

UNIVERSIDAD DE GRANADA

Departamento de Ciencias de la Computación  
e Inteligencia Artificial



**Mantenimiento Incremental  
de Reglas de Asociación y sus extensiones  
mediante Bases de Datos Activas**

MEMORIA QUE PRESENTA

**Alain Pérez Alonso**

PARA OPTAR AL GRADO DE DOCTOR EN INFORMÁTICA

Programa Oficial de Doctorado en Tecnologías  
de la Información y la Comunicación

DIRECTORES:

**Ignacio José Blanco Medina**

**José María Serrano Chica**

Editor: Universidad de Granada. Tesis Doctorales  
Autor: Alain Pérez Alonso  
ISBN: 978-84-9163-052-4  
URI: <http://hdl.handle.net/10481/44475>



El doctorando Alain Pérez Alonso y los directores de la tesis los doctores Dr. Ignacio José Blanco Medina y Dr. José María Serrano Chica garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la dirección de los directores de la tesis y hasta donde nuestro conocimiento alcanza, en la realización del trabajo, se han respetado los derechos de otros autores a ser citados, cuando se han utilizado sus resultados o publicaciones.

Granada, octubre de 2016

El Doctorando

Alain Pérez Alonso

El Director

El Director

Fdo. Ignacio José Blanco Medina

Fdo. José María Serrano Chica



# Agradecimientos

La presente memoria no hubiese sido posible sin la ayuda y el apoyo de varias personas e instituciones a las cual quiero agradecer.

En primer lugar, mencionar a Dios y a mi familia pues sin su apoyo hubiese sido imposible llegar a esta meta. Gracias mamá por tu gran amor y sacrificio, a ti te dedico este resultado. Gracias papá. Gracias esposa por tu amor, tus palabras de aliento y tu sostén. Gracias abuelo, abuela y tío por tanto amor. Gracias amigos. Gracias a todos los que estuvieron conmigo y me dieron su apoyo.

En un lugar especial quiero referirme a mis dos directores de tesis: Ignacio J. Blanco y José M. Serrano. Les agradezco a ambos brindarme el privilegio de trabajar juntos, la confianza depositada y el tiempo dedicado en las revisiones que tanto me enseñaron. Todo esto bajo la gran distancia física, convertida en casi nula por la atención constante y la guía oportuna. También quiero agradecer a mi tutora Luisa M. González quien me ha encaminado y ayudado en el ámbito investigativo, a Antonio Cañas por su amistad y confianza, a Verdegay por su atención y empeño, a Amparo Vila por su acogida y al grupo de investigación de Base de Datos y colegas de la UCLV. A las familias granadinas que me brindaron su amistad y me abrieron sus hogares, muchas gracias.

Me siento profundamente agradecido por todos aquellos maestros y profesores que me formaron desde mi niñez y a todos los que ayudaron directa o indirectamente en la realización de esta investigación y no he mencionado.

Agradezco a la Asociación Universitaria Iberoamericana de Postgrado (AUIP) por auspiciar el programa de doctorado, así como la posibilidad que me brindó de realizar estancias de investigación en la Universidad de Granada, en los años 2013, 2014 y 2016.

¡¡¡ ETERNAMENTE AGRADECIDO A TODOS !!!

*“... y el árbol de la ciencia del bien y del mal.”*

*- Gén 2:9*

*“Porque mejor es la sabiduría que las piedras  
preciosas; Y todo cuanto se pueda desear,  
no es de compararse con ella.”*

*- Prov 8:11*

*“Y si tuviese profecía, y entendiase todos  
los misterios y toda ciencia, y si tuviese toda  
la fe, de tal manera que trasladase los montes,  
y no tengo amor, nada soy.”*

*- 1Cro 13:2*

# Resumen

El desarrollo sin precedentes en las tecnologías de la información ha provocado que el volumen de datos existente en los repositorios de todo el mundo haya alcanzado niveles extraordinarios y un ritmo de crecimiento vertiginoso. Esto supone una fuente de riqueza que es necesario comprender y convertir en información valiosa. En este sentido, juega un importante rol la extracción de conocimiento en bases de datos donde la minería de reglas de asociación es uno de los métodos para abordar el proceso no trivial de identificación de patrones en los datos.

A partir de las reglas de asociación se pueden identificar otras representaciones del conocimiento que son semánticamente significativas para el usuario. Una de ellas es la imprecisión o incertidumbre que puede acompañar a la información mediante el diseño de las reglas de asociación difusas. Otra son las dependencias aproximadas, las cuales pueden ser vistas como excepciones a reglas que se cuantifican mediante las reglas de asociación.

Independientemente del tipo de conocimiento extraído, su valor es relativo al momento en que se ejecutó el algoritmo. Sin embargo, los datos por naturaleza se encuentran en constante cambio. Este hecho invariablemente conduce a la modificación del conocimiento previamente extraído, convirtiéndolo de esta forma en inexacto y eventualmente, obsoleto.



La presente investigación se centra en desarrollar nuevos métodos para mantener incrementalmente, ante los cambios ocurridos en los datos, las reglas de asociación, reglas de asociación difusas y dependencias aproximadas previamente extraídas. En este sentido se tuvieron en cuenta los recursos activos ofrecidos por las bases de datos. Específicamente hemos marcado los siguientes objetivos:

- Realizar un estudio del estado del arte en el descubrimiento y mantenimiento de las reglas de asociación, reglas de asociación difusas y dependencias aproximadas.
- Analizar los vínculos existentes entre el mantenimiento incremental en bases de datos activas y el mantenimiento incremental de las reglas abordadas.
- Proponer nuevos algoritmos para el mantenimiento incremental de reglas de asociación, reglas de asociación difusas y dependencias aproximadas que integren distintas medidas de interés.
- Implementar una herramienta de software para el mantenimiento incremental de reglas en bases de datos activas.

Como primera contribución de la presente memoria se obtuvieron dos propuestas para el mantenimiento incremental de reglas previamente descubiertas, mediante recursos activos de las bases de datos. Una de ellas está enfocada en el mantenimiento inmediato de las reglas, donde inmediatamente después de ocurrido el cambio en el dato, se actualiza el conocimiento. La otra se enfoca en el mantenimiento diferido de las reglas, donde se actualiza el conocimiento luego de varias modificaciones ocurridas en los datos. Ambos algoritmos mantienen directamente las medidas mediante un conjunto de partes diferentes. Esto posibilita mantener eficientemente varias medidas de forma simultánea, lo cual es un aspecto favorable ante la gran variedad de métricas existentes. Las propuestas realizadas fueron evaluadas mediante un estudio experimental. Este estudio realiza un análisis del desempeño y la escalabilidad de los algoritmos utilizando diferentes conjuntos de datos de entornos reales.

Como segunda contribución se creó una herramienta para el mantenimiento incremental de reglas mediante bases de datos activas (DRIMS). Esta herramienta escrita en Java es capaz de gestionar los tipos de reglas estudiados mediante una base de reglas, brindando para ello la opción de crear nuevas reglas a través de asistentes. Las reglas existentes en la base de reglas pueden ser mantenidas incrementalmente en sistemas reales. DRIMS implementa los algoritmos inmediato y diferido en dos de los sistemas gestores de bases de datos bajo código abierto más utilizados: PostgreSQL y MySQL.

Con la investigación realizada en esta memoria se desarrollaron y evaluaron nuevos algoritmos para el mantenimiento incremental de reglas de asociación, reglas de asociación difusas y dependencias aproximadas. Estos algoritmos mostraron buenos resultados en su desempeño y superaron técnicas convencionales e incrementales para el mantenimiento de reglas previamente descubiertas. La implementación de ambas propuestas puede ser realizada mediante DRIMS, herramienta que permitirá llevar a la práctica los resultados teóricos obtenidos y aplicar las propuestas de mantenimiento incremental en bases de datos activas reales.



# Índice general

<b>Índice de Figuras</b>	<b>XV</b>
<b>Índice de Tablas</b>	<b>XIX</b>
<b>Índice de Términos</b>	<b>XXI</b>
<b>1 Introducción</b>	<b>1</b>
1.1 Planteamiento . . . . .	1
1.2 Objetivos científicos . . . . .	5
1.3 Estructura de la memoria . . . . .	6
<b>2 Preliminares</b>	<b>9</b>
2.1 Descubrimiento de Reglas de Asociación . . . . .	9
2.1.1 Descripción del problema . . . . .	10
2.1.2 Medidas de Interés . . . . .	12
2.1.3 Algoritmos clásicos de minería . . . . .	15
2.2 Reglas de Asociación Difusas . . . . .	18
2.3 Dependencias Aproximadas mediante Reglas de Asociación . . . . .	23
2.4 Minería incremental de Reglas de Asociación . . . . .	26
2.5 Fundamentos de Bases de Datos Activas . . . . .	30
2.6 Resumen . . . . .	35

---

<b>3</b>	<b>Modelado de reglas en Bases de Datos Activas</b>	<b>37</b>
3.1	Mantenimiento de reglas mediante Vistas Materializadas . . . . .	38
3.2	Chequeo de Restricciones de Integridad para las reglas abordadas .	42
3.3	Expresividad de las Reglas de Negocio . . . . .	47
3.4	Recursos activos en los sistemas relacionales . . . . .	49
3.4.1	Restricciones de Chequeo . . . . .	50
3.4.2	Aserciones . . . . .	51
3.4.3	Disparadores . . . . .	52
3.5	Resumen . . . . .	54
<b>4</b>	<b>Nuevos algoritmos para el mantenimiento incremental de reglas</b>	<b>57</b>
4.1	Propuesta de mantenimiento incremental directo . . . . .	58
4.2	Mantenimiento incremental de las medidas . . . . .	61
4.3	Enfoque Simplificado para el mantenimiento de las reglas . . . . .	65
4.4	Algoritmo para el mantenimiento incremental inmediato de las reglas	68
4.5	Algoritmo para el mantenimiento incremental diferido de las reglas	73
4.6	Análisis de complejidad temporal de los algoritmos propuestos . .	79
4.7	Estudio experimental . . . . .	81
4.7.1	Diseño de la experimentación . . . . .	82
4.7.2	Conjuntos de datos . . . . .	84
4.7.3	Análisis de desempeño . . . . .	86
4.7.4	Análisis de escalabilidad . . . . .	91
4.8	Resumen . . . . .	95
<b>5</b>	<b>Herramienta para el mantenimiento incremental de reglas: DRIMS</b>	<b>97</b>
5.1	Arquitectura del sistema . . . . .	98
5.1.1	Visión estática . . . . .	98
5.1.2	Visión dinámica . . . . .	103
5.2	Casos de estudio en la interfaz de usuario . . . . .	104

5.2.1	Barra de menús . . . . .	105
5.2.2	Caso de estudio del asistente para nuevas reglas . . . . .	106
5.2.3	Caso de estudio del asistente para implementar reglas . . . . .	109
5.3	Resumen . . . . .	111
<b>6</b>	<b>Conclusiones y trabajos futuros</b>	<b>113</b>
6.1	Conclusiones . . . . .	113
6.2	Publicaciones asociadas a la memoria . . . . .	116
6.3	Trabajos futuros . . . . .	118
<b>A</b>	<b>Sintaxis detallada de los disparadores en el estándar SQL:2011</b>	<b>119</b>
<b>B</b>	<b>Reglas extraídas del SWAD y utilizadas en los experimentos</b>	<b>123</b>
<b>C</b>	<b>Sintaxis de la gramática utilizada para las reglas en DRIMS</b>	<b>125</b>
	<b>Bibliografía</b>	<b>129</b>



# Índice de figuras

2.1	Cardinalidades de la regla de asociación $X \Rightarrow Y$ . . . . .	14
2.2	FP-tree construido a partir de la relación mostrada en la Tabla 2.2. . . . .	18
2.3	Etiquetas lingüísticas definidas sobre el atributo Edad (arriba) y el atributo Salario (abajo). . . . .	21
2.4	Método de re-ejecución “desde cero” de los algoritmos convencionales de minería. . . . .	28
2.5	Método de minería incremental para reglas de asociación. . . . .	28
2.6	Transición de la base de datos <i>BD</i> luego de ocurrir un evento estructural primitivo de inserción, actualización y eliminación. . . . .	34
3.1	Relación entre reglas de negocio, reglas de negocio en bases de datos y restricciones de integridad. . . . .	47
4.1	Métodos de minería convencional a) y minería incremental b) para la obtención de medidas exactas en tiempo real. . . . .	60
4.2	Propuesta incremental para la obtención de medidas exactas en tiempo real. . . . .	60
4.3	Ejemplos de las estructuras <i>MRA</i> , <i>MRAD</i> y <i>MDA</i> para el mantenimiento de la métrica <i>information gain</i> . . . . .	67



4.4	Ejemplo de mantenimiento incremental inmediato para una RA . . .	70
4.5	Ejemplo de mantenimiento incremental inmediato para una RAD .	71
4.6	Ejemplo de mantenimiento incremental inmediato para una DA . .	72
4.7	Manejo de los EEP en la propuesta inmediata a) y diferida b). . . .	74
4.8	Ejemplo del cómputo de instancias relevantes para una transición.	76
4.9	Ejemplo de actualización diferida de MRA para las instancias relevantes. . . . .	78
4.10	Diseño del análisis realizado en los algoritmos propuestos. . . . .	79
4.11	Representación de las cotas asintóticas superiores $O(n \log(n))$ , $O(n)$ y $O(n \log(n))$ . . . . .	81
4.12	Desempeño de las propuestas en el mantenimiento de RA. Tiempo ejecución (arriba) y tiempo de actualización de las medidas (abajo) en PostgreSQL. . . . .	87
4.13	Desempeño de las propuestas en el mantenimiento de RAD. Tiempo ejecución (arriba) y tiempo de actualización de las medidas (abajo) en PostgreSQL. . . . .	88
4.14	Desempeño de las propuestas en el mantenimiento de DA. Tiempo ejecución (arriba) y tiempo de actualización de las medidas (abajo) en MySQL. . . . .	89
4.15	Escalabilidad de las propuestas en el mantenimiento de DA para MySQL. . . . .	92
4.16	Escalabilidad de las propuestas en el mantenimiento de RA; tiempos de ejecución de la minería convencional e incremental. . .	93
4.17	Escalabilidad de las propuestas en el mantenimiento de RAD; tiempos de ejecución de la minería convencional e incremental. . .	94
4.18	Comparación de las propuestas con la minería convencional e incremental en el mantenimiento de RA para diferentes conjuntos de datos. . . . .	94

---

4.19 Comparación de las propuestas con la minería convencional e incremental en el mantenimiento de RAD para diferentes conjuntos de datos. . . . .	95
5.1 Diagrama de componentes de la herramienta. . . . .	99
5.2 Estructura de la base de reglas utilizada por DRIMS. . . . .	100
5.3 Diagrama de clases de la herramienta. . . . .	101
5.4 Diagrama de casos de uso de la herramienta. . . . .	103
5.5 Diagrama de estados de la base de reglas en el sistema. . . . .	104
5.6 Ventana principal de DRIMS. . . . .	105
5.7 Inicio del asistente para crear reglas. . . . .	107
5.8 Elección de la relación base para nuevas reglas. . . . .	108
5.9 Fin del asistente para crear reglas. . . . .	109
5.10 Implementación de las propuestas inmediata y diferida en DRIMS. . . . .	110
5.11 <i>Script</i> SQL que implementa las propuestas inmediata y diferida en el sistema. . . . .	110



# Índice de tablas

2.1	Lista de medidas de interés para reglas de asociación . . . . .	14
2.2	Ejemplo de una relación de transacciones con los ítems frecuentes ordenados . . . . .	17
2.3	Edad y salario de cinco personas ( $r_0: \mathbf{R}$ ) . . . . .	21
2.4	Transacciones difusas para la relación de la Tabla 2.3 . . . . .	22
2.5	Datos de tres estudiantes en la relación $r$ . . . . .	25
2.6	El conjunto de transacciones $T_r$ de la relación $r$ . . . . .	25
3.1	Datos de productos vendidos en un supermercado ( <i>Registro_Ventas</i> )	41
3.2	Relación virtual derivada de ejecución de la vista <i>total_ventas_año</i> en la relación de la Tabla 3.1, junto a cantidad de tuplas derivadas	41
3.3	Taxonomía de restricciones de integridad en el modelo relacional .	43
3.4	Sumario de referencias a los alias según el evento desencadenador	53
4.1	Definición de las partes necesarias para el mantenimiento de varias medidas . . . . .	63
4.2	Complejidad temporal de los algoritmos propuestos . . . . .	80



# Índice de Términos

- BD**      **Base de Datos**, es una colección de datos relacionados.
- BDA**      **Base de Datos Activa**, es toda **BD** que provee mecanismos para habilitar respuestas automáticas a eventos ocurridos dentro o fuera de la propia **BD**.
- DA**      **Dependencia Aproximada**, puede entenderse como una dependencia funcional con la posibilidad de excepciones.
- ECA**      **Evento Condición Acción**, son reglas definidas por estos tres componentes que describen el comportamiento reactivo de las **BDA**.
- EEC**      **Evento Estructural Compuesto**, son aquellos eventos que involucran múltiples **EEP** o múltiples **EEC**.
- EEP**      **Evento Estructural Primitivo**, en el contexto de la presente memoria es un evento que involucra la inserción, actualización o eliminación de una entidad.
- MDA**      **Medida de una Dependencia Aproximada**, estructura propuesta para mantener incrementalmente las medidas de una **DA**.
- MR**      **Medida de una Regla**, estructura general que bien puede ser una **MRA**, una **MRAD** o una **MDA**.

- MRA** **Medida de una Regla de Asociación**, estructura propuesta para mantener incrementalmente las medidas de una **RA**.
- MRAD** **Medida de una Regla de Asociación Difusa**, estructura propuesta para mantener incrementalmente las medidas de una **RAD**.
- RA** **Regla de Asociación**, regla que expresa una correlación o asociación interesante entre conjuntos de elementos en un repositorio de datos.
- RAD** **Regla de Asociación Difusa**, es una **RA** extendida en la teoría de conjuntos difusos.
- RI** **Restricción de Integridad**, condición que una **BD** debe cumplir en todo momento para garantizar su integridad o consistencia.
- RN** **Reglas de Negocio**, son sentencias que definen o restringen algún aspecto del negocio y que se encuentran bajo su jurisdicción.
- SGBDRA** **Sistema Gestor de Bases de Datos Relacionales Activas**, es un sistema que permite la gestión de **BDA** basadas en el modelo relacional.
- VM** **Vista Materializada**, es el almacenamiento en la **BD** de las tuplas obtenidas por la expresión de la vista.

# Introducción

## 1.1 Planteamiento

En la era actual, debido al desarrollo sin precedentes en las tecnologías de la información, el volumen de datos existente en los repositorios de todo el mundo ha alcanzado niveles extraordinarios y un ritmo de crecimiento vertiginoso [127]. Estos grandes repositorios incluyen tipos de datos complejos y heterogéneos como los estructurados, no estructurados, semiestructurados y mixtos [7]. Por décadas, los datos estructurados se han organizado con el modelo relacional de Bases de Datos (BD) presentado por Codd en el año 1970 [33]. Las bases de datos relacionales posiblemente sean el modelo de BD más extendido por su popularidad e inherente simplicidad, situándolas como uno de los orígenes de datos más importantes y utilizados.

Los grandes volúmenes de datos suponen una fuente de riqueza en bruto que es necesario comprender y convertir en información valiosa. En este sentido, juega un importante rol la extracción de conocimiento en bases de datos [48], conocida por su término anglosajón *Knowledge Discovery in Databases* (KDD). Los pasos del proceso KDD incluyen la selección, el pre-procesamiento, la minería de datos y la evaluación o interpretación. Dentro de ellos, la minería de datos es el paso más significativo y el que mayor cantidad de recursos exige. Este área de investigación



engloba múltiples métodos para abordar el proceso no trivial de identificación de patrones en los datos, hasta el momento desconocidos y potencialmente útiles. Algunos de estos métodos incluyen la asociación, clasificación y agrupamiento.

La minería de Reglas de Asociación (RA) es el descubrimiento de asociaciones o correlaciones entre elementos de un conjunto de datos y fue introducida por Agrawal y colaboradores en el año 1993 [1]. Actualmente, se ha convertido en una técnica ampliamente utilizada en la minería dada su descripción natural y la fácil comprensión del conocimiento que representa. Las reglas de asociación, en su definición, están sujetas a medidas que indican el grado de interés que reflejan. Existe una cantidad abrumadora de estas medidas [75, 71] puesto que ninguna puede considerarse la mejor para todas las situaciones. En su inicio, la minería de RA fue propuesta para extraer asociaciones entre los productos de las canastas compradas en un mercado. Sin embargo, ha probado ser útil en disímiles dominios de aplicación como sistemas de recomendación, distribución de mercancías, detección de intrusos, diseño de catálogos, minería de la Web, entre otros. Muchos son los algoritmos desarrollados para la extracción de RA en la actualidad [13, 88, 86, 134] desde que se propusiera Apriori [2] como el algoritmo pionero.

A partir de las reglas de asociación se pueden identificar otras representaciones del conocimiento que son semánticamente significativas para el usuario. Una de ellas es la imprecisión o incertidumbre que puede acompañar a la información. La teoría de subconjuntos difusos propuesta por Zadeh [129], es una de las técnicas más utilizadas para representar posibles tipos de vaguedad, permitiendo el diseño de las Reglas de Asociación Difusas (RAD) [39].

Otra forma de representación del conocimiento que puede ser abordada mediante las RA son las Dependencias Aproximadas (DA). Estas dependencias pueden ser vistas como excepciones a reglas que se cuantifican mediante las reglas de asociación [112].

Independientemente del tipo de conocimiento extraído a través de los algoritmos de minería de datos, su valor es relativo al momento en que se ejecutó dicho algoritmo en la BD [29]. Estas bases de datos, por naturaleza, se encuentran en constante cambio mediante el procesamiento de nuevas transacciones. Este hecho invariablemente conduce a la modificación del conocimiento previamente extraído, convirtiéndolo de esta forma en inexacto y eventualmente, obsoleto [77]. Dicho proceso puede verse acentuado en las emergentes aplicaciones de flujos de datos (*data streams*) donde usualmente los datos arriban de forma continua, con gran velocidad y volumen [85]. Tal es el caso del análisis de redes sensoriales, tráfico de red, clics de la web y registro de llamadas telefónicas, entre otras.

Una solución elemental para actualizar el conocimiento previamente descubierto consiste en volver a ejecutar los algoritmos de minería “desde cero” [46]. Obviamente, esta solución es muy costosa e ineficiente, pues desecha el procesamiento realizado anteriormente y los recursos utilizados. Este área de la minería en bases de datos dinámicas ha sido investigada bajo el término de minería incremental [93]. Desde que Cheung y colaboradores propusiesen el algoritmo FUP [29] en el año 1996, numerosas técnicas han sido desarrolladas [30, 116, 93, 9, 133, 124]. De forma general, todas ellas reutilizan el conocimiento previamente extraído, analizan las nuevas transacciones que se suceden y buscan disminuir la exploración completa de la BD. Este último aspecto es uno de los más delicados en todas las propuestas, dado que involucra considerables recursos de tiempo y memoria. En la era actual, los volúmenes de datos a explorar alcanzan frecuentemente órdenes de terabytes o incluso petabytes. Estos datos se encuentran almacenados en BD realmente grandes (*very large databases*) [29] que pueden involucrar datos masivos (*Big Data*) [127].

Sin embargo, aun cuando la minería incremental reutiliza el procesamiento realizado, consideramos este método ineficiente cuando se trata únicamente de mantener el conocimiento ya extraído [104]. En general, aquellas aplicaciones que se basan en el apoyo a la toma de decisiones en tiempo real (*real-time decision support systems*) [113, 42] pueden verse afectadas. Es decir, esta ineficiencia es un problema para la latencia necesaria en la toma de decisiones bajo entornos de cambios rápidos o de grandes volúmenes de datos. Existen además otras áreas

de investigación que también toman en cuenta los cambios del conocimiento. Un ejemplo es la etapa conocida como *post mining* en el proceso de KDD, donde se analizan los cambios en las reglas descubiertas a través del tiempo [15]. Otro ejemplo son los recientes retos de velocidad y volumen en *Big Data* [127].

Desde otro punto de vista, las **BD** que contienen este gran volumen de datos no deben considerarse inactivas. Tradicionalmente, los sistemas de bases de datos han sido vistos como repositorios pasivos, donde simplemente se guarda la información requerida por las aplicaciones [102]; siendo activadas únicamente por comandos u operaciones de manipulación sobre los datos. Sin embargo, las bases de datos poseen mayores facilidades para el modelado de aspectos relacionados con el comportamiento. Dentro de estos tipos de bases de datos se encuentran las Bases de Datos Activas (**BDA**) [92], término otorgado en años tan tempranos como 1983. Las **BDA** unifican la teoría tradicional de las tecnologías de bases de datos con la programación basada en reglas, a fin de expresar capacidades reactivas. Esto las habilita para monitorear y reaccionar a circunstancias específicas. Una propuesta común para el modelado del conocimiento utiliza reglas que tienen tres componentes: un evento, una condición y una acción. Estas reglas son conocidas como reglas de **Evento Condición Acción (ECA)** [103]. Las Vistas Materializadas (**VM**) [90], Restricciones de Integridad (**RI**) [22] y **Reglas de Negocio (RN)** [25] son algunas de las nuevas funcionalidades que proveen las **BDA**. Las **VM** se enfrentan a la propagación de cambios realizados sobre aquellas relaciones que intervienen en la definición de la vista. Por otro lado, las **RI** son condiciones sobre los datos que deben ser válidas en todo momento. Ellas se encuentran incluidas dentro de las **RN**, las cuales permiten, entre otras características, definir acciones más suaves.

Estas áreas de investigación en **BDA** aportan una gran variedad de técnicas y algoritmos para el mantenimiento incremental de sus objetivos. Estos algoritmos se encuentran especialmente diseñados, bajo varias décadas de estudio, para mantener altas prestaciones en entornos con gran flujo de datos y enormes volúmenes de datos. Dichas características originan la idea de analizar estas técnicas y algoritmos para proponer nuevos algoritmos que mantengan incrementalmente, desde las propias **BDA**, el conocimiento previamente extraído [104].

## 1.2 **Objetivos científicos**

El objetivo principal de esta memoria es desarrollar nuevos métodos para el mantenimiento incremental de reglas de asociación, reglas de asociación difusas y dependencias aproximadas mediante recursos de bases de datos activas. Para ello hemos marcado específicamente los siguientes objetivos:

- **Realizar un estudio del estado del arte en el descubrimiento y mantenimiento de las RA, RAD y DA.** El primer objetivo planteado es el de estudiar las técnicas y algoritmos más representativos en el área de extracción de RA, RAD y DA. En este estudio incluimos aquellas técnicas y algoritmos que realicen una extracción incremental de las reglas en aras de mantener el conocimiento previamente extraído.
- **Analizar los vínculos existentes entre el mantenimiento incremental en BDA y el mantenimiento incremental de las reglas abordadas.** Con este segundo objetivo se plantea, en el área del mantenimiento incremental, vincular las BDA y las reglas abordadas. El ánimo de este objetivo es analizar las propuestas y recursos existentes en las BDA que apoyen nuestro objetivo principal. Dentro del mantenimiento incremental en BDA se analizan principalmente los algoritmos de mantenimiento de vistas materializadas y chequeo de restricciones de integridad.
- **Proponer nuevos algoritmos para el mantenimiento incremental de RA, RAD y DA que integren distintas medidas de interés.** Como un tercer objetivo nos planteamos, mediante los resultados previos, proponer algoritmos novedosos para el mantenimiento incremental de las reglas estudiadas. Estos algoritmos deben ser diseñados considerando la gran variedad de métricas existentes siendo, consecuentemente, eficientes e inclusivos.

- **Implementar una herramienta de software para el mantenimiento incremental de reglas en BDA.** Por último, para que estos algoritmos puedan ser estudiados y aplicados, es necesario crear una herramienta de software sencilla y amigable. Esta herramienta permitirá llevar a la práctica los resultados teóricos obtenidos y aplicar las propuestas de mantenimiento incremental en BDA reales.

### 1.3 Estructura de la memoria

La presente memoria está compuesta de seis capítulos y tres apéndices. La estructura de cada una de estas partes se introduce brevemente a continuación.

- **Capítulo 1.** Introduce el planteamiento, los objetivos a cumplir y la estructura de la presente memoria.
- **Capítulo 2.** Aborda una breve, pero comprensiva, revisión del descubrimiento de reglas de asociación. Esto incluye definiciones básicas, medidas de interés y algoritmos clásicos de minería de datos. Posteriormente, muestra dos extensiones de las reglas de asociación: las reglas de asociación difusas y las dependencias aproximadas. Seguidamente, presenta otra técnica en la extracción del conocimiento que se relaciona estrechamente con esta tesis: la minería incremental. Concluye con los fundamentos de las BDA que se utilizan en la memoria.
- **Capítulo 3.** Muestra tres áreas de investigación existentes en las BDA. Cada una de estas áreas es vinculada con las reglas de asociación y sus extensiones, presentando, al mismo tiempo, sus técnicas y algoritmos más relevantes. Posteriormente, provee varios recursos de las BDA que posibilitan la especificación de comportamientos reactivos.

- **Capítulo 4.** Propone los nuevos algoritmos de mantenimiento incremental. Para ello, formaliza inicialmente el problema de investigación y sus diferencias con las técnicas de minería convencional e incremental. Luego, establece cómo se utilizan las medidas de las reglas en la propuesta realizada. A continuación, procede a mostrar una solución simple de nuestro problema y los dos algoritmos de mantenimiento incremental elaborados en la presente memoria: uno inmediato y otro diferido. Seguidamente, realiza un estudio experimental de los algoritmos propuestos. Para ello detalla el diseño de los experimentos realizados, los conjuntos de datos utilizados y los resultados de los análisis en diferentes entornos.
- **Capítulo 5.** Implementa los resultados teóricos mostrados en la memoria mediante una herramienta de software. Inicialmente presenta el análisis y diseño de la herramienta mediante su arquitectura. Este diseño se divide en una visión estática y dinámica del sistema para una mejor comprensión del mismo. Posteriormente, describe la interfaz del usuario. En la misma se indican las ventanas principales de la herramienta mediante su barra de menús y sus asistentes para crear e implementar las reglas.
- **Capítulo 6.** Presenta las conclusiones, un listado de las publicaciones asociadas a la tesis y las líneas de trabajo futuras que quedan abiertas tras el estudio realizado.
- **Apéndice A.** Detalla la sintaxis de los disparadores en el estándar SQL.
- **Apéndice B.** Presenta las reglas de asociación, reglas de asociación difusas y dependencias aproximadas extraídas del SWAD y utilizadas en los experimentos.
- **Apéndice C.** Detalla la sintaxis de la gramática utilizada para las reglas en la herramienta desarrollada.



# Capítulo 2

## Preliminares

El propósito de este capítulo es presentar los tópicos, áreas y trabajos relacionados con la presente investigación. Primeramente, se incluye una revisión en el descubrimiento de reglas de asociación. Para ello, además de introducir algunas definiciones básicas, se estudian otras medidas de interés y se exponen los algoritmos convencionales de minería. Posteriormente, se muestran dos formas del conocimiento como extensiones de las reglas de asociación: las reglas de asociación difusas y las dependencias aproximadas. A continuación, se proporciona una ojeada a otra técnica de extracción del conocimiento: la minería incremental. Finalmente, se concluye con los fundamentos de las bases de datos activas que son utilizados en el resto de la memoria.

### 2.1 Descubrimiento de Reglas de Asociación

En la era actual, el ser humano ha visto crecer rápidamente su capacidad de generar y coleccionar datos [127]. La utilización generalizada de sistemas informáticos en las actividades cotidianas, junto al desarrollo de herramientas para el almacenamiento, han suministrado y suministran enormes volúmenes de datos. Estos datos presentan gran diversidad en sus formatos, teniéndose los tipos estructurados, no estructurados, semiestructurados y mixtos [7]. Entre ellos, los



datos estructurados han sido objeto de atención desde fechas tan tempranas como los años 70, mediante el modelo relacional de Bases de Datos (BD) presentado por Codd [33]. Estas BD continúan representando hoy día uno de los orígenes de datos más utilizados en las técnicas actuales, debido a su fuerte cimiento matemático y a su inherente simplicidad.

El análisis de los datos constituye una tarea de creciente importancia. Este es el objetivo que distingue actualmente al proceso de *Knowledge Discovery in Databases* (KDD). Mediante su aplicación es posible descubrir conocimiento útil dentro de grandes colecciones, aumentando considerablemente el valor de los datos. En la actualidad es ampliamente utilizado para la detección de fraudes, telecomunicaciones, marketing, finanzas, ventas, y en general aquellos campos relacionados con la toma de decisiones. Dentro de los pasos que conforman el proceso KDD se distingue la minería de datos como el paso principal. La minería de datos engloba múltiples técnicas para el descubrimiento de estructuras, patrones y modelos en los datos. Dentro de estas técnicas destaca la minería de reglas de asociación, y su tarea de inferir reglas que establecen asociaciones o correlaciones entre elementos de un conjunto de datos.

### 2.1.1 Descripción del problema

El origen de las Reglas de Asociación (RA) se encuentra en el contexto del análisis de las canastas de compra en un mercado, donde se estudia la compra conjunta de algunos artículos. Estas asociaciones entre artículos comprados por los clientes pueden ayudar, por ejemplo, a establecer nuevas estrategias de marketing. Es por ello que la definición formal de las RA se establece en el contexto de conjuntos de elementos (ítemsets).

Sea  $It = \{It_1, It_2, \dots, It_m\}$  un conjunto no vacío de  $m$  atributos o ítems diferentes. Sea  $T$  el esquema de una base de datos de transacciones que contiene un conjunto de elementos tal que  $It \subseteq T$ . Una regla de asociación es una implicación en la forma  $X \Rightarrow Y$  donde  $X, Y \subset It$  tal que  $X \neq \emptyset$ ,  $Y \neq \emptyset$  y  $X \cap Y = \emptyset$ . En esta sentencia  $X$  y  $Y$  son ítemsets conocidos como el antecedente y el consecuente de la regla, respectivamente.

Para evaluar el cumplimiento de las RA existen dos métricas clásicas: el soporte y la confianza. Estas medidas expresan el grado de interés o el grado de cumplimiento que pueden tener las reglas desde un punto de vista estadístico. Ambas se basan en el soporte de un conjunto de ítems  $I$ , el cual es simplemente la fracción de registros que contienen  $I$  en todos los registros como expresa (2.1).

$$sop(I) = \frac{|\{t \in T \mid I \subseteq t\}|}{|T|} \quad (2.1)$$

Luego, el soporte y la confianza de una regla de asociación  $X \Rightarrow Y$  se definen como expresan (2.2) y (2.3) respectivamente.

$$Sop(X \Rightarrow Y) = Sop(XY) = \frac{|\{t \in T \mid X \cup Y \subseteq t\}|}{|T|} \quad (2.2)$$

$$Conf(X \Rightarrow Y) = \frac{Sop(XY)}{Sop(X)} = \frac{|\{t \in T \mid X \cup Y \subseteq t\}|}{|\{t \in T \mid X \subseteq t\}|} \quad (2.3)$$

Considere, a modo de ejemplo de una RA junto a su soporte y confianza, la siguiente afirmación en el contexto de un restaurante:

El 85 % de los comensales que ordenan pescado, lo acompañan con vino blanco. (soporte=15 %)

Para este ejemplo, las transacciones son las órdenes realizadas por los comensales y los ítems serían los alimentos y bebidas ofrecidos por el restaurante,  $X=\{pescado\}$ ,  $Y=\{vino\ blanco\}$ . La RA planteada es  $pescado \Rightarrow vino\ blanco$  y establece que existe una confianza de 0.85 a que se ordene vino blanco, dado que se ha pedido pescado. La fracción de las cantidades de órdenes que contienen pescado y vino blanco, dentro de todas las órdenes realizadas por los comensales en el restaurante, es de 0.15. Tanto el soporte como la confianza toman valores, según (2.1) y (2.2), en el intervalo  $[0,1]$  aunque usualmente también se expresan en porcentajes.

El uso del soporte, además de medir la relevancia estadística de una regla, es una medida con un significado intuitivo lo que constituye un aspecto importante para la interpretación sencilla del usuario. También, como medida, contribuye al diseño de algoritmos lo cual lo convierte en una métrica ampliamente aceptada y generalizada.

A diferencia del soporte, la confianza ha recibido diversas críticas [115, 12, 112]. Esta métrica no establece adecuadamente el grado de independencia estadística, ni refleja las dependencias negativas entre el antecedente y el consecuente. Asimismo, no es intuitiva de entender por el usuario que usualmente debe definir su umbral.

### 2.1.2 Medidas de Interés

Con el ánimo de superar estas deficiencias, mejorar la calidad de las reglas y proveer al usuario de información más interesante, muchas métricas diferentes han sido propuestas [71, 122, 75, 50, 121, 49]. De hecho, escoger una medida de interés es una decisión difícil de tomar debido a que todas tienen sus cualidades y deficiencias [122]. Por lo tanto, no existe una medida óptima y una forma de resolver este problema es encontrar buenos compromisos según las particularidades de los datos y los intereses del usuario.

El interés de una regla es un amplio concepto que enfatiza características como la consistencia, cobertura, fiabilidad, peculiaridad, diversidad, novedad, sorpresa, utilidad y accionabilidad [49]. Según esta definición, Geng y Hamilton [49] clasificaron las RA en tres categorías: objetivas, subjetivas y semánticas. Las medidas objetivas se basan o bien en los factores estadísticos de los datos o bien en la forma de las reglas. Por su lado, las medidas subjetivas toman en cuenta los datos y a los usuarios de esos datos que pueden aportar su conocimiento. Mientras las medidas semánticas consideran un conocimiento del dominio no asociado a los datos, sino que está más relacionado con los objetivos del usuario.

La presente memoria se enfoca en las medidas objetivas dado que éstas dependen solamente de los datos. A través de dichas medidas se consideran factores como la consistencia, cobertura, fiabilidad, peculiaridad y diversidad, en las cuales no interviene el usuario ni otro conocimiento del dominio. Una de estas medidas es el Factor de Certeza (FC) [117] presentado en (2.4), el cual ha sido propuesto como una alternativa a la confianza [12].

$$FC(X \Rightarrow Y) = \begin{cases} \frac{Conf(X \Rightarrow Y) - Sop(Y)}{1 - Sop(Y)}, & Conf(X \Rightarrow Y) > Sop(Y) \\ \frac{Conf(X \Rightarrow Y) - Sop(Y)}{Sop(Y)}, & Conf(X \Rightarrow Y) < Sop(Y) \\ 0, & \text{en otro caso.} \end{cases} \quad (2.4)$$

Usualmente, las medidas como el factor de certeza se definen contando la frecuencia de ejemplos o contraejemplos de una regla. Es importante para ello diferenciar la representación de una regla de asociación  $X \Rightarrow Y$  que se enfoca en la ocurrencia simultánea de  $X$  y  $Y$ , a la implicación lógica  $P \rightarrow Q$ . Esta segunda se refiere a la no ocurrencia de  $P$  o a la ocurrencia de  $Q$  ( $\neg P \vee Q$ ). Seguidamente, se muestran otros ejemplos de medidas desde esta perspectiva, pero antes, para mayor claridad, denótese dada la regla de asociación  $X \Rightarrow Y$ :

- $n = |T|$ , el número total de registros.
- $n_X = |X|$ , el número total de registros que satisfacen  $X$ .
- $n_Y = |Y|$ , el número total de registros que satisfacen  $Y$ .
- $n_{XY} = |X \cup Y|$ , el número total de registros que satisfacen  $X$  y  $Y$  (ejemplo de la regla).
- $n_{X\bar{Y}} = |X \cup \bar{Y}|$ , el número total de registros que satisfacen  $X$  pero no  $Y$  (contraejemplo de la regla).

Para mayor detalle de estas cardinalidades definidas obsérvese la Figura 2.1.

Finalmente, se muestra mediante estas notaciones varias medidas de interés en la Tabla 2.1, junto a su referencia bibliográfica.

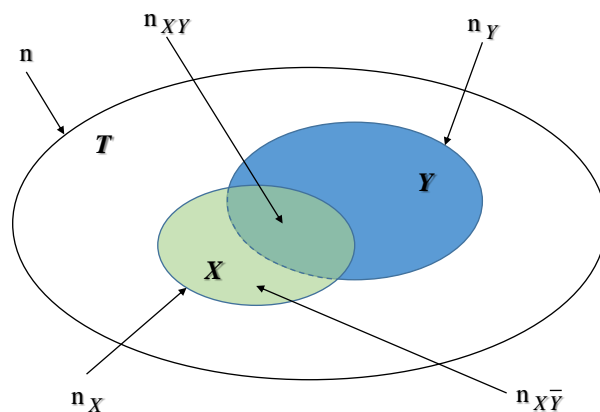


Figura 2.1: Cardinalidades de la regla de asociación  $X \Rightarrow Y$ .

Tabla 2.1: Lista de medidas de interés para reglas de asociación

Medida	Definición	Referencia
<i>Support</i>	$\frac{n_{XY}}{n}$	[1]
<i>Confidence</i>	$\frac{n_{XY}}{n_X}$	[1]
<i>Information Gain</i>	$\log\left(\frac{nn_{XY}}{n_X n_Y}\right)$	[32]
<i>Bayes Factor</i>	$\frac{n_{XY} n_{\bar{X}\bar{Y}}}{n_Y n_{X\bar{Y}}}$	[67]
<i>Conviction</i>	$\frac{n_X n_{\bar{Y}}}{m_{X\bar{Y}}}$	[19]
<i>Least Contradiction</i>	$\frac{n_{XY} - n_{X\bar{Y}}}{n_Y}$	[10]
<i>Laplace</i>	$\frac{n_{XY} + 1}{n_X + 2}$	[52]
<i>Lift</i>	$\frac{m_{XY}}{n_X n_Y}$	[18]
<i>Zhang</i>	$\frac{m_{XY} - n_X n_Y}{\max\{n_{XY} n_{\bar{Y}}, n_Y n_{X\bar{Y}}\}}$	[132]

### 2.1.3 Algoritmos clásicos de minería

Numerosos son los algoritmos propuestos en la actualidad para descubrir RA [13, 88, 86, 134]. Dentro de ellos los métodos clásicos más conocidos son el algoritmo Apriori [2], ECLAT (*Equivalence CLAss Transformation*) [130] y FP-Growth [58]. El problema de encontrar dichas RA por éstos y otros algoritmos es usualmente descompuesto en dos subproblemas:

- Encontrar aquellos ítems cuyos ocurrencias en la base de datos excedan un umbral predefinido. Estos ítems son llamados ítems frecuentes.
- Generar las RA desde estos ítems frecuentes.

Existe un amplio consenso en la literatura de que el primer subproblema es el más importante. Esto es debido al consumo de tiempo y memoria que involucra encontrar los ítems frecuentes.

El algoritmo Apriori, presentado por Agrawal y colaboradores en el año 1993 [2], es considerado el algoritmo básico de minería de RA, siendo el primer gran salto en la historia de las RA. Este comienza cada nivel generando los ítems candidatos. Posteriormente, explora la BD para hallar el soporte de cada ítem y desechar aquellos que no cumplen con el umbral predefinido. En su primera exploración a la BD colecciona, a partir de los ítems candidatos, todos los 1-ítems frecuentes, es decir, los ítems frecuentes de un único ítem. Estos 1-ítems frecuentes son utilizados para generar los 2-ítems candidatos de los cuales se desechar, en una segunda exploración a la BD, aquellos que no cumplen con el soporte establecido, obteniéndose de esta forma los 2-ítems frecuentes. Así, sucesivamente, el algoritmo procede generando los ítems frecuentes de tamaño  $k$  con los ítems frecuentes de tamaño  $k-1$ . El algoritmo termina cuando no se pueden generar nuevos candidatos para la próxima iteración. El esquema de funcionamiento general de Apriori se describe en el Algoritmo 1.

**Algoritmo 1:** Pseudocódigo del algoritmo Apriori**Entrada:**

base de datos  $BD$ ,  
 soporte mínimo  $\varepsilon$ ,  
 confianza mínima  $\xi$

**Salida:**

todas las reglas de asociación  $Rgls$

**Método:**

```

1  $L_1 \leftarrow$  1-ítemsets frecuentes;
2 for ( $k \leftarrow 2$ ;  $L_{k-1} \neq \emptyset$ ;  $k++$ ) do begin
3    $C_k \leftarrow$  Apriori-gen( $L_{k-1}$ ); /* generar nuevos candidatos */
4   forall transacciones  $T \in BD$  do
5      $C_t \leftarrow$  subconjunto( $C_k, T$ ); /* candidatos contenidos en  $T$  */
6     forall candidatos  $C \in C_t$  do
7       Cantidad( $C$ )  $\leftarrow$  Cantidad( $C$ ) + 1; /* incrementar el
8         soporte */
9        $L_k \leftarrow \{C \in C_t \mid \text{Cantidad}(C) \geq \varepsilon \times |BD|\}$ ;
9  $L_f \leftarrow \bigcup_k L_k$ ;
10  $Rgls \leftarrow$  GenerarReglas( $L_f, \xi$ );
11 retorna:  $Rgls$ 

```

De este procedimiento se puede observar que el algoritmo realiza  $k$  exploraciones a la **BD**. Convirtiéndose en uno de sus principales inconvenientes, especialmente cuando se enfrenta a grandes volúmenes de datos. Otro problema es el complejo proceso de generación de candidatos. Basados en el algoritmo Apriori, nuevos algoritmos han sido diseñados con modificaciones o mejoras [100, 114, 83]. No obstante, estos algoritmos mantienen múltiples exploraciones a la **BD**.

Para superar estos retos presentados por Apriori algunos trabajos han utilizado algoritmos basados en árboles, clasificándolos según las estructuras utilizadas en: algoritmos similares a Apriori y algoritmos basados en árboles. Uno de ellos es el algoritmo ECLAT, propuesto por Zaki y colaboradores en el año 1997 [130]. Este algoritmo combina una búsqueda de primero en profundidad con una lista de todos los identificadores de las transacciones por ítems. Además, utiliza una optimización llamada intersecciones rápidas. La cual establece que donde se intersecten dos listas de identificadores, solo interesa el resultado si su cardinalidad alcanza el soporte mínimo. ECLAT explora la base de datos una única vez, pero el número de ítemsets generados es aún mayor que el generado por el algoritmo Apriori.

Otro gran salto en la minería de RA basadas en estructuras de árboles es el algoritmo FP-Growth [58], introducido por Han y colaboradores en el año 2000. Este algoritmo supera las limitaciones de Apriori y ECLAT. Según varios autores, esta es la mejor técnica para la generación de ítemsets propuesta hasta el momento [93]. El algoritmo consiste en dos fases. La primera fase construye la estructura arbórea *Frequent Pattern tree* (FP-tree), la cual es utilizada para compactar la base de datos almacenando solamente los ítems frecuentes. Esta estructura constituye el núcleo de dicho algoritmo puesto que evita el proceso de generación de candidatos y los accesos múltiples a la base de datos. En un primer paso, para la construcción de esta estructura, se explora la base de datos y se encuentran los 1-ítemsets frecuentes. Luego, en un segundo paso de exploración, se ordenan las transacciones y se obvian los ítems no frecuentes. Suponga una BD con los ítems  $It=\{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p\}$ , un soporte mínimo de 0.6, y cinco transacciones de estos ítems en la Tabla 2.2.

Tabla 2.2: Ejemplo de una relación de transacciones con los ítems frecuentes ordenados

ID_Transacción	Ítems Comprados	Ítems Frecuentes Ordenados
101	f,a,c,d,g,i,m,p	f,c,a,m,p
102	a,b,c,f,l,m,o	f,c,a,b,m
103	b,f,h,j,o	f,b
104	b,c,k,s,p	c,b,p
105	a,f,c,e,l,p,m,n	f,c,a,m,p

Los ítems frecuentes de este ejemplo son:  $f=4$ ,  $c=4$ ,  $a=3$ ,  $b=3$ ,  $m=3$  y  $p=3$ . Con estos ítems ordenados en cada transacción se construye, en una segunda exploración, el FP-tree como muestra la Figura 2.2.



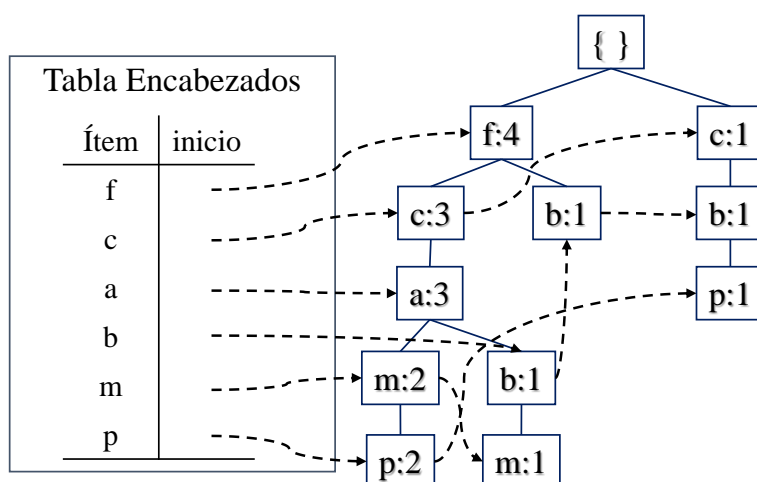


Figura 2.2: FP-tree construido a partir de la relación mostrada en la Tabla 2.2.

Todo el procesamiento posterior del algoritmo es realizado mediante accesos al FP-tree. La segunda fase involucra generar los patrones frecuentes a partir del FP-tree. Esta fase es realizada mediante el procedimiento llamado FP-growth, que da nombre al algoritmo [58].

## 2.2 Reglas de Asociación Difusas

Originalmente, los algoritmos y definiciones presentados en la literatura de reglas de asociación abarcan solamente aquellas reglas binarias. Estas reglas pueden ser fácilmente representadas en bases de datos transaccionales binarias interpretando cada ítem como un atributo cuyo dominio es  $[0,1]$  lo cual refleja la presencia/ausencia de los ítems respectivos. Sin embargo, en las aplicaciones del mundo real los datos poseen una estructura mucho más rica. Tal es el caso de los atributos cuantitativos (e.g. hora, edad, salario, etc.) o los atributos categóricos (e.g. sexo, color, nombre, etc.).

En este contexto, las bases de datos relacionales consideran cada ítem como un par  $\langle \text{atributo}, \text{valor} \rangle$ . Esta solución presenta un problema en cuanto a la cantidad de ítems que pueden ser generados. En el caso del dominio de los atributos categóricos, aunque posiblemente grande, la cantidad de ítems es generalmente finito. Para los atributos cuantitativos el dominio usualmente es infinito, presentando un problema de granularidad [41, 39, 38]. La complejidad de la búsqueda aumenta exponencialmente debido a esta granularidad. Las reglas de asociación cuantitativas [119] han manejado este problema agrupando, tanto manual como automáticamente, los valores de los dominios numéricos de alta granularidad. Estos grupos o clústeres son entonces considerados como el nuevo dominio del atributo, representando cada ítem como un par  $\langle \text{atributo}, \text{intervalo} \rangle$ .

Esta solución mediante agrupamiento presenta algunos retos inherentes [41, 38]. Uno de dichos retos está relacionado con el significado de los clústeres. En varias ocasiones, los conceptos son imprecisos y resulta imposible representarlos mediante intervalos que sean sencillos de entender por el usuario. Para estos conceptos, además, resulta complejo definir una frontera clara entre dos intervalos consecutivos pues ¿cómo explicar que los puntos cercanos a la frontera no pertenecen al siguiente intervalo? Otro de los retos en el agrupamiento está relacionado con la sensibilidad a las variaciones de fronteras. Incluso el más mínimo movimiento de una frontera puede cambiar drásticamente el soporte de un ítem.

La teoría de subconjuntos difusos, propuesta Zadeh desde el año 1965 [129], provee una herramienta importante que resuelve los problemas anteriormente expuestos. Mediante clústeres difusos del dominio de un atributo se puede obtener una buena representación de los conceptos imprecisos. En esta propuesta, los ítems son considerados como un par  $\langle \text{atributo}, \text{etiqueta} \rangle$ . Una *etiqueta* tiene la representación interna de un conjunto difuso sobre el dominio del *atributo*. Estos ítems son llamados ítems difusos y las reglas, Reglas de Asociación Difusas (RAD).

Existen varios trabajos que estudian las RAD [40, 59]. La presente memoria se enfoca en el marco de trabajo presentado por Delgado y colaboradores en el año 2003 [39], una de las propuestas más representativas para este tipo de reglas. Formalmente, sea  $I$  un conjunto de ítems y  $T'$  un conjunto de transacciones difusas donde cada transacción difusa es un subconjunto difuso de  $I$ . Sea  $\tilde{\tau} \in T'$  una transacción difusa y  $\tilde{\tau}(i_k)$  el grado de membresía de  $i_k$  en  $\tilde{\tau}$ . Una regla de asociación difusa es una implicación de la forma  $A \Rightarrow B$  tal que  $A \subseteq I$ ,  $B \subseteq I$ ,  $A \cap B = \emptyset$  y  $A, B \neq \emptyset$ . En esta sentencia  $A$  y  $B$  son dos subconjuntos clásicos o *crisp*, llamados antecedente y consecuente de la regla respectivamente.

En las bases de datos relacionales, sea  $R = \{At_1, \dots, At_m\}$  un conjunto de atributos y  $Etq(At_k) = \{E_1^{At_k}, \dots, E_n^{At_k}\}$  un conjunto de etiquetas lingüísticas definidas en el  $\text{Dom}(At_k)$ ,  $\forall At_k \in R$  y  $E_j^{At_k}: \text{Dom}(At_k) \rightarrow [0,1]$ . Se puede inferir que el conjunto de transacciones donde el ítem dado aparece, es un conjunto difuso [24]. Para el ítem  $i_k$  en  $T'$  el siguiente subconjunto difuso de  $T'$  es obtenido:

$$\tilde{\Gamma}_{i_k} = \sum_{\tilde{\tau} \in T'} \tilde{\tau}(i_k) / \tilde{\tau} \quad (2.5)$$

Esta representación en (2.5) puede ser extendida a los ítemsets. Sea  $I_0 \in R$  un ítemset, su representación es el próximo subconjunto de  $T'$  en (2.6) [24].

$$\tilde{\Gamma}_{I_0} = \bigcap_{i \in I_0} \tilde{\Gamma}_i = \min_{i \in I_0} \tilde{\Gamma}_i \quad (2.6)$$

Para ilustrar esta definición, considere la siguiente información acerca de seis personas en la relación  $r_0$ , mostrada en la Tabla 2.3.

Tabla 2.3: Edad y salario de cinco personas ( $r_0: R$ )

ID	Edad	Salario
1	26	800
2	52	1100
3	65	3000
4	65	3000
5	26	800

Estos atributos pueden ser fuzificados, bien para obtener una mayor información semántica o bien para disminuir la granularidad involucrada en los dominios de ambos atributos. La Figura 2.3 muestra tres etiquetas lingüísticas definidas en el dominio Edad y en el dominio Salario. Estas etiquetas son:  $Etq(Edad) = \{joven, media, anciano\}$  y  $Etq(Salario) = \{bajo, medio, alto\}$ .

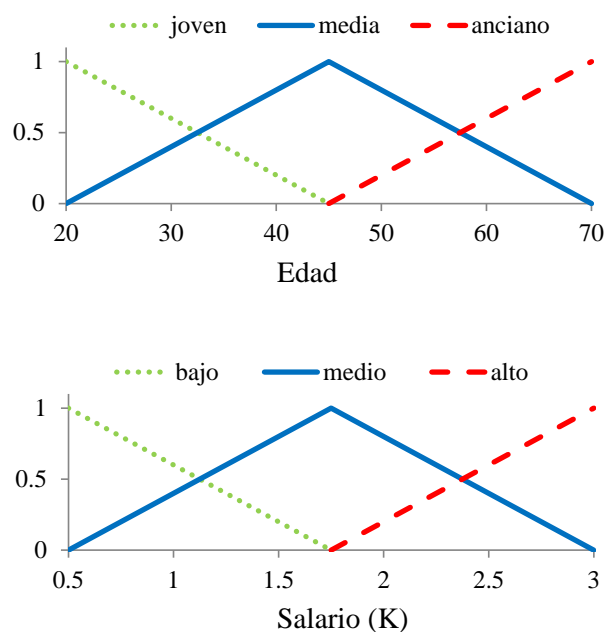


Figura 2.3: Etiquetas lingüísticas definidas sobre el atributo Edad (arriba) y el atributo Salario (abajo).

Cada tupla  $t \in r_0$  se asocia a una única transacción difusa. En este caso, una transacción difusa contiene más de un ítem debido a que es posible, para un único valor en la Tabla 2.3, ajustarse a más de una etiqueta con cierto grado de membresía [89]. En la Tabla 2.4 se muestran todas las transacciones difusas asociadas a  $r_0$  para las etiquetas lingüísticas definidas.

Tabla 2.4: Transacciones difusas para la relación de la Tabla 2.3

	$\tilde{\tau}_L^{t_1}$	$\tilde{\tau}_L^{t_2}$	$\tilde{\tau}_L^{t_3}$	$\tilde{\tau}_L^{t_4}$	$\tilde{\tau}_L^{t_5}$
$\langle \text{Edad, joven} \rangle$	0.76	0	0	0	0.76
$\langle \text{Edad, media} \rangle$	0.24	0.72	0.2	0.2	0.24
$\langle \text{Edad, anciano} \rangle$	0	0.28	0.8	0.8	0
$\langle \text{Salario, bajo} \rangle$	0.76	0.52	0	0	0.76
$\langle \text{Salario, medio} \rangle$	0.24	0.48	0	0	0.24
$\langle \text{Salario, alto} \rangle$	0	0	1	1	0

Verifíquese en esta tabla los subconjuntos difusos:

$$\begin{aligned} \tilde{\Gamma}_{\{\langle \text{Edad, media} \rangle\}}^{r_0} &= 0.24/\tilde{\tau}_L^{t_1} + 0.72/\tilde{\tau}_L^{t_2} + 0.2/\tilde{\tau}_L^{t_3} + 0.2/\tilde{\tau}_L^{t_4} + 0.24/\tilde{\tau}_L^{t_5} \\ &= 1.60 \quad \text{Según (2.5)}. \end{aligned}$$

$$\begin{aligned} \tilde{\Gamma}_{\{\langle \text{Salario, medio} \rangle\}}^{r_0} &= 0.24/\tilde{\tau}_L^{t_1} + 0.48/\tilde{\tau}_L^{t_2} + 0.24/\tilde{\tau}_L^{t_5} \\ &= 0.96 \quad \text{Según (2.5)}. \end{aligned}$$

$$\begin{aligned} \tilde{\Gamma}_{\{\langle \text{Edad, media} \rangle, \langle \text{Salario, medio} \rangle\}}^{r_0} &= 0.24/\tilde{\tau}_L^{t_1} + 0.48/\tilde{\tau}_L^{t_2} + 0.24/\tilde{\tau}_L^{t_5} \\ &= 0.96 \quad \text{Según (2.6)}. \end{aligned}$$

Para las RAD, en aras de medir su interés, es necesario aplicar herramientas de razonamiento aproximado. En [39] se propuso un método basado en la evaluación de sentencias cuantificadas. Este método es una generalización del marco de trabajo ordinario en métricas de RA. Por ejemplo, el factor de certeza (2.4) puede ser utilizado, bajo las definiciones de este método, para medir el interés de las RAD del mismo modo que para las RA.

Tómese por ejemplo la RAD  $\langle Edad, media \rangle \Rightarrow \langle Salario, medio \rangle$  desde los ítems difusos de la Tabla 2.4. Según el razonamiento apuntado en [39], el soporte y la confianza de esta regla se definen como:

$$Sop(\langle Edad, media \rangle \Rightarrow \langle Salario, medio \rangle) = \frac{|\tilde{\Gamma}_{A \cup B}|}{|T|} = \frac{0.96}{5} = 0.19$$

$$Conf(\langle Edad, media \rangle \Rightarrow \langle Salario, medio \rangle) = \frac{|\tilde{\Gamma}_{A \cup B}|}{|\tilde{\Gamma}_A|} = \frac{0.96}{1.60} = 0.6$$

Estas medidas, al igual que en las RA, han sido utilizadas por algoritmos para extraer RAD. Varios de estos algoritmos propuestos para las RAD están basados en Apriori [60, 61]. Ellos extienden las técnicas conocidas en la extracción de RA para la teoría difusa generando ítemssets candidatos difusos y ítemssets frecuentes difusos.

## 2.3 Dependencias Aproximadas mediante Reglas de Asociación

Un concepto fundamental en la teoría relacional de bases de datos son las dependencias funcionales [47, cap. 10]. Ellas establecen una restricción entre dos conjuntos de atributos de un esquema relacional  $R$  denotado por  $C \rightarrow D$  donde  $C, D \subseteq R$ . La restricción impone que para cualquier instancia  $r$  de  $R$  se cumple que  $\forall t_1, t_2 \in r$  si  $t_1[C] = t_2[C]$  entonces  $t_1[D] = t_2[D]$ .

Como se puede observar, esta restricción es muy fuerte y por lo tanto difícil de descubrir como un nuevo conocimiento [115]. Permitir algunas excepciones a esta restricción es una forma de relajarla y encontrar información nueva e interesante. Una alternativa a estas excepciones son las Dependencias Aproximadas (DA), también conocidas en la literatura como determinaciones parciales o dependencias funcionales aproximadas. Las DA pueden ser informalmente definidas como dependencias “casi” funcionales o dependencias funcionales con excepciones, en donde se admite un cierto grado de “error”. La discusión acerca de las dependencias parciales se remonta a los años 70, donde se presentaba el reto de definir medidas

para las DA [87]. En este sentido, varios autores han realizado interesantes aportes [105, 36, 66, 118]. Un ejemplo de ello es la ampliamente utilizada medida  $g_3$  (2.7). Esta medida se define como una fracción, donde el numerador es el máximo de tuplas que son necesarias eliminar para que se cumpla la dependencia funcional y el denominador es la cantidad total de tuplas [69].

$$g_3(C \rightarrow D) = \frac{|r| - \max\{|s| \mid s \subseteq r, s \models C \rightarrow D\}}{|r|} \quad (2.7)$$

Un trabajo muy original en la extracción de DA es el presentado por Sánchez y colaboradores en el año 2008 [112]. La particularidad que distingue este estudio es que definen las DA basado en conceptos de las RA. Esta definición permite por tanto, medir y extraer DA bajo las métricas y algoritmos de las RA. En la presente memoria se consideran las DA bajo esta definición, la cual se presenta a continuación con mayor detalle.

Sea  $r$  una instancia de un esquema relacional  $R$  donde se tienen los conjuntos de atributos  $V, W \subset R$  con  $V \cap W = \emptyset$ . Téngase además  $R = \{At_1, \dots, At_m\}$  donde  $n = |r|$ . Una dependencia aproximada entre los conjuntos de atributos  $V$  y  $W$  se denota como  $V \rightarrow W$ . Esta DA es considerada una RA mediante la interpretación de conceptos abstractos como ítem, ítemset y conjunto de transacciones [112].

**Definición 2.3.1** [112] Un ítem es un objeto asociado a un atributo de  $R$ . Para cada atributo  $At_k \in R$  se denota  $it_{At_k}$  el ítem asociado.

Este ítem asociado a un atributo es un objeto abstracto, independiente a las instancias de  $R$  y al dominio del atributo. Concepto que permite definir inmediatamente a un ítemset.

**Definición 2.3.2** [112] Sea  $V \subseteq R$ . Entonces un ítemset  $I_V$  se define como:

$$I_V = \{it_{At_k} \mid At_k \in V\}$$

**Definición 2.3.2** [112] En un conjunto de transacciones  $T_r$  se cumple que para cada par de tuplas  $\langle t, s \rangle \in r \times r$  existe una transacción  $ts \in T_r$  donde:

$$it_{At_k} \in ts \Leftrightarrow t[At_k] = s[At_k]$$

De la definición anterior se puede observar que el conjunto de transacciones presentado contiene  $r^2$  transacciones, i.e.  $|T_r| = |r \times r| = n^2$  considerando  $|r| = n$ . Aparentemente esto conlleva a aumentar la complejidad de obtención de DA a un  $O(n^2)$  con respecto al número de tuplas, sin embargo, es posible mantener un  $O(n)$  en esta propuesta [112]. Un ejemplo de este estudio es presentado a través de la Tabla 2.5 y la Tabla 2.6.

Tabla 2.5: Datos de tres estudiantes en la relación  $r$ 

ID	Año	Curso	Apellido
1	1991	3	Smith
2	1991	4	Smith
3	1991	4	Smith

Tabla 2.6: El conjunto de transacciones  $T_r$  de la relación  $r$ 

Par	$\dot{it}_{ID}$	$\dot{it}_{Año}$	$\dot{it}_{Curso}$	$\dot{it}_{Apellido}$
$\langle 1, 1 \rangle$	1	1	1	1
$\langle 1, 2 \rangle$	0	1	0	1
$\langle 1, 3 \rangle$	0	1	0	1
$\langle 2, 1 \rangle$	0	1	0	1
$\langle 2, 2 \rangle$	1	1	1	1
$\langle 2, 3 \rangle$	0	1	1	1
$\langle 3, 1 \rangle$	0	1	0	1
$\langle 3, 2 \rangle$	0	1	1	1
$\langle 3, 3 \rangle$	1	1	1	1

Finalmente, con las definiciones anteriormente presentadas se pueden definir las DA.



**Definición 2.3.3** [112] Una dependencia aproximada  $V \rightarrow W$  en la relación  $r$  es una regla de asociación  $I_V \Rightarrow I_W$  en  $T_r$ .

De acuerdo con esta definición, una regla de asociación en  $T_r$  se corresponde a una dependencia aproximada en  $r$ . Como resultado directo se obtiene que las métricas de una regla de asociación en  $T_r$  pueden ser utilizadas para evaluar las dependencias aproximadas en  $r$ . De esta forma el soporte de la regla  $I_V \Rightarrow I_W$ ,  $Sop(I_V \Rightarrow I_W)$ , puede ser establecido como se presentó en la Sección 2.1, al igual que la confianza:

$$Conf(I_V \Rightarrow I_W) = \frac{Sop(I_V \Rightarrow I_W)}{Sop(I_V)} = \frac{Sop(I_{VW})}{Sop(I_V)}$$

**Definición 2.3.3** [112] El soporte y la confianza de una **DA**  $V \rightarrow W$  es el soporte y la confianza de la correspondiente **RA**  $I_V \Rightarrow I_W$ , i.e:

$$Sop(V \rightarrow W) = Sop(I_V \Rightarrow I_W)$$

$$Conf(V \rightarrow W) = Conf(I_V \Rightarrow I_W)$$

Desde esta propuesta de **DA** se puede definir una dependencia funcional como aquella **DA** en la cual el factor de certeza (2.4) es igual a uno. Es decir si  $FC(V \rightarrow W) = 1$ , entonces  $V \rightarrow W$  es una dependencia funcional [115]. Mayor información y ejemplos detallados de esta representación pueden ser encontrados en [115, 112].

## 2.4 Minería incremental de Reglas de Asociación

Los algoritmos de minería para la extracción del conocimiento que se han expuesto hasta el momento responden a una estrategia estática de los datos. Es decir, independientemente del algoritmo que se utilice, el conocimiento extraído responde al momento específico en que se ejecutó dicho algoritmo [29]. Estas propuestas no consideran ningún resultado de algoritmos que se hayan ejecutado anteriormente por lo que se ejecutan “desde cero”. Además, siempre incluyen la exploración completa de la base de datos.

En las aplicaciones reales no es común que los datos permanezcan estáticos. De hecho, su naturaleza es que estén en constante cambio mediante el procesamiento de nuevas transacciones y generalmente, aumentando su volumen. Esto produce que el conocimiento obtenido se convierta en inválido y eventualmente, obsoleto [77]. Particularmente, en algunas aplicaciones reales este proceso de obsolescencia es más marcado. Tal es el caso de las emergentes aplicaciones con flujo de datos donde usualmente los datos arriban con gran velocidad, volumen y de forma continua [85]. Un ejemplo lo constituyen los sistemas que realizan análisis de redes sensoriales, tráfico de red, actividad de la web, registro de llamadas telefónicas y redes sociales [85, 84].

Bajo estas circunstancias dinámicas los algoritmos convencionales simplemente pueden re-ejecutarse en la base de datos modificada, encontrando nuevas reglas de asociación o simplemente actualizando las reglas que se conocían previamente como muestra la Figura 2.4. Esta solución elemental presenta varias dificultades asociadas al ineficiente manejo de los recursos que fueron utilizados en ejecuciones anteriores:

- Los algoritmos consumen en la nueva base de datos casi el mismo tiempo que en la base de datos original.
- La información previamente obtenida, como los ítemsets frecuentes, no se utilizan en las nuevas ejecuciones de los algoritmos.

La minería incremental de reglas de asociación es una extensión natural al reto de extraer conocimientos en datos que se acumulan incrementalmente. A diferencia de los algoritmos convencionales, los algoritmos de minería incremental consideran los resultados que se obtuvieron en ejecuciones anteriores, enfocándose de esta forma en las nuevas transacciones introducidas a la base de datos como muestra la Figura 2.5.

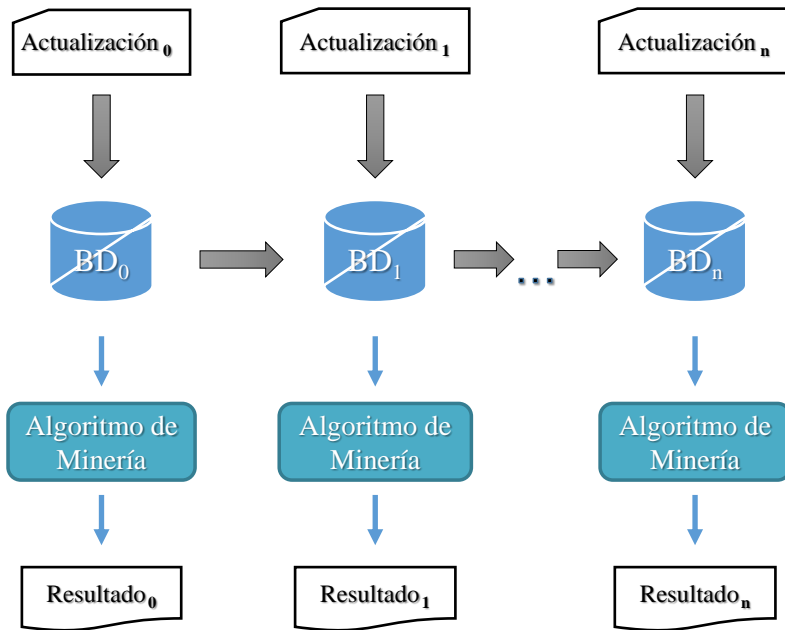


Figura 2.4: Método de re-ejecución "desde cero" de los algoritmos convencionales de minería.

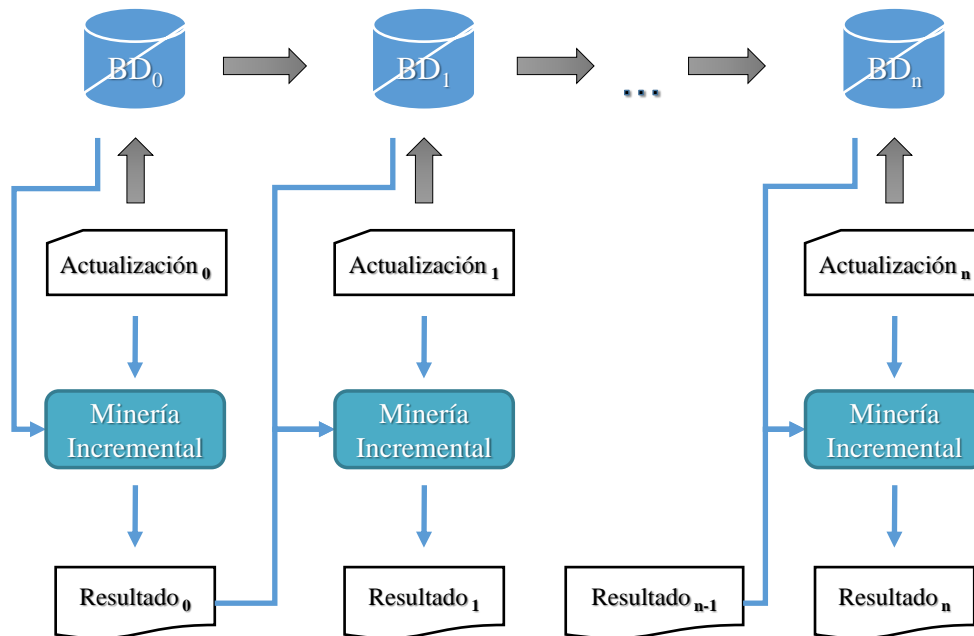


Figura 2.5: Método de minería incremental para reglas de asociación.

El trabajo inicial que llamó a la reflexión en este sentido fue el presentado por Cheung y colaboradores en el año 1996 [29]. En este artículo se presenta el primer algoritmo incremental, el algoritmo *Fast UPdate algorithm* (FUP), como una modificación de Apriori. Dicho algoritmo explora primeramente las nuevas transacciones y detecta los ítemset perdedores (i.e. aquellos ítemsets que se convierten en infrecuentes) y los ítemsets ganadores (i.e. ítemsets no frecuentes que se convierten en frecuentes). Detectar la primera categoría es relativamente simple dado que los ítemsets frecuentes ya se conocen. La segunda categoría es la que presenta mayores dificultades. Para esta categoría se establece que un ítemset infrecuente en las nuevas transacciones no puede convertirse en ganador [29]. El algoritmo FUP fue inicialmente concebido únicamente para transacciones con inserciones, sin embargo, las bases de datos manejan también modificaciones y eliminaciones. FUP<sub>2</sub> [30] se introdujo para manejar estas transacciones no concebidas en el algoritmo original. Otros algoritmos incrementales basados en Apriori son: UWEP (*Update With Early Pruning*) [9], MAAP (*Maintaining Association rules with Apriori Property*) [133] y PELICAN [124].

Aunque estos algoritmos mejoran significativamente el desempeño de los algoritmos convencionales, aún es necesario realizar una cantidad considerable de exploraciones a la base de datos. Un concepto introducido en la minería incremental para disminuir la cantidad de accesos a la base de datos fue el de los ítemsets potencialmente frecuentes (*pre-large-itemsets*) [65]. Estos ítemsets no son realmente frecuentes, sino que prometen convertirse frecuentes en el futuro. Para ello se definen dos umbrales, uno de soporte mínimo y otro de soporte máximo. El soporte máximo es el mismo utilizado por los algoritmos convencionales. La novedad la presenta el soporte mínimo estableciendo qué ítemsets tienen mayor probabilidad de convertirse frecuentes en el futuro [65].

Otros algoritmos de minería incremental han extendido estructuras basadas en árboles para disminuir el acceso a la base de datos. Un ejemplo de ello son las extensiones realizadas a la estructura FP-tree. Dentro de ellas destacan: EFP-tree (Extended FP-tree) [77], FUFPP-tree (*Fast-Updated FP-tree*) [63] y Pre-FUFPP [81]. EFP-tree presenta una estructura idéntica al FP-tree en un inicio,

expandiéndose luego durante las iteraciones del algoritmo. FUFP-tree es una estructura que facilita la actualización. Es similar a FP-tree con la excepción de tener un enlace bidireccional entre el nodo padre e hijo. Por último, Pre-FUFP es una modificación de FUFP-tree bajo el concepto de *pre-large* ítemsets. Estos algoritmos, aunque difieren en la forma que manejan los nuevos incrementos a la base de datos, mantienen una característica en común: todos ejecutan el algoritmo FP-Growth en sus pasos finales para hallar los ítemsets frecuentes. Existen además otras estructuras basadas en árboles utilizadas por algoritmos incrementales. Un ejemplo es CATS-tree (*Compressed and Arranged Transaction Sequences tree*) [31] que comparte varias similitudes con FP-tree. Otro ejemplo es CAN-tree (*Canonical-order-tree*) [76] el cual se construye en una única exploración de la base de datos.

Es importante señalar que muchas de estas investigaciones han sido extendidas para manejar la extracción de reglas de asociación difusas. Específicamente, varias de estas técnicas incrementales basadas en árboles para las RAD han extendido la estructura FP-tree, estableciendo un FP-tree difuso (*fuzzy* FP-tree) [99, 80, 82]. La estructura *fuzzy* FP-tree, al igual que su homóloga *crisp*, es utilizada para condensar la base de datos en una estructura que solo contiene los ítemsets difusos frecuentes, reduciendo de esta forma la cantidad de exploraciones a la base de datos. Una de las extensiones a esta estructura más estudiada en la literatura corresponde a MFFP-tree (*Multiple Fuzzy-term* FP-tree) [64, 62]. Las RAD son posteriormente extraídas de estas estructuras mediante FFP-Growth (Fuzzy FP-Growth) [126] de forma similar a FP-Growth.

## 2.5 Fundamentos de Bases de Datos Activas

El descubrimiento de reglas de asociación junto a sus extensiones difusas y aproximadas no ocurren únicamente en entornos estáticos. Constantemente nuevas transacciones son llevadas a cabo en las bases de datos, desactualizando de esta forma el conocimiento previamente extraído.

Los sistemas gestores de bases de datos tradicionalmente han sido vistos como repositorios pasivos, que guardan y ofrecen datos en respuesta directa a las demandas de sus usuarios, sin realizar ninguna operación propia. Contrario a este popular punto de vista, las bases de datos no son simples repositorios inactivos. A lo largo del desarrollo de sistemas de información, una rama de las bases de datos fue desarrollando potentes mecanismos activos para crear nuevas funcionalidades, siendo utilizados por las aplicaciones más avanzadas. A los sistemas de esta rama se les conoce como sistemas gestores de Bases de Datos Activas (BDA) [51, 44, 68].

Este área permitió modelar no solo el aspecto estructural de las aplicaciones, sino además modelar su comportamiento. Por tanto, algunos autores han definido los sistemas gestores de BDA como sistemas gestores de bases de datos convencionales que permiten además, expresar un comportamiento reactivo en los datos [44]. Un enfoque común para modelar este comportamiento es utilizando reglas que tienen tres componentes en su definición: un evento, una condición y una acción. El término de reglas, en bases de datos activas no se debe confundir con las reglas de asociación y sus extensiones.

DEFINICIÓN REGLA *nombre\_regla*

**ON** *evento*

**IF** *condición*

**DO** *acción*

Estas reglas activas son típicamente conocidas como reglas de **Evento Condición Acción** (ECA) [103]. El evento de una regla describe algo sucedido en el tiempo por lo cual la regla debe responder o reaccionar. Por otro lado, la acción describe la(s) tarea(s) que debe(n) ser llevada(s) a cabo si la condición resulta verdadera. Según Paton y Díaz [103], los eventos pueden tener diferentes fuentes generadoras del evento, algunas de ellas son:

- **Operaciones estructurales.** Son eventos ocurridos por operaciones sobre las estructuras de los datos como inserciones, actualizaciones o eliminaciones en tipos de entidades.
- **Invocación por comportamiento.** Son eventos lanzados cuando ocurre alguna operación definida por el usuario.
- **Excepciones.** Eventos surgidos como resultado de alguna excepción producida.
- **Programadas.** Eventos que ocurren según una programación predefinida en el tiempo.
- **Externas.** Situación en la cual el evento se origina fuera de la base de datos.

En la presente memoria nos centraremos en los eventos vinculados a operaciones que ocurren directamente sobre los datos. Estas operaciones permiten definir reglas totalmente independientes al usuario, que toman en consideración únicamente el comportamiento establecido sobre los cambios de los datos.

Otro concepto muy relacionado con los eventos es su granularidad. La granularidad indica si el evento se encuentra definido para una única entidad, o definido sobre un conjunto de entidades. Estos tipos de granularidad son por tanto primitivos en el primer caso y compuestos en el segundo. Para la presente memoria que involucra operaciones estructurales en bases de datos relacionales establecemos:

- **Eventos Estructurales Primitivos (EEP).** Son aquellos eventos que involucran las operaciones de inserción, actualización o eliminación de una única entidad.
- **Eventos Estructurales Compuestos (EEC).** Son aquellos eventos que involucran múltiples EEP o múltiples EEC.

Actualmente, los sistemas gestores de bases de datos relacionales, definidos bajo el modelo relacional de bases de datos presentado por Codd en 1970 [33], han desarrollado amplias capacidades para las reglas ECA. La presente memoria se enfoca en estos Sistemas Gestores de Bases de Datos Relacionales Activas (SGBDRA), los cuales son ampliamente utilizados en los sistemas de información de hoy día. Un SGBDRA captura el comportamiento reactivo del dominio de aplicación, con lo cual tiene la posibilidad de responder automáticamente a sucesos relevantes [44]. Por lo tanto, para considerar un sistema como SGBDRA este debe:

- Ser necesariamente un sistema gestor de base de datos relacional.
- Soportar mecanismos para la definición y manejo de reglas ECA por medio de definiciones sintácticas de eventos, condiciones y acciones.
- Soportar la evolución de las reglas.
- Tener un modelo de ejecución bien definido, capaz de detectar la ocurrencia de eventos, evaluación de condiciones y ejecución de acciones. Así como proveer mecanismos de resolución de problemas definidos por el usuario.

Los eventos estructurales en los SGBDRA producen, tal como se pudo observar en la Figura 2.4 y Figura 2.5, una serie de transiciones en la BD. Denótese una transición de la base de datos de  $BD_{s0}$  a  $BD_{s1}$  como  $(BD_{s0}, BD_{s1})$  donde  $BD_{s0}$  es el estado inicial y  $BD_{s1}$  es el estado final. En la presente memoria se consideran tres tipos de EEP que pueden provocar una transición:

- **Inserción de una tupla.** Denotado como  $\Delta t^+$ , es un evento estructural primitivo ocurrido sobre  $R$  en el momento  $m$  si la tupla  $t$  es un registro de la relación  $r:R$  en  $m$ , y  $t$  no es un registro de  $r$  en  $m-1$ .
- **Eliminación de una tupla.** Denotado como  $\Delta t^-$ , es un evento estructural primitivo ocurrido sobre  $R$  en el momento  $m$ , si la tupla  $t$  es un registro de la relación  $r:R$  en  $m$  y  $t$  no es un registro de  $r$  en  $m+1$ .



- **Actualización de una tupla.** Denotado como  $\Delta t^{-+}$ , es un evento estructural primitivo ocurrido sobre  $R$  en el momento  $m$ , si la tupla  $t$  es un registro de la relación  $r:R$  en  $t-1$ ,  $r$  no es un registro de  $r$  en  $m$  y  $t$  es un registro de  $r$  en  $m+1$ .

Es usual considerar la actualización de una tupla como su eliminación seguido de su inserción. Sin embargo, en el presente trabajo la definimos como un evento independiente  $\Delta t^{-+}$ , tal como se ha mostrado. En tal caso,  $t_0^{-+}$  es la tupla antes del evento de actualización correspondiente al estado  $BD_{s0}$  y  $t_1^{-+}$  es la tupla luego de la actualización, en el estado  $BD_{s1}$ . Estas notaciones son definidas además en la Figura 2.6, donde se puede observar la transición de una base de datos luego de ocurrir un evento de inserción, actualización y eliminación. En esta figura, se define además aquella sección de la base de datos que no ha sufrido cambios:  $BD_{sc}$ .

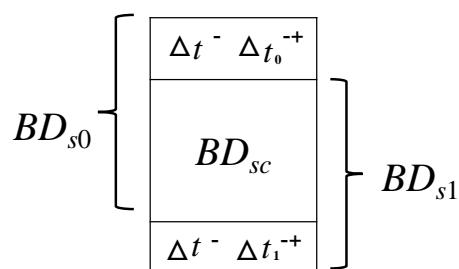


Figura 2.6: Transición de la base de datos  $BD$  luego de ocurrir un evento estructural primitivo de inserción, actualización y eliminación.

Luego de la ocurrencia de un evento es necesario, en las reglas activas, evaluar una condición para posteriormente llevar a cabo la acción. Este paso condición-acción no es necesariamente inmediato. Luego de evaluar la condición afirmativamente se presentan dos opciones acerca de cuándo ejecutar la acción:

- **Inmediata.** La acción es ejecutada inmediatamente después de evaluar la condición como verdadera.
- **Diferida.** La acción es ejecutada en un momento posterior a la evaluación verdadera de la condición, pero no necesariamente en la primera oportunidad. Generalmente se ejecutan bajo demanda o periódicamente.

Dentro del campo de investigación en **BDA** se debe considerar un área de investigación fundamental que agrupa aquellas ramas que necesitan métodos para el procesamiento de los cambios: el cómputo de los cambios (*change computation*) [123]. Dentro de este área se pueden encontrar el mantenimiento de Vistas Materializadas (**VM**), el chequeo de Restricciones de Integridad (**RI**) y las **Reglas de Negocio (RN)**.

## 2.6 Resumen

A lo largo de este capítulo se han valorado conceptos generales para la extracción de conocimiento en bases de datos (KDD), enfocándose en la minería de reglas de asociación. En particular, se detalló la definición de estas reglas, se mostraron diferentes propuestas para establecer su interés y se presentaron varios algoritmos para su extracción.

También se ofrecieron dos secciones claves en el desarrollo de esta memoria. Ellas definieron dos extensiones a las reglas de asociación: las reglas de asociación difusas y las dependencias aproximadas.

Además se abordó otra técnica de minería que abarca condiciones dinámicas en los datos: la minería incremental de reglas de asociación. De esta técnica se presentaron algunas de las estructuras que posibilitan reducir considerablemente las exploraciones en la base de datos.

Sobre el aspecto dinámico en los datos se continuó profundizando mediante los fundamentos de las bases de datos activas. Este apartado concluyó el presente capítulo definiendo varios términos que serán utilizados en el resto de la memoria.



## Modelado de reglas en Bases de Datos Activas

El presente capítulo está destinado a mostrar que las bases de datos activas poseen valiosos recursos para manejar las constantes transacciones realizadas en los datos. Indicaremos que en diferentes áreas de investigación se han propuesto múltiples técnicas que realizan tareas de mantenimiento incremental en las bases de datos. Estas técnicas desarrollan un poder reactivo muy cercano a los datos que pudiesen ser utilizadas para definir eficientes acciones sobre el conocimiento extraído. Pero antes, es necesario establecer cómo abordar las reglas de asociación y sus extensiones en este contexto activo y cómo implementarlas en las bases de datos.

Primeramente, se presentan tres secciones encaminadas a presentar tres áreas de investigación estrechamente relacionadas entre sí: Vistas Materializadas (**VM**), Restricciones de Integridad (**RI**) y **Reglas de Negocio (RN)**. En estas secciones se vincularán cada una de estas áreas con las reglas de asociación y sus extensiones, además de presentar las técnicas y algoritmos más relevantes en dichos campos para nuestro trabajo. Por último, se proveen varios recursos de las Bases de Datos Activas (**BDA**) que posibilitan la especificación de comportamientos reactivos. Estos recursos son provistos desde un alto nivel de generalización.

### 3.1 Mantenimiento de reglas mediante Vistas Materializadas

Los sistemas de bases de datos relacionales poseen la capacidad de crear relaciones *bases* o relaciones *derivadas*. Las relaciones derivadas, más conocidas como *vistas*, se encuentran definidas por una expresión en una o varias relaciones bases. Estas relaciones derivadas son virtuales en el sentido que la relación resultante debe ser re-evaluada cada vez que un usuario haga referencia a ella. Este es el término más popular acerca de la implementación de vistas, no obstante, las vistas también pueden ser materializadas.

Una **Vista Materializada (VM)** no es más que el almacenamiento en la base de datos de las tuplas obtenidas por la expresión de la vista. Consecuentemente, no es necesario re-evaluar la expresión cada vez que se solicite la vista, sino que se accede directamente a una relación base que ya contiene el resultado. Incluso, algunos autores consideran las vistas materializadas como un almacenamiento *caché* de una expresión, donde el acceso es realmente muy rápido [55]. Además del rápido acceso a los datos, las vistas materializadas también presentan otras ventajas [35]:

- Son transparentes para las aplicaciones y los usuarios.
- Permiten gestionar la obsolescencia de los datos.
- Pueden actualizarse automáticamente cuando cambian los datos del origen.

Esta última ventaja, dentro de las **VM**, constituye un área de investigación denominada mantenimiento de vistas materializadas. El mantenimiento de estas vistas es una tarea fundamental que responde a la necesidad de actualizar la *caché*, luego que la ocurrencia de nuevas transacciones en las relaciones bases desactualizen sus datos. Realizar este mantenimiento desde “cero” es altamente ineficiente debido a que se desecha toda la información previamente obtenida, muy similar al análisis realizado en la Sección 2.4. Las propuestas que toman en consideración solo los cambios realizados a las relaciones bases

son llamadas propuestas de mantenimiento incremental de vistas materializadas. Este mantenimiento incremental, como se apunta de forma genérica en la sección anterior, puede ser inmediato o diferido [34].

Una definición formal de **VM** puede ser establecida mediante la expresión que define dicha vista. Sea una vista definida por una expresión relacional  $Q$  y materializada en  $VM$ . Una correcta materialización de  $VM$  en el estado  $BD_s$  de la base de datos, siempre retorna los mismos datos que la expresión  $Q$ , siendo de esta forma equivalentes:

$$VM(BD_s) = Q(BD_s)$$

$$VM \equiv Q$$

Un importante resultado de materializar una vista es, como se ha mencionado, la velocidad de acceso a sus datos. Si  $tmp_1$  es el tiempo de calcular  $VM$  en el estado  $s$  de la Bases de Datos (**BD**) y  $tmp_2$  el tiempo para calcular similarmente la expresión relacional  $Q$  entonces, en términos generales,  $tmp_1 \ll tmp_2$ . Otro resultado deseable en las vistas materializadas involucra la total independencia de la vista con la exploración a las relaciones base. Cuando una **VM** es mantenida sin el acceso a las relaciones bases se dice que es una vista automantenida [34].

De forma similar a las **VM**, una **Regla de Asociación (RA)** puede ser definida mediante una expresión que establezca su condición de selección en la base de datos. Esta condición de selección a que hacemos referencia es parte del álgebra relacional [47, cap. 6], definida como  $\sigma_{C(G)}(R)$  donde  $C$  es la condición de selección de la expresión booleana en  $G \subseteq R$ . Como se discutió brevemente en la Sección 2.2, las reglas de asociación cuantitativas [119] consideran sus atributos como un par  $\langle \text{atributo}, \text{intervalo} \rangle$  donde el intervalo especifica un valor inferior  $i$  y uno superior  $s$  para el atributo  $A$ :  $\langle i, A, s \rangle$ . Sea  $X \Rightarrow Y$  una **RA**,  $R$  un esquema relacional donde  $X, Y \subseteq R$ ,  $k$  la cantidad de atributos en  $X$  y  $l$  la cantidad de atributos en  $Y$ . Una Condición de Selección (CS) correspondiente a la **RA** puede definirse mediante (3.1).

$$CS = \bigwedge_{p=1}^k i_p \leq A_p^X \leq s_p \wedge \bigwedge_{q=1}^l i_q \leq A_l^Y \leq s_q \quad (3.1)$$

La selección final sobre  $R$ , asociada a la **RA**, utiliza una condición de selección  $CS$  para filtrar aquellas tuplas que satisfagan dicha condición:  $\sigma_{CS}(R)$ . Esta y otras selecciones pueden ser consideradas como vistas y materializarse, a fin de establecer reglas de asociación materializadas.

De igual forma, se pueden tratar las extensiones difusas y aproximadas de las **RA**. En el caso aproximado la definición es directa, teniendo en consideración la cantidad cuadrática de tuplas. Para las reglas de asociación difusas no es necesario filtrar ninguna tupla, dado que todas intervienen en las medidas difusas. Lo fundamental en la selección de estas reglas es la sumatoria de los mínimos por etiqueta lingüística en toda la relación, tal como plantea (3.2). Según estableció la Sección 2.2, sea  $Etq(At_k) = \{E_1^{At_k}, \dots, E_n^{At_k}\}$  el conjunto de etiquetas lingüísticas establecidas para los atributos de la regla.

$$Q = \sum_{t \in r:R} \min_{k \in (XUY)} E_k^{At_k}(t[At_k]) \quad (3.2)$$

Por varias décadas se han estudiado diferentes técnicas y se han propuesto múltiples algoritmos para el mantenimiento incremental de las **VM**. Estos mecanismos están básicamente clasificados según la expresión relacional que involucra la definición de su vista. Para ampliar brevemente en este sentido, solo analice cuántos tipos de consultas diferentes son posibles obtener desde el estándar *emph*Structured Query Language (SQL) [131] y la complejidad de las mismas.

Algunos de estos algoritmos son definidos para vistas que involucran selecciones, proyecciones y acoples del álgebra relacional [14]. Otras vistas que son estudiadas incluyen duplicados, agregaciones, uniones, acoples exteriores, diferencias, subconsultas, recursividad, entre otras [55, 27]. Para la presente memoria es de especial interés el mantenimiento de las vistas agregadas [90, 106, 55].

Las vistas agregadas incluyen el agrupamiento por atributos (*group-by*) y funciones de agregación como el mínimo (*min*), máximo (*max*), cantidad (*count*), promedio (*avg*), suma (*sum*), entre otros. Tómese por ejemplo la relación de la Tabla 3.1 y suponga que desea obtener el total de ventas mensuales del año.

Tabla 3.1: Datos de productos vendidos en un supermercado (*Registro\_Ventas*)

ID_Compra	ID_Producto	Fecha	Precio_Venta
1001	26	12 sep. 2016	10
1002	44	17 oct. 2016	30
1003	32	15 oct. 2016	40
1004	30	12 nov. 2016	100
1005	12	22 nov. 2016	20

Para ello puede definir una vista *total\_ventas\_año* como se presenta a continuación. Mediante la ejecución de esta vista en la relación de la Tabla 3.1, se obtiene una relación mostrada en la Tabla 3.2.

```
CREATE VIEW total_ventas_año AS
SELECT Mes(Fecha) as Mes, sum(Precio_Venta) as Total_Ventas
FROM Registro_Ventas
WHERE Año(Fecha) = 2016
GROUP BY Mes(Fecha)
```

Tabla 3.2: Relación virtual derivada de ejecución de la vista *total\_ventas\_año* en la relación de la Tabla 3.1, junto a cantidad de tuplas derivadas

Mes	Total_Ventas	Cantidad_Tuplas_Derivadas
sep.	10	1
oct.	70	2
nov.	120	2



La relación derivada *total\_ventas\_año* se considera una vista agregada que puede ser materializada y mantenida incrementalmente. En tal caso, su relación base sería Registro\_Ventas y todos los eventos estructurales ocurridos en esta relación se utilizarían para actualizar la materialización de la vista.

Varios algoritmos han sido propuestos para el mantenimiento de vistas materializadas de agregación. Dentro de ellos destacan los algoritmos de conteo [55, 54] y los algoritmos específicos para el mantenimiento de agregaciones [90, 56]. La idea subyacente de estos y otros métodos es precisamente crear la materialización de los atributos agrupados y actualizar eficientemente esta relación para los diferentes eventos estructurales que sucedan en la relación base. En el caso del algoritmo de conteo propuesto por Gupta y colaboradores [54] se almacena además, la cantidad de tuplas en la relación base derivadas de una tupla materializada, tal como muestra la Tabla 3.2. Este atributo extra también permite calcular eficientemente consultas con o sin duplicados. Para las vistas de agregación existe una estrecha correspondencia entre este atributo extra y la función de agregación *count*, dado que esta última también puede almacenar la cantidad de tuplas derivadas de la materialización.

## 3.2 Chequeo de Restricciones de Integridad para las reglas abordadas

Desde el mismo surgimiento del modelo relacional se reflejó la necesidad de limitar los valores presentes en un estado de la base de datos. Estas limitaciones fueron especificadas mediante las Restricciones de Integridad (RI) que, junto a la lista de atributos  $A$  y sus dominios  $D$ , definen un esquema de relación  $R(A:D, RI)$ . Algunas de estas restricciones se establecen implícitamente en el modelo relacional, otras se definen explícitamente mediante un Lenguaje de Definición de Datos (LDD) y algunas simplemente son imposibles de representar desde este modelo [47, cap. 5]. Las RI pueden estar localizadas en diferentes niveles de la base de datos [47, cap. 5]: nivel de Dominio, nivel de Atributo(s), nivel de Relación o nivel de Base de Datos tal como muestra la Tabla 3.3.

Tabla 3.3: Taxonomía de restricciones de integridad en el modelo relacional

Nivel	Restricción
Dominio	Tipos de Datos
Atributo(s)	Not Null
	Unicidad (Llaves)
	Comparación entre Atributos
Relación	Integridad de Entidad
	Dependencia Funcional
Base de Datos	Integridad Referencial
	Integridad Semántica

El término *integridad* se refiere a la exactitud o corrección de los datos en la base de datos. Según Olivé [94], dos componentes que definen la integridad son la validez y la completitud:

$$\text{Integridad} = \text{Validez} + \text{Completitud}$$

Una base de datos se dice que se encuentra íntegra o consistente cuando posee todos los hechos relevantes y éstos son válidos. Es decir, la validez establece que, si una sentencia  $f$  se encuentra en la base de datos  $BD$ , esto implica que la sentencia  $f$  forma parte del universo de discurso  $UD$  ( $BD \rightarrow f \Rightarrow UD \rightarrow f$ ). Por su parte, la completitud especifica que, si el universo de discurso contiene una sentencia, entonces esta sentencia se encuentra también en la base de datos ( $UD \rightarrow f \Rightarrow BD \rightarrow f$ ).

La falta de integridad en una base de datos normalmente trae consecuencias negativas. Por lo tanto, es imprescindible garantizar que cada transición de la base de datos sea consistente con todas las **RI** definidas. Esto se puede garantizar verificando que cada evento estructural no viole ninguna **RI**. Sea  $CRI$  un conjunto de restricciones de integridad en forma de negación (forma similar a las reglas deductivas [89]) y  $F$  una función que evalúa si una **RI** es violada en el estado  $s$  de la base de datos  $BD$ . Un estado consistente de  $BD$  garantiza que todas las restricciones de integridad en  $CRI$  evaluadas con  $F$  son falsas.

$$\forall ri \in CRI ( F(ri, BD_s) = false )$$

Este proceso, conocido como *chequeo de restricciones de integridad* o simplemente *chequeo de integridad* debe ser desarrollado tan eficientemente como sea posible. Realizar este chequeo “desde cero” es, como se ha venido reiterando hasta el momento, altamente ineficiente ¿Será necesario para los nuevos eventos estructurales volver a chequear toda la **BD**? Como se apuntó en la Figura 2.6, el grueso de la **BD** permanece invariable a los cambios sucedidos y continúa cumpliendo gran parte de las **RI** establecidas. Las propuestas que toman únicamente en consideración los cambios realizados son llamadas propuestas de *chequeo incremental de restricciones de integridad*.

Las **RI** y las **VM** tienen un estrecho vínculo entre sus campos de estudio. Varias propuestas para el chequeo de restricciones de integridad establecen una condición de selección como las **VM** [55]. La particularidad de esta perspectiva es que dicha vista debe permanecer siempre vacía, indicando que la base de datos no presenta ninguna inconsistencia. Posteriormente, es posible aplicar las técnicas conocidas en el mantenimiento de vistas materializadas presentadas en la Sección 3.1.

A diferencia de las **RI**, las reglas de asociación junto a sus extensiones difusas y aproximadas no prohíben estados de la base de datos. Ellas solamente expresan un conocimiento sobre los datos que eventualmente, si no es mantenido, se convierte en obsoleto. Sin embargo, existe una importante restricción en la propia definición de las **RA**: el umbral de la medida. Como anteriormente comentamos, las medidas expresan el nivel de interés en la regla, además de establecer un valor mínimo aceptable que define su existencia. Sea  $CR$  un conjunto de reglas extraídas del estado  $s$  de la base de datos  $BD$ ,  $F$  una función que evalúa la medida de una regla y  $\delta$  el umbral mínimo para dicha medida. Un estado correcto del conjunto de reglas  $CR$  establece que, para todas las reglas evaluadas con  $F$ , el valor de ésta debe ser mayor o igual que  $\delta$ :

$$\forall r \in CR ( F(r, BD_s) \geq \delta )$$

Desde esta perspectiva, las técnicas de chequeo de restricciones de integridad pueden contribuir a mantener este umbral para todas las reglas definidas. Esto equivale directamente a colaborar en una eficiente evaluación de  $F$ .

Las restricciones de integridad son un campo de investigación ampliamente estudiado, tanto desde un nivel lógico [47, cap. 5] [53] como más recientemente desde un nivel conceptual [23, 98] mediante UML\OCL (*Unified Modeling Language\Object Constraint Language*) [96, 95]. Múltiples propuestas se encaminan a mantener incrementalmente el chequeo de estas restricciones, considerando únicamente los cambios ocurridos en la base de datos y detectando, dentro de estos cambios, cuáles potencialmente pueden afectar a las restricciones establecidas [43, 23, 22, 21, 98]. Por ejemplo, si la siguiente restricción en forma de negación es establecida en el esquema de relación de la Tabla 3.1: “*el precio de venta de un producto no puede ser menor que 10*” ¿tiene sentido verificar dicha restricción cuando se modifica otro atributo? Y aún más, ¿tiene sentido verificar dicha restricción cuando se aumenta el valor del atributo Precio\_Venta? Esta rama del mantenimiento eficiente de las restricciones de integridad se enfoca en identificar aquellas *instancias relevantes* que pueden violar una restricción [22].

En la propuesta realizada por Cabot y Teniente [22] para calcular las instancias relevantes se utilizan *relaciones auxiliares*. Estas relaciones almacenan los nuevos eventos estructurales (incrementos) ocurridos en la base de datos que, dado el contexto conceptual de dicha investigación, son denominadas tipos de entidades auxiliares. Estas estructuras auxiliares se definen sobre los tipos de entidades involucradas en la restricción y solo guardan una referencia a la entidad afectada por el evento. Por lo tanto, para cada tipo de entidad en el contexto de una restricción, se definen tres tipos de entidades auxiliares correspondientes a los eventos de inserción, actualización y eliminación.

La utilización de relaciones auxiliares permite elaborar eficientes propuestas diferidas en el chequeo de RI, dado que todos los nuevos eventos pueden quedar registrados en estas estructuras intermedias. Posteriormente, estas estructuras permiten validar las RI de forma diferida. Generalmente las relaciones auxiliares disminuyen los tiempos de acceso a la base de datos, puesto que filtran los datos

necesarios para lograr un objetivo determinado. Es por ello que el empleo de relaciones auxiliares no es exclusivo al chequeo de **RI**, también ha sido propuesto para el mantenimiento incremental de **VM** [107, 128].

El manejo de varios eventos estructurales mediante relaciones auxiliares impone otro aspecto a tomar en consideración. Digamos, por ejemplo, que un evento estructural compuesto inserta una tupla y posteriormente, el mismo **Evento Estructural Compuesto (EEC)** la elimina ¿Esta tupla interviene de algún modo en las **RI** o en la base de datos de forma general? O, por ejemplo, si una misma tupla es actualizada y luego eliminada, ¿tiene la actualización realizada algún efecto? Estas interacciones entre las operaciones típicamente no son tenidas en cuenta por las técnicas de minería incremental vistas en la Sección 2.4. Sin embargo, son ampliamente utilizadas en el chequeo de **RI** [21, 26] y el mantenimiento de vistas materializadas [14], donde se conocen bajo el término *net effect* (efectos de red).

Estas políticas de efecto de red, si bien añaden un procesamiento extra al tiempo de ejecución de las transacciones, sin dudas pueden reducir drásticamente la cantidad de eventos estructurales a considerar. En la propuesta realizada por Ceri y Widom en el año 1990 [26] se proponen cuatro políticas de efecto de red:

- Si una tupla es varias veces actualizada, solamente se considera la actualización compuesta.
- Si una tupla es actualizada y luego eliminada, solamente se considera la actualización.
- Si una tupla es insertada y luego actualizada, se considera la inserción actualizada.
- Si una tupla es insertada y luego eliminada, no se considera.

### 3.3 Expresividad de las Reglas de Negocio

Las **Reglas de Negocio (RN)** según Ross, son sentencias que definen o restringen algún aspecto del negocio y que se encuentran bajo su jurisdicción, con lo cual pueden ser creadas, revisadas y eliminadas cuando el negocio lo estime conveniente [109]. Esta sentencia debe ser una declaración compacta, utilizando un lenguaje simple, inequívoco y accesible a todas las partes interesadas: dueño del negocio, analista, arquitecto, técnico, etc.

De esta forma, las **RI** abordadas en la Sección 3.2 forman parte de las reglas de negocio como componentes claves en los esquemas conceptuales y lógicos de las bases de datos. No obstante, las reglas de negocio son mucho más abarcadoras y complejas, considerando perspectivas de los sistemas de información, del negocio o de ambas [135]. La presente memoria se enfoca en las **RN** que actúan directamente en el comportamiento de los datos, las cuales se encuentran en los repositorios centrales de información del negocio [8]. La Figura 3.1 muestra la relación entre **RN**, **RN** asociadas a los datos y **RI**.

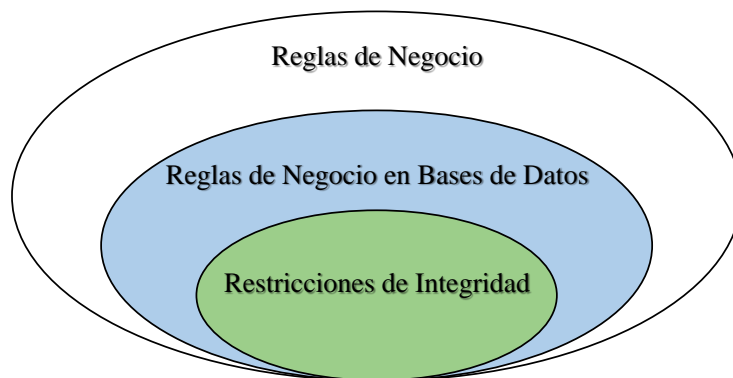


Figura 3.1: Relación entre reglas de negocio, reglas de negocio en bases de datos y restricciones de integridad.

Dentro las reglas de negocio que involucran los datos se encuentran una amplia gama de categorías propuestas por varios autores [25, 57, 91, 110, 5, 4]. A continuación, se presentan algunas de las más significativas en la literatura especializada junto a su definición, puesto que su denominación puede variar de un autor a otro.

- **RN de restricción** [25]. Exactamente las restricciones de integridad abordadas en la Sección 3.2, expresadas como reglas de negocio.
- **RN de cómputo o cálculo** [91, 110]. Estas reglas son declaraciones completas que proveen operaciones para lograr el valor de un término. El resultado del cómputo puede ser, por ejemplo, el valor para un atributo.
- **RN de definición o clasificación** [5, 110]. Reglas que permiten organizar el conocimiento básico del negocio, contribuyendo a especificar el significado de los conceptos.
- **RN de notificación** [4]. Reglas que informan a los usuarios autorizados del negocio sobre algún conocimiento básico.

Las **RN**, independientemente de su tipo, pueden ser expresadas en tres niveles según la audiencia que las necesite [91]. El primer nivel de las **RN** es el *informal*, donde la regla de negocio es expresada tal como el usuario del negocio necesita en una conversación informal. El segundo nivel es el técnico, nivel intermedio con forma matemática que se auxilia de datos estructurados y operadores matemáticos. Y el tercer nivel es el formal, con una sintaxis definida que le proporciona el potencial para ser ejecutada, brindándole la funcionalidad completa a las **RN**.

El nivel técnico ha sido ampliamente abordado en múltiples investigaciones como un nivel apto para los desarrolladores de sistemas, y también para usuarios avanzados de los mismos [25, 91, 22, 21]. Dentro de estos lenguajes destaca OCL [95], desde el cual múltiples autores han propuesto transformaciones automáticas para SQL, brindando su implementación para las bases de datos relacionales del negocio [97, 98].

Entre los componentes fundamentales para trabajar con reglas de negocio se encuentra el repositorio de reglas. El repositorio de reglas es utilizado para almacenar las declaraciones de las reglas en sus diferentes niveles de expresión. Además, incluye todos aquellos aspectos necesarios para garantizar el control y el mantenimiento de las mismas. Mediante este repositorio se logra un grado de independencia de las reglas con los sistemas de información.

Si bien las reglas de asociación y sus extensiones aproximadas y difusas tienen una forma de expresión bien definida, éstas se encuentran expuestas directamente en los datos. Trabajos recientes utilizan las reglas de negocio para brindar una expresividad adicional a las reglas de asociación, acercándolas al usuario final [70, 125]. Resultan interesantes las potencialidades de las RN tipo clasificación y tipo cálculo en las reglas de asociación difusas, donde existe la necesidad de utilizar etiquetas lingüísticas y funciones matemáticas.

Las RN de restricción también han realizado aportes a la implementación en bases de datos relacionales. En [25] se utilizan varios recursos de BDA para el mantenimiento eficiente de estas RN, y se define un lenguaje técnico para expresarlas.

### 3.4 Recursos activos en los sistemas relacionales

La implementación del mantenimiento de vistas materializadas, chequeo de restricciones de integridad y reglas de negocio está estrechamente relacionada con las capacidades que sean provistas para tal efecto. A estas capacidades que provee un **Sistema Gestor de Bases de Datos Relacionales Activas (SGBDRA)** se les denomina recursos, y si estos recursos están destinados para la definición o el manejo de reglas **Evento Condición Acción (ECA)**, se les denomina recursos activos.

La mayoría de los recursos o mecanismos activos definidos en los sistemas comerciales de bases de datos relacionales difieren entre sí. Con la perspectiva de proveer una base consistente de estos mecanismos en los sistemas relacionales, se presentan estos recursos desde su estándar. El estándar *Structured Query Language* (SQL) es un lenguaje de programación de propósito especial. Según el *American National Standards Institute* (ANSI) y la *International Organization for Standardization* (ISO) con su afiliación a la *International Electrotechnical Commission* (IEC), el SQL es el lenguaje estándar para las bases de datos relacionales.



El lenguaje SQL fue originalmente desarrollado por IBM bajo el nombre de *Structured English QUery Language* (SEQUEL) [28]. Inicialmente, fue diseñado para manipular y recuperar los datos de su gestor de base de datos original: *System R*. Desde su primera versión formalizada por la ANSI en el año 1986, el SQL ha sido revisado en ocho ocasiones. El presente documento se remite a su versión actual, el SQL:2011 (SQL 2011) [6].

El estándar SQL define varios recursos que permiten reaccionar a determinados eventos y realizar acciones, tal como el modelo ECA establece. En la presente memoria se presentan tres de estos recursos: restricciones de chequeo, aserciones y disparadores.

### 3.4.1 Restricciones de Chequeo

Las restricciones de chequeo (*check constraint*) permiten definir una condición de búsqueda (*search condition*) con el objetivo de garantizar la integridad de los datos. Dicha condición debe ser booleana y en caso de resultar falsa, la restricción es violada y automáticamente no aceptada (acción implícita); si el resultado es desconocido (*null*) no se viola [6]. Su sintaxis en el estándar SQL es:

```
<check constraint definition> ::=  
CHECK <left paren><search condition><right paren>
```

Las restricciones de chequeo pueden definirse para restricciones de dominio, columna o tabla. Las restricciones de columna son definidas únicamente en la creación de las tablas y deben ser para una única columna. Las restricciones de tabla por otro lado, pueden ser establecidas en las modificaciones de las tablas e incluir varias columnas. Las restricciones de dominio deben ser expresadas al crear o modificar un dominio y su condición de búsqueda debe contener una especificación de valor. Un ejemplo de una restricción de chequeo, en la definición de una columna, puede ser que el valor de esa columna sea mayor que una constante:

```
CREATE TABLE Libros (  
    nro_páginas SMALLINT CHECK ( páginas > 0 ));
```

La condición de búsqueda de una restricción de chequeo se encuentra restringida por varias pautas. En general, no pueden invocar rutinas no deterministas o rutinas que directa o indirectamente modifiquen los datos. Tampoco pueden acceder a otras tablas si pertenecen a tablas persistentes. De hecho, sin una característica muy específica, las restricciones de chequeo no pueden tener una expresión de consulta [6]. Esta característica es omitida con toda intención por la mayoría de los gestores relacionales actuales.

### 3.4.2 Aserciones

El estándar SQL define aserciones (*assertions*) para las tablas. Una aserción es un tipo de restricción especial que permite establecer limitaciones para una tupla, para varias tuplas de una única tabla o para un estado de la base de datos que involucra a más de una tabla [6]. Por ello, su definición no está asociada a una tabla específica como muestra la sintaxis:

```
<assertion definition> ::=  
    CREATE ASSERTION <constraint name>  
    CHECK <left paren><search condition><right paren>  
    [<constraint characteristics>]
```

A esta característica de poder especificar una restricción para toda la base de datos y no para una tabla en particular, se le llama restricción autónoma. Esto tiene grandes ventajas a nivel conceptual, pero las hace difíciles de implementar. Lo cual es la causa principal de que la mayoría de los **SGBDRA** no implementan las aserciones.

A diferencia del chequeo de restricciones, una aserción es satisfecha solo si su condición de búsqueda no es falsa, siendo su acción implícita la de no aceptar el nuevo estado. Las características de la restricción (*constraint characteristics*) especifican el momento en el que debe ser chequeada. Este momento puede definirse *deferrable*, con lo cual se permite una violación temporal de la restricción.

Otras opciones son *not deferrable*, *initially deferred* e *initially immediate*, según la sintaxis de las características. Un ejemplo sencillo de una aserción es establecer que una tabla no puede quedar vacía:

```
CREATE ASSERTION algún_Libro
  CHECK (0 <>(SELECT COUNT(*) FROM Libros));
```

### 3.4.3 Disparadores

Un disparador (*trigger*) es una especificación que debe llevarse a cabo dado un evento en un objeto. Este objeto es una tabla persistente de la base de datos conocida como tabla sujeto [6]. Los disparadores amplían la lógica de comprobación de integridad, valores predeterminados y reglas del estándar SQL.

Los disparadores permiten ajustar completamente el modelo ECA a conveniencia del usuario, lo que lo diferencia de las restricciones de chequeo o las aserciones que poseen acciones o eventos implícitos. La acción de un disparador es una sentencia procedural de SQL o una lista de dichas sentencias, en las cuales puede estar contenida la propia condición. El evento puede ser una inserción, eliminación o modificación de un conjunto de tuplas. Una sintaxis detallada de los disparadores en SQL se puede encontrar en el Apéndice A. Seguidamente, se muestra su definición básica:

```
<trigger definition> ::=
  CREATE TRIGGER <trigger name>
    <trigger action time><trigger event>
  ON <table name>
  [REFERENCING<transition table or variable list>]
  <triggered action>
```

Los componentes principales de los disparadores son:

- **Evento disparador.** Provocado por una inserción (*insert*), eliminación (*delete*) o actualización (*update*) en la tabla sujeto.

- **Tiempo de activación.** Especifica si se activa antes (*before*), después (*after*) o en lugar de (*instead of*) el evento.
- **Granularidad de la acción.** Define el campo de la acción. Puede ser para cada tupla (*for each row*) o para cada sentencia (*for each statement*).
- **Cuerpo de la acción.** Define la acción en sí misma que se ejecuta cuando el disparador es activado.
- **Referencia.** Define una lista de alias para la(s) tupla(s) nueva(s) (*new*) o antigua(s) (*old*).

Los alias definidos en la referencia permiten acceder a los valores existentes en la tabla sujeto antes de una actualización/inserción, o después de una actualización/eliminación. La siguiente Tabla 3.4 resume qué valores son recuperados por el nuevo (*new*) o antiguo (*old*) alias, luego de varios eventos.

Tabla 3.4: Sumario de referencias a los alias según el evento desencadenador

<b>Evento disparador</b>	<b>old.columna1</b>	<b>new.columna1</b>
Inserción	ningún valor (error)	valor insertado
Actualización (columna1 actualizada)	valor original	valor actualizado (puede ser modificado en la acción)
Actualización (columna1 no actualizada)	valor original	valor original (no puede ser modificado en la acción)
Eliminación	valor original	ningún valor (error)

Un ejemplo sencillo en la utilización de disparadores puede ser el registro de todas las eliminaciones realizadas a una tabla:

```
CREATE TRIGGER Libros_eliminados
  AFTER DELETE ON Libros
  REFERENCING OLD ROW AS old
  FOR EACH ROW
  INSERT INTO Registro_Libros_Eliminados
  VALUES (old.título);
```

Los disparadores son ampliamente soportados por la mayoría de los actuales sistemas gestores de bases de datos relacionales. Algunos de estos sistemas han implementado sus propias versiones con ligeras variaciones del estándar, lo cual ha provocado desajustes entre sí. Algunas de estas variaciones son causa de problemas para los desarrolladores de aplicaciones.

### 3.5 Resumen

A lo largo de este capítulo, se observa cómo las bases de datos establecen un comportamiento activo en las áreas de Vistas Materializadas (**VM**), Restricciones de Integridad (**RI**) y **Reglas de Negocio (RN)**. En cada una de estas áreas se pueden resumir algunos puntos imprescindibles para la presente investigación:

- **Vistas Materializadas.** Proponen, a través de su mantenimiento incremental, algoritmos específicos según la forma de la condición de selección que define la vista. Dentro de estas consultas se destacan las agregaciones con algoritmos de conteo y algoritmos específicos para agregaciones.
- **Restricciones de Integridad.** Proponen, mediante su chequeo incremental, estructuras auxiliares para el manejo diferido de los eventos estructurales ocurridos en la base de datos. Además, proponen políticas de efecto de red, útiles para el manejo eficiente de eventos estructurales compuestos.
- **Reglas de Negocio.** Proponen la capacidad de reflejar el conocimiento en niveles muy cercanos al usuario del negocio. En especial, el conocimiento que se encuentra relacionado con los datos.

Para cada una de estas áreas del conocimiento se establecieron consideraciones acerca de las RA junto a sus extensiones difusas y aproximadas. Esto permite abordar los algoritmos y técnicas del cómputo de los cambios en el mantenimiento incremental de las RA y sus extensiones, desde una perspectiva totalmente novedosa. Por último, se presentaron tres recursos de BDA que posibilitan la especificación de estos algoritmos y su implementación en sistemas reales.



# Capítulo 4

## Nuevos algoritmos para el mantenimiento incremental de reglas previamente descubiertas

Este capítulo puede perfectamente considerarse el núcleo de la presente memoria. A lo largo de la misma, se han esbozado los conceptos más relevantes sobre las reglas de asociación y sus extensiones difusas y aproximadas. De igual forma, se ha mostrado que en los entornos reales las bases de datos no permanecen estáticas, sino que constantemente nuevas transacciones son realizadas con un impacto directo en el conocimiento previamente extraído.

Por otro lado, también se ha detallado la capacidad reactiva de las bases de datos ante las nuevas transacciones, mostrando numerosos estudios que mantienen diferentes estructuras asociadas a los datos de una forma incremental. En este aspecto, se han establecido vínculos entre estos estudios y las reglas abordadas en la presente memoria.

Partiendo de los preceptos expuestos, el presente capítulo está en disposición de ofrecer nuestras propuestas para el mantenimiento incremental de reglas previamente descubiertas en Bases de Datos Activas (BDA). El capítulo inicialmente describe nuestro problema de investigación y sus diferencias con



otros estudios, además de mostrar algunas aplicaciones donde se evidencian. A continuación, se establece cómo utilizar las medidas de reglas en la propuesta realizada. Posteriormente, se procede a mostrar una solución inicial de nuestro problema, seguido de los dos algoritmos que se elaboran en la presente memoria: uno inmediato y otro diferido.

Finalmente, se realiza un estudio experimental de los algoritmos propuestos. Para ello se detallará el diseño de los experimentos realizados, los conjuntos de datos utilizados y los resultados de análisis en diferentes entornos.

## 4.1 Propuesta de mantenimiento incremental directo

Como se ha establecido en la Sección 2.1, las medidas establecidas sobre el conocimiento extraído juegan un importante rol en la minería de reglas de asociación, dependencias aproximadas, reglas de asociación difusas y en general en la minería de datos. Sin embargo, en el contexto de las bases de datos estas medidas no son estáticas. Las constantes transacciones ocurridas en las Bases de Datos (BD) modifican los datos y como consecuencia, el conocimiento previamente extraído. Por lo tanto, la tarea de hallar las nuevas medidas del conocimiento en la base de datos actualizada, constituye un objetivo de primordial importancia.

Considere el estado  $i$  de la base de datos  $BD$  ( $BD_i$ ), y un evento estructural compuesto sobre este estado inicial que produce la transición de la base de datos  $BD$  a un estado final  $f$ , ( $BD_i, BD_f$ ). Sea  $CR_i = \{(R_1, MR_1^i), (R_2, MR_2^i), \dots, (R_n, MR_n^i)\}$  un conjunto de reglas extraídas junto al valor de su medida en el estado  $BD_i$ . Nuestro objetivo consiste en hallar las nuevas medidas de las reglas descubiertas en el estado final de la base de datos  $BD_f$ . En otras palabras, el problema del mantenimiento de las reglas previamente descubiertas se reduce a hallar el conjunto de reglas  $CR_f = \{(R_1, MR_1^f), (R_2, MR_2^f), \dots, (R_n, MR_n^f)\}$ .

Como se puede apreciar, nuestro objetivo difiere completamente a los propósitos de la minería incremental presentados en la Sección 2.4. Los esfuerzos de nuestras propuestas están encaminados a mantener las reglas directamente en su

forma final y no en una forma intermedia (ítemsets) para posteriores procesos de minería. Esta independencia lo convierte en un objetivo muy general y flexible que puede ser utilizado para diferentes situaciones. Por lo tanto, nuestras propuestas no están encaminadas a descubrir nuevo conocimiento, la intención principal es mantener el conocimiento previamente obtenido de forma eficiente.

Se puede resumir brevemente que los algoritmos de minería incremental buscan descubrir o eliminar los ítemsets de la forma más eficiente posible. Para ello, analizan únicamente los incrementos ocurridos a la base de datos y determinan qué ítems dejan de cumplir el soporte establecido, tratando de evitar tanto como sea posible, la exploración a la base de datos. Esto trae como consecuencia que se mantenga, en tiempo real, un control muy eficiente del soporte de los ítemsets y que se pueda realizar posteriormente una minería incremental superior a la minería “desde cero”. Es posible considerar que la minería incremental en este sentido realiza un mantenimiento incremental indirecto de las reglas. Sin embargo, las estructuras utilizadas por los algoritmos incrementales no han sido concebidas para evaluar las medidas en tiempo real. En tal caso, para actualizar las medidas exactas de las reglas, es necesario esperar a ejecutar los algoritmos sobre estas estructuras intermedias. La Figura 4.1 muestra la obtención de las medidas exactas mediante los algoritmos convencionales y los algoritmos incrementales. Desde una perspectiva diferente, nuestra investigación propone mantener directamente las medidas mediante su actualización en la base de reglas, tal como ilustra la Figura 4.2.

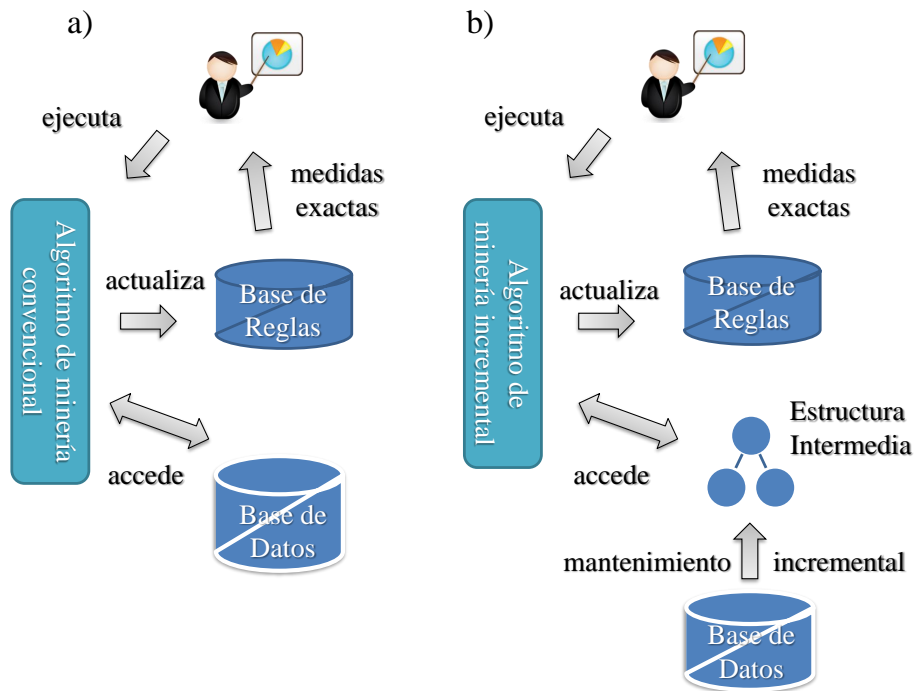


Figura 4.1: Métodos de minería convencional a) y minería incremental b) para la obtención de medidas exactas en tiempo real.

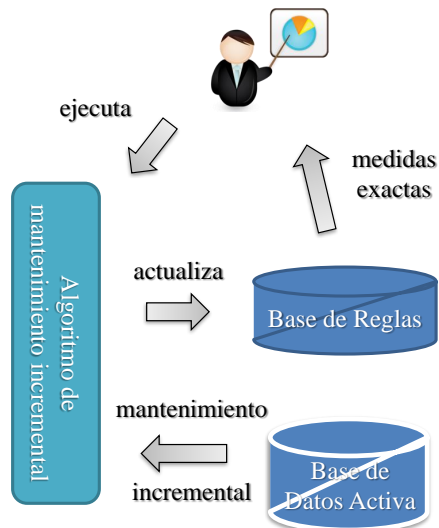


Figura 4.2: Propuesta incremental para la obtención de medidas exactas en tiempo real.

La obtención de las medidas exactas en tiempo real es una ventaja que puede ser utilizada por varias aplicaciones. En especial, destacan aquellas que mantienen un flujo de datos constante, a gran velocidad y que necesitan tomar decisiones a partir del conocimiento previamente extraído tan rápido como sea posible. En este caso se encuentran los sistemas que realizan análisis de redes sensoriales, tráfico de red, actividad de la web, registro de llamadas telefónicas y redes sociales [85, 84]. La toma de decisiones en estas aplicaciones usualmente se encuentra relacionada con el ranqueo de las reglas [37, 16, 122, 49]. Es decir, una regla es más interesante que otra para una decisión si su medida es mayor. Este orden, en el contexto de las aplicaciones con cambios rápidos, puede ser sensible y variante. Estos aspectos son fundamentales en los sistemas de apoyo a la toma de decisiones (*Decision Support Systems*, DSS) [113] y su objetivo de siempre proveer la mejor información. Esta información actualizada (*up-to-date*) es vital en los DSS de tiempo real (*Real Time DSS*, RTDSS) donde los sistemas convencionales no proveen la latencia necesaria para la toma de decisiones en entornos de cambios rápidos [42].

Otras áreas de investigación también toman en cuenta los cambios en las medidas de las reglas. Un ejemplo es la etapa de evaluación o interpretación en el proceso de KDD donde, en una etapa posterior a la minería (conocida como *post mining*), se analizan los cambios en las reglas descubiertas a través del tiempo. Estos estudios buscan detectar cambios en las medidas que expresen una evolución del conocimiento, como la estabilidad, cambios no homogéneos y cambios rápidos [15]. Para ello se hace necesario contar con un historial de las reglas (*rules' history*). Otro ejemplo puede ser hallado en los retos impuestos por las actuales investigaciones en *Big Data* [127], donde nuestra propuesta puede encontrar aplicaciones directas en temas relacionados con el volumen y la velocidad.

## 4.2 Mantenimiento incremental de las medidas

Como se apuntó en la Sección 2.1, la cantidad de medidas existentes en la literatura para evaluar el interés en las reglas de asociación es abrumadora. Incluso, esta cantidad puede verse ampliada si se considera la definición de nuevas medidas

a partir de medidas ya existentes [121]. Esta variedad provee a las aplicaciones reales de un gran banco de medidas a utilizar según las particularidades del sistema y sus datos. En aras de proponer un mantenimiento incremental para estos sistemas, es necesario proveer soluciones generales capaces de ser inclusivas con respecto a esta variedad de medidas existentes e incluso, de medidas que puedan ser definidas en el futuro. Dichas soluciones deben realizarse, por supuesto, de forma controlada y eficiente.

Asimismo, es importante considerar la definición de múltiples medidas para las reglas. En la práctica, es necesario mantener más de una medida de interés sobre las reglas para reducir su cantidad y mejorar su calidad [121, 75, 71]. Desde la propia definición de algunas medidas se puede observar que existen estrechos vínculos con otras medidas. Tal es el caso por ejemplo del factor de certeza (2.4), el cual utiliza en su definición la confianza y el soporte.

En nuestras propuestas [104], el mantenimiento de cada regla se realiza a través del mantenimiento directo de su(s) medida(s). Para ello cada **Medida de una Regla (MR)** es considerada como un conjunto de  $k$  partes diferentes:

$$MR_n = \{Mp_n^1, Mp_n^2, \dots, Mp_n^k\}$$

Cada parte de una medida representa una porción de la fórmula que define la métrica. Estas partes resultantes deben cumplir una serie de características deseables para la correcta evaluación posterior de la regla y su interpretación:

- **Atómicas.** No pueden ser descompuestas en partes más pequeñas que continúen evaluando correctamente la medida.
- **No ambiguas.** Tienen solamente una obvia interpretación dentro de la fórmula de la medida.
- **Compactas.** Típicamente son partes pequeñas.
- **Compatibles.** Utilizan las mismas condiciones en el resto de las medidas definidas.

Por ejemplo, la confianza de una regla  $X \Rightarrow Y$  puede ser dividida, según las consideraciones anteriores, en dos partes: cantidad de registros que cumplen el antecedente ( $|X|$ ) y cantidad de registros que cumplen el antecedente unión consecuente ( $|X \cup Y|$ ). Por otro lado, la métrica *information gain* [32] se divide en cuatro partes: cantidad de registros que cumplen el antecedente ( $|X|$ ), cantidad de registros que cumplen el consecuente ( $|Y|$ ), cantidad de registros que cumplen el antecedente unión consecuente ( $|X \cup Y|$ ) y cantidad total de registros ( $|T|$ ). Se muestra a continuación, en la Tabla 4.1, la definición de las partes necesarias para las medidas presentadas en el Capítulo 2.

Tabla 4.1: Definición de las partes necesarias para el mantenimiento de varias medidas

Medida	Definición	Partes de la Medida
<i>Support</i>	$\frac{n_{XY}}{n}$	$\{  X \cup Y ,  T  \}$
<i>Confidence</i>	$\frac{n_{XY}}{n_X}$	$\{  X ,  X \cup Y  \}$
<i>Information Gain</i>	$\log\left(\frac{nn_{XY}}{n_X n_Y}\right)$	$\{  X ,  Y ,  X \cup Y ,  T  \}$
<i>Bayes Factor</i>	$\frac{n_{XY} n_{\bar{Y}}}{n_Y n_{X\bar{Y}}}$	$\{  Y ,  \bar{Y} ,  X \cup Y ,  X \cup \bar{Y}  \}$
<i>Conviction</i>	$\frac{n_X n_{\bar{Y}}}{nn_{X\bar{Y}}}$	$\{  X ,  \bar{Y} ,  X \cup \bar{Y} ,  T  \}$
<i>Least Contradiction</i>	$\frac{n_{XY} - n_{X\bar{Y}}}{n_Y}$	$\{  Y ,  X \cup Y ,  X \cup \bar{Y}  \}$
<i>Laplace</i>	$\frac{n_{XY} + 1}{n_X + 2}$	$\{  X ,  X \cup Y  \}$
<i>Lift</i>	$\frac{nn_{XY}}{n_X n_Y}$	$\{  X ,  Y ,  X \cup Y ,  T  \}$
<i>Zhang</i>	$\frac{nn_{XY} - n_X n_Y}{\max\{n_{XY} n_{\bar{Y}}, n_Y n_{X\bar{Y}}\}}$	$\{  X ,  Y ,  \bar{Y} ,  X \cup Y ,  X \cup \bar{Y} ,  T  \}$

La cantidad de métricas diferentes que nuestra propuesta puede utilizar es amplia y puede ser expandida a métricas futuras. Básicamente, puede incluir aquellos tipos de métricas de reglas  $X \Rightarrow Y$  que se basen en:

- $|X|$ , el número total de registros que satisfacen  $X$ .
- $|Y|$ , el número total de registros que satisfacen  $Y$ .
- $|\neg X|$ , el número total de registros que no satisfacen  $X$ .
- $|\neg Y|$ , el número total de registros que no satisfacen  $Y$ .
- Cualquier unión combinada de los anteriores elementos.
- $|T|$  el número total de registros.

Esto cubre una gran cantidad de métricas basadas en el antecedente y el consecuente [71], donde además se incluyen las negaciones de los mismos para mantener una mayor expresividad. En este punto, es necesario especificar que la Tabla 4.1 puede ser afectada por la posibilidad de expresar las negaciones dentro de las métricas mediante expresiones no negativas, tal como muestra [71]:

- $|\neg X| = |T| - |X|$ .
- $|\neg Y| = |T| - |Y|$ .
- $|\neg X \cup Y| = |Y| - |X \cup Y|$ .
- $|X \cup \neg Y| = |X| - |X \cup Y|$ .
- $|\neg X \cup \neg Y| = |T| - |X| - |Y| + |X \cup Y|$ .

Es sencillo comprobar desde la propia Tabla 4.1 que varias métricas comparten algunas partes e incluso pueden incluirlas completamente. Por lo tanto, un resultado importante de la división propuesta es que se pueden mantener varias medidas simultáneamente, de forma eficiente. Este resultado es especialmente apreciable en nuestro objetivo de integrar varias medidas de interés. La evaluación final de las medidas se realiza a partir de las operaciones sobre estas partes, según establezca su fórmula correspondiente.

## 4.3 Enfoque Simplificado para el mantenimiento de las reglas

Una solución Simplificada (conocida comúnmente por su término anglosajón, *naïve*) al problema del mantenimiento incremental de las reglas, tal como se ha planteado, puede ser mediante la utilización de consultas en la base de datos. Es decir, una primera idea puede ser sencillamente realizar una consulta de selección, siguiendo (3.1), para cada parte de una regla:

$$Mp^1 = CSP^1, Mp^2 = CSP^2, \dots, Mp^k = CSP^k \quad MR_n = \{CSP_n^1, CSP_n^2, \dots, CSP_n^k\}$$

De esta forma, los cambios ocurridos en las medidas por los eventos estructurales son obtenidos a partir de la exploración completa de todos los datos. Este enfoque, aunque eficaz, dista mucho de ser eficiente. Véase que la información previa de las partes de las medidas es simplemente desechada. Para cada transición de la base de datos es necesario ejecutar estas consultas “desde cero”, convirtiendo el enfoque en costoso y poco viable. Quizás sus únicos atractivos sean su fácil implementación, no requerir de recursos activos por parte de la **BD** y no estar vinculada a eventos estructurales. Sin embargo, esto es irrelevante cuando los datos almacenados son extensos y las exploraciones a la base de datos se convierten en altamente ineficientes.

Este enfoque puede ser interpretado sencillamente como un conjunto de vistas de la base de datos, donde se extrae la información exacta de todas las reglas definidas. Estas consultas de selección que conforman las vistas presentan ciertas diferencias según el tipo de la regla. Para las Reglas de Asociación (**RA**) son necesarias consultas de cuenta (*count*), representando qué cantidad de tuplas cumplen con la condición impuesta por la parte de la medida.

Las consultas para Reglas de Asociación Difusas (**RAD**) son similares a las **RA**. La diferencia radica en que las etiquetas lingüísticas siempre tienen un grado de pertenencia para cada valor del atributo. De esta forma, es necesario sumar todos estos grados de pertenencia según la función definida mediante consultas de suma (*sum*).



En el caso de **Dependencia Aproximada (DA)** las consultas tienen un mayor grado de complejidad puesto que deben realizarse mediante agrupaciones (*group by*). Esto responde a la necesidad de realizar un cálculo eficiente de las medidas. En la Sección 2.3 comentamos que la definición de las DA mediante reglas de asociación se realizaba en  $n^2$  tuplas. Sin embargo, es posible explorar las  $n$  tuplas de la base de datos en vez de las  $n^2$  tuplas correspondientes. Esta solución proviene de dos fuentes diferentes que igualmente reducen la complejidad. La primera es brindada desde la propia definición de DA mediante RA en [112], donde cada tupla  $t_i$  representa explorar  $2i-1$  tuplas. La segunda fuente es la propia definición de las consultas agrupadas, en el cual se obtienen aquellos grupos con igual valor en sus atributos. De esta forma cada grupo representa una cantidad de  $n^2$  tuplas.

Esta diferencia entre consultas de cada tipo de regla hace necesaria la definición de diferentes tipos de estructuras. Para las consultas de una RA que involucra contar tuplas en la base de datos, se establece una lista de enteros a la cual llamamos **Medida de una Regla de Asociación (MRA)**. Las partes de esta MRA toman valores enteros positivos,  $Mp_1, Mp_2, \dots, Mp_k \in \mathbb{Z}^+$ . Muy similar a la MRA es la estructura necesaria para una RAD, la cual llamamos **Medida de una Regla de Asociación Difusa (MRAD)**. Como única diferencia podemos exponer que las partes de la MRAD toma valores reales, resultado de la suma de etiquetas lingüísticas,  $Mp_1, Mp_2, \dots, Mp_k \in \mathbb{R}^+$ .

Para las DA es necesario considerar una estructura más compleja debido a la utilización de grupos. A esta estructura la llamamos **Medida de una Dependencia Aproximada (MDA)**. Básicamente, cada consulta de agrupación es materializada en una vista, según la propuesta de mantenimiento incremental de vistas presentada en [106]. De esta forma,  $\forall Mp_k \in MDA$  se define una vista materializada  $VM$ , entonces  $MDA = \{VM_1, VM_2, \dots, VM_k\}$ . La lista de atributos del esquema  $VM_k$  contiene los atributos de la relación base envueltos en la parte  $k$ , más un atributo *cnt*. El atributo *cnt* representa la cantidad de tuplas de la relación base incluidas en dicho grupo.

A modo de ejemplo de las estructuras descritas presentamos, en la Figura 4.3, la definición una RA, una RAD y una DA sobre la relación presentada en la Tabla 2.3. En el caso de RAD, las etiquetas lingüísticas son las mismas definidas en la Figura 2.3. Como se ha comentado, la actualización de las medidas en el enfoque Simplificado involucra la exploración completa de la base de datos para recalcular las estructuras implicadas, desechando de esta forma los valores que previamente se conocían.

$r_0 : R$		
ID	Edad	Salario
1	26	800
2	52	1100
3	65	3000
4	65	3000
5	26	800

RA: Edad=26  $\Rightarrow$  Salario=800  
**MRA** = { | Edad=26|, | Salario=800|, | Edad=26  $\wedge$  Salario=800|, |  $r_i$  | }  
 = { 2, 2, 2, 5 }

RAD: Edad=media  $\Rightarrow$  Salario=medio  
**MRAD** = {  $\tilde{\Gamma}^{r_0}_{\{<Edad,media>\}}$ ,  $\tilde{\Gamma}^{r_0}_{\{<Salario,medio>\}}$ ,  $\tilde{\Gamma}^{r_0}_{\{<Edad,media>,<Salario,medio>\}}$ , |  $r_i$  | }  
 = { 1.6, 0.96, 0.96, 5 }

RDA: Edad  $\Rightarrow$  Salario  
**MDA** = {  $VM_1, VM_2, VM_3, VM_4$  }

$VM_1$		$VM_2$		$VM_3$			$VM_4$
Edad	cnt	Salario	cnt	Edad	Salario	cnt	cnt
26	2	800	2	26	800	2	5
52	1	1100	1	52	1100	1	
65	2	3000	2	65	3000	2	

Figura 4.3: Ejemplos de las estructuras MRA, MRAD y MDA para el mantenimiento de la métrica *information gain*.

Estas estructuras, en una implementación para bases de datos relacionales, son incluidas en el esquema relacional de los metadatos, estando disponibles como parte del catálogo de la base de datos. De esta forma, cada estructura se convierte en una relación donde las tuplas corresponden a las reglas mantenidas y los atributos corresponden a las partes de la(s) medida(s) definida(s). Estas medidas definidas también son incluidas a dicha relación como atributos, reflejando el valor de dicha métrica a partir de sus partes. El tipo de dato de los atributos que forman parte de la medida se obtiene directamente de la definición de su estructura como se puede observar en la Figura 4.3, es decir, tipo entero para las MRA y tipo real para las MRAD. En el caso de las MDA los atributos son de tipo cadena, reflejando el

nombre de la relación asociada a la vista materializada. Estas últimas relaciones también forman parte del catálogo y son definidas exactamente como el esquema  $VM_k$ . Dentro del catálogo deben incluirse además las funciones o procedimientos que calculen la(s) medida(s), a partir de las partes definidas.

## 4.4 Algoritmo para el mantenimiento incremental inmediato de las reglas

El algoritmo para el mantenimiento incremental inmediato de las reglas se encuentra orientado a actualizar la base de reglas inmediatamente después de ocurrido un evento estructural primitivo [104]. Por lo tanto, un solo registro es analizado a cada momento. A diferencia del enfoque Simplificado esta propuesta no necesita explorar la base de datos, sino que reutiliza los valores previos de las medidas directamente a través de sus estructuras. Para este mantenimiento se considera el incremento realizado a la base de datos y se considera qué reglas deben ser analizadas. Está claro que un **Evento Estructural Primitivo (EEP)** no involucrado en la relación  $r_0$  de la Figura 4.3 no afecta ninguna de las reglas en dicho ejemplo. Además de la regla, también se verifica qué parte es necesaria analizar pues, si continuando con el ejemplo de la Figura 4.3, modificamos la edad de una persona ¿es necesario analizar los soportes del salario? El Algoritmo 2 presenta el pseudocódigo de la propuesta inmediata para el mantenimiento incremental de las reglas previamente extraídas. Este algoritmo considera de forma genérica las reglas de asociación junto a sus extensiones difusas y aproximadas, por lo cual hemos denotado **Medida de una Regla (MR)** como la estructura que bien puede ser una **MRA**, una **MRAD** o una **MDA**.

Como se puede observar, una regla es analizada en un evento de actualización si alguno de sus atributos cambia de valor, por lo cual la lista de atributos modificados se define como  $L = \{a \in R \mid t_0^+[a] \neq t_1^+[a]\}$ . También es necesario incluir en esta lista el cambio de estado desconocido (*null*) de los atributos. Para los eventos de inserción y actualización siempre es necesario analizar las reglas, aunque también se debe considerar el caso de valores desconocidos.

**Algoritmo 2:** Mantenimiento incremental inmediato para una Regla

**Entrada:** evento estructural compuesto  $EEC$  que modifica los atributos listados en  $L$ , y las partes de la medida genérica  $MR$  de la regla  $X \Rightarrow Y$

**Salida:** actualización de  $MR$

**Método:**

```

1 foreach  $EEP \in EEC$  do
2   if ( $EEP = \Delta t^{-+}$ ) then                                /* evento de actualización */
3     if ( $L \cap \{X \cup Y\} \neq \emptyset$ ) then
4       foreach  $Mp_k \in MR$  do
5         if ( $L \cap \{\text{atributos en } Mp_k\} \neq \emptyset$ ) then
6           update  $Mp_k$ , incrementar con  $t_0^{-+}$ ;
7           update  $Mp_k$ , decrementar con  $t_1^{-+}$ ;
8     else if ( $EEP = \Delta t^+$ ) then                            /* evento de inserción */
9       foreach  $Mp_k \in MR$  do
10        update  $Mp_k$ , incrementar con  $t^+$ ;
11    else                                                        /* evento de eliminación ( $EEP = \Delta t^-$ ) */
12      foreach  $Mp_k \in MR$  do
13        update  $Mp_k$ , decrementar con  $t^-$ ;
14 retorna:  $MR$ 

```

Si bien se ha definido este algoritmo de forma genérica para los diferentes tipos de reglas, el incremento de las mismas es diferente según la estructura utilizada. Para ilustrar esta propuesta de mantenimiento incremental en las RA se presenta la Figura 4.4. En la misma, tres eventos estructurales primitivos modifican la relación de la Tabla 2.3 e inmediatamente, para cada evento, su estructura es actualizada. En dicha figura se consideran los nuevos (*new*) y antiguos (*old*) datos involucrados en la regla.

De igual forma se presenta un ejemplo de nuestra propuesta inmediata para las RAD. A diferencia del mantenimiento de las RA, cada modificación en los atributos de una RAD tiene asociado un grado de membresía que es necesario procesar. La Figura 4.5 muestra esta diferencia.

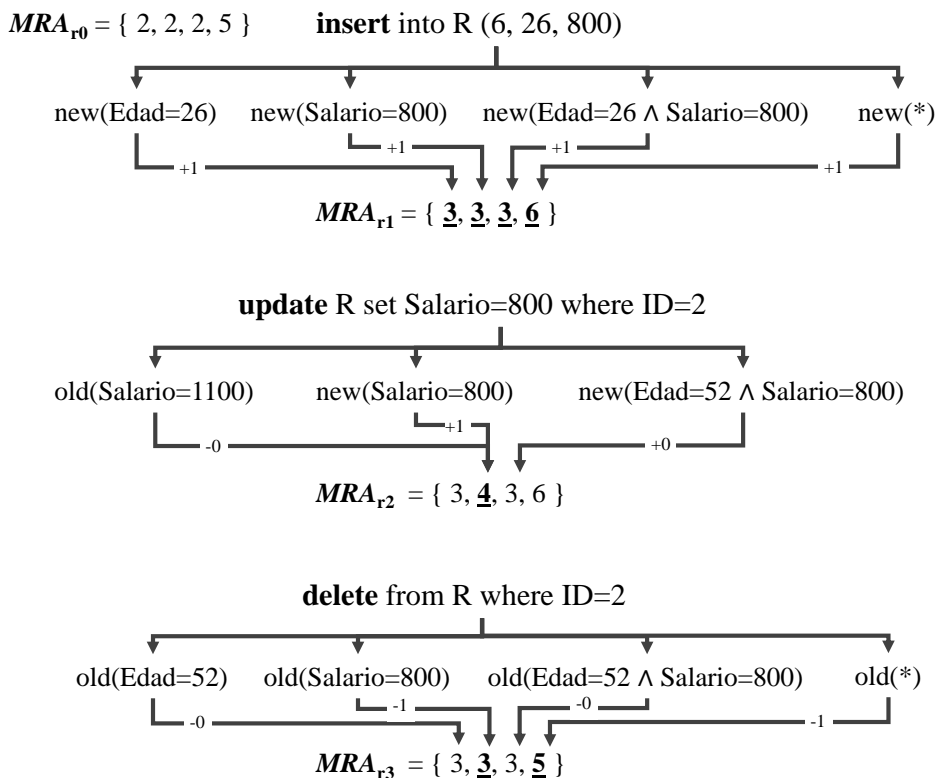


Figura 4.4: Ejemplo de mantenimiento incremental inmediato para una RA

Continuando con estos ejemplos se provee el mantenimiento inmediato para las DA. De igual forma se utilizan los eventos estructurales mostrados y se verifican los cambios realizados en las vistas materializadas mediante la Figura 4.6.

Esta propuesta inmediata, como se ha descrito, utiliza los resultados previos para recalculer las modificaciones producidas por los nuevos incrementos. Lo cual se realiza sin ser necesaria la exploración a la base de datos, clasificándola de esta forma como autosostenible. Otra característica a citar es la sencillez en evaluar el valor exacto de las medidas. Para ello únicamente es necesario aplicar las operaciones matemáticas involucradas en las fórmulas de las medidas. Sin embargo, un aspecto desfavorable de esta propuesta es que el mayor peso del procesamiento se realiza en los propios eventos estructurales del sistema. Esto conlleva a aumentar los tiempos de respuesta de la base de datos y el riesgo de disminuir seriamente el desempeño en bases de datos estresadas.

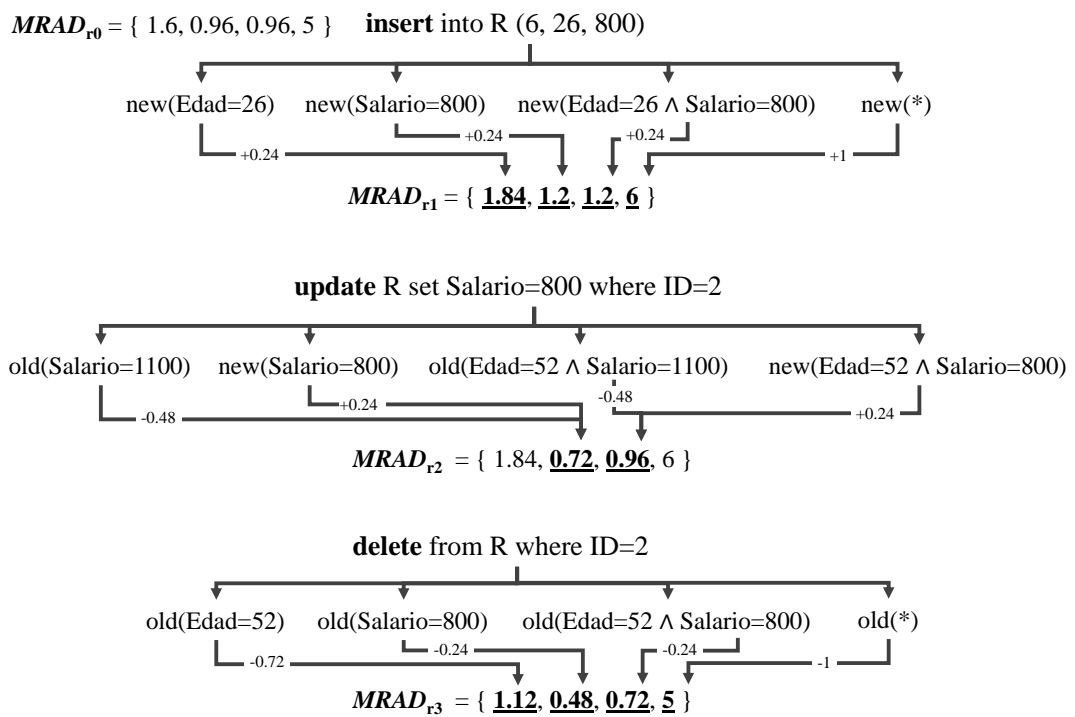


Figura 4.5: Ejemplo de mantenimiento incremental inmediato para una RAD

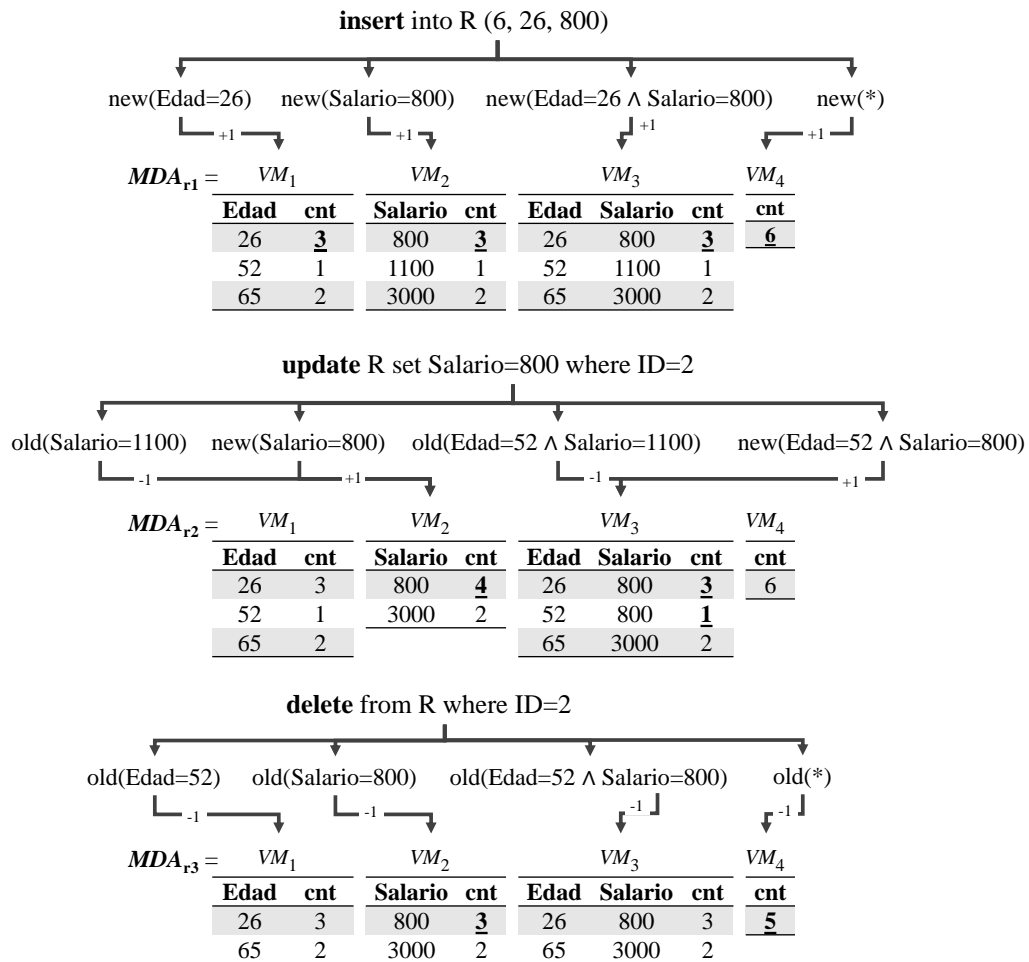


Figura 4.6: Ejemplo de mantenimiento incremental inmediato para una DA

## 4.5 Algoritmo para el mantenimiento incremental diferido de las reglas

El algoritmo para el mantenimiento incremental diferido de las reglas se encuentra orientado a actualizar la base de reglas luego de ocurrir un evento estructural compuesto [104]. Esto posibilita, a diferencia del algoritmo inmediato, la ocurrencia de varios eventos estructurales primitivos o compuestos antes de actualizar las medidas de las reglas. Mediante esta forma, se garantiza que el mayor peso del procesamiento de actualización de la base de reglas recaiga en un momento diferente a la ocurrencia del evento.

Para almacenar estos incrementos ocurridos en la base de datos se utilizan relaciones auxiliares. Estas relaciones auxiliares están vinculadas a aquellas relaciones que participan en las reglas mantenidas. Es usual considerar varios tipos, según los tipos de eventos estructurales. Es decir, una relación auxiliar para las inserciones, otra para las actualizaciones y una tercera para las eliminaciones [22]. Sin embargo, en la presente propuesta diferida se analizan tres tipos de eventos estructurales y solo se consideran dos relaciones auxiliares: de inserción y de eliminación. Así, las primeras almacenan las tuplas insertadas y el nuevo valor de las tuplas actualizadas,  $t^+ \cup t_1^{-+}$ . Por su lado, las relaciones auxiliares de eliminación almacenan las tuplas eliminadas y el antiguo valor de las tuplas actualizadas,  $t^- \cup t_0^{-+}$ . El esquema de una relación auxiliar es idéntico al esquema de la relación sobre la cual se define. Esto permite el almacenamiento completo de los nuevos datos, a diferencia de almacenar únicamente su referencia como se propone en [22]. Como aspecto negativo, esta solución conlleva a mantener valores duplicados en la base de datos. Sin embargo, también posibilita que nuestra propuesta diferida no necesite explorar la base de datos y mantenga un comportamiento autosostenible. Estas relaciones auxiliares, en la base de datos, forman parte del esquema relacional de los metadatos estando accesibles desde el catálogo.



Como se apuntó en la Sección 3.2, la consideración de Eventos Estructurales Compuestos (EEC) impone el análisis de las interacciones entre los eventos primitivos. Estas interacciones conocidas como “efectos de red” tienen el objetivo de calcular las instancias relevantes para el EEC. Las principales diferencias con respecto al procesamiento de los eventos estructurales de las propuestas inmediata y diferida son ilustrados mediante la Figura 4.7.

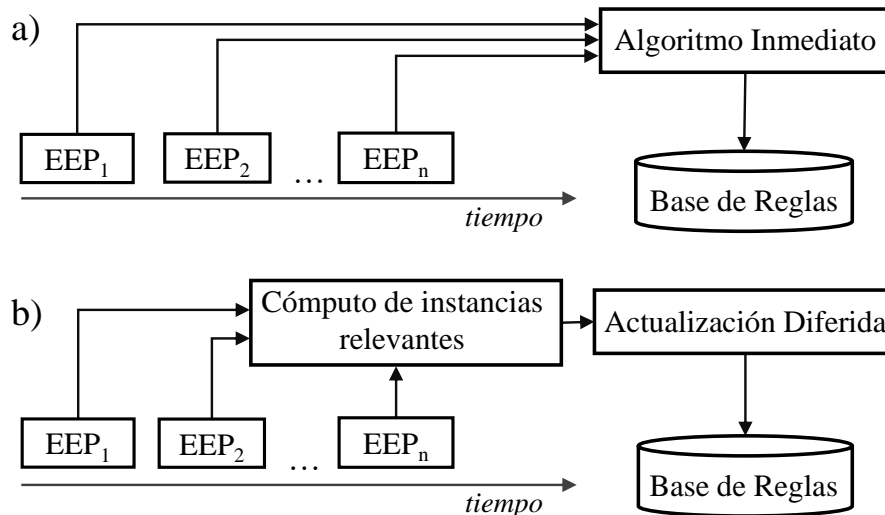


Figura 4.7: Manejo de los EEP en la propuesta inmediata a) y diferida b).

La propuesta diferida, como se puede apreciar en la Figura 4.7, se compone de dos subproblemas. El primero consiste en computar las instancias afectadas que son relevantes en la transición de la base de datos. Mientras el segundo está relacionado con actualizar la base de reglas a partir de estas instancias relevantes. Los efectos de red considerados para el primer subproblema incluyen las cuatro políticas presentadas en la Sección 3.2 más una quinta. Esta política añadida especifica la actualización de una tupla como la ocurrencia de su eliminación seguida de su inserción. Con lo cual, los efectos de red considerados en nuestra propuesta diferida definen:

- Si una tupla es varias veces actualizada, solamente se considera la actualización compuesta.

- Si una tupla es actualizada y luego eliminada, solamente se considera la actualización.
- Si una tupla es insertada y luego actualizada, se considera la inserción actualizada.
- Si una tupla es insertada y luego eliminada, no se considera.
- Si una tupla es eliminada y luego insertada, se considera como actualizada.

Estas políticas son verificadas de forma activa en las relaciones auxiliares que registran los nuevos eventos ocurridos. El siguiente Algoritmo 3 la considera eficientemente y constituye nuestra solución para el primer subproblema del mantenimiento diferido.

---

**Algoritmo 3:** Cómputo de instancias relevantes en una transición

---

**Entrada:** evento estructural compuesto  $EEC$ , relación auxiliar de inserción  $RAI$  y eliminación  $RAE$

**Salida:** actualización de las relaciones auxiliares  $RAI$  y  $RAE$  para  $EEC$

**Método:**

```

1 foreach  $EEP \in EEC$  do
2   if ( $EEP = \Delta t^{-+}$ ) then                                /* evento de actualización */
3     if ( $\{t_0^{-+} \cap RAI\} = \emptyset$ ) then
4        $\lfloor$  insert into  $RAI$  values  $t_1^{-+}$ ;
5        $\lfloor$  insert into  $RAE$  values  $t_0^{-+}$ ;
6     else
7        $\lfloor$  update  $u \in RAI$  set  $u = t_1^{-+}$  where  $u = t_0^{-+}$ ;
8   else if ( $EEP = \Delta t^{+}$ ) then                            /* evento de inserción */
9      $\lfloor$  insert into  $RAI$  values  $t^{+}$ ;
10  else                                                         /* evento de eliminación ( $EEP = \Delta t^{-}$ ) */
11    if ( $\{t^{-} \cap RAI\} = \emptyset$ ) then
12       $\lfloor$  insert into  $RAE$  values  $t^{-}$ ;
13    else
14       $\lfloor$  delete  $u \in RAI$  where  $u = t^{-}$ ;
15 retorna:  $RAI, RAE$ 

```

---

Este algoritmo activo adiciona un mínimo de procesamiento sobre las operaciones regulares de la base de datos. Solo la necesaria para almacenar las instancias relevantes y aplicar las políticas de red. Un pequeño ejemplo para mostrar el comportamiento de dicho algoritmo es ilustrado por la Figura 4.8, donde para cada evento estructural primitivo se observan los cambios en las relaciones auxiliares. En este ejemplo, los tres eventos estructurales son reducidos a dos instancias relevantes en las relaciones auxiliares.

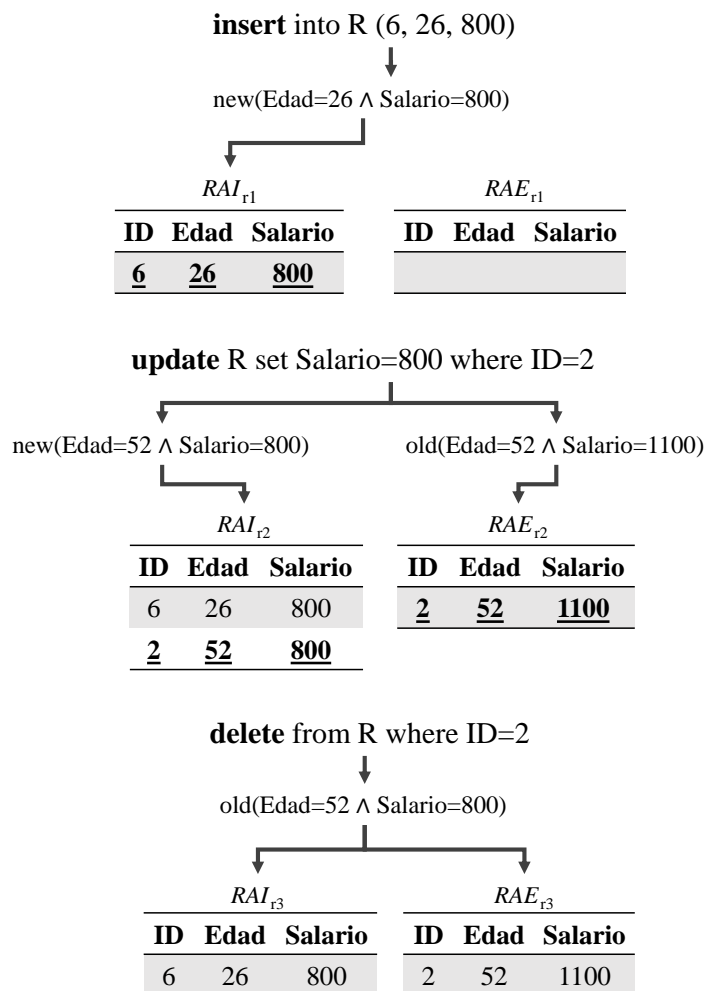


Figura 4.8: Ejemplo del cómputo de instancias relevantes para una transición.

La base de reglas es actualizada únicamente con estas instancias, mediante el mantenimiento de las estructuras asociadas a los tipos de reglas. Este segundo subproblema puede ser interpretado como una actualización inmediata de las

estructuras, donde se consideran que las instancias computadas en *RAI* son eventos de inserción y las instancias computadas en *RAE* son eventos de eliminación. Esta actualización a la base de reglas puede ser realizada a petición del usuario bajo demanda para la toma de decisiones o bien programadas cada cierto tiempo. Este paso es cumplido por el Algoritmo 4.

---

**Algoritmo 4:** Mantenimiento incremental diferido para las instancias relevantes

---

**Entrada:** relación auxiliar de inserción *RAI* y eliminación *RAE* resultantes del Algoritmo 3 y las partes de la medida genérica *MR* de la regla

**Salida:** actualización de *MR*

**Método:**

```
1 foreach  $Mp_k \in MR$  do
2   |   update  $Mp_k$ , incrementar con RAI;
3   |   update  $Mp_k$ , decrementar con RAE;
4 vaciar RAI;
5 vaciar RAE;
6 retorna: MR
```

---

El incremento de las partes de las medidas, a partir de relaciones auxiliares, es realizado de forma similar al enfoque Simplificado; con la diferencia de que las consultas no se realizan sobre la base de datos, sino sobre las relaciones auxiliares. Al igual que el algoritmo inmediato, la actualización de las partes se realiza según el tipo de la regla. A modo de ejemplo del Algoritmo 3 se presenta en la Figura 4.9 la actualización de una *RA* a partir de las relaciones auxiliares. El comportamiento para las *RAD* y *DA* es fácil de deducir a partir del comportamiento del algoritmo inmediato.

RA: Edad=26  $\Rightarrow$  Salario=800

$$MRA_{r_0} = \{ | \text{Edad}=26 |, | \text{Salario}=800 |, | \text{Edad}=26 \wedge \text{Salario}=800 |, | r_1 | \}$$

$$= \{ 2, 2, 2, 5 \}$$

$RAI_{r_3}$			$RAE_{r_3}$			$CSp^1(RAI_{r_3}) = 1$	$CSp^1(RAE_{r_3}) = 0$
ID	Edad	Salario	ID	Edad	Salario	$CSp^2(RAI_{r_3}) = 1$	$CSp^2(RAE_{r_3}) = 0$
6	26	800	2	52	1100	$CSp^3(RAI_{r_3}) = 1$	$CSp^3(RAE_{r_3}) = 0$
						$CSp^4(RAI_{r_3}) = 1$	$CSp^4(RAE_{r_3}) = 1$

$$MRA_{r_3} = \{ 2 + CSp^1(RAI_{r_3}) - CSp^1(RAE_{r_3}), 2 + CSp^2(RAI_{r_3}) - CSp^2(RAE_{r_3}),$$

$$2 + CSp^3(RAI_{r_3}) - CSp^3(RAE_{r_3}), 5 + CSp^4(RAI_{r_3}) - CSp^4(RAE_{r_3}) \}$$

$$MRA_{r_3} = \{ \underline{3}, \underline{3}, \underline{3}, \underline{5} \}$$

$RAI_{r_3}$			$RAE_{r_3}$		
ID	Edad	Salario	ID	Edad	Salario

Figura 4.9: Ejemplo de actualización diferida de  $MRA$  para las instancias relevantes.

Nótese que el incremento de las partes de las medidas es realizado sin el acceso a la base de datos, por lo cual afirmamos que esta propuesta al igual que la inmediata es automantenida. Luego de actualizar las reglas con las instancias relevantes se vacían las relaciones auxiliares, permitiendo de esta forma comenzar nuevamente su cómputo para nuevos eventos estructurales. Heurísticamente, el tiempo de consultar las relaciones auxiliares debe ser mucho menor que el de consultar la base de datos. Solo en raras ocasiones pudiera no ocurrir de esta forma. Por ejemplo, si la base de datos fuese eliminada completamente las consultas sobre ella serían muy veloces, sin embargo, las relaciones auxiliares almacenarían tantas modificaciones que consultarlas no sería eficiente.

## 4.6 Análisis de complejidad temporal de los algoritmos propuestos

Debido a que nuestras propuestas consideran un componente activo que interviene en las operaciones regulares de la base de datos, el análisis de complejidad temporal ha sido dividido. Por un lado, es necesario analizar el procesamiento añadido a los eventos estructurales mediante los recursos activos. Lo cual posee un impacto directo en el tiempo de respuesta de la base de datos. Por otro lado, es necesario analizar la actualización de la base de reglas; un aspecto que influye directamente en el tiempo de obtener las medidas exactas para la posible toma de decisiones. La Figura 4.10 muestra los dos momentos analizados en nuestros algoritmos: propuesta activa y actualización de medidas.

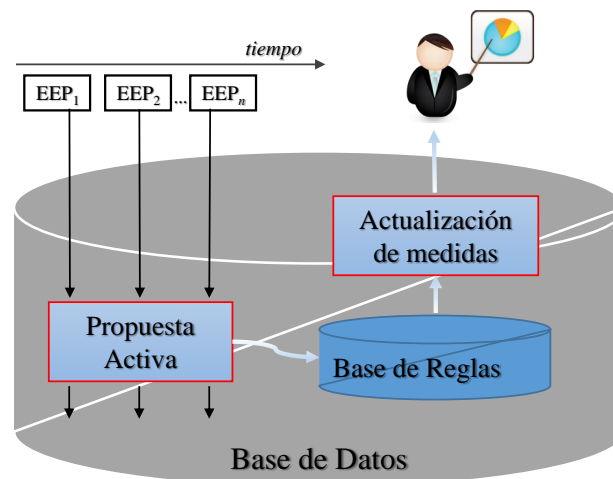


Figura 4.10: Diseño del análisis realizado en los algoritmos propuestos.

Para la complejidad temporal de los algoritmos se consideraron constantes en los diferentes tipos de reglas analizadas. Una de estas constantes es la cantidad de reglas que deben ser actualizadas, pues aunque la base de reglas puede variar, es posible acotarlas. Otra constante es la cantidad de partes necesarias para mantener la(s) medida(s). Según lo descrito en la Sección 4.2 este número de partes también es posible de acotar con una pequeña constante. En el caso del algoritmo diferido el

componente activo es indiferente de la cantidad de reglas y de la cantidad de partes, solo la actualización de las medidas considera esta última como una constante. Estos y otros análisis se resumen en la Tabla 4.2.

Tabla 4.2: Complejidad temporal de los algoritmos propuestos

	Propuesta Inmediata		Propuesta Diferida	
	Comp. Activo	Actualizar Medida	Comp. Activo	Actualizar Medida
RA	$O(n)$	$O(1)$	$O(n \log n)$	$O(\log n)$
RAD	$O(n)$	$O(1)$	$O(n \log n)$	$O(\log n)$
DA	$O(n \log m)$	$O(\log m)$	$O(n \log n)$	$O(\log n \log m)$

En estos análisis  $n$  es considerado el número de eventos estructurales primitivos y  $m$ , en las dependencias aproximadas, representa la mayor cardinalidad para las vistas materializadas de esta propuesta. Se deben tener en cuenta las constantes definidas para no confundir, por ejemplo, el doble bucle existente en el algoritmo inmediato con una complejidad cuadrática.

Para todas las relaciones auxiliares y vistas materializadas se considera la estructura de datos Árbol-B (*B-tree*). Dicha estructura en su variante B+ es comúnmente utilizada por los índices de las tablas en los gestores relacionales donde posee operaciones evaluadas en  $O(\log n)$ . Esto conlleva a establecer para la propuesta diferida, en su componente activo, una complejidad de  $\log 1 + \log 2 + \dots + \log n = \log(n!) \in n \log n$ . Mediante la Figura 4.11 se representan las funciones que acotan superiormente nuestras propuestas en su componente activo.

Desde esta gráfica se puede observar cómo para un los valores de  $n < n_0$  el comportamiento del algoritmo diferido ( $O(\log n!)$ ), en su componete activo, es superior al inmediato ( $O(n)$ ). Sin embargo, para valores mayores a  $n_0$  el algoritmo diferido degrada rápidamente su comportamiento. En este aspecto, teóricamente, también es posible considerar una implementación del algoritmo diferido mediante tablas hash con una complejidad temporal de  $O(n)$  en el peor caso, pero esperar un desempeño de  $O(1)$  o considerar estructuras *B-tree* particionadas [73]. En

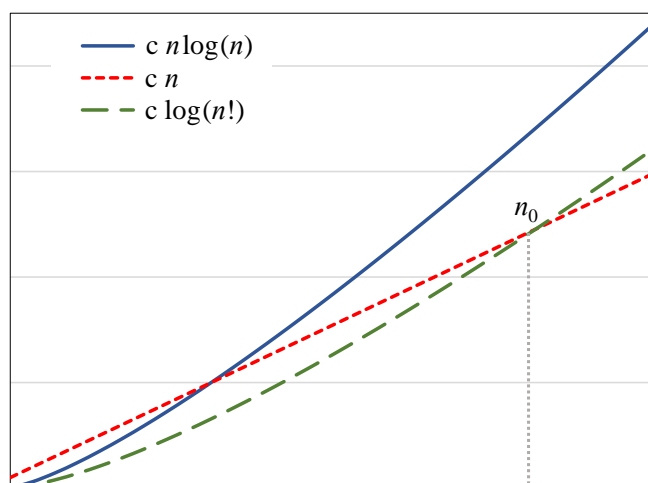


Figura 4.11: Representación de las cotas asintóticas superiores  $O(n \log(n))$ ,  $O(n)$  y  $O(n \log(n))$ .

cualquier caso nótese que el algoritmo diferido mantiene una homogeneidad entre los diferentes tipos de reglas, lo cual se debe a que no toma en cuenta el tipo ni la cantidad de reglas.

La principal ventaja de la propuesta inmediata es su rapidez para calcular las medidas en tiempo real como se puede observar en la Tabla 4.2. La actualización de las estructuras de las medidas luego de cada evento estructural primitivo evita grandes procesamientos en el acceso a las medidas. Únicamente es necesario su cálculo mediante las partes de las medidas. Sin embargo, para las dependencias aproximadas presenta una complejidad temporal basada en  $m$  que reduce su eficiencia. En el caso de la propuesta diferida el cálculo de las medidas necesita un mayor procesamiento que la propuesta inmediata.

## 4.7 Estudio experimental

Esta sección del capítulo evalúa el comportamiento de las propuestas para el mantenimiento incremental de las reglas en diferentes experimentos. Primeramente, se describe el diseño experimental que permite observar las



características principales de nuestras propuestas. A continuación, se presentan los conjuntos de datos utilizados en el estudio, sobre los cuales se analiza posteriormente el desempeño y la escalabilidad de los algoritmos inmediato y diferido.

### 4.7.1 Diseño de la experimentación

Es importante aclarar que nuestro objetivo no es comparar las propuestas inmediata y diferida, sino comprobar bajo qué condiciones se puede elegir una de ellas como vía de mantenimiento incremental del conocimiento previamente extraído. Los experimentos realizados en la presente memoria se encuentran diseñados sobre esta premisa.

La propuesta activa de los algoritmos inmediato y diferido ha sido implementada en la propia base de datos mediante disparadores. Este recurso activo abordado en la Sección 3.4 se definió para ser ejecutado con un tiempo de activación *after*, granularidad *for each row* y todos los eventos desencadenadores. Esto brinda una perspectiva real en la implementación de nuestros algoritmos para los sistemas de información desde la propia base de datos del negocio. La actualización de las medidas y la base de reglas para nuestros experimentos también se encuentran en la base de datos. Sin embargo, en entornos de desarrollo reales pueden implementarse fuera de la base de datos y garantizar su independencia con el sistema.

La Figura 4.10 nos ayuda también a explicar los dos análisis realizados: análisis de desempeño y análisis de escalabilidad. El primero se encuentra diseñado para comprobar el comportamiento de los algoritmos en diferentes cantidades de **EEP**, con un tamaño fijo de la base de datos. Este análisis es ampliamente utilizado en la literatura incremental [74, 79, 81, 63, 77, 46]. El segundo se encuentra diseñado para mostrar el comportamiento de los algoritmos cuando se mantiene fija la cantidad de **EEP** y aumenta el tamaño de la base de datos [30].

Los experimentos de nuestras propuestas han sido completamente realizados en dos de los sistemas gestores de BDA de código abierto más utilizados hoy día: PostgreSQL® [108] y MySQL® [45]. Para PostgreSQL® se utilizó su dialecto PL/pgSQL en su versión 9.2.2, disponible de forma gratuita<sup>1</sup>. Para MySQL® y su dialecto que implementa SQL/PSM (*SQL/Persistent Stored Module*) se escogió su edición *Community*, versión 5.6.13, la cual también es gratuita<sup>2</sup>. Ambos sistemas gestores de bases de datos fueron instalados en un servidor dedicado para los experimentos, equipado con ocho procesadores Intel® Core™ i7-2600 a 3.4GHz y 15GB de memoria principal.

Para ejecutar estos experimentos se escribió una pequeña aplicación en Java. La misma recibe como parámetros: el gestor donde se va a ejecutar el algoritmo, los tipos de reglas y algoritmos del experimento, la cantidad de ejecuciones cuyos resultados serán promediados, los archivos con los EEP y un archivo con la base de datos inicial. Posteriormente, ejecuta de una vez, todos los experimentos necesarios en varias bases de datos, donde se encuentran implementados los algoritmos correspondientes. Cada ejecución que realiza esta aplicación comienza con la limpieza y organización de las estructuras necesarias en el experimento. Posteriormente se colocan todas las operaciones a realizar en la base de datos. Finalmente se obtiene y almacena el tiempo de la ejecución que luego es promediado.

Para otros experimentos que incluyeron algoritmos de extracción de reglas se utilizó la herramienta de minería de datos KEEL [3]. KEEL es una herramienta implementada en Java, lo que le permite al usuario utilizar diversos algoritmos para la extracción de reglas. Por otro lado, también permite la ejecución de dichos algoritmos fuera de su entorno, de forma desconectada (*off-line*). Esta característica fue utilizada para realizar todos los experimentos en el mismo servidor.

---

<sup>1</sup>Núcleo de PostgreSQL® para varios sistemas operativos, públicamente disponible para su descarga desde *PostgreSQL Global Development Group*: <https://www.postgresql.org/download>.

<sup>2</sup>Versión comunitaria gratuita de MySQL®, públicamente disponible para su descarga desde *Oracle Corporation*: <https://dev.mysql.com/downloads/mysql/5.6.html#downloads>.

## 4.7.2 Conjuntos de datos

Para la experimentación fueron seleccionados conjuntos de datos de diferentes procedencias, todos de entornos reales. De forma general se tomaron dos clases de conjuntos de datos. Uno de estos conjuntos fue extraído desde un sistema ampliamente utilizado por la Universidad de Granada (UGR), mientras que el segundo está formado por varios conjuntos disponibles en repositorios de aprendizaje automatizado.

### SWAD: Sistema Web de Apoyo a la Docencia

El Sistema Web de Apoyo a la Docencia (SWAD) de la Universidad de Granada es un sistema de gestión de aprendizaje virtual y libre para gestionar las asignaturas, estudiantes y profesores de una o varias instituciones docentes [20]. SWAD permite el acceso de diferentes tipos de usuarios; principalmente estudiantes, profesores y administradores. Este sistema de ambiente educativo facilita la actividad docente en la UGR. Brinda funcionalidades para las asignaturas, evaluaciones de estudiantes por asignatura, estadísticas, redes sociales para foros y chat, información personal, entre otras.

Los conjuntos de datos utilizados del SWAD son tres relaciones derivadas de respaldos (*backups*) realizados al sistema el 28-febrero-2014 (330 994 registros), el 8-mayo-2014 (333 218 registros) y el 4-julio-2014 (337 359). Estas relaciones se extrajeron con el objetivo de aplicar métodos para la extracción de datos educacionales [111], conteniendo nueve atributos que describen la información de un estudiante en un curso (asignatura). Estos atributos son:

- **cnt\_asignaciones.** Cantidad de asignaciones (tareas) realizadas al estudiante en el curso. Pertenece al dominio de los enteros positivos.
- **cnt\_otrosTrab.** Cantidad de trabajos voluntarios entregados por el estudiante en el curso. Pertenece al dominio de los enteros positivos.
- **cnt\_actividades.** Cantidad de actividades obligatorias subidas por el estudiante en el curso. Pertenece al dominio de los enteros positivos.

- **sum\_ficheros.** Suma de accesos a ficheros del estudiante en el curso. Pertenece al dominio de los enteros positivos.
- **cnt\_ficheros.** Cantidad diferentes de ficheros vistos por el estudiante en el curso. Pertenece al dominio de los enteros positivos.
- **cnt\_clics.** Cantidad de acciones (clics) del estudiante en el curso. Pertenece al dominio de los enteros positivos.
- **cnt\_mensajes.** Cantidad de mensajes enviados por el estudiante en el curso. Pertenece al dominio de los enteros positivos.
- **pc\_pcontestadas.** Porcentaje de preguntas contestadas del estudiante en todos los exámenes del curso. Pertenece al dominio de los reales en el intervalo  $[0,1]$ .
- **avg\_puntos.** Promedio de puntos por pregunta en todos los exámenes del estudiante en el curso. Pertenece al dominio de los reales en el intervalo  $[-1,1]$ .

La última relación extraída del SWAD permitió descubrir las reglas expresadas en el Apéndice B, las cuales fueron utilizadas en los experimentos. Para la extracción de estas reglas se utilizó la herramienta de minería de datos KEEL, con su algoritmo *apriori* para las RA y *fuzzy-apriori* para las RAD. Los parámetros de soporte y confianza se ajustaron al 10% y 80%, respectivamente. Para las DA se utilizó la herramienta XPressDMiner, desarrollada en el seno del Grupo de Investigación en Bases de Datos y Sistemas de Información Inteligentes (IdBIS<sup>3</sup>), con parámetros mínimos de soporte, confianza y factor de certeza.

Por otro lado, se tomaron las tres relaciones extraídas como diferentes transiciones de la base de datos, con las cuales se obtuvieron los eventos estructurales que respondían a estos cambios. Para ello se aplicó de forma inversa la definición de eventos estructurales primitivos de la Sección 2.5.

---

<sup>3</sup>Grupo de Investigación del Departamento de Ciencias de la Computación e Inteligencia Artificial, Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación (Universidad de Granada): <https://idbis.ugr.es/>.

## Repositorio de aprendizaje automatizado UCI Irvine

El segundo repositorio utilizado para los experimentos fue extraído de las bases de datos brindadas por el repositorio de aprendizaje automatizado UCI Irvine (Universidad de California, Irvine) [78].

Específicamente, se utilizaron las características *Color Moments* y *Color Texture* de la colección de imágenes de coral *Corel Image Features*, la cual está compuesta por 68 040 instancias. El conjunto de datos *Color Moments* contiene diez atributos (nueve reales y uno entero) mientras que el conjunto de datos *Color Texture* contiene diecisiete atributos (dieciséis reales y uno entero). Otro conjunto de datos utilizado es la base clínica de registros de pacientes *Diabetes* [120], la cual contiene 101 766 instancias descritas por veinte atributos (categóricos y enteros). Además, se empleó la base de registros *Census Income* del censo de 1994, la cual se compone de 48 842 instancias descritas por catorce atributos (categóricos y enteros). Mayores detalles de estos conjuntos de datos pueden ser obtenidos desde el portal web de la UCI <sup>4</sup>.

### 4.7.3 Análisis de desempeño

Los primeros experimentos realizados tuvieron como objetivo observar el comportamiento de nuestras propuestas ante las nuevas operaciones que constantemente modifican la base de datos. Este desempeño de los algoritmos propuestos en las operaciones regulares de la base de datos tiene una gran importancia para su posible implantación en sistemas reales. Por este motivo se presentan los resultados sobre el conjunto de datos SWAD.

En este análisis se organizaron diferentes conjuntos de eventos estructurales con cantidades que fuesen aumentando constantemente. Estos eventos se ejecutaron en las implementaciones de la propuesta incremental y diferida de las RA, RAD y DA expuestas en el Apéndice B. El tamaño de la base de datos es constante

---

<sup>4</sup>Conjunto de datos públicamente disponible para su descarga desde la UCI (Universidad de California, Irvine): <http://archive.ics.uci.edu/ml/datasets>.

y relativamente pequeño (5K registros). Con estas ejecuciones se promedió el tiempo de ejecución de los eventos estructurales y posteriormente, el tiempo de actualización de las medidas. Este experimento se realizó en el mantenimiento de cada tipo de regla, tal como muestran la Figura 4.12, Figura 4.13 y Figura 4.14.

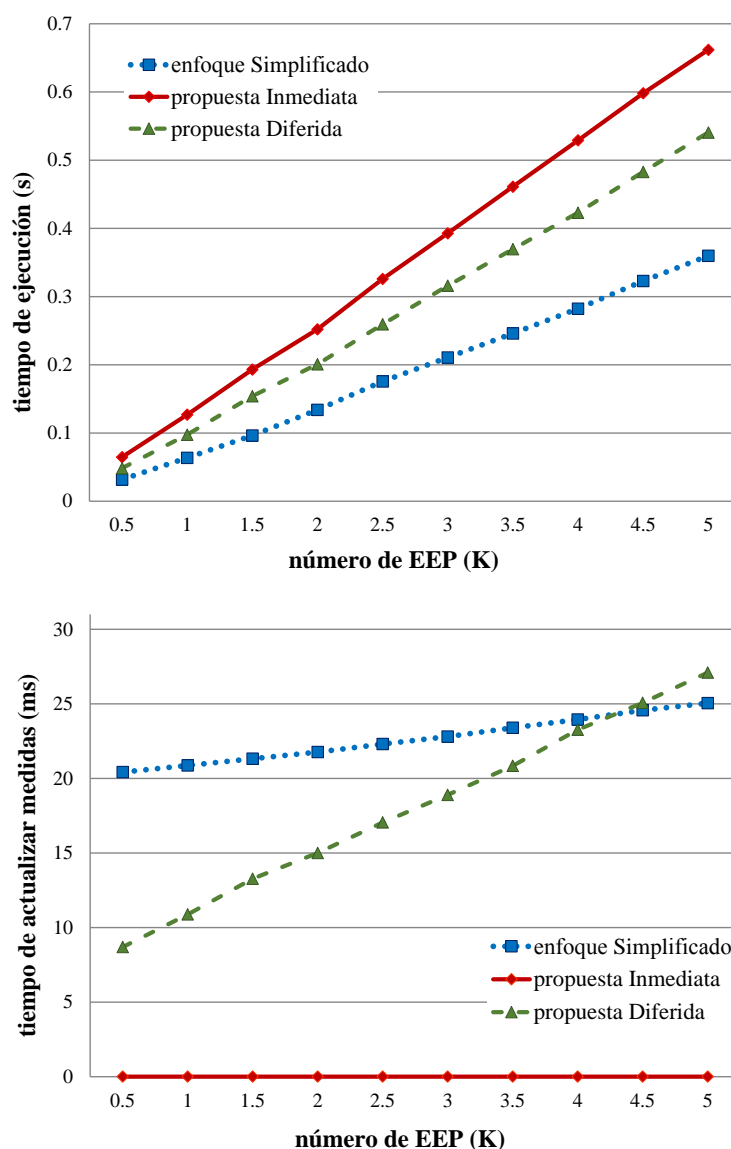


Figura 4.12: Desempeño de las propuestas en el mantenimiento de RA. Tiempo ejecución (arriba) y tiempo de actualización de las medidas (abajo) en PostgreSQL.

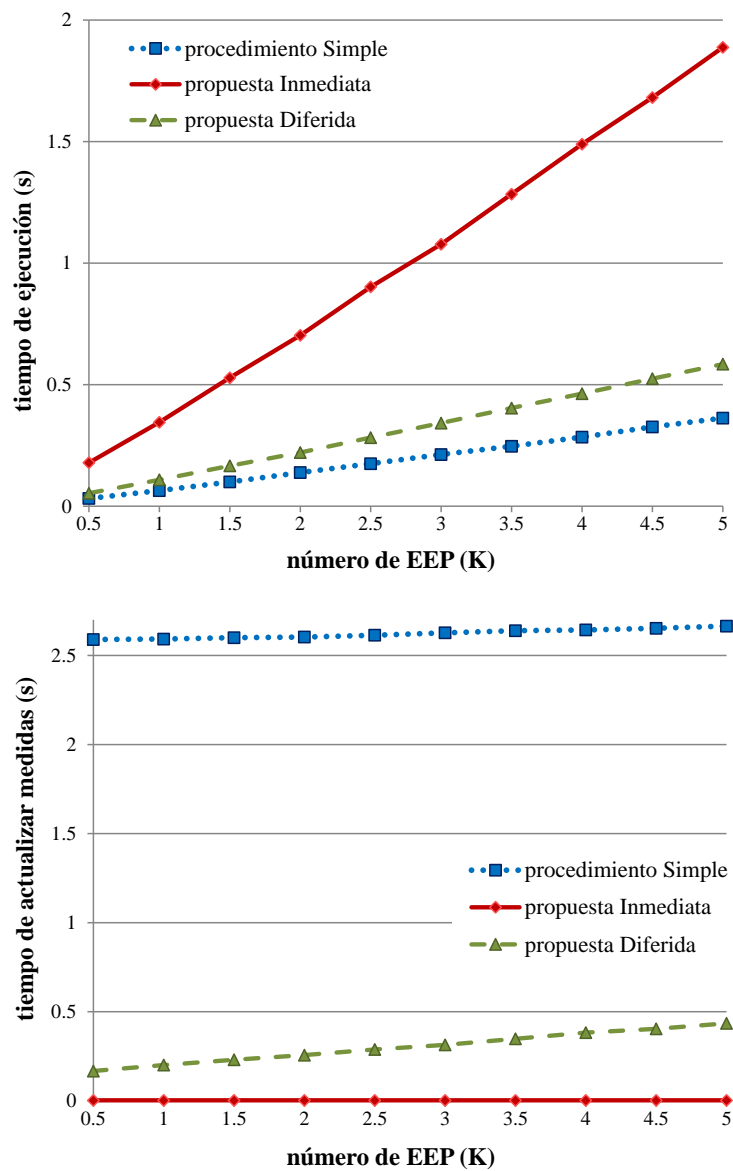


Figura 4.13: Desempeño de las propuestas en el mantenimiento de RAD. Tiempo ejecución (arriba) y tiempo de actualización de las medidas (abajo) en PostgreSQL.

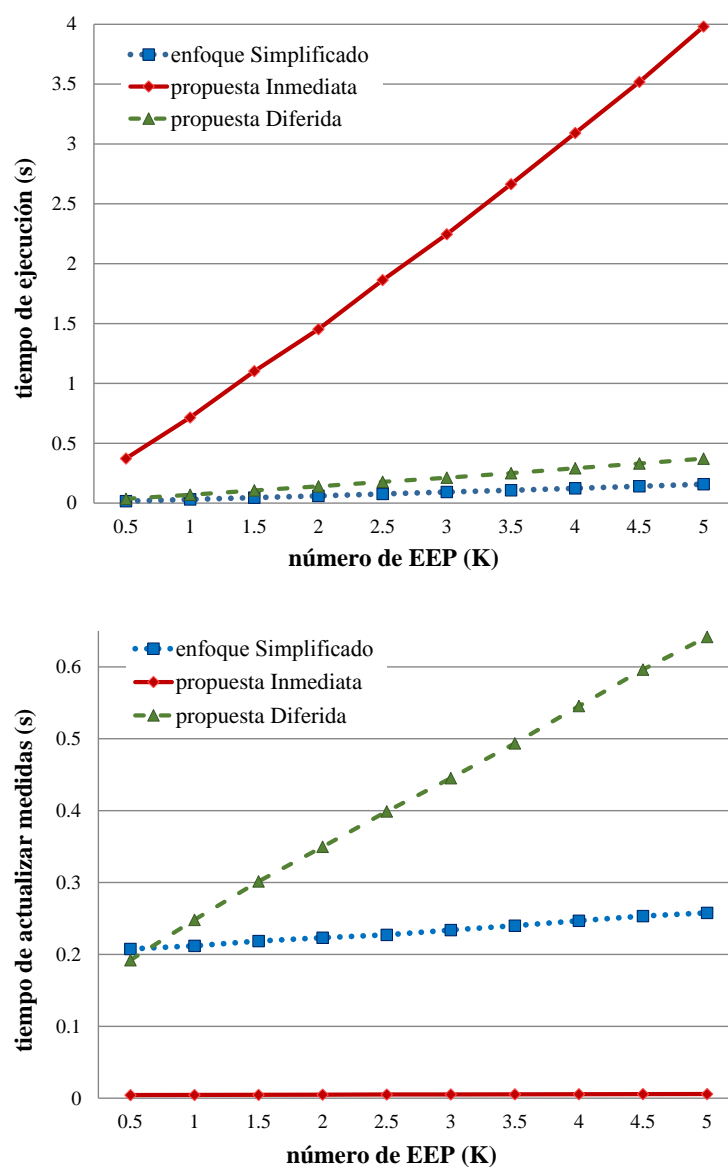


Figura 4.14: Desempeño de las propuestas en el mantenimiento de DA. Tiempo ejecución (arriba) y tiempo de actualización de las medidas (abajo) en MySQL.

Además de las propuestas inmediata y diferida se implementó el enfoque Simplificado. Este enfoque indica, en el tiempo de ejecución (arriba), el tiempo mínimo que toma realizar los EEP puesto que no contiene ningún recurso activo. Comparando su diferencia con el resto, se puede observar la carga adicional que nuestros algoritmos imponen a las operaciones regulares de la base de datos. En este caso, la propuesta inmediata supera la propuesta diferida. La diferencia más marcada se observa en las dependencias aproximadas, donde la propuesta



activa necesita acceder a las complejas estructuras [MDA](#). Es de señalar que este comportamiento superior de la propuesta diferida, tal como se analizó en la complejidad temporal [4.6](#), se degrada para grandes cantidades de [EEP](#). Por este motivo la propuesta diferida debe considerar actualizar las medidas periódicamente si no existe una demanda frecuente de las medidas, o considerar aspectos de implementación en los [SGBDRA](#) que incluyan el procesamiento eficiente de grandes relaciones.

Por otro lado, el tiempo de actualización de las medidas (abajo) en el método inmediato es prácticamente instantáneo, y no depende de la cantidad de eventos estructurales. En contraposición con el método diferido, que necesita actualizar las medidas a partir de las instancias relevantes de las relaciones auxiliares y sí depende de la cantidad de eventos. Este comportamiento del método diferido no es preocupante puesto que se realiza bajo poco consumo de tiempo. Debe señalarse, por demás, que las relaciones auxiliares son vaciadas luego de cada actualización de las medidas, lo que posibilita mantener un bajo consumo para constantes actualizaciones. El enfoque Simplificado tiene un buen comportamiento en este estudio de actualización de medidas, sin embargo, su resultado está estrechamente relacionado con el pequeño tamaño de la base de datos. Para bases de datos mayores es necesario analizar el desempeño de sus consultas.

El análisis del desempeño y la complejidad temporal indican en forma de resumen que:

- **La propuesta inmediata no es aconsejable en escenarios donde la base de datos se encuentre bajo intensas modificaciones.** Esto es debido al aumento del procesamiento en los eventos estructurales, lo cual probablemente comprometa el desempeño de la base de datos en el mantenimiento de las [DA](#).
- **La propuesta diferida es aconsejable en escenarios donde la base de datos se encuentre bajo intensas modificaciones.** Un resultado de que el procesamiento activo no aumente considerablemente el tiempo de las operaciones regulares. Sin embargo, se necesita considerar la frecuente actualización de las medidas bajo demanda o periódicamente.

- **La propuesta inmediata es aconsejable en escenarios donde se necesite consultar frecuentemente las medidas de las reglas.** Esto es producto a su bajo procesamiento en la actualización de las medidas.
- **La propuesta diferida no es aconsejable en escenarios donde se necesite consultar frecuentemente las medidas de las reglas.** Un resultado del procesamiento de las relaciones auxiliares en la actualización de las medidas.

La elección de una propuesta inmediata o diferida debe considerar estos elementos, los cuales dependen en gran medida de las características específicas de los sistemas donde se apliquen. Por ejemplo, en sistemas donde el flujo de datos es constante y su volumen es considerable (*data streams*) [85], es posible que el método diferido sea la mejor opción. Sin embargo, en sistemas donde la toma de decisiones en tiempo real (*real-time decision support systems*) [113, 42] sea el objetivo principal, la propuesta inmediata sea quizás más recomendable. La decisión final por un método u otro, en cualquier caso, debe estudiarse más detalladamente bajo los requerimientos del sistema, e incluso puede considerarse un sistema vivo que autoevalúe cuándo cambiar automáticamente de un algoritmo a otro.

#### 4.7.4 Análisis de escalabilidad

Una característica fundamental de las propuestas realizadas es su autosostenibilidad. Básicamente, esta característica expresa que el desempeño del método inmediato o diferido no depende del tamaño de la base de datos. Para comprobar esta característica se realizó un análisis de escalabilidad donde se mantuvo constante la cantidad de eventos estructurales y se aumentó el tamaño de la base de datos. La Figura 4.15 muestra este resultado para las DA en la base de datos SWAD. El tiempo de ejecución total en estos experimentos de escalabilidad se obtiene mediante la suma de ejecutar 5K eventos estructurales más el tiempo de la actualización posterior de las medidas.

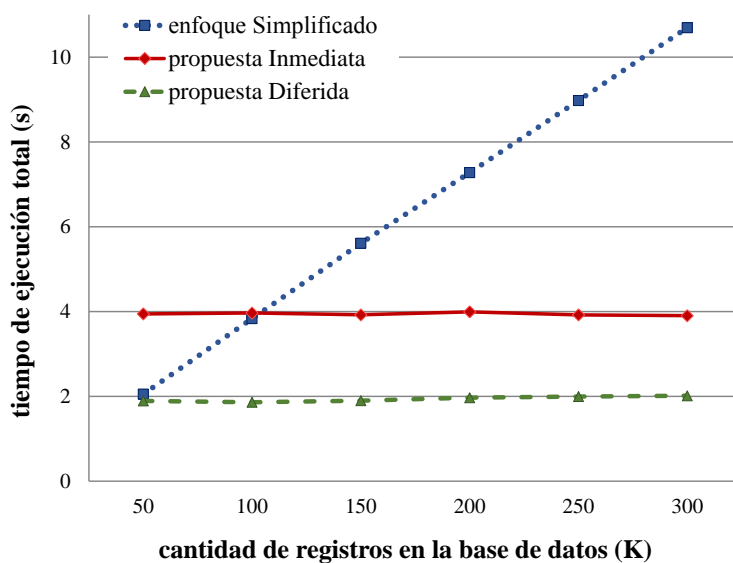


Figura 4.15: Escalabilidad de las propuestas en el mantenimiento de **DA** para MySQL.

Nótese que nuestras propuestas mantienen prácticamente el mismo tiempo de ejecución total, tal como se esperaba. También se ratifica el bajo desempeño del enfoque Simplificado cuando aumenta el tamaño de la base de datos, incluso para bases de datos que pueden ser consideradas de tamaño mediano.

La escalabilidad de nuestras propuestas también se comprobó para las **RA** y **RAD**. A este análisis se añadió el tiempo total de ejecución de un algoritmo de minería convencional “desde cero” y un algoritmo de minería incremental en la propia base de datos SWAD. En el caso de las **RA** se utilizó el algoritmo Apriori como minería convencional. Para la minería incremental, en aras de incluir varios de los algoritmos que utilizan el FP-tree de modo incremental [63, 77, 79, 74], se despreció el tiempo de construcción de esta estructura arbórea (asumiéndose que ha sido incrementalmente mantenida) y se consideró únicamente la ejecución del FP-growth. El resultado se presenta mediante la Figura 4.16.

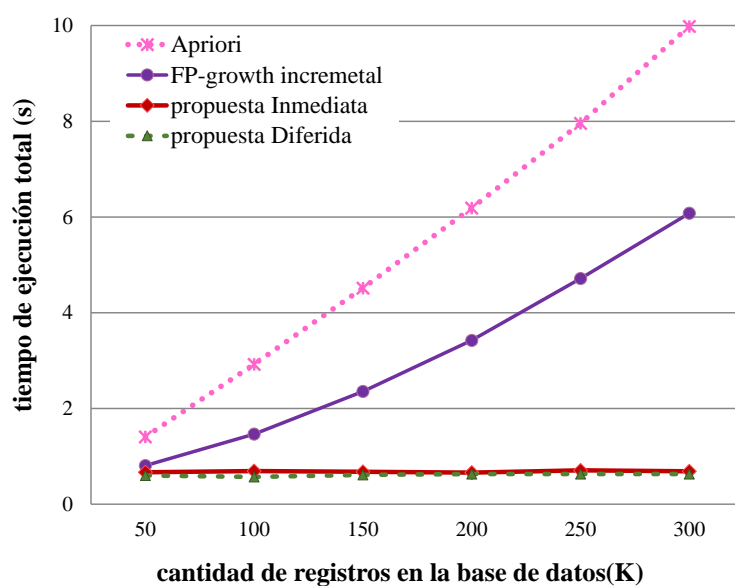


Figura 4.16: Escalabilidad de las propuestas en el mantenimiento de RA; tiempos de ejecución de la minería convencional e incremental.

Un estudio similar se realizó para las RAD. En este caso el algoritmo fuzzy-Apriori representa el método para la minería convencional. Para la minería incremental fue necesario implementar el algoritmo fuzzy FP-growth, variante MFFP-tree [62], en la herramienta KEEL, la cual no cuenta aún con este importante algoritmo. Posteriormente, se utilizó dicho algoritmo para realizar el estudio difuso de la minería incremental, de igual forma que su homólogo *crisp*; despreciando de esta forma la construcción del árbol MFFP-tree y considerando únicamente el tiempo de ejecución del MFFP-growth. El resultado se presenta mediante la Figura 4.17.

Este mismo estudio de las RA y RAD se realizó además en otras bases de datos de UCI Irvine. Nuestras propuestas en este estudio mantienen siete RA y siete RAD extraídas de cada una de las bases de datos. La Figura 4.18 y la Figura 4.19 muestran estos resultados para las RA y RAD, respectivamente.

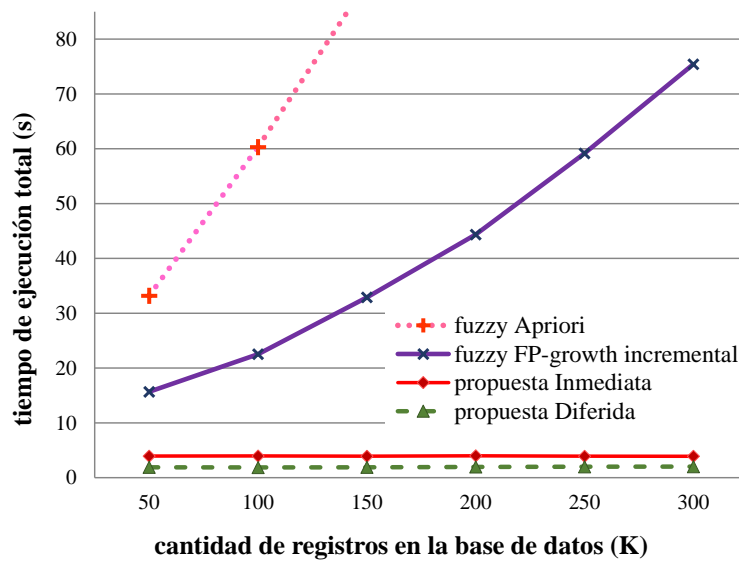


Figura 4.17: Escalabilidad de las propuestas en el mantenimiento de RAD; tiempos de ejecución de la minería convencional e incremental.

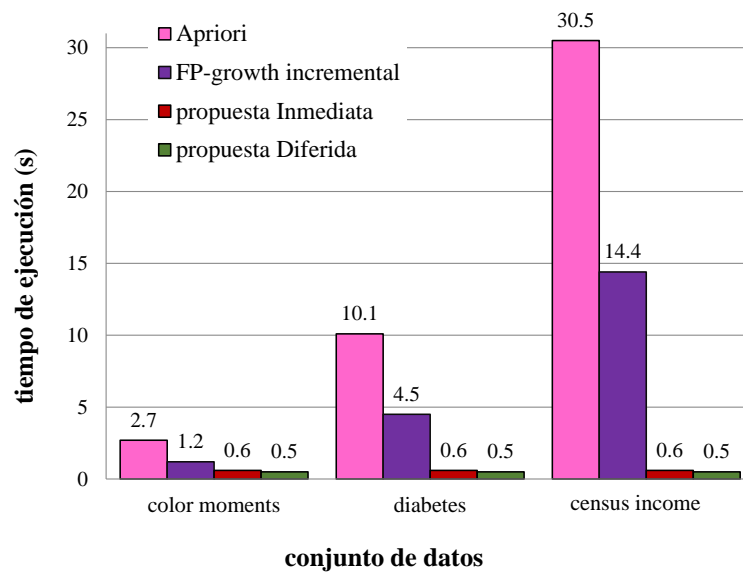


Figura 4.18: Comparación de las propuestas con la minería convencional e incremental en el mantenimiento de RA para diferentes conjuntos de datos.

Se puede concluir desde la Figura 4.16 y la Figura 4.17 que los algoritmos propuestos superan la ejecución de algoritmos de minería convencional o minería incremental para el mantenimiento de reglas previamente descubiertas. De hecho, esta diferencia se amplía a medida que el tamaño de la base de datos aumenta,

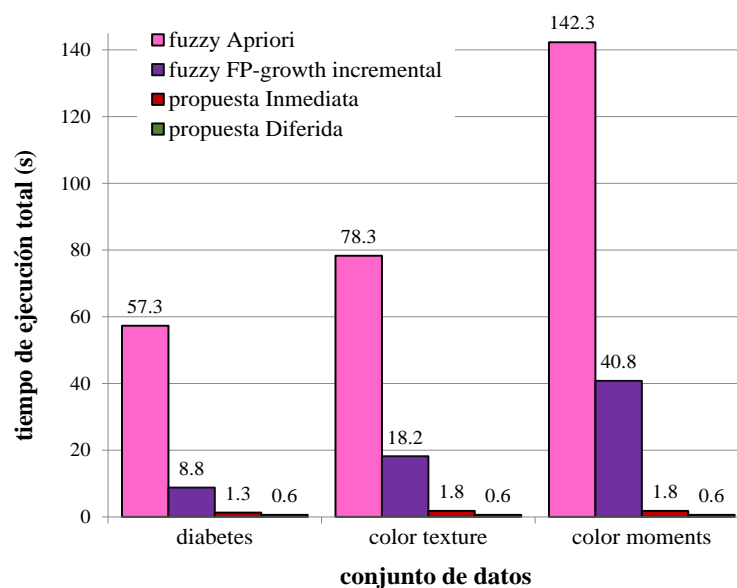


Figura 4.19: Comparación de las propuestas con la minería convencional e incremental en el mantenimiento de RAD para diferentes conjuntos de datos.

en cuya situación nuestras propuestas permanecen prácticamente constantes. La Figura 4.18 y Figura 4.19 afirman este mismo resultado desde otros conjuntos de datos. Además, puede observarse que la propuesta inmediata y diferida poseen casi el mismo tiempo de ejecución total para los diferentes conjuntos de datos. Este resultado demuestra que nuestras propuestas son independientes del universo del discurso donde sean abordadas.

## 4.8 Resumen

El mantenimiento incremental de las reglas previamente extraídas resulta de vital importancia en la toma de decisiones. Especialmente hoy día, donde estas decisiones necesitan ser tomadas en tiempo real. El volumen de datos a analizar para cumplir este objetivo en la minería de datos es inmenso, a lo que se debe agregar el alto tiempo de procesamiento de sus algoritmos.

En este capítulo se han presentado dos propuestas para el mantenimiento incremental de reglas previamente descubiertas, mediante recursos activos de las bases de datos. Una de ellas está enfocada en el mantenimiento inmediato de las

reglas, donde inmediatamente después de ocurrido el evento estructural primitivo en la base de datos, se actualiza la base de reglas. La otra se enfoca en el mantenimiento diferido de las reglas. En una primera etapa de esta propuesta se procesa cada evento estructural primitivo y se extraen las instancias relevantes de la transición. En una segunda etapa diferida se evalúan las modificaciones que estas instancias provocan en la base de reglas.

Un aspecto analizado en las propuestas inmediata e incremental fue su desempeño en la base de datos. Por un lado, se analizó el procesamiento de los recursos activos y por otro, se analizó la actualización de las medidas en la base de reglas. Los resultados de estos experimentos permitieron definir bajo qué características deben ser elegidas nuestras propuestas.

Otro aspecto analizado fue la escalabilidad de ambas propuestas. Debido a la autosostenibilidad de nuestros métodos la base de datos nunca es explorada. Esto permite aislar los algoritmos presentados del volumen existente en la base de datos, lo que los convierte en opciones interesantes para grandes bases de datos. En este estudio se compararon además, las propuestas inmediata y diferida con los algoritmos de minería convencional e incremental. Como resultado se obtuvieron, en todos los casos, mejores tiempos de ejecución total para nuestros algoritmos; superando los algoritmos convencionales e incrementales para el mantenimiento de reglas previamente descubiertas.

## Herramienta para el mantenimiento incremental de reglas mediante bases de datos activas: DRIMS

Si bien las propuestas realizadas en la presente memoria mejoran el mantenimiento del conocimiento previamente extraído, su implementación y gestión en los sistemas reales no es un proceso trivial. En el presente capítulo se realizará una descripción del Sistema para el Mantenimiento Incremental de Reglas de Datos, el cual nombramos mediante sus siglas anglosajonas *Data Rules Incremental Maintenance System* (DRIMS). Esta herramienta resume todo el trabajo desarrollado en la presente memoria. Así, se obtiene un resultado práctico para los desarrolladores de sistemas, el cual les permite gestionar las reglas previamente extraídas e implementar su mantenimiento incremental en los **Sistema Gestor de Bases de Datos Relacionales Activas (SGBDRA)**. Por otro lado, DRIMS es una herramienta de código libre disponible en la plataforma GitHub<sup>1</sup>, por lo que cualquier usuario puede acceder a ella y colaborar o contribuir como considere necesario.

---

<sup>1</sup><https://github.com/AlainPerez/DRIMS-Repository>.



El capítulo se organiza de la siguiente forma. Primeramente, se presenta una sección relacionada con el análisis y diseño de la herramienta mediante su arquitectura. Este diseño se divide en una visión estática y dinámica del sistema para una mejor comprensión del mismo. Posteriormente, se cuenta con una sección que describe la interfaz del usuario. En la misma se indican las ventanas principales de DRIMS mediante su barra de menús y sus asistentes para crear e implementar las reglas.

## 5.1 Arquitectura del sistema

En la presente sección se muestran los aspectos más relevantes del diseño e implementación de DRIMS. Estos aspectos son descritos desde una visión estática y dinámica para su mejor comprensión. La visión estática se encuentra orientada a representar los principales componentes de DRIMS desde una perspectiva estructural, mientras que la dinámica puntualiza cómo estos componentes se comportan e interactúan entre sí. El lenguaje gráfico elegido para estas definiciones es *Unified Modeling Language* (UML) [96], el cual es uno de los más conocidos y utilizados en la actualidad.

### 5.1.1 Visión estática

La herramienta de software DRIMS se encuentra dividida por varios componentes principales en su arquitectura. Estos componentes son la interfaz de usuario, la base de reglas, el compilador y la base de datos del negocio. Dichos componentes físicos interactúan entre sí, brindándole funcionalidad al sistema como un todo. Esta organización, vista desde un alto nivel de abstracción, se muestra mediante la Figura 5.1.

El lenguaje de programación orientado a objetos elegido para implementar la herramienta es Java. Esto permite ejecutar la aplicación en cualquier máquina virtual de Java, independientemente de la arquitectura de computadora subyacente [11].

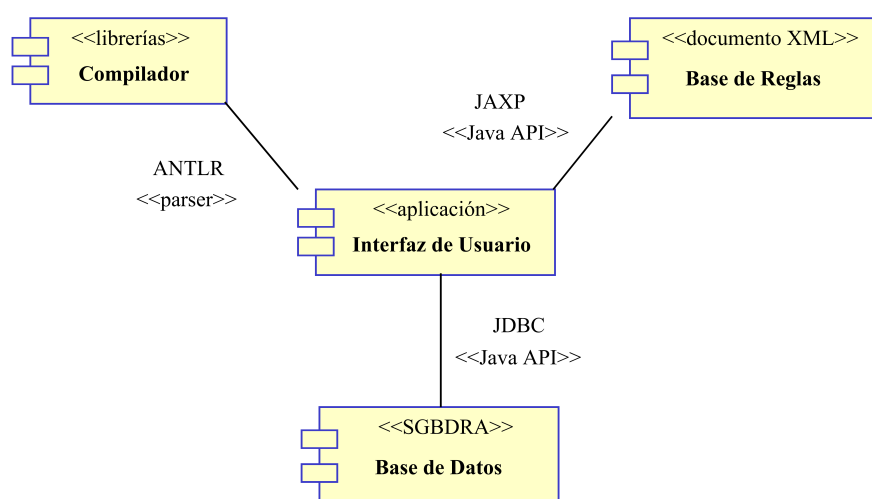


Figura 5.1: Diagrama de componentes de la herramienta.

La base de reglas por su parte, se estableció mediante documentos *eXtensible Markup Language* (XML). Este lenguaje de marcas es un estándar de intercambio de información estructurada, útil para varias plataformas de forma segura y fiable [17]. Su empleo responde también a aspectos vinculados a las **Reglas de Negocio** (RN), tratados en la Sección 3.3. La interfaz de usuario se comunica con este tipo de documentos y los valida mediante *Java API for XML Processing* (JAXP). Para esta validación se utiliza un documento esquema *XML Schema Definition* (XSD), el cual se presenta parcialmente mediante la Figura 5.2. Esta descripción de la estructura de la base de reglas muestra la información almacenada para los diferentes tipos de reglas.

Las reglas almacenadas en la base de reglas son previamente analizadas gramaticalmente por un pequeño compilador. Este compilador se encuentra integrado a través de librerías escritas en Java, obtenidas mediante la herramienta *ANother Tool for Language Recognition* (ANTLR), versión 4.0 [101]. Dicha herramienta, utilizando *StringTemplates*, genera el *script* SQL que implementa nuestras propuestas en la **Base de Datos** (BD). Una mirada detallada a la gramática concebida para todas las reglas puede ser realizada mediante el Apéndice C.

Otro componente a considerar es la propia base de datos del negocio. Actualmente DRIMS se encuentra desarrollado para interactuar con los gestores de

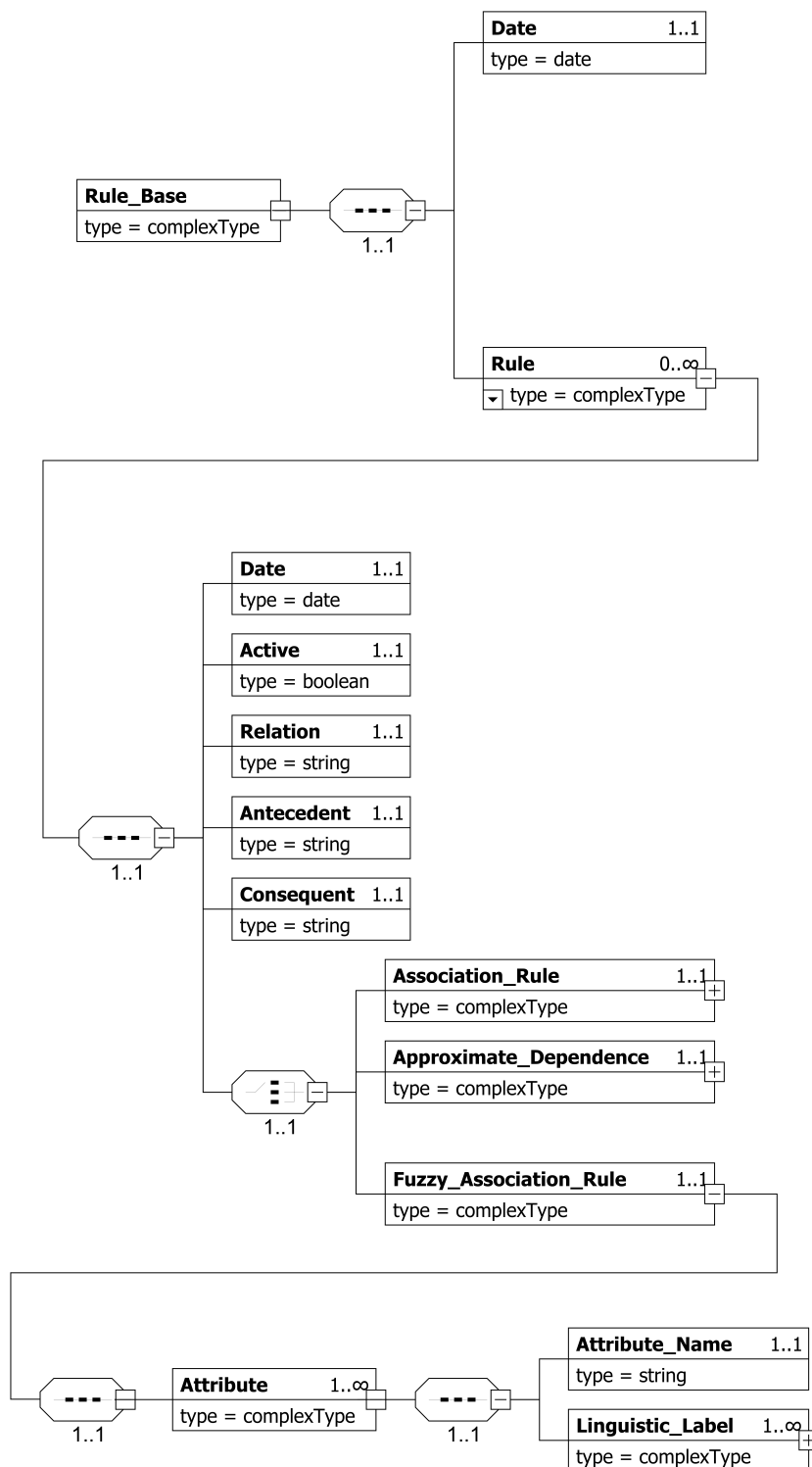


Figura 5.2: Estructura de la base de reglas utilizada por DRIMS.

BD MySQL<sup>®</sup> y PostgreSQL<sup>®</sup>. Sin embargo, toda la comunicación de la interfaz de usuario con la base de datos se realiza a través de la API *Java DataBase Connectivity* (JDBC). Esto permite ampliar la herramienta a casi cualquier SGBDRA con un esfuerzo mínimo, e incluso, considerar diversos orígenes de datos. Mediante esta conexión, la herramienta implementa las propuestas realizadas en la presente memoria para el mantenimiento incremental de las reglas. Además, esta conexión también es utilizada para la extracción de información del catálogo de la BD, y ayudar de esta forma al usuario en la creación de nuevas reglas y su validación.

El patrón de arquitectura adoptado en la implementación de DRIMS fue el modelo vista controlador. Este patrón de arquitectura de software permite separar la interfaz de usuario de los datos y la lógica del negocio mediante la construcción por separado de tres componentes fundamentales: el modelo, la vista y el controlador. Siguiendo este paradigma, junto a la filosofía orientada a objetos, se conformaron las principales clases de la herramienta, las cuales se presentan en la Figura 5.3.

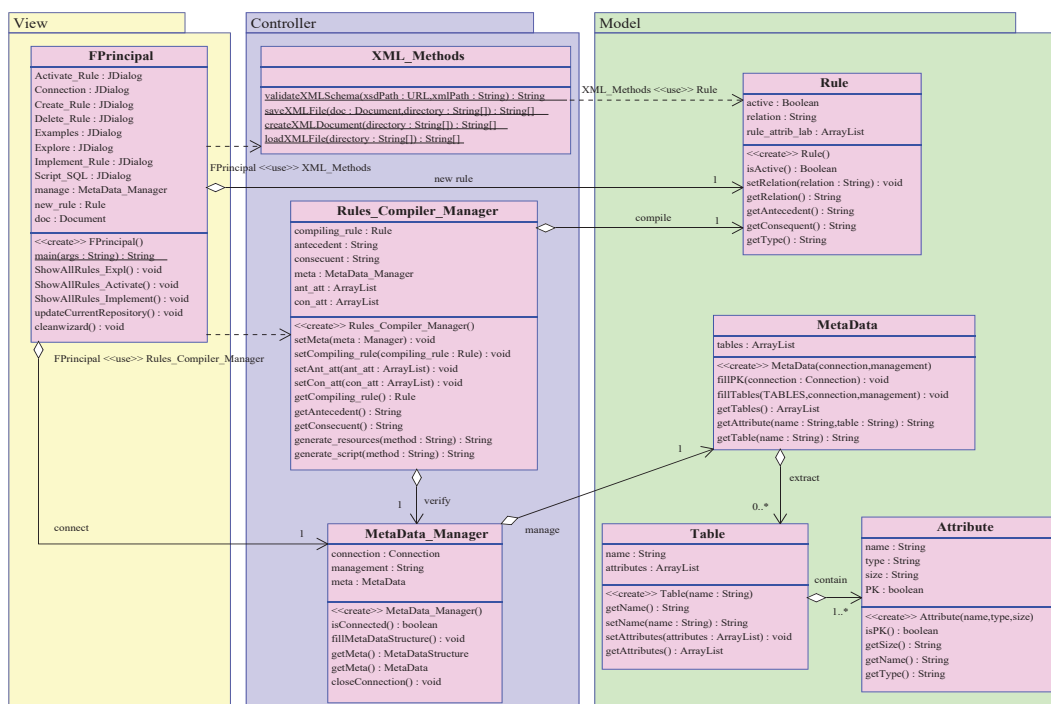


Figura 5.3: Diagrama de clases de la herramienta.

Una breve descripción de estas clases se brinda a continuación:

- ***FPrincipal***, clase encargada de toda la interacción con el usuario mediante *JDialogs*. Al mismo tiempo, contiene los objetos principales que el usuario utiliza.
- ***XML\_Methods***, clase encargada de controlar la base de reglas mediante métodos que interactúan con un documento XML.
- ***Rules\_Compiler\_Manager***, clase encargada de controlar el compilador. Utiliza otras dos clases generadas automáticamente (no incluidas), las cuales contienen el *parser* y el *lexer*.
- ***Data\_Manager***, clase que controla los metadatos principales de la base de datos del negocio. Es utilizada para guiar al usuario en la creación de reglas y para validarlas.
- ***Rule***, clase que define la información necesaria para una regla. De forma genérica incluye las Reglas de Asociación (**RA**), Reglas de Asociación Difusas (**RAD**) y Dependencias Aproximadas (**DA**).
- ***MetaData***, clase que define la información necesaria para almacenar los metadatos principales de la base de datos del negocio. Utiliza la clase ***Table*** para las tablas de la **BD** y esta, a su vez, la clase ***Attribute*** para sus atributos.

Si bien una descripción estática de la herramienta es de gran utilidad para comprender sus componentes principales, ésta es incapaz de reflejar las interacciones existentes entre los mismos. Para ello es necesaria una perspectiva dinámica que ofrezca información acerca de las funcionalidades del sistema como un todo.

### 5.1.2 Visión dinámica

Desde una perspectiva dinámica de la herramienta se puede observar el comportamiento de sus objetos a lo largo del tiempo y cómo éstos interactúan entre sí. Una vista general y simple de los casos de uso es ilustrada mediante una notación gráfica en la Figura 5.4. Desde ella se pueden observar las principales funcionalidades brindadas por DRIMS a sus usuarios.

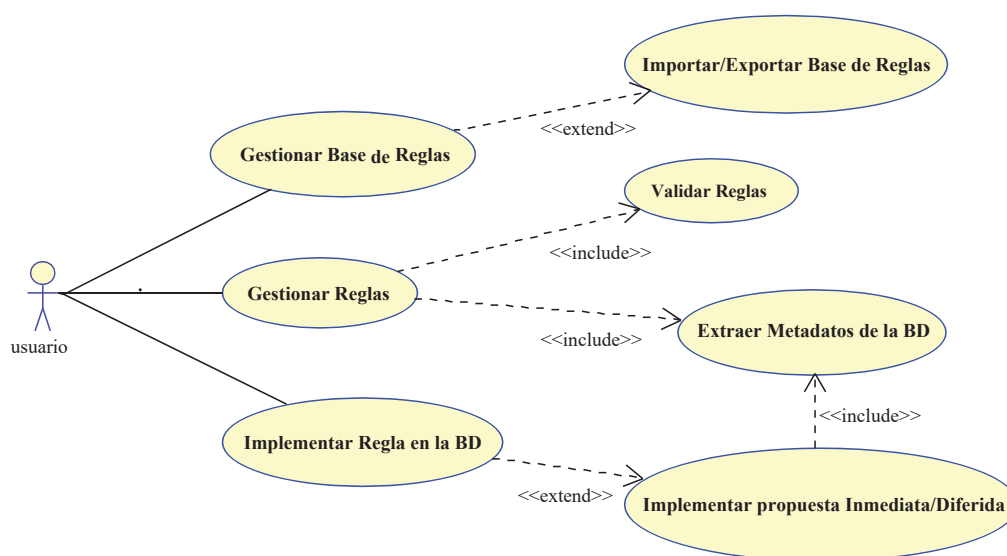


Figura 5.4: Diagrama de casos de uso de la herramienta.

Como se puede observar, el sistema ofrece funciones vinculadas a la implementación de los algoritmos propuestos en la **BD**. Sin embargo, no se limita únicamente a este objetivo, sino que además extiende casos de uso relacionados con la gestión de la base de reglas. Estas reglas almacenadas son también el resultado de prestaciones ofrecidas por DRIMS. Para la creación de estas reglas y su validación se realizan búsquedas en el catálogo de la **BD**.

Por otro lado, la herramienta desarrollada presta una especial atención al estado en que se encuentra la base de reglas. Estos estados le brindan al usuario el resultado de sus acciones sobre la base de reglas, evitando de esta forma que se pierda información en la misma. Mediante la Figura 5.5 se muestran los posibles estados por los cuales transita la base de reglas en respuesta a diferentes eventos.

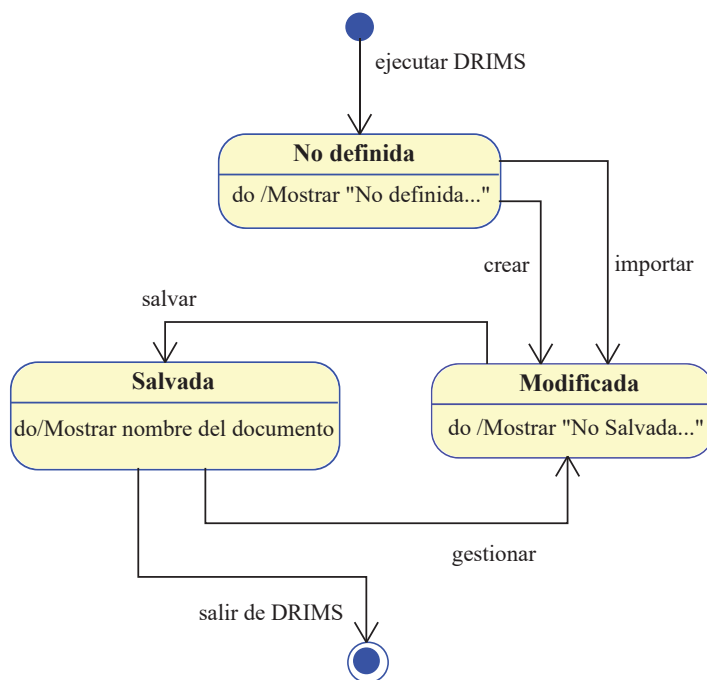


Figura 5.5: Diagrama de estados de la base de reglas en el sistema.

El comportamiento general del sistema contiene una amplia cantidad de actividades, secuencias e interacciones. Por esta razón, consideramos que resulta más interactivo mostrar gran parte de esta información a través de una descripción de la interfaz de usuario. A su vez, esto permite una forma de asistencia primaria a los usuarios de la herramienta.

## 5.2 Casos de estudio en la interfaz de usuario

A través de una interfaz gráfica se ha pretendido que los usuarios sean capaces de implementar los algoritmos inmediato y diferido directamente en las BD de sus negocios. Varias de las concepciones utilizadas en esta interfaz encuentran su origen en los sistemas orientados a RN [25, 4], expuestas en la Sección 3.3. Esta sección está encaminada, luego de exponer una breve descripción general, a presentar dos casos de estudio en dicha interfaz que muestren ejemplos reales de su funcionamiento.

Tras ejecutar el programa, lo primero que distinguiremos será su ventana principal, la cual se muestra en la Figura 5.6. En este momento no se podrá acceder a mayores funcionalidades hasta tanto no se establezca una base de reglas sobre la cual se gestionan los tipos de reglas manejados en esta memoria: RA, RAD y DA.

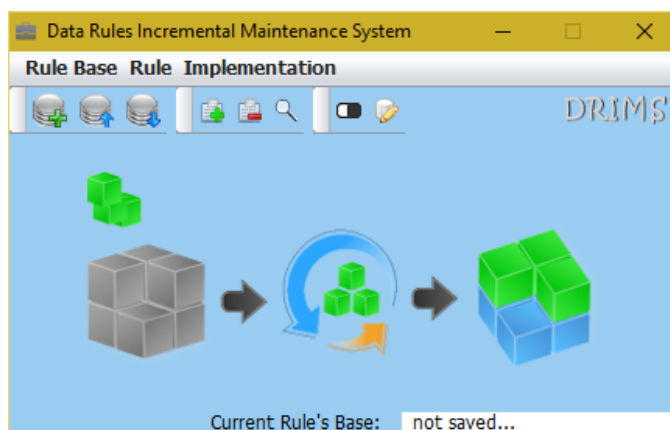




Figura 5.6: Ventana principal de DRIMS.








Inicialmente se distinguen dos zonas principales. La barra de menús y la barra de herramientas con los comandos más utilizados. Los estados de la base de reglas presentados en la Figura 5.5 también se pueden localizar desde la ventana principal, en la esquina inferior derecha.

### 5.2.1 Barra de menús

En la parte superior de la ventana principal, Figura 5.6, se encuentra la barra de menús. Desde la misma se acceden a todas las funcionalidades de la herramienta, aunque de forma más directa se muestran sus comandos más útiles. Los menús se desglosan en los siguientes conjuntos de operaciones:

- **Menú *Rule Base*** (Base de Reglas), permite gestionar la base de reglas y salir de la aplicación mediante las operaciones:
  -  *New* (Nueva), crea una nueva base de reglas de extensión *.xml*.
  -  *Open* (Abrir), abre una base de reglas previamente creada, según el esquema mostrado en la Figura 5.2.



-  *Save* (Salvar), salva la base de reglas que la herramienta tenga activada.
  -  *Exit* (Salir), termina la sesión actual y sale del sistema.
- **Menú *Rule*** (Regla), permite gestionar las reglas almacenadas en la base de reglas activa mediante las operaciones:
-  *Create* (Crear), crea una **RA**, **RAD** o **DA** en la base de reglas. Esta operación se realiza mediante un asistente que les facilita la labor a los usuarios.
  -  *Delete* (Eliminar), elimina una regla de la base de reglas que se encuentra activa. Esta operación se desplaza por la base de reglas y elige la regla a eliminar.
  -  *Explore* (Explorar), permite explorar la base de reglas en busca de reglas.
- **Menú *Implementation*** (Implementación), permite gestionar la base de reglas y salir de la aplicación mediante las operaciones:
-  *Activate/Desactivate Rule* (Activar/Desactivar Regla), activa o desactiva una regla. Una regla que no se encuentre activa en la base de reglas no se puede implementar en el gestor.
  -  *Generate SQL Script* (Generar SQL Script), genera el *script* SQL que implementa el algoritmo para el mantenimiento incremental inmediato o diferido de las reglas previamente descubiertas en la **BD**.

### 5.2.2 Caso de estudio del asistente para nuevas reglas

Una de las principales funcionalidades que brinda la herramienta es la posibilidad de crear nuevas reglas en la base de reglas. Esta operación se realiza a través de un asistente especialmente creado para facilitarle este proceso a los usuarios. Para ilustrar su comportamiento presentamos un caso de estudio a partir de la creación de una regla extraída del SWAD, expuesta en el Apéndice **B**.

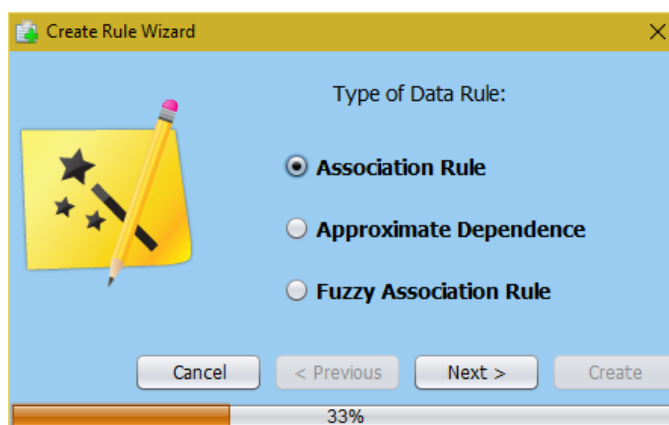




Figura 5.7: Inicio del asistente para crear reglas.

Inicialmente el asistente pregunta por el tipo de regla que se desea crear, como se muestra en la Figura 5.7. A lo largo de este asistente se muestra el progreso del mismo, además de opciones para adelantar o regresar un paso en las definiciones.

A continuación, el asistente brinda facilidades para elegir la relación base sobre la cual se encuentra establecida la regla. Esta relación puede ser simplemente escrita, aunque el usuario tiene la opción de conectarse  a la propia base de datos y a su catálogo. Dicha conexión extrae los metadatos relacionados a la creación total de la regla mediante los cuales se asiste al usuario en varias actividades de este proceso. Ambas funcionalidades se ilustran en la Figura 5.8.

Por último, se crea la regla en la ventana que muestra la Figura 5.9. Como se puede notar, existen dos áreas de edición destinadas a introducir el antecedente y el consecuente de la regla. Para ello se facilitan los operadores definidos en la sintaxis de la gramática (ver Apéndice C) y los atributos (junto a su tipo) presentes en la relación base, si se definió esta última. Conjuntamente, se presentan ejemplos  de los diferentes tipos de reglas y de la utilización de los operadores.

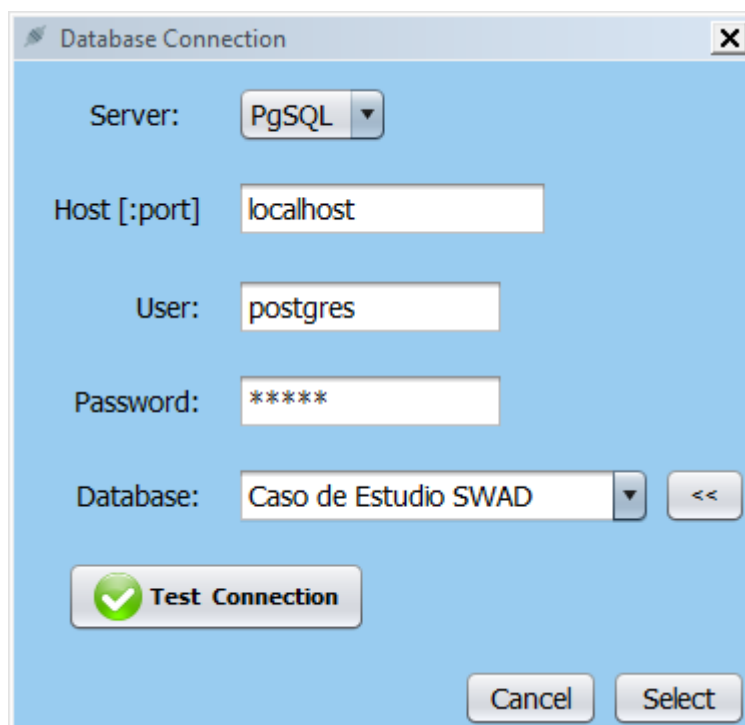
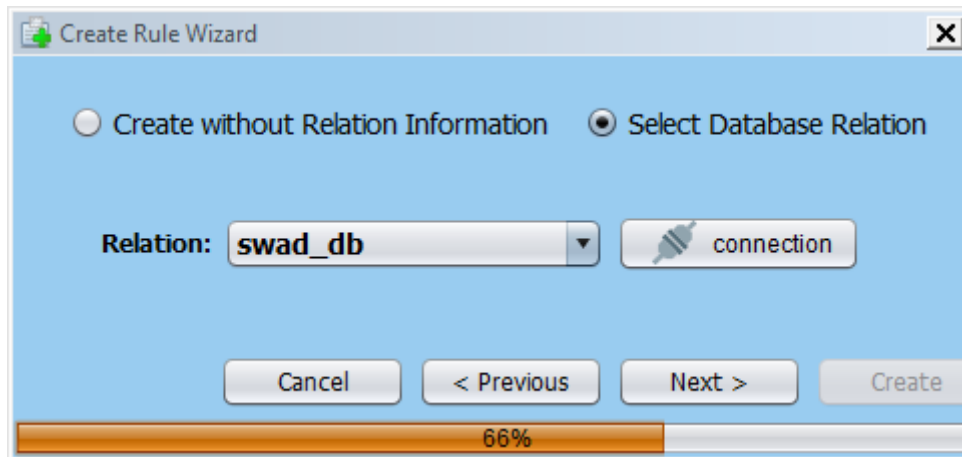


Figura 5.8: Elección de la relación base para nuevas reglas.

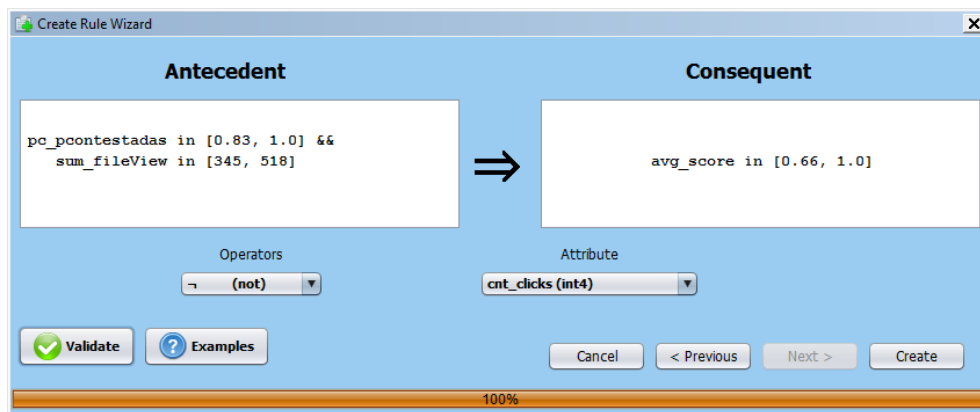







Figura 5.9: Fin del asistente para crear reglas.

En DRIMS es posible crear RA positivas y negativas [72] mediante el operador lógico de negación ( $\neg$ ) y además, expresar reglas más complejas con el operador lógico de disyunción. Luego de definir la regla es necesario validarla , siguiendo la sintaxis de la gramática para el tipo de regla. Si la regla expresada se valida correctamente se habilita la opción de crearla, quedando almacenada de esta forma en la base de reglas la cual aparecerá como modificada en la ventana principal.

### 5.2.3 Caso de estudio del asistente para implementar reglas

Otra funcionalidad indispensable en DRIMS es la implementación del mantenimiento incremental de las reglas existentes. Continuando con el caso de estudio previo, se muestra el caso de estudio para la implementación de la misma regla. Para este paso inicialmente se debe comprobar si dicha regla se encuentra activada . El proceso de implementación  puede estar vinculado a varias reglas, para ello se permite navegar entre las reglas existentes como muestra la Figura 5.10.

En esta ventana, luego de establecida la conexión  de forma similar a la expuesta en la sección precedente, se habilita la posibilidad de elegir una propuesta de mantenimiento incremental. En este instante, se presentan las propuestas inmediata y diferida, ambas ampliamente discutidas en la presente memoria. Luego se pasa a generar  el *script*, el cual provee todo el código SQL que

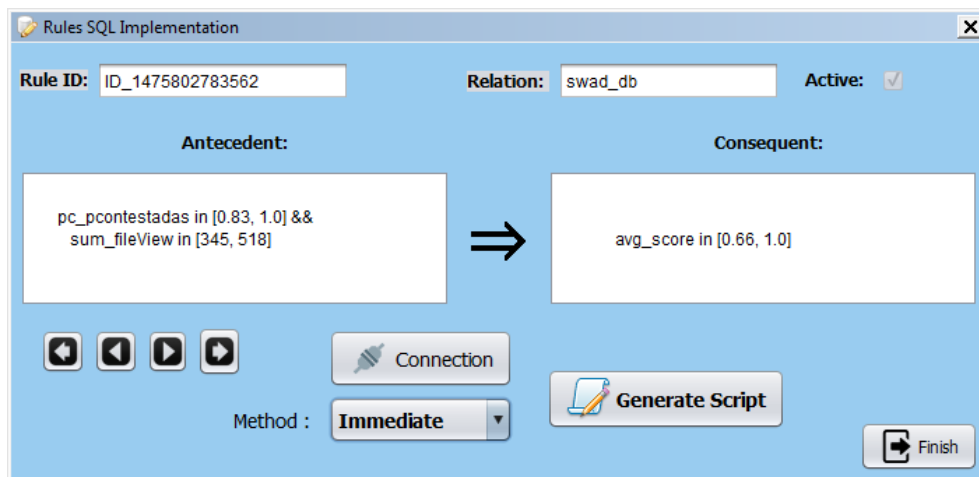


Figura 5.10: Implementación de las propuestas inmediata y diferida en DRIMS.

implementa la propuesta seleccionada, como refleja la Figura 5.11. Actualmente esta implementación se encuentra limitada a mantener las reglas únicamente bajo el factor de certeza. Sin embargo, no es relativamente difícil añadir al menos otras medidas que involucren el soporte del antecedente, el soporte del consecuente y el soporte del antecedente  $\cup$  consecuente.

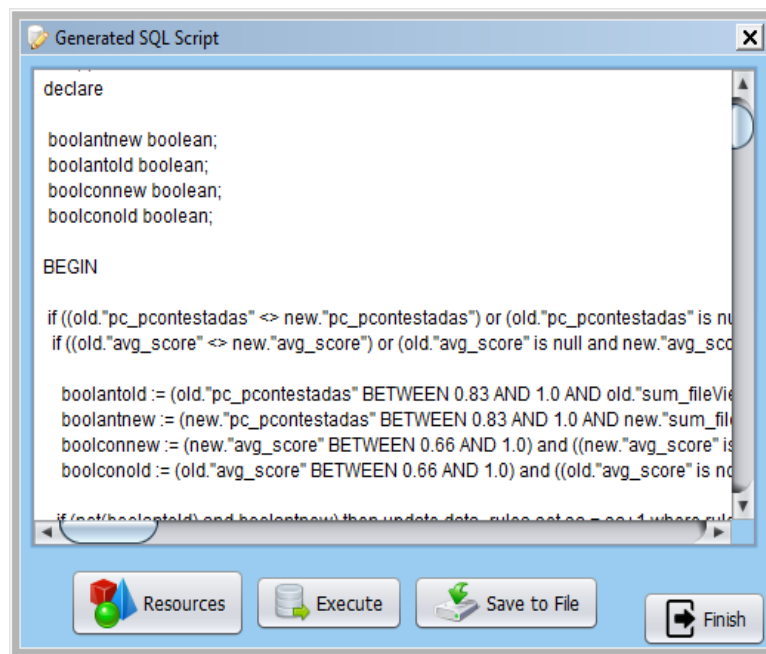


Figura 5.11: *Script* SQL que implementa las propuestas inmediata y diferida en el sistema.

Existen tres opciones que nos brinda la herramienta con el *script* generado. La primera es comprobar los recursos 🗄️ de la **BD** utilizados en la implementación, los cuales se encuentran en el catálogo, como se expresó en la Sección 4.3. Dentro de ellos, los principales son los disparadores expuestos en la Sección 3.4, aunque también se crean tablas, vistas, procedimientos y funciones, entre otros recursos. Una segunda opción es ejecutar 📄 el *script* directamente en la base de datos mediante una conexión 🌐. La tercera y última opción es guardar el código SQL. Esta copia se realiza en un documento de texto con extensión .sql, el cual se puede ejecutar posteriormente en el propio gestor de base de datos del negocio.

## 5.3 Resumen

El presente capítulo se ha dedicado a la descripción de la herramienta de software DRIMS. Este sistema provee funcionalidades para definir y gestionar sus propias bases de reglas. La gestión se realiza a partir de documentos útiles para su integración con varias plataformas de forma segura y fiable. De forma general, la arquitectura de DRIMS es lo suficientemente flexible como para permitir añadir nuevas funcionalidades, así como para extender las actualmente desarrolladas. Su implementación en un lenguaje de propósito general, ampliamente utilizado y con la mayoría de sus tecnologías bajo licencia pública, apoya este criterio.

Esta herramienta se encuentra orientada a la implementación de los algoritmos propuestos por la presente memoria en los **SGBDRA**, mediante una interfaz de usuario sencilla y de fácil manejo. Como resultado práctico de la actual investigación DRIMS agrupa y resume todos los aspectos abordados en los capítulos precedentes. Finalmente, DRIMS es un sistema de código abierto y libre que se encuentra en continuo período de actualización.



## Conclusiones y trabajos futuros

Este capítulo está dedicado a la presentación de un resumen de los resultados obtenidos por esta memoria. Se presentan las publicaciones asociadas a esta tesis y se comentan las líneas de trabajo futuras que quedan abiertas tras el estudio realizado.

### 6.1 Conclusiones

La investigación realizada en esta memoria se ha centrado, principalmente, en desarrollar y evaluar nuevos algoritmos para el mantenimiento incremental de Reglas de Asociación (**RA**), Reglas de Asociación Difusas (**RAD**) y Dependencias Aproximadas (**DA**) mediante recursos de Bases de Datos Activas (**BDA**). Estos algoritmos han sido diseñados para considerar la gran variedad de métricas existentes en la literatura. Los objetivos propuestos para lograr nuestro propósito principal y que fueron desarrollados a lo largo de esta tesis son:

- Realizar un estudio del estado del arte en el descubrimiento y mantenimiento de las **RA**, **RAD** y **DA**.
- Analizar los vínculos existentes entre el mantenimiento incremental en **BDA** y el mantenimiento incremental de las reglas abordadas.



- Proponer nuevos algoritmos para el mantenimiento incremental de **RA**, **RAD** y **DA** que integren distintas medidas de interés.
- Implementar una herramienta de software para el mantenimiento incremental de reglas en **BDA**.

Con respecto al primer objetivo, la conducción de un estudio del estado del arte en el descubrimiento de reglas de asociación junto a sus extensiones difusas y aproximadas, mostró la gran variedad de algoritmos y métricas existentes en la minería de estas reglas. Dentro de estos algoritmos destacan Apriori [2] y FP-Growth [58]. Por otro lado, se relató la gran variedad de medidas de interés existentes y se observaron las relaciones que estas métricas guardaban entre sí.

Este objetivo también permitió observar el mantenimiento incremental de las reglas desde una perspectiva de extracción del conocimiento: la minería incremental. Dentro de los algoritmos más representativos en este área se expuso el algoritmo *Fast Update algorithm* (FUP) [29], el algoritmo FUP-tree [63] para las **RA** y el algoritmo MFFP-tree [64] para las **RAD**.

En relación al segundo objetivo, el estudio de los fundamentos de las bases de datos activas permitió profundizar en sus notaciones y conceptos. En este punto, se detalló el modelo evento-condición-acción, el cual establece un comportamiento reactivo. Este modelo es ampliamente utilizado para el mantenimiento de Vistas Materializadas (**VM**), el chequeo de Restricciones de Integridad (**RI**) y las **Reglas de Negocio (RN)**.

Para cada uno de estos campos de investigación se establecieron los puntos en común con las reglas de asociación y sus extensiones difusas y aproximadas. Estos vínculos permitieron utilizar métodos y algoritmos de mantenimiento incremental provistos por las **VM**, **RI** y **RN**, en el mantenimiento incremental de las reglas estudiadas.

Por otro lado se presentó, desde este mismo objetivo, los recursos que las **BDA** proveen para el mantenimiento incremental. Dichos recursos se estudiaron desde el estándar SQL, lo cual permite su implementación en una gran mayoría de los actuales Sistemas Gestores de Bases de Datos Relacionales Activas (**SGBDRA**).

Para cumplir con el cuarto objetivo se aplicó el conocimiento adquirido en los objetivos precedentes. De esta forma se propusieron dos nuevos algoritmos para el mantenimiento incremental de RA, RAD y DA mediante BDA. Además, se presentó un enfoque Simplificado como una solución intuitiva inicial a nuestro problema, siendo utilizado como punto de referencia para los algoritmos propuestos.

Estas soluciones fueron diseñadas para ser inclusivas en cuanto a la gran variedad de métricas existentes. Para ello, se dividió una medida en un conjunto de partes más sencillas de mantener. Aun cuando estas partes de la medida deben cumplir una serie de propiedades, generalmente se pueden definir intuitivamente de la propia fórmula de la medida.

La propuesta de nuevos algoritmos contiene, implícitamente, un estudio experimental de los mismos. De esta forma, se diseñaron experimentos específicos que aumentarían los eventos estructurales. Esto permitió la evaluación de sus comportamientos en entornos de flujos de datos. Los resultados arrojaron que nuestros algoritmos tienen un buen comportamiento y que uno u otro debe ser elegido según los requisitos del sistema. También se diseñaron experimentos de escalabilidad cuyos resultados demostraron que nuestras propuestas superan los enfoques de minería convencional e incremental para el mantenimiento de las reglas. En general, ambos resultados obtenidos, apoyan el empleo de nuestras propuestas para las ingentes aplicaciones de apoyo a la toma de decisiones en tiempo real [113, 42], en entornos de flujos de datos [85] y grandes volúmenes de datos [29]. Asimismo, se pueden emplear en el análisis temporal de las reglas [15].

Con respecto al quinto objetivo se desarrolló una herramienta en Java que es capaz de implementar los algoritmos propuestos en las BDA. Esta herramienta, la cual nombramos *Data Rules Incremental Maintenance System* (DRIMS), es el resultado práctico del cumplimiento de todos los objetivos propuestos.

La arquitectura de la herramienta se mostró mediante una visión estática y dinámica. Entre sus componentes estáticos se tienen: la base de reglas, el compilador, la base de datos del negocio y la interfaz de usuario. Estos componentes interactúan entre sí para brindar varias funcionalidades al sistema. Una de ellas es la gestión de la base de reglas, la cual está concebida para brindar portabilidad sobre varias plataformas. Otra funcionalidad que destaca es la generación del *script* SQL que implementa nuestros algoritmos. Este código SQL utiliza los recursos activos y puede ejecutarse directamente (para la versión actual de la herramienta) en los [SGBDRA PostgreSQL](#) y [MySQL](#). Actualmente DRIMS continúa transitando por un proceso de perfeccionamiento.

## 6.2 Publicaciones asociadas a la memoria

A continuación, se presenta un listado de las publicaciones, artículos desarrollados y premios que se encuentran asociados con la presente memoria. Estos resultados se encuentran organizados según los capítulos.

- Capítulo 3. De este capítulo se heredan los resultados investigativos en Reglas de Negocio llevados a cabo en el grupo de investigación de Bases de Datos de la Universidad Central “Marta Abreu” de Las Villas.
  - Revista Indexada: M. B. B. Castillo, A. P. Toledo, A. P. Alonso, L. M. G. González, and R. P. Vázquez. Inserción automática de reglas de negocio en bases de datos. *Revista Técnica de la Facultad de Ingeniería. Universidad del Zulia*, 36(3):9, 2013. (revista indexada por la Journal Citation Reports 2013).
  - Publicación en Congreso: A. Pérez-Alonso, A. P. Toledo, M. B. Boggiano and L. M. González-González. Generación de restricciones de integridad en bases de datos relacionales. In *II Conferencia Internacional en Ciencias Computacionales e Informáticas, Informática2013*, La Habana, Cuba. 2013.

- Premio: Premio Anual Provincial de la Academia de Ciencias de Cuba. *Enfoque de reglas de negocio y sus implementaciones en bases de datos relacionales*, Ministerio de Ciencia Tecnología y Medio Ambiente, 2013.
- Capítulo 4. De este capítulo se obtienen los resultados vinculados a los algoritmos propuestos en la presente memoria.
  - Revista Indexada: A. Pérez-Alonso, I. Blanco, L. M. González-González, and J. M. Serrano. Incremental maintenance of discovered association rules and approximate dependencies. *Intelligent Data Analysis An International Journal*, 21(1), 2016. (revista indexada por la Journal Citation Reports 2016).
  - Publicación en colecciones: A. Pérez-Alonso, I. Blanco, L. M. González-González, and J. M. Serrano. Mantenimiento Incremental directo de Reglas de Asociación existentes. In D. A. Pelta, R. Pérez-Rodríguez, C. C. Corona, and J. L. Verdegay, editors, *Contribuciones en Soft Computing*, pages 35-56. Editorial Universidad de Granada, 2014.
  - Artículo sometido: A. Pérez-Alonso, I. Blanco, L. M. González-González, and J. M. Serrano. Incremental Maintenance of Discovered Fuzzy Association Rules. *Journal of Intelligent Information Systems*.
  - Premio: Premio Anual Provincial de la Academia de Ciencias de Cuba. *Mantenimiento incremental de reglas en bases de datos activas*, Ministerio de Ciencia Tecnología y Medio Ambiente, 2016.
- Capítulo 5. De este capítulo se obtiene los resultados asociados a la herramienta implementada.
  - Artículo sometido: A. Pérez-Alonso, I. Blanco, L. M. González-González, and J. M. Serrano. DRIMS: A software tool to incrementally maintain previous discovered rules.

## 6.3 Trabajos futuros

Esta última sección está dedicada a ofrecer las nuevas ideas, posibilidades de mejora y líneas de investigación que hemos considerado más interesantes de cara a futuras aportaciones.

- Incluir en nuestro mantenimiento incremental otros tipos de reglas como reglas graduales, reglas de posibilidad, reglas de dependencias aproximadas difusas, nuevas extensiones de reglas de asociación, entre otras.
- Estudiar la aplicación de nuestros algoritmos en otros tipos de bases de datos como las Deductivas-Relacionales-Difusas [89], Orientada a Objetos, Orientada a Documentos, Deductivas, NoSQL y NewSQL.
- Realizar nuevos estudios sobre el comportamiento de los algoritmos propuestos con respecto a la utilización de memoria.
- Ampliar y mejorar la herramienta de software DRIMS:
  - Incorporar la implementación de las propuestas realizadas a nuevos [SGBDRA](#).
  - Diseñar e implementar la posibilidad de que el usuario introduzca las partes de la medida que desea mantener incrementalmente.
  - Desarrollar un módulo que permita establecer diferentes funciones de membresía para las etiquetas lingüísticas de las [RAD](#).
  - Desarrollar un módulo que permita vincular nuestra herramienta con herramientas de extracción del conocimiento como KEEL.

## Sintaxis detallada de los disparadores en el estándar SQL:2011

Inicialmente, basado en el álgebra relacional y el cálculo relacional de tuplas, el SQL consiste en un lenguaje de definición, manipulación y control de datos. Especialmente diseñado para el modelo relacional de Codd originalmente se llamó *Structured English Query Language* (SEQUEL) y fue adoptado como un estándar por la *American National Standards Institute* (ANSI) en 1986. Actualmente la última revisión realizada es la del SQL:2011.

Este apéndice muestra la sintaxis detallada de los disparadores en este último estándar. Este recurso activo es ampliamente utilizado para implementar nuestras propuestas de mantenimiento incremental en las bases de datos activas. Si bien los gestores relacionales presentan diferencias significativas entre sus dialectos, la sintaxis presentada establece perfectamente todos los componentes del modelo evento-condición-acción. Esto favorece ampliamente a la futura implementación de nuestras propuestas en diferentes Sistemas Gestores de Bases de Datos Relacionales Activas ([SGBDRA](#)).

```
<trigger definition>::=
CREATE TRIGGER <trigger name>
<trigger action time><trigger event>
ON <table name>
[ REFERENCING <transition table or variable list>]
<triggered action>

<trigger action time>::=
BEFORE
| AFTER
| INSTEAD OF

<trigger event>::=
INSERT
| DELETE
| UPDATE [ OF <trigger column list>]

<trigger column list>::=
<column name list>

<triggered action>::=
[ FOR EACH { ROW | STATEMENT } ]
[ <triggered when clause>]
<triggered SQL statement>

<triggered when clause>::=
WHEN <left paren><search condition><right paren>

<triggered SQL statement>::=
<SQL procedure statement>
| BEGIN ATOMIC { <SQL procedure statement><semicolon>}
... END

<transition table or variable list>::=
<transition table or variable>...

<transition table or variable>::=
OLD [ ROW ] [ AS ] <old transition variable name>
```

```
| NEW [ ROW ] [ AS ] <new transition variable name>  
| OLD TABLE [ AS ] <old transition table name>  
| NEW TABLE [ AS ] <new transition table name>
```

```
<old transition table name>::=  
  <transition table name>
```

```
<new transition table name>::=  
  <transition table name>
```

```
<transition table name>::=  
  <identifier>
```

```
<old transition variable name>::=  
  <correlation name>
```

```
<new transition variable name>::=  
  <correlation name>
```





## Apéndice **B**

# Reglas extraídas del SWAD y utilizadas en los experimentos

Para evitar entorpecer la lectura de la presente memoria, traspasamos a este apéndice como una única tabla, todas las reglas extraídas del Sistema Web de Apoyo a la Docencia (SWAD). Dichas reglas fueron ampliamente utilizadas en la mayoría de los experimentos que se llevaron a cabo en el Capítulo 4. Los detalles de su extracción se abordan en la Subsección 4.7.2.

Este resultado no debe considerarse propiamente como un conocimiento sólido del estudiantado de la Universidad de Granada (UGR). Hubiésemos deseado contar con respaldos realizados en momentos finales del curso académico, puesto que los datos manejados contienen una gran cantidad de valores desconocidos (nulos).

<b>Tipo</b>	<b>Antecedente</b>	<b>Consecuente</b>	<b>Sop.</b>	<b>FC</b>
RA	<i>sexo</i> ='femenino'	<i>pc_pcontestadas</i> [0.75, 1.0]	0.37	0.75
	<i>pc_pcontestadas</i> [0.85, 1.0]	<i>avg_puntos</i> [0.5, 1.0]	0.65	0.60
	<i>sum_ficheros</i> [259, 518]	<i>avg_puntos</i> [0.5, 1.0]	0.13	0.48
	<i>sexo</i> ='masculino'	<i>cnt_clics</i> [511, 1022]	0.45	0.67
	<i>pc_pcontestadas</i> [0.83, 1.0] $\wedge$ <i>sum_ficheros</i> [345, 518]	<i>avg_puntos</i> [0.66, 1.0]	0.05	0.75
	<i>cnt_clics</i> [766, 1533]	<i>avg_puntos</i> [0.66, 1.0]	0.23	0.56
	<i>pc_pcontestadas</i> [0.0, 0.11] $\wedge$ <i>sum_ficheros</i> [0, 115] $\wedge$ <i>cnt_ficheros</i> [0, 61]	<i>avg_puntos</i> [-0.11, 0.11]	0.03	0.83
DA	<i>sexo</i>	<i>pc_pcontestadas</i>	0.07	0.26
	<i>pc_pcontestadas</i>	<i>avg_puntos</i>	0.03	0.27
	<i>sum_ficheros</i>	<i>avg_puntos</i>	0.06	0.35
	<i>sexo</i>	<i>cnt_clics</i>	0.13	0.37
	<i>sum_ficheros</i>	<i>pc_pcontestadas</i>	0.04	0.18
	<i>cnt_clics</i>	<i>avg_puntos</i>	0.05	0.38
	<i>cnt_ficheros</i>	<i>avg_puntos</i>	0.07	0.20
RAD	$\langle pc\_pcontestadas, medio \rangle$	$\langle avg\_puntos, medio \rangle$	0.22	0.26
	$\langle cnt\_ficheros, bajo \rangle$	$\langle sum\_ficheros, bajo \rangle$	0.67	0.67
	$\langle pc\_pcontestadas, medio \rangle$	$\langle cnt\_clicks, bajo \rangle$	0.25	0.58
	$\langle Sexo, masculino \rangle$ $\wedge$ $\langle avg\_puntos, alto \rangle$	$\langle pc\_pcontestadas, alto \rangle$	0.24	0.59
	$\langle cnt\_clicks, medio \rangle$	$\langle avg\_puntos, medio \rangle$	0.16	0.67
	$\langle pc\_pcontestadas, alto \rangle$	$\langle avg\_puntos, alto \rangle$	0.14	0.77
	$\langle sum\_ficheros, medio \rangle$ $\wedge$ $\langle cnt\_ficheros, medio \rangle$	$\langle avg\_puntos, medio \rangle$	0.13	0.71

## Sintaxis de la gramática utilizada para las reglas en DRIMS

ANTLR, siglas de *ANother Tool for Language Recognition*, es una herramienta que proporciona un marco de trabajo para la construcción de reconocedores, intérpretes, compiladores y traductores de lenguajes a partir de gramáticas enriquecidas con acciones. DRIMS utiliza esta herramienta para validar sus reglas e implementar los algoritmos de mantenimiento incremental. Este análisis es realizado mediante reglas gramaticales que generan un analizador gramatical descendente-recursivo.

El presente apéndice muestra dichas reglas gramaticales. Mediante esta sintaxis se puede comprobar en su totalidad los tipos de reglas que DRIMS soporta y cómo deben ser expresadas.

```
rule_set : rule_type+ ;
rule_type : ar
           | ad
           | far ;

ar : expr_AR IMPLIES expr_AR ;

ad : expr_AD IMPLIES expr_AD ;

far : expr_FAR IMPLIES expr_FAR ;

expr_AR : value_AR (((','|'&&') | ('||')) value_AR)* ;

value_AR : ID EQUAL (INT
                  | FLOAT
                  | STRING)
          | ID IN (('['INT ', 'INT']')
                 | ('['FLOAT ', 'FLOAT']'))
          | LEFTP expr_AR RIGTHP
          | '¬'LEFTP expr_AR RIGTHP ;

expr_AD : ID (('','|'&&') ID)* ;

expr_FAR: value_FAR (('','|'&&') value_FAR)* ;

value_FAR: ID EQUAL LABEL ;

IMPLIES : '->';

IN : 'in' || 'IN';

EQUAL : '=' || '==';

RANGE_INT: '['INT', 'INT']';
```

---

```
RANGE_FLOAT : '['FLOAT','FLOAT']';

LEFTP : '(';

RIGHTP : ')';

ID : ('a'..'z'|'A'..'Z')
    ('a'..'z'|'A'..'Z'|'0'..'9'|'_')* ;

LABEL : '\\''('a'..'z'|'A'..'Z'|'_')
        ('a'..'z'|'A'..'Z'|'0'..'9'|'_')* '\\'';

INT : '0'..'9'+ ;

FLOAT : ('0'..'9')+ '.'('0'..'9')* EXPONENT?
        | '.'('0'..'9')+ EXPONENT?
        | ('0'..'9')+ EXPONENT ;

STRING : '''( ESC_SEQ | ~('\\'|''' ) )* ''';

EXPONENT : ('e'|'E') ('+'|'-')? ('0'..'9')+ ;

ESC_SEQ : '\\\\('b'|'t'|'n'|'f'|'r'|'\\'|'\\'|'\\')
          | UNICODE_ESC
          | OCTAL_ESC ;
```



# Bibliografía

- [1] Agrawal, R., T. Imieliński y A. Swami: *Mining Association Rules Between Sets of Items in Large Databases*. SIGMOD REC, 22(2):207–216, Jun. 1993, ISSN 0163-5808. [2](#), [14](#)
- [2] Agrawal, R. y R. Srikant: *Fast Algorithms for Mining Association Rules in Large Databases*. En *Proceedings of the 20th International Conference on Very Large Data Bases, VLDB '94*, págs. 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc., ISBN 1-55860-153-8. [2](#), [15](#), [114](#)
- [3] Alcalá-Fdez, J., L. Sánchez, S. García, M. Jesus, S. Ventura, J. Garrell, J. Otero, C. Romero, J. Bacardit, V. Rivas, J. Fernández y F. Herrera: *KEEL: a software tool to assess evolutionary algorithms for data mining problems*. SOFT COMPUT, 13(3):307–318, 2009, ISSN 1432-7643. [83](#)
- [4] Alonso, A. P.: *Reglas de Negocio en Bases de Datos Relacionales*. Tesis de Licenciatura, Universidad Central “Marta Abreu” de Las Villas, 2010. [47](#), [48](#), [104](#)
- [5] Andreescu, A. y M. Mircea: *Managing Knowledge as Business Rules*. Informática Económica, 13(4):63–74, 2009. [47](#), [48](#)



- [6] ANSI: *ISO/IEC FDIS 9075-2 Information technology - Database languages -SQL-Part 2: Foundation (SQL/Foundation)*. Informe técnico., ISO International Standard, 2011. [50](#), [51](#), [52](#)
- [7] Assunção, M. D., R. N. Calheiros, S. Bianchi, M. A. Netto y R. Buyya: *Big Data computing and clouds: Trends and future directions*. Journal of Parallel and Distributed Computing, 79-80:3–15, 2015, ISSN 0743-7315. Special Issue on Scalable Systems for Big Data Management and Analytics. [1](#), [9](#)
- [8] Ataullah, A. A. y F. W. Tompa: *Business Policy Modeling and Enforcement in Databases*. Proceedings of the VLDB Endowment, 4(11), 2011. [47](#)
- [9] Ayan, N. F., A. U. Tansel y E. Arkun: *An Efficient Algorithm to Update Large Itemsets with Early Pruning*. En *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, págs. 287–291, New York, NY, USA, 1999. ACM, ISBN 1-58113-143-7. [3](#), [29](#)
- [10] Azé, J. y Y. Kodratoff: *Evaluation de la résistance au bruit de quelques mesures d'extraction de règles d'association*. En *EGC*, vol. 1, págs. 143–154, 2002. [14](#)
- [11] Backiel, A. y S. vanden Broucke: *Beginning Java Programming: The Object-Oriented Approach*. Wiley, 2015, ISBN 9781118739358. [98](#)
- [12] Berzal, F., I. Blanco, D. Sánchez y M. A. Vila: *Measuring the accuracy and interest of association rules: A new framework*. INTELL DATA ANAL, 6(3):221–235, 2002. [12](#), [13](#)
- [13] Berzal, F., J. C. Cubero, N. Marín y J. M. Serrano: *TBAR: An efficient method for association rule mining in relational databases*. DATA KNOWL ENG, 37(1):47–64, 2001. [2](#), [15](#)
- [14] Blakeley, J. A., P. A. Larson y F. W. Tompa: *Efficiently Updating Materialized Views*. SIGMOD Rec., 15(2):61–71, Jun. 1986, ISSN 0163-5808. [40](#), [46](#)

- [15] Boettcher, M., G. Ruß, D. Nauck y R. Kruse: *From change mining to relevance feedback a unified view on assessing rule interestingness*. Post-Mining of association rules: Techniques for effective knowledge extraction, págs. 12–37, 2009. [4](#), [61](#), [115](#)
- [16] Bouker, S., R. Saidi, S. B. Yahia y E. M. Nguifo: *Ranking and Selecting Association Rules Based on Dominance Relationship*. En *2012 IEEE 24th International Conference on Tools with Artificial Intelligence*, vol. 1, págs. 658–665, Nov 2012. [61](#)
- [17] Boulanger, T.: *XML práctico: Bases esenciales, conceptos y casos prácticos (2ª edición)*. Recursos Informáticos. Ediciones ENI, 2015, ISBN 9782746097360. [99](#)
- [18] Brin, S., R. Motwani y C. Silverstein: *Beyond Market Baskets: Generalizing Association Rules to Correlations*. *SIGMOD Rec.*, 26(2):265–276, Jun. 1997, ISSN 0163-5808. [14](#)
- [19] Brin, S., R. Motwani, J. D. Ullman y S. Tsur: *Dynamic Itemset Counting and Implication Rules for Market Basket Data*. *SIGMOD Rec.*, 26(2):255–264, Jun. 1997, ISSN 0163-5808. [14](#)
- [20] Cañas, A., D. Calandria, E. Ortigosa, E. Ros y A. Díaz: *SWAD: Web System for Education Support*. En Fernández-Manjón, B., J. Sánchez-Pérez, J. Gómez-Pulido, M. Vega-Rodríguez y J. Bravo-Rodríguez (eds.): *Computers and Education*, págs. 133–142. Springer Netherlands, 2007, ISBN 978-1-4020-4913-2. [84](#)
- [21] Cabot, J. y E. Teniente: *Computing the Relevant Instances That May Violate an OCL Constraint*. En Pastor, O. y J. a. Falcão e Cunha (eds.): *Advanced Information Systems Engineering*, vol. 3520 de *Lecture Notes in Computer Science*, págs. 48–62. Springer Berlin Heidelberg, 2005, ISBN 978-3-540-26095-0. [45](#), [46](#), [48](#)
- [22] Cabot, J. y E. Teniente: *Incremental Evaluation of OCL Constraints*. En Dubois, E. y K. Pohl (eds.): *Advanced Information Systems Engineering*,

- vol. 4001 de *Lecture Notes in Computer Science*, págs. 81–95. Springer Berlin Heidelberg, 2006, ISBN 978-3-540-34652-4. [4](#), [45](#), [48](#), [73](#)
- [23] Cabot, J. y E. Teniente: *Incremental integrity checking of UML/OCL conceptual schemas*. *Journal of Systems and Software*, 82(9):1459–1478, 2009, ISSN 0164-1212. SI: {QSIC} 2007. [45](#)
- [24] Calero, J., G. Delgado, M. Sánchez-Marañón, D. Sánchez, M. Vila y J. Serrano: *An Experience in Management of Imprecise Soil Databases by Means of Fuzzy Association Rules and Fuzzy Approximate Dependencies*. En Seruca, I., J. Cordeiro, S. Hammoudi y J. Filipe (eds.): *Enterprise Information Systems VI*, págs. 158–166. Springer Netherlands, 2006, ISBN 978-1-4020-3674-3. [20](#)
- [25] Castillo, M. B. B., A. P. Toledo, A. P. Alonso, L. M. G. González y R. P. Vázquez: *Inserción automática de reglas de negocio en bases de datos*. *Revista Técnica de la Facultad de Ingeniería. Universidad del Zulia*, 36(3):9, 2013. [4](#), [47](#), [48](#), [49](#), [104](#)
- [26] Ceri, S. y J. Widom: *Deriving Production Rules for Constraint Maintenance*. En *Proceedings of the 16th International Conference on Very Large Data Bases, VLDB '90*, págs. 566–577, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc., ISBN 1-55860-149-X. [46](#)
- [27] Ceri, S. y J. Widom: *Deriving Production Rules for Incremental View Maintenance*. En *Proceedings of the 17th International Conference on Very Large Data Bases, VLDB '91*, págs. 577–589, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc., ISBN 1-55860-150-3. [40](#)
- [28] Chamberlin, D. D. y R. F. Boyce: *SEQUEL: A Structured English Query Language*. En *Proceedings of the 1974 ACM SIGFIDET (Now SIGMOD) Workshop on Data Description, Access and Control, SIGFIDET '74*, págs. 249–264, New York, NY, USA, 1974. ACM. [50](#)
- [29] Cheung, D., J. Han, V. Ng y C. Y. Wong: *Maintenance of discovered association rules in large databases: an incremental updating technique*.

- En *Data Engineering, 1996. Proceedings of the Twelfth International Conference on*, págs. 106–114, Feb 1996. [3](#), [26](#), [29](#), [114](#), [115](#)
- [30] Cheung, D. W. L., S. D. Lee y B. Kao: *A General Incremental Technique for Maintaining Discovered Association Rules*. En *Proceedings of the Fifth International Conference on Database Systems for Advanced Applications (DASFAA)*, págs. 185–194. World Scientific Press, 1997. [3](#), [29](#), [82](#)
- [31] Cheung, W. y O. R. Zaiane: *Incremental mining of frequent patterns without candidate generation or support constraint*. En *Database Engineering and Applications Symposium, 2003. Proceedings. Seventh International*, págs. 111–116, July 2003. [30](#)
- [32] Church, K. W. y P. Hanks: *Word Association Norms, Mutual Information, and Lexicography*. *Comput. Linguist.*, 16(1):22–29, Mar. 1990, ISSN 0891-2017. [14](#), [63](#)
- [33] Codd, E. F.: *A Relational Model of Data for Large Shared Data Banks*. *Commun. ACM*, 13(6):377–387, Jun. 1970, ISSN 0001-0782. [1](#), [10](#), [33](#)
- [34] Colby, L. S., T. Griffin, L. Libkin, I. S. Mumick y H. Trickey: *Algorithms for Deferred View Maintenance*. *SIGMOD REC*, 25(2):469–480, Jun. 1996, ISSN 0163-5808. [39](#)
- [35] Connolly, T. y C. Begg: *Sistemas de bases de datos: un enfoque práctico para diseño, implementación y gestión*, cap. OLAP, págs. 1089–1113. Pearson Educación, 4ª ed., 2005, ISBN 84-7829-075-3. [38](#)
- [36] Cubero, J., F. Cuenca, I. Blanco y M. Vila: *Incomplete functional dependencies versus knowledge discovery in databases*. *Proceedings of the EUFIT*, 98:731–741, 1998. [24](#)
- [37] Datta, S. y S. Bose: *Mining and Ranking Association Rules in Support, Confidence, Correlation, and Dissociation Framework*, págs. 141–152. Springer India, New Delhi, 2016, ISBN 978-81-322-2695-6. [61](#)

- [38] Delgado, M., N. Marín, M. Martín-Bautista, D. Sánchez y M. A. Vila: *Mining Fuzzy Association Rules: An Overview*. En Nikravesh, M., L. Zadeh y J. Kacprzyk (eds.): *Soft Computing for Information Processing and Analysis*, vol. 164 de *Studies in Fuzziness and Soft Computing*, págs. 351–373. Springer Berlin Heidelberg, 2005, ISBN 978-3-540-22930-8. [19](#)
- [39] Delgado, M., N. Marín, D. Sánchez y M. A. Vila: *Fuzzy association rules: general model and applications*. *Fuzzy Systems, IEEE Transactions on*, 11(2):214–225, Apr 2003, ISSN 1063-6706. [2](#), [19](#), [20](#), [22](#), [23](#)
- [40] Delgado, M., M.D. Ruiz, D. Sánchez y M. A. Vila: *Fuzzy quantification: a state of the art*. *Fuzzy Sets and Systems*, 242(0):1–30, 2014, ISSN 0165-0114. Theme: Quantifiers and Logic. [20](#)
- [41] Delgado, M., D. Sánchez y M. A. Vila: *Acquisition of Fuzzy Association Rules from Medical Data*. En Barro, S. y R. Marín (eds.): *Fuzzy Logic in Medicine*, vol. 83 de *Studies in Fuzziness and Soft Computing*, págs. 286–310. Physica-Verlag HD, 2002, ISBN 978-3-7908-2498-8. [19](#)
- [42] Delic, K. A., L. Douillet y U. Dayal: *Towards an architecture for real-time decision support systems: challenges and solutions*. En *Database Engineering and Applications, 2001 International Symposium on.*, págs. 303–311, 2001. [3](#), [61](#), [91](#), [115](#)
- [43] Demuth, B., T. U. Dresden, C. Wilke y T. U. Dresden: *Model and Object Verification by Using Dresden OCL*. En *Technical University*, pág. 81, 2009. [45](#)
- [44] Dittrich, K. R., S. Gatzju y A. Geppert: *The active database management system manifesto: A rulebase of ADBMS features*, págs. 1–17. Springer Berlin Heidelberg, Berlin, Heidelberg, 1995, ISBN 978-3-540-45137-2. [31](#), [33](#)
- [45] DuBois, P.: *MySQL Cookbook: Solutions for Database Developers and Administrators*. O'Reilly Media, 2014, ISBN 9781449374150. [83](#)

- [46] Dudek, D.: *RMAIN: Association rules maintenance without reruns through data*. INFORM SCIENCES, 179(24):4123–4139, 2009, ISSN 0020-0255. [3](#), [82](#)
- [47] Elmasri, R. y S. Navathe: *Fundamentals of Database Systems*. N<sup>o</sup> v. 2 en *Fundamentals of Database Systems*. Pearson/Addison Wesley, 5<sup>a</sup> ed., 2007. [23](#), [39](#), [42](#), [45](#)
- [48] Frawley, W. J., G. Piatetsky-Shapiro y C. J. Matheus: *Knowledge discovery in databases: An overview*. AI magazine, 13(3):57, 1992. [1](#)
- [49] Geng, L. y H. J. Hamilton: *Interestingness Measures for Data Mining: A Survey*. ACM Comput. Surv., 38(3), Sep. 2006, ISSN 0360-0300. [12](#), [61](#)
- [50] Glass, D. H.: *Confirmation measures of association rule interestingness*. Knowledge-Based Systems, 44:65–77, 2013, ISSN 0950-7051. [12](#)
- [51] Goldin, D., S. Srinivasa y V. Srikanti: *Active databases as information systems*. En *Database Engineering and Applications Symposium, 2004. IDEAS '04. Proceedings. International*, págs. 123–130, July 2004. [31](#)
- [52] Good, I. J., I. Hacking, R. C. Jeffrey y H. Törnebohm: *The Estimation of Probabilities: An Essay on Modern Bayesian Methods*. Synthese, 16(2):234–244, 1966. [14](#)
- [53] Grefen, P. W. y P. M. Apers: *Integrity control in relational database systems: an overview*. Data & Knowledge Engineering, 10(2):187–223, 1993, ISSN 0169-023X. [45](#)
- [54] Gupta, A., D. Katiyar y I. S. Mumick: *Counting solutions to the View Maintenance Problem*. En *Workshop on Deductive Databases, JICSLP*, págs. 185–194. Citeseer, 1992. counting problem. [42](#)
- [55] Gupta, A., I. S. Mumick y cols.: *Maintenance of materialized views: Problems, techniques, and applications*. IEEE Data Eng. Bull., 18(2):3–18, 1995. [38](#), [40](#), [42](#), [44](#)

- [56] Gupta, H. y I.S. Mumick: *Incremental maintenance of aggregate and outerjoin expressions*. Information Systems, 31(6):435–464, 2006, ISSN 0306-4379. [42](#)
- [57] Halle, B. von: *Business Rules Applied: Building Better Systems Using the Business Rules Approach*. Wiley Publishing, 1st ed., 2001, ISBN 0471412937, 9780471412939. [47](#)
- [58] Han, J., J. Pei y Y. Yin: *Mining Frequent Patterns Without Candidate Generation*. SIGMOD REC, 29(2):1–12, May. 2000, ISSN 0163-5808. [15](#), [17](#), [18](#), [114](#)
- [59] Harihar, K., D. Satchidananda y G. Ashish: *A Survey on Fuzzy Association Rule Mining*. International Journal of Data Warehousing and Mining (IJDWM), 9(1):1–27, 2013, ISSN 1548-3924. [20](#)
- [60] Hong, T.P. y J.B. Chen: *Finding relevant attributes and membership functions*. Fuzzy Sets and Systems, 103(3):389–404, 1999, ISSN 0165-0114. [23](#)
- [61] Hong, T.P., C.S. Kuo y S.C. Chi: *TRADE-OFF BETWEEN COMPUTATION TIME AND NUMBER OF RULES FOR FUZZY MINING FROM QUANTITATIVE DATA*. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 09(05):587–604, 2001. [23](#)
- [62] Hong, T.P., C.W. Lin y T.C. Lin: *THE MFFP-TREE FUZZY MINING ALGORITHM TO DISCOVER COMPLETE LINGUISTIC FREQUENT ITEMSETS*. Computational Intelligence, 30(1):145–166, 2014, ISSN 1467-8640. [30](#), [93](#)
- [63] Hong, T.P., C.W. Lin y Y.L. Wu: *Incrementally fast updated frequent pattern trees*. EXPERT SYST APPL, 34(4):2424–2435, 2008, ISSN 0957-4174. [29](#), [82](#), [92](#), [114](#)

- [64] Hong, T. P., T. C. Lin y T. C. Lin: *Mining complete fuzzy frequent itemsets by tree structures*. En *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, págs. 563–567, Oct 2010. [30](#), [114](#)
- [65] Hong, T. P., C. Y. Wang y Y. H. Tao: *A new incremental data mining algorithm using pre-large itemsets*. *Intelligent Data Analysis*, 5(2):111–129, 2001. [29](#)
- [66] Huhtala, Y., J. Kärkkäinen, P. Porkka y H. Toivonen: *Tane: An Efficient Algorithm for Discovering Functional and Approximate Dependencies*. *The Computer Journal*, 42(2):100–111, 1999. [24](#)
- [67] Jeffreys, H.: *Some tests of significance, treated by the theory of probability*. En *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 31, págs. 203–222. Cambridge Univ Press, 1935. [14](#)
- [68] Kalanat, N. y M. R. Kangavari: *Data Mining Methods for Rule Designing and Rule Triggering in Active Database Systems*. *International Journal of Database Theory and Application*, 8(1):39–44, 2015, ISSN 2005-4270. [31](#)
- [69] King, R. S. y J. J. Legendre: *Discovery of Functional and Approximate Functional Dependencies in Relational Databases*. *Journal of Applied Mathematics and Decision Sciences*, 7(1):49–59, 2003. [24](#)
- [70] Kliegr, T., J. Kuchař, D. Sottara y S. Vojtř: *Learning Business Rules with Association Rule Classifiers*, págs. 236–250. Springer International Publishing, Cham, 2014, ISBN 978-3-319-09870-8. [49](#)
- [71] Kryszkiewicz, M.: *A Lossless Representation for Association Rules Satisfying Multiple Evaluation Criteria*, págs. 147–158. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016, ISBN 978-3-662-49390-8. [2](#), [12](#), [62](#), [64](#)
- [72] Kumar, N. P., L. J. Rao y G. V. Kumar: *A study on positive and negative association rule mining*. En *International Journal of Engineering Research and Technology*, vol. 1. ESRSA Publications, 2012. [109](#)



- [73] Kuno, H. y G. Graefe: *Deferred Maintenance of Indexes and of Materialized Views*, págs. 312–323. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, ISBN 978-3-642-25731-5. [80](#)
- [74] Lee, Y.S. y S.J. Yen: *Incrementally Mining Frequent Patterns from Large Database*. En Pedrycz, W. y S.M. Chen (eds.): *Information Granularity, Big Data, and Computational Intelligence*, vol. 8 de *Studies in Big Data*, págs. 121–140. Springer International Publishing, 2015, ISBN 978-3-319-08253-0. [82](#), [92](#)
- [75] Lenca, P., P. Meyer, B. Vaillant y S. Lallich: *On selecting interestingness measures for association rules: User oriented description and multiple criteria decision aid*. EUR J OPER RES, 184(2):610–626, 2008, ISSN 0377-2217. [2](#), [12](#), [62](#)
- [76] Leung, C. K. S., Q. I. Khan y T. Hoque: *CanTree: a tree structure for efficient incremental mining of frequent patterns*. En *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pág. 8 pp., 2005, ISBN 1550-4786. [30](#)
- [77] Li, X., Z.H. Deng y S. Tang: *A Fast Algorithm for Maintenance of Association Rules in Incremental Databases*. En Li, X., O. Zaïane y Z.h. Li (eds.): *Advanced Data Mining and Applications*, vol. 4093 de *Lecture Notes in Computer Science*, págs. 56–63. Springer Berlin Heidelberg, 2006, ISBN 978-3-540-37025-3. [3](#), [27](#), [29](#), [82](#), [92](#)
- [78] Lichman, M.: *UCI Machine Learning Repository*, 2013. [86](#)
- [79] Lin, C.W. y T.P. Hong: *Maintenance of prelarge trees for data mining with modified records*. INFORM SCIENCES, 278(0):88–103, 2014, ISSN 0020-0255. [82](#), [92](#)
- [80] Lin, C.W., T.P. Hong y W.H. Lu: *Fuzzy data mining based on the compressed fuzzy FP-trees*. En *Fuzzy Systems, 2009. FUZZ-IEEE 2009. IEEE International Conference on*, págs. 1068–1072, Aug 2009. [30](#)

- [81] Lin, C. W., T. P. Hong y W. H. Lu: *The Pre-FUFP algorithm for incremental mining*. Expert Systems with Applications, 36(5):9498–9505, 2009, ISSN 0957-4174. [29](#), [82](#)
- [82] Lin, C. W., T. P. Hong y W. H. Lu: *Linguistic data mining with fuzzy FP-trees*. Expert Systems with Applications, 37(6):4560–4567, 2010, ISSN 0957-4174. [30](#)
- [83] Liu, B., W. Hsu y Y. Ma: *Mining Association Rules with Multiple Minimum Supports*. En *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, págs. 337–341, New York, NY, USA, 1999. ACM, ISBN 1-58113-143-7. [16](#)
- [84] Liu, C. Y., C. Y. Tseng y M. S. Chen: *Incremental Mining of Significant URLs in Real-Time and Large-Scale Social Streams*. En Pei, J., V. Tseng, L. Cao, H. Motoda y G. Xu (eds.): *Advances in Knowledge Discovery and Data Mining*, vol. 7819 de *Lecture Notes in Computer Science*, págs. 473–484. Springer Berlin Heidelberg, 2013, ISBN 978-3-642-37455-5. [27](#), [61](#)
- [85] Liu, H., Y. Lin y J. Han: *Methods for mining frequent items in data streams: an overview*. KNOWL INF SYST, 26(1):1–30, 2011, ISSN 0219-1377. [3](#), [27](#), [61](#), [91](#), [115](#)
- [86] Liu, X., K. Zhai y W. Pedrycz: *An improved association rules mining method*. Expert Systems with Applications, 39(1):1362–1374, 2012, ISSN 0957-4174. [2](#), [15](#)
- [87] Łlukasiewicz, J.: *Die logischen Grundlagen der Wahrscheinlichkeitsrechnung*. Krak ow (1913). I. Borkowski (ed.): Jan Łlukasiewicz-Selected Works, North Holland Publishing Company, Amsterdam, London, Polish Scientific Publishers, Warszawa, 1970. [24](#)
- [88] Martín, D., A. Rosete, J. Alcalá-Fdez y F. Herrera: *A New Multiobjective Evolutionary Algorithm for Mining a Reduced Set of Interesting Positive and Negative Quantitative Association Rules*. IEEE Transactions on

- Evolutionary Computation, 18(1):54–69, Feb 2014, ISSN 1089-778X. [2](#), [15](#)
- [89] Medina, I. J. B.: *Deducción en Bases de Datos Relacionales Difusas*. Tesis de Doctorado, Universidad de Granada, 2001. [22](#), [43](#), [118](#)
- [90] Mohapatra, A. y M. Genesereth: *Incremental Maintenance of Aggregate Views*. En Beierle, C. y C. Meghini (eds.): *Foundations of Information and Knowledge Systems*, vol. 8367 de *Lecture Notes in Computer Science*, págs. 399–414. Springer International Publishing, 2014, ISBN 978-3-319-04938-0. [4](#), [40](#), [42](#)
- [91] Morgan, T.: *Business Rules and Information Systems: Aligning IT with Business Goals*. Pearson Education, 2002, ISBN 9780672334177. [47](#), [48](#)
- [92] Morgenstern, M.: *Active Databases As a Paradigm for Enhanced Computing Environments*. En *Proceedings of the 9th International Conference on Very Large Data Bases, VLDB '83*, págs. 34–42, San Francisco, CA, USA, 1983. Morgan Kaufmann Publishers Inc., ISBN 0-934613-15-X. [4](#)
- [93] Nath, B., D. K. Bhattacharyya y A. Ghosh: *Incremental association rule mining: a survey*. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 3(3):157–169, 2013, ISSN 1942-4795. [3](#), [17](#)
- [94] Olivé, A.: *Conceptual Modeling of Information Systems*, cap. Integrity Constraints, págs. 181–208. Springer Berlin Heidelberg, 2007, ISBN 978-3-540-39389-4. [43](#)
- [95] OMG: *Object constraint language specification version 2.4*. Informe técnico., Object Management Group, 2014. [45](#), [48](#)
- [96] OMG: *Unified Modeling Language version 2.5*. Informe técnico., Object Management Group, 2015. [45](#), [98](#)
- [97] Oriol, X. y E. Teniente: *Incremental Checking of OCL Constraints through SQL Queries*. En *OCL@ MoDELS*, págs. 23–32. Citeseer, 2014. [48](#)

- [98] Oriol, X. y E. Teniente: *Incremental Checking of OCL Constraints with Aggregates Through SQL*, págs. 199–213. Springer International Publishing, Cham, 2015, ISBN 978-3-319-25264-3. [45](#), [48](#)
- [99] Papadimitriou, S. y S. Mavroudi: *The Fuzzy Frequent Pattern Tree*. En *Proceedings of the 9th WSEAS International Conference on Computers, ICCOMP'05*, págs. 31–37, Stevens Point, Wisconsin, USA, 2005. World Scientific and Engineering Academy and Society (WSEAS), ISBN 960-8457-29-7. [30](#)
- [100] Park, J. S., M. S. Chen y P. S. Yu: *An Effective Hash-based Algorithm for Mining Association Rules*. SIGMOD Rec., 24(2):175–186, May. 1995, ISSN 0163-5808. [16](#)
- [101] Parr, T.: *The Definitive ANTLR 4 Reference*. Pragmatic Bookshelf, 2nd ed., 2013, ISBN 1934356999, 9781934356999. [99](#)
- [102] Paton, N.: *Active Rules in Database Systems*. Monographs in Computer Science. Springer New York, 2012, ISBN 9781441986566. [4](#)
- [103] Paton, N. W. y O. Díaz: *Active Database Systems*. ACM COMPUT SURV, 31(1):63–103, Mar. 1999, ISSN 0360-0300. [4](#), [31](#)
- [104] Pérez-Alonso, A., I. Blanco, L. M. González-González y J. M. Serrano: *Incremental Maintenance of Discovered Association Rules and Approximate Dependencies*. Intelligent Data Analysis An International Journal, 21(1), 2016. [3](#), [4](#), [62](#), [68](#), [73](#)
- [105] Piatetsky-Shapiro, G.: *Probabilistic data dependencies*. En *Machine Discovery Workshop Aberdeen, Scotland*, 1992. [24](#)
- [106] Quass, D.: *Maintenance Expressions for Views with Aggregation*. En *Views'96*, 1996. [40](#), [66](#)
- [107] Quass, D., A. Gupta, I. S. Mumick y J. Widom: *Making views self-maintainable for data warehousing*. En *Parallel and Distributed*

- Information Systems, 1996., Fourth International Conference on*, págs. 158–169, Dec 1996. [46](#)
- [108] Riggs, S., G. Ciolli, H. Krosing y G. Bartolini: *PostgreSQL 9 Administration Cookbook - Second Edition*. Packt Publishing, 2015, ISBN 9781849519076. [83](#)
- [109] Ross, R.: *Principles of the Business Rule Approach*. Addison-Wesley, 2003, ISBN 9780201788938. [47](#)
- [110] Ross, R. G.: *What Is a Business Rule?* *Business Rules Journal*, 11(3), Marzo 2010. [47](#), [48](#)
- [111] Sachin, R. y M. Vijay: *A Survey and Future Vision of Data Mining in Educational Field*. En *Advanced Computing Communication Technologies (ACCT), 2012 Second International Conference on*, págs. 96–100, Jan 2012. [84](#)
- [112] Sánchez, D., J. M. Serrano, I. Blanco, M. J. Martín-Bautista y M. A. Vila: *Using association rules to mine for strong approximate dependencies*. *DATA MIN KNOWL DISC*, 16(3):313–348, 2008. [2](#), [12](#), [24](#), [25](#), [26](#), [66](#)
- [113] Sauter, V.: *Decision Support Systems for Business Intelligence*. Wiley, 2014, ISBN 9781118627235. [3](#), [61](#), [91](#), [115](#)
- [114] Savasere, A., E. Omiecinski y S. B. Navathe: *An Efficient Algorithm for Mining Association Rules in Large Databases*. En *Proceedings of the 21th International Conference on Very Large Data Bases*, págs. 432–444. Morgan Kaufmann Publishers Inc., 1995. [16](#)
- [115] Serrano Chica, J. M.: *Fusión de conocimiento en bases de datos relacionales : Medidas de agregación y resumen*. Tesis de Doctorado, Universidad de Granada, 2003. [12](#), [23](#), [26](#)
- [116] Shah, S., N. Chauhan y S. Bhanderi: *Incremental Mining of Association Rules: A Survey*. *International Journal of Computer Science and Information Technologies*, 3(3):4071–4074, 2012. [3](#)

- [117] Shortliffe, E. H. y B. G. Buchanan: *A model of inexact reasoning in medicine*. *Mathematical Biosciences*, 23(3):351–379, 1975, ISSN 0025-5564. [13](#)
- [118] Simovici, A. D., D. Cristofor y L. Cristofor: *Impurity measures in databases*. *Acta Informática*, 38(5):307–324, 2002, ISSN 1432-0525. [24](#)
- [119] Srikant, R. y R. Agrawal: *Mining quantitative association rules in large relational tables*. En *ACM SIGMOD Record*, vol. 25, págs. 1–12. ACM, 1996. [19](#), [39](#)
- [120] Strack, B., J. P. DeShazo, C. Gennings, J. L. Olmo, S. Ventura, K. J. Cios y J. N. Clore: *Impact of HbA1c measurement on hospital readmission rates: analysis of 70,000 clinical database patient records*. *BioMed research international*, 2014(781670):11, 2014. [86](#)
- [121] Suksut, K., P. Kongchai, S. Phoemhansa, R. Sutamma, K. Kerdprasop y N. Kerdprasop: *Single Versus Multiple Measures for Fuzzy Association Rule Mining*. En *3rd International Conference on Industrial Application Engineering 2015*, 2015. [12](#), [62](#)
- [122] Tan, P.N., V. Kumar y J. Srivastava: *Selecting the Right Interestingness Measure for Association Patterns*. En *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, págs. 32–41, New York, NY, USA, 2002. ACM, ISBN 1-58113-567-X. [12](#), [61](#)
- [123] Urpí, T. y A. Olivé: *Semantic change computation optimization in active databases*. En *Research Issues in Data Engineering, 1994. Active Database Systems. Proceedings Fourth International Workshop on*, págs. 19–27. IEEE, 1994. [35](#)
- [124] Veloso, A., B. Pôssas, W. Meira, J. Márcio y B. Carvalho: *Knowledge management in association rule mining*. En *In Integrating Data Mining and Knowledge Management, held in conjunction with the 2001 IEE International Conference on Data Mining (ICDM, 2001)*. [3](#), [29](#)

- [125] Vojir, S., T. Kliegr, A. Hazucha, R. Škrabal y M. Šimunek: *Transforming association rules to business rules: EasyMiner meets Drools*. RuleML-2013 Challenge. CEUR-WS. org, 2013. [49](#)
- [126] Wang, C. H., W. H. Lee y C. T. Pang: *Applying Fuzzy FP-Growth to Mine Fuzzy Association Rules*. International Journal of Computer, Electrical, Automation, Control and Information Engineering, 4(5):986–992, 2010, ISSN PISSN:2010-376X, EISSN:2010-3778. [30](#)
- [127] Wu, X., X. Zhu, G. Q. Wu y W. Ding: *Data mining with big data*. IEEE T KNOWL DATA EN, 26(1):97–107, Jan 2014, ISSN 1041-4347. [1](#), [3](#), [4](#), [9](#), [61](#)
- [128] Yang, J., K. Karlapalem y Q. Li: *Algorithms for Materialized View Design in Data Warehousing Environment*. En *Proceedings of the 23rd International Conference on Very Large Data Bases, VLDB '97*, págs. 136–145, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc., ISBN 1-55860-470-7. [46](#)
- [129] Zadeh, L. A.: *Fuzzy sets*. Information and control, 8(3):338–353, 1965. [2](#), [19](#)
- [130] Zaki, M. J., S. Parthasarathy, M. Ogihara y W. Li: *New Algorithms for Fast Discovery of Association Rules*. En *In 3rd Intl. Conf. on Knowledge Discovery and Data Mining*, vol. 97, págs. 283–286, Rochester, NY, USA, 1997. [15](#), [16](#)
- [131] Zemke, F.: *What.S New in SQL:2011*. SIGMOD Rec., 41(1):67–73, Abr. 2012, ISSN 0163-5808. [40](#)
- [132] Zhang, T.: *Knowledge Discovery and Data Mining. Current Issues and New Applications: 4th Pacific-Asia Conference, PAKDD 2000 Kyoto, Japan, April 18–20, 2000 Proceedings*, cap. Association Rules, págs. 245–256. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000, ISBN 978-3-540-45571-4. [14](#)

- 
- [133] Zhou, Z. y C. Ezeife: *A Low-Scan Incremental Association Rule Maintenance Method Based on the Apriori Property*, págs. 26–35. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001, ISBN 978-3-540-45153-2. [3](#), [29](#)
- [134] Zia, Z.K., S.K. Tipu y M.I. Khan: *Research on association rule mining*. *Advances in Computational Mathematics and its Applications*, 2(1):226–236, 2012. [2](#), [15](#)
- [135] Zoet, M., J. Versendaal, P. Ravesteyn y R.J. Welke: *Alignment of business process management and business rules*. En *ECIS*, 2011. [47](#)