funcionalidad descrita a continuación:

- Realiza consultas clásicas o difusas.
- Consulta sobre un número cualquiera de relaciones.
- Incluye un número indeterminado de condiciones.
- Permite la posibilidad de negar una condición.
- Identifica qué tipo de comparadores existen en función del atributo que se trate (sea clásico o difuso).
- Permite realizar comparación de atributos o valores.
- Permite realizar comparaciones con valores difusos incluidos en los dominios difusos.
- Permite la definición de nuevos valores difusos con los que realizar una comparación.
- Permite asignar un grado de certeza a la condición que utilice atributos difusos.
- Permite negar una condición.

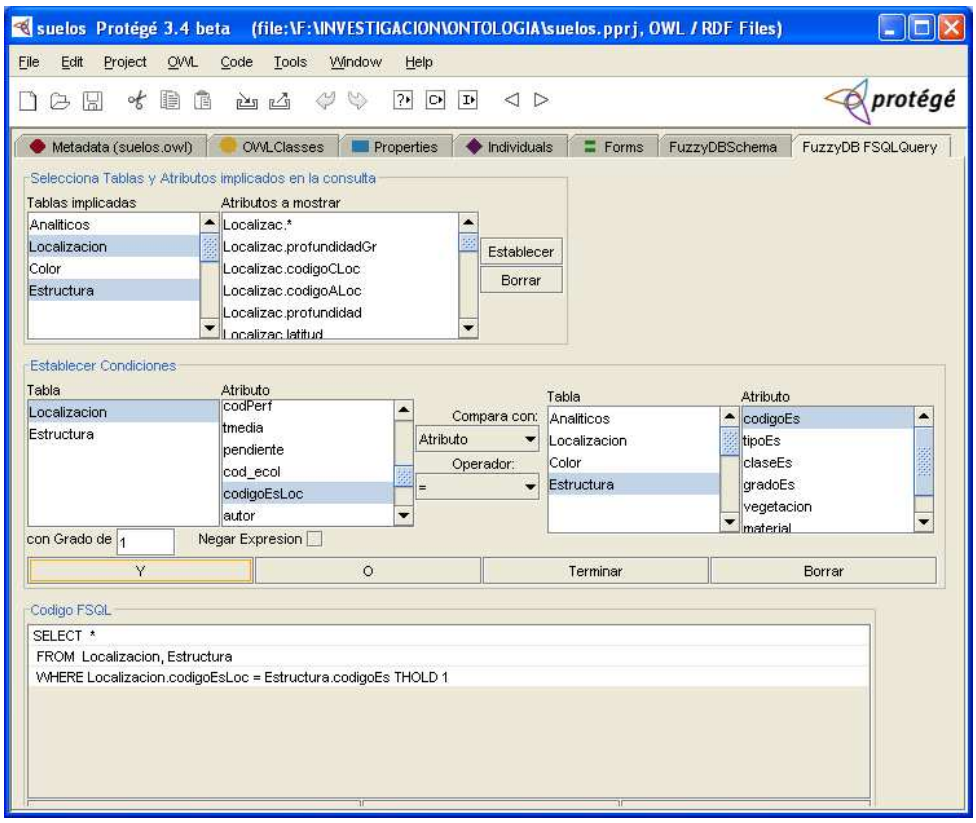


Figura 5.17: Interfaz de generación de consultas difusas en FSQL sobre la herramienta *Protégé*. Establece una comparación entre atributos

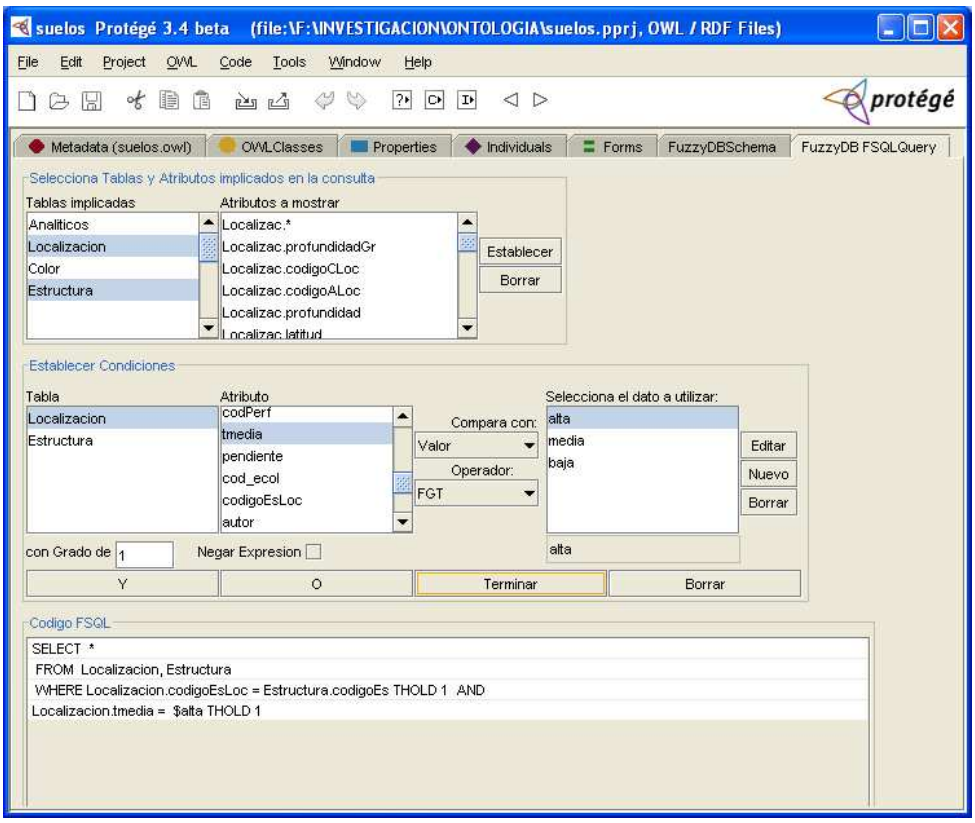


Figura 5.18: Interfaz de generación de consultas difusas en FSQL sobre la herramienta *Protégé*. Establece una comparación con Valor Difuso.

La consulta generada será traducida al lenguaje FSQL y posteriormente a través del botón *Ejecutar* podrá ser lanzada al SGBDRD que interprete dicho lenguaje a través de la utilización de la herramienta FuzzyQuery [Bla02b].

En la figura 5.17 se visualiza cómo puede establecerse una condición entre atributos, y en la figura 5.18 se muestra una comparación difusa entre un atributo y un valor del dominio del atributo.

5.4. Casos de Uso de la Arquitectura

La definición de esta arquitectura para realizar la conexión semántica entre la definición de una Base de Datos Relacional Difusa y una ontología permite llevar a cabo la realización de múltiples operaciones que sin su combinación

no hubiesen sido posibles.

A continuación se van a detallar todas las aplicaciones o casos de uso que dicho modelo de representación de datos presenta, diferenciando el tipo de datos que se representa y la operación que realizan.

Además se detallará qué casos de uso están implementados y cuales de ellos no, especificando el proceso que se requiere llevar a cabo para dicha implementación. Los casos de uso que no se encuentran implementados en este trabajo de tesis, se establecen como trabajos futuros, tal y como puede verse en el capítulo siguiente.

5.4.1. Definición de Datos. Creación de Esquemas

5.4.1.1. Interacción con otros Esquemas de BD

El trabajo de definición de un Esquema de BD Difusas, es el más importante en el proceso de definición de un sistema de información de estas características y se obtiene siguiendo los pasos descritos en la arquitectura. Dicho Esquema, una vez seguidos estos pasos, se obtendrá en dos formatos diferentes, uno como esquema SQL o FSQL dependiendo de la vista en que lo queramos obtener, y otro como ontología definida en el lenguaje OWL (comprensible para la Web).

Gracias a este doble formato las posibilidades de definición e interacción de un esquema de bases de datos con el entorno se multiplican de forma exponencial, ya que la representación del esquema en forma de ontología se ve libre de dependencias con los SGBDR en donde se pudieran hallar almacenados. A continuación se detallan los diferentes casos que se pueden presentar entre los diferentes SGBDRD gracias a esta propuesta.

Definir un Esquema de BDD en SGBDR Heterogéneos Es posible, una vez definida la *Ontología del Esquema de BD*, que dicho esquema sea definido en cualquier SGBDR con capacidades Difusas, siendo indiferente el SGBDR comercial de que se trate, ya que las particularidades del mismo son transparentes al usuario. La figura 5.19 ilustra dicha aplicación. Para llevar ésto a cabo es necesaria la definición de la ontología y posteriormente su declaración en el SGBDR tal y como se describe en este capítulo.

Esta operación es la más sencilla y habitual para definir cualquier BDD y uno de los principales objetivos alcanzados en este trabajo.

Exportar un Esquema de BDD a cualquier SGBDR Es posible, gracias a la utilización de la *Ontología del Esquema* el portar esquemas de BD entre diferentes SGBDR. Dicho proceso implica la exportación del esquema

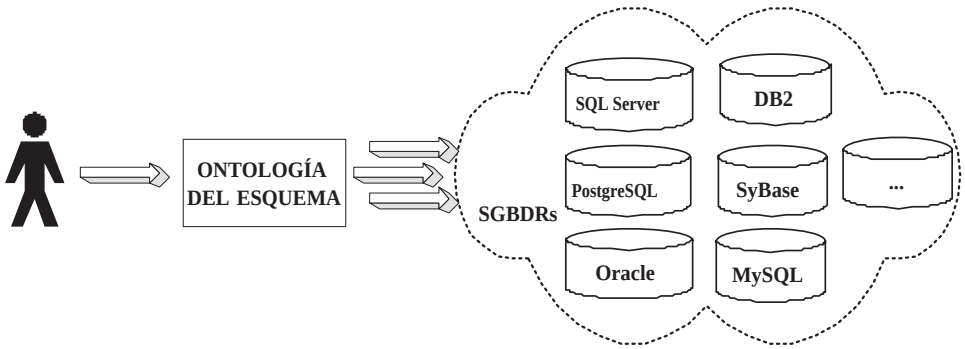


Figura 5.19: Definición de Esquemas de BDD en SGBDR Heterogéneos

definido en un SGBDR concreto al formato de instancias de la *Ontología del Catálogo* descrito en la arquitectura y a continuación el proceso de definición del mismo en cualquier otro SGBDRD distinto. Ni siquiera haría falta su traducción a forma de *Ontología del Esquema*. La figura 5.20 muestra el proceso de forma gráfica.

Este proceso es habitual en las migraciones de BD, a otros sistemas más actuales o simplemente a otros SGBDS comerciales. Esta operación requiere implementar el proceso de convertir una BDD a *Ontología del Catálogo* (funcionalidad que actualmente habría que hacer de forma manual y que se deja planteada como un trabajo futuro).

Unificar Esquemas Complementarios Esta aplicación consiste en la posibilidad de Unificar Esquemas definidos o distribuidos en diferentes SGBDs y cuya información representada no coincide en contenido semántico puesto que representan realidades diferentes. En la figura 5.21 puede observarse cómo el esquema final será una unión de los esquemas iniciales.

Este proceso suele ser utilizado para aunar BDD diferentes en un mismo entorno. Se lleva a cabo a través del uso de una ontología que contiene definidos todos los conceptos de las BDDs de origen. Dicha ontología unirá las representaciones de las BDDs en forma de instancias de la *Ontología del Catálogo*. A partir de la ontología que une los esquemas, se genera bien otro SGBDRD con la estructura final o bien, se trabaja directamente con la misma.

Unificar Esquemas Compatibles Esta última opción es la más compleja de las anteriores y la única cuya puesta en marcha no sería inmediata. Se trata de combinar dos SGBDRD que contiene conceptos semánticamente similares.

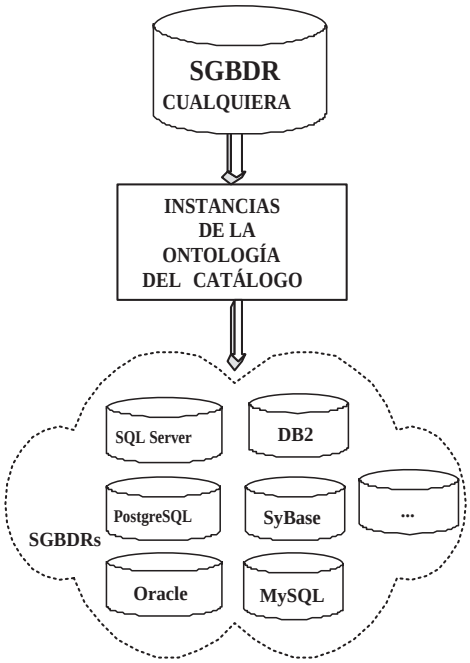


Figura 5.20: Exportación de un Esquema de BDD a cualquier SGBDR

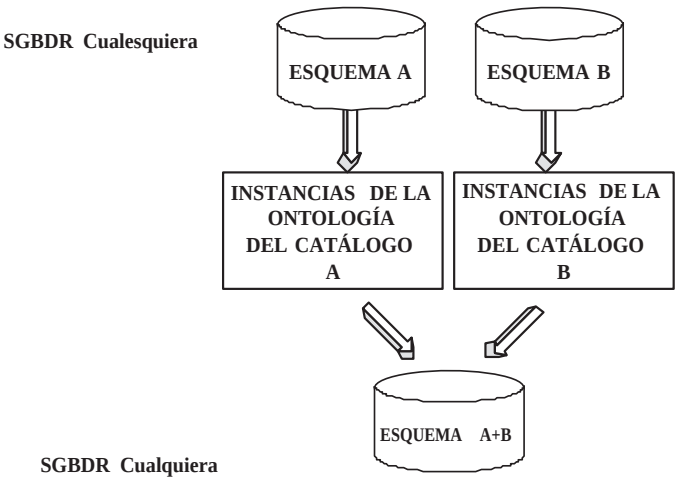


Figura 5.21: Unificar Esquemas Complementarios

Para llevar a cabo esta unificación se requiere de la utilización de operaciones sobre ontologías que permitan la identificación de correspondencias entre conceptos, tal como se ve en la figura 5.22. Este tipo de operaciones denominados genéricamente de mezcla, búsqueda de correspondencias o alineación, se han descrito en el apartado A.2.4 y son complejas y costosas, debido a los múltiples parámetros que han de ser tenidos en cuenta para identificar y relacionar conceptos representados en esquemas diferentes que varían desde conflictos de tipos de datos, de nombres, de semántica, etc. (véase [Ma06] para conocer más detalles sobre la problemática del unificado de esquemas).

Este proceso suele usarse para compartir información entre SGBDRD que se encuentren distribuidas y compartan información o bien entre SGBDRD diferentes que contengan información en común. Un ejemplo de uso podría consistir en disponer de una *BDD Universitaria* con información académica de sus alumnos, y otra *BDD de Jugadores de Fútbol*, con información acerca de las características y resultados de cada uno los jugadores. Una combinación de ambas bases de datos permitiría extraer información acerca, por ejemplo, del rendimiento académico del jugador en función de su valía como deportista, o sacar cualquier otra conclusión a través del uso de técnicas de minería de datos.

Para realizar esta operación se necesita obtener la *Ontología del Esquema* de cada BDD y aplicar las operaciones anteriormente mencionadas sobre los conceptos representados en las mismas. Este caso de uso implica el desarrollo de un gran número de aspectos teóricos y se incluye como uno de los trabajos futuros de esta tesis.

5.4.1.2. Interacción con Esquemas del Entorno

La información contenida en un esquema de BD incorpora todo el conocimiento necesario para describir una parte de la realidad. Es por esto que se considera comparable un esquema de BD, a cualquier ontología de representación de un dominio, a cualquier modelos de datos orientados a objetos, esquema en XML, folksonomía, u otro modo de representación de la realidad que sea computacionalmente comprensible (léase apartado 2.3).

De esta forma, un Esquema de BDD no solamente interacciona con otros Esquemas de BDDs a través de los SGBDRD, sino que también puede hacerlo con otras representaciones encontradas en otros entornos, compartiendo la riqueza semántica de los conceptos que representa. Para ello la representación del Esquema de BDD en forma de ontología (en OWL) es un arma excelente, ya que es formato aceptado por la amplia mayoría de la comunidad que representa conocimiento en la Web (lugar dónde se puede encontrar la mayoría de las representaciones de datos anteriormente expuestas).

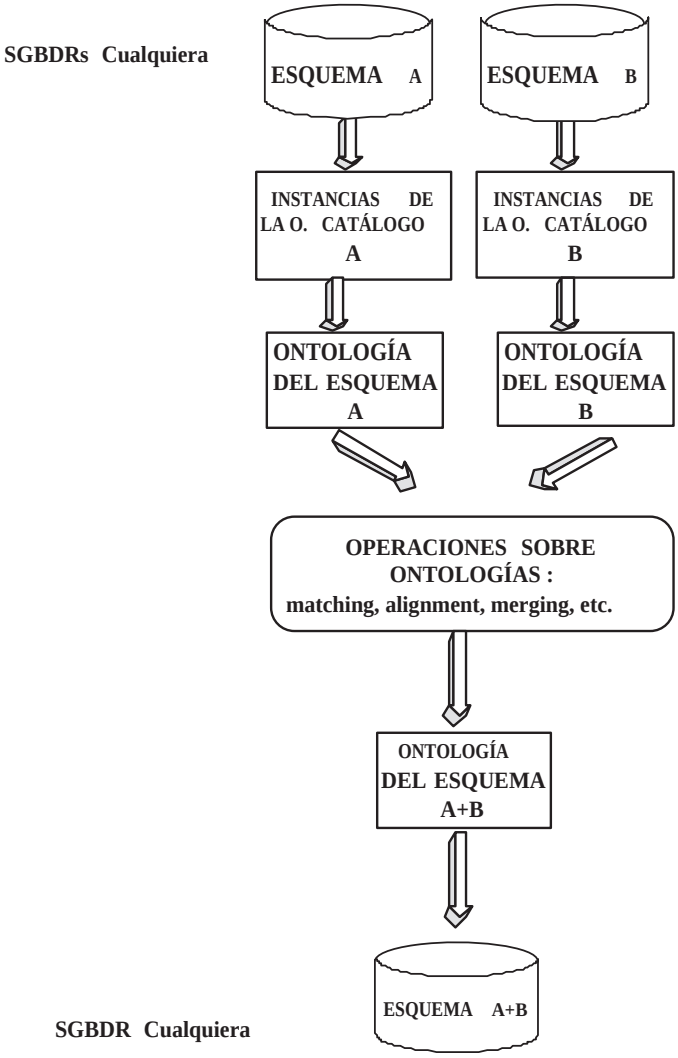


Figura 5.22: Unificar Esquemas Compatibles

Sin embargo, dada la naturaleza heterogénea de las estructuras de información que existen y con las que se puede interaccionar, las operaciones que pueden realizarse con nuestra propuesta de representación están supeditadas a la incorporación de herramientas que permitan la traducción de los diferentes formatos a uno común comprensible por todas para que puedan interaccionar. Una solución puede consistir en el uso de OWL dada la amplia aceptación que tiene este lenguaje de representación. Así, las operaciones que pueden realizarse con la *Ontología del Esquema* de un BDD y el resto del entorno son muy similares a las descritas anteriormente para los SGBDRDs, como vemos a continuación:

Publicación de un Esquema de BDD Esta operación consiste en aportar al entorno, fundamentalmente Web, un nuevo tipo de información que pueda enriquecer a la comunidad: un esquema de BDD. Dichos esquemas proporcionan una representación de la realidad que puede ser procesable computacionalmente y por consiguiente usada por agentes, usuarios u otros sistemas de gestión del conocimiento para, por ejemplo, acceder a través de herramientas de consulta a la información deseada gracias a que la estructura de la información está disponible. La figura 5.23 trata de ilustrar el proceso que lleva esta publicación del esquema de BDD.

Definir y publicar una BDD utilizando la *Ontología del Esquema* es un proceso inmediato (operación descrita en [Bla07]). Sin embargo, traducir a otros formatos requeriría la elaboración de los traductores pertinentes (ámbito que no compete a este trabajo de tesis).

Definición de Esquemas de BDD a partir de un Esquema Cualquiera Esta operación trata de incorporar a un modelo de BD Relacional Difuso un esquema no relacional. Para realizar este proceso, sería necesario la generación de la *Ontología del Esquema* a partir de los conceptos obtenidos de cualquier otro tipo de representación. En la figura 5.24 se muestra el sentido del flujo de información. Una vez obtenida la *Ontología del Esquema* sería traducida a instancias de la *Ontología del Catálogo* y a continuación al SGBDRD correspondiente.

Esta herramienta puede ser muy útil en aquellos casos en los que la cantidad de información a representar sea demasiado grande, y se determine que un SGBDR haría una gestión mas eficiente de la misma.

Esta operación requiere la implementación de traductores que representen la estructura de datos pertinente en una *Ontología del Esquema* adecuada. Su desarrollo se pospondrá para trabajos futuros.

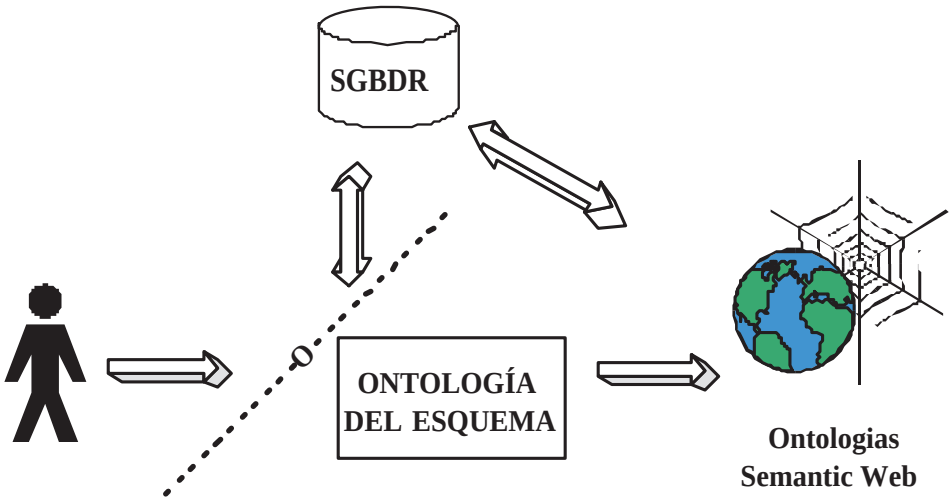


Figura 5.23: Incorporar a la Web un Esquema de BDD

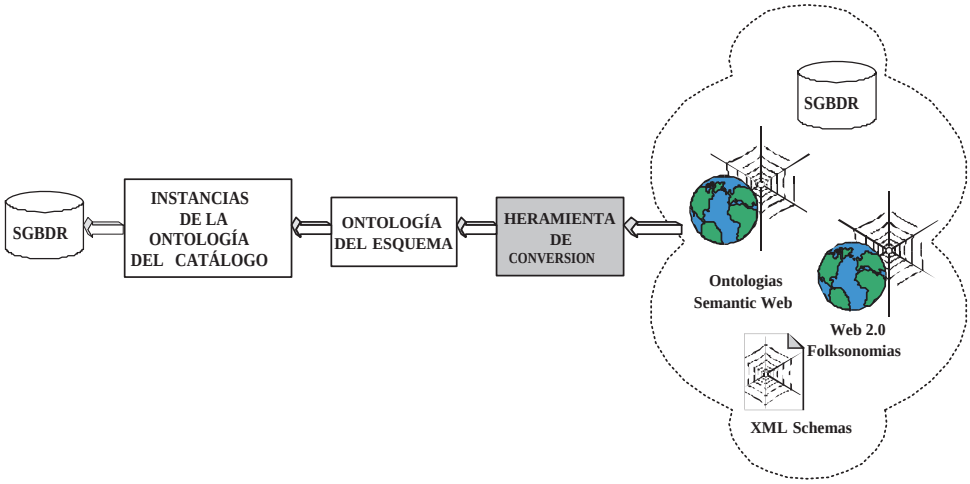


Figura 5.24: Definición de un Esquema de BDD a partir de cualquier tipo de Esquema

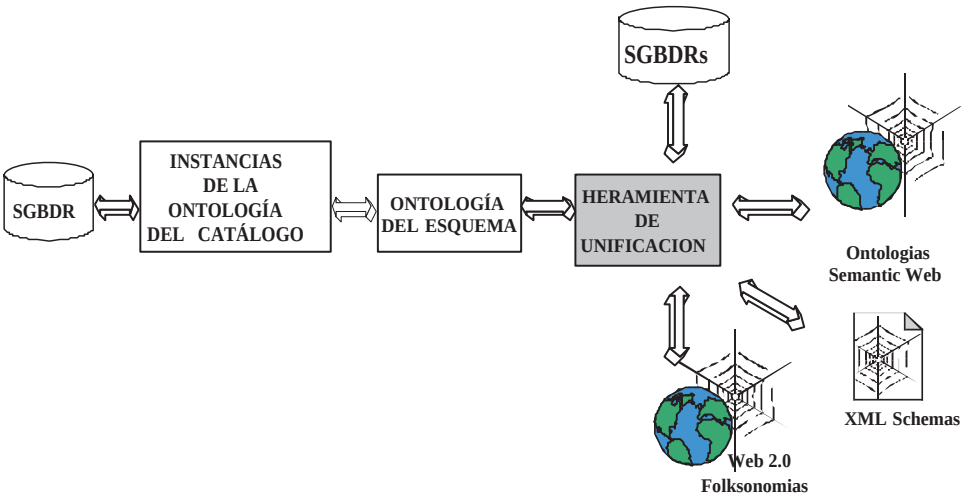


Figura 5.25: Combinación de Fuentes Heterogéneas

Combinación de Esquemas a partir de Fuentes Heterogéneos Se plantea la posibilidad de generar un Esquema a partir de esquemas provenientes de fuentes heterogéneas, tal y como puede verse en la figura 5.25. Para realizar esta operación es necesario además de la traducción de los diferentes esquemas a un lenguaje común, como podría ser el OWL, un mecanismo de interpretación de conceptos, tal y como ocurría en el proceso de *Unificar Esquemas Compatibles* descrito en el subapartado anterior. Dicho mecanismo debe identificar los conceptos representados en los diferentes esquemas, resolviendo los posibles conflictos que puedan darse. Este proceso es costoso y complejo, pero en la actualidad es un problema muy estudiado en el ámbito de la combinación de ontologías.

La utilidad de esta operación reside en la posibilidad de obtener una ontología nueva a partir de diversas fuentes de información con independencia de la estructura que las describa. Esta nueva representación en forma de *Ontología del Esquema* de una BDRD permite además de la gestión eficiente de los datos que forman parte de los mismos en un SGBDR, compartir información entre fuentes de datos heterogéneas a través de una única representación de datos.

La posibilidad de representar información imprecisa aporta flexibilidad a la hora de resolver conflictos en el proceso de unificación de conceptos. Sin embargo, dada la complejidad de esta tarea se plantea su desarrollo como un trabajo futuro.

5.4.2. manipulación de Datos

En las BBDD, tanto la estructura de la información como la información en sí cobran la misma importancia. A pesar de que el valor de la información se obtiene a través de los datos almacenados, la obtención del conocimiento posterior dependerá de la correcta organización de los mismos. De este modo, la representación de los datos en la BDD juega un papel fundamental.

A pesar de que una ontología no es el mecanismo más adecuado para manipular la información, dado que los SGBDs hacen una gestión más eficiente de la misma, las ontologías permiten la definición de dicha información en forma de instancias. Si se pretende realizar las mismas operaciones de manipulación de datos sobre las ontologías que sobre un SGBDR entonces se requiere una descripción del esquema de datos fácilmente legible para el usuario, aislando al mismo de cualquier detalle de implementación donde dicha información este almacenada, para permitirle realizar las tareas de inserción y consulta. Es por esto que una *Ontología del Esquema* que represente la información descrita en un SGBDRD aporta una gran utilidad para desarrollar esta tarea.

5.4.2.1. Inserción de Datos

A continuación se analizan las operaciones que aporta la *Ontología del Esquema* aporta a la hora de manipular la información imprecisa almacenada en un SGBDRD. Estas operaciones son las referentes al proceso de inserción de datos (*INSERT* en SQL) en los SGBDRDs .

Definición de Datos Esta operación consiste en utilizar la ontología como plataforma para definir una nueva tupla de datos. Dicha tupla de datos será definida mediante la instanciación de la *Ontología del Esquema* correspondiente. Una vez definida dicha información, se hará el traspaso de la misma al SGBDRD deseado utilizando las herramientas que proporciona la arquitectura.

Esta operación suele utilizarse cuando el número de datos a insertar no es muy elevado (para lo cual se utilizarían scripts). También puede resultar interesante si se mantienen copias simultáneas de la BDD en diferentes plataformas SGBDs, tal y como se mostraba en la figura 5.19.

Esta operación si ha sido implementada en este trabajo.

Exportación de Datos a otros SGBDS Al igual que ocurría con los esquemas, véase la figura 5.20, los datos también pueden ser portados de un SGBDRD a otro diferente. La migración de la información contenida en la misma a través del uso de la ontología puede ahorrar mucho esfuerzo en el proceso

puesto que evita la elaboración de ficheros que busquen las incompatibilidades entre los sistemas origen y destino. Para llevarse a cabo es requisito que el sistema pueda generar una ontología a partir de un SGBDRD (competencia no incluida en este trabajo).

Inserción de datos a partir de fuentes heterogéneas Esta operación consiste en incluir datos a un SGBD a partir de otras fuentes, posiblemente encontradas en la Web. Esta operación consistiría en primer lugar en la adaptación del esquema donde están los datos en origen al modelo de *Ontología del Esquema* de la arquitectura. A continuación la inserción de datos en el mismo sería perfectamente compatible.

Los inconvenientes que se pueden encontrar a la hora de realizar esta operación residen en la adaptación de un Esquema cualquiera al modelo de *Ontología del Esquema* adoptado en nuestro sistema. Dicha operación puede tener dos casuísticas:

- Que la BDD u *Ontología del Esquema* correspondiente no exista previamente. Para lo cual no existiría ningún problema, puesto que el proceso consistiría simplemente en crear el Esquema en primer lugar, a partir de dicha fuente, tal y como se describe en el subapartado anterior *Definición de Esquemas de BDD a partir de un Esquema Cualquiera* (figura 5.24), y a continuación transportar los datos entre plataformas.
- Que la BDD u *Ontología del Esquema* correspondiente exista previamente. Este proceso requiere la operación de unificación de esquemas, descrita en el subapartado, *Combinación de Esquemas a partir de Fuentes Heterogéneos* (figura 5.25).

Ninguna de estas dos alternativas se ha desarrollado en este trabajo de tesis, y queda planteada como trabajo futuro.

5.4.2.2. Consulta

Consulta a un único SGBDRD Es la operación de consulta común, que en lugar de realizarse directamente a través del SGBDR utilizando el lenguaje FSQL, se realiza a través de la ontología (véase figura 5.26). Será esta última la que proporcione el código en OWL necesario para que la fase correspondiente de la *Arquitectura de Consulta* lo traduzca en dicho lenguaje FSQL y lo lance al SGBDRD correspondiente.

Esta operación realizada a través de la ontología es independiente de las restricciones que imponga el SGBDRD a la hora de realizar la consulta, ya que dicha consulta se genera utilizando el código OWL de la *Ontología del Esquema*

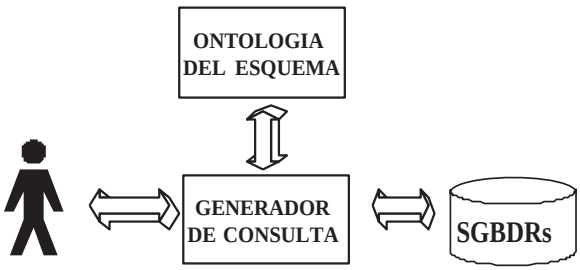


Figura 5.26: Consulta a un SBDRD único

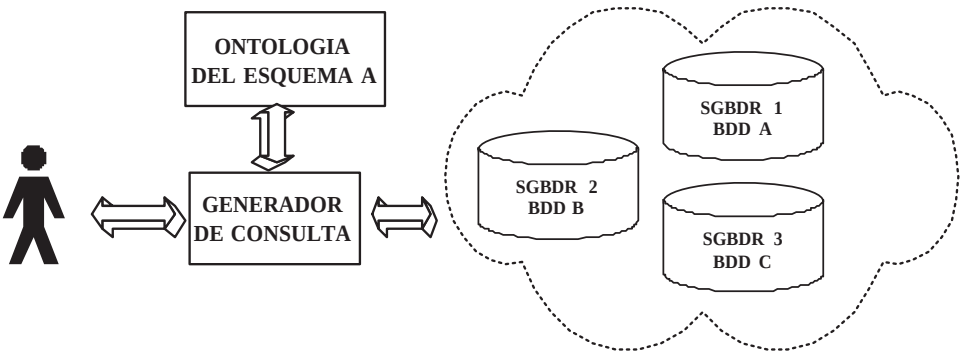


Figura 5.27: Consulta a SGBDRD con el mismo Esquema

y a través de una aplicación de traducción será el *Interprete o Adaptador del SGBD* el encargado de resolver estas cuestiones.

Esta tarea se resuelve en este trabajo de tesis.

Consulta entre SGBDRD con el mismo Esquema Esta operación se daría en entornos cuyos datos están repartidos en diferentes plataformas SGBDRD pero que comparten el mismo Esquema de Información, tal como se ve en la figura 5.27. La consulta a través de la *Ontología del Esquema* permitiría conectar con todos los sistemas cuya información se desea obtener y devolver a usuario de forma transparente el resultado en un único formato.

Este caso sería muy fácilmente tratable puesto que su desarrollo no tiene ningún coste computacional a excepción de la unión del resultado dado por la consulta en los sistemas particulares donde ha sido lanzada.

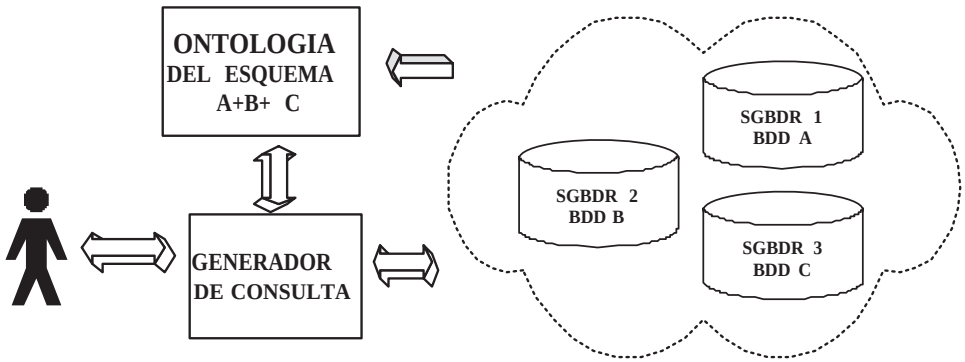


Figura 5.28: Consulta a SGBDRD con Esquemas Complementarios

Consulta entre SGBDRD con Esquemas Complementarios Esta casuística se daría en consultas que desean buscar información sobre sistemas que contienen información complementaria. Este es el mismo caso expuesto en el subapartado anterior *Unificar Esquemas Complementarios* ilustrado en la figura 5.21. En dicho caso, únicamente sería necesario la generación de la consulta utilizando la *Ontología del Esquema* que engloba cada uno de los esquemas involucrados. En la figura 5.28 se ilustra dicha operación.

Consulta entre SGBDRD con Esquemas Compatibles En esta ocasión se desea consultar información sobre varios esquemas que contienen información común o similar. Este hecho obliga a desarrollar una ontología unificada donde existan conceptos que comparten el mismo significado (tal y como se vio en el apartado *Unificar Esquemas Compatibles* ilustrado en la figura 5.22 y en este caso en la figura 5.29. Una vez los esquemas han sido adaptados y generada una única *Ontología del Esquema* que aune a las participantes en la consulta, la consulta podrá ser realizada y el resultado devuelto al usuario final.

Dicha operatividad supone un gran costo dada la complejidad y naturaleza del problema, tal y como especificamos en el anterior apartado y se plantea como un trabajo a realizar en un futuro.

Consulta al Entorno de Esquemas Heterogéneos Esta operación propone la consulta de información a cualquier esquema fuente de datos, sin importar que sea relacional o contenga datos difusos. En este caso, y tal y como se describió para la *Combinación de Esquemas a partir de Fuentes Heterogéneas* descrita en la figura 5.25 se considera necesaria, una primera adaptación a

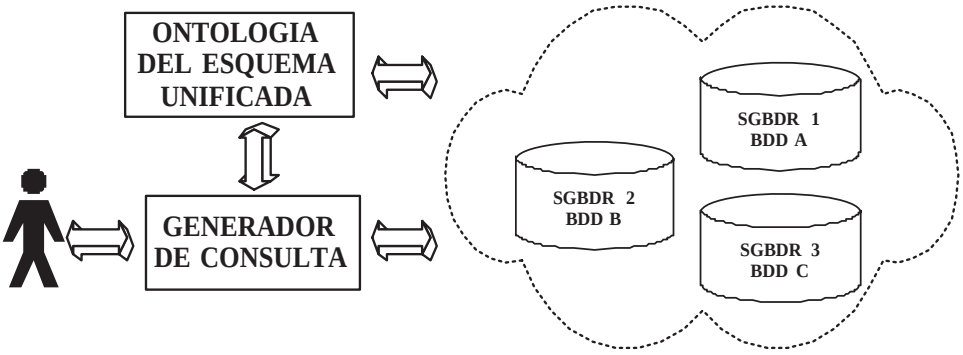


Figura 5.29: Consulta a SGBDRD con Esquemas Compatibles

ontología en OWL de las distintas fuentes y una posterior generación de una *Ontología del Esquema* común en las que se hallen identificados todos los elementos comunes encontrados en las mismas y las correspondencias entre los diferentes elementos comunes. Véase la figura 5.30 para ilustrar dicha operación.

Esta propuesta es la más completa de todas pero la puesta en marcha de este problema sería comparable a la construcción de un buscador Web, y sería un problema inabordable para ser resuelto en el ámbito de este trabajo de tesis.

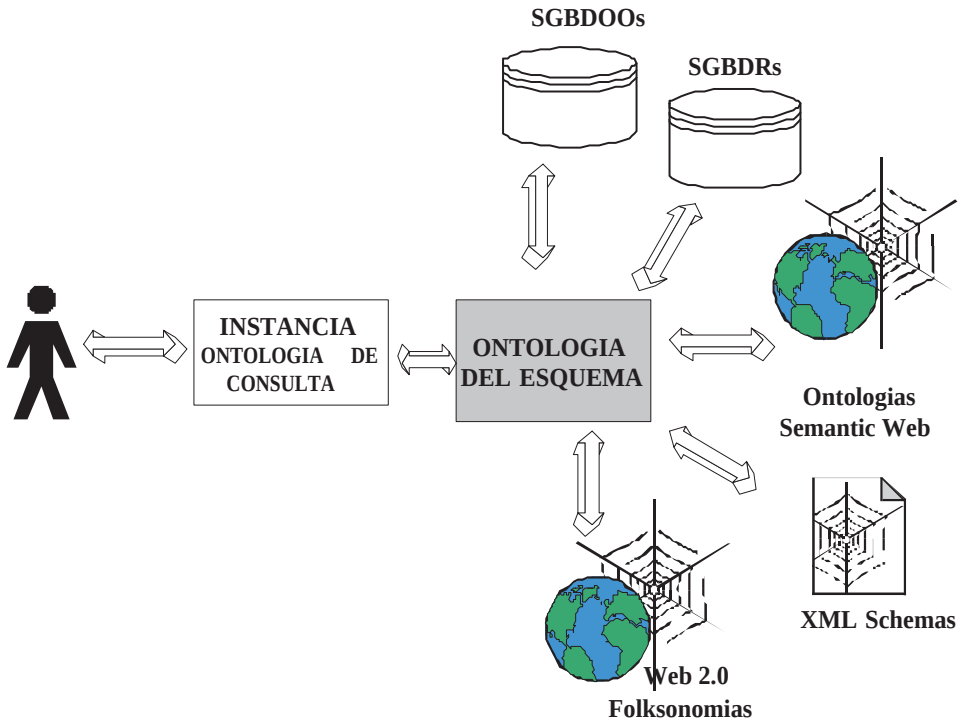


Figura 5.30: Consulta a SGBDRD con Esquemas Heterogéneos

Capítulo 6

Conclusiones y Trabajos Futuros

6.1. Conclusiones

Extensión de los Sistemas de Gestión de Bases de Datos Relacionales

Los Sistemas de Gestión de Bases de Datos en la actualidad están siendo extendidos continuamente. El añadir la gestión de información imprecisa a los mismos no supone un gran coste, tal y como se ha probado en numerosos modelos. Sin embargo, extender el SGBDR para incrementar el número de operaciones con dicha información imprecisa no es un proceso tan común.

En este trabajo se ha propuesto de manera teórica una *Arquitectura Multipropósito* para la unificación de extensiones realizadas a SGBDR Difusas. Dicha arquitectura se ha basado concretamente en dos extensiones: la que permite deducir sobre información difusa, y la que añade técnicas de minería de datos sobre un SGBD difuso. Las extensiones planteadas con este propósito habían sido implementados *ad hoc*, teniendo en cuenta, únicamente, los parámetros planteados a priori, y obviando el hecho de que ambas propuestas se basaban en el mismo modelo teórico GEFRED y arquitectura FIRST, provocando así una incompatibilidad entre sistemas innecesaria.

La *Arquitectura Multipropósito* propuesta permite aumentar en gran parte la operatividad de la Base de Datos Difusa, en el sentido de que hace posible la combinación de operaciones deductivas y de minería de datos mediante el uso de cualquiera de las estructuras definidas en las mismas. Así, es posible la ejecución de operaciones de diversa índole como el uso de reglas lógicas para almacenar resultados de minería de datos (DM), o el uso de tablas intensivas

para hacer cálculos de DM, etc. que aumentan en gran medida la capacidad de consulta del sistema.

Pero la unificación de dichas extensiones devuelve como resultado una arquitectura de SGBD sumamente compleja, dado que además de las relaciones del catálogo necesarias para la representación de cada arquitectura, la comunicación entre ellas requiere de la inclusión de otras nuevas. Lo que provoca a priori un alto coste de recursos en la tarea de comprensión del sistema.

Además, sobre cada implementación de la arquitectura se establecen restricciones propias del SGBD sobre el que ha llevado a cabo, ya que cada SGBD tiene su propia estructura del catálogo, restricciones de acceso, particularidades de lenguaje que implementa, etc. Dichas restricciones establecen un vinculo nada deseable con el sistema concreto. Este es el caso de las implementaciones que ya se encuentran desarrolladas, que utilizan Oracle © como SGBDR y PL/SQL para la implementación de su funcionalidad. Dicho sistema no es portable y únicamente puede ser usado para esta plataforma.

Todos estos problemas han desembocado en un proceso de ingeniería inversa consistente en buscar soluciones que faciliten la compresión del sistema, para hacer menos dificultosa la tarea de definición de información sobre el mismo. La solución encontrada se basa en el uso de Ontologías.

Uso de una Ontología para Representar BD Difusas

La utilización de ontologías es una práctica que actualmente esta dando resultado en el campo de la inteligencia artificial a la hora de representar el conocimiento de forma sencilla y portable. Dicha metodología consiste en una representación jerárquica de un Universo del Dominio concreto que permite al usuario utilizar, compartir y representar el conocimiento de forma comprensible. Dicho conocimiento puede consistir tanto en un campo específico, como en la representación de metaconocimiento que define las estructuras que permiten establecer las características de la información que va a ser almacenada.

Por otro lado, hoy día gran parte de la información disponible, y la gran mayoría de las aplicaciones se utilizan a través de la Web, estando así disponibles en cualquier lugar y momento sin necesidad de instalar nada más que un simple navegador. La Web Semántica, un nuevo entramado de servicios de Internet, consiste en dotar de semántica a cada contenido encontrado en la misma y utiliza principalmente ontologías para acometer dicho propósito.

De esta forma las ontologías han cobrado la suficiente relevancia para generar un gran número de lenguajes (casi estandarizados), herramientas de desarrollo y servicios asociados a las mismas, convirtiéndose así, en el principal mecanismo de representación del conocimiento utilizado en la actualidad.

En este trabajo se ha propuesto una *Ontología Representacional* como

solución a los problemas surgidos en la *Arquitectura Multipropósito*. De esta forma, la representación de la arquitectura en forma de ontología permite aislar de una representación de información vinculada a un SGBDR concreto y además, generaliza los conceptos modelados en arquitectura, simplificando su representación y posibilitando a su vez que cualquier usuario sea capaz de interpretar la información sin necesidad de ser un experto del modelo relacional.

Evidentemente, el proceso de representación del *Servidor de BDD Multipropósito Unificado* debe pasar por una serie de fases. En concreto, se han diferenciado tres: una primera para la representación de la información difusa, la segunda para la representación de la información lógica-deductiva, y por último la representación de las diferentes estructuras de datos y técnicas que permiten realizar tareas de minería de datos sobre un SGBD.

El trabajo que se ha presentado en esta tesis aborda la primera fase. Dicha fase, que consiste en el modelado de la información imprecisa, ha permitido comprobar la viabilidad de esta propuesta, identificando las dificultades encontradas y las herramientas o metodologías de representación más idóneas.

Ontología para la Representación del Conocimiento Difuso

La *Ontología para la Representación del Conocimiento Difuso*, como se ha denominado, ha simplificado el proceso de representación de información difusa haciéndolo más sencillo e intuitivo. La jerarquía de clases que forma la ontología, ha permitido la diferenciación clara de cada elemento de información: atributos, relaciones, dominios de atributos, etiquetas lingüísticas, valores discretos, y sobre todo, los tipos de datos básicos o clásicos han podido ser definidos de forma genérica e independiente del entorno en el que son representados.

El inconveniente encontrado en la ontología se halla en la gran disociación que existe entre el proceso de definición de estructuras, y el proceso de definición o inserción de información. Dicha disociación, que en sistemas de bases de datos es apenas perceptible, en la ontología se manifiesta de forma que es necesario definir por un lado la estructura de la información, tablas, atributos, dominios, restricciones de datos, restricciones de tipos de datos, etc. en forma de instancias. Y, por otro lado, un conjunto de nuevas clases generadas a partir de las definiciones anteriores, que permitan la inserción de las tuplas de información a través de su instanciación. La representación de la estructura de la información se ha denominado *Ontología del Catálogo*. Esta metaontología representa toda la estructura del modelo relacional extendida para la representación de datos difusos. Por otro lado, la representación de una BD concreta, descrita en forma de instancias de dicha *Ontología del Catálogo* se denomina *Ontología del Esquema* en el momento en que dichas instancias

se convierten en clases y atributos y permiten ser instanciables para así poder contener la misma información contenida en las tuplas de una BDD.

El problema reside en que la definición de dichas clases (a pesar de utilizar una metaontología) tiene que ser explícita y realizarse de manera manual en gran parte del proceso. Otro problema encontrado consiste en que dicha disociación provoca que la definición de un gran número de restricciones sobre las estructuras no sea aplicable en las clases generadas para el almacenamiento de la información en la ontología. Dichas restricciones (como las que permiten la inserción de valores únicos, nulos, de clave primaria, de clave ajena, o de tipos de datos) no se pueden mantener o establecer en la ontología (dada la naturaleza de la misma), y pueden provocar un gran problema en la integridad de la información. Este problema puede ser resuelto de forma parcial mediante el uso de razonadores para encontrar incompatibilidades en la definición de la información.

Otra solución para reducir el número de restricciones consiste en el desarrollo de un método que permite la generación automática de la *Ontología del Esquema* a partir de su definición sobre la *Ontología del Catálogo*. Este proceso intenta incluir en la *Ontología del Esquema* basada en OWL el mayor número de restricciones, como: restricción sobre el tipo de dato que puede contener un atributo, restricción sobre el número de valores asociado a un atributo a uno solo, restricción sobre los valores que un tipo difuso 3 puede contener, etc. Sin embargo, la mejor solución está a posteriori, cuando la estructura generada es trasladada a un SGBD real, siendo los mecanismos de integridad implementados por el propio SGBD los encargados de velar por el cumplimiento de dichas restricciones.

De cualquier forma, y a pesar de esta desventaja, la representación de la arquitectura unificada a través de una única ontología permite demostrar la viabilidad de una representación de nuevos tipos de información de una forma más simple que la que proporcionan los SGBD y cómo, a través de un interfaz adecuado, la definición de la misma puede llevarse a cabo de forma efectiva y sencilla para el usuario.

Herramientas para la explotación de la Ontología

Se debe tener en cuenta que la creación de la *Ontología para la Representación del Conocimiento Difuso* consiste básicamente en una ayuda en el proceso de definición de la información, considerado como el más costoso. A este proceso habrá que añadir el de ayuda a la elaboración de consultas sobre datos difusos en un SGBD Extendido. Para ello se han desarrollado una serie de aplicaciones:

- Herramienta para definir estructuras del Catálogo: esta herramienta basada en tecnología web, permite la definición de las estructuras necesarias para permitir el almacenamiento de BDD en cualquier SGBD. En la misma interfaz se permite hacer una consulta del contenido del catálogo.

- Herramienta para definir esquemas de BDRD: esta herramienta permite definir esquemas de BDRD de manera guiada al usuario que no tiene por qué conocer las particularidades del sistema donde dicha información vaya a ser almacenada. Incluso el usuario puede ser ajeno a las particularidades del lenguaje de ontologías, gracias al uso de asistentes que permitan realizar las correspondientes definiciones basándose en la ontología. Esta herramienta está implementada mediante tecnologías web, y también como extensión a la herramienta de gestión integral de ontologías *Protégé*. Además permite la gestión de los esquemas a través de la ontología de manera simultánea con varios SGBDR.

- Herramienta de consulta sobre esquemas de BDRD: esta herramienta permite guiar la consulta sobre un esquema de BDD que se encuentra representado en forma de ontología. Dicha interfaz de consulta proporciona independencia del lenguaje de consulta necesario para manipular datos difusos (y por consiguiente a las estructuras especiales que este tipo de información requiere). El resultado será una sentencia en FSQL que podrá ser lanzada sobre un SGBD que soporte dicha funcionalidad a través del uso de herramientas como el *FuzzyQuery2+* [Bla02b]. Está desarrollada como extensión de la plataforma *Protégé*.

- Herramienta de definición de datos: esta herramienta ayuda al usuario a definir tuplas de una BDD a través del uso de su correspondiente *Ontología del Esquema*. Dicha herramienta permite visualizar de manera intuitiva las instancias de la *Ontología del Esquema* y volcarlas al SGBDRD correspondiente de manera directa. A su vez también permitirá comunicarse con diversos SGBDRD heterogéneos de manera simultánea. Está desarrollada como extensión de la plataforma *Protégé*.

La utilización de herramientas a través de Web evita al usuario la necesidad de instalar otro tipo de aplicaciones para definir la información sobre un SGBDR concreto, además de incorporar la posibilidad de usarlas en remoto.

Sobre la utilización de la herramienta *Protégé* para desarrollar las aplicaciones se ha argumentado mucho a lo largo de este trabajo. Resumiendo, se trata de un entorno que facilita la incorporación de nuevas operaciones a su entorno, a su vez aporta un gran número de bibliotecas de funciones y métodos para gestionar las ontologías representadas sobre dicha herramienta. Por

otro lado, presenta una interfaz intuitiva, una representación eficiente de las ontologías en diversos lenguajes que, junto con otras características hacen que la herramienta sea una de las más conocidas y extendidas de la comunidad.

En cuanto a la elección del lenguaje para la representación de la ontología, se ha seleccionado OWL-Full debido a que permite definir metadatos. A su vez, es el más extendido gracias a sus otras dos representaciones de naturaleza más restrictiva (OWL-DL y OWL-Lite) y a que está orientado a la web. Gracias a estas cualidades dicho lenguaje es implementado por un gran número de herramientas que bien trabajan con el, bien son capaces de importarlo o exportarlo. Todas estas ventajas lo convierten en la opción más atractiva para llevar a cabo la representación de la *Ontología de Representación del Conocimiento Difuso*.

6.2. Beneficios de la Propuesta

Gracias a las propuestas planteadas en este trabajo, el *Servidor Multipropósito* y la *Ontología para la Representación del Conocimiento Difuso*, se han obtenido una serie de ventajas. Dichas ventajas, resumidas a continuación, se encuentran clasificadas en función de los beneficios proporcionados por la propuesta teórica o por la implementación realizada de la misma.

Ventajas Provenientes de la Propuesta Teórica

- La extensión de un SGBDR para manipular datos imprecisos y otro tipo de operaciones de forma conjunta, tal y como se ha propuesto con la *Arquitectura Multipropósito*, permite multiplicar las posibilidades de explotación de la información difusa sobre un SGBDR a la vez que confiere al sistema una mayor escalabilidad.

- La definición de la *Ontología de Representación del Conocimiento Difuso* ha permitido definir una interfaz entre el SGBDRD y el usuario de tal forma que, aisla al esquema de BDD y al mismo usuario de las representaciones particulares que hacen los diferentes SGBDR en los que la información es representada.

- El uso de la *Ontología de Representación del Conocimiento Difuso* simplifica al usuario el proceso de definición de datos difusos, puesto que lo aisla de las particularidades de representación que tienen los datos difusos.

- El uso de la *Ontología de Representación del Conocimiento Difuso* para representar un esquema de BD permite portar dicha representación a

cualquier SGBD Relacional.

- Se puede publicar la estructura de un esquema de BD en Internet simplemente con su representación en forma de ontología, y acceder a la información de dicha base de datos utilizando interfaces genéricos de consulta como el ISQLPlus ©.

- Una ontología que representa a una BD Relacional permite representar semánticamente una interfaz web que carezca de dicha semántica asociada, debido a que sus datos se obtienen gracias a una consulta realizada a través de un formulario Web. La ontología estará incluida en el sitio web dónde se encuentre dicho formulario.

- El uso de *Ontologías del Esquema* permite, mediante mecanismos de unificación y búsqueda de correspondencias, intercambiar información entre fuentes de datos heterogéneas (desde diferentes esquemas almacenados en SGBDs hasta esquemas con diferente tipo de representación al margen del modelo relacional).

- La ontología propuesta permite extender la representación de los SGBDD con otras operaciones o estructuras, de forma sencilla e intuitiva, dada la generalidad con la que representa la información en este medio.

Ventajas Provenientes de la Implementación de la Ontología

- La propuesta de la *Arquitectura de Comunicación Unificada*, que establece la comunicación entre el usuario con la ontología y los SGBDRD, contempla todas las alternativas en el proceso de definición de información de una BDD sean cuales sean las características del SGBDR sobre el que dicha BDD vaya a ser definida.

- La utilización de una interfaz web para manipular la ontología del Catálogo/Esquema permite al usuario utilizar un simple navegador para operar sobre las mismas además de ejecutar los procesos de definición en remoto.

- La elección de *Protégé* como entorno para desarrollar la herramienta de definición de Bases de Datos Difusos, aporta los beneficios propios que tiene dicho entorno (interfaz colaborativa, intuitiva, extensible, etc.) a las utilidades implementadas añadidas a dicha herramienta.

- La herramienta de consulta facilita al usuario la definición de una consulta en FSQL. Así el usuario no tiene la necesidad de conocer las particularidades de este lenguaje. Esta consulta sólo podrá ser utilizada

sobre SGBDRD que dispongan de la implementación que interprete este lenguaje de consulta extendido.

- La interfaz de definición de esquemas proporciona asistentes que guían al usuario a la hora de definir un esquema de bases de datos, simplificando así el proceso además de aislar a dicho usuario de los detalles específicos de representación de datos difusos.

- La interfaz web de definición de estructuras del catálogo permite representar las estructuras de datos que necesita un SGBDR cualquiera para almacenar datos difusos. Dicha interfaz permite esta definición de manera inmediata y remota.

- La interfaz de inserción de datos facilita el proceso de inserción de las instancias definidas en la *Ontología del Esquema* a la BDD, en caso de ser necesario. Además facilita la definición de las tuplas, aislando al usuario de las particularidades de representación de los datos difusos o del lenguaje FSQL.

- La herramienta de definición de esquemas o inserción de datos permite conectar simultáneamente a varios SGBDR, y por tanto realizar dichas operaciones de forma paralela.

- La interfaz web permite visualizar el esquema definido sobre la *Ontología del Catálogo* en OWL de forma estructurada y comprensible al usuario (ignorando detalles de sintaxis que complican su interpretación).

- La generación automática de la *Ontología del Esquema* a partir de las instancias de la *Ontología del Catálogo* facilita al usuario el trabajo de definición de la misma además de incluir de manera automática un gran número de restricciones en dicha ontología que facilitan la tarea de mantener la integridad de la información.

6.3. Trabajos Futuros

A continuación se exponen algunas propuestas de trabajos futuros que surgen a raíz de este trabajo de tesis, que dada su naturaleza y carácter novedoso, se encuentran en gran número. Dichas propuestas se basan en extensiones a este trabajo con el objetivo de ampliar los mecanismos de representación y gestión de información imprecisa, o en introducir mejoras a alguno de los objetivos alcanzados en la misma evaluadas a partir de los resultados obtenidos. Las propuestas se clasificarán en función del objeto u operación a extender:

Con Respecto a la Ontología

- La Extensión de la *Ontología de Representación del Conocimiento difuso* para permitir representar información lógica para deducción sobre los SGBDD. Para ello, deberá establecerse una aproximación a la representación de reglas lógicas con grado de acoplamiento difuso y los conceptos de tablas intensivas y su interrelación con las estructuras que permiten representar el resto de información del modelo relacional y difuso.

- La Extensión de la *Ontología de Representación del Conocimiento Difuso* para permitir representar datos con un dominio complejo. La definición de dicho tipo de datos, permitirá la posterior definición de las estructuras necesarias para integrar operaciones de minería de datos sobre un sistema de información. Para llevar ésto a cabo será necesario ampliar el tipo de información que puede representarse en la ontología y el límite de dominios que puede definir un atributo. Además, las operaciones de minería de datos que se pueden realizar sobre el sistema deberán ser definidas sobre la ontologías junto con sus parámetros y restricciones, al igual que se hace a través del concepto de proyecto propuesto en la extensión del SGBD, *DmFirst*.

- Desarrollo de una *Ontología de Consulta* que represente la información implicada en dicha consulta, proveniente de la *Ontología del Esquema*, y las operaciones que intervienen en la misma. Esta ontología se formará a partir de una operación de proyección sobre la *Ontología del Esquema*, de esta forma se obtiene una sub-ontología de la *Ontología del Esquema* origen donde las instancias de la misma pueden ser los resultados de la ejecución de la consulta. Dicha *Ontología de Consulta* permitirá independizar la consulta de la herramienta donde se esté ejecutando, convirtiéndola así en portable a cualquier entorno donde pueda ser realizada y evitando vincular la misma únicamente a sistemas de modelado relacional.

Con Respecto a la Implementación

- Desarrollo de una herramienta que implemente los asistentes para la generación de esquemas de BDD a través de la Web como alternativa a la utilización de la herramienta *Protégé*. De esta manera se independiza esta función de la representación particular que hace Protégé de las ontologías, y del requerimiento de su instalación para poder acceder a dicha funcionalidad.

- Desarrollo de una herramienta de Consulta utilizando un entorno Web como alternativa a la herramienta desarrollada en *Protégé*.
- Desarrollo de una herramienta de Inserción de datos sobre la *Ontología del Esquema* utilizando un entorno Web como alternativa a la herramienta desarrollada en *Protégé*.

Con Respecto a su Relación con el Entorno. Casos de Uso.

- Desarrollo de un mecanismo que permita establecer las correspondencias entre una *Ontología del Esquema* cualquiera en forma de instancias de la *Ontología del Catálogo*. Este proceso será el inverso al visto en esta tesis y conducirá a una inmediata traducción de una Ontología del Esquema proveniente de cualquier fuente a un SGBDRD. Por ejemplo, este proceso sirve para que cualquier ontología de la Web pueda ser definida de manera inmediata en un SGBDRD siempre y cuando se establezca la correspondencia entre dicha ontología y la *Ontología del Catálogo*. Una vez definida la ontología original como instancias de la *Ontología del Catálogo*, el proceso de comunicación con el SGBDRD es inmediato.
- Estudiar y desarrollar una herramienta para realizar el proceso de importación de un esquema establecido en un SGBDRD concreto en forma de instancias de la Ontología del Catálogo. Esta herramienta permitiría Exportar un Esquema de BDD a cualquier SGBDR.
- Implementación de operaciones que permitan la Unificación de Ontologías del Esquema. Se trata de estudiar y desarrollar algún mecanismo de identificación de elementos de *Ontologías del Esquema* diferentes que representen conceptos similares. Si dicha identificación es positiva, se deben establecer reglas para definir las correspondencias ente conceptos similares (por ejemplo: cambios de formato, rango, escala, etc.).
- Desarrollar la operación de unificación de esquemas compatibles de SGBDR heterogéneas. Esta operación, descrita en el apartado 5.4, consiste en combinar dos SBDRD que representen conceptos similares. Para ello se utilizarán procesos de identificación de correspondencias entre conceptos representados a través de las Ontologías del Esquema correspondientes. Una vez identificadas, realizar el proceso de mezcla de las mismas.
- Definición de Esquemas de BDD a partir de un esquema cualquiera. Consiste en desarrollar una herramienta capaz de realizar el proceso de definición de un esquema cualquiera (un esquema XML, una folksonomía o una Ontología cualquiera) en forma de instancias de Ontología del

Catálogo. Para ello primero deberán convertirse en una *Ontología del Esquema* y a partir de este momento hacer la identificación de elementos (proceso descrito en el objetivo anterior). Una vez en este estado, se procederá a la implantación de dicho esquema en el SGBDRD correspondiente.

- Combinación de Esquemas a partir de Fuentes de Datos Heterogéneas. Se trata de un caso de uso descrito el apartado 5.4, que pretende realizar una representación común para cualquier esquema incorporado en la Web y de esta forma hacer toda la información compatible entre sí. Para llevar a cabo esta operación se requieren por un lado traductores de esquemas cualesquiera a Ontologías del Esquema y por otro, la implementación de herramientas para la unificación de Ontologías que representen conceptos similares.

El resto de los casos de uso descritos en la sección 5.4, que no han sido implementados en este trabajo de tesis, son combinaciones de las operaciones que acaban de ser descritas como trabajos futuros en este apartado. Por tanto, el desarrollo de cada uno de los casos de uso descritos en dicha sección 5.4, se consideran trabajos futuros.

Con respecto al Servidor Multipropósito

- Desarrollo de una biblioteca de funciones y métodos en un lenguaje de programación ampliamente aceptado (como Java o C) implementando las capacidades funcionales para la gestión de datos difusos necesarias para que cualquier SGBDR que no disponga de esta funcionalidad de manera interna, pueda utilizar datos difusos y el lenguaje FSQL. De esta manera se le conferirá al sistema propuesto independencia del SGBDR sobre el que se desee implantar. En el caso en que se desee ganar en eficiencia lo conveniente sería realizar una implementación concreta para cada SGBDR que incluya un lenguaje incrustado propio para la gestión de sus datos.

- Extensión de la librería anterior para gestionar el Servidor SGBDRD Multipropósito.

Otras Propuestas

- Extensión del concepto de ontología para añadir imprecisión en la definición de algunos términos, esto es, una Ontología Difusa. Este concepto

puede ser útil sobre todo en ontologías generadas por procesos de unificación (*merging*) en los cuales los conceptos alineados puede que no coincidan semánticamente al ciento por ciento.

- Desarrollo de un modelo de representación para almacenar eficientemente la información que se encuentra definida sobre Ontologías de Dominio OWL cualquiera en un SGBDR Difuso. Dichas representaciones, denominadas OBDB, son las utilizadas en la actualidad por las herramientas de representación de Ontologías (*Protégé, JENA, Sesame*, etc.) para almacenar las ontologías y sus instancias de las mismas en forma de BD, pero sin tener en cuenta las particularidades de la información representada. Un almacenamiento eficiente de dicha información aportaría una mayor eficiencia a la hora de trabajar con esos datos.

Apéndice A

Conceptos Básicos de Ontologías

A.1. Introducción

Han surgido muchas definiciones del concepto de *Ontología*, a partir de su aparición como nueva técnica de representación del conocimiento en el campo de la tecnología de la información. Algunas de ellas las podemos encontrar recopiladas en trabajos como el de Gómez-Pérez et al. [GP03a] donde se encuentra un estudio muy detallado de los diferentes tipos de ontologías que existen, y propone una metodología para el desarrollo de las mismas, Agarwal [Aga05] hace un repaso por la aparición del concepto de ontología desde su significado filosófico hasta el actual en el campo de la Ingeniería del Conocimiento o Sharman et al. [Sha06b] que revisa el concepto de ontología desde su aparición, hasta las diferentes aplicaciones que tienen en la actualidad, planteando una visión actual de las mismas.

En este anexo se hace un repaso del concepto de ontología, desde sus definiciones más conocidas, las diferentes metodologías que existen para su desarrollo, los formalismos y lenguajes más utilizados para representarlas, o incluso las herramientas más comunes que permiten su definición. Además se repasan conceptos básicos acerca de las operaciones que se pueden realizar con las ontologías que permitirán aclarar al lector acerca de las posibilidades de manipulación de las mismas.

A.1.1. Concepto de Ontología

El concepto de ontologías no es nuevo, proviene del campo de la filosofía y es una disciplina que se suele identificar con la Metafísica general e indica que

la ontología estudia lo que es en tanto que es y existe [Wik07, Aga05, Sha06b, Cor06].

En las áreas de la Inteligencia Artificial, la Ingeniería del Software y las Bases de Datos, de manera independiente concluyeron que la representación del conocimiento era importante para la evolución de las mismas. Así, aunque cada una de estas disciplinas de la Ciencia de la Computación (CC) representaba el problema de la representación del conocimiento de diferente manera, puesto que cada una de estas, está interesada en un problema específico, los investigadores elaboraron una representación válida para una parte específica de la realidad. De esta forma aparece el concepto de ontología en el campo de la CC adquirido como una nueva forma de representar los elementos del mundo real, y tomando este nombre como analogía a su significado filosófico.

Ya en los "80, John McCarthy propuso, en el campo de la Inteligencia Artificial, el concepto de una ontología de entorno como aquella compuesta no sólo por una lista de conceptos de un problema, sino también de sus significados en el contexto. Desde entonces, las ontologías se han asociado con la representación de conceptos. A continuación, muchas otras definiciones de Ontologías surgieron en este campo de la Inteligencia Artificial y la Ciencia de la Computación como la que dio Neches et al. [Nec91] definiéndola como *los términos y relaciones básicas que componen el vocabulario de cualquier área tanto como las reglas para combinar los términos y relaciones que definirán las extensiones de este vocabulario.* A partir de esta definición Guarino y Giaretta [Gua95] clarifican el uso de la palabra ontología que es definida como: *un sistema conceptual informal, un informe semántico conceptual, una especificación de una conceptualización, una representación de un sistema conceptual mediante la teoría lógica, el vocabulario usado por una teoría lógica y una especificación de una teoría lógica.* Gruber [Gru93] define una ontología como *una especificación formal de una conceptualización*, y Borst [Bor97] extiende esta definición estableciendo ontología como *una especificación formal de una conceptualización compartida.* Posterior a esto, Swartout et al. [Swa96] volverá a extender esta definición de ontología como *una estructura jerarquizada de un conjunto de términos para describir un dominio que puede usarse como el esqueleto de una base de conocimiento.* Studer et al. [Stu98] no crea una nueva definición, sino que toma la definición de Borst et al. y explica los conceptos relevantes de la misma:

1. *Formal* se refiere a que es entendida computacionalmente.

2. *Especificación explícita* quiere decir que se definen explícitamente los conceptos, propiedades, relaciones, funciones, restricciones y axiomas.

3. *Compartida* significa que el conocimiento representado será consensuado.

4. *Conceptualización* significa el hecho de que una ontología debe ser un modelo abstracto y a la vez representar una visión simplificada de algún fenómeno del mundo real el cual queremos definir o representar.

Esta definición de Studer et al. [Stu98] es quizá la más clara de todas las expuestas y una de las más referenciadas en la literatura actual, pero existen otras muchas, también muy referenciadas como la propuesta por Gruber [Gru93] o la de Guarino [Gua95] que, obviamente, son muy similares a la expuesta anteriormente.

En resumen y tal y como establece Agarwal [Aga05], una ontología no es más que una manifestación de un conocimiento compartido de un dominio, que es aceptado por un grupo de agentes, y tal acuerdo facilita una correcta y efectiva comunicación del significado, que conduce a otros beneficios como la interoperatividad, reutilización y compartición.

A.1.2. Clasificaciones de Ontologías

Las ontologías pueden representar el conocimiento acerca de realidades de muy diversa índole. Es por este motivo que no existe un consenso en cuanto a la clasificación de las mismas dependiendo del contenido de la materia que representen. Así, encontramos clasificaciones atendiendo al modo en que estas estén representadas o la naturaleza de la realidad que representan como podemos ver en resúmenes hechos por Gómez-Pérez et al. [GP03a], Dang Bui Bach [Bac07] y Agarwal [Aga05] donde encontramos resumidas algunas de estas clasificaciones A continuación, se expone un resumen de las más significativas, organizadas según los autores, y el criterio de clasificación.

Atendiendo al lenguaje usado para implementar las ontologías, o la riqueza semántica de las mismas, se encuentran tres clasificaciones en la literatura (resumidas en la figura A.2):

Uschold y Gruninger [Usc96] distinguen cuatro tipos de ontologías dependiendo del lenguaje usado para implementarlas. Estas son: *Ontologías muy informales*, si están escritas en lenguaje natural, *Ontologías semiformales*, si están expresadas en una forma restringida y estructurada del lenguaje natural, *Ontologías formales*, si están definidas en un lenguaje definido formalmente, y *Ontologías rigurosamente formales*, si están definidas en un lenguaje con una semántica, una teoría y unas comprobaciones formales con propiedades como la completitud.

Lassila y McGuiness [Las02] consideran las ontologías como de mayor o menor "peso"(*light-heavy weight*) de acuerdo con la riqueza semántica con la que representan la realidad. Esto vendrá dado por la cantidad

de estructuras utilizadas para representar el conocimiento expresado en ellas. Por tanto, serán consideradas *ontologías de peso "ligero"* (incluso algunos autores no las considerarían como ontologías propiamente dichas) aquellas que consistan simplemente de un vocabulario o incluso que lleguen únicamente a la representación de la realidad utilizando la estructura *es-un* de manera informal. A partir de la representación formal de una jerarquía *es-un*, hasta llegar a representar restricciones de todo tipo (*inverso de, es parte de* o *no se encuentra en el conjunto*, etc.) se tratará entonces de ontologías con un cierto peso, hasta llegar al máximo nivel correspondiente a la categoría de *ontologías "pesadas"*. En la figura A.1 se presenta dicha clasificación con mayor detalle.

Zhang [Zha07] no llega a clasificar las ontologías realmente hace un repaso exhaustivo por las diferencias entre las ontologías y el resto de sistemas de representación típicos. La clasificación queda establecida en: las *listas de términos*, compuestas por diccionarios y vocabularios controlados, *las listas jerárquicas*, compuestas por esquemas de clasificación y taxonomías, y las *listas de relaciones*, compuestas por tesauros, y ontologías.

Resumiendo, según Ruiz e Hilera [Rui06], cualquier ontología pertenecerá a una de las siguientes categorías según la riqueza de estructura interna:

- Vocabulario controlado: Formado por una lista finita de términos
- Glosarios: Listas de términos con sus definiciones en lenguaje natural
- Tesauros: Se diferencian de las anteriores en que ofrecen semántica adicional a los términos, incluyendo sinónimos.
- Jerarquías Informales: Son jerarquías de términos que no se corresponden a subclases estrictas.
- Jerarquías Formales: En este caso existe la relación *es-un* entre las instancias de una clase y su correspondiente superclase (para explotar el concepto de herencia).
- Marcos (Frames): Son ontologías que incluyen clases y propiedades que pueden heredarse por otra clases en niveles inferiores de una taxonomía formal *"es-un"*.
- Ontologías con restricciones de valor: Incluyen restricciones de valor.
- Ontologías con restricciones lógicas genéricas: Son las mas expresivas permiten especificar restricciones entre los términos de la ontología usando lógica de primer orden.

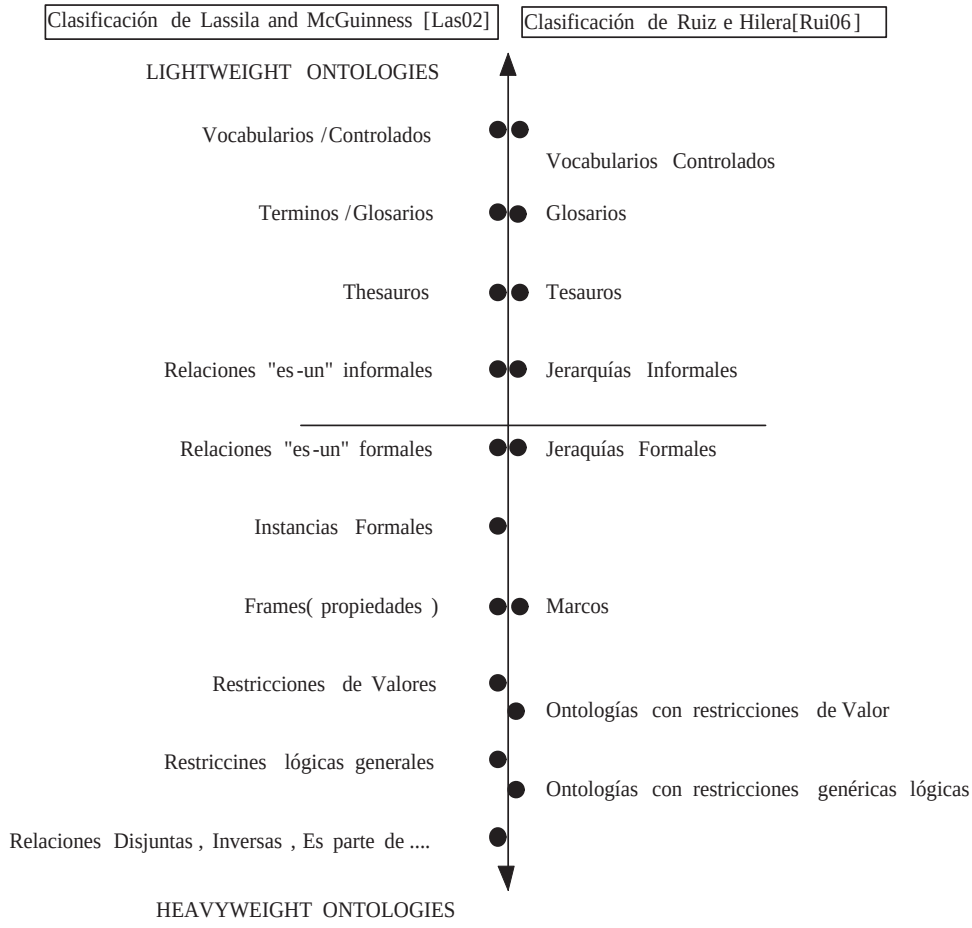


Figura A.1: Clasificaciones de Lassila y McGuinness [Las02] y de Ruiz e Hilera [Rui06]

Atendiendo a la naturaleza de la conceptualización, podemos encontrar los siguientes tipos de ontología (en la figura A.2 se presenta una clasificación muy simplificada de las propuestas siguientes):

Steve et al. [Ste98] existen tres tipos fundamentales de ontologías, las de *dominio*, en las que se representa el conocimiento especializado pertinente de un dominio o sub-dominio, como la medicina, las aplicaciones militares, la cardiología, etc. Las *ontologías genéricas*, en las que se representan conceptos generales y fundacionales del conocimiento como las estructuras parte/todo, la cuantificación, los procesos o los tipos de objetos. Por ultimo las *ontologías representacionales*, en las que se especifican las conceptualizaciones que subyacen a los formalismos de representación del conocimiento, por lo que también se denominan *meta-ontologías o ontologías de alto nivel*.

Guarino [Gua98] Es la más concreta y clara propuesta de clasificación de ontologías, ya que diferencia únicamente cuatro tipos de ontologías: *las Ontologías de Alto Nivel*, las de *Dominio*, las de *Tareas* y por último las *Ontologías de Aplicación*. Esta últimas son aquellas que se crean para describir una actividad o tarea específica, como por ejemplo la venta de un producto, o el diagnóstico de una enfermedad y las ontologías creadas para una aplicación específica. Además Guarino, describe las dependencias que existen entre estas categorías.

Mizoguchi et al. [Miz95] Propone cuatro tipos de Ontologías: *Las ontologías de Contenidos*, entre las que se encontrarían las ontologías que representan los dominios, las tareas, o bien conocimiento en general. Las *ontologías de Comunicación (Tell&Ask)* que permiten compartir conocimiento, las de *Indexado*, que permite recuperar información y las *Meta-Ontologías u Ontologías de representación del Conocimiento*, que son las que describe la estructura de la ontología en si.

van Heisjt et al. [vH97] Estos autores son los primeros en establecer los dos niveles para la clasificación de ontologías que se han especificado previamente(dependiendo del contenido o de la riqueza semántica de la información que representan). Atendiendo a la categorización que se está tratando, dividen las ontologías en dos grandes grupos: aquellas que representan la cantidad y el tipo de estructura (Son las *Ontologías de Representación del Conocimiento*, las *Ontologías de Información* (BD) y los *Lexicons*), y por otro lado, aquellas que representan el objetivo de la conceptualización, es decir, la *Ontologías de Dominio*, de *Aplicación*, de *Representación* o *Genéricas*.

Fensel [Fen04] Establece la clasificación en: *Ontologías de Sentido Común* (ofrecen un conocimiento general del mundo), *Ontologías Representacionales* (representan conceptos que expresan conocimiento en un enfoque orientado a objetos o marcos). No pertenecen a ningún dominio en particular. Las *Ontologías de Dominio*, las *Ontologías de Métodos y Tareas* (estas últimas ofrecen una terminología especifica para de métodos resolutivos o para tareas específicas).

Jurisica et al. [Jur99] establece la clasificación de ontologías como: *Ontologías Estáticas*, que son aquellas que describen cosas que existen y sus relaciones. *Ontologías Dinámicas*: que son aquellas que describen aspectos del mundo que pueden cambiar con el tiempo. *Ontologías Intensionales*: Describen motivaciones, intenciones, objetivos, creencias, elecciones, etc. relacionadas con agentes. *Ontologías Sociales*: Describen estructuras organizativas, redes, interdependencias, etc.

Gomez-Perez et al. [GP03a] realiza una clasificación de las ontologías basándose en las clasificaciones expuestas por los anteriores autores. Esta propuesta establece taxonomía muy rica en matices, que incluso permite establecer unos niveles de "usabilidad/reusabilidad" de las ontologías dependiendo del nivel donde se encuentren clasificadas las mismas. Así podemos encontrar una clasificación en la que las ontologías de alto nivel que describía Guarino ahora se desglosan en tres tipos: *Ontologías de Representación, Ontologías Generales y Ontologías de Alto Nivel.* A continuación y aumentando el nivel de usabilidad, podríamos hallar las *Ontologías de Dominio y Tareas* al mismo nivel, pero desglosando ambas a la vez dada la generalidad de las mismas: las *Genéricas* estarían en primer lugar, a continuación las de *Dominio*, y las últimas y menos reutilizables, serían las de *Aplicación.*

Sharmen et al. [Sha06b] se centra en la web semántica para clasificar las ontologías, en 4 categorías: las *Meta-ontologías* (las que describen los lenguajes), las *Ontologías de Alto Nivel exhaustivas* (expresan el conocimiento global, como WordNet, Cyc, ...), las *Ontologías de Dominio Específico Sistemático* (describen un dominio específico pero de manera general), y las *Ontologías Especializadas Simples.*

Al igual que en el caso anterior Ruiz e Hilera [Rui06] hacen un resumen de todas estas propuestas, afirmando que cualquier ontología estará incluida en una de las siguientes categorías:

- Ontología de Representación del Conocimiento: Es aquella que representa las primitivas para formalizar el conocimiento bajo un paradigma

Figura A.2: Clasificación genérica de Ontologías basada en la naturaleza de la conceptualización

concreto de representación del conocimiento.

- Ontología Genérica o Común: Es aquella que representa el conocimiento de sentido común reutilizable en diferentes dominio. Por ejemplo: eventos, espacio, tiempo, etc.

- Ontologías de Alto Nivel: Describen conceptos muy generales que se pueden relacionar con las bases de cualquier ontología.

- Ontologías de Dominio: Son ontologías reutilizables de un dominio particular.

- Ontologías de tareas: Describen un vocabulario relacionado con algunas actividades genéricas. Proporcionan un un vocabulario sistemático de términos usados para resolver problemas que pueden o no pertenecer al mismo dominio.

- Ontologías de Tareas de un Dominio: Igual que la anterior, pero sólo usable en un dominio.

- Ontologías de Métodos: Son definiciones de conceptos relevantes y sus relaciones, aplicables a procesos de razonamiento diseñados específicamente para llevar a cabo una tarea particular.

- Ontologías de Aplicación: Dependen de las aplicaciones. Suelen extender y especializar el vocabulario de una ontología de dominio o tareas para una aplicación particular.

A.2. Ingeniería de Ontologías

Desde su aparición las ontologías han sido una de las ramas más importantes en el área de las Ciencias de la Computación. Se ha desarrollado toda una plataforma alrededor de las mismas, desde técnicas, metodologías y aplicaciones que han formado lo que hoy en día se conoce como *Ingeniería Ontológica* (*Ontological Engineering*). Tal como Gómez-Pérez et al. describió en [GP03b] o Sharman et al. en [Sha06b], la *Ingeniería Ontológica* se refiere al conjunto de actividades que concierne al proceso de desarrollo de ontologías, el ciclo de vida de las ontologías y las metodologías, herramientas y lenguajes que permiten construir las mismas.

A continuación resumiremos brevemente algunas técnicas, metodologías y aplicaciones relacionadas con las ontologías para ofrecer una imagen resumida de este campo de estudio. En el trabajo de Cardoso [Car07] se puede encontrar un análisis detallado de cada uno de los elementos que forman parte de la ingeniería ontología más populares en la actualidad.

A.2.1. Técnicas de Representación de Ontologías

Cualquier formalismo que permita representar ontologías deberá permitir representar los conceptos y las relaciones que existen entre estos. Para ello existen diversas propuestas que clasificaremos según el modo de representar la información y que podemos encontrar resumido en [Sha06b]:

Marcos y lógica de Primer orden Gruber [Gru93] propone utilizar marcos (*frames*) y lógica de primer orden. Su propuesta utiliza clases, relaciones, funciones, axiomas formales e instancias. Las clases son los conceptos relevantes y se organizan en taxonomías, las relaciones, representan diferentes tipos de asociaciones entre los conceptos en un dominio. Las funciones son un caso especial de relaciones. Los axiomas formales son sentencias que son siempre ciertas, y se usan para generar conocimiento y verificar la consistencia de la ontología, y por último las instancias se utilizan para representar los elementos o individuos de las ontologías.

Lógica Descriptiva Baader et al. [Baa04] proponen utilizar Lógica Descriptiva para representar las ontologías. La lógica descriptiva es un formalismo de la lógica que se divide en dos ramas: los TBox y los ABox. Los primeros contienen las definiciones de conceptos y roles, también llamados conocimiento intensional. Los ABox contienen las definiciones de los individuos, y se denominan conocimiento extensional. Se usan tres elementos para representar estas ontologías, los conceptos, que representan

las clase de objetos, los roles, que describen las relaciones entre los conceptos y los individuos, que representan las instancias de las clases. Los conceptos y los roles se especifican basándose en un conjunto de términos pre-existentes y constructores cuyos elementos se mezclan para obtener cualquier tipo de lenguaje DL. Los conceptos primitivos son aquellos cuya especificación no necesita basarse en otros conceptos, pero si sobre condiciones que los individuos deben satisfacer. Los conceptos derivados, son aquellos cuya especificación esta basada en otro concepto, del cual hereda alguna propiedad. Los individuos representan una instancia de los conceptos y sus valores.

Técnicas de Ingeniería del Software Técnicas como el Lenguaje de Modelado Unificado (UML) permiten representar ontologías, aunque existan varios autores que defienden que dicho lenguaje es demasiado pobre para ello (véase sección 2.3). De esta manera dichas ontologías serían clasificadas como de *"peso ligero"* según la taxonomía de Lassila y McGuiness [Las02]. Si a este lenguaje se le añade el lenguaje de restricción de objetos (OCL), dichas ontologías se convertirían mas ricas semánticamente hablando y por tanto mas *"pesadas"* según la anterior clasificación. Los diagramas UML se usan para representar los conceptos donde cada clase representa un concepto. Las taxonomías sería las relaciones de generalización y la relaciones binario serían las relaciones de asociación.

Modelado Conceptual Alternativa para representar ontologías basada en el modelado conceptual [GP03b], usualmente se relacionada con la representación de bases de datos usando, por ejemplo, diagramas Entidad-Relación. En estos diagramas, los conceptos se representan mediante entidades, que tienen atributos, que son a su vez propiedades del concepto. Estos atributos tienen un nombre y un tipo. Las relaciones entre los conceptos se representan mediante las relaciones del E/R, que tienen cardinalidad y permiten expresión, no sólo para relaciones de asociación, sino también de generalización, que crea las taxonomías de los conceptos. Los axiomas formales, pueden representarse usando restricciones de integridad. El UML también entraría dentro de esta clasificación dada la gran capacidad expresiva que tiene. Sin embargo, un gran número de autores no consideran estas técnicas las más apropiadas para representar una ontología, como describe Ruiz e Hilera en [Rui06] o podemos analizar con más detalle en la sección 2.3.

A.2.2. Metodologías de Representación

Al igual que ocurre con cualquier elemento software, o base de datos, una ontología, debe ser construida atendiendo a algún tipo de metodología que permita establecer los criterios a seguir hasta su definición completa.

Las metodologías de la ingeniería ontológica necesitan seguir tres actividades fundamentales [GP03a, GP03b]:

- Actividades de gestión de Ontologías. Esta actividades incluyen la organización en la tarea de ingeniería de la ontologías. Necesita definir mecanismo de control y pasos para asegurar la calidad.

- Actividades para el proceso de desarrollo de ontologías. Entre estas actividades están la elección del entorno de desarrollo, el estudio de viabilidad. Además, en ella se desarrollan los procesos de conceptualizar, formalizar e implementar la ontología, y finalmente, la generación de unas guías para el mantenimiento, generación de instancias, uso y evolución de la misma.

- Actividades de Soporte. Entre ellas están las de adquisición, evaluación, integración, unión, compatibilización, y configuración. Estas actividades se llevan a cabo durante las actividades de desarrollo y gestión de las ontologías.

A continuación se presentaran las metodologías de diseño de ontologías más significativas de los últimos tiempos. Existen comparativas de dichas metodologías en las siguientes referencias bibliográficas [Sha06b, Sur04, Sur06, GP03b, hom06].

- *Cyc*, esta metodología se extrajo gracias a la existencia de un proyecto que consiste en la creación de una la ontología*Cyc* [Len95] que trata de representar toda la realidad del universo (*Ontología de Alto Nivel*). Más allá de éxito o fracaso de la misma, se han extraído las tareas más significativas a través dicho proceso de generación: en primer lugar, la extracción manual del conocimiento general o de sentido común, en segundo lugar, la codificación manual guiada por herramientas que utilizan el conocimiento almacenado en la base de conocimiento *Cyc*. En tercer lugar, la extracción del conocimiento.

- Proyecto *Enterprise Ontology*. Uschold and King [Usc95] establecen las siguientes guías a la hora de crear una ontología: identificar el propósito de la ontología, construir la ontología mediante la captura de los conceptos y la relaciones de la ontología, la codificación de la misma usando

un lenguaje formal, y la integración de la misma con otras ontologías preexistentes. Por ultimo habrá que evaluar la ontología.

- *TOVE.* Gruninger and Fox [Grü95] desarrollaron una metodología para implementar la ontología basada en preguntas de competencia. Consiste en seis pasos: 1º Identificar los escenarios, 2º elaborar preguntas informales sobre el tema, en lenguaje natural (servirá para establecer las restricciones), 3º especificar la terminología usando lógica de primer orden. 4º Escribir las preguntas formalmente, usando la terminología formal. 5º especificar axiomas utilizando lógica de primer orden. 6º especificar los teoremas de completitud.

- *DOGMA* [Jar02]. Este método basado en las bases de datos consiste en descomponer explícitamente los recursos ontológicos en bases de ontologías que consisten en hechos simples llamados *lexons* y acuerdos ontológicos del tipo: reglas y restricciones. Meersman et al. [Jar] pretenden con este modelo demostrar que los modelos conceptuales de la información son completamente válidos para representar ontologías. Usan diagramas en ORM para definir la realidad.

- *Amaya* [KAC05]. Establece la posibilidad de reutilizar el conocimiento. Tiene tres etapas: la primera es la de especificar la aplicación para identificar el contexto y los elementos que queremos representar. La segunda es el diseño preliminar basándose en las categorías de alto nivel relevantes. Los elementos identificados en la etapa anterior se usarán como entrada a la ontología de alto nivel para obtener una visión global del modelo. Durante este proceso es posible establecer la reutilización de una ontología ya existente. La tercera etapa es refinar y estructura la ontología, mediante la especialización de términos para obtener el diseño definitivo con la máxima modularización.

- *CommonKADS* [Sch99]. No es una metodología en sí misma para el desarrollo de ontologías. Cubre varios aspectos en el campo de la gestión del conocimiento, desde su definición hasta la implementación de sistemas de información.

- *Methontology* [GP03a, GP03b]. Esta metodología se plantea como una de las más completas, formulada para el desarrollo de ontologías hasta la fecha. Esta inspirada en las metodologías de desarrollo del software, no obstante, contempla todas las casuísticas que puedan darse en el proceso de desarrollo de una ontología. Esta metodología sigue las tres actividades fundamentales para la representación de ontologías anterior-

mente descritas y divide el proceso de modelado de conocimiento en las ocho fases siguientes:

1. Construir el glosario de términos.
2. Construir la taxonomía de conceptos.
3. Construir los diagramas de relaciones *ad hoc* para identificar las relaciones *ad hoc* entre los conceptos de la ontología y los conceptos de otras ontologías.
4. Construir un diccionario de conceptos, que contendrá los conceptos de dominio, sus relaciones, sus instancias y sus clases, y los atributos de instancia.
5. Describir con detalle cada relación binaria *ad hoc* que aparece en el diagrama.
6. Describir en detalle cada atributo de instancia que aparece en el diccionario de conceptos.
7. Describir en detalle cada atributo de clase que aparece en el diccionario de conceptos.
8. Describir cada constante, que especifica la información relativa al conocimiento del dominio.

- Método *SENSUS* [Kni94]. Este método consiste en construir el esqueleto del dominio de una ontología comenzando a partir de una ontología muy grande, e ir recortando aquellos términos irrelevantes para la ontología que se propone. Los procesos principales en esta metodología son: 1º identificar las semillas, 2º unir manualmente los términos semilla a SENSUS, 3º añadir las rutas obtenidas, 4º añadir nuevos términos de dominio, 5º añadir subárboles completos, que dependen de las semillas encontradas.

- *On-To-Knowledge (OTK)* [Sur04]. Esta metodología tiene en cuenta si la ontología será utilizada para futuras aplicaciones. Esta metodología al igual que muchas otras, se divide en dos grandes fases: el proceso de definición de la estructura (*Knowledge Meta Process*), y el proceso de gestión o manejo de la misma (*Knowledge Process*). La primera fase es la que permite definir la ontología consiste en los siguientes pasos: 1. Estudio de viabilidad, consiste en identificar los problemas y oportunidades, las aplicaciones, herramientas y las personas, 2. El arranque, que consiste en capturar los requerimientos en un documento de especificación de documentos (ORSD) y crear una descripción de la ontología semiformal, 3. El refinamiento, se realiza cuando se refina la descripción semiformal

de la ontología, se formaliza y se crea un prototipo, 4. Evaluación, en este paso se evalúa la tecnología, los usuarios y la ontología), 5. La aplicación y Evolución. La segunda fase comprende los pasos de creación de conocimiento e importación de documentos y metadatos, extracción y acceso al conocimiento, etc.

- *IDEF5* [KBS]. Construyen una ontología tratando de catalogar los descriptores (como un diccionario de datos) y creando un modelo del dominio (construido por estos descriptores). Consiste en tres tareas básicas: 1) catalogar los términos, 2. capturar las restricciones que existen sobre estos términos y el dominio y 3. construir un modelo que puede generar las sentencias descriptivas apropiadas. Así una ontología será similar a un diccionario de datos que incluyen tanto la gramática como el modelo de comportamiento del dominio.

- *HCOME* [Kot06]. Esta metodología esta centrada en el papel que tienen los desarrolladores de ontologías en el proceso de creación. Aparece debido a las carencias que tienen en este aspecto el resto de ellas. Identifica tres fases principales: 1. Especificación, define objetivo, ámbito, requerimientos y equipo, 2. Conceptualización, etapa de adquisición de conocimiento (importar, consultar ontologías de alto nivel, consulta a expertos) y Desarrollo y Mantenimiento de la misma (improvisar, unir, gestionar, comparar versiones,añadir documentación, etc.), 3. Explotación, usa y evalua la ontología. Esta metodología soporta el desarrollo de ontologías de forma descentralizada. Por un lado se permite desarrollar una ontología de forma personal, utilizando versiones, ontologías de alto nivel, uniendo ontologías o mapeando términos. En segundo lugar permite mediante un espacio compartido discutir acerca de las decisiones tomadas [Kot04]. .

- *HOLSAPPLE* [Nic05]. Metodología propuesta para el trabajo colaborativo. Soporta la creación de una ontología estática, que es propuesta inicialmente y extendida mediante la retroalimentación (*feedback*) de un grupo de expertos. Utiliza cuestionarios para llevar a cabo esta retroalimentación.

- *UPON (Unified Process for ONtology building)* [Hol02]. Basado en el desarrollo de ontologías utilizando UML. Las fases cíclicas que propone son: 1. identificación de requerimientos, 2. análisis, 3. diseño y conceptualización, 4. implementación, 5. testeo la cobertura del dominio de aplicación.

- *DILIGENT* [Cas07, Sur06]. Esta metodología de generación de ontología trata de suplir la carencia de otras en el proceso de compartir o discutir durante el proceso y ser adaptable a cualquier cambio. Existen las siguientes fases: 1. construcción, 2. adaptación local, 3. análisis, 4. revisión y 5. adaptación local.

- Hüsemann y Vossen en [Hü05] proponen una metodología para representar ontologías basándose en la fase de diseño de bases de datos tradicionales. Esta metodología trata de suplir la carencia de un estándar en este aspecto. El proceso de diseño consiste en una fase de análisis de requisitos, un diseño conceptual, lógico y por último físico.

En el caso de las ontologías, el proceso para construir una metodología ha ido surgiendo a raíz de la construcción de las mismas. Se han observado tres diferentes generaciones de metodologías para ontologías [Rib06], a partir de su aparición sobre el año 1995. La primera generación intenta comprender como las ontologías son construidas, entre ellas podemos encontrar la metodología *TOVE, SENSUS, Cyc* y la *ENTERPRISE*. La segunda generación considera que la especificación, conceptualización, integración e implementación ya es un requisito durante todo el ciclo de vida, y en ella están *METHONTOLOGY, Amaya, DILIGENT, HOLSAPPLE. OTK y HCOME*. Esta última también puede estar incluida en la última generación que es la que incorpora conceptos como reusabilidad y gestión de la configuración. También en los últimos años *METHONTOLOGY*, podría considerarse de esta ultima generación, ya que ha incluido estos aspectos en su definición [Cor06].

A pesar de todas estas metodologías propuestas, aún sigue existiendo el profundo convencimiento de que no hay ninguna metodología estándar o ampliamente aceptada para los constructores del conocimiento [Rib06, Hü05]. Esto se hace más evidente en el campo de la Web Semántica donde los detractores de estas metodologías son más numerosos. De hecho, existe un alto porcentaje de desarrolladores de ontologías que no siguen ninguna metodología [Car07]

A.2.3. Formalismos y Lenguajes en la Representación del Conocimiento

Existen numerosos formalismos y lenguajes que permiten representar el conocimiento (véase [Gae06, GP03b, Sha06b, Par04]) que puede estar definido utilizando diversas técnicas de representación, como por ejemplo: Tripletas objeto-atributo-valor, hechos inciertos, marcos (*frames*), hechos difusos, reglas, redes semánticas, etc.. Cada uno de estos formalismos utilizan unos recursos diferentes para representar la información. Este hecho hace que no todos los

lenguajes puedan representar la misma información, ni el mismo grado de semántica de dicha información. Elegir el lenguaje de representación idóneo es crucial, por lo tanto, para representar la realidad de la manera que más se adapte a nuestro problema.

Los lenguajes o formalismos que nos permiten representar el conocimiento pueden clasificarse en tres grupos [Gae06, Ech07b]:

Lenguajes Basados en Lógica Los lenguajes basados en lógica parten de una sintaxis y semántica bien definida que detallan perfectamente la forma de construir sentencias y razonamientos sobre ellas. Entre los lenguajes basados en lógica podemos encontrar:

- *Lógica Proposicional*, que consiste en la definición de proposiciones y los valores de verdad que cada una tiene asociados. Se infiere información a través de la generación de proposiciones más complejas y los operadores de AND, OR, IMPLIES, EQUIVALENCE.
- *Lógica de Primer Orden*, amplía la anterior, añadiendo dos operadores más, el cuantificador Universal $\forall$ y el Existencial $\exists$. Además los símbolos pueden representar constantes, variables, predicados y funciones.
- *KIF*, lenguaje lógico basado en lógica de primer orden que fue creado con el objetivo de actuar como interlingua entre diferentes formalismos y lenguajes de representación. KIF dispone de su propia sintaxis y algunos añadidos semánticos sobre la lógica de primer orden.
- *Lógica Descriptiva*. Son las mas relacionadas con el desarrollo de las ontologías tal como se usan en la actualidad en la Web Semántica. La lógica descriptiva se basa en representar el conocimiento utilizando por una una terminología o vocabulario del dominio (TBOX) y por otra un conjunto de afirmaciones (ABOX). Se pueden construir y existen razonadores que permiten razonar sobre las TBOX y ABOX, pudiendo determinar por ejemplo si el contenido de la TBOX es factible, o qué relaciones están incluidas en otras.

Lenguajes Basados en Marcos (*Frames*) Estos lenguajes son similares a los lenguajes de programación orientados a objetos, en el sentido de que representan el conocimiento utilizando clases (marcos), atributos, objetos y relaciones, y utilizan relaciones de generalización y especialización para representar la organización jerárquica de los conceptos. Es importante mencionar que muchos de los lenguajes basados en marcos se pueden considerar como una sintaxis diferente de la lógica de primer

orden y que por lo tanto no ofrecen más expresividad que ella. Esto implica, por otro lado, que tengan representaciones equivalentes en el lenguaje KIF visto en un punto anterior. La mayoría de las herramientas de definición de ontologías utiliza estos lenguajes.

Lenguajes Basados en Reglas Estos lenguajes han sido durante mucho tiempo posiblemente los más usados de todos, principalmente debido a su estrecha relación con los *Sistemas Expertos* utilizados en *Inteligencia Artificial*. Estos lenguajes son fáciles de entender debido a su sencillez conceptual y a su paralelismo con las estructuras de control más simples utilizadas en programación. Una de las tendencias más recientes pasa por mezclar los conceptos de marcos y reglas, como en el caso de Jess, para disponer de lenguajes que permitan reunir la información de cada concepto y asociar a alguno de sus slots conjuntos de reglas. Este tipo de lenguajes han recibido también un fuerte impulso a partir de la aparición de la Web Semántica, ya que que se piensa en ellos como herramientas para definir servicios web.

A.2.3.1. Lenguajes de Representación de Ontologías

Una vez definidos cada uno de los formalismos se hará un breve repaso por los lenguajes más utilizados para representar ontologías [GP03a, Dui00, Su02, Sha06b]. Estos lenguajes aparecen a continuación clasificados atendiendo a la naturaleza de la información que representan:

- *Lenguajes usados tradicionalmente para la especificación de Ontologías*, entre estos encontramos: *CycL*, que usa lógica de primer orden, *Loom*, que usa lógica descriptiva, *Ontolingua* que se basa en marcos y en Kif, F-Logic, *CML*, *OKBC*, *OCML*, están basados en Marcos también.
- *Lenguajes de Web estándar para representar ontologías en la Web Semántica*. Estos lenguajes son *RDF*, *RDFS*, *DAML+OIL* y *OWL*. En general, todos están basados en lógica descriptiva y en los lenguajes basados en marcos. Existe un lenguaje muy reciente, el WSML (*Web Service Modeling Language*) que describe los servicios de la Web Semántica usando formalismos lógicos, y que está ganando en popularidad.
- *Otros lenguajes Web*, creados para la especificación de ontologías como *XOL* (utiliza sintaxis XML y OKBC), *SHOE* (basado en macos y reglas), y *OIL* (basado en marcos)

Las ontologías pueden representarse utilizando cualquiera de estos formalismos, incluso podríamos añadir dos más como:

- *Modelos conceptuales para representar Bases de Datos.* En estos formalismos, podríamos incluir las bases de datos relacionales y las bases de datos orientadas a objetos (SQL, Modelos E-R, ..)
- *Modelos de representación del Software.* En este formalismo incluiremos el UML como máximo exponente.

Sin embargo, no todos los lenguajes conducen a la generación de ontologías *pesadas* y los dos últimos formalismos, concretamente, son más "pobres" según algunos autores (ver apartado 2.3) a la hora de representar semántica en la información dada la naturaleza de la información que están destinadas a representar. Las restricciones en este tipo de formalismos vendrán ligadas al uso de alguno de los lenguajes anteriores, como el uso de reglas o lógica para completar la semántica de los mismos.

Además de todos estos lenguajes, otros mecanismos de representación de ontologías han surgido al margen de estos. Son las herramientas de representación de ontologías descritas en el apartado A.2.3.3. Estas herramientas, además de ayudar a la hora de representar una ontología, definen su propio modelo de datos cuando definen la información que dicha ontología describe. De esta forma, la representación de la ontología está sujeta a la modelización que estas herramientas hacen. Sin embargo la gran mayoría introducen la opción de exportar o traducir la realidad que representan en alguno de los lenguajes más populares, como OWL, KIF, RDF, etc.

A.2.3.2. Lenguajes Estándar de Web para Representar Ontologías

La incorporación, y en algunos casos aparición de estos lenguajes en la web surgió como necesidad de incluir significado a los documentos web estructurados que, representados en HTML o XML, carecían del mismo para ser procesados computacionalmente.

A continuación expondremos brevemente las características de los lenguajes más comúnmente utilizados para la representación de ontologías, sobre todo en la Web [Cha99, GP03b, Ech07a]. Dichos lenguajes se han convertido en estándares, gracias a la propuesta y aceptación por parte de la W3C[1] [w3c06].

RDF - Resource Description Framework

RDF [Cha01, W3C99] es un marco de trabajo o *framework* para describir e intercambiar metadatos, por tanto, no es un lenguaje únicamente, sino que es en sí mismo un modelo de datos. Está construido en base a las reglas siguientes:

[1] Sitio web: www.w3c.com

- Recursos: Cualquier cosa que puede tener un URI, esto incluye a todas las páginas web, los elementos individuales de un documento XML, etc.

- Propiedades: Es un recurso que tiene un nombre y que puede usarse como una propiedad, por ejemplo el autor o el título.

- Sentencias: Consisten en la combinación de un recurso, una propiedad y un valor. Estas tres partes se conocen como sujeto, predicado y el objeto de la sentencia.

Otros elementos de la definición de una página RDF estan brevemente descritos en la tabla A.1.

Tabla A.1: Elementos de una página RDF

Elemento	**Descripción**
rdf:Statement	Reificación: Cuando se construyen sentencias utilizando otras sentencias
rdf:RDF	Cada documento RDF al estar basado en XML debe estar bien formado (*Well-Formed RDF*) y tener este elemento raíz.
Espacio de Nombres	*http://www.w3.org/1999/02/22-rdf-syntax-ns#*
rdf:Description	Declaración de Recursos. Se usa como valor del atributo una referencia URI
rdf:about	Se refiere al recurso que esta fuera del documento
rdf:ID	Se refiere al recurso si esta dentro del documento
rdf:datatype	Usado para Valores de Datos si el valor no es un literal
rdf:resource	Especificación de un Recurso como un Valor de Propiedad
rdf:type	Declara instancias
rdf:Bag, rdf:Seq, y rdf:Alt	Elementos contenedores
-	Las definiciones se pueden anidar

RDF Schema

Es una extensión semántica de RDF [w3c06, Bac07]. La primera versión fue publicada en Abril de 1998 por la W3C, la última versión se publico en 2004. Los archivos RDFS son archivos RDF que tienen la misma estructura y sintaxis que la que se usa en RDF.

A pesar de que aparece para enriquecer al anterior, ya que el RDF sólo es capaz de representar instancias y las relaciones entre las mismas. No puede representar ni siquiera la estructura más fundamental de las ontologías, como son las clases, o hacer taxonomías. El RDF(S) incorpora constructores que permiten especificar los elementos que muestra la tabla A.2.

Tabla A.2: Elementos de una página RDF Schema

Elemento	**Descripción**
rdfs:Class	Definición de clases
rdfs:SubclassOf	Definición de Jerarquías
rdfs:subPropertyOf	Especificación de jerarquías de propiedades
rdfs:domain	Especificación de dominio en las propiedades
rdfs:range	Especificación de rangos en las propiedades
rdfs:seeAlso, rdfs:isDefinedBy, rdfs:comment, rdfs:label	Comentarios
Espacio de Nombres	*http://www.w3.org/1999/02/22-rdf-syntax-ns.*

El RDF(S), no es demasiado expresivo, sólo permite la representación de conceptos, taxonomías de conceptos y relaciones binarias. Concretamente, el RDF(S) no puede expresar [Bac07]:

- El ámbito local de las propiedades. Por ejemplo, podemos decir que un *profesor* puede enseñar a un *estudiante*, pero no podemos especificar que un *asistente* sólo puede enseñar a un *estudiante no graduado*.
- Clases Disjuntas.
- Combinación booleana de clases, por ejemplo, que un *HombreBelga* en una intersección de un *Belga* y un *Hombre*.
- Restricciones de cardinalidad.

- Características de las propiedades, como la transitividad..

El OWL solventa estos problemas, manteniendo la compatibilidad con el RDF(S).

OWL o OWL Web Ontology Language

Lenguaje de Ontologías para la Web(OWL) [Ant03, Bac07, Bec07a] es un lenguaje de etiquetado semántico para publicar y compartir ontologías en la web. Se trata de una recomendación del W3C, y puede usarse para representar ontologías de forma explícita, es decir, permite definir el significado de términos en vocabularios y las relaciones entre aquellos términos. OWL es una extensión del lenguaje RDF y emplea las tripletas de RDF, aunque es un lenguaje con más poder expresivo que éste [Ech07a].

El OWL está diseñado para usarse cuando la información contenida en los documentos necesita ser procesada por programas o aplicaciones, en oposición a situaciones donde el contenido solamente necesita ser presentado a los seres humanos. OWL surge como una revisión al lenguaje DAML-OIL y es mucho más potente que éste.

Al igual que OIL, OWL se estructura en capas que difieren en la complejidad y puede ser adaptado a las necesidades de cada usuario, al nivel de expresividad que se precise y a los distintos tipos de aplicaciones existentes como motores de búsqueda, agentes, etc.

OWL propone tres tipos de lenguaje, puesto que identifica tres tipos de aplicaciones que el usuario puede desarrollar. Así aparecen los tres tipos de OWL, ordenados de mayor a menor complejidad:

1. OWL-Full. Desarrollado para aplicaciones que requieren mucha expresividad y libertad en la sintaxis RDF. Así pues es un conjunto que engloba a todos los demás lenguajes (RDF, OWL-DL,OWL-LITE). Permite incluso la re-definición del meta esquema del lenguaje mismo. Un documento en RDF es equivalente a un documento OWL-FULL en cuanto a validez. Es muy flexible, pero nunca se podrá razonar utilizando este lenguaje.

2. OWL-DL. Usado para aplicaciones que requieren expresividad y completitud, y la habilidad del razonamiento. Comparte parte del lenguaje de OWL-Full, pero restringe el uso de algunas primitivas. Por ejemplo: las propiedades de Objetos y de tipos de datos están disjuntas, así que no pueden compartir propiedades. No existen restricciones de cardinalidad sobre propiedades transitivas. Además los axiomas deben estar bien formados.

3. OWL-LITE. Desarrollado para aplicaciones que requieren simplicidad y rapidez en la ejecución. Es un subconjunto de OWL-DL y es el más utilizado por las herramientas de construcción de ontologías. OWL-LITE prohibe el uso de los atributos: owl:oneOf, owl_unionOF (excepto en los constructores de clases y clases que tengan nombre tras la intersección), owl:complementOf, owl:hasValue, owl:disjointWith, owl:DataRange.

Un documento en OWL contiene los elementos resumidos en la tabla A.3.

A.2.3.3. Herramientas de Ontologías

El establecer una metodología y un lenguaje de representación de ontologías no es suficiente para manipular las mismas. Es necesaria la creación de herramientas que faciliten al usuario o al desarrollador dicha manipulación. Es por este motivo que se han desarrollado un gran número de herramientas al respecto. Estas herramientas pueden clasificarse siguiendo dos criterios diferentes: El mecanismo de representación en el que están basadas, o la finalidad que tengan las mismas. En cuanto al primer criterio nos encontramos con (para mayor detalle ir a Gómez-Pérez et al. [Cor06]):

- Herramientas que se corresponden o representan directamente utilizando un lenguaje de representación de ontologías (*OilEd, RdfEdit, SWOOP, etc.*)
- Herramientas integradas, cuya característica principal es tener una arquitectura extensible y su representación del conocimiento es independiente de cualquier lenguaje (*Protege, WebOde, OntoEdit, etc.*).

Dependiendo de la finalidad que tengan estas herramientas, se han clasificado en (clasificación extendida a la dada por Gómez-Pérez [GP03b]):

- Herramientas de desarrollo o representación de ontologías.
- Herramientas de evaluación de ontologías.
- Herramientas de integración y alineamiento de ontologías.
- Herramientas para anotar ontologías.
- Herramientas de consulta e inferencia.
- Herramientas de evolución y versionado.
- Herramientas de generación de ontologías (*learning tools*).

- Herramientas de integración con bases de datos.

Las herramientas de representación de ontologías, permiten la definición de las mismas siguiendo el ciclo de vida de desarrollo. Estas herramientas, se caracterizarán por la capacidad para seguir una metodología concreta, o bien seguir cada una de las opciones del ciclo de vida de una ontología, desde la definición de la ontología, creación de instancias, traducción a un lenguaje de representación formal y mantenimiento. La mayoría de estas herramientas se basan el el formalismo de representación de marcos (*frames*).

A continuación se presentan las herramientas de representación de ontologías más conocidas. Entre las herramientas que se exponen, están aquellas que únicamente se crean para poder editar o definir ontologías en un lenguaje determinado y herramientas que son una herramienta de construcción de ontologías en sí mismas que guian al usuario en la elaboración de las mismas. Un resumen más detallado de estas herramientas lo podemos encontrar en cualquiera de estas referencias [Dui00, Su02, GP03a, You06] y en las propias páginas de descarga de cada una de ellas.

- Protégé [atSUSoM06, Knu]. Esta herramienta tiene su propio modelo de representación de ontologías, no obstante, proporciona mecanismos para importar y exportar ontologías en los lenguajes más comunes de representación. Es una herramienta que trabaja en modo local y la información se puede guardar en documentos de texto o en una base de datos. Protege incluso aporta una herramienta especial para trabajar directamente con ontologías de tipo OWL. Además, permite que toda la comunidad incorpore nuevas aplicaciones a la herramienta mediante la creación de extensiones o *plug-ins*, para ello pone a disposición de los usuarios una API que gestiona su propio modelo de datos. Esta herramienta, permite manipular cualquier aspecto de la ontología, incluyendo la definición de metadatos. Además incluye opciones para permitir el trabajo colaborativo. Actualmente es la herramienta líder con un 68.2 % de usuarios [Car07].

- WebODE [otTUoMS07, Arp01].Herramienta que permite representar ontologías también siguiendo su propio modelo de representación de datos. Dichas ontologías serán almacenadas en una base de datos. Proporciona herramientas para importar y exportar ontologías en lenguajes estándar como OWL, y RDF. No soporta la gestión de metadatos. Se usa a través de un navegador Web y permite gestionar casi todas las fases del desarrollo de una ontología, concretamente se basa en la metodología *methontology*.

- OntoEdit [Ont07a]. Herramienta de edición de ontologías que apoya el desarrollo y mantenimiento de las mismas utilizando medios gráficos en un entorno web. Esta herramienta representa gráficamente las ontologías, las almacena y posteriormente manipula en una base de datos relacional. Su interfaz permite incorporar plugins y exporta a la mayoría de lenguajes de ontologías. Está basado en la metodología *On-To-Knowledge*.
- KAON [Obe04, Obe03]. Editor de ontologías para su creación y mantenimiento, a través de un navegador web. Sólo representa RDF(s). Almacena la información en una base de datos.
- WebOnto [Uni07a]. Representa las ontologías usando una interfaz gráfica a través de la web usando de diagramas. Permite el trabajo colaborativo pero no permite crear o modificar metadatos. Dispone en abierto de una base de ontologías para reutilizar.
- OilEd [Bec07b]. Editor de ontologías que usa el lenguaje OIL (precursor del OWL). No es una herramienta que permita desarrollar todo el proceso de vida de desarrollo de una ontología, pero permite definir axiomas, importar y exportar a la mayoría de los lenguajes conocidos.
- OntoLingua [Uni07d]. Permite realizar trabajo colaborativo, navegar, crear, editar, modificar, utilizar y reutilizar ontologías a través de un navegador web. Dispone también de librerías.
- Chimaera [Uni07b]. Software que guia al usuario para crear y mantener ontologías distribuidas a través de la web. Su objetivo principal es el de combinar ontologías y analizar una o varias. Soporta la carga de bases de conocimiento en diferentes formatos, resolviendo conflictos de nombres, permite navegar entre ontologías, editar términos, etc.
- DL-Workbench [Org07]. Plataforma de edición de ontologías que consta de tres módulos: una API para definir los metamodelos (que describen los formalismos ontológicos), una interfaz para definir ontologías basadas en los metamodelos (implementado como una extensión (*plug-in*) de la plataforma Eclipse). El último módulo personaliza la interfaz de usuario. Usa el lenguaje DAML+OIL, permite la gestión de varias ontologías, y la integración de las mismas con otros datos en una aplicación distribuida o independiente.
- OntoStudio [Ont07b]. Herramienta de desarrollo de ontologías profesional y que da soluciones basadas en la administración de ontologías. Basado en un diseño modular que soporta la incorporación de extensiones (*plug-ins*). Utiliza el Eclipse como entorno. Incorpora herramientas para hacer

correspondencias con documentos, archivos y aplicaciones. Tiene editor de reglas. Soporta visualización, y tiene interfaz Web.

- WSMO Studio [wg08] Entorno que permite representar ontologías para la Web Semántica.Admite extensiones (*plug-ins*) y esta implementado para el entorno Eclipse. Además esta orientado a la creación de servicios en la Web Semántica, generar anotaciones, etc.

- POWL [Aue07], Plataforma de desarrollo de la Web Semántica. Editor para generar ontologías en OWL y RDF, permite trabajo colaborativo y desarrollo distribuido de ontologías. Es una solución de código abierto basada en Web.

- Apollo [Uni07c]. Editor de ontologías que las organiza las ontologías jerárquicamente.Permite incluir plugins. Representa la información utilizando el lenguaje OCLM, y CLOS únicamente. No exporta a ningún lenguaje.

Todos las herramientas anteriores son entornos de gestión de ontologías completos y generalistas, que con mayor o menor funcionalidad permiten realizar un gran número operaciones básicas en el proceso de desarrollo de las mismas. Existen otras herramientas que cubren solamente algunos aspectos de este proceso de desarrollo (por lo que siguen encajando en la clasificación de *Herramientas para el Desarrollo de Ontologías*). Podemos encontrar en este nuevo grupo desde simples editores, traductores a otros lenguajes, o clientes para incluir contenidos en la Web Semántica, entre otras. A continuación se muestran algunas de estas herramientas,

Como editores encontramos:

- RDFedt. Permite generar documentos RDFS.

- IsaViz. Permite generar ontologías en RDFS, mediante interfaz visual, y gráficos.

- OntoLingua. Permite realizar trabajo colaborativo, herramientas de tutoría, librerías, y ontologías reutilizables. Se usa a través de un navegador Web.

- ICOM. Herramienta que puede usarse para la representación de conocimiento en una base de datos o en una ontología. Representa la información usando XML o UML.

- k-Infinity. Sólo representa ontologías en RDF con un editor gráfico. Almacena la información en una BD.

- OntoSaurus. Representa la información en el lenguaje LOMM. Tiene una interfaz web pero no tiene visualización gráfica. No exporta en lenguajes web.

- SWOOP. Herramienta que permite representar-visualizar ontologías en OWL. Lleva incluido un programa que valida la especie de OWL con la que se trabaje.

- Amaya. Es un editor web, para crear ontologías. Proporciona un espacio para la colaboración y permite representar lenguajes web como RDF, y otros.

Como traductores:

- DOE. Herramienta que permite exportar e importar varios formatos. No tiene interfaz de usuario.

- Medius VOM. La representación de ontologías se realiza mediante diagramas UML, permite importarlas y exportarlas en lenguajes Web estándar.

- LinkFactory. No tiene interfaz, pero permite soporte multiusuario, exportar e importar en los lenguajes web estándar. Almacena la información en una BD.

Como clientes de la Web Semántica:

- DBIN. Es un cliente que permite construir y publicar, un espacio semántico personal donde hay reglas definidas por el usuario, métricas de confianza y filtros. Permite la generación de anotaciones.

- Schema Web. Directorio de esquemas en RDF, OWL, etc. para que los desarrolladores trabajen con RDF.

- Proton, plataforma creada para realizar anotaciones semánticas, indexado y recuperación de información. Desarrollo para la ontología KIMO, incluida en el proyecto KIM.

Existen otro tipo de herramientas para la gestión/representación de ontologías, como *Sesame, Inkling, rdfDB, Redland, JENA y Cerebra*, que son herramientas para almacenar ontologías y consultarlas. Algunas de estas, incluso proporcionan una API, que permite utilizar sus propios métodos para desarrollar una herramienta propia de gestión. La mayoría de estas APIs permiten manipular ontologías definidas en un lenguaje como RDF(S) u OWL.

Algunas de estas interfaces son Sesame [Kam07], JENA [Pro07] o OWLAPI [Bec07a].

Sin embargo las herramientas de representación de ontologías más generalistas, tienen la ventaja con respecto a otros mecanismos de representación, como los lenguajes, la facilidad de uso, de aprendizaje gracias a la intuitividad de la herramienta, la posibilidad de visualización de la información en forma de diagramas, o de manera organizada, la posibilidad de la definición de las ontologías a través de una interfaz visual y de marcos. Estas ventajas, convierten en favoritos estos mecanismos a la hora de definir ontologías, con respecto a aquellos que consisten en la definición de las mismas utilizando un editor de texto para representar la información en el lenguaje de definición pertinente.

Herramientas como Chimaera, FCA-merge y iPROMPT (Protege), OntoBuilder, OntoStudio (KAON2) se usan para unificar ontologías o integrarlas, entrarían dentro de las consideradas *Herramientas de Integración y Alineamiento.*

Las *Herramientas de Anotación* son aquellas que se usan para anotar ontologías, las mas conocidas son AeroDamL, COHSE, MnM y OntoAnnotate, SMORE, etc.

Además de todas estas herramientas, existen otro conjunto de ellas que permiten razonar o deducir información representada en las mismas, algunas de estas herramientas son: Racer, FaCT ++, F-OWL, Pellet, Jena, OWLJessKB, etc.

Con respecto a las *Herramientas para la Evaluación*, usadas para validar la consistencia de la ontología, existen algunas páginas web como la de WonderWeb [Bec07c] y otras herramientas como ODEval, u OntoManager [Har05].

Las *Herramientas para Generar Ontologías* serían aquellas que a partir de una información ya existe (ya sean textos, esquemas, BD) nos permiten generar una nueva. Con respecto a textos, podemos encontrar: Asium, LTG, OntoLearn, SOAT, TextStorm, Text-to-Onto, TERMINAE, Camaleon, etc. (Un resumen más detallado se encuentra en et al. [GP04]). La generación de ontologías utilizando una BD estaría detallada en la sección 2.3.

Las *Herramientas de Evolución y Versionado*, sirven para representar aquella parte de la realidad que está continuamente cambiando y es necesaria entonces, una buena gestión de las versiones que se van obteniendo de la misma. Gómez-Pérez et al. en [Cor06] hace un repaso por las herramientas más populares en este ámbito, como el plugin KAON [Obe03] y el algoritmo PromptDiff implementado en la API con el mismo nombre. Además también existen en algunas herramientas de propósito general que controlan el versionado de las detalladas previamente.

En cuanto a las herramientas que permiten la *Interacción con BBDD*, son tan numerosas y están tan relacionadas con la temática de esta investigación que se pueden encontrar de forma detallada en la sección 2.3.

A.2.4. Técnicas de Manipulación de Ontologías

Alrededor del concepto de ontología, existen un gran número de operaciones, que permiten la manipulación de las mismas, de tal forma que puedan relacionarse entre ellas, definir nuevas, compartir información, extender su definición, refinarla, etc. Todas estas técnicas, que forman parte de la ingeniería ontológica, tienden a resultar confusas, y a veces en la literatura se no deja claro cual es su función. A continuación se hará un repaso por todas ellas para clarificar el uso de estos términos en este trabajo [Ehr07]:

Mapeo (*Mapping*) Trata de relacionar conceptos similares o relaciones de diferentes fuentes usando relaciones de equivalencia [Car07]. Es la más utilizada, puesto que su uso fundamentalmente es la consulta de diferentes ontologías. Un mapeo entre ontologías representa una función entre las mismas. La ontología original no se cambia pero se añaden acciones adicionales de mapeo que expresen los conceptos, relaciones o instancias en términos de la ontología secundaria. Estos mapeos se almacenan en un lugar diferente y podrán usarse sólo en una dirección. El uso típico de un mapeo es en una consulta sobre una ontología que se reescribe y maneja sobre otra ontología. Las respuestas entonces son mapeadas de nuevo. Mientras que la alineación identifica la relación entre las ontologías, los mapeos se centran en la representación y la ejecución de las relaciones para una cierta tarea. Tiene ciertas similitudes con el concepto de correspondencia como veremos a continuación.

Correspondencia (Matching) Proceso de buscar relaciones o correspondencias entre entidades en diferentes ontologías. Estas correspondencias, a veces se denominan mapeos en algunos textos [Euz07]. La diferencia más significativa con el proceso de Mapeo es que el mapeo es como una alineación pero que esta orientada y dirigida, o sea que mapea las entidades de una ontología a al menos otra entidad de otra ontología, también puede verse como una colección de reglas de mapeo todas orientadas en una dirección.

Integración (*Integrating*) Consiste en juntar, extender o especializar una ontología usando otras que hay disponibles. Es decir, para integrar una o más ontologías tienen que ser utilizadas para formar una nueva. Los conceptos originales son adoptados, y posiblemente extendidos, pero su

origen se mantiene (por ejemplo a través del espacio de nombres). Las ontologías son integradas, no están completamente mezcladas. Este concepto se utiliza fundamentalmente si las ontologías provienen de diferentes dominios. A través de la integración, la ontología final, cubrirá un dominio mayor. Las operaciones más comunes en esta práctica son la unión y la intersección.

Mezcla (*Merging*) Toma diferentes ontologías del mismo campo y crea una nueva unificada. En este caso, la nueva ontología reemplazará a las originales. Esto requiere de una adaptación y extensión considerable. Los elementos individuales de las ontologías originales se presentan dentro de la nueva ontología, pero no pueden volver a sus fuentes. El alineamiento es un paso previo a esta técnica, puesto que detecta la coincidencia de elementos.

Alineación (*Alignment*) , consiste en traer dos o más ontologías en un consenso mutuo y hacerlas consistentes y coherentes. Es decir, dadas dos ontologías, alinear una ontología con otra significa que para cada entidad (concepto, relación o instancia) en la primera ontología, trataremos de encontrar una entidad correspondiente que tenga el mismo significado en la segunda ontología. Es entonces un Alineamiento una relación de igualdad uno-a-uno [Ehr07].

Mediación (*Mediation*) Es el nivel más alto para conciliar diferencias entre ontologías heterogéneas con el objetivo de conseguir la interoperación entre las fuentes de datos anotadas con ellas y las aplicaciones que usan estas ontologías. La mediación incluye el descubrimiento y la especificación de alineamientos ontologías, además del uso de estas alineaciones para ciertas tareas como el mapeado para una reescritura de una consulta y la transformación de la instanciación. Además el termino mediación incluye la mezcla de ontologías.

Combinación (*Combining*) Es cuando dos o más ontologías diferentes se utilizan para una tarea en la que su relación mutua es relevante. La relación de combinación puede ser de cualquier tipo, no únicamente de identificación. Además no se da información acerca de como se establece la relación.

Transformación (*Transformation*) Cuando se transforma una ontología su semántica cambia para hacerla compatible con los nuevos propósitos. Por ejemplo, las relaciones entre las entidades de la primera ontología y aquellas de la segunda se añaden a la primera.

Traducción (*Translating*) Este es el caso de traducir una ontología representada en un lenguaje a otro. La semántica debe ser preservada, aunque cambia la sintaxis.

Versionado (Versioning) Es el proceso de ir aplicando modificaciones a una ontología e ir guardando los resultados parciales que se van obteniendo.

Conciliación (Reconciliation) Es el proceso que permite armonizar los contenidos de dos o más ontologías, típicamente requiere cambios en uno de los lados incluso en los dos. En este caso no se combinan las ontologías, sino que co-evolucionan. La conciliación de ontologías puede ser llevada a cabo con el propósito de mezclar dos ontologías o bien hacerlas independientes.

Recortado (*Pruning*) Es el proceso de recortar partes de una ontología para formar una nueva más especializada. A veces, se parte de una ontología más genérica o de alto nivel, y a partir de la misma se van realizando operaciones de poda para obtener la ontología adecuada a la realidad que se desea representar.

Aprendizaje(*Learning*) Es el proceso de obtener una ontología, a través de métodos o herramientas, de forma automática o semiautomática. Las fuentes de datos de las cuales se puede extraer una ontología aplicando técnicas de aprendizaje, son textos, esquemas, contenidos web, bases de datos, etc.

Tabla A.3: Elementos de una página OWL

Elemento	**Descripción**
Prefijo	*xmlns:owl*
Espacio de Nombres	*http://www.w3.org/2002/07/owl#*
owl:class	Identificador de clase. En OWL-FULL es igual que *rdfs:class*)
owl:oneOf	enumeración de individuos
owl:allValuesForm, *owl:someValuesFrom,*	Restricción de valores en un propiedad
owl:cardinality, *owl:minCardinality,* *owl:maxCardinality*	Restricción de Cardinalidad en una propiedad
owl:intersectionOf, owl:unionOf, *owl:complementOf*	Descripciones que usan expresiones booleanas
owl:subClassOf, *owl:equivalentClass,* *owl:disjointWith*	Axiomas de Clases
owl:ObjectProperty	Propiedades de objeto
owl:DatatypeProperty	Propiedades de tipo de datos
owl:equivalentProperty, *owl:inverseOf,* *owl:equivalentProperty*	Axiomas de propiedades para las relaciones entre propiedades
owl:FucntionalProperty y *owl:InverseFuctionalProperty*	Axiomas de propiedades para la cardinalidad
owl:SymmetricProperty y *owl:TransitiveProperty*	Axiomas de propiedades para características lógicas
owl:sameAs, owl:differentFrom	Especifican la identidad de los individuos
rdfs:Literal	Tipos de datos:literales
owl:OneOf	tipos de datos enumerados
owl:versionInfo, rdfs:label, *rdfs:comment, rdfs:seeAlso,* *rdfs:isDefinedBy*	Propiedades de anotación
owl:imports, owl:priorVersion, *owl:backwardCompatibleWith,* *owl:incompatibleWith*	Propiedades de la ontología

Apéndice B

Extensiones Difusas al Modelo Relacional de BD

B.1. Modelo Generalizado para Bases de Datos Relacionales Difusas (GEFRED)

A continuación se expone brevemente los fundamentos planteados en GEFRED.

B.1.1. Fundamentos Teóricos de GEFRED

Representación de Datos Imprecisos

Definición B.1. *Sea D un dominio de discurso, $\tilde{\mathcal{P}}(D)$ el conjunto de distribuciones de posibilidad definidas sobre D, entre las que se incluyen aquellas que describen los valores* DESCONOCIDO *y* NO APLICABLE. *Consideremos también el valor* NULL. *El* dominio difuso generalizado *se define como D_G donde $D_G \subseteq \tilde{\mathcal{P}}(D) \cup NULL$.*

Un dominio difuso generalizado es un conjunto formado por elementos que pueden ser:

1. *un escalar (por ejemplo, Comportamiento = bueno, que se representa mediante la distribución de posibilidad $\{1/buena\}$),*

2. *un número (por ejemplo, Edad = 27, que se representa mediante la distribución de posibilidad $\{1/27\}$),*

3. *un conjunto de asignaciones escalares posibles (por ejemplo, Comportamiento = $\{bueno, malo\}$, que se representa por la distribución de posibilidad $\{1/good, 1/bad\}$),*

4. *un conjunto de asignaciones numéricas posibles (por ejemplo, Edad* $= \{20, 21\}$*, que se representa mediante la distribución de posibilidad* $\{1/20, 1/21\}$*),*

5. *una distribución de posibilidad construida sobre un dominio escalar (por ejemplo, Aptitud* $= \{0{,}6/buena, 0{,}7/mala\}$*),*

6. *una distribución de posibilidad construida sobre un dominio numérico (por ejemplo, Edad* $= \{0{,}3/20, 0{,}5/21\}$*, números difusos o etiquetas lingüísticas),*

7. *un número real en en intervalo* $[0, 1]$*, que se refiere al grado de acoplamiento (por ejemplo, Calidad* $= 0{,}9$*),*

8. *un valor* DESCONOCIDO *con distribución de posibilidad* $\{1/u : \forall u \in U\}$*,*

9. *un valor* NO APLICABLE *con distribución de posibilidad* $\{0/u : \forall u \in U\}$*,*

10. *un valor* NULL *con distribución de posibilidad*

$$NULL = \{1/UNDEFINED, 1/NULL\}$$

Definición B.2. *Una* Relación Difusa Generalizada*, R, es un par de conjuntos* $(\mathcal{H}, \mathcal{B})$*, definidos como sigue:*

- $\mathcal{H}$ *es el conjunto llamado "cabecera" y describe la estructura de la relación mediante un conjunto de ternas atributo-dominio-compatibilidad (donde el último es opcional),*

$$\mathcal{H} = \{(A_{G1} : D_{G1}[, C_{A_{G1}}]), \ldots, (A_{Gn} : D_{Gn}[, C_{A_{Gn}}])\}$$

  donde a cada atributo A_j*, le subyace un dominio difuso generalizado, no necesariamente distinto,* $D_j, j \in [1, n]$*.* C_j *es el llamado atributo de compatibilidad y toma valores en* $[0, 1]$*.*

- $\mathcal{B}$ *es el conjunto llamado "cuerpo" y está formado por una serie de tuplas generalizadas difusas distintas, donde cada tupla está compuesta por un conjunto de ternas atributo-valor-grado (donde este último es opcional),*

$$\mathcal{B} = \{\{(A_{G1} : \tilde{d}_{i1}[, c_{i1}], \ldots, (A_{Gn} : \tilde{d}_{in}[, c_{in}])\}\}$$

  con $i = 1, \ldots, m$ *y donde* m *es el número de tuplas de la relación,* $\tilde{d_{ij}}$ *representa el valor del dominio que toma la tupla* i *sobre el atributo* A_j *y* c_{ij} *es el grado de compatibilidad asociado a este valor.*

Los operadores de comparación tienen que ser flexibilizados de modo que sea posible comparar dos valores que no son exactamente iguales.

Definición B.3. *Sea U el dominio de discurso considerado. Se llama* comparador extendido *θ a cualquier relación difusa definida sobre U y expresada como sigue:*

$$\theta : U \times U \to [0,1]$$
$$\theta(u_i, u_j) \mapsto a$$

con $u_i, u_j \in U$ y $a \in [0,1]$.

Definición B.4. *Sea U un dominio de discurso, D un dominio difuso construido sobre el mismo y θ un comparador extendido definido sobre U. Consideremos una función Θ^θ definida como sigue:*

$$\Theta^\theta : D \times D \to [0,1]$$
$$\Theta^\theta(\tilde{d_1}, \tilde{d_2}) \mapsto [0,1]$$

Se dice que Θ^θ es un comparador difuso generalizado *sobre D inducido por el comparador extendido θ, si cumple:*

$$\Theta^\theta(\tilde{d_1}, \tilde{d_2}) = \theta(d_1, d_2), \forall d_1, d_2 \in U$$

donde $\tilde{d_1}$ y $\tilde{d_2}$ representan las distribuciones de posibilidad $\{1/d_1\}$ y $\{1/d_2\}$ inducidas, respectivamente, por los valores d_1 y d_2.

Manejo de Datos Imprecisos

Definición B.5. *Sea R una relación difusa generalizada como la de la definición B.2 y X un subconjunto de $\mathcal{H}$ definido como sigue:*

$$x \subseteq \mathcal{H}, x = \{(A_s : D_s[, C_{s'}]) : s \in S, s' \in S'; S, S' \subseteq \{1, \ldots, n\}\}$$

Entonces, se llama proyección difusa generalizada *de R sobre X, y se nota por $\mathcal{P}_X(R)$, a una relación difusa generalizada de la forma:*

$$\mathcal{P}_X(R) = \begin{cases} \mathcal{H}_\mathcal{P} = X \\ \mathcal{B}_\mathcal{P} = \{(A_s : \tilde{d}_{is}[, c_{is'}])\} \end{cases} \tag{B.1}$$

donde $s \in S$, $s' \in S'$ y $S, S' \subseteq \{1, \ldots, n\}$.

Definición B.6. *Sea R una relación difusa generalizada como la de la definición B.2, $\tilde{a} \in D$ una constante, Θ^θ un comparador difuso generalizado y γ un "umbral de cumplimiento". Entonces, se llama* selección difusa generalizada *sobre la relación R inducida por Θ^θ compuesto con $\tilde{a}$ y el atributo*

$A_k, k \in \{1, \ldots, n\}$ y cualificada por γ, y se nota por $\mathcal{S}_{\Theta^\theta(A_k,\tilde{a})\geq\gamma}(R)$ a la relación difusa generalizada de la forma:

$$\mathcal{S}_{\Theta^\theta(A_k,\tilde{a})\geq\gamma}(R) = \begin{cases} \mathcal{H}_\mathcal{S} = \{(A_1 : D_1[, C_{A_1}]), \ldots, (A_n : D_n[, C_{A_n}])\} \\ \mathcal{B}_\mathcal{S} = \{\{(A_1 : \tilde{d}_{r1}[, c_{r1}], \ldots, (A_k : \tilde{d}_{rk}[, c'_{rk}]), \\ \qquad \ldots, (A_n : \tilde{d}_{rn}[, c_{rn}])\}\} \end{cases} \tag{B.2}$$

con

$$c'_{rk} = \Theta^\theta(\tilde{d}_{rk}, \tilde{a}) \geq \gamma \tag{B.3}$$

donde $r = 1, \ldots, m'$ con m' el número de tuplas de la selección.

B.1.2. Representación Relacional de un Dominio Generalizado Difuso: FIRST

Representación de la Información Imprecisa

Los elementos que forman parte del tratamiento impreciso pueden ser representados de diversas maneras. De ese modo, una distribución de posibilidad normalizada puede representarse mediante parábolas, hipérbolas, etc. Sin embargo, la implementación FIRST [Med94a, Gal99] y su servidor de consultas imprecisas, construidos sobre el modelo GEFRED [Med94b], asume la representación trapezoidal descrita por cuatro puntos que se muestra en la figura B.1. Esta simplificación se explica en función de la contradicción que supone representar datos intrínsecamente imprecisos mediante distribuciones de posibilidad definidas de forma altamente precisa, que además añaden el factor del incremento del coste computacional.

Datos difusos o con tratamiento difuso Los valores que pueden formar parte de un dominio generalizado difuso pueden dividirse en dos grupos:

1. **Datos precisos:** también llamados *crisp* o *clásicos*. Según se muestra en la figura B.2 y dado que lo que se almacena son datos clásicos, el almacenamiento dependerá directamente de la capacidad de representación del SGRBD sobre el que se aplique la implementación.

2. **Datos imprecisos:** también llamados *fuzzy* o *difusos*. Se corresponden con datos de dos subtipos recogidos en las figuras B.3 y B.4:

   - *Datos imprecisos sobre un referencial ordenado*: que engloban a todos aquellos datos descritos mediante una distribución de posibilidad construida sobre un conjunto referencial discreto o continuo ordenado (con una relación de orden definida). A este tipo pertenecen

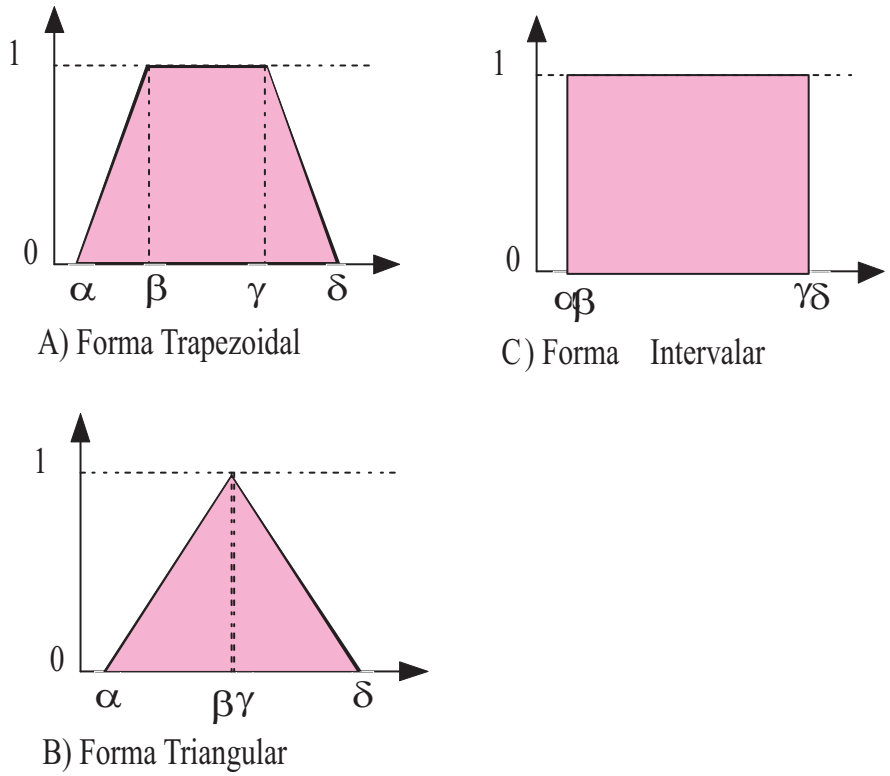


Figura B.1: Posibles Representaciones trapezoidales de una distribución de posibilidad

los valores de tipo 6 que aparece en la definición de dominio difuso generalizado B.1 (página 221). Para su representación recurriremos a:

*Distribución de Posibilidad Trapezoidal***:** cuya función de pertenencia viene descrita por cuatro puntos $[\alpha, \beta, \gamma, \delta]$ que se muestran en la figura B.1 apartado A. Aquellos valores que estén en el intervalo $[\beta, \gamma]$ pertenecen al conjunto difuso descrito por la distribución trapezoidal con grado de pertenencia 1.

*Etiqueta lingüística***:** estos se refieren a un concepto impreciso introducido por Zadeh [Zad75] definido mediante una distribución de posibilidad.

*Valores Aproximados***:** sea n un valor del dominio subyacente, el concepto impreciso *aproximadamente* n se construye a partir de un valor llamado *margen* el cual nos permite constru-

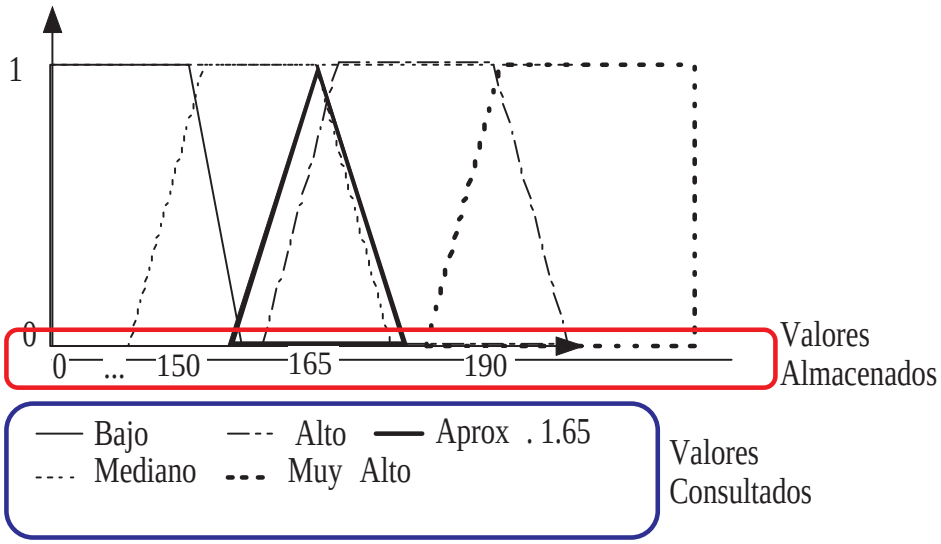


Figura B.2: Tipo difuso 1

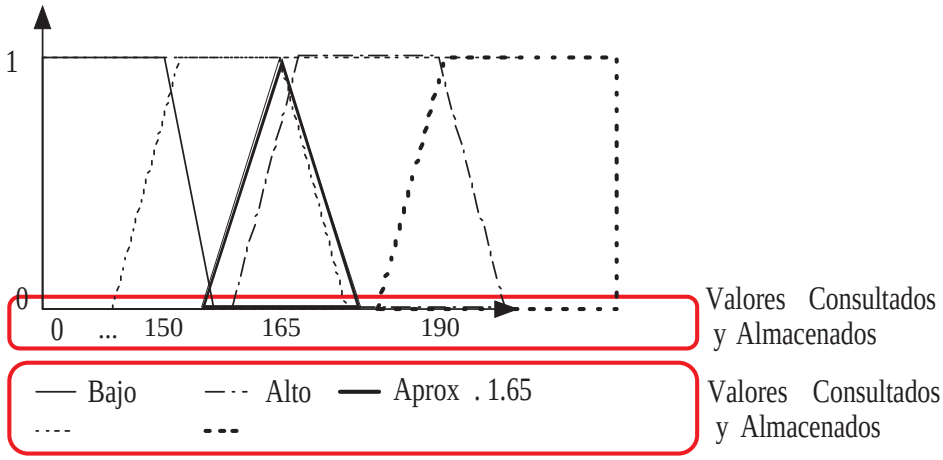


Figura B.3: Tipo difuso 2

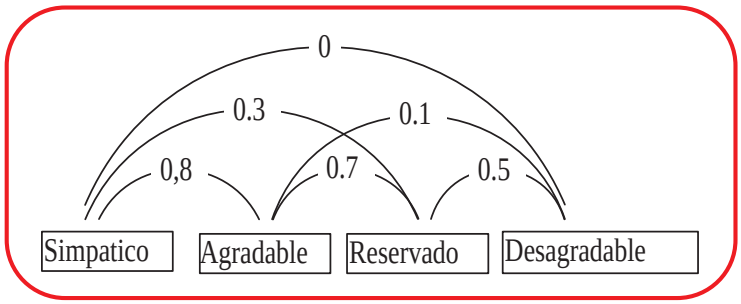


Figura B.4: Tipo difuso 3. Valores que pueden tomar las relaciones de similitud.

ir una distribución de posibilidad triangular de la forma $[n - margen, n, n, n + margen]$, como se muestra en la figura B.1.

Intervalos de posibilidad: los únicos valores que pertenecen al conjunto difuso con grado 1 son aquellos del dominio subyacente que están en el intervalo $[n, m]$, por lo que se representa mediante una distribución con los siguientes parámetros $[n, n, m, m]$ que se puede ver en la figura B.1 apartado C. Esta representación permite asumir la extensión que Grant [Gra80] hace del modelo relacional para representar valores intervalares como valor posible de un atributo.

- *Datos imprecisos con analogía sobre referencial no ordenado*: que engloban a todos aquellos datos difusos cuyo dominio subyacente es un conjunto discreto no ordenado sobre el que se define una *relación de semejanza* o similitud entre cada par de valores del mismo. Los datos que pueden representarse en este grupo son:

  Escalares Simples: este tipo está representado por una única pareja de datos en la que el único valor de posibilidad es $\{1/d\}$, lo que quiere decir que el único valor posible es d con grado de posibilidad 1 (normalización).

  Distribución de Posibilidad sobre Escalares: este tipo se le asocia una representación del tipo $\{(p_1, d_1), \ldots, (p_n, d_n)\}$ en la que a cada valor del dominio subyacente se le asigna un grado de pertenencia al conjunto difuso.

Hay que proporcionar una representación para los tres valores especiales *Null*, *Unknown* y *Undefined* cuyas distribuciones de posibilidad se ven en la figura B.5.

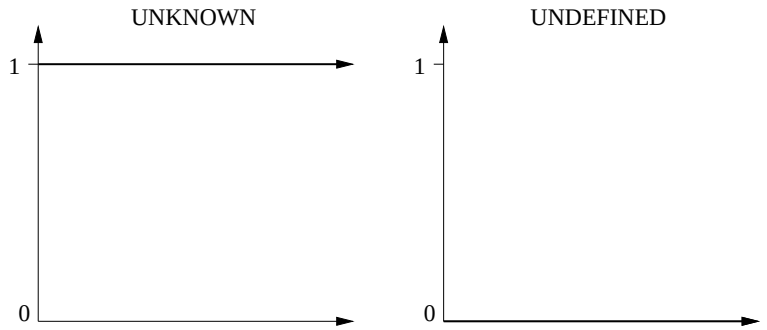


Figura B.5: Distribuciones de posibilidad de los valores *Unknown* y *Undefined*

Para recoger todos los valores que se pueden formar parte de un dominio generalizado difuso Medina [Med94b], y Galindo [Gal99] posteriormente, plantearon tres tipos de atributos en su implementación FIRST del modelo GEFRED, sobre el sistema gestor relacional de bases de datos Oracle®. A grandes rasgos, estos atributos son:

- *Tipo difuso 1*: representa datos precisos que pueden ser consultados de forma imprecisa.

- *Tipo difuso 2*: representa datos imprecisos pertenecientes a un dominio difuso construido sobre un referencial ordenado y que pueden ser consultados de forma imprecisa.

- *Tipo difuso 3*: representa datos imprecisos pertenecientes a un dominio difuso construido sobre un referencial discreto no ordenado, sobre el que se define una relación de similitud, y que pueden ser consultados de forma imprecisa.

En el Modelo Relacional los atributos toman valores en dominios de valores atómicos (enteros, reales, cadenas de caracteres, caracteres, ...) que son las unidades mínimas de información. Dicho modelo no especifica los dominios, pero cualquier *Sistema Gestor de Bases de Datos* construido sobre este paradigma (excepción hecha de aquellos que poseen características de orientación a objetos, que no es nuestro caso) usa únicamente valores atómicos.

En el caso de los atributos de tipo difuso 1, no es necesaria ninguna estructura adicional para la representación de valores, ya que no es la representación lo que se flexibiliza, sino la consulta.

En el caso de los atributos de tipo 2 y tipo 3, no es sólo la consulta la que se flexibiliza, sino también la representación de valores. Galindo propone en [Gal99] una estructura para la representación de atributos de los tipos 2 y 3, que pueden verse en las tablas B.1 y B.2.

Tipos de valor	**Atributos de tabla para valores de tipo difuso 2**				
	FT	**F1**	**F2**	**F3**	**F4**
DESCONOCIDO	0	NULL	NULL	NULL	NULL
NO DEFINIDO	1	NULL	NULL	NULL	NULL
NULL	2	NULL	NULL	NULL	NULL
CLASICO	3	d	NULL	NULL	NULL
ETIQUETA	4	FUZZY_ID	NULL	NULL	NULL
INTERVALO$[n, m]$	5	n	NULL	NULL	m
APROX(d)	6	d	$d - margen$	$d + margen$	$margen$
TRAPEZOIDE$[\alpha, \beta, \gamma, \delta]$	7	α	β	γ	δ

Tabla B.1: Representación para atributos de tipo 2

En la tabla B.1, el atributo FT almacena el tipo de valor y los atributos $F1$, $F2$, $F3$ y $F4$ se usan para almacenar los parámetros de cada valor. Los valores NULL que aparecen en esta tabla representan *valores inaplicables*.

Tipos de valor	**Atributos de tabla para valores de tipo difuso 3**					
	FT	**FP1**	**F1**	...	**FPn**	**Fn**
DESCONOCIDO	0	NULL	NULL	...	NULL	NULL
NO DEFINIDO	1	NULL	NULL	...	NULL	NULL
NULL	2	NULL	NULL	...	NULL	NULL
SIMPLE	3	p	d	...	NULL	NULL
DISTRIBUCIÓN	4	p_1	d_1	...	p_n	d_n

Tabla B.2: Representación para atributos de tipo 3

En la tabla B.2, el atributo FT almacena el tipo de valor y los pares de atributos (p_n, d_n) representan que el valor d_n del dominio tiene un grado de posibilidad p_n. Los valores NULL que aparecen en esta tabla representan *valores inaplicables*.

De este modo, cuando se quiera representar un atributo de tipo 2 en una relación, serán necesarias cinco atributos de tipo básico. Para representar un atributo de tipo 3, serán necesarias $2 \times n + 1$ atributos de tipo básico, donde n es el número de valores que forman el dominio en cuestión.

Comparadores difusos generalizados En la literatura pueden encontrarse diversos métodos de comparación de número difusos, los cuales pueden

clasificarse en dos categorías: los que usan una función que va del conjunto del números difusos a un conjunto ordenado y los que usan relaciones difusas para el proceso de comparación. Las primer tipo pertenecen a las propuestas recogidas en [Ada80, Yag78, Yag81]. Sobre el segundo tipo se encuentran diferentes aproximaciones en [Bal79, Bas77, Del88, Dub83].

GEFRED permite representar una amplia variedad de comparadores, sin embargo FIRST se centra en 15 comparadores, mostrados en la figura B.7 y se aplican a valores de dominios difusos construidos sobre referenciales ordenados.

Un desarrollo de todos ellos y su significado se desarrolla en [Gal99].

Grado de cumplimiento de una condición: umbral El grado de cumplimiento de una consulta que implique comparadores difusos generalizados pertenece al intervalo $[0, 1]$. Sin embargo, el umbralizar una consulta permite hacer poda de todos aquellas tuplas que no superen dicho grado de cumplimiento.

En el caso de un mecanismo de deducción orientado a tuplas, el grado obtenido desde el comienzo del cálculo de predicados de una regla sometido a un umbral puede ayudar a realizar una poda importante durante la exploración de las posibilidades.

B.1.3. Base de Metaconocimiento Difuso (FMB)

Toda la información adicional sobre la estructura de los dominios y los valores que puede tomar cada atributo construido sobre un dominio generalizado difuso, así como cualificadores para el acceso a estos atributos constituye lo que se conoce como *Base de Metaconocimiento Difuso*. En la figura B.6 se describen el diagrama de clases de dicha base de metaconocimiento.

Las relaciones que constituyen dicha estructura tienen la siguiente finalidad:

- *FUZZY_COL_LIST*: contiene la lista de aquellos atributos de tabla de la base de datos que son susceptibles de tratamiento difuso. Cada atributo queda descrito en esta por una referencia a la tabla a la que pertenecen (OBJ#) y columna en la que se almacenan (COL#), el tipo difuso de la columna (F_TYPE), el número de valores de la distribución de posibilidad si se trata de un atributo de tipo difuso 3 (LEN) y un comentario acerca del atributo (COM).

- *FUZZY_OBJECT_LIST*: almacena los objetos difusos que pertenecen al dominio del atributo en cuestión. Cada uno de los objetos están representados por la tabla y columna a cuyo dominio pertenecen (OBJ# y COL#), un identificador (FUZZY_ID), un nombre (FUZZY_NAME) y un tipo de objeto (FUZZY_TYPE).

- *FUZZY_LABEL_DEF*: contiene información sobre las distribuciones de posibilidad trapezoidales que se asocian a etiquetas lingüísticas. Cada una de ellas está descrita por la tabla y columna a cuyo dominio pertenece (OBJ# y COL#), la etiqueta a la que se asocia (FUZZY_ID) y los parámetros que la definen α, β, γ y δ.

- *FUZZY_APPROX_MUCH*: contiene los parámetros *margen* y *much* para cada atributo de tipo difuso 1 o 2 los cuales se usan para la comparación de valores dentro del dominio difuso. Para cada columna (OBJ#,COL#) se almacenan dos atributos (MARGEN, MUCH).

- *FUZZY_NEARNESS_DEF*: contiene los valores de similitud entre cada par de valores posible en el dominio discreto de valores escalares que se asocia a un atributo de tipo difuso 3. A cada par de valores posible (FUZZY_ID1, FUZZY_ID2) del dominio discreto de escalares de un atributo de tipo difuso 3 (OBJ#, COL#) hay asociado un grado de compatibilidad (DEGREE).

- *FUZZY_COMPATIBLE_COL*: contiene información sobre aquellos atributos de tipo difuso 3 que comparten dominio con otro atributo difuso del mismo tipo, de modo que no sea necesario volver a definir todos los valores de dicho dominio y sus grados de compatibilidad. Para cada atributo difuso de tipo 3 que comparte dominio discreto de escalares con otro atributo (OBJ#1, COL#1) se almacena la referencia al atributo con el que comparte dominio (OBJ#2, COL#2).

- *FUZZY_QUALIFIERS_DEF*: contiene el umbral mínimo de satisfacción para cada cualificador (QUALIFIER) definido sobre una etiqueta lingüística (FUZZY_ID) que pertenece al dominio de un atributo difuso (OBJ#, COL#).

B.1.4. Lenguaje SQL Difuso (*FSQL*): consulta imprecisa

Una vez definido el dominio generalizado difuso es necesario ampliar los lenguajes relacionales que gestionan estos dominios así como las relaciones en el seno del modelo relacional. En nuestro caso, nos centraremos en una ampliación del lenguaje SQL.

La extensión del SQL para permitir la representación de datos difusos comprende tanto el Lenguaje de Definición de Datos (DDL) como el Lenguaje de Manipulación de Datos (DML), dando lugar a la creación y extensión de las sentencias que se exponen en la tabla B.3. Esta sintaxis puede verse con detalle en [Bla00b, Bla00a, Bla01].

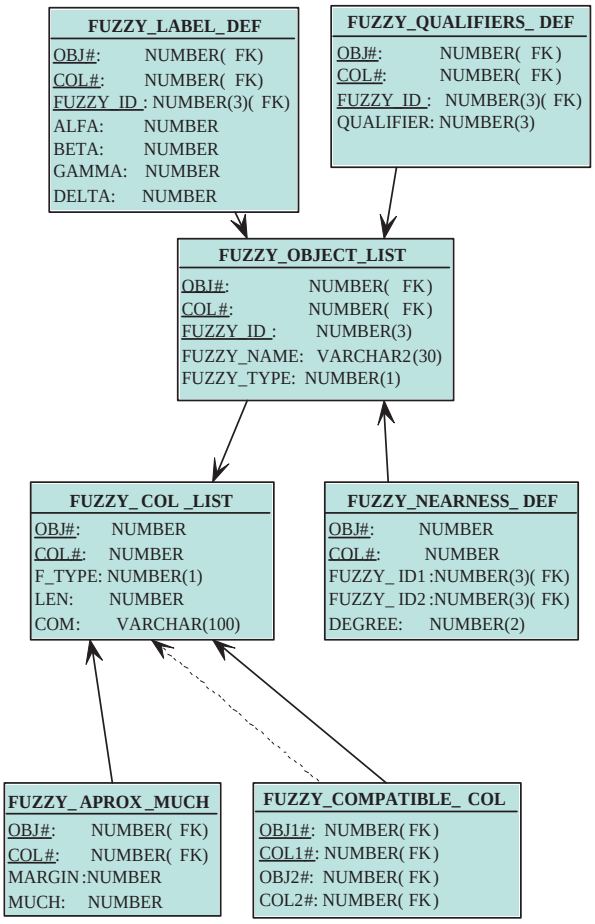


Figura B.6: Estructura relacional de la FMB

Tabla B.3: Resumen de FSQL

Tipo	Sentencia	Descripción
DDL	Create_Table	Crea una relación difusa
DDL	Create_Label	Crea una etiqueta
DDL	Create_Nearness	Crea los valores de un dominio difuso tipo 3
DDL	Alter_Table	Cambia la definición de una relación difusa
DDL	Alter_Label	Cambia la definición de una etiqueta
DDL	Alter_Nearness	Cambia la definición de un dominio difuso tipo 3
DDL	Drop_Table	Borra de una relación difusa
DDL	Drop_Label	Borra de una etiqueta
DDL	Drop_Nearness	Borra de un dominio difuso de tipo 3
DML	Select	realiza consultas difusas
DML	Insert	inserta tuplas en una relación difusa
DML	Delete	borra tuplas de una relación difusa

B.2. Extensión Lógica-Deductiva al Modelo de BDRD

B.2.1. Fundamentos Teóricos para la Representación del Modelo Lógico y Lógico Difuso para Bases de Datos Relacionales

La teoría de bases de datos esta muy ligada con la Lógica, especialmente a la hora de construir consultas, definir vistas o restricciones de integridad [Gal84].

Sobre las bases de datos deductivas se pueden definir dos tipos de relaciones básicas [Bla01]:

Relación Extensiva en el sentido del Modelo Relacional Clásico consisten en un par de conjuntos (R,r) donde R es un conjunto de atributos que define la estructura de la relación y r es un conjunto de tuplas que define el contenido de la relación.

Relación Intensiva se trata de un par de conjuntos (R,I) donde R es el conjunto de atributos que definen la estructura de la relación e I es un conjunto de fórmulas lógicas, denominado *Generador de Instancias*, que permite obtener el contenido de la relación.

Las reglas en el *Generador de Instancias* I tienen una estructura diferente dependiendo de si se trata del caso clásico o difuso. Una propuesta para el esquema de reglas en el modelo clásico deductivo lo introdujo Medina et al.

[Med97] y Blanco et al. [Bla00a]. Se estableció que cada regla en el conjunto I de la base de datos clásica deductiva tuviera la siguiente estructura:

$$P(X_1, X_2, \ldots, X_n) \longleftarrow Q_1(Y_{1,1}, Y_{1,2}, \ldots, Y_{1,n_1}) \wedge Q_2(Y_{2,1}, Y_{2,2}, \ldots, Y_{2,n_2}) \wedge \\ \wedge \ldots \wedge Q_m(Y_{m,1}, Y_{m,2}, \ldots, Y_{m,n_m}) \tag{B.4}$$

El predicado P, cuyos hechos son calculados, se denomina *cabeza de la regla* y la fórmula lógica a la derecha del :- se denomina *cuerpo de la regla* (se trata de reglas tipo *Prolog*). Además, cada regla tiene impuestas las siguientes restricciones:

1. Cada variable en la cabecera de la regla aparezca por lo menos una vez en el cuerpo de la regla, y
2. Cada variable en el cuerpo de la regla aparezca en el predicado de la cabecera de la regla o en cualquier otro predicado del cuerpo de la regla.

Aplicando estos conceptos al modelo relacional extendido difuso GEFRED se extienden algunos de los conceptos expuestos [Bla01, Bla02a, Bla03a, Bla03b]:

Definición B.7. *Una* Relación Extensiva Difusa *es una Relación Difusa Generalizada desde el punto de vista del modelo GEFRED [Med94b], definido en el apartado anterior, es decir, un par* $(\mathcal{H}, \mathcal{B})$ *donde* $\mathcal{H}$ *equivale al esquema o cabecera de la relación y* $\mathcal{B}$ *es la instancia o cuerpo.*

En nuestro caso, nos centraremos en el caso en el que, cuando un valor de dominio acopla con un atributo dentro de una tupla, lo hace con grado 1, es decir, la expresión para la cabeza y el cuerpo de la relación son:

$$\begin{aligned} \mathcal{H} &= \{(A_{G_1} : D_{G_1}[, C_{A_{G_1}}]), \ldots, (A_{G_n} : D_{G_n}[, C_{A_{G_n}}])\} \\ \mathcal{B} &= \{(A_{G_1} : \tilde{d_{i_1}}[, 1]), \ldots, (A_{G_n} : \tilde{d_{i_n}}[, 1])\} \end{aligned} \tag{B.5}$$

Definición B.8. *Una* Relación Intensiva Difusa *es un par* $(\mathcal{H}, \mathcal{I})$ *donde:*

- $\mathcal{H}$ *es el conjunto llamado* cabecera*, que describe la estructura de la relación como un conjunto fijo de ternas atributo-dominio-compatibilidad (siendo este último opcional),*

  $$\mathcal{H} = \{(A_1 : D_1[, C_1]), (A_2 : D_2[, C_2]), \ldots, (A_n : D_n[, C_n])\}$$

  donde a cada atributo A_j *le subyace un dominio difuso generalizado, no necesariamente distinto,* D_j *con* $j \in [1, n]$*.* C_j *es un* atributo de compatibilidad *que toma valores en el intervalo* $[0, 1]$*.*

- *$\mathcal{I}$ es el conjunto llamado generador de instancia, que está constituido por una serie de reglas orientadas a datos difusos, las cuales permiten el cálculo de la instancia de la relación. Estas reglas se desarrollarán en los siguientes apartados del presente capítulo.*

La generalización de la definición de regla en el generador de instancias $\mathcal{I}$ sirve para la representación de reglas clásicas y además reglas flexibles. En [Bla03a, Bla01] se detalla el proceso de generalización de la regla hasta obtener la *Regla Generalizada con grado de acoplamiento* que permite la representación de dichas reglas para deducir con datos flexibles.

Definición B.9. *Se llama* regla generalizada con grado de acoplamiento *a la regla cuya expresión es la siguiente:*

$$\begin{aligned}\tilde{P}(X_1, X_2, \ldots, X_n, \gamma_{\tilde{P}}) \leftarrow \tilde{Q_1}(Y_{1,1}, Y_{1,2}, \ldots, Y_{1,n_1}, \gamma_{\tilde{Q_1}}) \wedge \ldots \wedge \\ \wedge \tilde{Q_m}(Y_{m,1}, Y_{m,2}, \ldots, Y_{m,n_m}, \gamma_{\tilde{Q_m}}) \wedge \Psi\end{aligned} \tag{B.6}$$

con Ψ definida como sigue:

$$\begin{aligned}\Psi \equiv \bigwedge \geq (\Theta^{=}_{i,j,k}(X_i, Y_{j,k}), \alpha_{i,j,k}) \bigwedge \geq (\Theta^{=}_{j,k,l,p}(Y_{j,k}, Y_{l,p}), \beta_{j,k,l,p}) \bigwedge \\ \bigwedge \phi_{j,k,l,p}(\Theta^{\theta}_{j,k,l,p}(Y_{j,k}, Y_{l,p}), \gamma_{j,k,l,p})\end{aligned} \tag{B.7}$$

donde $\gamma_{\tilde{P}}$ es una función grado de acoplamiento construida en base a los grados de acoplamiento de las variables y a los grados de acoplamiento de los hechos obtenidos.

B.2.2. La Representación Relacional de las Reglas Generalizadas Difusas: FREDDI Extendido

La arquitectura FREDDI fue propuesto por Medina et al. en [Med97, Pon96, Pon97] como un mecanismo para unificar un sistema de consulta deductivo con un sistema de consulta difuso, ambos construidos sobre un sistema gestor de bases de datos relacional. Este conjunto de relaciones permitía almacenar la definición de un predicado como una disyunción de una o varias reglas. Cada una de las reglas se definía como una conjunción de predicados y comparadores.

Representación de Información Deductiva

Una relación intensiva puede verse como [Bla01, Bla02a, Bla03a, Bla03b]:

- Una relación existente con su propio esquema, pero cuya instancia ha de ser calculada en función de la instancia de los predicados que intervienen en el cuerpo de sus reglas en el instante de ser consultada, o bien

- Una relación temporal que se construye en el momento de la consulta y que no posee entidad más allá del alcance de dicha consulta.

Una relación extensiva, por el contrario, no necesita una representación concreta dado que trata de una relación ordinaria, o relación difusa que ya fue expuesta en el apartado anterior.

La representación de las reglas lógicas vendrá dada por su estructura, dividiéndose en sus predicados y variables y almacenando el orden en el que se encuentran y el grado de acoplamiento que se especifique.

Por otro lado el motor de inferencia ha de ser un módulo externo o interno SGBDR (dependiendo o no si permite su inclusión en el mismo) que estará implementado en un lenguaje de programación lógico.

B.2.3. Base de Metaconocimiento Deductivo: Base de Reglas (RB)

Tal y como se explicó anteriormente, la base de metaconocimiento permite la representación de datos de mayor complejidad puedan ser representados en el modelo relacional. La representación de la información deductiva está formada por dos bases de Metaconocimiento, la FMB descrita en el apartado B.1.3 y la RB o Base de Reglas que a continuación se describe, y que permite la representación de las relaciones intensivas y las reglas generalizadas con grado de acoplamiento difuso.

Las diversas relaciones que forman parte de la RB aparecen en la figura B.7 en forma de diagrama de clases [Bla01, Bla02a, Bla03a, Bla03b].

Estas tablas tienen el siguiente cometido:

- *INTENSIONAL_TABLE_DESCRIPTION*: almacena los predicados intensivos (TABLE_ID) definidos como la disyunción de reglas P_i (RULE_ID).

- *RULE_DESCRIPTION*: describe cada una de las reglas P_i (identificada mediante TABLE_ID y RULE_ID) como una secuencia de predicados extensivos e intensivos y comparaciones concatenados con el operador de conjunción. Cada predicado puede aparecer negado y puede aparecer varias veces en la misma regla. El par (PRED_ID, RULE_ID) identifica a la regla descrita, PRED_ID establece qué predicado aparece en la posición OCC_NUMBER de la regla, si está negado o no (NEGATED) y su tipo (0 para extensivo, 1 para intensivo y 2 para comparación).

- *PREDICATE_DESCRIPTION*: describe el orden de las variables en cada uno de los predicados K_i. La misma variable puede aparecer más de

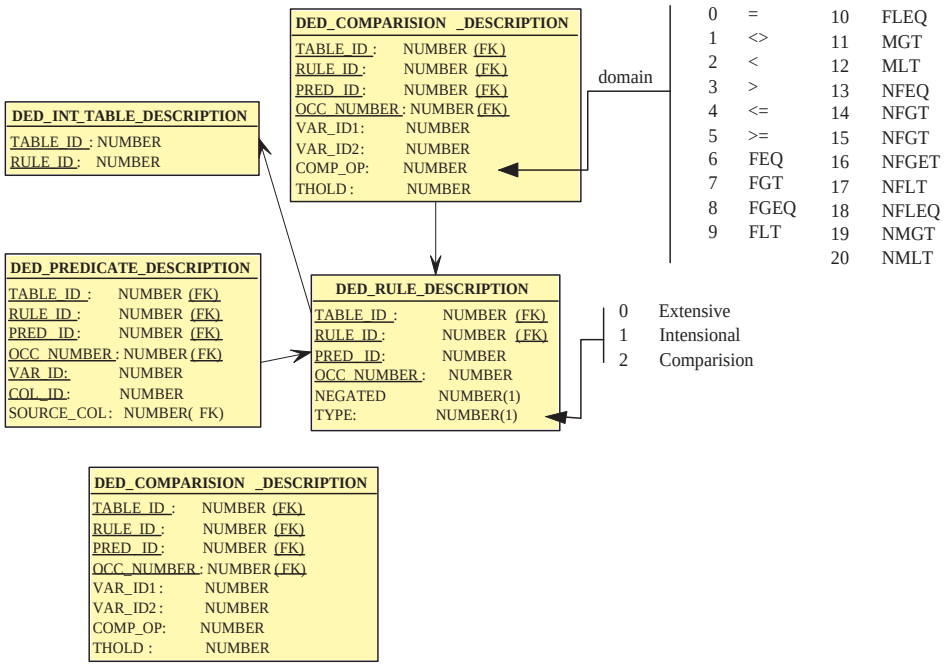


Figura B.7: Catálogo de datos deductivos

una vez en cada predicado pero varias apariciones se distinguen por su posición dentro del predicado. Una variable VAR_ID ocupa una posición COL_ID dentro de un predicado PRED_ID que aparece en una posición dada OCC_NUMBER de una regla RULE_ID que define a un predicado intensivo TABLE_ID.

- *COMPARISON_DESCRIPTION*: describe las condiciones, tipo especial de predicados, que sólo poseen dos variables y su tipo es uno de los siguientes: $=$, $\neq$, $\leq$, $<$, $\geq$, $>$, *FEQ*, *FGT*, *FGEQ*, *FLT*, *FLEQ*, *MGT*, *MLT*, *NFEQ*, *NFGT*, *NFGEQ*, *NFLT*, *NFLEQ*, *NMGT* y *NMLT*. Una condición compara dos variables (VAR_ID1 y VAR_ID2), aparece en una posición dada (OCC_NUMBER) de una regla (RULE_ID) que define a un predicado (TABLE_ID). En nuestro caso, la columna PRED_ID no es de utilidad (ya que la condición queda totalmente identificada por la posición que ocupa en la regla) pero se ha mantenido por cuestiones de uniformidad. La arquitectura FREDDI extendida es la que permite flexibilizar la representación de las reglas difusas y aumentar el número de comparadores difusos tal y como se ve en la figura B.7.

B.2.4. Sintaxis Extendida Deductiva de FSQL

Al igual que ocurría con el *FSQL*, es decir, al lenguaje de manejo flexible para consultar valores precisos e imprecisos, el *DSQL: D*eductive *SQL* (o SQL Deductivo), es el lenguaje de consulta que permite realizar deducciones [Bla01, Bla00b].

Este lenguaje permite crear y manipular nuevas estructuras (DDL). En la tabla B.4 se encuentran las cabeceras de estas sentencias. En cuanto a la manipulación de datos, o DML, se extiende únicamente la funcionalidad de la sentencia SELECT pero a nivel interno no en la sintaxis (más detalle en la sintaxis en [Bla01, Bla02a, Bla03a]).

Tabla B.4: Resumen de DFSQL

Tipo	Sentencia	Descripción
DDL	Create Intensional Table	crea tablas intensivas
DDL	Create Rule	define y almacena una regla en la BD
DDL	Delete Rule	elimina una regla previamente creada
DDL	Drop Intensional Table	borra una tabla intensiva

B.3. Minería de Datos en el Modelo Relacional

B.3.1. Ampliación Teórica de GEFRED para el Manejo de Múltiples Tipos de Datos (GEFRED*)

Se va a proceder a la definición de GEFRED* que está basado en GEFRED (definido en B.1.1) de tal manera que la definición del dominio difuso generalizado va a tener un sentido más universal [Car03a]. Con ello se pretende:

- No restringir la definición a ningún dominio en concreto,
- Formalizar la representación de tipos de datos complejos en el sentido de requerir mas de un atributo *clásico*.

Definición B.10. *Sean una serie de dominios $D_1, D_2, ..., D_n$, tal que cada D_i (con i= 1,2,...,n) es un dominio atómico en el sentido clásico de las Bases de Datos Relacionales y además esa serie de atributos conjuntamente implica una característica importante y variable que tiene una entidad.*

Se define entonces dominio complejo y se nota como D' al dominio descrito por $D_1 \times D_2 \times ... \times D_n$ siempre y cuando esos dominios $D_1, D_2, ..., D_n$ modelen conjuntamente una característica importante y variable que tiene una entidad

Definición B.11. *Sea D'un dominio complejo, $\Pi(D')$ el conjunto de distribuciones de posibilidad definidas sobre $\mathcal{D}$, entre las que se incluyen aquellas que describen los valores Unknown y Undefined. Se considera también el valor Null. el dominio Difuso Generalizado Complejo se define como D'_G donde $D'_G \subseteq \Pi(D') \cup Null$*

Los atributos que se definan sobre el dominio difuso generalizado complejo podrán tomar cualquier valor simple, excluyente o distribución de posibilidad. Dicho dominio puede implicar tanto dominios precisos, como difusos de cualquier naturaleza, siendo un caso particular los tipos de datos reflejados en la definición B.1. La gestión de los tipos de datos va a ser posible mediante la definición de una serie de operaciones especiales a realizar sobre los elementos del dominio, que permitirán incorporar significado a la representación de los datos, en definitiva, para que se consiga el objetivo de modelar más certeramente la realidad. En cualquier caso, todas estas capacidades de representación encontrarán en el comparador difuso generalizado complejo (que se definirá más adelante), el mecanismo mediante el que modelar su actuación.

Definición B.12. *Una relación difusa generalizada compleja R es un par de conjuntos $(\mathcal{H}, \mathcal{B})$, definidos como sigue:*

- $\mathcal{H}'$ *es el conjunto llamado* cabecera *y describe la estructura de la relación mediante un conjunto de ternas atributo complejo-dominio complejo-compatibilidad (donde el último es opcional),*

$$\mathcal{H}' = \{(A'_{G1} : D'_{G1}[, C'_{A_{G1}}]), \ldots, (A'_{Gn} : D'_{Gn}[, C'_{A'_{Gn}}])\}$$

donde a cada atributo A'_j*, le subyace un dominio difuso generalizado cinokehi, no necesariamente distinto,* $D'_j, j \in [1, n]$*.* C'_j *es el llamado atributo de compatibilidad y toma valores en* $[0, 1]$*.*

- $\mathcal{B}'$ *es el conjunto llamado* cuerpo *y está formado por una serie de tuplas generalizadas difusas distintas, donde cada tupla está compuesta por un conjunto de ternas atributo-valor-grado (donde este último es opcional),*

$$\mathcal{B}' = \{\{(A'_{G1} : \tilde{d'}_{i1}[, c'_{i1}], \ldots, (A'_{Gn} : \tilde{d'}_{in}[, c'_{in}])\}\}$$

con $i = 1, \ldots, m$ *y donde* m *es el número de tuplas de la relación,* $\tilde{d_{ij}}$ *representa el valor del dominio que toma la tupla* i *sobre el atributo* A_j *y* c_{ij} *es el grado de compatibilidad asociado a este valor.*

Definición B.13. *Sea* U' *el dominio complejo de discurso considerado. Se llama* comparador extendido θ' *a cualquier relación difusa definida sobre* U' *y expresada como sigue:*

$$\theta' : U' \times U' \to [0, 1]$$
$$\theta'(u'_i, u'_j) \mapsto a$$

con $u'_i, u'_j \in U'$ *y* $a' \in [0, 1]$*.*

Definición B.14. *Sea* U' *un dominio complejo de discurso,* D' *un dominio difuso complejo construido sobre el mismo y* θ' *un comparador extendido definido sobre* U'*. Consideremos una función* $\Theta'^{\theta'}$ *definida como sigue:*

$$\theta'^{\theta'} : D' \times D' \to [0, 1]$$
$$\Theta'^{\theta'}(\tilde{d'_1}, \tilde{d'_2}) \mapsto [0, 1]$$

Se dice que $\Theta'^{\theta'}$ *es un* comparador difuso generalizado complejo *sobre* D' *inducido por el comparador extendido complejo* θ'*, si cumple:*

$$\Theta'^{\theta'}(\tilde{d'_1}, \tilde{d'_2}) = \theta(d'_1, d'_2), \forall d'_1, d'_2 \in U'$$

donde $\tilde{d'_1}$ *y* $\tilde{d'_2}$ *representan las distribuciones de posibilidad* $\{1/d'_1\}$ *y* $\{1/d'_2\}$ *inducidas, respectivamente, por los valores complejos* d'_1 *y* d'_2*.*

Definición B.15. *Se llama* proyección difusa generalizada compleja *de* R' *sobre* X'*, y se nota por* ${\mathcal{P}'}_{X'}(R')$*, a una relación difusa generalizada compleja de la forma:*

$$ {\mathcal{P}'}_{X'}(R') = \begin{cases} {\mathcal{H}'}_{\mathcal{P}'} = X' \\ {\mathcal{B}'}_{\mathcal{P}'} = \{(A'_s : \tilde{d'}_{is}[, c'_{is'}])\} \end{cases} \tag{B.8} $$

donde $s \in S$*,* $s' \in S'$ *y* $S, S' \subseteq \{1, \ldots, n\}$*.*

Definición B.16. *Sea* R' *una relación difusa generalizada compleja como la de la definición B.12,* $\tilde{a'} \in D'$ *una constante,* $\Theta'^{\theta'}$ *un comparador difuso generalizado y* γ *un* umbral de cumplimiento*. Entonces, se llama* selección difusa generalizada compleja *sobre la relación* R' *inducida por* $\Theta'^{\theta'}$ *compuesto con* $\tilde{a'}$ *y el atributo* $A'_k, k \in \{1, \ldots, n\}$ *y cualificada por* γ*, y se nota por* ${\mathcal{S}'}_{\Theta'^{\theta'}(A'_k, \tilde{a'}) \geq \gamma}(R')$ *a la relación difusa generalizada de la forma:*

$$ {\mathcal{S}'}_{\Theta'^{\theta'}(A'_k, \tilde{a'}) \geq \gamma}(R') = \begin{cases} {\mathcal{H}'}_{\mathcal{S}'} = \{(A'_1 : D'_1[, C'_{A'_1}]), \ldots, (A'_n : D'_n[, C'_{A'_n}])\} \\ {\mathcal{B}'}_{\mathcal{S}'} = \{\{(A'_1 : \tilde{d'}_{r'1}[, c''_{r'1}], \ldots, (A'_k : \tilde{d'}_{rk}[, c''_{rk}]), \\ \qquad \ldots, (A'_n : \tilde{d'}_{rn}[, c''_{rn}])\}\} \end{cases} \tag{B.9} $$

con

$$ c''_{rk} = \theta'^{\theta'}(\tilde{d'}_{rk}, \tilde{a'}) \geq \gamma \tag{B.10} $$

donde $r = 1, \ldots, m'$ *con* m' *el número de tuplas de la selección.*

B.3.2. Representación de Múltiples Tipos de Datos en el Modelo Relacional(FIRST*)

Estructura de los Tipos de Datos

*FIRST** [Car03a] es la interfaz que proporciona acceso a múltiples tipos de datos, definidos en el modelo *GEFRED**, con objeto de realizar tareas de minería de datos sobre un SGBDR. *FIRST** está basado en la arquitectura FIRST que permite la manipulación de datos difusos, lo que proporciona al sistema dicha característica también.

Dentro de un SGBDR clásico, una serie de atributos del mismo implicarán un dominio difuso generalizado complejo si modelan conjuntamente una característica importante y variable que tiene una entidad y además permiten algún tipo de tratamiento difuso. *FIRST** añade un nuevo tipo de datos a los tres que ya se habían definido en *FIRST*, el *tipo difuso 4*, que representa a la serie de atributos clásicos que determinan un dominio difuso generalizado complejo y que, por tanto, pueden ser consultados de forma imprecisa. Se trata de un supertipo puesto que contiene la definición de los otros 3, y esta formado por:

- *Atributos de datos* que contienen la información en sí. Por ejemplo si se trata de representar una distribución trapezoidal $[\alpha, \beta, \gamma, \delta]$ con las mismas características que un tipo difuso 2 estos atributos se corresponderían con los F1, F2, F3, F4 (véase tabla B.1) de la representación de dicho atributo.

- *Atributos de metadatos*, contienen información que permiten entender los atributos de datos. Este tipo de atributos no siempre son necesarios. Siguiendo con el ejemplo anterior, este atributo se correspondería con la columna FT de la tabla B.1 que indica si los atributos de datos contienen un valor *Null*, *Unknown*, *Undefined*, un trapecio, etc.

Comparadores Difusos Generalizados Complejos

Para representar un dominio difuso generalizado complejo debe ser posible que los atributos permitan algún tipo de tratamiento difuso. Esto se lleva a cabo sobre los atributos difusos de tipos 4 mediante la definición de al menos uno de los comparadores difusos usados en FIRST. Por tanto el usuario debe definir los comparadores difusos que crea necesarios para cada atributo de tipo 4 del que se vaya a hacer un tratamiento difuso.

La semántica del comparador puede ser completamente subjetiva a quien lo defina o al problema a resolver. Para los comparadores que no tienen un referencial ordenado, los comparadores pueden tener un significado cualitativo más que cuantitativo.

Lo único que es necesario establecer es las restricciones de los comparador difusos para que tengan un significado coherente para un mismo tipo de datos difuso 4 [Car03a].

B.3.3. Base de Metaconocimiento Difuso*(FMB*)

Toda la información adicional sobre la estructura de los dominios y valores que puede tomar cada atributo construido sobre un dominio generalizado difuso complejo constituye lo que se conoce como Base de Metaconocimiento Difuso* (FMB*) [Car03a]. Este toma más importancia si cabe, en FIRST* que en el anterior modelo, debido a que va a hacer posible tanto la definición, como el tratamiento de los *tipos difuso 4*.

Además de las estructuras que ya formaban parte de FIRST para el tratamiento de los tipos de datos difusos 1, 2 y 3, a las cuales se le realizan algunas modificaciones para adaptarlas al modelo. Se añaden nuevas estructuras relacionales que posibilitan la definición y tratamiento del *tipo difuso 4*.

Las funciones de los comparadores difusos que definen el comportamiento para los distintos tipos de *datos difuso 4* son añadidas a la base de FMB*. La representación de estas funciones es:

CDEG (A *fcomp* B, $\{constante_{A_1}, constante_{A_2}, \ldots, constante_{A_{n\_fcomp}}\}$, $\{constante_{B_1}, constante_{B_2}, \ldots, constante_{B_{n\_fcomp}}\}) \rightarrow [0, 1]$

Las funciones de representación definen la visualización para los distintos *tipos de datos difuso 4*. Su estructura es muy parecida a la de los comparadores, y es la siguiente:

FSHOW (A, $\{constante_{A_1}, constante_{A_2}, \ldots, constante_{A_{n\_fcomp}}\}$)

A continuación se exponen cada una de las tablas de la FMB que han sido ampliadas para utilizar el *tipo difuso 4*:

- *FUZZY_COL_LIST*: contiene la lista de aquellos atributos de tabla de la base de datos que son susceptibles de tratamiento difuso. Añade en el atributo F_TYPE un 4 para los *tipos difuso 4*.
- *FUZZY_OBJECT_LIST*: almacenan los objetos difusos que pertenecen al dominio de atributo en cuestión. Añaden en el atributo FUZZY_TYPE un 5 para las etiquetas lingüísticas definidas para un tipo de *dato difuso 4* en la tabla DMFSQL_LABEL_DEFINITION

El resto de tablas que forman parte única y exclusivamente de la FMB* y por tanto solo se utilizan para atributos difusos de tipo 4 son (ver figura B.8):

- *DMFSQL_COL_COL*: contiene la lista de aquellos atributos de la tabla de la base de datos que forman parte de, lo que se ha llamado, un dominio difuso generalizado complejo. Este dominio se ha implementado, tanto con atributos de datos, como de metadatos, aunque ambos son definidos de igual forma en esta tabla. Cada uno de estos dominios vendrá cualificado por un único atributo existente en la tabla y definido por el par: referencia a la tabla a la que pertenecen (OBJ#) y columna en la que se almacenan (COL#). En el resto de tablas de la FMB cada uno de los tipos difusos 4 vendrán caracterizados por este único atributo. Los atributos integrantes del dominio complejo vendrán definidos por el par: referencia a la tabla a la que pertenecen (OBJ#) y columna en la que se almacenan (COL2#). Mediante un atributo adicional, y por cuestiones de implementación, se establece un orden en cada uno de estos atributos integrantes del dominio complejo (ORDER#). En definitiva, en esta tabla se especifica la representación interna del atributo difuso tipo 4.
- *DMFSQL_LABEL_DEFINITION*: contiene información sobre las etiquetas lingüísticas definidas para los tipos difusos 4. Cada una de ellas

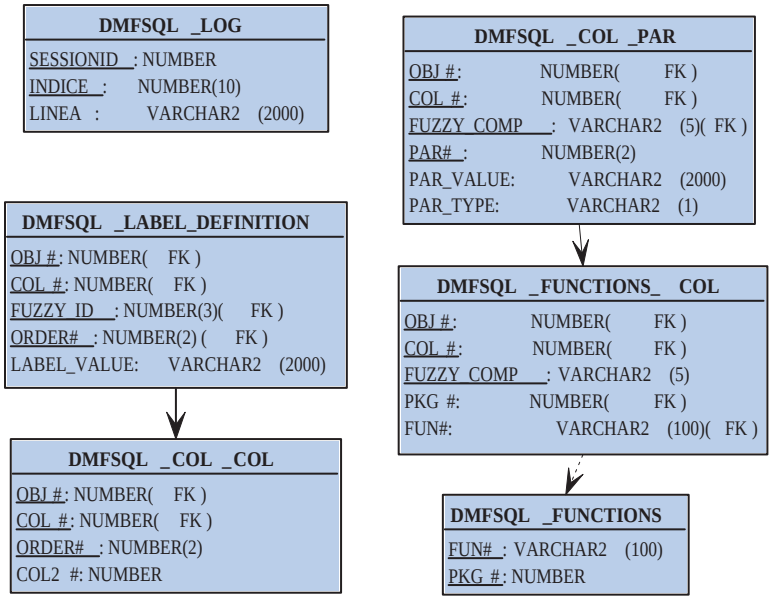


Figura B.8: Estructura relacional de la extensión de la FMB para manejo de múltiples datos

está descrita por la tabla y columna a cuyo dominio pertenece (OBJ# y COL#), la etiqueta a la que se asocia (FUZZY_ID) y los parámetros que la definen (LABEL_VALUE) siguiendo un cierto orden (ORDER#).

- *DMFSQL_FUNCTIONS*: en esta tabla se define la referencia de las funciones tanto que implementan a los distintos comparadores difusos de los atributos difusos de tipo 4, como las funciones de representación de los mismos. Dichas funciones vienen referenciados por el identificador del paquete (PKG#) y de la función dentro del mismo (FUN#).
- *DMFSQL_FUNCTIONS_COL*: contiene la definición para cada atributo *difuso tipo 4*, prototipado, por la tabla a la que pertenece (OBJ#) y la columna (COL#), y para cada comparador difuso (FUZZY_COMP) la función que se le asocia, encontrándose ésta en el paquete (PKG#) e identificándose de forma univoca (FUN#) dentro del citado paquete. Los comparadores difusos posibles son los de la figura B.7. También es posible especificar, mediante esta tabla, la función de representación que se quiere usar para el atributo difuso de tipo 4, es decir, como se quiere visualizar el mismo en las sentencias SELECT. Para ello, el campo con el comparador difuso (FUZZY_COMP) debe tener el valor 'FSHOW'.
- *DMFSQL_COL_PAR*: contiene la información de los parámetros adicionales para construir las llamadas a funciones que implica cada tipo difuso 4 respecto a cada comparador. Como se ha comentado, es tremendamente útil para la reusabilidad de la codificación de las funciones ya implementadas en la FMB. Así, para un atributo difuso tipo 4 determinado (OBJ#, COL#) y para un comparador difuso que implica (PAR_VALUE) y en el orden establecido (PAR#), el valor de los parámetros de PAR_TYPE tendrá valor C para un dominio de tipo carácter, N para numérico y D para fechas.

B.3.4. Ampliación de FIRST* para el Data Mining

Técnicas de Minería de Datos Definidas

Una vez implementado el Interfaz Difuso para Sistemas Relacionales para el manejo de múltiples tipos de datos (FIRST*), se ha obtenido la implementación de un modelo de BDRD sobre un SGBDR en el que el tratamiento difuso de la diversidad de dominios susceptibles de ser tratados por un sistema de Minería de Datos está resuelto.

Una vez solucionado el problema de gestionar la información, cualquiera que sea un forma, se va proceder a implementar una interfaz que permite utilizar FIRST* como base a la aplicación de distintas técnicas de Minería de

Datos en el marco del modelo de BDRD ya implementado. Dicha interfaz se denominará DmFIRST y permite realizar las operaciones de [Car00, Car03b, Car98, Car99]:

- Clustering: donde se propone una forma de obtención de la matriz de distancias de la población.
- Caracterización: incluye una técnica que permite especificar el nivel de abstracción al que se quiere describir los datos tal y como se consideraba deseable en los sistemas de DM.
- Clasificación difusa: donde se propone dos formas de obtener la clasificación difusa, basándose en los centroides obtenidos tras la caracterización o usando los vecinos más cercanos.
- Dependencias difusas entre atributos: donde se propone un concepto de dependencias, llamadas dependencias globales difusas que constituye el marco común que integra tanto las dependencias difusas como las dependencias graduales difusas.

B.3.5. Base de Metaconocimiento Difuso para Minería de Datos (DMFMB)

Los problemas que producidos al utilizar estos los servidores de FSQL cuando se realizan tareas de DM no se pueden resolver con el uso de sentencias simples en mayoría de los casos (y en última instancia de SQL). Es por esto que se introduce un nuevo esquema, dmFIRST que va ha definir un nuevo tipo de objeto que no existe ni en FSQL ni en SQL que se denomina *"proyecto"* y que tiene como misión:

- Servir de soporte para guardar las condiciones iniciales, resultados intermedios y finales del proceso de DM a realizar.
- Englobar lógicamente una serie de resultados intermedios, en forma de tablas, en dicho proceso de DM. Estos resultados intermedios estarán encaminados a agilizar el proceso de DM iterativo ya que el tener estos datos precalculados hará que ante determinados refinamientos de los requerimientos del proceso de DM, la respuesta del sistema sea más o menos inmediata.

La base de Metaconocimiento dmMB queda formada ahora por la FMB completa de FIRST y unas nuevas estructuras relacionales que posibilitan el tratamiento del objeto *proyecto* comentado (ver figura B.9):

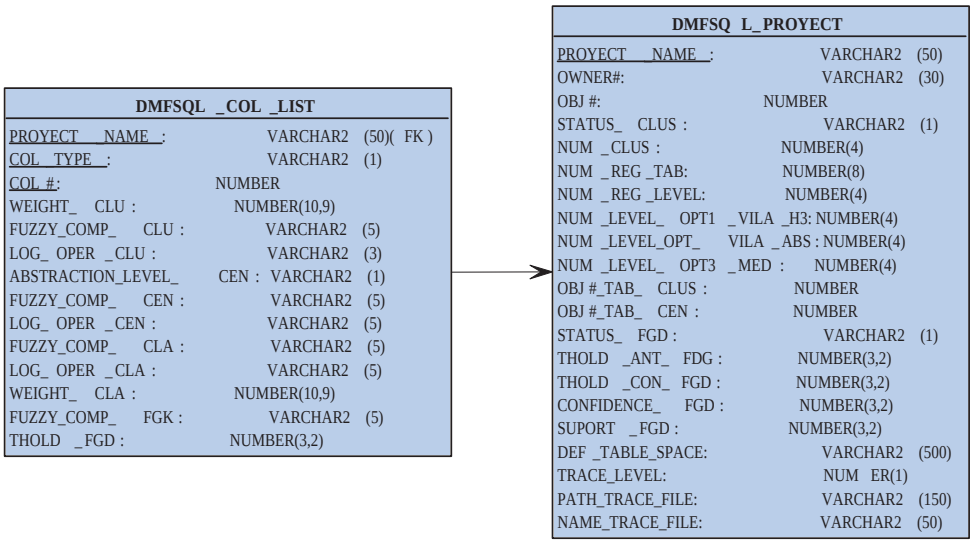


Figura B.9: Estructura relacional de la DmFMB

- *DMFSQL_PROJECT*: contiene la información general sobre los proyectos de DM. Se identifica de forma unívoca el proyecto y se da un identificativo al propietario. Se establece la tabla de origen de datos (OBJ#) y el estado actual del proceso de clustering especificado en la tabla 9, si el estado es C, entonces se especifican el número de clusters obtenidos tras el proceso (NUM_CLUS). STATUS_CLUS contendrá valor es C se trata de un tipo carácter, si es N de tipo numérico y si es D de tipo fecha. NUM_REG_TAB, indica el número de filas en la tabla id_tabla_orig_proyecto, NUM_REG_LEVEL indica el numero de posibles alfa-cortes que se pueden hacer dentro del dendograma. Los atributos (NUM_LEVEL-_opt1_vila_h3), (NUM_LEVEL_opt2_vila_abs), (NUM_LEVEL_opt3_med) almacenan el nivel del dendograma al que se obtiene una partición óptima basado en H3, absoluta y media. Se almacenará un identificativo de la última tabla generada con una partición en concreto de la población dentro del proceso de clustering en el atributo OBJ#_TAB_CLUS, y un identificativo de la ultima tabla de centroides generada, que caracterizan a los distintos grupos existentes en la tabla anterior en OBJ#_TAB_CEN. STATUS_FGD es el estado del proceso de obtención de DGDs, si su valor es C se trata de un tipo carácter, si es N de tipo numérico y si es D de tipo fecha. THOLD_ANT_FGD y THOLD_CON_FGD son los umbrales alfa y beta. CONFIDENCE_FGD la confianza obtenida de la

DGD y el soporte estará en SUPPORT_FGD. En DEF_TABLE_SPACE están las especificaciones físicas para el almacenamiento de las distintas tablas que se generen como resultado de la ejecución del proyecto. TRACE_LEVEL, PATH_TRACE_FILE, NAME_TRACE_FILE: son de uso interno del servidor para gestionar trazas de ejecución que tienen como objeto devolver resultados intermedios a aplicaciones cliente, depuración de errores, etc.

- *DMFSQL_COL_LIST*: contiene la información sobre las distintas columnas trascendentes para el proceso de DM de la tabla con el origen de datos del proyecto (ID_TABLA_ORIG_PROYECTO). Se identifica de manera única el proyecto de DM con (PROYECT_NAME). El atributo (COL_TYPE) indica el tipo de procesamiento de DM para la columna (COL#), si su valor es C se trata de Clasificación, clustering o caracterización, si es A su dominio es el antecedente dentro del ámbito de las DGDs y si es Q se trata del consecuente dentro del ámbito de las DGDs. En (WEIGHT_CLU) se almacena el peso por el que se pondera la columna para clustering. FUZZY_COM_CLU es el comparador difuso de igualdad (FEQ o NFEQ) que se le va a aplicar a la columna para la obtención de la matriz de distancias en el proceso de clustering. En el caso de que el comparador difuso no sea simétrico, se indica en LOG_OPER_CLU el operador lógico que se va a usar para obtener una comparación simétrica dentro del proceso de clustering. (puede ser AND si se quiere usar una T-norma u OR si se quiere usar una t-conorma). FUZZY_COM_CEN y LOG_OPER_CEN se usan igual que en el caso anterior, para el cálculo de centroides. FUZZY_COM_CLA y LOG_OPER_CLA se utilizan para el proceso de clasificación y WEIGHT_CLA almacena el peso por el que se pondera la columna para clasificación. FUZZY_COMP_FGD es el comparador difuso de la Tabla 7 que se le va a aplicar a la columna para la obtención de la DGD donde los umbrales *alfa y beta* se almacenarán en THOLD_FGD para cada columna del antecedente y consecuente.

Toda la Base de metaconocimiento que permite la utilización de técnicas de DM, se encuentra representada en la figura B.10.

B.3.6. Sintaxis Extendida para Operaciones de DM: DMFSQL

FSQL como se ha mencionado con anterioridad no cumple los requisitos mínimos para ser considerado un lenguaje propiamente de DM. Con objeto de solucionar este hecho, se ha extendido FSQL, creándose el dmFSQL que resuelve las tareas de DM.

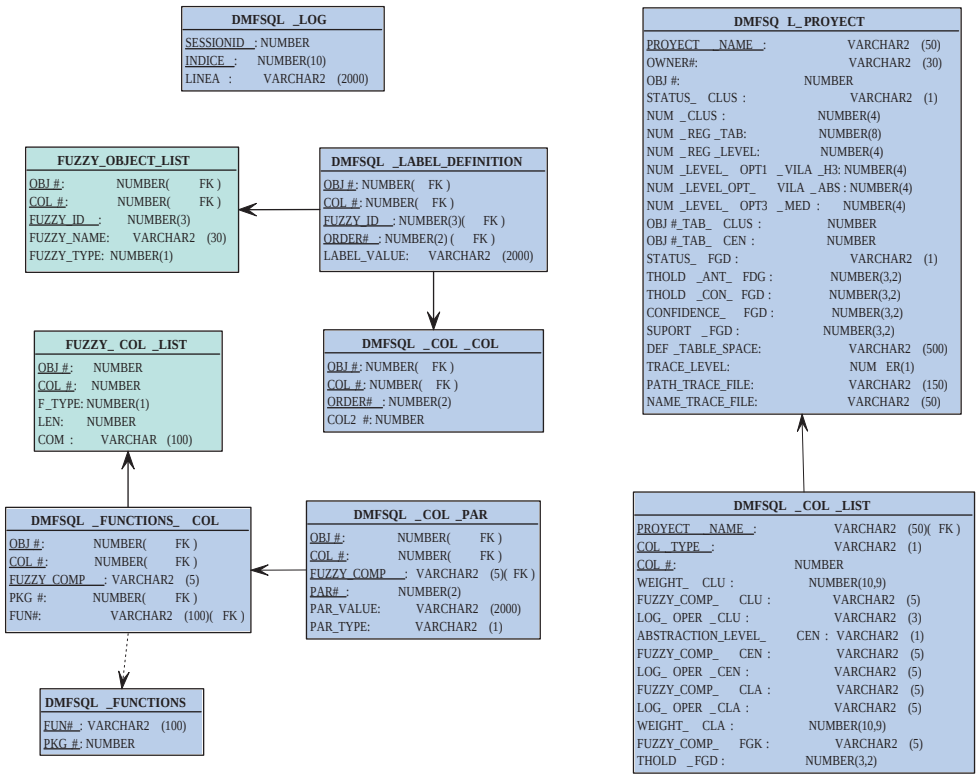


Figura B.10: Estructura relacional de la Base de Metaconocimiento para el DM

Este lenguaje consta de las siguientes partes: El DDL de DmFSQL permitirá la consulta y la modificación de las estructuras en las que se almacenan los datos, y va a consiste en las operaciones sobre el *proyecto* que se muestran en la tabla B.5. El DML de DmFSQL permite la consulta y la modificación de los datos almacenados en la base de datos. Básicamente consiste en extender el comando SELECT_MINING para que pueda utilizar las distintas técnicas de DM, que se ven en la tabla B.5. La descripción completa de la sintaxis íntegra de estas sentencias puede verse en [Car03a, Car03b].

Tabla B.5: Resumen de DMFSQL

Tipo	Sentencia	Descripción
DDL	CREATE_MINING	Crea un nuevo proyecto
DDL	ALTER_MINING	Modifica un proyecto
DDL	DROP_MINING	Borra un proyecto
DDL	GRANT_MINING	Da permiso para la gestión de un proyecto a un usuario
DDL	REVOKE_MINING	Elimina los permisos previamente concedidos para la gestión de un proyecto a un usuario
DML	CLUSTERING	Para el proceso de clustering y caracterización
DML	CLASIFICATION	Para el proceso de clasificación
DML	FGLOBAL_ DEPENDENCIES	Para la obtención de dependencias globales difusas entre atributos

Apéndice C

Base de Datos de Suelos

C.1. Descripción del Esquema de la Base de Datos

La base de datos de suelos describe la información acerca de las características que tienen los suelos recopilada a través de encuestas realizadas a agricultores. Esta Base de Datos ha sido desarrollada gracias al proyecto "Fuzzy-KIM, un Sistema de Minería de Datos con Ayuda Inteligente basado en Técnicas de Soft-Computing (Plan Nacional I+D)", cuya referencia es CICYT TIC2002-04021-C02-02, llevado a cabo en entre los años 2002-2005 y financiado por el Ministerio de Ciencia y Tecnología.

La base de datos de suelos, tiene como particularidad que la mayoría de los datos que la componen son de carácter difuso, es decir, los valores de los diferentes campos, son descritos dependiendo de su contenido, con etiquetas lingüísticas o valores discretos sobre un referencial no ordenado. Al realizar la base de datos con este tipo de datos se ha facilitado la tarea de realizar la encuesta al encuestado, dado que este tipo de información confiere flexibilidad a la hora rellenar datos en el formulario.

C.1.1. Descripción de Clases

A continuación se describe el diseño conceptual de la BD de suelos. Para realizar este diseño se ha utilizado un diagrama de clases de UML, mostrado en la figura es C.1. Las clases principales de que está compuesta esta BD y la información que representa está descrita a continuación y en la tabla C.1 donde se especifica con más detalle el contenido semántico de algunos de los atributos de esta BD:

- *Localización*: Describe el lugar exacto del suelo del que se trate y las características propias del suelo en esta localización. Los atributos se

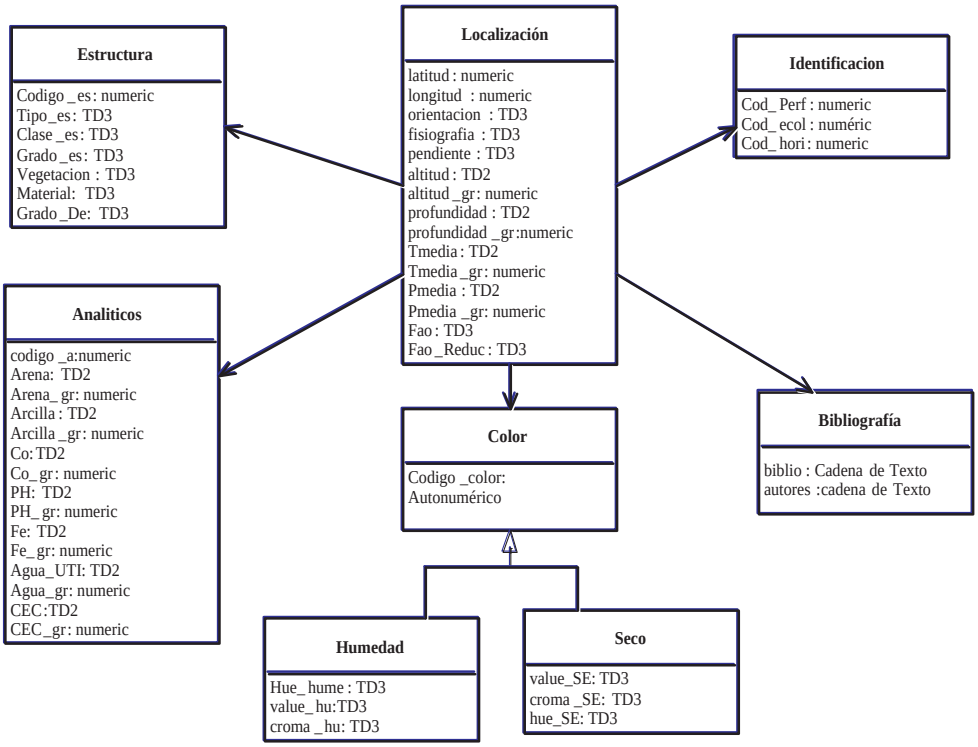


Figura C.1: Diagrama de Clases de la BD de Suelos

encuentran descritos en la tabla C.2.

- *Estructura*: Describe las características generales de la estructura del suelo. La descripción de las característica está en la tabla C.3.
- *Analíticos*: Describen las características analíticas o composición del suelo. La descripción de los atributos de esta clase se encuentra en la tabla C.4.
- *Identificación*: Describe los códigos de identificación ecológicos, de perfil y horizonte relacionados con los suelos. Las propiedades se describen en la tabla C.5.
- *Bibliografía*: Describe las fuentes que realizaron las encuestas. La composición de esta clase se describe en la tabla C.6.
- *Color*: Describe el color que tiene el suelo. Las propiedades de esta clase y de sus subclase están descritas en la tabla C.7.

- *Humedad*: Describe el color que tiene el suelo cuándo está húmedo.
- *Seco*: Describe el color que tiene el suelo cuándo está seco.

C.1.2. Paso a Tablas: Modelo Relacional

A partir del diagrama en UML de la figura C.1 obtenemos la siguiente descripción de tablas del modelo relacional extendido que soporta datos difusos. Esta descripción se representa utilizando los tipos de datos y estructuras del lenguaje SQL para especificar los tipos de datos y restricciones que hay definidos sobre los datos clásicos, y FSQL para los difusos.

```
Estructura (
    Codigo_es NUMERIC PRIMARY KEY,
    Tipo_es FTYPE3(1),
    Clase_es FTYPE3(1),
    Grado_es FTYPE3(1),
    Vegetacion FTYPE3(1),
    Material FTYPE3(1),
    Grado_de FTYPE3(1),
)

Analíticos (
    Codigo_a NUMERIC PRIMARY KEY,
    Arena FYTPE2 (10,40) FLOAT (2),
    Arena_gr NUMERIC(4,2),
    Arcilla FYTPE2 (5,50) FLOAT (2),
    Arcilla_gr NUMERIC(4,2),
    Co FYTPE2 (5,20) FLOAT (2),
    Co_gr NUMERIC(4,2),
    PH FYTPE2 (1,10) FLOAT (2),
    PH_gr NUMERIC(4,2),
    Fe FYTPE2 (0.5, 2) FLOAT (2),
    Fe_gr NUMERIC(4,2),
    Agua FYTPE2 (0.5, 2) FLOAT (2),
    Agua_gr NUMERIC(4,2),
    CEC FYTPE2 (5, 20) FLOAT (2),
    CEC_gr NUMERIC(4,2),
)

Identificación (
```

```
    cod_perf NUMERIC,
    cod_ecol VARCHAR(2),
    cod_hori NUMERIC,
    PRIMARY KEY (cod_perf,col_ecol,cod_hori))

Bibliografía (
    biblio VARCHAR(14),
    autor VARCHAR(8),
    PRIMARY KEY (biblio, autor)
)

Color (
    Codigo_c NUMERIC PRIMARY KEY,
    hue_hume  FTYPE3(1),
    value_hu FTYPE3(1),
    croma_hu FTYPE3(1),
    hue_se FTYPE3(1),
    value_se FTYPE3(1),
    croma_se FTYPE3(1))

Localización (
    latitud NUMERIC NOT NULL,
    longitud NUMERIC NOT NULL,
    orientación FTYPE3(1)  ,
    fisiografía FTYPE3(1) ONLY LABEL,
    pendiente FTYPE2(5,20) FLOAT (2) ONLY LABEL,
    altitud FYTPE2 (0.5, 2) FLOAT (2) ,
    altitud_gr NUMERIC(4,2),
    profundidad FYTPE2 (0.5, 2) FLOAT (2),
    profundidad_gr NUMERIC(4,2),
    Tmedia FYTPE2 (4, 10) FLOAT (2) NOT UNKNOWN NOT UNDEFINED,
    Tmedia_gr NUMERIC(4,2),
    Pmedia FYTPE2 (0.5, 2) FLOAT (2),
    Pmedia_gr NUMERIC(4,2),
    Fao FTYPE3(1),
    tipo_hori FTYPE(3),
    Fao_reduc FTYPE3(1),
    codigo_es REFERENCES estructura(codigo_es),
    codigo_a REFERENCES analiticos(codigo_a),
    codigo_c REFERENCES color(codigo_color),
```

```
biblio VARCHAR(14),
autor VARCHAR(8),
cod_perf NUMERIC,
cod_ecol VARCHAR(2),
cod_hori NUMERIC,
PRIMARY KEY (latitud,longitud)
FOREIGN KEY (bib    lio,autor) REFERENCES
                Bibliografía(biblio,autor),
FOREIGN KEY (cod_perf,cod_ecol,cod_hori)
    REFERENCES  Identificacion(cod_perf,cod_ecol,cod_hori))
```

Dado que se tratan como claves ajenas en la tabla *Localización* todos los atributos de las tablas *Bibliografía* e *Identificación*, estas tablas se suprimirán de la definición de la BD. De esta manera también se eliminarán de la definición de la estructura *Localización* todas las referencias a dichas tablas, es decir las claves ajenas a *Bibliografía* y a *Identificación*.

C.1.3. Etiquetas Lingüísticas para los TD2

En esta base de datos se han transformado los atributos numéricos, normalmente identificados en la BD por el nombre del atributo seguido de ¨*_gr*¨, por otros atributos de carácter difuso con el mismo nombre pero sin dicha extensión. Estos atributos están definidos bajo un referencial ordenado pero los valores que van a contener estarán formados fundamentalmente por etiquetas lingüísticas cuyos valores se muestran en las tablas que se describen a continuación.

Los atributos de la tabla *Analiticos*, definen sus etiquetas lingüísticas dependiendo del atributo en las siguientes tablas: *Arena* está descrita en la tabla C.13, *Arcilla* en la C.14, *Co* en la C.15, *Carbonat* en la C.16, *Ph* en la C.17, *Agua_Uti* en la C.18, *Fe* en la C.19, *CEC* en la C.20. Con respecto a la tabla *Localización* encontramos los atributos: *PMedia* descrita en la C.8, *Tmedia* en la C.9, *Altitud* en la C.10, *Profundi* en la C.11, *pendiente* en la C.12. Por último el atributo *Clase_es* en la C.21 en *Estructura*.

Tabla C.1: Atributos de la base de datos de color de suelos, agrupados de acuerdo a su semántica

Grupo semántico	**Atributo**	**Comentarios**
Estaciones ambientales	Mesoambiente	Son combinaciones multidimensionales de factores ambientales que definen espacios más o menos homogéneos de influencia en el desarrollo posterior del suelo. A los factores ambientales se les ha denominado factores formadores del suelo.
Factores formadores	Altitud	Los factores formadores no forman parte del individuo suelo y no son partes o componentes de su estructura. Son factores ambientales generales, susceptibles de ser medidos, que actúan como agentes causales de los procesos edafogenéticos que conducen al desarrollo del suelo.
	Precipitación media anual	
	Temperatura media anual	
	Material original	
Horizontes	Tipo de horizonte	Expresan las características de zonas homogéneas dentro del suelo y son resultado final de una serie de procesos edafogenéticos y de la actuación de los agentes (factores) formadores.
Componentes	% Arena	Los componentes y propiedades son características, morfológicas o analíticas, susceptibles de ser medidas o descritas en cada horizonte; pueden actuar como diagnósticos del mismo. En el aspecto de la génesis, las propiedades son consecuencia de los componentes.
	% Arcilla	
	% Carbono orgánico	
	% Hierro libre	
Propiedades	Value	
	Chroma	
	Hue	

Tabla C.2: Descripción de las propiedades la clase *Localización*

Atributo	Tipo	Descripción
latitud	Numérico(11)	Posición geográfica
longitud	Numérico(11)	Posición geográfica
orientación	Categórico	Orientación del lugar (NSEO)
fisiografía	Categórico	Fisiografía del lugar (ladera, llano, etc.)
pendiente	Numérico(FDT2)	Tipo de pendiente (llano, de escalón, en cuesta,etc.)
altitud	Numérico (FDT2[1])	Altitud del terreno en valor difuso
altitud_gr	Numérico(4,2)	Altitud del terreno en valor preciso medio
profundidad	Numérico (FDT2)	Profundidad media efectiva del terreno en valor difuso
profundidad_gr	Numérico(4,2)	Profundidad media efectiva del terreno en valor precio
Tmedia	Numérico (FDT2)	Temperatura media anual del lugar en valor difuso
Tmedia_gr	Numérico(4,2)	Temperatura media anual del lugar en valor preciso
Pmedia	Numérico (FDT2)	Precipitación media anual del lugar en valor difuso
Pmedia_gr	Numérico(4,2)	Precipitación media anual del lugar en valor preciso
Tipo_hori	Categórico	Tipo de horizonte
FAO_Red	Categórico	Descriptores del suelo de la FAO pero reducido el nº de variables tras la aplicación de un proceso

Tabla C.3: Descripción de las propiedades de la clase *Estructura*

Atributo	Tipo	Descripción
Codigo_es	Numérico	Código autonumérico de la tabla
Tipo_es	Categórico	Tipo de estructura de suelo(granular, migajosa, etc.)
Clase_es	Numérico (FDT2)	Clase de estructura de suelo (fina, media, compacta,etc.)
Grado_es	Categórico	Grado de la estructura (frágil, fuerte, etc.)
Vegetación	Categórico	Tipo de Vegetación del suelo (bosque, cultivo, regadío,etc.)
Material	Categórico	Tipo de material Original del suelo (ácido , calcáreo, roca, etc.)
Grado_de	Categórico	Grado de Erosión del suelo

Tabla C.4: Descripción de las propiedades de la clase *Analíticos*

Atributo	Tipo	Descripción
Codigo_a	Numérico	Código autonumérico de la tabla
Arena	Numérico (FDT2)	Cantidad de Arena del suelo en valor difuso
Arena_gr	Numérico(4,2)	Porcentaje de Arcilla del suelo en valor preciso
Arcilla	Numérico (FDT2)	Cantidad de Arcilla del suelo en valor difuso
Arcilla_gr	Numérico(4,2)	Porcentaje de Arena del suelo en valor preciso
Co	Numérico (FDT2)	Cantidad de Carbono Orgánico del suelo en valor difuso
Co_gr	Numérico(4,2)	Porcentaje de Carbono Orgánico del suelo en valor preciso
PH	Numérico (FDT2)	Cantidad de PH del suelo en valor difuso
PH_gr	Numérico(4,2)	Porcentaje de PH del suelo en valor preciso
Fe	Numérico (FDT2)	Cantidad de Hierro Total del suelo en valor difuso
Fe_gr	Numérico(4,2)	Porcentaje de Hierro Total del suelo en valor preciso
Agua	Numérico (FDT2)	Cantidad de Agua Útil del suelo en valor difuso
Agua_gr	Numérico(4,2)	Porcentaje de Agua Útil del suelo en valor preciso
CEC	Numérico (FDT2)	Cantidad de CEC del suelo en valor difuso
CEC_gr	Numérico(4,2)	Porcentaje de CEC del suelo en valor preciso

Tabla C.5: Descripción de las propiedades de la clase *Identificación*

Atributo	Tipo	Descripción
Cod_ecol	Numérico	Código ecológico
Cod_per	Numérico	Código de Perfil
Cod_hori	Numérico	Código de horizonte

Tabla C.6: Descripción de las propiedades de la clase *Bibliografía*

Atributo	Tipo	Descripción
biblio	Cadena(14)	Identificador del lugar donde se encuentra el lugar de la encuesta
autores	Cadena(8)	Identificador del encuestador

Tabla C.7: Descripción de las propiedades de la clase *Color* y sus subclases

Atributo	Clase	Tipo	Descripción
Codigo_Color	Color	Numérico	Identificador del Registro
hue_hume	Húmedo	Categórico	Valor del Hue del Color en entorno húmedo
value_hu	Húmedo	Categórico	Valor del color de suelo en entorno húmedo
croma_hu	Húmedo	Categórico	Valor del cromado del suelo en entorno húmedo
value_se	Seco	Categórico	Valor del color de suelo en entorno seco
croma_se	Seco	Categórico	Valor del cromado del suelo en entorno seco
hue_se	Seco	Categórico	Valor del Hue del Color en entorno seco

Tabla C.8: Etiquetas lingüísticas (Atributo PMEDIA)

Etiqueta	α	β	γ	δ
Baja	183	183	315	490
Media	490	664	731	818
Alta	818	905	1287	1287

Tabla C.9: Etiquetas lingüísticas (Atributo TMEDIA)

Etiqueta	α	β	γ	δ
Baja	0	0	6.5	8.5
Media	8.5	10.5	12.5	14.7
Alta	14.7	16.9	21.0	21.0

Tabla C.10: Etiquetas lingüísticas (Atributo ALTITUD)

Etiqueta	α	β	γ	δ
Baja	65	65	380	860
Media	860	1341	1460	1700
Alta	1700	1940	3020	3020

Tabla C.11: Etiquetas lingüísticas (Atributo PROFUNDI)

Etiqueta	α	β	γ	δ
Baja	2	2	12	17
Media	17	22	28	37
Alta	37	45	66	66

Tabla C.12: Etiquetas lingüísticas (Atributo PENDIENT)

Etiqueta	α	β	γ	δ
Flat	0	0	1	2
Gently sloping	2	3	4	5
Sloping	5	6	9	10
Strongly sloping	10	11	14	15
Moderately steep	15	16	29	30
Steep	30	31	59	60
Very steep	60	61	100	100

Tabla C.13: Etiquetas lingüísticas (Atributo ARENA)

Etiqueta	α	β	γ	δ
Baja	0.4	0.4	21.2	30.6
Media	30.6	40.0	48.9	56.4
Alta	56.4	63.9	91	91

Tabla C.14: Etiquetas lingüísticas (Atributo ARCILLA)

Etiqueta	α	β	γ	δ
Baja	1.31	1.31	10.0	15.0
Media	15.0	20.0	26.0	33.1
Alta	33.1	40.1	69.5	69.5

Tabla C.15: Etiquetas lingüísticas (Atributo CO)

Etiqueta	α	β	γ	δ
Baja	0	0	0.37	0.57
Media	0.57	0.77	1.40	1.94
Alta	1.94	2.48	19.5	19.5

Tabla C.16: Etiquetas lingüísticas (Atributo CARBONAT)

Etiqueta	α	β	γ	δ
Baja	0.00	0.00	8.2	15.75
Media	15.75	23.3	31.0	46.4
Alta	46.4	61.8	85.60	85.60

C.1.4. Relaciones de Similitud de los TD3

En cuanto a las variables de Tipo Difuso 3, se han de definir las etiquetas que forman cada una de ellas y la relación que existe entre dichas etiquetas mediante tablas de similitud. A continuación se describe dicha información. Así tendremos en la tabla *Estructura*: *Tipo_Es* descrita en las tablas en la C.38 y C.39, *Grado_es* en la C.40, *Vegetacion* en la y C.28, *Material* en la C.29 y C.30, y *Grado_de* en la C.31. En la tabla *Color* encontramos: *hue_hume* descrito en la tabla C.32, *value_hu* en la C.33, *croma_hu* en la C.34, *hue_se* en la C.35, *value_se* en la C.36 y *croma_se* en la C.37. Por último en la tabla *Localización*: *Orientación* esta descrita en la tabla C.25, *fisiografía* en la C.26, *tipo_hori* en la C.24 y *fao_reduc* en las tablas C.22 y C.23

Tabla C.17: Etiquetas lingüísticas (Atributo PH)

Etiqueta	α	β	γ	δ
Baja	0.37	0.37	5.60	6.35
Media	6.35	7.10	7.50	7.85
Alta	7.85	8.20	8.90	8.90

Tabla C.18: Etiquetas lingüísticas (Atributo AGUA)

Etiqueta	α	β	γ	δ
Baja	0.06	0.06	0.8	1.1
Media	1.1	1.40	1.70	2.0
Alta	2.0	2.30	8.6	8.6

Tabla C.19: Etiquetas lingüísticas (Atributo FE)

Etiqueta	α	β	γ	δ
Baja	0.0	0.0	0.9	1.25
Media	1.25	1.60	2.1	2.9
Alta	2.9	3.7	5.70	5.70

Tabla C.20: Etiquetas lingüísticas (Atributo CEC)

Etiqueta	α	β	γ	δ
Baja	0.26	0.26	6.54	9.01
Media	9.01	11.48	17.21	25.11
Alta	25.11	33.0	53.20	53.20

Tabla C.21: Etiquetas lingüísticas (Atributo CLASE_ES)

Etiqueta	α	β	γ	δ
Very fine	0	0	0.75	1.0
Fine	1.0	1.25	1.75	2.0
Medium	2.0	2.25	4.75	5.0
Coarse	5.0	5.25	9.75	10.0
Very coarse	10.0	10.25	20.0	20.0

Tabla C.22: Relaciones de similitud (Atributo FAOREDUC)

FAOREDUC	2	3	4	5	6	7	8	9	10	11	12	13
1	0.3	0.3	0.5	0.3	0.3	0.3	0.3	0.5	0.3	0.5	0.3	0.5
2		0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.5	0.3
3			0.3	0.5	0.3	0.3	0.5	0.3	0.3	0.3	0.3	0.3
4				0.3	0.3	0.3	0.3	0.5	0.3	0.5	0.3	0.5
5					0.3	0.3	0.5	0.3	0.3	0.3	0.3	0.3
6						0.3	0.3	0.3	0.3	0.3	0.3	0.3
7							0.3	0.3	0.3	0.3	0.3	0.3
8								0.3	0.3	0.3	0.3	0.3
9									0.3	0.5	0.3	0.5
10										0.3	0.3	0.3
11											0.3	0.5
12												0.3

Tabla C.23: Códigos para el atributo FAOREDUC

Valor	clave
Arenosol	1
Cambisol	2
Chernozems	3
Fluvisol	4
Kastanozems	5
Litosol	6
Luvisol	7
Phaeozems	8
Regosol	9
Rendzina	10
Solonchack	11
Xerosol	12
Yermosol	13

Tabla C.24: Relaciones de similitud (Atributo TIPO_HOR)

TIPO_HOR	Ah	Ap	Bk	Bt	Btk	Bw	Bwk	C	Ck
A	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
Ah		0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
Ap			0.3	0.3	0.3	0.3	0.3	0.3	0.3
Bk				0.3	0.3	0.3	0.3	0.3	0.3
Bt					0.3	0.3	0.3	0.3	0.3
Btk						0.3	0.3	0.3	0.3
Bw							0.3	0.3	0.3
Bwk								0.3	0.3
C									0.3

Tabla C.25: Relaciones de similitud (Atributo ORIENTAC)

ORIENTAC	NE	E	SE	S	SW	W	NW
N	0.4	0.4	0.4	0.4	0.4	0.4	0.4
NE		0.4	0.4	0.4	0.4	0.4	0.4
E			0.4	0.4	0.4	0.4	0.4
SE				0.4	0.4	0.4	0.4
S					0.4	0.4	0.4
SW						0.4	0.4
W							0.4

Tabla C.26: Relaciones de similitud (Atributo FISIOGRA)

FISIOGRA	Fondo ladera	Ladera	Cima	Meseta
Llano	0.5	0.2	0.2	0.2
F. lad.		0.2	0.2	0.2
Ladera			0.2	0.2
Cima				0.5

Tabla C.27: Relaciones de similitud (Atributo VEGETACI)

VEGETACI	2	3	4	5	6	7	8
1	0.5	0.5	0.2	0.2	0.2	0.2	0.2
2		0.5	0.2	0.2	0.2	0.2	0.2
3			0.2	0.2	0.2	0.2	0.2
4				0.5	0.5	0.5	0.5
5					0.5	0.5	0.5
6						0.5	0.5
7							0.5

Tabla C.28: Códigos para el atributo VEGETACI

Valor	clave
Bosque natural	1
Bosque de repoblación	2
Matorral alto	3
Herbácea	4
Matorral bajo degradado	5
Cultivo arbolado	6
Cultivo herbáceo	7
Regadío	8

Tabla C.29: Relaciones de similitud (Atributo MATERIAL)

MATERIAL	2	3	4	5	6	7	8	9
1	0.2	0.2	0.2	0.5	0.2	0.2	0.2	0.0
2		0.2	0.2	0.2	0.5	0.2	0.2	0.0
3			0.2	0.2	0.2	0.2	0.2	0.0
4				0.2	0.2	0.2	0.2	0.0
5					0.2	0.2	0.2	0.0
6						0.2	0.2	0.0
7							0.2	0.0
8								0.0

Tabla C.30: Códigos para el atributo MATERIAL

Valor	clave
Ácido coluvial	1
Ácido aluvial	2
Ácido sobre mat. compacto	3
Ácido sobre mat. no compacto	4
Calcáreo coluvial	5
Calcáreo aluvial	6
Calcáreo sobre mat. compacto	7
Calcáreo sobre mat. no compacto	8
Roca volcánica	9

Tabla C.31: Relaciones de similitud (Atributo GRADO)

GRADO	Moderate	Severe
Slight	0.5	0.5
Moderate		0.5

Tabla C.32: Relaciones de similitud (Atributo HUE_HUME)

HUE_HUME	1.25YR	2.5YR	3.75YR	6.75YR	7.5YR	8.75YR	10YR	1.25Y	2.5Y	5Y	10Y
10R	0.3	0.3	0.3	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.0
1.25YR		0.3	0.3	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.0
2.5YR			0.3	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.0
3.75YR				0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.0
6.75YR					0.5	0.3	0.3	0.3	0.3	0.1	0.0
7.5YR						0.3	0.3	0.3	0.3	0.1	0.0
8.75YR							0.3	0.3	0.3	0.1	0.0
10YR								0.3	0.3	0.1	0.0
1.25Y									0.3	0.1	0.0
2.5Y										0.1	0.0
5Y											0.0

Tabla C.33: Relaciones de similitud (Atributo VALUE_HU)

VALUE_HU	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	8
2	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.2
2.5		0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.2
3			0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.2
3.5				0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.2
4					0.3	0.3	0.3	0.3	0.3	0.3	0.2
4.5						0.3	0.3	0.3	0.3	0.3	0.2
5							0.3	0.3	0.3	0.3	0.2
5.5								0.3	0.3	0.3	0.2
6									0.3	0.3	0.2
6.5										0.3	0.2
7											0.2

Tabla C.34: Relaciones de similitud (Atributo CROMA_HU)

CROMA_HU	0.5	1	1.5	2	2.5	3	3.5	4	5	6	7	8
0	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0.2	0.2	0.2
0.5		0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0.2	0.2	0.2
1			0.3	0.3	0.3	0.3	0.3	0.3	0.2	0.2	0.2	0.2
1.5				0.3	0.3	0.3	0.3	0.3	0.2	0.2	0.2	0.2
2					0.3	0.3	0.3	0.3	0.2	0.2	0.2	0.2
2.5						0.3	0.3	0.3	0.2	0.2	0.2	0.2
3							0.3	0.3	0.2	0.2	0.2	0.2
3.5								0.3	0.2	0.2	0.2	0.2
4									0.2	0.2	0.2	0.2
5										0.2	0.2	0.2
6											0.2	0.2
7												0.2

Tabla C.35: Relaciones de similitud (Atributo HUE_SECO)

HUE_SECO	2.5YR	3.75YR	5YR	6.75YR	7.5YR	8.75YR	10YR	1.25Y	2.5Y	5Y
10R	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2.5YR		0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.1
3.75YR			0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.1
5YR				0.3	0.3	0.3	0.3	0.3	0.3	0.1
6.75YR					0.5	0.3	0.3	0.3	0.3	0.1
7.5YR						0.3	0.3	0.3	0.3	0.1
8.75YR							0.3	0.3	0.3	0.1
10YR								0.3	0.3	0.1
1.25Y									0.3	0.1
2.5Y										0.1

Tabla C.36: Relaciones de similitud (Atributo VALUE_SE)

VALUE_SE	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5
3	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
4		0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
4.5			0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
5				0.3	0.3	0.3	0.3	0.3	0.3	0.3
5.5					0.3	0.3	0.3	0.3	0.3	0.3
6						0.3	0.3	0.3	0.3	0.3
6.5							0.3	0.3	0.3	0.3
7								0.3	0.3	0.3
7.5									0.3	0.3
8										0.3

Tabla C.37: Relaciones de similitud (Atributo CROMA_SE)

CROMA_SE	1	1.5	2	2.5	3	3.5	4	4.5	5	6	7	7.5	8
0	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
1		0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.3	0.3	0.3	0.3
1.5			0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.3	0.3	0.3	0.3
2				0.5	0.5	0.5	0.5	0.5	0.5	0.3	0.3	0.3	0.3
2.5					0.5	0.5	0.5	0.5	0.5	0.3	0.3	0.3	0.3
3						0.5	0.5	0.5	0.5	0.3	0.3	0.3	0.3
3.5							0.5	0.5	0.5	0.3	0.3	0.3	0.3
4								0.5	0.5	0.3	0.3	0.3	0.3
4.5									0.5	0.3	0.3	0.3	0.3
5										0.3	0.3	0.3	0.3
6											0.3	0.3	0.3
7												0.5	0.5
7.5													0.5

Tabla C.38: Relaciones de similitud (Atributo TIPO_ES)

TIPO_ES	2	3	4	5	6	7	8	9
1	0.4	0.4	0.4	0.4	0.2	0.2	0.2	0.2
2		0.4	0.4	0.4	0.2	0.2	0.2	0.2
3			0.4	0.4	0.2	0.2	0.2	0.2
4				0.4	0.2	0.2	0.2	0.2
5					0.2	0.2	0.2	0.2
6						0.2	0.2	0.2
7							0.2	0.2
8								0.2

Tabla C.39: Códigos para el atributo TIPO_ES

Valor	clave
Granular	1
Migajosa	2
Subangular blocky	3
Angular blocky	4
Prismatic	5
Platy	6
Rock structure	7
Massive	8
Single grain	9

Tabla C.40: Relaciones de similitud (Atributo GRADO_ES)

GRADO_ES	Weak	Moderate	Strong
Very Weak	0.3	0.3	0.3
Weak		0.3	0.3
Moderate			0.3

C.2. Cuerpo de la Base de Datos

En esta sección se visualizan algunos de los datos que hay contenidos en la BD Difusa de Suelos, que pueden ser útiles a la hora de ejemplificar cualquier implementación desarrollada en esta tesis. A continuación se muestra el contenido de algunas de estas tablas. El contenido completo de la BD se adjunta en el archivo denominado *bdfuzzy* que se proporciona con la tesis en formado CD-ROM.

Tabla C.41: Tabla *Color*, parte de su contenido

Codigo_c	hue_hume	value_hu	croma_hu	hue_se	value_se	croma_se
1	10YR	3.0	1	10YR	5	1
2	10YR	3.0	1	10YR	5	1
5	2,5YR	8	0	2,5YR	7	2
16	7,5YR	4	0	7,5YR	6	4
26	10YR	2	0	10YR	3	3
28	10YR	2	0	10YR	4	1
...	...	...	...	...	...	...

Tabla C.42: Tabla *Estructura*, parte de su contenido

codigo_es	tipo_es	clas_es	grado_es	vegetacion	material	grado_de
1	PLATY	FINE	WEAK	5	3	
2	MIGAJOSA	VERY FINE	WEAK	5	3	
5	MASSIVE			5	7	
16	MASSIVE			5	7	SLIGHT
26	MIGAJOSA	MEDIUM	MODERATE	3	3	SEVERE
28	SINGLE GRAIN			5	1	MODERATE
...	...	...	...	...	...	...

Tabla C.43: Tabla *Analíticos*, parte de su contenido

codigo_a	arena	arcilla	Co	PH	Fe	Agua	CEC
1	alta	baja	baja	baja	alta	media	baja
2	alta	baja	media	baja	media	media	baja
5	baja	media	baja	alta	alta	alta	baja
16	alta	baja	baja	baja	alta	media	baja
26	alta	baja	alta	baja	media	baja	media
28	alta	baja	baja	baja	alta	baja	baja
...	...	...	...	...	...	...	...

Tabla C.44: Tabla *Localización*, parte de su contenido

latitud	longitud	orientación	fisiografía	pendiente	altitud	...
41045	5478	SW	LADERA	STEEP	baja	...
41135	5598	NW	LADERA	STEEP	baja	...
4103	5705	NW	LADERA	GENTLY SLOPING	baja	...
41082	5675	LLANO	FLAT	baja	media	...
40963	5636	N	LADERA	STEEP	media	...
41049	5578		LLANO	FLAT	baja	...
...	...	...	...	...	...	...

...	profundidad	Tmedia	Pmedia	Fao_Reduc	biblio	autor	...
...	media	baja	alta	REGOSOL	TABERNAS	PEREZ	...
...	baja	baja	alta	REGOSOL	TABERNAS	PEREZ	...
...	media	baja	alta	XEROSOL	TABERNAS	PEREZ	...
...	baja	baja	alta	XEROSOL	TABERNAS	PEREZ	...
...	media	baja	media	REGOSOL	TABERNAS	PEREZ	...
...	media	baja	alta	FLUVISOL	TABERNAS	PEREZ	...
...	...	...	...	...	...	...	...

...	codecol	codhori	codperf	codigo_es	codigo_a	codigo_c
...	SE	1	1	1	1	1
...	SE	2	2	2	2	2
...	SE	5	3	5	5	5
...	SE	16	7	16	16	16
...	SE	26	10	26	26	26
...	SE	28	11	28	28	28

Bibliografía

[Ada80] J. M. Adamo. Fuzzy decision trees. *Fuzzy Sets and Systems*, 4:207–219, 1980.

[Aga05] P. Agarwal. Ontological considerations in giscience. *International Journal of Geographical Information Science*, 19(5):501–536(36), May 2005.

[An04] Y. An, A. Borgida y John Mylopoulos. Refining semantic mappings from relational tables to ontologies. En Val Tannen Christoph Bussler y Irini Fundulaki, editores, *Semantic Web and Databases, Second International Workshop, SWDB 2004*, tomo 3372, páginas 84–90. Springer, 2004.

[Ank07] A. Ankolekar, M. Krötzsch, T. Tran y D. Vrandecic. The two cultures: mashing up web 2.0 and the semantic web. En *WWW '07: Proceedings of the 16th international conference on World Wide Web*, páginas 825–834. ACM, New York, NY, USA, 2007. ISBN 978-1-59593-654-7.

[Ant03] G. Antoniou y F. van Harmelen. *Handbook on Ontologies in Information Systems*, capítulo Web Ontology Language: OWL, páginas 67–92. Springer-Verlag, 2003.

[Apa05] A. S. Aparcio, O. L. M. Farias y N. dos Santos. Applying ontologies in the integration of heterogeneous relational databases. En *AOW '05: Proceedings of the 2005 Australasian Ontology Workshop*, páginas 11–16. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 2005. ISBN 1-920-68240-6.

[Arp01] J. C. Arperez, O. Corcho, M. Fernandez-López y A. Gómez-Pérez. Webode: a scalable workbench for ontological engineering. En *K-CAP '01: Proceedings of the 1st international conference on Knowledge capture*, páginas 6–13. ACM Press, New York, NY, USA, 2001. ISBN 1-58113-380-4.

[Ast04] I. Astrova. Reverse engineering of relational databases to ontologies. En *Proceedings of the 1st Europan Semantic Web Symposium (ESWS), LNCS*, tomo 3053, páginas 327–341. 2004.

[Ast05] I. Astrova. Towards the semantic web - an approach to reverse engineering of relational databases to ontologies. En *Advances in Databases and Information Systems: 9th East-European Conference, ADBIS 2005*, páginas 111–122. September 2005.

[atSUSoM06] Stanford Medical Informatics at the Stanford University School of Medicine. Protege. http://protege.stanford.edu/, January 2006.

[Aue07] S. Auer. powl. a web based platform for collaborative semantic web development. http://powl.sourceforge.net/overview.php, 2007.

[Aum05] D. Aumueller, H. Do, S. Massmann y E. Rahm. Schema and ontology matching with coma++. En *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, páginas 906–908. ACM Press, New York, NY, USA, 2005. ISBN 1-59593-060-4.

[Baa04] F. Baader, I. Horrocks y U. Sattler. *Handbook on Ontologies*, capítulo Description Logics, páginas 1–28. Springer, 2004.

[Bac07] D. Bui Bach. *Import/Export of OWL Ontologies into/from DOGMA*. Master of computer science, Vjire Universiteit Brussel. Faculty of Science. Departament of Computer Science. Semantic Technology and Applications Lab, 2006-2007.

[Bal79] J. F. Baldwin y N. C. F. Guild. Comparison of fuzzy sets on the same decision space. *Fuzzy Sets and Systems*, 2:213–233, 1979.

[Bal84] J. F. Baldwin y S. Q. Zhou. A fuzzy relational inference language. *Fuzzy Sets and Systems*, (14):155–174, 1984.

[Bar03] J. Barrasa, O. Corcho y A. G. . Perez. Fund finder: A case study of database to ontology mapping. En *In International Semantic Web Conference, number 2870 in Lecture Notes in Computer science,*, páginas 17–22. Springer-Verlag., 2003.

[Bas77] S. M. Bass y H. Kwakernaak. Rating and ranking of multiple-aspect alternatives using fuzzy sets. *Automatica*, 13:47–58, 1977.

[Bec] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. Mcguinness, P. F. Patel-Schneider y L. A. Stein. Owl web ontology language reference. Informe técnico, W3C.

[Bec07a] S. Bechhofer. Api for owl ontologies. http://owl.man.ac.uk/api/readme.html, 2007.

[Bec07b] S. Bechhofer y G.Ñg. Oiled. http://oiled.man.ac.uk/, April 2007.

[Bec07c] S. Bechhofer y R. Volz. Wonderweb owl ontology validator. http://www.mygrid.org/OWL/Validator, 2007.

[Ben06] S. M. Benslimane, D. Benslimane, M. Malki, Y. Amghar y H. Saliah. Acquiring OWL ontologies from data-intensive web sites. En ACM, editor, *The Sixth International Conference on Web Engineering (ICWE'06)*, páginas 361–368. julio 2006.

[Biz03] C. Bizer. D2r map - a database to rdf mapping language. En *In 12th Intl World Wide Web Conf*, páginas 17–22. 2003.

[BL01] T. Berners-Lee, J. Hendler y O. Lassila. The semantic web. *Scientific American*, 284(5):28–37, May 2001.

[BL06] T. Berners-Lee, W. Hall, J. A. Hendler, K. O'Hara, N. Shadbolt y D. J. Weitzner. A framework for web science. *Foundations and Trends® in Web Science*, 1(1), 2006.

[Bla00a] I. Blanco, J. C. Cubero, O. Pons y M. A. Vila. An implementation for fuzzy relational databases. En G. Bordogna y G. Passi, editores, *Recent Research Issues on the Management of Fuzziness in Databases*, Studies in Fuzziness and Soft Computing, páginas 183–207. Physica-Verlag, 2000.

[Bla00b] I. Blanco, N. Marín, O. Pons y M. A. Vila. An extension of data description language (ddl) for fuzzy data handling. En Larsen, Kacprzyk, Zadrozny, Andreasen y Christiansen, editores, *Flexible Query Answering Systems, Recent Advances*, Advances in Soft Computing, páginas 54–64. Physica-Verlag, 2000.

[Bla01] I. Blanco. *Deducción en Bases de Datos Relationales Difusas*. Tesis Doctoral, E. T. S. I. Informática, Universidad de Granada, Spain, 2001.

[Bla02a] I. Blanco, M. J. Martín-Bautista, O. Pons y M. A. Vila. A mechanism for deduction in a fuzzy relational database. En *Proceedings of the 9th Information Processing and Management of Uncertainty in Knowledge-based Systems IPMU 2002 Conference*, páginas 291–298. Annecy, France, July 2002.

[Bla02b] I. Blanco, D. Sánchez, J.M. Serrano, M.A. Vila y J. Fuzzy-queries 2+, una herramienta para la integración de consultas flexibles, cálculo de agregaciones y resúmenes, y extracción de conocimiento. En *Actas del XI Congreso Español sobre tecnologías y Lógica Fuzzy (ESTYLF02)*, páginas 337–342. 2002.

[Bla03a] I. Blanco, M. J. Martin-Bautista, O. Pons y M. A. Vila. A mechanism for deduction in a fuzzy relational database. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 11:47–66, September 2003.

[Bla03b] I. Blanco, O. Pons, J. M. Serrano y M. A. Vila. Deduction in a gefred database using datalog. En *International Conference in Fuzzy Logic and Technology EUSFLAT 2003*, páginas 550–553. September 2003.

[Bla04] I. Blanco, C. Martinez-Cruz, J.M. Serrano y M. A. Vila. Servidor de bases de datos relacionales difusas para deducción y minería de datos. En *Actas del XII Congreso Español Sobre Teconologías y Lógica Fuzzy*, páginas 135–141. 2004.

[Bla05a] I. Blanco, C. Martinez-Cruz, J.M. Serrano y M.A. Vila. A first approach to multipurpose relational database server. *Mathware and Soft Computing*, 12(2-3):129–153, 2005.

[Bla05b] I. Blanco, C. Martínez-Cruz, N. Marín y M. A. Vila. About the use of ontologies for fuzzy knowledge representation. En *International Conference in Fuzzy Logic and Technology EUSFLAT 2005*, páginas 106–111. September 2005.

[Bla07] I. Blanco, C. Martínez-Cruz y M. A. Vila. *Handbook of Research on Web Information Systems Quality*, capítulo Looking for Information in Fuzzy Relational Databases accessible via the

Web, páginas Chapter XVIII,300–324. Idea Group Reference, 2007.

[Bla08a] I. Blanco, C. Martinez-Cruz y M. A. Vila. Arquitectura para la integración de esquemas relacionales difusos basada en ontologías: una aplicación para la web. En *Actas del XIV Congreso Español sobre tecnologías y Lógica Fuzzy (ESTYLF08)*, páginas 651–656. 2008.

[Bla08b] I. J. Blanco, M. A. Vila y Carmen Martinez-Cruz. The use of ontologies for representing database schemas of fuzzy information. *International Journal of Intelligent Systems*, 23(4):419–445, February 2008.

[Bor97] P. Borst, H. Akkermans y J. Top. Engineering ontologies. *Int. J. Hum.-Comput. Stud.*, 46(2-3):365–406, 1997. ISSN 1071-5819.

[Bos88] P. Bosc, M. Galibourg y G. Hamon. Fuzzy querying with sql: Extensions and implementation aspects. *Fuzzy Sets and Systems*, 28:333–349, 1988.

[Bre04] M. Breu y Y. Ding. Modelling the world: Databases and ontologies. *Whitepaper by IFI, Institute of Computer Science, University of Innsbruck*, 2004.

[Bre07] C. Brewster y K. O'Hara. Knowledge representation with ontologies: Present challenges–future possibilities. *International Journal of Human-Computer Studies*, 65(7):563–568, July 2007.

[Bro02] J. Broekstra, A. Kampman y F. van Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. En I. Horrocks y J. Hendler, editores, *The Semantic Web - ISWC 2002. First International Semantic Web Conference, Sardinia, Italy, June 9-12, 2002, Proceedings*, tomo 2342 de *LNCS*, páginas 54–68. Springer, 2002.

[Bro06] S. Brockmans, P. Haase y P. Hitzler. A metamodel and uml profile for rule-extended owl dl ontologies. En J. Domingue Y. Sure, editor, *The Semantic Web: Research and Applications: 3rd European Semantic Web Conference, ESWC 2006*, tomo 4011, páginas 303–316. 2006.

[Buc82a] B. P. Buckles y F. E. Petry. *Fuzzy Information and Decision Processes*, tomo 2, capítulo Fuzzy Databases and their Applications, páginas 361–371. North-Holland, Amsterdam, 1982.

[Buc82b] B. P. Buckles y F. E. Petry. A fuzzy representation of data for relational databases. *Fuzzy Sets and Systems*, (7):213–226, 1982.

[Buc84] B. P. Buckles y F. E. Petry. Extending the fuzzy database model with fuzzy numbers. *Information Sciences*, 34:145–155, 1984.

[Buc89] B. P. Buckles, F. E. Petry y H. S. Sachar. A domain calculus for fuzzy relational databases. *Fuzzy Sets and Systems*, 29:327–340, 1989.

[Cal05] C. Calero, F. Ruiz, A. Baroni, F. Brito e Abreu y M. Piattini. An ontological approach to describe the sql:2003 object-relational features. *Computer Standards and Interfaces Journal*, páginas 1–28, 2005.

[Cal06] C. Calero y M. Piattini. *An Ontological Approach to SQL:2003*, capítulo Ontological Engineering: Principles, Methods, Tools and Languages, páginas 49–102. Springer-Verlag, 2006.

[Car98] R. A. Carrasco, J. Galindo, M.C. Aranda, J.M. Medina y M.A. Vila. Classification in databases using a fuzzy query language. En *9th International Conference on Management of Data, COMAD'98*. 1998.

[Car99] R. A. Carrasco, J. Galindo, M.A. Vila y J.M. Medina. Clustering and fuzzy classification in a financial data mining environment. En *3rd International ICSC Symposium on Soft Computing, SOCO'99*, páginas 713–720. 1999.

[Car00] R. A. Carrasco, J. Galindo, M.A. Vila y J.C. Cubero. Fsql: a tool for obtaining fuzzy dependencies. En *8th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU'2000*, páginas 1916–1919. 2000.

[Car03a] R. A. Carrasco. *Lenguajes e Interfaces de Alto Nivel para Data Mining con Aplicación Práctica a Entornos Financieros*. Tesis Doctoral, E. T. S. I. Informática, Universidad de Granada, Spain, 2003.

[Car03b] R. A. Carrasco, M. A. Vila y J. Galindo. Fsql: a flexible query language for data mining. *Enterprise information systems IV*, páginas 68–74, 2003.

[Car07] J. Cardoso. The semantic web vision: Where are we? *IEEE Intelligent Systems*, 22(5):84–88, 2007. ISSN 1541-1672.

[Cas07] P Casanovas, N. Casellas, C. Tempich, D. Vrandecic y R. Benjamins. Opjk and diligent: ontology modeling in a distributed environment. *Artificial Intelligence and Law*, 15(2):171–186, June 2007. ISSN 0924-8463.

[Cha99] B. Chandrasekaran, J.R. Josephson y V.R. Benjamins. What are ontologies, and why do we need them? *IEEE Intelligent Systems*, páginas 20–26, January/February 1999.

[Cha01] P. A. Champin. *RDF Tutorial*. http://www710.univ-lyon1.fr/ champin/rdf-tutorial/, April 2001.

[Cha07] P. A. Champin, G.J. Houben y Ph. Thiran. Cross: An owl wrapper for reasoning on relational databases. En C. Parent, K.D. Schewe, Veda C. Storey y Bernhard Thalheim, editores, *ER*, tomo 4801 de *Lecture Notes in Computer Science*, páginas 502–517. Springer, 2007. ISBN 978-3-540-75562-3.

[Che92] G. Q. Chen, j. Vandenbulcke y E. E. Kerre. A general treatment of data redundancy in a fuzzy relational data model. *Journal American Society of Information Sciences*, 43(3):304–311, 1992.

[Che99] G. Q. Chen. *Fuzzy Logic in Data Modeling; Semantics, Constraints and Database Desing*. kluwer Academic Publisher, 1999.

[Cho06] N. Choi, I.Y. Song y H. Han. A survey on ontology mapping. *SIGMOD Rec.*, 35(3):34–41, 2006. ISSN 0163-5808.

[Cod70] E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.

[Cod79] E. F. Codd. Extending the database relational model to capture more meaning. *ACM Transactions on Database Systems*, 4:262–296, 1979.

[Cod86] E. F. Codd. Missing information (applicable and inapplicable) in relational databases. *ACM SIGMOD Record*, 15(4), 1986.

[Cod87] E. F. Codd. More commentary on missing information in relational databases. *ACM SIGMOD Record*, 16(1), 1987.

[Cod90] E. F. Codd. *The Relational Model for Database Management, Version 2*. Reading Mass. Addison-Wesley, 1990.

[Cod07] E. F. Codd. Relational database: A practical foundation for productivity. *ACM Turing award lectures*, página 1981, 2007.

[Cor06] O. Corcho, M. FernándezLópez y A. GómezPérez. *Ontologies for Software Engineering and Software Technology*, capítulo Ontological Engineering: Principles, Methods, Tools and Languages, páginas 49–102. Springer-Verlag, 2006.

[Cul03] N. Cullot, C. Parent, S. Spaccapietra y Christelle Vangenot. Ontologies: A contribution to the dl/db debate. En Isabel Cruz y Vipul Kashyap, editores, *First International Workshop on Semantic Web and Databases (VLDB workshop)*. September 2003.

[Del88] M. Delgado, J. L. Verdegay y M. A. Vila. A procedure for ranking fuzzy numbers using fuzzy relations. *Fuzzy Sets and Systems*, 26:49–62, 1988.

[Doa02] A. Doan, J. Madhavan, P. Domingos y A. Halevy. Learning to map between ontologies on the semantic web. En *The Eleventh International WWW Conference*. Hawaii,, 2002.

[Dou06] D. Dou y P. LePendu. Ontology-based integration for relational databases. En *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, páginas 461–466. ACM Press, New York, NY, USA, 2006. ISBN 1-59593-108-2.

[Dub83] H. Dubois y H. Prade. Ranking fuzzy numbers in the setting of possibility theory. *Information Sciences*, 30:183–224, 1983.

[Dui00] A. J. Duineveld, R. Stoter, M. R. Weiden, B. Kenepa y V. R. Benjamins. Wondertools? a compartive study of ontological engineering tools. *International Journal of Human-Cumputer Studies*, 52(6):1111–1133, Jun 2000.

[Ech07a] P. Echarte. La web semántica. http://www.lawebsemantica.com/contents/webSemantica/ontologias4.html, 2007.

[Ech07b] P. Echarte. Técnicas y lenguajes para la representación del conocimiento.

http://www.eslomas.com/index.php/archives/2006/12/14/tecnicas-y-lenguajes-para-la-representacion-del-conocimiento/, 2007.

[EG06] H. El-Ghalayini, M. Odeh y R. McClatchey. Engineering conceptual data models from domain ontologies: A critical evaluation. *CoRR*, abs/cs/0601119, 2006. Informal publication.

[Ehr07] M. Ehrig. *Ontology Alignment. Bringing the Semantic Gap.*. Springer, 2007.

[Eis04] A. Eisenberg, J. Melton, K. G. Kulkarni, J-E Michels y Fred Zemke. Sql: 2003 has been published. *SIGMOD Record*, 33:119–126, 2004.

[Eri07] Ulric Eriksson. Libsbd: Database library for supporting multiple database management. http://siag.nu/libsdb/, January 2007.

[Euz07] J. Euzenat y P. Shvaiko. *Ontology Matching*. Springer, 2007.

[Fen04] D. Fensel. *Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce*. Springer-Verlag, Berlin, 2nd edición, 2004.

[Fin05] T. Finin, J. Mayfield, A. Joshi, R. S. Cost y C. Fink. Information retrieval and the semantic web. En *HICSS '05: Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 4*, página 113.1. IEEE Computer Society, Washington, DC, USA, 2005. ISBN 0-7695-2268-8-4.

[fSIIT03] International Organization for Standardization (ISO). Information Technology. Database language sql. parts 1 to 4 and 9 to 14. *9075-1:2003 to 9075-14:2003 International Standards*, (Standard No. ISO/IEC 9075:2003), September,2003.

[fSIIT99] International Organization for Standardization (ISO). Information Technology. Ansi/iso/iec international standard (is) database language sql part 2: Foundation (sql/foundation). *ISO/IEC 9075-2:1999 (E)*, September, 99.

[Fuk79] S. Fukami, M. Umano, M. Muzimoto y H. Tanaka. Fuzzy databases retrieval and manipulation language. *IEICE Technical Reports*, 78(233):65–72, 1979.

[Gae06] D. Gaevic, D. Djuric, V. Devedic y B. Selic. Model driven architecture and ontology development. *Springer*, 2006.

[Gal84] H. Gallaire, J. Minker y J. M. Nicholas. Logic and databases: A deductive approach. *Computing Surveys*, 16(2):153–185, June 1984.

[Gal99] J. Galindo. *Tratamiento de la Imprecisión en Bases de Datos Relacionales: Extensión del modelo y adaptación de los SGBD actuales*. Tesis Doctoral, Department of Computer Science and Artificial Intelligence, University of Granada, España, 1999.

[Gal05a] A. Gal, G. Modica, H. Jamil y A. Eyal. Automatic ontology matching using application semantics. *AI Mag.*, 26(1):21–31, 2005. ISSN 0738-4602.

[Gal05b] A. Gali, C.X. Chen, K.T. Claypool y R. Uceda-Sosa. From ontology to relational databases. *Shan Wang et all(Eds.): Conceptual Modeling for Advanced Application Domains, LNCS*, 3289:278–289, 2005.

[Gal06] J. Galindo, A. Urrutia y M. Piattini. *Fuzzy Databases Modeling, Design and Implementation*. Idea Group Publishing, 2006.

[Gen06] J. Gennick. *SQL Pocket Guide*. O'Reilly, 2006.

[Gen07] J. Gennari, M.Ñguyen y A. Silberfein. Datagenie. http://protege.cim3.net/cgi-bin/wiki.pl?DataGenie, March 2007.

[Geo05] D. George. Understanding structural and semantic heterogeneity in the context of database schema integration. En *Journal of the Dept. of Computing*, 4, páginas 29–44. IEEE Computer Society, UCLan, May 2005. ISBN 1476-9069.

[Gob03] C. Goble. Guest editorial: the semantic web: an evolution for a revolution. *Comput. Networks*, 42(5):551–556, 2003. ISSN 1389-1286.

[Gog05] J. A. Goguen. Data, schema, ontology and logic integration. *Logic Journal of the IGPL*, 13(6):685–715, November 2005. ISSN 1367-0751.

[GP03a] A. Gómez-Pérez, M. Férnandez-López y O. Corcho-García. Metodologies, tools and languages for building ontologies. where

is their meeting point? *Data and Knowledge Engineering*, (46):41–64, 2003.

[GP03b] A. Gómez-Pérez, M. Férnandez-López y O. Corcho-García. *Ontological Engineering*. Springer-Verlag New York, Inc., 2003.

[GP04] A. Gómez-Pérez y D. Manzano-Macho. An overview of methods and tools for ontology learning from texts. *Knowl. Eng. Rev.*, 19(3):187–212, 2004. ISSN 0269-8889.

[Gra80] J. Grant. Incomplete information in a relational database. *Fundamenta Informaticae*, 3:363–378, 1980.

[Gru93] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *Technical Report KSL 93-04, Knowledge Systems Laboratory, Standford University*, 1993.

[Grü95] M. Grüninger y M. Fox. Methodology for the design and evaluation of ontologies. En *IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing, April 13, 1995*. 1995.

[Gua95] N. Guarino. Formal ontology, concept analysis and knowledge representation. *International Journal of Human and Computer Studies*, 43(5/6):625–640, 1995.

[Gua98] N. Guarino. Formal ontologies and information systems. En *Proc. of FOIS98*, páginas 3–15. 1998.

[Hü05] B. Hüsemann y G. Vossen. Ontology engineering form a database perspective. En S. Grumbach, L. Sui y V. Vianu, editores, *ASIAN 2005, LNCS 3818*, páginas 49–63. Springer-Verlag, 2005.

[Hai05] D. Hong Hai. *Schema Matching ans Mapping-Based Data Integration*. Tesis Doctoral, Interdisciplinary Center for Bioinformatics and Department of Computer Science. University of Leipzig. Germany, 2005.

[Ham04] A. Hameed, A. Preece y D. Sleeman. *Handbook on Ontologies*, capítulo Ontology Reconciliation, páginas 231–250. Springer, 2004.

[Har05] J. Hartmann, P. Spyns, A. Giboin, D. Maynard, R. Cuel, M. C. SuárezFigueroa y Y. Sure. Deliverable d1.2.3 methods for ontology evaluation. document identifier: Kweb/2004/d1.2.3/v1.3. Informe técnico, Knowledge Web Consortium, 2005.

[Hen02] J. Hendler, T. Berners-Lee y E. Miller. Integrating applications on the semantic web. *Journal of the Institute of Electrical Engineers of Japan*, 122(10):676–680, October 2002.

[Her02] M. C. Pérez Hernández. *Explotación de los córpora textuales informatizados para la creación de bases de datos terminológicas basadas en el conocimiento*. Estudios de Lingüística Española (ELiEs), http://elies.rediris.es/elies18/, 2002.

[Hol02] C.W. Holsapple y K. D. Joshi. A collaborative approach to ontology design. *Commun. ACM*, 45(2):42–47, 2002. ISSN 0001-0782.

[hom06] Ontology Engineering homepage. http://www.aifb.uni-karlsruhe.de/WBS/cte/ontologyengineering/, 2006.

[Hu96] J. Hu, D.Ñicholson, C. Mungall, A. L. Hillyard y A. L. Archibald. Webintool: a generic web to database interface building tool. En *DEXA '96: Proceedings of the 7th International Workshop on Database and Expert Systems Applications*, página 285. IEEE Computer Society, Washington, DC, USA, 1996. ISBN 0-8186-7662-0.

[Imi96] T. Imielinski y Heikki Mannila. A database perspective on knowledge discovery. *Communications of the ACM*, 39(11):58–64, 1996.

[Jar] M. Jarrar y R. Meersman. Scalability and knowledge reusability in ontology modeling. citeseer.ist.psu.edu/jarrar02scalability.html.

[Jar02] M. Jarrar y R. Meersman. Formal ontology engineering in the dogma approach. En Robert Meersman y Zahir Tari, editores, *CoopIS/DOA/ODBASE*, tomo 2519 de *Lecture Notes in Computer Science*, páginas 1238–1254. Springer, 2002. ISBN 3-540-00106-9.

[Jea06] S. Jean, G. Pierra y Y. AitAmeur. Domain ontologies: A database-oriented analisys. En *Proceedings of the Web Information Systems and Technologies (WEBIST'2006)*. April 2006.

[Jur99] I. Jurisica, J. Mylopoulos y E. Yu. Using ontologies for knowledge management: An information systems perspective. En *Proceedings of 62nd Annual Meeting of the American Society for Information Science (ASISI99)*, páginas 482–496. 1999.

[Jur07] D. Juric y Z. Skocir. Building owl ontologies by analyzing relational database schema concepts and wordnet semantic relations. En *The 9th International Conference on Telecommunications. ConTEL 2007*. June 13-15 2007.

[KAC05] Espirit Proyect 8145 KACTUS. The kactus. http://www.swi.psy.uva.nl/projects/NewKACTUS/ Reports.html, April 2005.

[Kal03] Y. Kalfoglou y M. Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18(1):1–31, 2003.

[Kam07] Arjohn Kampman y Jeen Broekstra. Sesame. http://www.openrdf.org/, 2007.

[Kas99] V. Kashyap. Design and creation of ontologies for environmental information retrieval, 1999.

[KBS] Inc. Knowledge Based System. Idef. integrated definition methods. http://www.idef.com/IDEF5.html.

[Kni94] K. Knight y S. K. Luk. Building a large-scale knowledge base for machine translation. En *Proceedings of the Twelfth National Conference on Artificial Intelligence.Seattle.Washington.*. AAAI Press, 1994.

[Knu] H. Knublauch. An ai tool for the real world. knowledge modeling with protègè. Informe técnico, http://www.javaworld.com/javaworld/jw-06-2003/jw-0620-protege.html.

[Kot04] K. Kotis, G. A. Vouros y J. Padilla Alonso. Hcome: tool-supported methodology for collaboratively devising living ontologies. En Christoph Bussler, Val Tannen y Irini Fundulaki, editores, *Semantic Web and Databases, Second International Workshop, SWDB 2004,Toronto, Canada*, tomo 3372, páginas 29–30. Springer-Verlag, 2004.

[Kot06] K. Kotis y A. Vouros. Human-centered ontology engineering: The hcome methodology. *Knowl. Inf. Syst.*, 10(1):109–131, 2006. ISSN 0219-1377.

[Kup06] A. Kupfer, S. Eckstein, K.Ñeumann y B. Mathiak. Handling changes of database schemas and corresponding ontologies. En

J. F. Roddick, V. R. Benjamins, S. S. Cherfi, R. H. L. Chiang, C. Claramunt, R. Elmasri, F. Grandi, H. Han, M. Hepp, M. D. Lytras, V. B. Misic, G. Poels, I. Song, J. Trujillo y C. Vangenot, editores, *ER (Workshops)*, tomo 4231 de *Lecture Notes in Computer Science*, páginas 227–236. Springer, 2006. ISBN 3-540-47703-9.

[Las02] O. Lassila y D. McGuinness. The role of frame-based representation on the semantic web. dsl-01-02. Informe técnico, Knowledge Systems Laboratory. Stanford University. Stanford. California, 2002.

[Lau04] H. Lausen y M. Stolberg. Semantic web portals - state of the art survey. Informe técnico, DERI, digital Enterprise Research Institute. Technical Report 2004-04-03, April 2004.

[Len95] D.B. Lennat. Cyc: a large-scale investment in knowledge infrastructure. *Communications of the ACM*, 33(8):33–38, 1995.

[Lub07] L. Lubyte y S. Tessaris. Extracting ontologies from relational databases.krdb research centre technical report krdb07-4. Informe técnico, Faculty of Computer Science, Free University of Bozen-Bolzano, Italy, 2007.

[Ma00] Z. Ma, W. J. M., Zhang y W. Y. Ma. Semantic measure of fuzzy data in extended possibility-based fuzzy relational databases. *International Journal of Intelligent System*, 15(8):705–716, 2000.

[Ma05] Z. Ma. *Fuzzy Database Modeling with XML*. Springer, 2005.

[Ma06] Z. Ma. *Fuzzy Database Modeling of Imprecise and Uncertain Engineering Information*. Springer, 2006.

[McC05] R. McCool. Rethinking the semantic web, part i. *IEEE Internet Computing*, páginas 86–88, Nov-Dec 2005.

[McC06] R. McCool. Rethinking the semantic web, part ii. *IEEE Internet Computing*, páginas 93–96, Jan-Feb 2006.

[Med94a] J. M. Medina. *Bases de Datos Relacionales Difusas: Modelo Teórico y Aspectos de su Implementación*. Tesis Doctoral, Departamento de Ciencias de la Computación e Inteligencia Artificial, E.T.S. de Ingeniería Informática, Universidad de Granada, España, 1994.

[Med94b] J. M. Medina, O. Pons y M. A. Vila. Gefred. a generalized model of fuzzy relational databases. *Information Sciences*, 76(1-2):87–109, 1994.

[Med95] J. M. Medina, M. A. Vila, J. C. Cubero y O. Pons. Towards the implementation of a generalized fuzzy relational database model. *Fuzzy Sets and Systems*, 75:273–289, 1995.

[Med97] J. M. Medina, O. Pons, J. C. Cubero y M. A. Vila. Freddi: A fuzzy relational deductive database interface. *International Journal of Intelligent Systems*, 12:597–613, 1997.

[Mee01a] R. Meersman. Ontologies and databases: More than a fleeting resemblance. *Information Technology and Management. Ed. Springer*, Rome Workshop , Luiss Publications(6):97–122, 2001.

[Mee01b] R. Meersman. Ontologies and databases: More than a fleeting resemblance. citeseer.ist.psu.edu/article/meersman01ontologies.html, 2001.

[Men01] E. Mena y A. Illarramendi. *Ontology-based query processing for global information systems*. Kluwer Academic Publishers, Norwell, MA, USA, 2001. ISBN 0-7923-7375-8.

[Miz95] R. Mizoguchi, J. Vanwelkenhuysen y M. Ikeda. Task ontology for reuse of problem solving knowledge. En *Mars N (ed) Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing (KBKS'95)*, páginas 46–57. University of Twente, Enschede, The Netherlands, IOS Press, 1995.

[Myl07] J. Mylopoulos. Ontologies. http://www.cs.toronto.edu/ jm/2507S/Notes04/Ontologies.pdf, 2007.

[Nec91] R.Ñeches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator y W. R. Swartout. Enabling technology for knowledge sharing. *AI Mag.*, 12(3):36–56, 1991. ISSN 0738-4602.

[Nic05] A. De Nicola, R.Ñavigli y M. Missikoff. *Innovation and Knowledge Economy: Issues, Applications, Case Studies*, capítulo Building an eProcurement Ontology with UPON methodology, páginas 177–184. IOS Press, 2005.

[Noy04] N. F. Noy. *Handbook on Ontologies*, capítulo Tools for Mapping and Merging Ontologies, páginas 366–384. Springer, 2004.

[Obe03] D. Oberle, S. Staab, R. Studer y R. Volz. Kaon server demonstrator. WonderWeb Deliverable D7, 2003. Http://wonderweb.semanticweb.org.

[Obe04] D. Oberle, R. Volz, B. Motik y S. Staab. An extensible ontology software environment. En Steffen Staab y Rudi Studer, editores, *Handbook on Ontologies*, International Handbooks on Information Systems, capítulo III, páginas 311–333. Springer, 2004.

[Ont07a] Ontoprise. Ontoedit datasheet 2003. http://electronic-office.de/pdf/ontoprise/ontoedit_data_sheet.pdf, 2007.

[Ont07b] Ontoprise. Ontostudio. http://www.ontoprise.de/content/e1171/e1249/index_eng.html, April 2007.

[Ora07] Oracle. Isqlplus web enviroment. http://150.214.108.124/isqlplus, January 2007.

[Org07] Open Cascade Organizacion. Dl-workbench. http://projects.opencascade.org/dl-workbench/, April 2007.

[otTUoMS07] Ontological Engineering Group (OEG) of the Technical University of Madrid (Spain). Webode ontology engineering platform. http://webode.dia.fi.upm.es/WebODEWeb/index.html, December 2007.

[Pan03] Z. Pan y J. Heflin. Dldb: Extending relational databases to support semantic web queries. En *Workshop on Practical and Scaleable Semantic Web Systms, ISWC 2003*, páginas 109–113. 2003.

[Par04] WP8 Partners. Deliverable d8.1. state of the art and state of the practice including initial possible research orientations. Informe técnico, Network of Excellence - Contract no.: IST-508 011, 2004.

[Par05] E. Pardede y J. Wenny Rahayu. Impact of new sql standard to database modeling. *Encyclopedia of Information Science and Technology. IDEA Publishing*, páginas 488–494, 2005.

[PdL05] C. Pérez de Laborda y S. Conrad. Relational.owl: a data and schema representation format based on owl. En *CRPIT '43: Proceedings of the 2nd Asia-Pacific conference on Conceptual*

modelling, páginas 89–96. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 2005. ISBN 1-920-68225-2.

[Pet96] F. E. Petry. *Fuzzy Databases: Principles and Applications*. International Series in Intelligent Technologies. Kluwer Academic Publishers, 1996.

[Pon96] O. Pons, J. M. Medina, J. C. Cubero y A. Vila. An architecture for a deductive fuzzy relational database. En Z.W. Ras y M. Michaliewicz, editores, *Foundations of Intelligent Systems*, tomo 1079 de *Lectures Notes in Artificial Intelligence*. Springer, 1996.

[Pon97] O. Pons, J. M. Medina, J. C. Cubero y M. A. Vila. *Flexible Query Answering Systems*, capítulo A fuzzy deductive relational database. Kluwer Academic Publishers, 1997.

[Pra84a] H. Prade. Lipski's approach to incomplete information databases restated and generalized in the setting of zadeh's possibility theory. *Information Sciences*, 9:27–42, 1984.

[Pra84b] H. Prade y C. Testemale. Generalizing database relational algebra for the treatment of incomplete/uncertain information and vague queries. *Information Sciences*, (34):113–143, 1984.

[Pra87a] H. Prade y C. Testemale. *Analysis of Fuzzy Information*, tomo 2, capítulo Representation of Soft Constraints and Fuzzy Attribute Values by means of Possibility Distributions in Databases. CRC Press, 1987.

[Pra87b] H. Prade y C. Testemale. Fuzzy relational databases: Representational issues and reduction using similaruty measures. *Journal of American Society for Information Sciences*, 38(2):118–126, 1987.

[Pro07] HP Labs Semantic Web Programme. Jena/ a semantic web framework for java. http://jena.sourceforge.net/, 2007.

[Raj88] K. V. S. V.Ñ. Raju y A. K. Majumdar. Fuzzy functional dependencies and lossless join decomposition of fuzzy relational database systems. *ACM Transactions on Database Systems*, 13(2):129–166, 1988.

[Rib06] R. Ribeiro, F. Batista, J. P. Pardal, N. J. Mamede y H. S. Pinto. Cooking an ontology. En Jérôme Euzenat y John Domingue, editores, *AIMSA*, tomo 4183 de *Lecture Notes in Computer Science*, páginas 213–221. Springer, 2006. ISBN 3-540-40930-0.

[Rol05] M.M. Roldán y J. F. Aldana Montes. A tool for storing owl using database technology. En *Bernardo Cuenca Grau, et al. (Eds.). Proceedings of the OWLED '05 Workshop on OWL: experiences and Directions, Galway, Ireland, November 11-12, 2005*, tomo 188, páginas 1–10. CEUR-Workshop Proceedings, septiembre 2005.

[Rui06] F. Ruiz y J. R. Hilera. *Ontologies for Software Engineering and Software Technology*, capítulo Using Ontologies in Software Engineering and Technology, páginas 49–102. Springer-Verlag, 2006.

[Run89] E. A. Rundensteiner, L. W. Hawkes y W. Bandler. On nearness measures in fuzzy relational data models. *International Journal Approximate Reasoning*, (3):267–298, 1989.

[Sch99] G. Schreiber y R. de Hoog. *Knowledge Engineering and Management: The CommonKADS Methodology*. MIT Press, 1999. ISBN 0-262-19300-0. 472 pages.

[Sha06a] N. Shadbolt, Berners T. Lee y W. Hall. The semantic web revisited. *Intelligent Systems, IEEE [see also IEEE Intelligent Systems and Their Applications]*, 21(3):96–101, 2006.

[Sha06b] R. Sharman, R. Kishore y R. Ramesh. *Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems (Integrated Series in Information Systems)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387370196.

[She89] S. Shenoi y A. Melton. Proximity relations in the fuzzy relational databases. *Fuzzy sets and Systems*, 31(3):285–296, 1989.

[She05] A. Sheth, C. Ramakrishnan y C. Thomas. Semantics for the semantic web: The implicit, the formal and the powerful. *Journal on Semantic Web and Information Systems*, 1(1):1–18, Jan-March 2005.

[Spy02] P. Spyns, R. Meersman y M. Jarrar. Data modelling versus ontology engineering. En *SIGMOD Record*, páginas 12–17. September 2002.

[Sta04] S. Staab y R. Studer. *Handbook on Ontologies*. Springer, 2004.

[Ste98] G. Steve, A. Gangemi y D. Pisanelli. Integrating medical terminologies with onions methodology. *http://saussure.irmkant.rm.cnr.it*, 1998.

[Sto02] L. Stojanovic, N. Stojanovic y R. Volz. Migrating data-intensive web sites into the semantic web. En *SAC '02: Proceedings of the 2002 ACM symposium on Applied computing*, páginas 1100–1107. ACM, New York, NY, USA, 2002. ISBN 1-58113-445-2.

[Stu98] R. Studer, VR. Benjamins y D. Fensel. Knowledge engineering: Principles and methods. *IEEE Transactions on Data and Knowledge Engineering*, 25(1-2):161–197, 1998.

[Su02] X. Su y L. Ilebrekke. A comparative study of ontology languages and tools. En *CAiSE 2002*, páginas 761–765. 2002.

[Sur04] Y. Sure, S. Staab y R. Studer. *Handbook on Ontologies*, capítulo On-To-Knowledge Methodology, páginas 117–132. Springer, 2004.

[Sur06] Y. Sure, C. Tempich y D. Vrandecic. *Semantic Web Technologies, trends and research in ontology-based systems*, capítulo Ontology Engineering Methodologies, páginas 171–190. Wiley, 2006.

[Swa96] B. Swartout, R. Patil, K. Knight y T. Russ. Toward distributed use of large-scale ontologies. En *the 10th Workshop on Knowledge Acquisition*. Banff, Canada, 1996.

[Tij05] Y. A. Tijerino, D. W. Embley, D. W. Lonsdale, Y. Ding y G.Ñagy. Towards ontology generation from tables. *World Wide Web*, 8(3):261–285, 2005. ISSN 1386-145X.

[Tri06] Q. Trinh, K. Barker y R. Alhajj. Rdb2ont: A tool for generating owl ontologies from relational database systems. En *AICT/ICIW*, página 170. 2006.

[Uma80] M. Umano, S. Fukami, M. Mizumoto y K. Tanaka. Retrieval processing from fuzzy databases. Informe técnico, IECE, Japón, 1980.

[Uma82a] M. Umano. Freedom-o: A fuzzy database system. En M. Gupta y E. Sanchez, editores, *Fuzzy Information and Decision Processes*, páginas 339–347. North-Holland, Amsterdam, Pub. Comp., 1982.

[Uma82b] M. Umano. *Fuzzy Information and Decision Processes*, capítulo FREEDOM-0: A Fuzzy Database System, páginas 339–347. North Holland Pub. Co., 1982.

[Uma94] 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.

[Unc04] M. Unchold y M. Gruninger. Ontologies and semantics for seamless connectivity. *SIGMOD Record*, 33(4):58–64, 2004.

[Uni07a] Open University. Webonto. http://kmi.open.ac.uk/projects/webonto/, 2007.

[Uni07b] Stanford University. Chimaera, April 2007.

[Uni07c] Stanford University. Chimaera. http://www.ksl.stanford.edu/software/chimaera/, April 2007.

[Uni07d] Stanford University. Ontolingua. http://www.ksl.stanford.edu/software/ontolingua/, April 2007.

[Upa05] S. R. Upadhyaya y P. S. Kumar. Eronto: a tool for extracting ontologies from extended e/r diagrams. En *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, páginas 666–670. ACM, New York, NY, USA, 2005. ISBN 1-58113-964-0.

[Usc95] M. Uschold. Towards a methodology for building ontologies. citeseer.ist.psu.edu/uschold95toward.html, 1995.

[Usc96] M. Uschold y M. Grüninger. Ontologies: principles, methods, and applications. *Knowledge Engineering Review*, 11(2):93–155, 1996.

[vH97] G. van Heijst, Ath. Schereiber y BJ. Wielinga. Using explicit ontologies in kbs development. *International Journal of Human-Computer Studies*, (45):193–292, 1997.

[Vys06] E. Vysniauskas y L.Ñemuraite. Transforming ontology representation from owl to relational database. *Information Technology and Control*, 35(3A):333–343, 2006.

[W3C99] W3C Recommendation, http://www.w3.org/RDF/. *Resource Description Framework (RDF)*, february 1999.

[w3c06] World wide web consortium. http://www.w3.org/, 2006.

[wg08] ESSI WSMO working group. Wsmo studio. http://www.wsmostudio.org/, January 2008.

[Wik07] Wikipedia. Looking for: Ontology. www.wikipedia.org, December 2007.

[Xu04] Z. Xu, X. Cao, Y. Dong y W. Su. Formal approach and automated tool for translating er schemata into owl ontologies. En *PAKDD*, páginas 464–475. 2004.

[Yab07] L. Yabloko y Next Generation Software. Ontobase plug-in for protégé. http://www.ontospace.net/pages/3/index.htm, April 2007.

[Yag78] R. R. Yager. Ranking fuzzy subsets over the unit interval. En *Proceedings of CDC*, páginas 1435–1437. 1978.

[Yag81] R. R. Yager. A procedure for ordering fuzzy subsets of the unit interval. *Information Sciences*, 24:143–161, 1981.

[You06] S. Youn y D. McLeod. Ontology development tools for ontology-based knowledge management. *IDea Group*, 2006.

[Zad65] L. A. Zadeh. Fuzzy sets. *Information and Control*, 83:338–353, 1965.

[Zad75] L. A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning. *Information Sci.*, 8:(8) 199–248, 301–357, (9) 43–80, 1975.

[Zem84] M. Zemankova y A. Kandel. *Fuzzy Relational Databases - A Key to Expert Systems*. Verlag TUV Rheinland, 1984.

[Zem85] M. Zemankova y A. Kandel. Implementing imprecision in information systems. *Information Sciences*, 37:107–141, 1985.

[Zha07] J. Zhang. Ontology and the semantic web. En *Proceedings of the North American Symposium on Knowledge Organization*, tomo 1. 2007.