LORCDB: Gestor de Bases de Datos Objeto-Relacionales de Restricciones

Departamento de Lenguajes y Sistemas Informáticos

Memoria de Tesis Doctoral para optar al grado de Doctora en Informática por la Universidad de Sevilla

presentado por

Dña María Teresa Gómez López

Directores:

Dr. D. Rafael Martínez Gasca

Dr. D. Carmelo Del Valle Sevillano

Sevilla, Diciembre de 2007

María Teresa Gómez López
Primera Edición, Septiembre 2007
Grupo de investigación Quivir
Departamento de Lenguajes y Sistemas Informáticos
ETSI Informática
Universidad de Sevilla
Avd. de la Reina Mercedes, s/n
Sevilla, 41012. España

http://www.lsi.us.es/∼mayte
mayte@lsi.us.es

Financiación: Este trabajo ha sido financiado por el Ministerio de Ciencia y Tecnología del Gobierno de España (CICYT $DPI2000-0666-C02-02$),
(CICYT $DPI2003-07146-C02-01$), (CICYT $DPI2006-15476-C02-00$).

Don Rafael Martínez Gasca y Don Carmelo Del Valle Sevillano, Profesores Titulares de Universidad del Área de Lenguajes y Sistemas Informáticos de la Universidad de Sevilla,

HACEN CONSTAR

que Doña María Teresa Gómez López, Ingeniera en Informática por la Universidad de Sevilla, ha realizado bajo su supervisión el trabajo de investigación titulado

LORCDB: Gestor de Bases de Datos Objeto-Relacionales de Restricciones

Una vez revisado, autorizo el comienzo de los trámites para su presentación como Tesis Doctoral al tribunal que ha de juzgarlo.

Fdo. Rafael Martínez Gasca Carmelo del Valle Sevillano

Área de Lenguajes y Sistemas Informáticos

Universidad de Sevilla

Sevilla, 19 de Octubre de 2007

Yo, María Teresa Gómez López, con DNI número 28.631.064-C,

DECLARO BAJO JURAMENTO

Ser la autora del trabajo que se presenta en la memoria de esta tesis doctoral que tiene por título:

LORCDB: Gestor de Bases de Datos Objeto-Relacionales de Restricciones

Lo cual firmo en Sevilla, 19 de Octubre de 2007.

Fdo. María Teresa Gómez López

A mis padres que me enseñaron a luchar todos los días para conseguir lo que quería, y a Jesús que con su cariño diario me da fuerzas para poderlo hacer

Índice general

Índice de tablas

Índice de figuras

Agradecimientos

Sólo quien ha hecho una tesis sabe lo gratificante que es escribir los agradecimientos, intentar devolver un poco de lo mucho que se ha recibido. No me gustaría que el agradecimiento a mis directores, los Doctores Rafael Martínez Gasca y Carmelo del Valle, quedara como una mención normal. Ellos han sido excepcionales, su ayuda, comprensión y disposición absoluta, han hecho que todos estos años de investigación hayan sido un placer. Sin ellos nada de esto habría sido posible. Importante también ha sido la participación de el Dr. Miguel Toro, por sus grandes ideas y seguimiento durante todo este tiempo.

Durante estos años, el departamento se ha convertido también en mi casa, y mis compañeros me han ofrecido siempre ayuda y cariño, especialmente Fernando de la Rosa, Pablo Fernández, Roberto, Sergio, Joaquín, David, Diana, Irene, Víctor Díaz, Carlos G. Vallejo y Juan Antonio Álvarez. Pero han sido esenciales los que se han convertido en amigos del alma, Rafael Ceballos, Toñi Reina, Octavio Martín, Antonio Tallón y Beatriz Bernárdez, que han aguantado todos estos años con incondicional amistad y comprensión.

Todos mis amigos, los de toda la vida, han participado en esta tesis dándome infinitos buenos momentos, recordándome que aunque la investigación es todo, no todo es investigar. Muchos de ellos han seguido cada publicación y capítulo de esta tesis, como si fuera suya. Tantas llamadas de preocupación le tengo que agradecer a Mónica, Eva, Sara, Migue y Josefina. Tantas charlas y buenos momentos con Guillermina, Sandra y Nélida. Algunos, como Antonio, Loren y Jorge, han soportado mis nervios desde la carrera, compartido mis primeras penas y alegrías en la Escuela de Informática.

Pero sin duda, lo más importante en todo esto ha sido mi familia, mis sobrinos, que siempre sacan de mí una sonrisa y me han dado fuerzas para continuar. Y mis hermanas, que siempre han estado ahí, y sé que siempre lo estarán. Las que me han enseñado con el ejemplo que querer es poder.

No puede quedar fuera de este agradecimiento Carmen, no sólo por dejar que le robe a Rafael innumerables veces, sino por su disposición incondicional a prestarnos su ayuda y cariño.

Resumen

Esta tesis doctoral aborda el problema de almacenar y tratar en una base de datos información compleja, como los datos económicos, geográficos o de ingeniería. Para ello se realiza un análisis de los diferentes modelos lógicos, tipos de gestores de bases de datos y aplicaciones en las que intervienen dichos datos complejos. Debido a que los datos pueden ser de naturalezas diferentes, en la actualidad es necesario diseñar soluciones a medida para cada tipo de problema cuando sus datos son heterogéneos. De entre todas las propuestas existentes, las Bases de Datos de Restricciones son el centro de esta tesis, ya que dicha información se puede representar mediante el uso de restricciones, lo que significa un conjunto de variables definidas sobre un dominio relacionadas entre sí. Si se extendiera el modelo de datos para soportar datos complejos utilizando las Bases de Datos de Restricciones, existiría una forma única de tratar toda aquella información que pudiera ser representada mediante restricciones. Por lo tanto, la propuesta en la que se centra esta tesis es: Definir un Gestor de Bases de Datos que dé una solución genérica para las aplicaciones que utilizan datos complejos representados mediante restricciones.

En esta memoria de tesis se realiza un análisis de las propuestas existentes para las Bases de Datos de Restricciones, detectando aquellos aspectos que pueden ser mejorados. Estas mejoras implicarán la redefinición del modelo de datos de las Bases de Datos de Restricciones, proponiendo tres tipos de atributos: atributo clásico, atributo restricción y atributo variable de restricción. Derivado de este nuevo modelo y estos nuevos atributos, será necesario redefinir las cinco operaciones primitivas del álgebra relacional para el uso de restricciones, ampliando el lenguaje de consulta SQL. Esto significará modificar y ampliar dichas operaciones, y los operadores que pueden verse involucrados para los diferentes tipos de atributos.

Una vez ampliada la semántica y la sintaxis de SQL, se propone una arquitectura que permite que la evaluación de las diferentes consultas sea eficiente, diseñando diversas

estrategias en función de los datos y el tipo de consulta. Junto a esto, se ha definido un conjunto de algoritmos para cada una de las operaciones mejorando la eficiente computacional de la evaluación de consultas.

Finalmente, se realiza un análisis experimental mediante casos de prueba para comprobar la efectividad y eficiencia de un conjunto de consultas. La cobertura de los casos de prueba se ha hecho considerando los casos más desfavorables, mostrando los tiempos medios de evaluación para cada una de las operaciones del álgebra relacional extendida para restricciones.

Capítulo 1

Introducción

Desde la definición de las bases de datos relacionales, éstas han avanzado para dar soluciones cada vez más eficientes y flexibles al tratamiento de datos, dando soporte a estructuras cada vez más complejas. Sin embargo, la investigación en el área de las bases de datos no ha crecido tan rápido como la complejidad de los datos que tienen que almacenar. Ante dicha problemática, aplicaciones que trabajan con datos 'no tradicionales', como médicos, multimedia, científicos, de ingeniería o geográficos, tienen que diseñar estructuras a medida de sus necesidades. En general, como datos 'no tradicionales' se conocen a aquellos que no toman valores únicos, sino que dependen de parámetros, variables o vienen definidos por relaciones entre variables mediante ecuaciones o inecuaciones matemáticas. Esta creciente complejidad de las aplicaciones y los datos que tratan, orientó a la comunidad de bases de datos a marcar en su investigación nuevos retos, lo que en los años 90 dio origen a las llamadas *Constraint Databases (CDB)*. Las *Constraint Databases* extendieron el modelo de las bases de datos relacionales clásicas para dar cabida a datos de tipo *Restricción*, donde una restricción es un conjunto de variables definidas sobre un dominio y relacionadas entre sí. Dicha relación entre variables se puede describir de una manera compacta mediante una combinación con operadores lógicos de ecuaciones e inecuaciones. La traducción al español del término *Constraint Database* no es fácil, ya que debe dar a entender que es una base de datos donde es posible almacenar y tratar restricciones numéricas como un tipo tradicional, al igual que los tipos numéricos, cadenas o fechas. El término en español que se ha elegido en esta tesis para designar las *Constraint Databases* es **Bases de Datos de Restricciones (BDdR)**, queriendo destacar la capacidad de almacenar datos de tipo *Restricción*, y marcando la diferencia con las

restricciones de integridad o acceso ya existentes en las bases de datos. La razón por la cual los datos 'no tradicionales' se representan como restricciones, es porque permite una forma genérica de representar cualquier dato que pueda ser descrito mediante una relación entre variables. Esta idea permitirá definir una manera única de representar todos los datos que puedan ser descritos mediante restricciones, independientemente del número de variables, su semántica o de la aplicación de los utilice. Esto significa que aplicaciones de características diferentes, como económicas, estadísticas, geográficas, médicas o de ingeniería, utilizarán la misma funcionalidad para almacenar y tratar sus datos, facilitando el desarrollo e implementación de dichas aplicaciones.

El origen en el desarrollo de las Bases de Datos de Restricciones, fue la representación de datos que pueden tomar un número de valores muy grande o incluso infinito de manera finita, ya que permiten mayor expresividad y compresión de la información. Un ejemplo sencillo para entender esta idea, es la representación mediante restricciones del área rectangular mostrada en la figura 1.1.a.

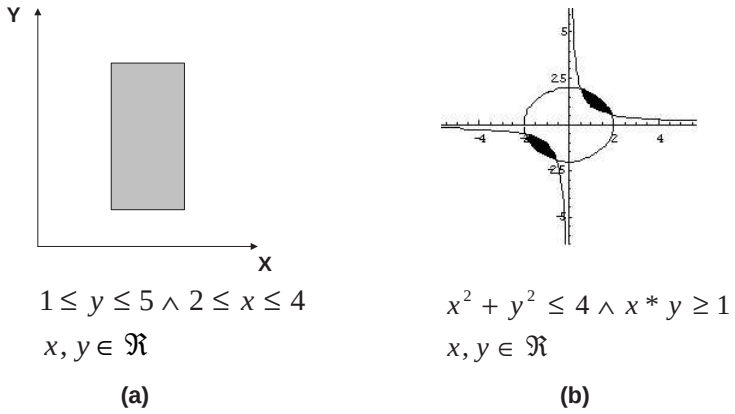

Figura 1.1: Ejemplos de restricciones

Si se utiliza una base de datos relacional para almacenar estas restricciones, una alternativa sería almacenar la restricción mediante los puntos que la acotan: $(2,1),(2,5),(4,1),(4,5)$. De esta forma, se obtiene una manera válida de representar puntos, segmentos, líneas, rectángulos o caminos. Sin embargo, esta solución no es la más conveniente principalmente por dos razones: no todos los tipos de restricciones pueden ser representadas de esta forma, como el ejemplo de la figura 1.1.b; y por la evidente pérdida de expresividad, ya que no se almacena la información en sí, sino un conjunto de valores que la representan. ¿Cómo se almacenaría la información de la figura 1.1.b, teniendo en cuenta que los valores de x e y son infinitos, en una base de datos relacional clásica? Ni que

decir tiene, que la discretización de los datos produciría pérdida de información, lo que implica pérdida de precisión, junto a la creación de bases de datos mucho más grandes. El crecimiento de las bases de datos también influirá en las operaciones de consulta y modificación, ya que tendrían que ser analizados y modificados muchos más registros que si la información estuviera almacenada con sólo una restricción. Sin embargo, si se tuviera la capacidad de almacenar la restricción en sí, la información sería completa y el espacio necesario para almacenarla muy pequeño, derivado de la compresión de la información. Además no sería necesario tener distintos tipos de información ni formas de almacenarlas en función de la morfología de la restricción, tanto la figura 1.1.a como la 1.1.b, serían de tipo *Restricción*. De esta forma, las Bases de Datos de Restricciones extienden el modelo de las bases relacionales clásicas, permitiendo tratar gran cantidad de datos o incluso infinitos representados de forma finita. Esto se realiza utilizando restricciones numéricas, como una combinación lógica (booleana) de ecuaciones e inecuaciones matemáticas.

Al utilizar datos más complejos, las Bases de Datos de Restricciones necesitan tener definido un lenguaje de consulta donde estén contempladas las características de los nuevos datos, diseñar cómo se almacenarán las restricciones y cómo se evaluarán las consultas de una manera eficiente. No es sólo importante saber cómo se almacenará la información, igual de importante es conocer cómo se inferirá conocimiento cuando la información se almacena de manera intensiva, mediante restricciones, en lugar de extensiva.

1.1. Ejemplos Motivadores

Para entender mejor qué tipos de problemas solucionan las Bases de Datos de Restricciones, se muestran tres ejemplos que se analizarán en mayor profundidad a lo largo de esta tesis. Estos ejemplos justifican que las bases de datos relacionales no son suficientemente expresivas para tratar datos no tradicionales. Los ejemplos que se muestran a continuación, pueden ser almacenados en una Base de Datos de Restricciones, no teniendo que buscar soluciones a medida para cada caso, y reduciendo la complejidad de las aplicaciones que utilizan dichos datos, ya que será resuelta por el gestor de bases de datos. Al igual que no es necesario adaptar los gestores de bases de datos relacionales para tratar datos de tipo administrativo, se busca definir un gestor de bases de datos para trabajar con restricciones.

- **Sistemas de Información Geográfica:**

Las BDdR han utilizado el ámbito de los sistemas de información geográfica, como uno de sus más destacables casos de estudio. Esto es debido a que los datos espacio-temporales son fácilmente representables mediante restricciones. Un ejemplo podría ser una base de datos que almacenara toda la información relacionada a las cosechas de una región, por lo que se podrían almacenar los propietarios, el tipo de plantación, u otros datos de tipo económico. Estos datos pueden ser almacenados en una base de datos relacional clásica, mientras que las parcelas podrían ser representadas mediante restricciones. Un ejemplo de las parcelas es el mostrado en la figura 1.2, donde también se representa el recorrido de un avión que va a fumigar dichas cosechas. El recorrido del avión también puede ser representado mediante restricciones donde la variable *tiempo* está involucrada.

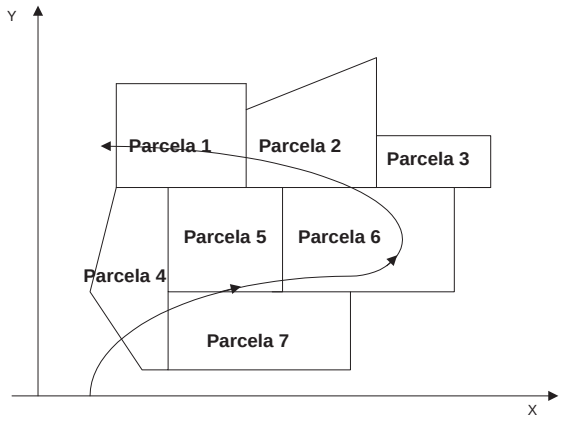

Figura 1.2: Ejemplo Motivador I. Sistema de Información Geográfica

Este tipo de información da cabida a consultas donde pueden estar involucrados datos tradicionales, como:

- ¿Quién es el propietario de la parcela 6?
- ¿Qué tipo de cosecha tiene la parcela 3?

Pero hay consultas que no pueden ser resueltas sin un tratamiento especial de las restricciones que representan la información, como por ejemplo:

- ¿Qué parcelas no son atravesadas por el avión?
- ¿Quiénes son los propietarios de las parcelas atravesadas por el avión?
- ¿Qué parcelas serán atravesadas por el avión entre los instantes de tiempo5 y 15?

- ¿Qué parcelas están al sur de la parcela 2?

- **Sistemas Económicos:**

  Los sistemas económicos también se pueden utilizar como un ejemplo apropiado para el uso de BDdR, ya que gran cantidad de datos, como los impuestos a pagar en función de los ingresos o el crecimiento del mercado a lo largo del tiempo, no pueden ser representados directamente en las bases de datos relacionales clásicas. Es posible encontrar variables que dependen de otros parámetros, no tomando valores únicos, como es el caso de los impuestos en función de los ingresos representados en la figura 1.3. En dicha figura se muestran los impuestos que se tienen que pagar, para cuatro países, en función de los ingresos.

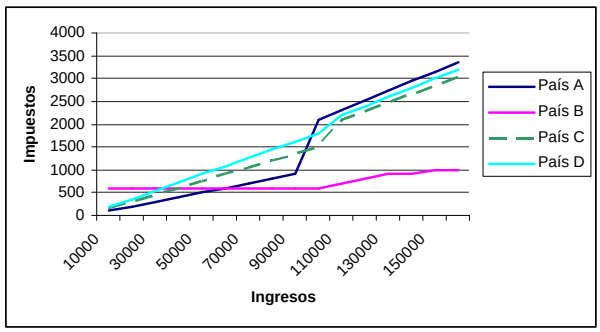

Figura 1.3: Ejemplo Motivador II. Sistema Económico

Si no se utilizaran restricciones, sería necesario almacenar todas las posibles relaciones entre *Ingresos* e *Impuestos* para todos los países. Esto generaría gran cantidad de tuplas y una información incompleta, ya que las combinaciones son infinitas, inconveniente que evita el uso de BDdR. Además, si la información se representa mediante restricciones, se podrían realizar consultas como:

- ¿En qué país se pagan menos impuestos para unos ingresos de 50.001 euros?
- ¿Qué impuestos se pagan en el país C para unos ingresos de 34.000 euros?
- ¿Para cualquier ingreso se paga menos en el país D que en el país A?

En la tesis de Gross [68] se muestra otro ejemplo económico, donde una empresa quiere almacenar la información relativa a sus labores de importación y exportación, junto a datos administrativos propios de cualquier empresa como son empleados, departamentos, productos ... Además de estos datos, tiene información relativa a

los impuestos que pagará en función de si es compra, venta, del país y la empresa con la que negocie, o de cómo fluctúe el mercado de valores. Para la empresa no sólo será un reto almacenar la información, también poder obtener el mayor rendimiento de ella, pudiendo minimizar gastos o prevenir cambios económicos. Sin duda, el ideal de tener una base de datos que pueda trabajar con esta información compleja al igual que con los datos administrativos, facilitaría el desarrollo de las aplicaciones que usan dichos datos, siendo el objetivo que persiguen las BDdR.

- **Diagnosis de Fallos Basada en Modelos:**

  Pese a que los datos espacio-temporales han sido el motor de las primeras justificaciones para la utilización de las bases de datos con restricciones, también existen otros campos que podrían ser sensiblemente mejorados, éste es el caso de la diagnosis de fallos basada en modelos. Aunque en el capítulo 7 se explica con más detalles en qué consiste la diagnosis de fallos, en este punto se describen algunas ideas para motivar su uso en las BDdR.

  La idea de la diagnosis basada en modelos descrita en [127] radica en la comparación entre el comportamiento esperado de un sistema y el comportamiento observado. El comportamiento esperado del sistema se establece mediante la creación de un modelo que describa cada uno de los subsistemas que lo forman mediante restricciones, y la relación entre ellos mediante variables. Para obtener el comportamiento observado es necesario colocar sensores en el sistema que va a ser diagnosticado. Un ejemplo muy utilizado para introducir la diagnosis basada en modelos es el mostrado en la Figura 1.4. Dicho ejemplo muestra 5 componentes, de los cuales 3 son multiplicadores (M_i) y 2 sumadores (A_i), relacionados mediante las variables (x, y, z) y con sensores colocados en las entradas y las salidas (a, b, c, d, e, f, g). Para almacenar esta información en una base de datos, será necesario utilizar una BDdR, ya que las combinaciones de valores que pueden tomar las variables son infinitos, y sería imposible almacenarlos extensivamente utilizando una base de datos relacional.

  El ejemplo de la diagnosis basada en modelos no es un caso de estudio más donde son necesarias las Bases de Datos de Restricciones por tratar datos no tradicionales, el eje de su aportación radica en que, en oposición a los datos espacio-temporales, existe mayor número de variables, y el valor de ellas puede depender a su vez de otras, teniendo una clara dependencia. En los sistemas de información geográfica, se trabajará como máximo con cuatro dimensiones (x, y, z, t), mientras que los

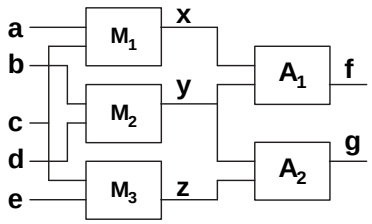

Figura 1.4: Ejemplo Motivador III. Sistema para la Diagnosis Basada en Modelos

sistemas formados por componentes pueden estar constituidos por gran cantidad de variables. Otra característica de la diagnosis basada en modelos es la relación entre el comportamiento de los distintos componentes, ya que los valores de salida de un componente no sólo está determinado por él, sino también por los valores de las variables que le suministren otros componentes. Esta ampliación del número de variables y de la complejidad del sistema, fuerza la definición de nuevos tipos de consultas y la búsqueda de diseños de bases de datos que las soporten, más allá de las propuestas en los sistemas de información geográfica.

Entre los ejemplos de consultas sobre este tipo de sistemas, donde existen restricciones involucradas, se encuentran:

- ¿Qué valor tiene f si $a = 4$, $b = 1$, $c = 2$, $d = 4$?
- ¿Qué componente está fallando si los sensores obtienen los valores...?
- ¿Qué componentes pueden estar fallando si la variable g es incorrecta?
- Obtener posibles valores que toman los sensores si el sistema funcionara correctamente. Lo que significa obtener instancias de las variables a, b, c, d y f. Un ejemplo de salida es: $\{a, b, c, d, e, f, g\} = \{1, 1, 2, 5, 3, 7, 11\}$.
- ¿Qué relación de comportamiento tienen las variables a, b, c, d y f? En este caso se obtendría una nueva restricción $(a * c + b * d = f)$ que relaciona las variables de manera simbólica.

1.2. Contexto del Problema

El modelo relacional propuesto por Codd [33], considerado como uno de los pilares más importantes en las bases de datos relacionales, está dividido en dos niveles, *lógico* y

físico. La capa lógica es sobre la cual se especifican las consultas, mientras que la capa física describe el almacenamiento físico de los datos, la organización de ficheros y las rutas de acceso a la información. Desde que se definió este modelo, la complejidad de los tipos de datos que se deben tratar ha ido creciendo, por lo que el modelo relacional ya no es suficiente. Estas nuevas necesidades fueron recogidas en los años80 con la aparición de la orientación a objetos y la consecuente adaptación de las bases de datos [3], aunque alejándose en algunos aspectos del modelo relacional original, características que se analizarán en el capítulo 2. El siguiente paso en la adaptación a datos más complejos, fue la creación de las bases de datos objeto-relacionales [143], que dan cabida a la orientación a objetos pero acercándose más a la estructura del modelo relacional.

En paralelo con estos cambios en las bases de datos surgieron las Bases de Datos de Restricciones, con el trabajo de Kanellakis, Kuper y Revesz [84], por lo que han convivido con muchos de los avances sufridos en las bases de datos relacionales, aunque no siempre haciéndose eco de las ventajas que éstas ofrecían. Las Bases de Datos de Restricciones se originaron basándose en las investigaciones relacionadas con DATALOG [28] y la Programación Lógica con Restricciones (Constraint Logic Programming - CLP) [34][82]. Inicialmente, la motivación consistió en combinar los trabajos en ambas áreas, con el objetivo de obtener versiones más óptimas de la Programación Lógica basada en Restricciones utilizando el modelo de las bases de datos. Para abordar esta propuesta, se partió de un modelo de datos introducido en [93], fundamentado en considerar una tupla de una base de datos como un conjunto de restricciones de igualdad o desigualdad en los atributos de dicha tupla, siendo las restricciones un mecanismo natural de especificar las consultas de similitud en series de datos. Un ejemplo aclarador de esta afirmación es el presentado en [94] mostrado en la figura 1.5, donde un conjunto de tuplas de una relación puede ser representado por una restricción. Cada tupla de una relación está relacionada con las restantes tuplas con el operador lógico de disjunción ($\vee$), mientras que los atributos que forman una tupla tienen una relación lógica de conjunción ($\wedge$) entre ellos.

La mayor contribución aportada por las BDdR no radica simplemente en utilizar las restricciones como objetivo, sino como medio para una mayor expresividad en los datos. Esto originó cambios en el modelo propuesto por Codd, incrementando el nivel de abstracción entre la capa física de datos y la semántica que dichos datos pueden tomar. Esto alentó a incluir una capa adicional, analizada en [60], a la arquitectura conceptual de las bases de datos que se muestra en la figura 1.6. Esta capa es el resultado de dividir

x	y
a_1	b_1
a_2	b_2
a_3	b_3

$$\equiv (x = a_1 \wedge y = b_1) \vee (x = a_2 \wedge y = b_2) \vee (x = a_3 \wedge y = b_3)$$

Figura 1.5: Ejemplo de representación de un conjunto de tuplas como una restricción

la capa lógica de las bases de datos clásicas en dos nuevas capas: *abstracta* y *concreta*. La capa *lógica abstracta* representa un conjunto de datos que pueden ser infinitos, como las relaciones entre variables mostrados en los ejemplos de las figuras 1.2, 1.3 ó 1.4, y deberá soportar la semántica de las consultas sobre dichos datos. Mientras que en la capa *lógica concreta* se representará dicha información de una manera finita mediante restricciones.

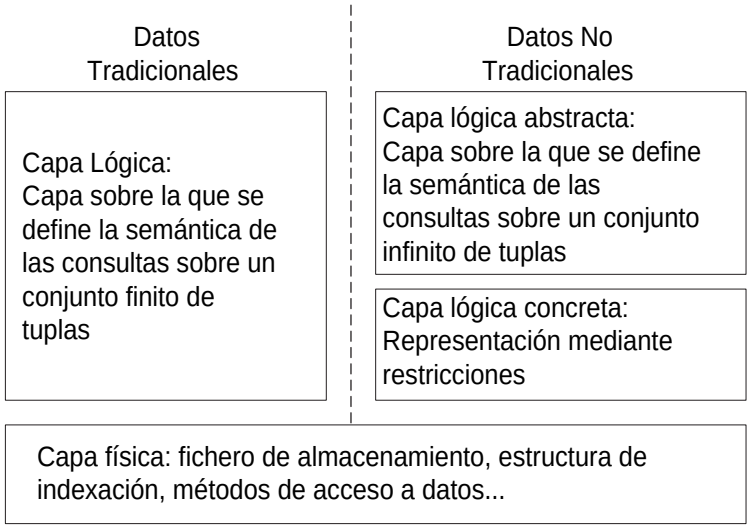

Figura 1.6: Modelos de datos relacional y con restricciones

Desde que se originaron las Bases de Datos de Restricciones hasta la actualidad, han surgido muchas teorías y propuestas pese a ser una comunidad pequeña. Y aunque muchos de los problemas ocasionados por el tratamiento y almacenamiento de restricciones fueron detectados y solucionados, no han sido incluidos todos los beneficios de las bases de datos relacionales y las objeto-relacionales. Las propuestas en el campo de las BDdR se mantienen alejadas de las bases de datos relacionales, haciéndose patente la pérdida de eficiencia en la evaluación de las consultas, en el almacenamiento, obtención y modificación de los datos, tal como se analizará a lo largo de esta tesis.

1.3. Justificación del Trabajo

Aunque las bases de datos comerciales se han desarrollado mucho para adaptarse a las nuevas necesidades, tanto en aspectos de expresividad como en eficiencia, sin duda queda mucho camino por recorrer. Una deficiencia destacable es que el tratamiento de los datos no tradicionales no está homogeneizado, y no existen propuestas fácilmente adaptables a los distintos tipos de problemas. Esto significa que las aplicaciones tienen que adaptarse a las propuestas existentes, como ocurre en los sistemas de información geográfica, en lugar de buscar soluciones que se adapten a las aplicaciones. Un ejemplo puede ser una empresa que tiene que tratar datos que no sólo son de tipo administrativos (nombre, teléfono, edad ...), como por ejemplo las curvas que describan sus beneficios por departamentos en función de distintos parámetros. Dicha empresa se enfrentará a mucho más que realizar el diseño y la implementación necesaria en cualquier aplicación, tendrá que decidir cómo se almacena la información y cómo será tratada para inferir el conocimiento que necesite en cada momento. Esta problemática es cada vez más frecuente, ya que cada vez los datos que deben ser almacenados son más complejos.

Aunque existen gran cantidad de prototipos de BDdR, hay características de gran importancia a las cuales no se les ha prestado la suficiente atención. Como se ha presentado en el apartado de *Contexto del Problema*, se ha definido la extensión del modelo relacional para las BDdR, quedando por estudiar en profundidad aspectos de importancia como:

- Los prototipos desarrollados hasta el momento no explotan por completo las ventajas de las bases de datos relacionales, tratando todas la información como restricciones e ignorando las ventajas del paradigma relacional para los datos que toman valores finitos.

- Otro de los inconvenientes de las soluciones existentes es que los lenguajes y arquitecturas propuestas son dependientes de la índole de los datos que tratan, mayoritariamente orientado a sistemas geográficos. Esto provoca que no sean soluciones genéricas y con sintaxis alejadas de SQL, lo que supone un mayor esfuerzo por parte del usuario para utilizar una BDdR.

- Ya que la semántica debe ser redefinida para los datos representados como restricciones, ¿los lenguajes propuestos hasta el momento para trabajar con BDdR tienen las mismas capacidades que el lenguaje SQL definido para las bases de datos

relacionales clásicas? El objetivo de esta pregunta es conocer si se pueden hacer el mismo tipo de consultas sobre los datos que son almacenados como restricciones, que sobre los datos tradicionales. A lo largo de esta tesis se darán las explicaciones pertinentes para demostrar que la respuesta a esta pregunta es**no**.

- Es fácil definir un modelo que represente las restricciones desde un punto de vista lógico y físico como se muestra en la figura 1.6, pero ¿es tan fácil realizar la transformación de la restricción entre una representación y otra en la práctica?; y ¿realmente se han ofrecido soluciones eficientes a nivel computacional? Esta pregunta sin duda está relacionada con una de las mayores deficiencias en el campo de la investigación de las Bases de Datos de Restricciones, y la razón por la que ha tenido épocas de pocos avances, la dificultad de pasar de la teoría a la práctica. De esto también se deriva la falta de estudios a nivel de optimización en la evaluación de consultas.

- Pese a que las restricciones se utilizan como herramienta para representar relaciones entre valores, no se puede olvidar la importancia de los elementos que componen y relacionan las restricciones, las variables. ¿Las propuestas en las BDdR han definido consultas sobre dichas variables? ¿Cómo se infiere información relativa a las variables y cómo relacionan distintas restricciones? Esta problemática es otra de las asignaturas pendientes en las BDdR. Por ejemplo para casos más complejos donde hay relacionadas muchas variables, como el ejemplo de los impuestos comentado con anterioridad, ¿cómo se obtiene qué impuesto es el mínimo a pagar si esta información no está almacenada explícitamente?

1.4. Objetivos de la Tesis

El objetivo general de esta tesis es **Definir un Gestor de Base de Datos de Restricciones que permita almacenar y consultar datos clásicos, restricciones y las variables que las forman, al igual que las bases de datos tradicionales tratan los datos clásicos**. A este Gestor de Bases de Datos de Restricciones se le llamará LORCDB, y para obtener este objetivo general se presentan distintos subobjetivos:

- **Redefinir el concepto de Base de Datos de Restricciones**para que se acerque lo más posible a las bases de datos relacionales. Por esta razón se define un nuevo

tipo de dato *Restricción* que tendrá la capacidad de representar un número de datos infinito de forma finita, permitiendo que la información sea más completa y precisa. Esto significa que tanto la sintaxis como la semántica de las consultas tiene que cambiar para contemplar este nuevo tipo reflejando su complejidad.

- **Redefinir formalmente las operaciones del álgebra relacional para el tratamiento de restricciones**. Existen cinco operaciones primitivas en el álgebra relacional (selección, proyección, producto cartesiano, unión y diferencia) con las que se pueden definir el resto de las operaciones sobre relaciones. Al tener un tipo de datos *Restricción*, estas cinco operaciones y sus operadores tienen que ser analizadas y ampliadas.

- **Ampliar el lenguaje SQL para el tratamiento de restricciones** Acorde con el álgebra relacional definida para restricciones, se describirá una extensión de SQL dando origen a un nuevo lenguaje llamado CORQL (Constraint Object-Relational Query Language).

- **Diseñar una arquitectura que permita almacenar restricciones, junto a la utilización de las técnicas necesarias para la inferencia de conocimiento de la información de la base de datos**. La arquitectura debe ser modular, fácilmente configurable y que permita adaptar bases de datos ya existentes, facilitando su migración. Para esto se utilizará un gestor de Bases de Datos Objeto-Relaciones, en este caso Oracle^{TM} 9.*i*. Para evaluar las consultas e inferir nuevo conocimiento, la arquitectura se apoyará en dos familias de técnicas: numéricas y simbólicas.

- **Analizar la complejidad computacional de la evaluación de cada uno de los operadores, buscando soluciones eficientes**. Para mejorar la eficiencia en la evaluación de consultas, se proponen un conjunto de mejoras que se explicarán con detalle a lo largo de la tesis. Entre ellas caben destacar la indexación de variables y restricciones, el etiquetado de restricciones por su tipo para su resolución, el almacenamiento de envolventes de los dominios de las variables, y la definición de heurísticas que ayuden a la resolución de problemas de satisfacción de restricciones.

- **Buscar una solución adaptable a distintos problemas** La arquitectura que se propone servirá como capa lógica para el tratamiento de cualquier aplicación cuyos datos se puedan representa mediante restricciones polinómicas y lineales. Con

el objetivo de mostrar cómo se utiliza para distintas aplicaciones, se usarán distintos ejemplos donde al arquitectura facilita el tratamiento de los datos, ejemplos económicos, sistemas de información geográfica y diagnosis de fallos basada en modelos. La incorporación de las BDdR no sólo facilita el desarrollo de aplicaciones al incorporar gran parte de la lógica a la base de datos, también implica la mejora de las aplicaciones en sí. En el caso de la diagnosis de fallos, muchas de sus fases han sido mejoradas gracias a la utilización de las BDdR y validado por la comunidad de diagnosis, más allá de la claras ventajas que ofrece tener almacenada la información de un sistema a nivel de persistencia y facilidad de acceso.

Todos estos objetivos dan como resultado una arquitectura sobre Bases de Datos de Restricciones a la que se ha llamado **Labelled Object-Relational Constraint Database - LORCDB**, intentando reflejar algunas de las características de dicha arquitectura, como es la utilización de bases de datos objeto-relacionales, y el etiquetado de restricciones para facilitar la evaluación de consultas.

1.5. Marco de la Tesis

Esta tesis doctoral se ha desarrollado en el marco de los siguientes proyectos de investigación:

- **Desarrollo de herramientas basadas en modelos semicualitativos para el análisis de sistemas dinámicos. Aplicación a la supervisión, control y predicción de comportamiento (CICYT** $DPI2000-0666-C02-02$**).** Dentro del proyecto, la tarea relacionada con esta tesis consistía en ampliar el tipo de conocimiento con el objetivo de realizar las consultas sobre sistemas semicualitativos tanto estáticos como dinámicos. En este sentido, se propuso un lenguaje de consultas sobre las propiedades cualitativas del estacionario y del transitorio de los sistemas dinámicos semicualitativos sujetos a restricciones. Este proyecto dio como resultado un conjunto de herramienta para el tratamiento computacional de las bases de datos empíricas, evitando información no relevante para su posterior tratamiento con técnicas de aprendizaje supervisado.

  Investigador principal: Rafael Martínez Gasca

  Período de duración: $2000-2003$

- **Automatización de la detección y diagnosis de fallos de sistemas estáticos y dinámicos usando conocimiento semicualitativo (CICYT$DPI2003-07146-C02-01$).** Este proyecto abarca la diagnosis basada en modelos con tratamiento simbólico de la información. La descripción del sistema se presenta mediante un conjunto de restricciones, almacenadas en una Base de Datos de Restricciones. El tratamiento de la información utilizando como capa lógica las bases de datos, no sólo mejoraron la persistencia, recuperación y fácil composición del los sistemas, también mejoró muchas de las fases del proceso de diagnosis. Gracias a la utilización de las Bases de Datos de Restricciones en la diagnosis basada en modelos, se plantearon nuevas necesidades semánticas de las restricciones y su tratamiento.

  Investigador principal: Rafael Martínez Gasca

  Período de duración: $2003-2006$

- **Detección automática de fallos, diagnosis y tolerancia a fallos en sistemas con incertidumbre y distribuidos. (CICYT$DPI2006-15476-C02-00$).** Este proyecto aborda entre sus tareas el tratamiento de los datos relativos a la diagnosis cuando el sistema está distribuido, tanto semánticamente como estructuralmente. En este caso, las Bases de Datos de Restricciones tienen que ser ampliadas para el tratamiento de restricciones distribuidas con la complejidad que conlleva la dispersión de los datos.

  Investigador principal: Rafael Martínez Gasca

  Período de duración: $2006-2009$

- **Asistente virtual interactivo: Información para orientación laboral (AVIOL)** Dentro de este proyecto se realizan las tareas relacionadas con el acceso a bases de datos heterogéneas: 'Análisis y diseño de KSB (Knowledge Service Bus)', 'Desarrollo de KSB', 'Configuración, parametrización y pruebas' y 'Puesta en marcha y simulación'. Proyecto realizado en colaboración con Sadiel S.A.

  Investigadores responsables: Rafael Martínez Gasca y María Teresa Gómez

  Período de duración: $2007-2008$

1.6. Estructura de la Tesis

El resto del documento está estructurado en los siguientes capítulos:

- **Capítulo 2. Antecedentes de las Bases de Datos** Las bases de datos han ido mejorado en los últimos años y estos avances han afectado claramente a las BDdR. En este capítulo se analizan diferentes características y su relación con las Bases de Datos de Restricciones, como los modelos lógicos, la naturaleza de la información y el dominio de las aplicaciones.
- **Capítulo 3. Antecedentes de las Bases de Datos de Restricciones** Desde los años 90, muchas han sido las definiciones y los prototipos desarrollados para BDdR, en este capítulo se analizarán dichas aportaciones y qué características pueden ser mejoradas.
- **Capítulo 4. Propuesta de Modelado para las Bases de Datos de Restricciones**. Partiendo de la definición original de Base de Datos de Restricciones, se redefine el concepto para ampliar la funcionalidad de las propuestas existentes. Esta redefinición conlleva la necesidad de ampliar la sintaxis y la semántica del álgebra relacional para restricciones, junto a las técnicas utilizadas para la evaluación de las consultas.
- **Capítulo 5. Arquitectura LORCDB. Un Gestor para BDdR** Este capítulo muestran la arquitectura propuesta con sus distintos módulos, los cuales permiten utilizar distintas técnicas en función del tipo de consulta que se realice y las restricciones relacionadas con dicha consulta. En esta parte de la tesis se trata el paso de la capa lógica a la física, aportando soluciones para mejorar la eficiencia, y dando soporte a un nuevo lenguaje de consulta llamado CORQL.
- **Capítulo 6. Implementación del lenguaje CORQL.** Muestra los detalles de diseño e implementación de las LORCDB, junto a cómo se almacenan las restricciones. Al representar la información de manera intensiva en forma de restricción, su tratamiento es computacionalmente costoso. Para paliar este hecho, la arquitectura se ayuda de técnicas de indexación de información, junto al desarrollo de un conjunto de algoritmos relacionados con cada una de las operaciones del álgebra relacional para restricciones. Los algoritmos se presentan sobre un ejemplo para un sistema de gestión de impuestos.

- **Capítulo 7. La Diagnosis de Fallos Basada en Modelos: Un caso de estudio** En este capítulo se presenta cómo la utilización de BDdR puede facilitar y mejorar el proceso de diagnosis de fallos en un sistemas. La diagnosis basada en modelos nunca ha sido utilizada como caso de estudio en el ámbito de las BDdR, y describe problemas de características muy diferentes a los sistemas de información geográfica. Utilizar las BDdR en áreas tan dispares es posible gracias a que la arquitectura LORCDB no es dependiente del problema.

- **Capítulo 8. Optimización y Pruebas en la Evaluación de Consultas en BDdR.** En el capítulo 6 se presentan las características de las implementaciones para evaluar de manera más eficiente las consultas, y en este capítulo se muestra de manera empírica cómo mejoran los tiempos de evaluación de consultas. Se realizan análisis para cada una de las operaciones del álgebra relacional que trabajan con restricciones, presentando resultados sobre sistemas para diagnosis de fallos basada en modelos, junto a sistemas de información geográfica.

- **Capítulo 9. Conclusiones y Trabajos Futuros** Finalmente se presentan las conclusiones de la tesis y las líneas de investigación abiertas que quedarán como trabajos futuros.

- **Anexo A. Algoritmos para la Proyección Simbólica** Este anexo presenta los detalles relativos a los algoritmos relacionados con la operación de proyección simbólica sobre atributos de tipo variable de restricción.

- **Anexo B. Tablas de Restricciones para el Análisis de Eficiencia.** Muestra las tablas de las Bases de Datos de Restricciones utilizadas en las pruebas empíricas del capítulo 8. También se presenta en este anexo algunos detalles de una aplicación desarrollada para obtener las restricciones que representan información relativa a sistemas de información geográfica.

Capítulo 2

Antecedentes de las Bases de Datos

La mayoría de las aplicaciones actuales hacen uso de las bases de datos para hacer persistente su información. Al ser un denominador común, la creación de métodos eficientes para la evaluación de consultas ha sido un área de investigación y desarrollo de gran importancia. Los distintos desarrollos han dado origen a diversas soluciones de características distintas. En este capítulo se analizan tres de las características fundamentales que ayudan a clasificar y definir las bases de datos. Esta clasificación ayudará a recorrer las distintas propuestas que se han desarrollado alrededor de las bases de datos hasta la actualidad, en función de:

- El dominio de las aplicaciones que utilizan y son almacenadas en las bases de datos.
- La naturaleza de la información a almacenar.
- El modelo lógico de almacenamiento que usan, y por extensión los lenguajes de consulta que se definen para utilizarlas.

Estas características describen aspectos que diferencian una base de datos de otra, con la idea de cumplir principalmente dos objetivos, que han sido los motores del desarrollo de este área de conocimiento: la búsqueda de representaciones de datos más expresivos, y el desarrollo de bases de datos cada vez más eficientes a nivel de evaluación de consultas. Sin duda alguna, la creación de nuevos dominios de aplicaciones cada vez más complejos ha originado la búsqueda de nuevas soluciones, tanto para dar mayor versatilidad en la naturaleza de los datos que se almacenan, como en la definición de nuevos soportes lógicos

que les puedan dar cabida. Esta ampliación a tipos más complejos, junto al crecimiento de la cantidad de información, ha originado sistemas más complejos y por lo tanto más difíciles de evaluar.

2.1. Introducción

Desde que en la década de los 60 se empezaron a desarrollar los primeros sistemas gestores de bases de datos hasta la actualidad, se han propuesto diferentes soluciones al problema del almacenamiento y gestión de la información. El continuo crecimiento de la cantidad de información y la aparición de nuevos ámbitos de trabajos y aplicaciones, han hecho que las necesidades de las bases de datos hayan crecido en concordancia. Dicha ampliación sucesiva de las bases de datos, ha venido principalmente provocada por la búsqueda de nuevas soluciones que permitieran tener mayor capacidad expresiva para representar los datos.

Una de las características de las bases de datos que permitirán su clasificación es *el dominio de la aplicación* a la cual dan soporte, ya que dichos dominios pueden necesitar menor o mayor expresividad. Con respecto a esta división, se analizarán algunos ejemplos como las bases de datos administrativas, de información geográfica, espacio-temporales, para diseño asistido por ordenador ...

La capacidad expresiva de datos que se pueden almacenar da origen a una clasificación en función de la *naturaleza de la información que almacena la base de datos*

- **Bases de Datos Extensivas:** Son bases de datos donde la información está almacenada de forma explícita, y donde las consultas obtienen un subconjunto de la información almacenada directamente en la base de datos. Un ejemplo de información extensiva puede ser cómo se almacena una persona, mediante su edad, nombre, DNI ...

- **Bases de Datos Intensivas:** Estas bases de datos pueden *deducir* información no almacenada explícitamente en la base de datos, sino mediante un grupo que datos que representan la información completa. Un ejemplo de información intensiva es la representación del movimiento de un avión, donde se almacena la ecuación que describe su ruta, pero no todos los puntos explícitos por los que pasa.

Independientemente de que los datos se representen de forma intensiva o extensiva, se pueden utilizar distintos gestores de bases de datos para almacenarlos. El modelo lógico para almacenar la información ha ido avanzando desde los primitivos sistemas de archivos, pasando por las bases de datos relacionales, las orientadas a objetos y las objeto-relacionales, dando origen a los distintos *modelos lógicos de almacenamiento*.

En función del dominio de la aplicación, se necesitará mayor o menor capacidad expresiva y será necesario decidir cómo se representará la información (intensiva o extensiva) y qué modelo lógico se utilizará para su almacenamiento y gestión.

Estas tres características (dominio de las aplicaciones, naturaleza de la información y modelo lógico) describirán cada base de datos, como se muestra en la figura 2.1, y que se detallarán a lo largo del capítulo.

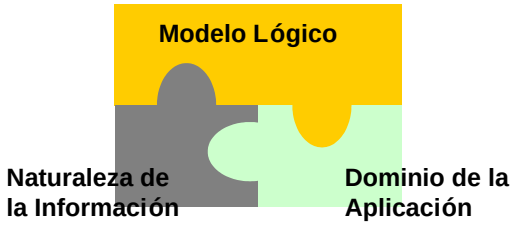

Figura 2.1: Características de las bases de datos

2.2. Dominio de Aplicación

Los incesantes cambios en los tipos de datos que gestionan las bases de datos han sido el principal impulsor de nuevas tecnologías. Aunque las bases de datos tradicionales dan cabida a gran cantidad de aplicaciones, ya que tratan datos administrativos, existen tipos de datos que no son gestionados de forma natural. En las últimas décadas la complejidad de las aplicaciones ha ido creciendo, demandando nuevas soluciones ya que la utilización de las bases de datos relacionales clásicas no es suficiente. En las próximas subsecciones se presentan algunos ejemplos de estos dominios especialmente relacionados con esta tesis.

2.2.1. CAD - Computer-Aided Design y CAM - Computer-Aided Manufacturing

El diseño asistido por ordenador (Computer-Aided Design) [154] trabaja con datos relativos a diseños mecánicos y eléctricos de coches, aviones, barcos, circuitos integrados ..., involucrando software y en algunas ocasiones hardware específico. Permite diseñar en dos o tres dimensiones mediante elementos geométricos, como puntos, líneas, arcos, superficies y sólidos, para obtener un modelo numérico de un objeto o conjunto de ellos.

La fabricación asistida por ordenador (Computer-Aided Manufacturing) almacena datos similares a los que utiliza el diseño asistido por ordenador, junto a datos relativos a la producción de elementos discretos, como el montaje, y a la producción de carácter continuo, como la síntesis química. Estas aplicaciones están a menudo organizadas en una jerarquía, existiendo una aplicación de nivel superior que monitoriza procesos individuales de fabricación. Estas aplicaciones deben responder en tiempo real y ser capaces de ajustar los procesos para mantener un rendimiento óptimo con unas tolerancias mínimas. Estas aplicaciones utilizan una combinación de algoritmos estándar y reglas personalizadas para responder a diferentes condiciones dependientes del sistema. Toda esta información debe ser almacenada en la base de datos, alejándose mucho de las características de los datos administrativos, por lo que claramente necesitará soportes de almacenamiento con otras cualidades. Un ejemplo de sistema que utiliza esta tecnología es FORAN [138], una aplicación que ayuda al diseño de barcos ofreciendo soluciones al ensamblaje, análisis de probabilidades de error en la construcción, características eléctricas...

La geometría de los objetos que se describen en estos sistemas, se definen mediante un conjunto de puntos, líneas, círculos, conos,..., en 2 ó 3 dimensiones. Además de su forma geométrica, se pueden definir su comportamiento individual y su relación con el resto de objetos, junto a la planificación de acciones para la composición entre ellos para construir objetos más complejos [135][139].

Entre las características que hacen más complejos estos tipos de sistemas destacan:

- El diseño no es estático, sino que evoluciona a lo largo del tiempo. Esto supone que las estructuras que almacenan el diseño del sistema tienen que ser fácilmente adaptables.

- Los sistemas suelen tener muchas dependencias topológicas y funcionales, lo que significa que un cambio afectará a gran cantidad de objetos del sistema. El cambio del diseño puede conllevar cambios en otras partes del diseño, y muy probables cambios en los programas que acceden a ellos, lo que resulta muy costoso de mantener.
- El problema de la *ingeniería cooperativa* es usual, donde distintos roles de usuarios pueden acceder al sistema teniendo que tomar decisiones parciales para obtener un resultado final coherente.

2.2.2. CIM - Computer Integrated Manufacturing y PDM - Product Data Management

La fabricación integrada por ordenador (Computer Integrated Manufacturing) es un área que se originó ante la creciente competitividad entre las empresas en el mercado actual. El mundo empresarial ha establecido que, estratégicamente hablando, la integración de todas las áreas de una empresa en un sistema único es la opción más viable para incrementar su productividad y crear empresas más competitivas. John W. Bernard lo define como "la integración de los computadores digitales en todos los aspectos del proceso de manufactura", ya que se trata de un sistema complejo, de múltiples capas, diseñado con el propósito de minimizar los gastos y crear riqueza en todos los aspectos.

La gestión de los datos del producto (Product Data Management) es un área orientada a controlar los datos relativos a los productos. Estos sistemas crean y controlan las relaciones entre los conjuntos de datos que definen un producto, almacenando dichas relaciones en una base de datos. Este tipo de aplicaciones son importantes para la gestión del ciclo de vida del producto, pudiendo contemplar desde los requisitos, especificaciones, planificación de fabricación o ensamblaje, y pruebas sobre el producto final.

La problemática de ambas tecnologías es que deben intervenir en todo el proceso de fabricación, desde la materia prima hasta el producto final. Esto conlleva que todos los departamentos, pese a ser independientes, tienen que estar coordinados para obtener el producto final, teniendo en cuenta tanto la característica software como hardware del proceso.

2.2.3. CASE - Computer-Aided Software Engineering

La ingeniería del software asistida por ordenador (Computer-Aided Software Engineering) almacena datos relativos a las etapas del ciclo de vida del software: planificación, recopilación y análisis de requisitos, diseño, implementación, pruebas, mantenimiento y documentación. Al igual que ocurre en la fabricación asistida por ordenador o la fabricación integrada por ordenador, la ingeniería cooperativa es imprescindible. Las herramientas CASE deben permitir compartir de forma concurrente el diseño del proyecto, el código y la documentación. También debe controlar las dependencias entre dichos componentes y ayudar a la gestión de cambios, facilitando la coordinación de diversas actividades de gestión de proyecto y la monitorización del progreso.

2.2.4. GIS - Geographic Information System y Bases de Datos Espacio-Temporales

Los sistemas de información geográfica (Geographic Information System) [71][125] tienen especial importancia en esta tesis, ya que es el caso de uso más extendido en las Bases de Datos de Restricciones. Buena parte de los datos de dichos sistemas derivan de las fotografías vía satélite, por lo que la cantidad de información tiende a ser muy grande. Las consultas sobre las bases de datos que almacenan dicha información pueden implicar la identificación de características, basándose por ejemplo en la forma, el color o la textura, utilizando técnicas avanzadas de reconocimiento de patrones. En los sistemas de información geográfica se pueden establecer relaciones entre datos de localización, como direcciones, edificios, parcelas y calles. También es posible representar distintos tipos de conocimiento sobre las regiones geográficas, lo que se describe mediante diferentes capas para que sean fácilmente tratables y entendibles.

Creación de Datos

Las aplicaciones modernas de sistemas de información geográfica se basan en la digitalización de la información, para lo cual existen un conjunto de métodos. La mayoría de ellos transforman los mapas impresos a digitales utilizando *diseño asistido por ordenador* y referencias geográficas.

Los distintos objetos que conforman un sistema geográfico deben estar relacionados, por ejemplo para establecer las correspondencias entre la información meteorológica, obtenida mediante satélite, con las regiones políticas de un mapa. Para que un sistema geográfico tenga la capacidad de relacionar distintas capas de información, se utilizan referencias espaciales mediante variables (típicamente x, y y z como coordenadas de longitud, latitud y altitud respectivamente). A su vez, esto le dará la posibilidad de trabajar con diferentes escalas.

Representación de Datos

Los datos de los GIS representan objetos reales del mundo, como calles, terrenos o montañas, mediante datos digitales. Estos datos del mundo real pueden ser divididos en dos grupos: objetos discretos (como una casa) y áreas continuas (como zonas de nubosidad). Existen dos métodos para almacenar estos tipos de información: mediante mallas rectangulares o componentes geométricos.

Las **mallas rectangulares (raster)** consisten en celdas distribuidas mediante filas y columnas, y donde cada celda lleva un valor discreto asociado, como un color, la cantidad de lluvia registrada o el tipo de terreno. Cada una de las celdas representará la información relativa a una porción del mapa, que normalmente tiene forma cuadrada.

Los **componentes geométricos** son puntos, líneas y polígonos. Son especialmente frecuentes para representar las elevaciones del terrenos en sistemas en tres dimensiones, utilizando técnicas de triangularización de superficies. Dichos componentes pueden a su vez tener información asociada como atributo del objeto

En ambos tipos de modelos existen tanto ventajas como inconvenientes, y se utilizará una implementación u otra en función de las características de la información que se almacena. En el caso de la representación mediante mallas, generalmente se necesitará mucho mayor espacio de almacenamiento que en la utilización de componentes geométricos. Sin embargo, el tamaño utilizado por la malla es estable, no depende de la cantidad de objetos que tenga el sistema, no como ocurre en los sistemas con componentes geométricos. A cambio, las implementaciones y las consultas con mallas de datos son mucho más fáciles que utilizando componentes geométricos. De todas formas existen métodos que transforman la información representada en un formato al otro.

Cuando se representan objetos que también tienen características espaciales y temporales, pero no exclusivamente ligadas a la información geográfica se hablará de *bases de datos espacio-temporales*. Un ejemplo de estas aplicaciones puede ser el control de tráfico de una ciudad, el tráfico aéreo en un aeropuerto, el movimiento de un robot o el análisis de fenómenos meteorológicos, donde es importante saber la variación en el tiempo de un objeto. Estas bases de datos añaden la capacidad de asociar un atributo tiempo a los objetos que se tratan en los sistemas de información geográfica de una manera estática. Esta característica de movilidad amplía mucho los tipos de consultas que se pueden realizar, como cuándo lloverá en una provincia o si dos aviones se encontrarán en su ruta.

2.3. Naturaleza de la información

Como se ha introducido en el comienzo del capítulo, una forma de clasificar las bases de datos es en función de la naturaleza de la información que almacena, si es de manera intensiva o extensiva. Esta división no es excluyente, ya que todas las bases de datos intensivas pueden tener características extensivas, permitiendo que ambas características convivan.

- **Bases de Datos Extensivas:**

  Son aquellas que almacenan la información de forma explícita, y donde las consultas devolverán un subconjunto de los datos almacenados. También se pueden llamar *por inducción* o *por extracción*, ya que la salida de una consulta sólo puede contener información almacenada explícitamente en la base de datos. Las características de *inducción* la poseen todas las bases de datos, independientemente de su modelo lógico o implementación, ya que todas tienen que dar la posibilidad de seleccionar una parte de la información que almacenan. Esta forma de almacenamiento es la más usada en las aplicaciones administrativas, por ejemplo para almacenar todas la información fiscal de un conjunto de personas, junto a sus datos personales. En este ejemplo, las consultas del tipo *obtener los nombres y direcciones de las personas cuyos impuestos superen los* 30.000 *euros* significará seleccionar las personas con dicha característica y devolver sólo los campos relativos a su nombre y dirección. En este caso toda la información que se obtiene tras la consulta está almacenada en la base de datos tal cual, sólo hay que realizar una extracción de ella.

- **Bases de Datos Intensivas:**

  En el caso de las bases de datos intensivas, la información que se almacena puede ser una representación no literal o explícita de los datos. Las consultas sobre estas bases de datos pueden *deducir* información nueva, o lo que es lo mismo, conocimiento almacenado en la base de datos pero no de forma directa o explícita, sino inferido partiendo de la almacenada. Un caso de uso de este tipo de información se presenta en los sistemas espacio-temporales; por ejemplo para almacenar características de los aviones de forma extensiva, como el modelo, la compañía o el número de asientos, e información intensiva como su ruta y sus probabilidades de retraso en función del tiempo. Sobre este tipo de información no sólo es posible hacer consultas del tipo*qué aviones son de un determinado modelo*, también se podrá inferir información como *cuándo pasa por encima de una determinada ciudad*o *si dos aviones se encuentran en algún instante de tiempo*. Representar de forma explícita esta información conllevaría almacenar la posición de los aviones en todos los instantes de tiempo, lo cual sería muy costoso a nivel de recursos de espacio. Para evitar el almacenamiento explícito, las bases de datos intensivas pueden representar su información de dos maneras:

  - **Valores Representativos:** Almacenando un subconjunto de datos que describan a los restantes, por ejemplo un segmento sobre las variables$x$, y puede ser representada mediante los extremos que la definen, por ejemplo $\{x = 0, y = 1\}$ y $\{x = 4, y = 9\}$, de forma que de estos valores se puede inferir la ecuación que representa los datos restantes. Esta forma de almacenar la información, es la que se utiliza en los sistemas de información geográfica cuando los datos son representados mediante componentes geométricos.
  - **Representación Simbólica de los datos:** Utilizando una representación lógica o numérica de los posibles valores que pueden tomar, lo que para el ejemplo anterior sería almacenar la restricción $\{y = 2x + 1 \wedge 0 \leq x \leq 4 \wedge 1 \leq y \leq 9\}$.

  Existen bases de datos específicas con esta característica de *deducción* cuando la representación de los datos es simbólica. Para inferir la información necesaria, en función de la consulta, utilizan técnicas de inferencia como la programación lógica y la programación con restricciones, que se describen a continuación. Las bases de

datos que han desarrollado este tipo de inferencia son las **Bases de Datos Deductivas**, **Bases de Datos de Restricciones** y las **Bases de Datos Deductivas de Restricciones**.

2.3.1. Bases de Datos Deductivas

Los sistemas de Bases de Datos Deductivas (DDB, Deductive Databases) [36] pueden hacer inferencia de nuevo conocimiento, deduciendo nuevos datos basándose en las reglas y hechos almacenados. Estas bases de datos se originaron con la fusión de las técnicas de la Programación Lógica (LP, Logic Programming) [100][63] y las bases de datos. Entre las características de las Bases de Datos Deductivas, se encuentran:

- Se basan en un conjunto de hechos y reglas, usando lenguajes declarativos como Datalog [28][72], para especificar dichas reglas y hechos.
- Mediante procesos de inferencia pueden deducir nuevas reglas y hechos partiendo de los almacenados.

Un ejemplo de lenguaje declarativo es *Prolog* [141], pero sin duda fue *Datalog* el lenguaje específicamente diseñado para trabajar con lógica y bases de datos. Datalog es un lenguaje de consulta basado en la Programación Lógica, definido como un subconjunto sintáctico de Prolog junto a un lenguaje de consulta de bases de datos. Un programa lógico consiste en un conjunto finito de cláusulas de la forma:

$$L_0 :- L_1, \ldots, L_n.$$

donde cada L_i es un literal expresado de la forma $p_i(t_1, \ldots, t_k)$, siendo p_i un *predicado* y t_j para $1 \leq j \leq k$ términos, donde los términos pueden ser constantes o variables. La parte de la izquierda de la cláusula se llama *cabecera* o *antecedente*, y la derecha *cuerpo* o *consecuente*, y donde los literales se relacionan mediante el operador de conjunción. Cuando el cuerpo de una cláusula está vacío, se define como *hecho*, en otro caso como *regla*. Los *hechos* representarán la información extensiva de la base de datos, mientras que las *reglas* describirán la información intensiva, pudiendo inferir más reglas y hechos partiendo de las cláusulas definidas.

Como diferencia funcional tangible, con respecto a otras bases de datos, hay que destacar que las Bases de Datos Deductivas permiten consultas recursivas, característica

que no poseen las bases de datos relacionales y el lenguaje de consulta SQL-92, la cual no fue añadida hasta SQL:1999.

Datalog y las bases de datos relacionales

Datalog fue desarrollado para aplicaciones que usan gran número de hechos y que pueden ser almacenados en una base de datos relacional [149]. De forma que los predicados se pueden dividir en:

- *Predicados BDE (Base de Datos Extensiva)*, que serán aquellos predicados almacenados como relaciones en una base de datos. Si $p(a_1, \ldots, a_k)$ es un *predicado DBE*, existirá una relación P en la base de datos con la tupla $(a_1, \ldots, a_k)$. Los predicados BDE también son conocidos como datos.
- *Predicados BDI (Base de Datos Intensiva)*, los cuales se definen mediante reglas. Un *predicado BDI* puede aparecer en la cabecera o en el cuerpo de una regla, mientras que un *predicado BDE* sólo puede aparecer en el cuerpo de las regla. Los *predicados BDI* también pueden ser identificados con relaciones, llamadas *relaciones BDI*. Las *relaciones BDI* no son almacenadas explícitamente, están más bien relacionadas con el concepto de *vistas* de las relaciones, y se obtienen con el compilador o intérprete de Datalog.

Un ejemplo de programa Datalog es el siguiente, donde se representan los caminos entre los distintos nodos de un grafo:

$$r_1 : camino(X, Y) :- arco(X, Y).$$

$$r_2 : camino(X, Y) :- arco(X, Z), camino(Z, Y).$$

Para el ejemplo, *camino(X, Y)* es un *predicado BDI*, mientras que *arco(X, Y)* es un *predicado BDE*. Un ejemplo de *relación BDI* es obtener una parte de la información, como el conjunto de nodos alcanzables desde el nodo A, donde la consulta sería: $? - camino(A, X)$.

Inconvenientes de las Bases de Datos Deductivas

Entre los inconvenientes de las Bases de Datos Deductivas se encuentran:

- Es una forma diferente de representar entidades y relaciones a como se realiza en las bases de datos relacionales, ya que los predicados BDI no corresponden con una relación tal como se conoce en las bases de datos relacionales, no siendo una transformación directa. Tampoco tienen operaciones de agregados (MIN, MAX, AVG...) ni de agrupación (GROUP BY) como tienen las bases de datos relacionales.
- La cabecera se deriva del cuerpo, pero no al contrario. Esto provoca que existan casos que necesitarán una doble inclusión de reglas para representar esta bidireccionalidad.
- Los cambios estructurales son muy costosos ya que no existe posibilidad de definir relaciones mediante referencias entre distintos predicados. Si por ejemplo la signatura de uno de los hechos cambia, todas las reglas que lo contienen deben ser modificadas.
- No están reflejados ninguno de los aspectos de integridad referencial para evitar inconsistencia en la información, como para el ejemplo sería definir arcos sobre vértices que no existen. Esto es debido a que es necesario garantizar que cuando los hechos son actualizados no violen las reglas de integridad [116][109]. El objetivo es mantener la consistencia de la Base de Datos Deductiva, lo que significa que satisface un conjunto de restricciones de integridad.
- Las implementaciones que hay de Bases de Datos Deductivas son poco eficientes para problemas grandes, con muchas reglas y hechos. Pese a que la información puede ser parcialmente almacenada en bases de datos relacionales, la evaluación de consultas es computacionalmente costosa ya que parte de la información tiene que ser inferida. Junto a la posibilidad de que exista una recursividad infinita, por ejemplo si se añade un ciclo al grafo.

2.3.2. Bases de Datos de Restricciones

Las Bases de Datos de Restricciones (BDdR) son el ámbito de trabajo y desarrollo más importante en esta tesis, por lo que serán analizadas durante toda la memoria, siendo en el capítulo 3 donde se estudien en profundidad, en esta sección sólo se hace una breve introducción.

Las Bases de Datos de Restricciones (CDBs, Constraint Databases) permiten representar la información mediante restricciones, donde una restricción es una combinación

de ecuaciones e inecuaciones con operadores lógicos. Las restricciones permiten describir las relaciones entre los valores que pueden tomar un conjunto de variables definidas en un dominio. Una base de datos tradicional se puede representar mediante restricciones, ya que no sería más que un conjunto de restricciones de igualdad o desigualdad de los atributos de dicha tupla. Esto hace que las BDdR sean una generalización del modelo relacional, donde las relaciones entre los atributos no tienen que ser exclusivamente de igualdad o desigualdad con un valor, sino que se pueden definir relaciones más complejas entre dichos atributos. Dichas restricciones permiten almacenar de manera simbólica los distintos valores que pueden tomar las variables en función de otras, lo que significa que esta información será almacenada de manera *intensiva*. Para la inferencia de nueva información, las Bases de Datos de Restricciones se basan en la Programación con Restricciones (CP, Constraint Programming) [42][118].

2.3.3. Bases de Datos Deductivas de Restricciones

Estas bases de datos combinan las dos anteriores, las Bases de Datos Deductivas que soportan la Programación Lógica y las Bases de Datos de Restricciones que soportan la Programación con Restricciones. La Programación Lógica y la Programación con Restricciones dan origen a las Programación Lógica con Restricciones (CLP, Constraint Logic Programming) [81][82]. Las Bases de Datos Deductivas de Restricciones (CDDB, Constraint Deductive Databases) permiten describir dominios infinitos de valores mediante restricciones junto a la capacidad de inferencia lógica de las Bases de Datos Deductivas. Esto hace posible que se añadan restricciones en programas lógicos en el cuerpo de las cláusulas, por ejemplo:

$A(X,Y) :- X + Y > 0, B(X), C(Y).$

donde $X + Y > 0$ es una restricción y $A(X,Y)$, $B(X)$ y $C(Y)$ son predicados.

La evaluación de las consultas en las CDDB [73] se realiza de forma similar a como se hace en las Bases de Datos Deductivas con la programación lógica, donde la consulta es evaluada en función de un objetivo. Para cumplir dicho objetivo y que su cuerpo sea satisfactible, se buscan todas las combinaciones de forma recursiva y utilizando backtracking. Las restricciones que se encuentren en esta búsqueda se irán añadiendo a un *conjunto de restricciones almacenadas*. Cada vez que una nueva restricción se añade a dicho conjunto, se analiza si dicho conjunto continúa siendo satisfactible, siguiendo con la búsqueda o

deshaciendo la decisión en función del resultado. Aunque en la práctica, la satisfactibilidad de las restricciones almacenadas a veces se analiza con algoritmos incompletos que no siempre detectan las inconsistencias [137].

Esta combinación de hechos y restricciones ha hecho necesario la extensión del álgebra relacional para estos tipos de datos [147][1] y la ampliación de los lenguajes de consultas [114].

2.4. Modelo lógico de almacenamiento de Bases de Datos

Los primeros almacenamientos de datos se realizaban sobre ficheros, tras estos sistemas basados en archivos llegaron los sistemas gestores de bases de datos y con ellos la primera generación de bases de datos basada en estructuras jerárquicas. Fue en 1967 cuando se creó el grupo de trabajo llamado DBTG (Data Base Task Group), el cual generó una serie de informes en busca de la estandarización de un entorno que permitiera la creación de bases de datos y la manipulación de la información. En 1970 Codd [32] elaboró uno de los artículos más influyentes en el área de las bases de datos, dando origen a las bases de datos relacionales y con ellas a la segunda generación de las bases de datos. Pese a sus ventajas, el modelo relacional tienen grandes inconvenientes entre los que destacan su limitada capacidad de modelado y la falta de expresividad a la hora de tratar con datos más complejos. Ante la creciente complejidad de los datos y las aplicaciones de las bases de datos, se originaron sistemas con mayor capacidad de expresión, definiéndose las bases de datos orientadas a objetos y más tarde las bases de datos objeto-relacionales, y con ellas a la tercera generación. En esta sección se hace un recorrido por estas propuestas.

2.4.1. Bases de Datos Relacionales

Sin duda alguna, la propuesta más revolucionaria que dio paso a las bases de datos relacionales fue el modelo propuesto por Codd [32], donde todos los datos están estructurados desde el punto de vista lógico mediante relaciones (tablas). Cada relación tiene un nombre que la representa de forma única y está compuesta de forma vertical por atributos (columnas) cuyo número determina la cardinalidad o grado de la relación. A

su vez está compuesta de manera horizontal por tuplas (filas), que contienen un valor para cada atributo. Pese a que la estructura lógica del modelo relacional es simple, su estructura teórica es muy sólida. Esta base teórica permite tratar con la semántica de los datos y con los problemas de coherencia y redundancia, junto a un conjunto de reglas de normalización. El modelo relacional propone la independencia entre los datos junto a la manera de almacenarlos, y los programas que los usan. A partir de esa idea se desarrolló el lenguaje de consulta SQL (Structured Query Language), que ha llegado a ser un estándar de facto para los sistemas relacionales. El lenguaje SQL es un lenguaje declarativo que permite definir datos (DDL - Data Definition Language) y manipularlos (DML - Data Manipulation Language). Desde su creación en 1986, sufrió diversas ampliaciones hasta llegar a la más utilizada hasta el momento, *SQL*:92 (SQL2), definida sobre bases de datos relacionales.

El término de *relación* R está definido basándose en su significado matemático. Siendo los conjuntos S_1, S_2, ..., S_n no necesariamente distintos, R es un subconjunto del producto cartesiano $S_1 \times S_2 \times \ldots \times S_n$, también llamado conjunto de *n-tuplas*. De esta forma se define una base de datos relacional como una colección de relaciones siguiendo una forma normalizada [47] en la que cada relación tiene un nombre distinto. Resumiendo, una tabla que represente una relación de grado n tiene las siguientes propiedades:

- Cada fila representa un tupla de grado n.
- El orden de las filas no es relevante.
- Todas las tuplas (filas) son distintas.
- El orden de las columnas es importante, y debe corresponder a un dominio determinado. Cada columna va asociada a una etiqueta y a un determinado dominio.

Aunque sin duda los sistemas gestores de bases de datos relacionales marcaron el comienzo de las bases de datos en general, tienen algunas deficiencias que se analizan a continuación:

- **Sobrecarga semántica:** El modelo relacional tiene una sola estructura para representar las entidades y las relaciones entre ellas, las tablas. No hay mecanismos para distinguir entre entidades y relaciones, o entre distintos tipos de relaciones, por lo que se dice que está **semánticamente sobrecargado** en ese aspecto.

- **Estructura de datos homogénea:** El modelo relacional define la homogeneidad de datos tanto horizontal como vertical. La homogeneidad horizontal significa que cada tupla de una relación debe estar compuesta por los mismos tipos de atributos. La homogeneidad vertical significa que los valores de una columna concreta de una relación deben provenir todos ellos del mismo dominio, además la intersección de una fila y de una columna debe ser un valor atómico. Esta estructura fija es demasiado restrictiva para muchos objetos del 'mundo real', que tienen una estructura compleja, e incluso recursivas, lo que condiciona combinaciones poco naturales que son muy ineficientes. Uno de los ejemplos clásicos de esta problemática es la planificación del ensamblaje de componentes [4], donde un componente está a su vez formado por más, que a su vez también pueden estar constituido por otros componentes. Este caso representa la necesidad de almacenar una estructura recursiva de tamaño indeterminado, no manteniendo la homogeneidad de datos.

- **Capacidad de definir nuevos tipos:** Los sistemas gestores de bases de datos exclusivamente relacionales no permiten la definición de nuevos tipos que se adapten a las necesidades de los usuarios y las aplicaciones. Muchos sistemas gestores de bases de datos relacionales (SGBDR) actuales han ampliado los tipos de datos que pueden utilizar, como por ejemplo los objetos binarios de gran tamaño (BLOB, Binary Large Object). Un BLOB es un dato que contiene información binaria relativa a una imagen, una secuencia de vídeo o de audio digitalizada, un procedimiento o algún tipo de objeto de gran tamaño no estructurado, de forma que el sistema no tienen ningún conocimiento sobre el contenido o la estructura interna. Esto impide realizar consultas y operaciones sobre tipos de datos que son inherentemente ricos y estructurados, por lo que no la gestionan, sino que simplemente contiene una referencia a la información.

- **Gestión de las consultas recursivas:** Como consecuencia, al no tener incluido los datos de carácter recursivo dentro del modelo relacional, aparece la falta de capacidad de las consultas recursivas de forma natural. Volviendo al ejemplo del ensamblaje de componentes, si una base de datos almacena la relación entre un componente y los que lo componen de forma recursiva hasta llegar a una pieza indivisible, ¿cómo es posible conocer todas las piezas relacionadas con una determinada a cualquier nivel, si a priori no se conoce el número de niveles que hay?

- **Operaciones limitadas:** El modelo relacional sólo tiene un conjunto fijo de operaciones, como son las operaciones de conjuntos y las orientadas a tuplas, no pudiendo especificar nuevas operaciones. Este problema se hace especialmente patente en los sistemas geográficos, donde la información está representada mediante puntos, líneas y polígonos, y se necesitan definir operaciones para conocer la distancia, la intersección, o el hecho de que un objeto está contenido dentro de otro.

- **No correspondencia de impedancias:** El lenguaje estándar para el tratamiento de bases de datos relacionales (SQL) carece de completitud computacional, ya que no tiene definido instrucciones de control de flujo como IF, WHILE, DO,... Aunque SQL sea un lenguaje declarativo, es posible incorporarlo en lenguajes procedimentales de más alto nivel como *Java*^{TM} o *C*. Sin embargo, esta técnica produce una *no correspondencia de impedancias* por estar mezclando diferentes paradigmas de programación, el procedimental y el declarativo.

2.4.2. Bases de Datos Orientadas a Objetos

El paradigma de la orientación a objetos apareció ante las necesidades de representar aspectos a un nivel de abstracción mayor. Un objeto contiene tanto la estructura de los datos como el conjunto de operaciones que pueden usarse para manipularlo, apareciendo el concepto de encapsulación. También es importante el concepto de ocultación de la información, donde se separa los aspectos externos de un objeto de sus detalles internos. De esta forma, las bases de datos se adaptaron a los lenguajes de programación orientados a objetos (OOPL, Object-Oriented Programming Languages) para poder hacer dichos objetos persistentes.

Entre las soluciones propuestas para el almacenamiento de objetos en bases de datos relacionales, existe la posibilidad de establecer una correspondencia entre las instancias de las clases (los objetos) y una o más tuplas distribuidas entre una o más relaciones. Hay diversas estrategias para asignar las clases a relaciones, aunque todas ellas tienen como resultado una pérdida de información semántica ya que hay que definir el código para descomponer los objetos en tuplas y el código para leer las tuplas de las relaciones y reconstruir los objetos.

Distintas definiciones han aparecido para describir qué es un Sistema Gestor de Base de Datos Orientadas a Objetos (SGBDOO) [3]. Por ejemplo en [156] donde se define que

un SGBDOO debe proporcionar funcionalidad de bases de datos, soportar el concepto de identidad de los objetos, establecer un mecanismo de encapsulación, y soportar objetos con estados complejos. Aunque otros trabajos [86] lo definen como orientación a objetos (tipos abstracto de datos, herencia e identidad de objetos) junto a las capacidades de las bases de datos. Pero estas definiciones dejan fuera algunas de las necesidades más importantes en las bases de datos, que más tarde fueron recogidas [120], como son la definición de un lenguaje de consulta de alto nivel con capacidad de optimización de las consultas, control de concurrencia, soporte de objetos complejos, e índices para hacer los accesos más eficientes. Otra de las características obligatorias que más tarde se añadieron a los SGBDOO es la capacidad de definir nuevos tipos por el usuario, partiendo del conjunto de tipos predefinidos por el sistema. Además, no debe haber ninguna distinción de uso entre los tipos definidos por el sistema y los definidos por los usuarios.

Entre algunas de las características específicas que deben cumplir los SGBDOO se encuentran:

- **Soporte de objetos complejos:** Deben ser capaces de ofrecer formas de construir objetos complejos basándose en otros simples, definiendo para ellos sus constructores. Entre estos objetos complejos se encuentran SET, TUPLE o LIST, con la consecuente dificultad que conlleva la integridad referencial entre ellos [117].
- **Capacidad de identificación de objetos:** Diferenciando entre el concepto de igualdad de tupla con respecto al contenido, que tiene el álgebra relacional clásica, y la igualdad entre objetos en función de su identificador (OID), que es independiente del contenido de sus atributos.
- **Polimorfismo y sobrecarga:** Pudiendo aplicarse el mismo método sobre objetos de diferente tipo (sobrecarga), y dando la posibilidad de definir distintos métodos con el mismo nombre en función de sus parámetros de entrada (polimorfismo).
- **Extensibilidad:** Dando la posibilidad al usuario de crear nuevos tipos a partir de otros ya definidos. Esos nuevos tipos no serán diferentes en uso a los definidos por el sistema.
- **El Lenguaje de Manipulación de Datos debe ser computacionalmente completo:** Por lo que dicho lenguaje debe ser de propósito general, idea que no se llevó al estándar hasta la aparición de SQL:1999, que sí es computacionalmente

completo al poder combinarse con otros lenguajes de programación. En *SQL*:1999 (SQL3) se incluyeron además algunas características como la ampliación de las expresiones en las condiciones de la consultas, conceptos de bases de datos activas con la definición de triggers, consultas recursivas, junto a las características propias de la orientación a objetos.

Estándar de objetos de datos ODMG

Existen muchas implementaciones de SGBDOO entre las que se pueden destacar *Poet*, *Jasmine*, *ObjectStore*, *Versant* o *GemStone*. Dichas soluciones se basan en el estándar del modelo de datos orientado a objetos (OODM, Object-Oriented Data Model), el cual está compuesto de un modelo de objetos, un lenguaje de definición de objetos y un lenguaje de consulta sobre objetos con una sintaxis similar a la de SQL. Dicho estándar fue definido por ODMG (Object Data Management Group), el cual propone la definición de la semántica de los objetos en las bases de datos. Desde las primeras publicaciones y definición del primer estándar al respecto en 1993 [26], se ha ido refinado hasta la actual versión 3.0 [13]. Dicho estándar sirve aún como referencia en publicaciones recientes [96] con el propósito de soportar todas las funcionalidades existentes en las bases de datos relacionales y el modelo de datos orientado a objetos. Los principales componentes del estándar son:

- El modelo de objetos (OM, Object Model)
- El lenguaje de definición de objetos (ODL, Object Definition Language)
- El lenguaje de consulta de objetos (OQL, Object Query Language)
- Enlaces con lenguajes de programación orientados a objetos como C^{++} y $Java^{TM}$.

El modelo de objetos

El modelo de objetos ODMG (Object Data Management System) soporta el concepto de bases de datos como un área de almacenamiento de objetos persistentes sobre un conjunto de tipos determinados, donde una fila de una relación puede ser una instancia de un objeto. El modelo de datos permite almacenar objetos, haciendo posible que múltiples usuarios y aplicaciones los compartan. Se define como un superconjunto del modelo de objetos que especifica las siguientes primitivas básicas de modelado:

- Objetos y literales (objetos inmutables que no cambian en el tiempo, como Integer, Float, Boolean o Character), donde sólo los objetos tienen identificadores unívocos y los literales se comparan por su contenido. A su vez, los objetos y literales pueden clasificarse en diferentes tipos (atómicos, colecciones y tipos estructurados).
- El comportamiento se define mediante un conjunto de operaciones que pueden realizarse sobre o por el objeto.
- El estado o vida del objeto se define mediante los valores que el objeto tiene para una serie de propiedades, que pueden ser atributos o una relación entre varios objetos. Los literales no tienen estado.

El lenguaje de definición de objetos

El lenguaje de definición de objetos (ODL, Objet Definition Language) es un lenguaje creado para definir las especificaciones de los tipos de objetos en los sistemas compatibles con ODMG, de una manera equivalente al lenguaje de definición de datos de los SGBDR tradicionales. El ODL define los atributos y relaciones de los tipos y especifica la signatura de las operaciones, aunque no su implementación. La sintaxis de ODL es una ampliación del lenguaje de definición de interfaces (IDL - Interface Definition Language) de CORBA.

El lenguaje de consulta de objetos

OQL (Object Query Language) es el lenguaje de consulta definido por ODMG, el cual ofrece un acceso declarativo a las bases de datos de objetos con una notación similar a *SQL*-92 para expresar consultas. Al igual que SQL, OQL se puede utilizar como lenguaje autónomo y como lenguaje embebido dentro de otros lenguajes procedimentales. Dicho lenguaje extiende nociones de la orientación a objetos como objetos complejos, identidad de objetos, expresiones de camino con la notación **'.'** o **'→'**, polimorfismo e invocación de operaciones, e incluso proporcionando primitivas de alto nivel para tratar con conjuntos de objetos. También proporciona primitivas para tratar con estructuras, listas y arrays gestionando estos constructores con la misma eficiencia. El resultado de una consulta cualquiera, es de un tipo que pertenece al modelo ODMG y puede consultarse de nuevo, por lo que cumple la propiedad del cierre transitivo. Es un lenguaje de consulta sencillo que permite acceder de forma fácil a un SGBDOO, aunque no es computacionalmente

completo, ya que no proporciona operadores explícitos de actualización; más bien invoca operaciones definidas sobre los objetos para este propósito, y así no infringe la semántica del modelo, en la cual por definición los objetos se manejan mediante los métodos definidos sobre ellos.

Sin duda alguna, los SGBDOO tienen tanto ventajas como inconvenientes. Entre sus ventajas se pueden destacar:

- Capacidad de ampliación, ya que permite que el usuario defina nuevos tipos que se adapten a sus necesidades. A esto hay que añadirle las características propias de la orientación a objetos como la herencia, lo que facilita la reusabilidad de los tipos definidos. La orientación a objetos permite encapsular información y comportamiento, por lo que tiene mayor capacidad de modelado. Esta capacidad facilita su adaptación a aplicaciones complejas, como las que se han presentado en la sección anterior.
- Existencia de un lenguaje de consulta más expresivo, gracias a la navegabilidad existente entre objetos. Esta característica ayuda a paliar la deficiencia en los SGBDR relativa a consultas recursivas.
- Eliminación de la no correspondencia de impedancias de los SGBDR, eliminando muchas de las deficiencias de unir los lenguajes declarativos como SQL y los imperativos como Java^{TM}.

Entre sus inconvenientes se encuentran:

- Carencia de un modelo de datos y de un lenguaje de consulta orientado a objetos con base teórica aceptada de forma genérica por la comunidad de SGBDOO, como sí tienen los SGBDR con SQL. Aunque ODMG ha hecho una importante aportación en este aspecto, sin duda no tiene el grado de aceptación del modelo relacional.
- La relación entre optimización de consultas y la encapsulación, ya que dicha encapsulación de objetos provoca dificultades en la evaluación de las consultas.
- Complejidad, ya que el tratamiento de referencias entre objetos, las transacciones de larga duración y la gestión de versiones, son más complejas que en el caso de las bases de datos relacionales. Una de las características de esta complejidad, radica en

la necesidad de serializar los objetos para que sean almacenados de una manera persistente. Una metáfora clara que representa esta complejidad podría ser establecer el símil con un coche que para ser guardado en un garaje tiene que ser desmontado, y cuando se vuelve a necesitar tiene que ser montado de nuevo. Estas acciones de 'desmontado' y 'montado' son las necesarias en los objetos para que se almacenen y hagan persistentes en los soportes físicos.

- Que el usuario pueda definir nuevos datos es sin duda alguna un salto cuantitativo, pero en el caso de bases de datos remotas, como es el caso de las existentes para aplicaciones web, significará que la funcionalidad no pertenece a la base de datos, sino al usuario.

2.4.3. Bases de Datos Objeto-Relacionales

Las bases de datos objeto-relacionales son una ampliación del modelo relacional para dar soporte a las nuevas aplicaciones en bases de datos. Este tipo de bases de datos usan el estándar *SQL*:2003, el cual incluye nuevos tipos y procedimientos definidos por el usuario, el tipo referencia, y soporta objetos de gran tamaño pero siempre basándose en el modelo relacional. Las bases de datos relacionales no son tan descriptivas como las orientadas a objetos para poder soportar las aplicaciones que están surgiendo en la actualidad, pero por otra parte tiene grandes ventajas. La idea que originó las bases de datos objetos-relacionales fue fusionar las ventajas de ambas e intentar reducir sus deficiencias, manteniendo las mismas tablas relacionales básicas y el mismo lenguaje de consulta aunque incorporando el concepto del tipo 'objeto', dando como resultado los Sistemas Gestores de Bases de Datos Objeto-Relacionales (SGBDOR).

Las bases de datos objeto-relacionales fueron analizadas por Stonebraker [143], proponiendo la división mostrada en la figura 2.2. En esta figura se muestra cómo la orientación a objetos permite ampliar la complejidad de los datos, mientras que la búsqueda de soluciones relacionales mejora la eficiencia en la evaluación de consultas. Entre las características de las bases de datos objeto-relacionales, se puede destacar que el acceso a la base de datos se puede hacer usando lenguajes de más alto nivel orientados a objetos. Pese a añadir la programación orientada a objetos, la información es relacional, y las tuplas y las relaciones mantienen el mismo significado que en las bases de datos relacionales, aprovechando su desarrollo y madurez.

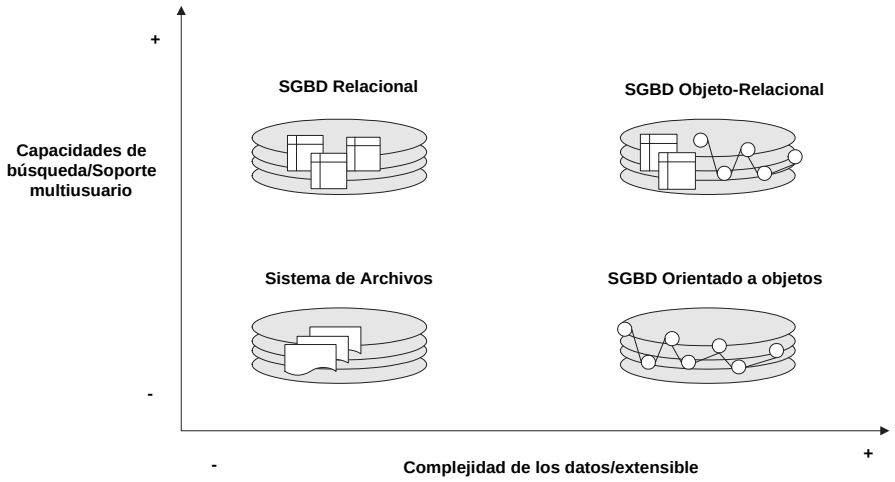

Figura 2.2: El mundo de las bases de datos según Stonebraker [1996]

Entre las ventajas de los SGBDOR se puede destacar:

- Reutilización y compartición, ya que surge la capacidad de ampliar los servicios del SGBD para implementar funcionalidad estándar de manera central, en lugar de codificar dicha funcionalidad en cada aplicación. Esto es muy utilizado en los sistemas geográficos, que necesitan una serie de funciones asociadas que calculen la distancia entre objetos, operaciones de inclusión o solapamientos. De esta forma no se tiene que definir la funcionalidad en cada aplicación sino sólo en el servidor, aumentando la productividad tanto del desarrollador como del usuario final.

- La adaptación de las aplicaciones relacionales a las objeto-relacionales, ya que los SGBDOR pueden introducirse de forma gradual. SQL:2003 es compatible con SQL:92, por lo que la transformación de un sistema ya existente se puede realizar por pasos.

- Los SGBDOR tienen la capacidad de almacenar instancias de clases y atributos clásicos relacionales, eso ofrece la posibilidad de escoger en función de las necesidades de la aplicación, cuál es la implementación más adecuada. En los SGBDOO toda la información es almacenada como objetos, por lo que se gana capacidad expresiva pero con deficiencias en la optimización de la evaluación de consultas. Por otro lado, en los SGBDR todos los datos son univaluados, sus dominios son atómicos, con un estructura fija y única pero encontrar las relaciones entre los datos es más fácil y eficiente por su simplicidad. Vistas estas características, sería más apropiado utilizar

en función de las necesidades un modelo u otro de representación, seleccionando en cada caso el necesario. De esta forma, ambos modelos podrían convivir en el mismo gestor de base de datos.

Aunque obviamente tiene algunas desventajas:

- La complejidad y el incremento de coste, ya que se pierde la simplicidad del modelo relacional.
- Los SGBDOR no añaden todos los valores de la orientación a objetos, ya que tan solo contemplan el modelo de objetos como una extensión del modelo relacional al que se le añade complejidad.
- La 'convivencia' de datos puramente relacionales y objetos hace que se pierda la homogeneidad de acceso a la información. Esta característica hace que el tratamiento de la información dependa de su naturaleza, y por lo tanto aumente la complejidad del desarrollo de los SGBDOR.

2.4.4. Comparación entre las propuestas

Aunque en las subsecciones anteriores se muestran muchas de las características de los distintos gestores de bases de datos, en las tablas 2.1 y 2.2 analizadas en [45] se muestra un resumen de algunas de sus características. En la tabla 2.1 se presentan las características más genéricas relativas a la facilidad de uso, de programación, junto a características de los lenguajes de consulta y la naturaleza de los datos que soportan.

El objetivo de muchas multinacionales es desarrollar nuevos sistemas gestores de bases de datos que lideren el mercado, ya que la selección de un buen gestor de bases de datos es una decisión muy importante para una empresa. En la tabla 2.2 se muestran algunas consideraciones al respecto de los productos que existen en el mercado y su madurez.

Comparación	SGBDR	SGBDOO	SGBDOR
Estándares	SQL2	ODMG-2.0	SQL:2003
Facilidad de uso	Fácil	Fácil para los programadores, no tanto para el usuario final	Similar a los SGBDR, aunque con algunas extensiones confusas
Facilidad de desarrollo	Independencia entre datos y aplicación	Los objetos son una forma natural de modelar, extensible y adaptable	Independencia entre datos y aplicación
El uso de SQL	Completo	Extensión de SQL con la creación de OQL adaptado a la orient. a objetos	Definición de SQL3 basado en SQL pero con orient. a objetos
Datos complejos y relaciones	Difícil de modelar	Facilita la definición de tipo complejos y relaciones	Facilita la definición de tipo complejos y relaciones
Soporte a la Orientación a Objetos	Pobre, es difícil almacenar los objetos en la BD	Directa y Extensible	Limitado sobre todo para los nuevos tipos de datos
Distribución	Muy buena	Depende del SGBDOO	Muy Buena

Tabla 2.1: Una comparación entre Sistemas Gestores de Bases de Datos (I)

2.5. Posición de la propuesta dentro del organigrama general

Tras analizar las tres características que definen una base de datos (dominio de la aplicación, naturaleza de la información y modelo lógico), en la figura 2.3 se resumen gráficamente. La combinación de estas tres características describirá una base de datos.

Se pueden encontrar infinidad de casos donde estas características están combinadas, presentados en la figura 2.4, como por ejemplo:

- No sólo son los ejemplos de aplicaciones administrativas los que almacenan información de manera extensiva, también son muy utilizadas en el área de investigación de *Data Mining* [74][136] utilizando un SGBDR.

Comparación	SGBDR	SGBDOO	SGBDOR
Madurez del producto	Mucha	Relativamente	Aún está siendo definida
Soporte comercial	Gran éxito	Tiene buen futuro	Falta el apoyo empresarial

Tabla 2.2: Una comparación entre Sistemas Gestores de Bases de Datos (II)

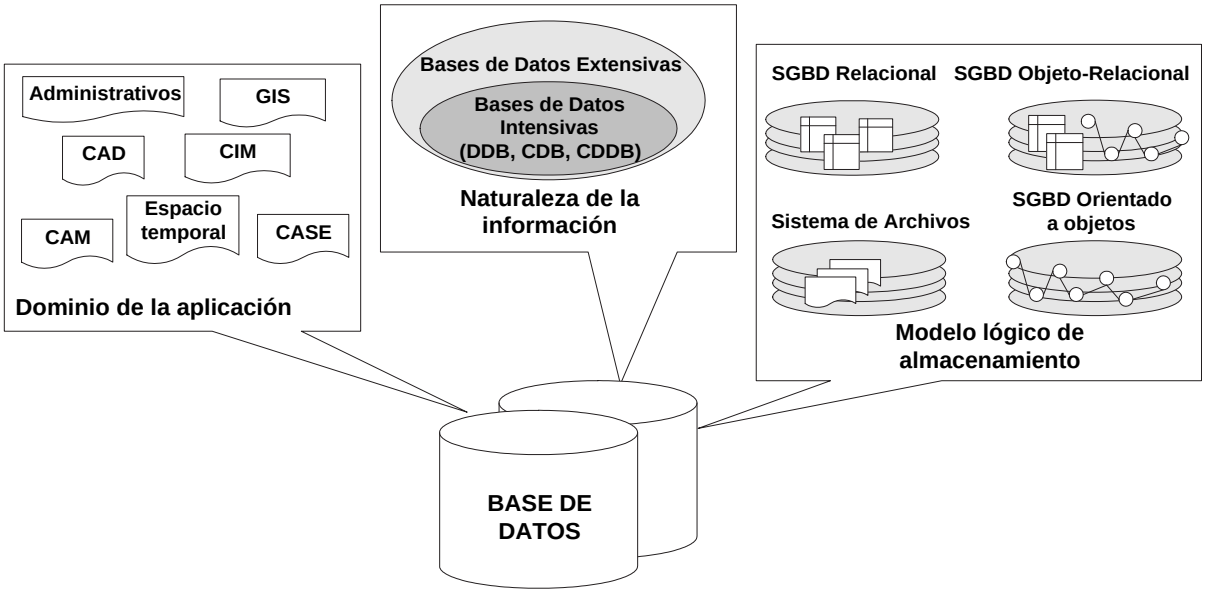

Figura 2.3: División detallada de una base de datos por características

- En el trabajo [140] se presenta una combinación donde el dominio de la aplicación es el tratamiento de imágenes, la información es almacenada de manera intensiva y el modelo lógico de almacenamiento un SGBDR. Este ejemplo es relevante ya que pese a utilizar el modelo relacional, diseñado originalmente para información extensiva, se puede inferir información no almacenada de manera explícita.

- Algunas soluciones para el dominio de los sistemas de información geográfica han utilizado las Bases de Datos Deductivas de Restricciones [1], basándose en ficheros Datalog.

- Las Bases de Datos Deductivas (intensivas) y los SGBDOO también han sido analizados [104], con ejemplos de aplicaciones como la planificación de horarios [25].

- El sistema MLPQ [133] define una solución para sistemas de información geográfica utilizando Bases de Datos de Restricciones, cuya información será intensiva pero almacenada en un sistema de archivos.
- El sistema CCUBE [17] presenta una solución a sistemas espacio-temporales utilizando Bases de Datos de Restricciones cuyo soporte lógico son los SGBDOO.
- Ante las emergentes aplicaciones relacionadas con Internet, nuevos tipos de datos han tenido que ser tratados, como es el caso de XML. Esto ha generado nuevas necesidades como las analizadas en [92] y [119], donde se almacenan documentos XML de manera extensiva utilizando SGBDOR.

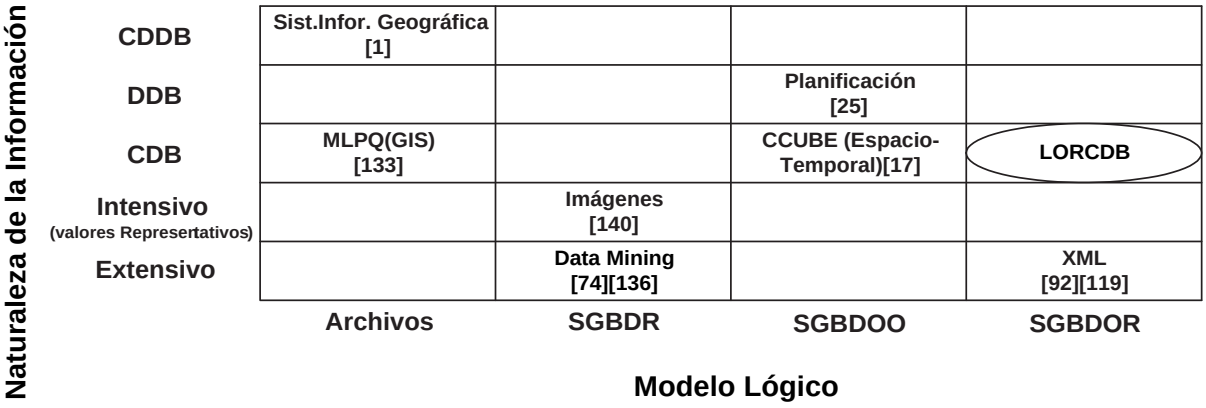

Figura 2.4: Ejemplos de propuestas combinando características de las Bases de Datos

Estos ejemplos muestran la independencia de estas tres características que se han definido, aunque sin duda hay combinaciones que son de mayor uso que otras. Como se ha presentado, en las bases de datos se pueden destacar dos necesidades básicas: la búsqueda de nuevas tecnologías para gestionar tipos más complejos de datos; y la necesidad de mejorar la evaluación de las consultas, ya que la cantidad de datos a tratar es cada vez mayor y cualquier propuesta tiene que tener en cuenta los aspectos de eficiencia. Para cubrir estas necesidades, los aspectos analizados en esta tesis para las distintas características de las base de datos son:

- **A nivel de dominio de la aplicación:** Se propone una nueva forma de tratar datos complejos, donde la información se representa de manera intensiva y extensiva independientemente de la aplicación. Esta solución pretende ser genérica, no

dependiente del dominio de la aplicación, aunque se utilizará la diagnosis de fallos basada en modelos como un tipo de CAD/CAM y los sistemas de información geográfica como ejemplos por su importancia en el mercado.

- **A nivel de naturaleza de la información:** Se utilizan las Bases de Datos de Restricciones para representar los datos de manera intensiva y extensiva. Aunque las Bases de Datos de Restricciones necesitan ser revisadas para aprovechar más sus cualidades. La utilización de restricciones es especialmente apropiado para el tratamiento de CAD/CAM [146], ya que son problemas formados por un conjunto finito de objetos geométricos junto a un grupo de restricciones entre ellos [99]. Estas restricciones se pueden describir mediante ecuaciones donde las variables son las diferentes coordenadas de los objetos.

- **A nivel de modelo lógico:** Se utilizan los Sistemas Gestores de Bases de Datos Objeto-Relacionales, añadiéndole funcionalidad a nivel de evaluación de consultas para el tratamiento de datos. El uso de SGBDOR, junto a un conjunto de algoritmos que harán posible optimizar los tiempos necesario para la evaluación de las consultas. Se utiliza dicho modelo lógico porque da la capacidad de representar datos complejos mediante objetos para información intensiva, mientras que mantiene algunas características de las bases de datos relacional que ayudan a trabajar con datos extensivos.

Todo esto da origen al sistema LORCDB (Labelled Objet-Relational Constraint Databases), que será descrito a lo largo de esta memoria de tesis, y cuyas características también se presentan en la figura 2.4.

2.6. Resumen

Para describir una base de datos es necesario contemplar tres características: El dominio de las aplicaciones que utilizan y gestionan las bases de datos, la naturaleza de la información a almacenar, y el modelo lógico de almacenamiento. En este capítulo se han presentado los detalles de cada una de estas familias, y cómo sus distintas combinaciones dan origen a las distintas bases de datos. Ante la creciente complejidad de los dominios de las aplicaciones, las demás características han tenido que ir avanzando de forma análoga para darles soporte. Los ejemplos de aplicaciones mostradas en este capítulo, entre

otras, necesitan buscar soluciones específicas a sus problemáticas, no pudiendo utilizar soluciones genéricas para almacenar sus datos. Esto significa que cada aplicación, debido a tener datos muy complejos, tendrá una base de datos con una lógica específica. Las propuestas de sistemas gestores de bases de datos, han ido acercando su funcionalidad a estas necesidades, dando la posibilidad al usuario a ampliar los tipos de datos y sus operaciones, teniendo a veces que apoyarse en representaciones intensivas de la información. Esto significa que cada problema particular, que no puede ser representado de forma exclusivamente extensiva, tendrá que desarrollar una solución adaptada a sus necesidades. El objetivo de esta tesis es definir y desarrollar una solución genérica para estos datos, de forma que se pudiera utilizar el mismo gestor de base de datos, pudiendo dale distinta funcionalidad con menor coste individual en cada aplicación. Para ello se utilizarán los SGBDOR, junto a funcionalidad basada en la programación con restricciones. De esta forma, todas las aplicaciones cuyos datos puedan ser descritos con ciertas restricciones, detalladas en los sucesivos capítulos, no tendrán que diseñar una funcionalidad específica independiente del gestor de bases de datos, dicha funcionalidad estará incluida en el gestor LORCDB.

Capítulo 3

Antecedentes de las Bases de Datos de Restricciones

Las Bases de Datos de Restricciones (DBdR) son una estrategia importante en el tratamiento de información intensiva. La búsqueda de cómo almacenar y gestionar la información intensiva, representando los datos en forma de restricción, ha sido el impulsor de la creación y el desarrollo de las BDdR, con el objetivo de buscar soluciones cada vez más expresivas y eficientes. Para su estudio exhaustivo se han creado varias definiciones, junto a la formalización de términos que ayudan a entender mejor tanto sus posibilidades como sus limitaciones.

Además de la formalización de las BDdR, en este capítulo se hace un recorrido desde sus orígenes hasta las propuestas más actuales. Así mismo, se presentan distintos prototipos, analizando sus aportaciones y deficiencias con el fin de realizar una propuesta que aporte soluciones a sus debilidades.

3.1. Introducción

La idea que dio origen al desarrollo de las BDdR surge de la necesidad de una nueva forma de almacenar datos que no toman valores únicos, sino que dependen de parámetros, variables, o vienen definidos por relaciones entre variables mediante ecuaciones o inecuaciones matemáticas. La representación de estos datos de manera extensiva podría crear bases de datos infinitas, por ejemplo al representar un espacio continuo, por lo que hay

que buscar alternativas para almacenar los datos de forma intensiva. Una posibilidad es discretizar los datos, pero eso podría conllevar la creación de bases de datos mucho más grandes a la vez que pérdida de información. El crecimiento de las bases de datos también influye en las operaciones de consulta y modificación, ya que tendrían que ser analizados y modificados muchos más registros que si la información estuviera almacenada de manera más compacta. Sin embargo, si se tuviera la capacidad de representar y almacenar dichas ecuaciones, inecuaciones e incluso combinaciones de ellas, la información sería completa y el espacio necesario para almacenarla muy pequeño debido a la compresión de la información. Las BDdR abordan la solución de este problema tratando la información no de forma discreta, sino como un conjunto de restricciones sobre variables definidas sobre unos determinados dominios.

El reto de las BDdR no acaba en la simple necesidad de almacenar la información de forma intensiva, va más allá, ya que se necesita tratar esa información definiendo un lenguaje de consulta, y una metodología que ofrezca una manera eficiente de realizar distintas operaciones sobre los datos.

3.2. Origen de las Bases de Datos de Restricciones

Las Bases de Datos de Restricciones (CDB, Constraint Databases) tuvieron su origen a principios de 1990 con el artículo de Kanellakis, Kuper y Revesz [84], ampliándose más tarde en los trabajos [93][130]. Originalmente, el objetivo de las BDdR fue definir una versión de bases de datos a la que se le añadió funcionalidad relacionada con la Programación Lógica con Restricciones (Constraint Logic Programming - CLP). La creación de las DBdR fue una evolución natural de la investigación en el campo de la Programación con Restricciones, al igual que las Bases de Datos Deductivas [36] y Datalog [28] fueron el resultado de la combinación de las bases de datos y la Programación Lógica. Considerando las bases de datos relacionales una forma eficiente, compacta y transparente de representación de datos, se buscó combinar las ventajas del modelo relacional y la Programación Lógica con Restricciones. Las BDdR se basaron en la equivalencia entre una tupla en una base de datos relacional, y un conjunto de restricciones de igualdad o desigualdad sobre los atributos de dicha tupla [93]. Debido a que las restricciones son un mecanismo natural de especificar las consultas de similitud en series de datos, muchas de las características del modelo relacional pueden extenderse al campo de la Programación

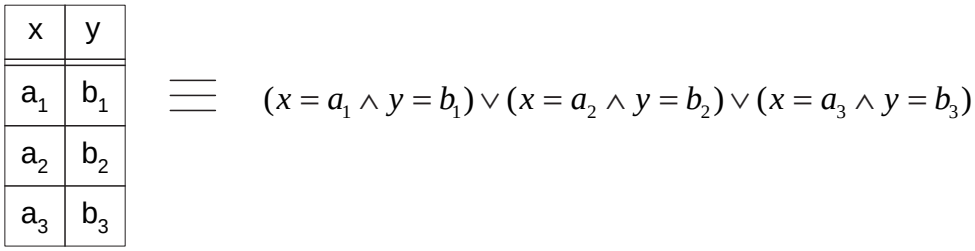

Figura 3.1: Ejemplo de representación de un conjunto de tuplas como una restricción

con Restricciones. Un ejemplo de equivalencia entre un conjunto de tuplas de una relación y una restricción es el presentado en [94], mostrado en la figura 3.1.

En sus inicios, las BDdR se crearon para representar problemas donde se involucran datos espaciales, temporales o ambos a la vez [9], aunque su capacidad descriptiva y de inferencia puede ser extendida a cualquier dominio cuya información se pueda representar mediante restricciones. Al utilizar originalmente este dominio de aplicación, las propuestas se centraron en las restricciones lineales. Las restricciones lineales permiten describir problemas espacio-temporales de una forma aproximada, siendo más fáciles de evaluar las consultas a nivel de coste computacional que utilizando restricciones polinómicas. Sin embargo, las restricciones polinómicas son más genéricas, teniendo mayor capacidad expresiva.

Las BDdR fueron definidas en base a la lógica y modelo de primer orden, tanto para la descripción de restricciones como para la evaluación de las consultas. El fundamento de las BDdR se apoya en la dualidad de representar un conjunto de datos mediante una restricción (fórmula) sobre un conjunto de variables libres $\{x_1, \ldots, x_m\}$, o representarlos como un conjunto de tuplas formadas por atributos $(a_1, \ldots, a_n)$. Esto proporciona gran poder declarativo y expresivo, como se muestra en la figura 3.2, donde se describe la equivalencia entre una restricción y su almacenamiento en una base de datos relacional, en este caso una restricción lineal de igualdad en el dominio de los naturales. Este es un caso, como muchos otros mostrados en el capítulo 1 de [130], donde se generaría una bases de datos relacional infinita si la información no se representara mediante restricciones.

a	b	c
1	1	2
1	2	3
1	3	4
2	1	3
3	1	4
4	1	5
5	1	6
...	...	...

≡ {a+b=c para a, b, c: Natural}

Figura 3.2: Ejemplo de representación de una restricción en una base de datos relacional

3.3. Formalizaciones

Las primeras definiciones relativas a BDdR se apoyaron en la lógica de primer orden, estableciendo un símil entre el modelo lógico y las restricciones. En este apartado se presentan algunos formalismos utilizados en el área de las BDdR.

3.3.1. Lógica y Modelo de Primer Orden

En esta sección se exponen algunas definiciones que introducen los lenguajes de primer orden y su interpretación mediante modelos. La lógica de primer orden se considera un lenguaje de consulta puramente declarativo, que presenta algunas limitaciones como la falta de expresividad en operaciones de conteo, y carencia de definiciones de estructuras recursivas. Estas definiciones son necesarias para entender la teoría básica de modelado de las BDdR [34][82].

Definición 3.1: Vocabulario. Representado por la signatura Ω, está constituido por tres conjuntos: $\mathcal{F}$ que representa los operadores, $\mathcal{C}$ que representa las constantes, y $\mathcal{P}$ que representa los predicados.

Un ejemplo de vocabulario es: $\Omega = (\{+, \cdot\}, \{0, 1\}, \{<\})$ que permitirá definir inecuaciones polinómicas.

Definición 3.2: Términos. La lógica de primer orden se basa en la utilización de términos, que pueden ser *variables*, *constantes* pertenecientes al conjunto $\mathcal{C}$, o una expresión de la forma $f(t_1, \ldots, t_n)$, donde $f \in \mathcal{F}$ y $t_1, \ldots, t_n$ son a su vez términos.

Definición 3.3: Fórmula atómica. Una fórmula atómica t puede venir representada mediante un término t' o por un predicado cuyos parámetros son términos $p(t_1, \ldots, t_n)$.

Definición 3.4: Fórmula. Una fórmula puede ser una *fórmula atómica*, o una combinación lógica de fórmulas atómicas. Por ejemplo, si φ y ψ son fórmulas, entonces $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\neg \varphi)$, $(\varphi \Rightarrow \psi)$ y $(\varphi \Leftrightarrow \psi)$ también son fórmulas.

Antes de introducir la formalización de las BDdR, es necesario formalizar algunas definiciones relativas a restricciones.

Definición 3.5: Estructura. La idea de estructura es añadir un dominio a un vocabulario, lo que implicará añadirlo a las fórmulas atómicas y a las fórmulas genéricas. Para un conjunto de valores ($\mathcal{U}$) que definen el dominio, y un vocabulario Ω, una Ω-estructura $\mathcal{M}$ en $\mathcal{U}$ se define mediante:

- La asignación de cada $f \in \mathcal{F}$ de aridad n a la función $f^{\mathcal{M}} : \mathcal{U}^n \rightarrow \mathcal{U}$, lo que significa que para todos los valores de entrada de f, pertenecientes a $\mathcal{U}$, se obtiene un valor de salida también perteneciente al dominio $\mathcal{U}$.
- Para cada predicado $p \in \mathcal{P}$, los parámetros de p de aridad n pertenecen a $\mathcal{U}$, lo que se expresa de la forma $\mathcal{P}^{\mathcal{M}} \subseteq \mathcal{U}^n$.
- Toda constante $c \in \mathcal{C}$ tiene que pertenecer a $\mathcal{U}$, representado de la forma $c^{\mathcal{M}} \in \mathcal{U}$.

Definición 3.6: Restricción. Sea Ω un vocabulario, una restricción sobre Ω (Ω-restricción) es una fórmula atómica de primer orden sobre Ω o la negación de una fórmula atómica.

El poder de expresivo de las restricciones radica en su capacidad de representar de una forma compacta un conjunto de datos extensivos, a lo que se denomina *Definible con Restricciones (Definability with Constraints)*. También es posible el proceso inverso, *extender una restricción*, lo que significa describir los valores que hacen consistente la restricción.

Definición 3.7: Definible con Restricciones (Extensión de Restricciones). Dado un vocabulario Ω y una estructura $\mathcal{M} = (\mathcal{U}, \Omega)$ donde $\mathcal{U}$ es el dominio, un conjunto $X \subseteq \mathcal{U}^n$ es definible en $\mathcal{M}$ con Ω-restricciones, si dicho conjunto puede ser representado con una fórmula (combinación finita de fórmulas atómicas con operadores booleanos):

$\{(a_1, ..., a_n) \in \mathcal{U}^n \mid \mathcal{M} \models \varphi(a_1, ..., a_n)\}$

Los tipos de restricciones tratados en esta tesis son:

- **Restricciones lineales de igualdad**, las cuales son restricciones sobre $\Omega = (\{+\}, \{0, 1\}, \{=\})$. Cada restricción corresponde con una condición de la forma $p = 0$, donde p es una función lineal de la forma $a_1x_1 + \ldots + a_jx_j + a_0$, a_i representa coeficientes y x_i variables. Estas restricciones son normalmente interpretadas sobre las estructuras $\mathbf{R}_{lin} = (I\!R, \Omega)$, $\mathbf{Q}_{lin} = (\mathbb{Q}, \Omega)$, $\mathbf{Z}_{lin} = (\mathbb{Z}, \Omega)$.

- **Restricciones polinómicas de igualdad**, las cuales son restricciones sobre $\Omega = (\{+, \cdot\}, \{0, 1\}, \{=\})$. Cada restricción corresponde a una condición de la forma$p = 0$, donde p es un polinomio. Estas restricciones son normalmente interpretadas sobre las estructuras $\mathbf{R}_{pol} = (I\!R, \Omega)$, $\mathbf{Q}_{pol} = (\mathbb{Q}, \Omega)$, $\mathbf{Z}_{pol} = (\mathbb{Z}, \Omega)$.

- **Restricciones lineales**, son restricciones sobre $\Omega = (\{+\}, \{0, 1\}, \{<\})$. Cada restricción corresponde con una condición de la forma $p > 0$ y $p \leq 0$, donde p es una función lineal de la forma $a_1x_1 + \ldots + a_jx_j + a_0$, a_i representa coeficientes y x_i variables. Estas restricciones son normalmente interpretadas sobre las estructuras $\mathbf{R}_{lin} = (I\!R, \Omega)$, $\mathbf{Q}_{lin} = (\mathbb{Q}, \Omega)$, $\mathbf{Z}_{lin} = (\mathbb{Z}, \Omega)$.

- **Restricciones polinómicas**, las cuales son restricciones sobre $\Omega = (\{+, \cdot\}, \{0, 1\}, \{<\})$. Cada restricción corresponde a una condición de la forma $p > 0$ o $p \leq 0$, donde p es un polinomio. Estas restricciones son normalmente interpretadas sobre las estructuras $\mathbf{R}_{pol} = (I\!R, \Omega)$, $\mathbf{Q}_{pol} = (\mathbb{Q}, \Omega)$, $\mathbf{Z}_{pol} = (\mathbb{Z}, \Omega)$.

3.3.2. Formalización de las Bases de Datos de Restricciones

La idea fundamental sobre la que se basa la definición de las BDdR es que *una conjunción de restricciones es la correcta generalización de un conjunto de hechos* [84]. Las tuplas y relaciones del modelo relacional pueden ser transformadas en fórmulas de primer orden libres de cuantificadores, lo que permite manipular información de una forma simbólica. Un ejemplo de esto se puede observar en la representación de los rectángulos, que están definidos por sus cuatro vértices. La forma clásica podría llevar a almacenar los datos en una relación de aridad 5, conteniendo la tupla de la forma (n, a, b, c, d), donde n es el nombre del rectángulo, y los puntos a, b, c y d definen las esquinas del rectángulo

$((a, b), (a, d), (c, b), (c, d))$. En un modelo de BDdR, dicho rectángulo se puede modelar mediante una relación ternaria $R(z, x, y)$, que puede ser interpretada como que (x, y) es un punto perteneciente al rectángulo de nombre z. El rectángulo ahora se puede almacenar en una única tupla de la forma:

$$(z = n) \wedge (a \leq x \leq c) \wedge (b \leq y \leq d)$$

El primer modelo de BDdR fue diseñado como una extensión del modelo relacional, donde las tuplas son definidas como conjunciones de fórmulas atómicas, y las relaciones como disyunciones entre tuplas. Estas definiciones fueron presentadas en los trabajos [84] y [93], de donde se pueden resaltar las siguientes definiciones:

Definición 3.8: Modelo de Base de Datos de Restricciones. Dado un vocabulario Ω.

- Una *k-tupla restricción* es una tupla de k variables $x_1, \ldots, x_k$ sobre Ω. De manera que una *k-tupla restricción* será una combinación de restricciones con el operador de conjunción de la forma $\varphi_1 \wedge \ldots \wedge \varphi_N$, donde cada φ_i, para $1 \leq i \leq N$, es una restricción sobre Ω, y las variables de cada φ_i pertenecen a $\{x_1, \ldots, x_k\}$.

- Una *relación restrictiva de aridad k sobre* Ω, es un conjunto finito $r = \{\psi_1, \ldots, \psi_M\}$, donde cada ψ_i para $1 \leq i \leq M$, es una *k-tupla restricción* sobre las variables $x_1, \ldots, x_k$. Una *relación restrictiva* se describe mediante una fórmula disyuntiva de la forma $\psi_1 \vee \ldots \vee \psi_M$.

- Partiendo de las definiciones anteriores, una *Base de Datos de Restricciones* es una colección finita de *relaciones restrictivas* de cualquier aridad finita.

Estableciendo la analogía con la teoría de base de datos, una relación r con aridad k se define como un conjunto finito de *k-tuplas restricción* (con k atributos), o puntos en un espacio de k-dimensiones. Pero si el número de tuplas que existe es infinito, no podrá ser representado mediante una relación tal como se conoce en las bases de datos relacionales. En la bibliografía, esta representación de la información mediante puntos se le denomina **relación no definida con restricciones (unrestricted relation)**. La relación mediante restricciones definida en el **Modelo de BDdR** proporciona la posibilidad de representar estos puntos aunque sean infinitos de una forma finita en un espacio k-dimensional, dando origen a las **relaciones restrictivas (constraint relation)**, lo que introduce las siguientes definiciones:

Definición 3.9: Interpretación de Relaciones Restrictivas. Sea Ω un vocabulario, y $\mathcal{M} = (\mathcal{U}, \Omega)$ una $\Omega-$estructura (un vocabulario Ω sobre un dominio $\mathcal{U}$). Sea r una relación restrictiva de aridad k sobre Ω, y $\varphi_r(x_1, \ldots, x_k)$ la fórmula correspondiente a la relación r. Entonces r representa todos los puntos $a_1, \ldots, a_k$ tales que $\varphi(a_1, \ldots, a_k)$ es satisfactible en $\mathcal{M}$. Lo que, acorde con la definición 3.7, se representa:

$$\{(a_1, \ldots, a_k) \in \mathcal{U}^k \mid \mathcal{M} \models \varphi_r(a_1, \ldots, a_k)\}$$

Definición 3.10: Relación Extendida. La representación de los posibles conjuntos infinitos de puntos en un espacio de k dimensiones que describe una relación restrictiva, se define como una extensión de la relación o relación extendida. De esta definición se puede inferir la definición de **Bases de Datos Extendidas o Extensivas** como la unión de relaciones extendidas.

Definición 3.11: Base de Datos Definible con Restricciones. Cualquier colección finita de relaciones no definidas con restricciones (unrestricted relations) se le llama **Base de Datos no definida con restricciones (unrestricted database)** Y una base de datos sin restricciones o extendida se le denomina *Definible con Restricciones*, si todas sus relaciones son definibles mediante restricciones siguiendo la definición3.7. Por lo tanto, una Base de Datos de Restricciones D representará una Base de Datos Definible D', si D representa con restricciones todas las relaciones extendidas de D'.

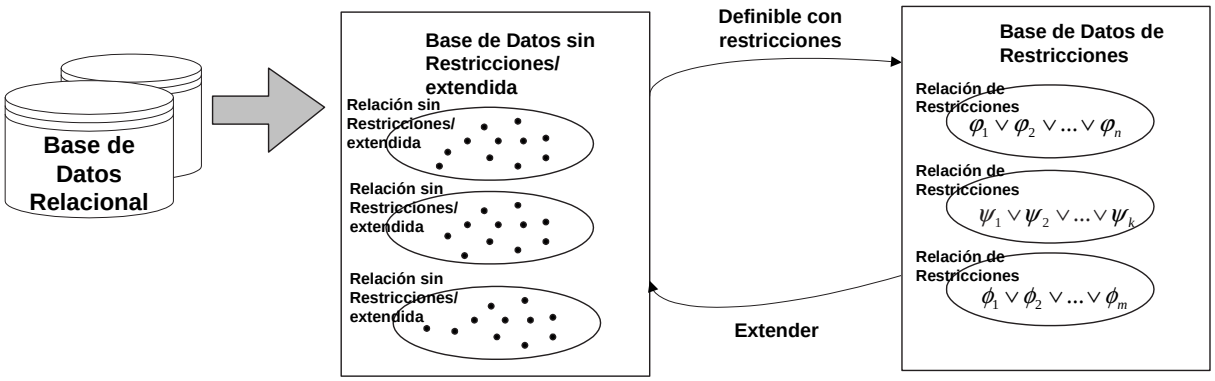

Figura 3.3: Comparación entre las Bases de Datos Extensivas y las BDdR

En la figura 3.3 se muestran las ideas de las equivalencias entre las bases de datos no descritas con restricciones y las BDdR. Las bases de datos sin restricciones o extensivas, son aquellas donde la información está almacenada mediante todos los puntos que la representan, como es el caso de las bases de datos relacionales clásicas, con la diferencia

de que teóricamente pueden almacenar infinitos puntos. Si estas relaciones pueden ser representadas como restricciones (definibles mediante restricciones), se podrá representar como una BDdR. Del mismo modo, las BDdR representan un conjunto de puntos de manera simbólica, cuya transformación conllevaría la *extensión de las relaciones* y por consiguiente de la base de datos.

3.3.3. Ejemplo de Modelado con Relaciones Restrictivas

Anteriormente se ha mostrado cómo es posible representar distintos tipos de restricciones en función del vocabulario (Ω) utilizado. A continuación se mostrarán algunos ejemplos de cómo las relaciones restrictivas, correspondientes con estas clases de restricciones, pueden ser utilizadas adecuadamente para representar diferentes tipos de datos.

Restricciones lineales: El uso de restricciones lineales está más extendido en el dominio de las BDdR, frente al uso de las polinómicas. Esto es debido a que la evaluación de las consultas sobre restricciones lineales es más eficiente que sobre restricciones polinómicas.

Sea la relación restrictiva s, formada por las restricciones:

$(y = 2x \wedge \neg(x = y))$ y $(1 \leq x + y)$

La fórmula DNF (Disjunctive Normal Form) de s será:

$\varphi_s(x, y) \equiv (y = 2x \wedge \neg(x = y)) \vee (1 \leq x + y).$

Donde $\varphi_s(x, y)$ describe el conjunto de puntos que forman cada una de las restricciones en el espacio bidimensional *(x, y)*.

Restricciones de igualdad a constantes: Son un caso particular de restricciones lineales, por ejemplo la relación q formada por las tuplas en el espacio bidimensional *(x, y)*, $(1, 2)$ y $(3, 4)$. Dichas tuplas son equivalentes a las restricciones $x = 1 \wedge y = 2$ y $x = 3 \wedge y = 4$. Por lo tanto q corresponde a la fórmula $\varphi_q \equiv (x = 1 \wedge y = 2) \vee (x = 3 \wedge y = 4)$.

En general, todas las relaciones finitas de aridad k en el conjunto $\mathcal{D}$ con n tuplas $a_1^i, \ldots, a_k^i$ para $i = 1, \ldots, n$ pueden representarse como una relación restrictiva de igualdad mediante la fórmula:

$\varphi_r \equiv \bigvee_{i=1}^{n}((x_1 = a_1^i) \wedge \ldots \wedge (x_k = a_k^i))$

Restricciones polinómicas con inecuaciones: Sea la relación restrictiva t formada por las restricciones:

$$(x \geq y \wedge y \leq 6 - x \wedge y \geq 0),\ (x - 3 \geq y \wedge y \leq 9 - x \wedge x \geq 0) \text{ y } (x^2 + y^2 \leq 9)$$

La fórmula DNF correspondiente será:

$$\varphi_t(x, y) \equiv (x \geq y \wedge y \leq 6 - x \wedge y \geq 0) \vee (x - 3 \geq y \wedge y \leq 9 - x \wedge x \geq 0) \vee (x^2 + y^2 \leq 9)$$

La figura 3.4 presenta de forma gráfica el conjunto de restricciones definida por la fórmula φ_t, la cual representa un conjunto infinito de puntos en el espacio bidimensional *(x, y)*.

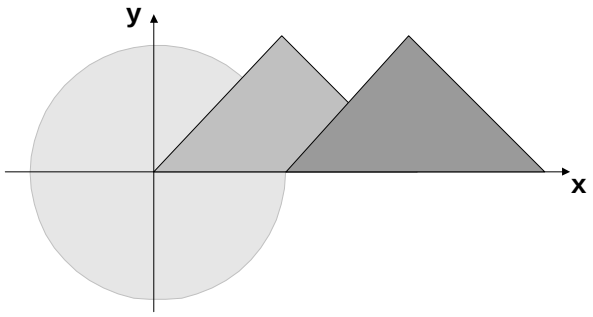

Figura 3.4: Figura representada por la fórmula φ_t

3.4. Consultas a Bases de Datos de Restricciones

Como se ha comentado anteriormente, la lógica de primer orden es un lenguaje declarativo, mientras que en las bases de datos se utilizan también lenguajes procedimentales como es el álgebra relacional, por lo que es necesario analizar cómo se adapta el álgebra relacional cuando se trata con restricciones descritas con lógica de primer orden.

3.4.1. Eliminación de cuantificadores

Una consulta a una base de datos clásica sobre una relación consiste básicamente en la selección de un conjunto de atributos que cumplen una determinada condición. Esta sección se centra en la selección de los atributos, lo que significa devolver sólo una parte de la relación involucrada en la consulta. En el caso de las BDdR, eliminar una parte de la información no siempre es trivial, e incluso en algunos casos no es posible.

Los algoritmos de eliminación de cuantificadores [101][152][98][145] transforman fórmulas con cuantificadores en otras equivalentes sin dichos cuantificadores. La operación de proyección sobre las variables de una restricción conllevan dicha eliminación de cuantificadores, donde un ejemplo muy utilizado es:

$\phi(a, b, c) \equiv \forall x(ax^2 + bx + c > 0)$, donde a, b y c son parámetros (variables libres) y x es la variable cuantificada a eliminar.

Tras la eliminación de x se obtendrá la fórmula equivalente a ϕ:

$\phi'(a, b, c) \equiv (a > 0 \wedge b^2 < 4ac)$

Como se comenta al principio de este apartado, no siempre es fácil o posible realizar esta eliminación [75], lo que se muestra en el siguiente ejemplo. Sea una$\Omega-$estructura ($\mathbb{Z}$; $\{+\}$, $\{0, 1\}$, $\{=\}$), y la fórmula:

$y = 3x + 2$

para eliminar la variable x obteniendo sólo y ($\forall x(y = 3x + 2)$), la fórmula no puede ser expresada sin la variable x, a menos que se incluyan otras funciones en el vocabulario, en este caso:

$y \ mod \ 3 = 2$

Una $\Omega-$estructura $\mathcal{M}$ admite eliminación de cuantificadores efectivos, si para todas las fórmulas $\varphi(x_1, \ldots, x_n)$ en la estructura $\mathcal{M}$, existe (y puede ser encontrada eficientemente) una fórmula libre de cuantificadores $\psi(x_1, \ldots, x_n)$ tal que satisfagan $\mathcal{M}$, lo que es equivalente a decir que:

$\mathcal{M} \models \forall(x_1, \ldots, x_n) \ \psi(x_1, \ldots, x_n) \leftrightarrow \varphi(x_1, \ldots, x_n)$

O lo que es lo mismo, que la relación extendida sin las variables que se quieren eliminar sea definible con restricciones. Sólo para la estructura ($\mathbb{R}$; $\{+, \cdot\}$, $\{0, 1\}$, $\{<\}$) es posible asegurar que se puede encontrar una fórmula libre de cuantificadores, para el resto de estructuras dependerá de las restricciones que se construyan con ellas.

3.4.2. Consultas sobre restricciones

Partiendo de la definición de las BDdR como un conjunto finito de relaciones de restricciones, un *Esquema de bases de datos (SC)* será definido como un conjunto finito no vacío de relaciones restrictivas con nombres. Estos *Esquemas* son necesarios para realizar

las consultas sobre la base de datos, pudiendo diferenciar una relación de otra. De esta forma, en las BDdR los *Esquemas* son equivalentes a lo que son las relaciones para las bases de datos relacionales.

Cuando nos enfrentamos a las consultas a BDdR, hay que tener en cuenta que a nivel conceptual son una representación simbólica de bases de datos extensivas, por lo que la consulta se complica. Esto significa que se pueden abordar de dos formas: a nivel conceptual sobre bases de datos extensivas, o a nivel de representación sobre las propias restricciones. Una consulta a nivel de representación será consistente sólo si corresponde con una consulta en el nivel conceptual, como se muestra en la figura 3.5.

Conceptualmente, las BDdR son representaciones simbólicas de lo que podrían ser bases de datos relacionales. De esta forma, se puede pensar en una consulta desde dos puntos de vista: en el nivel conceptual como una transformación sobre una base de datos relacional, donde se obtienen restricciones consultando por datos no restrictivos; o en el nivel de representación como una transformación de las restricciones (consulta sobre restricciones), donde se obtienen restricciones como salida.

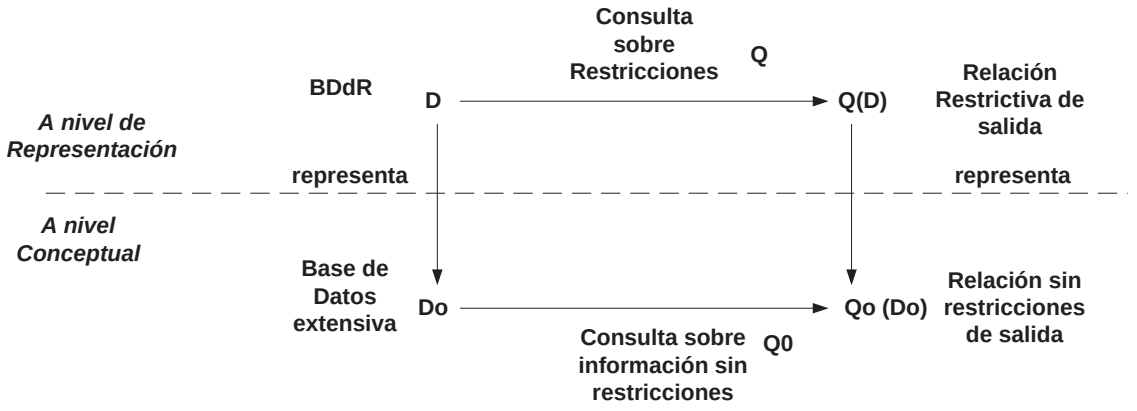

Figura 3.5: Consistencia en las consultas a restricciones

Un ejemplo de consulta presentada en [44] es la siguiente, donde un esquema se representa mediante la relación T de aridad 3 (x_1, x_2, x_3). En este caso, la base de datos extendida o sin restricciones consistirá en un conjunto de tuplas en $\mathbb{R}^3$, que puede ser representada como la unión finita de un conjunto de inecuaciones polinómicas en una BDdR. Las restricciones del esquema T para una base de datos D, se representarán de la forma $D(T) = \{\psi_1, \ldots, \psi_M\}$. Para realizar una consulta Q de aridad 2, donde sólo se quieren obtener las variables x_1 y x_2, se escribirá una fórmula de la forma:

$$\chi \equiv \exists x_3(\psi_1 \vee \ldots \vee \psi_M)$$

Si las restricciones del esquema están definidas con la estructura ($\mathbb{R}$; {+, ·}, {0, 1}, {<}), se admite la eliminación de cuantificadores, por lo tanto se puede obtener χ como una fórmula en función de x_1 y x_2 tras eliminar la variable x_3. χ se representará en DNF, con una fórmula de la forma:

$$\chi \equiv \gamma_1 \vee \ldots \vee \gamma_l$$

donde dicha fórmula será el resultado de evaluar la consulta Q sobre la base de datos D, representado como Q(D):={$\gamma_1 \vee \ldots \vee \gamma_l$}. La consulta a la restricción es consistente, ya que transforma cada conjunto semi-algebraico de $\mathbb{R}^3$ en sus proyecciones en el plano (x_1,x_2). Donde un conjunto X, consistente en tuplas de elementos sobre un dominio, se llama *semi-algebraico* si es definible con inecuaciones polinómicas sobre los reales.

Una vez construidas las consultas, éstas se evalúan reemplazando las instancias de los predicados de la base de datos por la fórmula que define la relación, y usando la eliminación de cuantificadores para convertir el resultado en la forma normal disyuntiva.

Las propiedades de las consultas a bases de datos relacionales, listadas abajo, contribuyen al éxito del modelo relacional. Por esta razón, las BDdR también deben cumplir estas propiedades:

- **Clausura:** Es una propiedad que poseen los lenguajes cuya álgebra relacional es cerrada, lo que significa que la salida de la evaluación de una consulta puede ser utilizada como relación de entrada para otra consulta. Esto significa que Q(D)=D', o lo que es lo mismo, que la salida de una consulta restrictiva aplicada a una relación restrictiva, debe ser también una relación restrictiva. La propiedad de clausura es muy útil, ya que se puede descomponer una determinada consulta, facilitando su escritura, en consultas más simples anidadas.

- **Eficiencia de evaluación:** Las consultas pueden ser evaluadas eficientemente en función de la complejidad de los datos y su cantidad, lo cual consiste en evaluar una consulta en términos del tamaño de la base de datos [31][91]. Si se utiliza el álgebra relacional como lenguaje de consulta, es posible definir una planificación para optimizar el tiempo o la memoria necesaria para su evaluación. Las propuestas que se analizan al final de este capítulo han tenido en cuenta este aspecto de las BDdR, cada una utilizando distintas estrategias.

- **Consistencia:** Sea D una BDdR y D_0 una base de datos extendida donde D representa a D_0. Una consulta sobre restricciones Q se le denomina consistente si existe una consulta no restrictiva Q_0 tal que $Q_0(D_0)$ es definible con restricciones, y $Q(D)$ representa a $Q_0(D_0)$.

Teniendo en cuenta que una consulta es una fórmula en la lógica apropiada, donde tanto la entrada como la salida son un conjunto de relaciones, se puede utilizar para el estudio de las BDdR dos lenguajes diferentes de consulta, el lenguaje del álgebra relacional y el cálculo relacional. Aunque ambos son equivalentes, en esta tesis se utiliza el álgebra relacional por ser procedimental en lugar de declarativo, como es el cálculo relacional. El álgebra relacional describe *cómo* se obtiene una relación partiendo de una o varias de entrada, especificándose de forma explícita un cierto orden, lo que implica una cierta estrategia al evaluar la consulta. Por otro lado, el cálculo relacional, derivando de una rama de la lógica simbólica denominada cálculo de predicados, simplemente especifica*qué* hay que extraer. En esta tesis se utiliza el álgebra relacional ya que permite establecer una equivalencia directa de cada una de las operaciones con las del lenguaje de consulta SQL, junto a la posibilidad de describir el orden en el que se evaluarán las distintas partes de las operaciones sobre una relación, con el objetivo de optimizar la evaluación de consultas.

3.4.3. Algebra Relacional para Restricciones

El lenguaje del **álgebra relacional** es un lenguaje procedimental, donde tanto los operandos como los resultados son relaciones (propiedad de clausura), permitiendo anidar expresiones. Dicho lenguaje tiene definido cinco operadores básicos: la proyección (π), la selección (σ), la unión ($\cup$), la diferencia ($-$), y el producto cartesiano ($\times$). Con los cuales se pueden construir expresiones de consulta, y definir operadores más complejos. Muchos trabajos han analizado cómo ampliar el álgebra relacional cuando las consultas se hacen sobre BDdR [44][10][58]. A continuación se presenta la redefinición propuesta en [44] del álgebra relacional para restricciones, que se basa en la definición de las *Expresiones del Álgebra Relacional* (Relational Algebra Expressions, RAEs). Sea un vocabulario Ω, una estructura $\mathcal{M}$ sobre Ω y un esquema de base de datos (SC) donde $\mathcal{U}$ denota el dominio de $\mathcal{M}$, las expresiones del álgebra relacional se definen de forma inductiva como:

- $\mathcal{U}$ es una *Expresión del Álgebra Relacional*

- Cada relación $R \in SC$ es una *Expresión del Álgebra Relacional*.
- Si e_1 y e_2 son *Expresiones del Álgebra Relacional* de aridad k_1 y k_2 respectivamente, el producto cartesiano $(e_1 \times e_2)$ es una *Expresión del Álgebra Relacional* de aridad k_1+k_2.
- Las *Expresiones del Álgebra Relacional*, involucradas en las operaciones de unión y diferencia, tienen que tener la misma aridad, y con ambas se obtiene una *Expresión del Álgebra Relacional* de esa aridad.
- Si e es una *Expresión del Álgebra Relacional* de aridad k, e $i_1, \ldots, i_p \in \{1, \ldots, k\}$, entonces la proyección $\pi_{i_1,\ldots,i_p}(e)$ es una *Expresión del Álgebra Relacional* de aridad p.
- Si e es una *Expresión del Álgebra Relacional* de aridad k, y ϑ es una fórmula libre de cuantificadores sobre Ω también de aridad k (con las variables $x_1, \ldots, x_k$), entonces la selección $\sigma_\vartheta(e)$ es una *Expresión del Álgebra Relacional* de aridad k.

Una vez definida la aridad de las operaciones, a continuación se muestra la equivalencia entre las bases de datos extensivas D y las consultas no restrictivas Q_e, en comparación con la BDdR D' y las consultas sobre restricciones Q'_e para las distintas *Expresiones del Álgebra Relacional*. Siendo e una *Expresión del Álgebra Relacional*:

- Si e es el dominio $\mathcal{U}$ (e de aridad 1), entonces $Q_e(D) := \mathcal{U}$, ya que se obtendrán todos los valores del dominio de forma extensiva. De forma intensiva será $Q'_e(D') := \{x_1 = x_1\}$, que representa de forma simbólica que x_1 puede tomar cualquier valor del dominio.
- Si e es $R \in SC$ (una relación de la base de datos), entonces $Q_e(D) := D(R)$ que representa todas las tuplas de los valores que pueden tomar las variables de la relación R, y $Q'_e(D) := D'(R)$ que representa las restricciones que forman las relación R.
- Si e es $(e_1 \times e_2)$, entonces $Q_e(D) := Q_{e_1}(D) \times Q_{e_2}(D)$. Siendo las variables de e_1 $\{x_1, \ldots, x_{k_1}\}$ y las de e_2 $\{x_1, \ldots, x_{k_2}\}$, se renombran las variables de e_2 usadas en $Q'_{e_2}(D')$ como $x_{k_1+1}, \ldots, x_{k_1+k_2}$ respectivamente. Por lo tanto:

  $$Q'_e(D') := \{\psi \wedge \chi\} \mid \psi \in Q'_{e_1}(D'), \chi \in Q'_{e_2}(D')$$

ya que cada restricción ψ y χ formarán una misma tupla y por lo tanto tendrán una relación de conjunción entre ellas.

- Si e es $(e_1 \cup e_2)$, entonces $Q_e(D) := Q_{e_1}(D) \cup Q_{e_2}(D)$, y $Q'_e(D') := Q'_{e_1}(D') \cup Q'_{e_2}(D')$, donde la relación entre las restricciones de $Q'_{e_1}(D')$ y $Q'_{e_2}(D')$ es de disyunción.

  $\gamma \equiv (\bigvee_{\psi \in Q'_{e_1}(D')} \psi) \vee (\bigvee_{\psi \in Q'_{e_2}(D')} \psi)$

- Si e es $(e_1 - e_2)$, entonces $Q_e(D) := Q_{e_1}(D) - Q_{e_2}(D)$. Considerando la fórmula $Q'_e(D')$, después de renombrar las variables $x_1, \ldots, x_k$ a $y_1, \ldots, y_k$ respectivamente, es $\psi(y_1, \ldots, y_k)$. Considerando la fórmula:

  $\gamma \equiv (\bigvee_{\psi \in Q'_{e_1}(D')} \psi) \wedge \neg(\bigvee_{\psi \in Q'_{e_2}(D')} \psi)$

  Poniendo γ en DNF, $\gamma_1 \vee \ldots \vee \gamma_l$, entonces $Q'_e(D') := \{\gamma_1, \ldots, \gamma_l\}$.

- Si e es $\pi_{i_1,\ldots,i_p}(e')$ con e' de aridad k y donde $i_1, \ldots, i_p \in \{1, \ldots, k\}$, esto implica que $Q_e(D) := \{(a_{i_1}, \ldots, a_{i_p}) \mid (a_1, \ldots, a_k) \in Q_{e'}(D)\}$.

  Para la representación mediante restricciones, se supone que la fórmula correspondiente a $Q'_{e'}(D')$ después de renombrar las variables x_1, ..., x_k a y_1, ..., y_k, construyendo $\psi(y_1, \ldots, y_k)$. De forma que a partir de la fórmula:

  $\chi(x_1, \ldots, x_k) \equiv \exists_{y_1}, \ldots, \exists_{y_k} \psi \wedge \bigwedge_{j=1}^{p} x_j = y_{i_j}$

  se genera una nueva fórmula libre de cuantificadores γ, donde se han eliminado las variables $y_m \in y_1, \ldots, y_k$ que no pertenecen a las p variables de la proyección. Dicha fórmula γ será equivalente a χ pero expresada en DNF tal que $\gamma \equiv \gamma_1 \vee \ldots \vee \gamma_l$, de forma que $Q'_e(D') := \{\gamma_1, \ldots, \gamma_l\}$

- Si e es $\sigma_\vartheta(e')$, entonces $Q_e(D) := \{\bar{a} \mid \bar{a} \in Q_{e'}(D)$ y $\mathcal{M} \models \vartheta(\bar{a})\}$. En este caso $\bar{a}$ representa los vectores de la misma aridad que e' con los valores de las variables que cumplen la condición ϑ. Suponiendo que la fórmula correspondiente a $Q'_{e'}(D')$ es ψ, poniendo $\psi \wedge \vartheta$ en DNF obtenemos $\gamma_1 \vee \ldots \vee \gamma_l$, de manera que $Q'_e(D') := \{\gamma_1, \ldots, \gamma_l\}$

De esta forma, para cada *Expresión del Álgebra Relacional* e, la consulta sobre restricciones Q'_e representa la consulta para información extensiva Q_e.

3.4.4. Evaluación de consultas en BDdR

Hay dos aspectos que hay que estudiar en las consultas a BDdR. La primera de ellas es cuándo se hace el tratamiento de las restricciones, si en tiempo de ejecución o en tiempo de compilación. Y la segunda de ellas es si dicha evaluación se realizará con estrategias ascendentes (bottom-up) o descendentes (top-down).

Los sistemas de BDdR en tiempo de ejecución estudian las restricciones únicamente cuando se evalúan las consultas, mientras que el tratamiento en tiempo de compilación conlleva simplificar y reescribir las restricciones cuando son almacenadas, antes de ser evaluadas para una consulta.

Cómo se realiza el proceso de evaluación se muestra en la figura 3.6, donde $\phi = \varphi(x_1, \ldots, x_m)$ representa una consulta sobre restricciones para las variables $x_1, \ldots, x_m$. Dicha consulta se evaluará sobre las relaciones cuyos nombres son R_1, ..., R_n, lo que se representa de la forma $\phi(R_1, \ldots, R_n{:}x_1, \ldots, x_m)$. Para una Base de Datos de Restricciones D concreta definida para una Ω-estructura $\mathcal{M}$, donde sus relaciones restrictivas expresadas en DNF son $r_1, \ldots, r_n$, el primer paso de la evaluación será la eliminación de cuantificadores. Tras la eliminación de cuantificadores, la salida de la consulta puede ser un conjunto infinito de puntos con m dimensiones que satisfacen la fórmula ϕ $(a_1, \ldots, a_m)$, o una representación mediante restricciones en DNF. La representación de los valores $(a_1, \ldots, a_m)$ que satisfacen ϕ de manera simbólica, puede definir un nuevo conjunto de relaciones definidas sobre $\mathcal{M}$ que a su vez pueden ser entrada de otra consulta (cumplen la propiedad de clausura).

3.5. Prototipos de Bases de Datos de Restricciones

Existen muchos prototipos e implementaciones para BDdR, cuyas principales diferencias radican en los modelos lógicos que utilizan para almacenar las restricciones, si tienen o no capacidad de inferencia basada en la Programación Lógica, los tipos de restricciones con las que trabajan, y cómo evalúan las consultas. A continuación se analizan las aportaciones más relevantes.

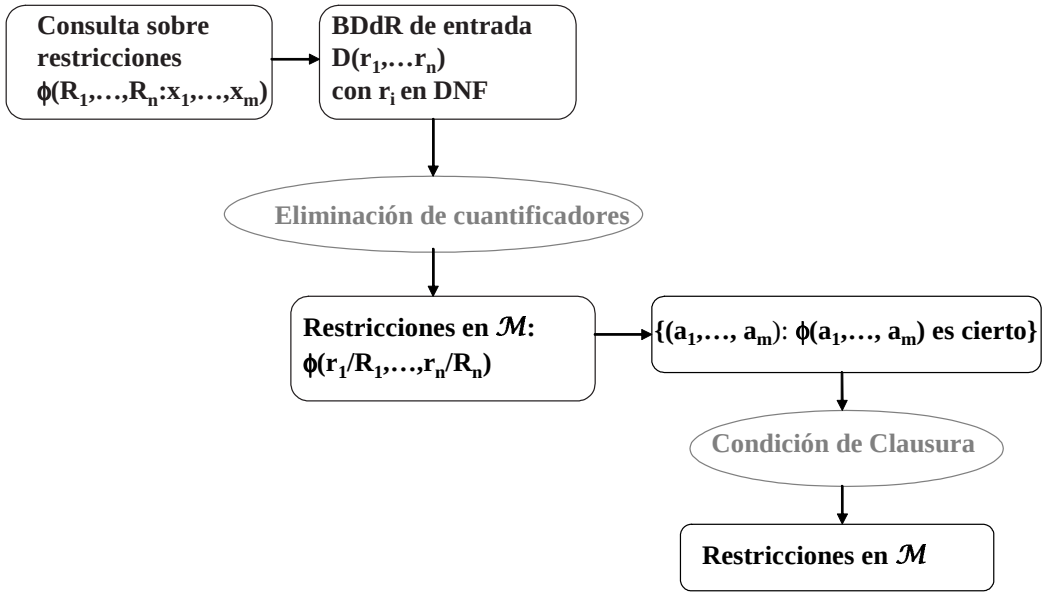

Figura 3.6: Evaluación de Consultas a Restricciones

3.5.1. DISCO

DISCO [24][129] (Datalog with Integer Set COnstraints) es un sistema de BDdR basado en la Programación Lógica con Restricciones, el cual implementa una reducción de Datalog para restricciones booleanas cuyas variables están establecidas en el dominio de los enteros, pudiendo definir el rango tanto de forma finita como infinita.

Como se explica en el capitulo 2, Datalog es un subconjunto de Prolog donde se definen reglas declarativamente, junto con un conjunto de relaciones existentes que se tratan como literales del lenguaje. Como los lenguajes declarativos, Datalog define lo que un programa desea lograr, en vez de especificar los detalles de cómo lograrlo, siendo una máquina de inferencia la que deduce hechos nuevos a partir de las reglas. DISCO añade a Datalog la posibilidad de incluir ciertas restricciones, que se describen a continuación, en el cuerpo de las cláusulas.

Sintaxis

La sintaxis de DISCO es básicamente la de Datalog, denotada como $Datalog^{<\mathbf{Z},\subseteq_{\mathbf{P}(\mathbf{Z})}}$, ya que el cuerpo de las reglas sólo puede contener conjunciones de restricciones de orden sobre enteros o conjuntos. Lo que significa que los problemas a resolver mediante DISCO están constituidos por un conjunto de *reglas* y *hechos* Datalog definidos con restricciones

booleanas, donde las reglas son transformadas a expresiones del álgebra relacional. En este caso, un programa es un conjunto finito de reglas de la forma: $R_0 :- R_1, \ldots, R_l$. La expresión de la cabecera R_0 es una fórmula atómica de la forma $p(v_1, \ldots, v_n)$, mientras que las reglas $R_1, \ldots, R_l$ son fórmulas atómicas del tipo:

- $p(v_1, \ldots, v_n)$: donde p es un predicado.
- $v\ \theta\ u$: donde u y v son variables o constantes enteras, y θ es un operador relacional $\{=, \neq, <, \leq, >, \geq, <_g\}$, donde g es un natural y $\{v <_g u\}$ representa la expresión $v + g < u$.
- $V \subseteq U$ o $V = U$: donde V y U son conjuntos de variables o constantes.
- $c \in U$ o $c \notin U$: donde c es un entero o una constante y U es un conjunto de variables o constantes.

Evaluación de Consultas

DISCO utiliza la evaluación 'bottom-up' basada en la eliminación de cuantificadores sobre las restricciones de la base de datos. Para mejorar los tiempos de consulta, y asegurar que terminan en el caso de que sean recursivas, propone una mejora reduciendo el dominio en cada búsqueda [131], junto a la planificación del tratamiento de las operaciones del álgebra relacional para optimizar la evaluación [95].

Ejemplo

Aunque los sistemas DISCO pueden utilizarse en gran cantidad de ejemplos [131], uno muy descriptivo es una base de datos donde se almacenan siete coches (a, b, c ,d , e, f, g) colocados en fila en una calle. Sobre su colocación conocemos algunos hechos: a y b no están aparcados al lado, e está delante de a con al menos 3 coches en medio, c está detrás de b con al menos 2 coches entre ellos, f está delante de g con al menos 4 coches entre ellos, y b y f están juntos. La base de datos que representa las posiciones entre coches de manera genérica es:

```
suc(1,2).
...
suc(6,7).
```

```
adj(x,y) :- succ(x,y).
adj(x,y) :- succ(y,x).
gap(x,y) :- x<1y. //x está a la izquierda de y, y no consecutivos
gap(x,y) :- y<1x. //y está a la izquierda de x, y no consecutivos

range(x) :- 0<x, x<8.

neq(x,y) :- x<y. //no son el mismo
neq(x,y) :- y<x. //no son el mismo

pos(1).
...
pos(7).
```

Y la consulta para conocer cuál es la posición de cada uno una de los coches será de la forma:

```
park(a,b,c,d,e,f,g) :- gap(a,g), e<3a, b<2c, f<4g, adj(b,f),
  range(a), range(b), range(c), range(d), range(e), range(f),range(g)
  neq(a,b), neq(a,c), neq(a,d), ..., neq(e,g), neq(f,g)
  pos(a), pos(b), ...
```

de donde se obtendrá la salida:

```
park(a,b,c,d,e,f,g) :- a=5, b=3, c=6, d=4, e=1, f=2, g=7.
```

Inconvenientes

Pese a que DISCO permite trabajar con datos infinitos mediante una representación finita, sí existen algunos inconvenientes a destacar en esta propuesta:

- Todos los avances en el campo de las bases de datos relacionales no se tienen en cuenta, ya que la información se almacena en un fichero, obviando aspectos como evitar redundancia de datos, mantener integridad referencial o la creación de estructuras de indexación.

- Al no estar la base de datos representada mediante entidades y relaciones, y al no estar normalizada, las modificaciones en el contenido de la base de datos conllevarán muchos cambios, lo que dificulta su mantenimiento. Por ejemplo, si en lugar de siete coches tuviéramos ocho, se tendrían que hacer cambios en las reglas *succ*, *range* y *pos*.
- Como consecuencia del inconveniente anterior, y al no existir la posibilidad de integridad referencial, sería fácil olvidar algunas de estas actualizaciones de contenido, originando bases de datos inconsistentes.
- Los cambios estructurales en la base de datos son muy costosos. Por ejemplo incluyendo una nueva variable en la cabecera de alguna regla o en algún hecho, todas las consultas donde dichas variables están involucradas tendrían que ser modificadas, junto a todas las reglas que hicieran uso de ella.
- El lenguaje de definición de datos y de consulta se aleja del estándar SQL, lo que hace difícil su utilización en el mercado.
- Las consultas son complejas y extensas. El rango en una base de datos relacional no se define de forma explícita, ya que la solución no puede tomar valores que no estén en la base de datos. Mientras que en el ejemplo, la definición del rango se define de forma explícita dos veces, en *range* y en las distintas instancias de *pos*.
- Pese a que la propuesta trabaja con valores enteros, y teóricamente sobre dominios infinitos, si la consulta infiere restricciones con infinitas soluciones el problema no podría ser tratado, al menos de forma completa.
- No permite trabajar con restricciones lineales ni polinómicas, sólo permitiendo restricciones de orden y sobre conjuntos, lo que limita mucho su uso.
- No ofrece la capacidad de inferir nuevas restricciones partiendo de las almacenadas, obteniendo como salida de una consulta un conjunto de restricciones tras realizar una eliminación de cuantificadores.

3.5.2. MLPQ/PReSTO

Esta propuesta presenta una combinación del sistema MLPQ (Management of Linear Programming Queries) y PReSTO (Parametric Rectangle Spatio Temporal Object), am-

bos desarrollados en la Universidad de Nebraska-Lincoln [129][133]. El sistema MLPQ da soporte a la gestión de bases de datos con restricciones lineales. MLPQ fue esecialmente diseñada para el tratamiento de información espacial, gracias a un entorno gráfico que permite trabajar en 2 ó 3 dimensiones, y a la interacción del usuario con el conocimiento almacenado mediante consultas. Por otra parte, PReSTO es un sistema de base de datos temporales, el cual permite consultas sobre sistemas que cambian en el tiempo.

Sintaxis

Las BDdR en MLPQ/PReSTO consisten en un conjunto de reglas Datalog [132], donde el cuerpo está formado por restricciones lineales. Estas restricciones lineales tienen la forma:

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n \ \theta \ b$$

donde a_i y b son constantes, x_i variables y θ es un operador de comparación de entre $\{=, <, \leq, >, \geq\}$.

La sintaxis del lenguaje de consulta de ambos sistemas (MLPQ y PReSTO) es muy similar a la de SQL, concretamente es un subconjunto de éste. En su lenguaje de consulta incorpora palabras reservadas de SQL como SELECT, WHERE, GROUP BY, HAVING, junto a agregados del tipo MIN y MAX.

Evaluación de Consultas

Pese a que la sintaxis de MLPQ/PReSTO es muy similar a SQL, los datos no son almacenados en una base de datos relacional, sino mediante reglas Datalog en un fichero de texto. Esto significa que la evaluación de las consultas conlleva una transformación de las consultas en SQL en una consulta Datalog. Los operadores que se permiten en las consultas contemplan la definición de vistas, agregados (como min y max), unión e intersección de relaciones, además de las consultas en función de una condición. Al basarse en Datalog, permite las consultas sobre estructuras recursivas, añadiendo un operador de aproximación **Apx** el cual garantiza obtener una solución en casos recursivos ya que establece el máximo número de soluciones que se pueden obtener.

Ejemplo

Un ejemplo de base de datos de MLPQ/PReSTO puede ser: almacenar un conjunto de países (*country(id, x, y, t)*) donde se relacionan los puntos geográficos pertenecientes a dicho país (x, y), el identificador del país (id) y el año (t) al que corresponde dichas características. La base de datos también almacena la localización de las ciudades (*location(c, x, y)*), donde (x, y) representa su situación geográfica y c el identificador de la ciudad. Junto a estas dos relaciones, *country* y *location*, también se almacena la relación *growth(t, c, p)* que representa la población p de una ciudad c en un instante de tiempo t. Un ejemplo de la información puede ser:

```
begin
  country(id,x,y,t):- id=1, x>=0, x<=4, y>=5, y<=15, t>=1800,
  t<=1950.
  country(id,x,y,t):- id=1, x>=0, x<=8, y>=5, y<=15, t>=1950,
  t<=2000.
  country(id,x,y,t):- id=2, x>=4, x<=12, y>=5, y<=15, t>=1800,
  t<=1950.
  ...
  location(c,x,y):- x=3, y=2, c=101.
  location(c,x,y):- x=7, y=3, c=102.
  location(c,x,y):- x=5, y=6, c=103.
  location(c,x,y):- x=7, y=10, c=104.
  ...
  growth(t,c,p):- c=101, p=10000, t>=1800, t<=2000.
  growth(t,c,p):- c=102, p=20000, t>=1800, t<=2000.
  growth(t,c,p):- c=103, p=10000, t>=1800, t<=2000.
  ...
end
```

Con esta información, se pueden realizar consultas como *'Obtener todas las ciudades que en 1900 tenían una población mayor de 10000'*, donde la consulta sería de la forma:

```
SELECT growth.c
FROM growth, location, country
```

```
WHERE growth.c=location.c,
  location.x=country.x,
  location.y=country.y,
  growth.t=1900,
  growth.p>10000
```

Inconvenientes

Esta solución comparte todas las desventajas que tiene DISCO con lo que respecta a la utilización de Datalog para el almacenamiento y representación de la información. De esta formas se hace patente la pérdida de la potencia de esta solución por no utilizar bases de datos relacionales.

- Cuando las soluciones tras la evaluación de las consultas tienen infinitas soluciones, MLPQ/PReSTO utiliza un operación de aproximación, llamado *Apx*, para asegurarse de su finalización. Al basarse en Datalog, permite tanto consultas como estructuras recursivas, lo que puede llevar a obtener un número de soluciones infinitas. Esto provoca que en algunos casos, el resultado de una consulta no sea completa, obteniéndose sólo algunas de las soluciones.
- No se pueden obtener nuevas restricciones derivadas de las almacenadas de una forma simbólica. Si esto fuera posible, se evitarían las limitaciones de devolver en una consulta infinitos valores y tenerlos que acotar mediante una aproximación.

3.5.3. DEDALE

DEDALE [69][70][134] es una de las primeras implementaciones de BDdR que utiliza la orientación a objetos para representar las restricciones. DEDALE fue un proyecto en el cual participaron los grupos de investigación *VERSO* y *VERTIGO* pertenecientes a la Universidad de París para dar soporte a restricciones lineales. DEDALE fue desarrollado para trabajar exclusivamente para aplicaciones geográficas, ya que su sintaxis y su semántica se centra en el tipo de consultas que se realizan en este ámbito.

Sintaxis

DEDALE dispone de un lenguaje de consulta muy similar a SQL, mejorando la manipulación de los objetos en el espacio. Entre las operaciones algebraicas que permite realizar entre los objetos geográficos, se encuentran la intersección, unión, diferencia, producto cartesiano, selección y la proyección. Para trabajar con las operaciones típicas sobre objetos geográficos define macros, por ejemplo para conocer si dos objetos son adyacentes, define funciones sobre los límites o los vértices de los objetos [46].

Evaluación de Consultas

Al almacenar la información en una base de datos orientada a objetos, realiza una transformación de las consultas en DEDALE a OQL (analizado en la sección 2.4.2) para el tratamiento de los atributos atómicos o simples, aquellos que no son restricciones. Para la mejora de la eficiencia en la evaluación de consultas referentes a las aplicaciones geográficas, sobre restricciones, aporta un conjunto de algoritmos para la optimización de las consultas presentados en [69]. Para las consultas sobre datos multidimensionales utiliza algoritmos basados en árboles como la variante de R-Tree, R*Tree [8].

Ejemplo

Uno de los ejemplos de consulta sobre una base de datos DEDALE se muestra a continuación. El lenguaje de consulta desarrollado en DEDALE se basa en la sintaxis de SQL, realizando transformaciones de las consultas usadas en los sistemas de información geográfica a OQL, como se observa en el siguiente ejemplo:

```
SELECT b.name
FROM Resort a , Resort b
WHERE a.name="Skiing Area"
  AND b.name="Building"
  AND b.geometry SOUTH-OF a.geometry
```

Lo que significa que siendo `p1` los puntos pertenecientes a `a` y `p2` los pertenecientes a `b`, si `b` está al sur de `a`, significa que siendo `p1.x`=`p2.x`, entonces `p1.y`>`p2.y`. Lo que DEDALE transformará en la siguiente consulta en OQL:

```
SELECT b.name
FROM Resort a, Resort b
WHERE a.name="Skiing Area"
  AND b.name="Building"
  AND (a.geometry JOIN b.geometry[1, 2 AS 3]) JOIN?
  'geometry.2>geometry.3'
```

donde `b.geometry[1, 2 AS 3]` describe que es necesario renombrar la variable `y` en `b`, para que pueda tomar valores diferentes a los que toma en `a`. También se puede destacar cómo la operación `a.geometry JOIN b.geometry[1, 2 AS 3]` es la unión de las variables que tienen en común ambos atributos (`x` en este caso), lo que permite seleccionar, mediante la comparación 'geometry.2 > geometry.3', aquellos nombres de `b` que está al sur de `a`.

Inconvenientes

Pese a que el salto cuantitativo fue importante para esta propuesta, ya que por primera vez se utilizaron las bases de datos orientadas a objetos, aún tiene aspectos que se pueden mejorar:

- Toda la sintaxis y la semántica del lenguaje está orientada al tratamiento de sistemas de información geográficos, lo que provoca que esta solución no sea genérica para cualquier sistemas definido mediante restricciones.
- Al utilizar las bases de datos orientadas a objetos como modelo lógico, toda la información se almacena como atributos dentro de un objeto, esto hace que mantenga todos los inconvenientes que se analizaron en la sección 2.4.2, con respecto las bases de datos orientadas a objetos.
- Al basarse en sistemas espacio-temporales, el número de dimensiones del dominio de los problemas no puede ser mayor que cuatro. Esta característica hace que la operación de proyección no contemple qué ocurre cuando las tuplas que forman una relación tengan variables distintas y cómo afecta esto a las distintas operaciones.

3.5.4. CCUBE

CCUBE (Constraint Comprehension Calculus) [19][20] define el lenguaje de manipulación de datos al que le debe su nombre. CCUBE presenta una integración de restricciones con la orientación a objetos para la optimización de consultas sobre objetos. El modelo de datos para el cálculo de restricciones está basado en los objetos que describen restricciones espacio-temporales (Constraint Spatio-Temporal, CST). En la implementación de CCUBE se plantea la definición de un sistema con capacidad expresiva para datos complejos, pero sin perder de vista la necesidad de que el sistema sea eficiente a nivel práctico.

Sintaxis

Con respecto a la sintaxis del lenguaje CCUBE, cabe destacar que utilizó como punto de partida la propuesta desarrollada por Kim et al. [87] donde se propone el lenguaje XSQL, el cual fue creado para adaptar la formalización de la lógica de primer orden a los lenguajes orientados a objetos. Otra de las aportaciones relevantes en el ámbito de la orientación a objetos, que influyó en CCUBE, fue una propuesta de Fegaras y Maier [50] llamado *monoid calculus*, probando su eficacia para lenguajes orientados a objetos como OQL. El *monoid calculus* es un formalismo algebraico para definir operaciones sobre estructuras algebraicas sencillas, proponiendo un tipo único de datos junto a los operadores definidos en las bases de datos relacionales y las orientadas a objetos. De esta forma, se presenta como una semántica formal para el lenguaje OQL.

Los formalismos anteriores fueron ampliados con restricciones lineales para la definición de un lenguaje de consulta sobre restricciones orientadas a objetos. El formalismo de Kim et al. fue extendido [18][21] con la definición del lenguaje $\mathcal{L}_{yri}\mathcal{C}$. La formalización de Fegaras y Maier [50] fue ampliada con el lenguaje CCUBE [17], creando un sistema de restricciones orientado a objetos que implementa disyunciones y conjunciones sobre restricciones lineales representadas con objetos en C^{++}.

Evaluación de Consultas

El modelo de datos de CCUBE se basa en restricciones espacio-temporales (CST, Constraint Spatio-Temporal) representadas mediante objetos. La evaluación de las consultas

utiliza un conjunto de bibliotecas implementadas en C^{++} que definen la clase CST, la cual implementa métodos sobre restricciones como $\vee$ y $\wedge$, permitiendo combinaciones con operadores lógicos entre restricciones lineales. Las operaciones de dicha clase aseguran tiempo polinómico en la evaluación de consultas, ya que utiliza el paquete CPLEX [79] desarrollado por Robert E. Bixby [16], el cual resuelve sistemas con restricciones lineales de gran tamaño.

Ejemplo

Un ejemplo de definición de objetos en CCUBE presentado en [20], es el objeto `my_desk`, el cual es de la clase DESK mostrado en la figura 3.7. En este ejemplo, el atributo `extent` es del tipo CST, el cual tiene la capacidad de representa restricciones lineales.

```
Desk my_desk = desk(
  catalogNo = 22354,
  name = 'one drawer desk',
  color = 'red',
  extent = (-4≤w≤ 4 ∧ -2≤z≤2)
  drawer = new drawer(
    color = 'blue'
    center = (p=−2 ∧ -3≤q≤−1)
    extent = (-1≤w1≤1 ∧ -1≤z1≤1)
    translation = (w=w1+p ∧ z=z1+q)
  )
 );
```

Algunos de los aspectos que presenta este ejemplo mediante sus atributos son: el tamaño de `desk` y `drawer` vienen definidos mediante sus correspondientes atributos `extent`, mientras que la posición de `drawer` con respecto a `desk` la define la restricción del atributo `translation`; el movimiento de `drawer` no puede ser cualquiera, ya que el atributo `center` define que sólo se puede mover hacia arriba o abajo, ya que `p` toma el valor único -2.

La siguiente consulta es un ejemplo donde se obtiene como salida nuevos objetos CST, encontrándose para cada escritorio (*desk*) de la clase `all_desks` su posición en una

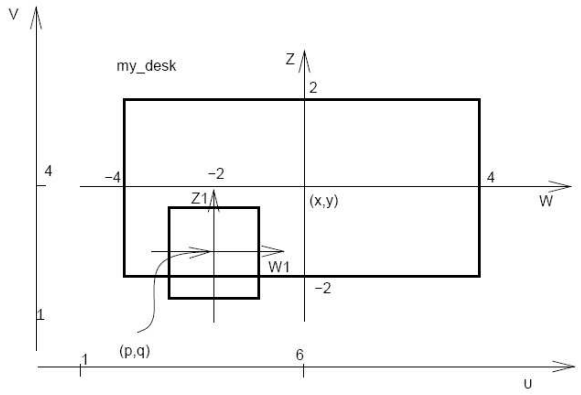

Figura 3.7: Ejemplo de instancia de un escritorio (desk) con un cajón (drawer) en una habitación

habitación, asumiendo que el centro del escritorio está en el punto (6, 4) y la ecuación de traslación (con respecto a las coordenadas de la habitación) es $u = w + x \wedge v = z + y$. La consulta será de la forma:

```
select new CST((u,v) | (dsk →extend ∧
  u=w+x ∧ v=z+y ∧ x=6 ∧ y=4 ))
into {bag⟨CST⟩} result
  all_desk as {desk} dsk
```

que obtiene nuevos objetos `CST` con las coordenadas u, v del escritorio con respecto a la habitación, para el nuevo centro del escritorio tras su traslado. Cabe denotar que {`bag⟨CST⟩`} determina el tipo del parámetro resultado, que en este caso será un conjunto de objetos del tipo `CST`.

Inconvenientes

Al igual que DEDALE, CCUBE representa un importante salto cuantitativo en las BDdR ya que utiliza la orientación a objetos, pero también tiene sus inconvenientes.

- Al igual que DEDALE, CCUBE mantiene todos los inconvenientes de las bases de datos orientadas a objetos.
- Una de las deficiencias de esta aportación, también al igual que DEDALE, es la dependencia de resolver problemas del ámbito espacio-temporal, no aportando una solución independiente de la aplicación.

- Los objetos de la clase `CST` tienen definidas operaciones booleanas pero no técnicas simbólicas de sustitución y eliminación de variables, por lo que la operación de proyección tiene ciertas limitaciones.

3.5.5. CQA/CDB

CQA/CDB [61] es un prototipo de BDdR especialmente desarrollado para el tratamiento de sistemas espacio-temporales implementado en Java^{TM}. Por esta razón, esta solución sólo utiliza restricciones lineales, que según sus creadores son suficientes para aproximar los problemas del ámbito espacio-temporal.

Sintaxis

La sintaxis de CQA/CDB es similar a la de SQL, pero incluyendo nuevas operaciones como *project* para trabajar sobre variables, y operaciones sobre agregados como *area* utilizado sobre sistemas espaciales [2]. CQA/CDB amplía los tipos de atributos que puede tener una tupla en dos: atributos clásicos o atributos que representan restricciones.

Evaluación de Consultas

CQA/CDB define una extensión del álgebra relacional definida sobre restricciones, ampliando las operaciones primitivas. Como mejora en la optimización de las consultas, propone la reordenación del orden de evaluación de los operadores y el uso de indexación utilizando R*-Tree para minimizar el número de objetos que se analizan en la evaluación de las consultas.

Ejemplo

Un ejemplo presentado en [59] para aclarar esta aportación, es una base de datos donde está almacenada el movimiento de un huracán sobre una región. Consiste en tres relaciones cuyos esquemas son:

```
Parcela:[IdParcela:cadena; x,y:restricción]
PropietarioParcela:[nombre:cadena; t:restricción; IdParcela:cadena]
Huracán: [x,y,t:restricción]
```

Este esquema define un conjunto de parcelas definidas sobre$x$ e y, y sus propietarios a lo largo del tiempo (definido por la variable t). La información *Huracán* define la posición con respecto a x e y de un huracán a lo largo del tiempo, también representando el tiempo con la variable t.

Y algunos ejemplos de consulta serán:

- Obtener todas las parcelas por las que pasó el huracán:

  R0 = `join Huracán and Parcela`
  R1 = `project` R0 `on IdParcela`

- Obtener los nombres de los propietarios de las parcelas que fueron azotadas por el huracán entre los instantes 4 y 9.

  R0 = `join PropietarioParcela and Parcela`
  R1 = `select t`$>= 4$, `t`$<= 9$ `from Huracán`
  R2 = `join` R0 `and` R1
  R3 = `project` R2 `on nombre`

Inconvenientes

Entre los inconvenientes de esta propuesta se encuentran:

- No hace una referencia expresa a qué modelo lógico utiliza para almacenar la información.
- Aunque CQA/CDB permite diferenciar los datos clásicos de las restricciones, ignora las características propias del álgebra relacional de Codd [32].
- Pese a que la sintaxis tiene elementos en común con SQL, aparecen operadores nuevos como PROJECT, y cambia la semántica de otros como SELECT, como ocurre en el segundo ejemplo mostrado para la sentencia $R1$.
- No describe de forma explícita que se ciña a soluciones exclusivamente espacio-temporales, aunque los ejemplos que se presentan en la la bibliografía no tratan restricciones de otra naturaleza.

- La definición de la selección y la proyección se hace sobre un conjunto de restricciones definidas sobre las misma variables, pero no se establece su comportamiento cuando las restricciones tienen distintas variables.

3.5.6. DeCoR

La porpuesta DeCoR [67][66] presenta una implementación de BDdR apoyándose en Bases de Datos Deductivas. Esta propuesta es de especial interés por la presentación de un conjunto de transformaciones y reescrituras de la base de datos en tiempo de compilación [68]. A esta reescritura se le denomina *Constraint Lifting*, mejorando la eficiencia en la evaluación de las consultas. Las BDdR de este sistema sólo permite trabajar con restricciones lineales sobre el dominio de los reales.

Sintaxis

Al estar soportada por bases de datos Datalog, las restricciones están descritas mediante reglas y hechos pero con una sintaxis que intenta acercarse a un lenguaje de bases de datos. Por ejemplo utiliza la palabra `create` para la creación de relaciones (tablas), `assert` para la inserción de información, y `query` para realizar consultas. Al basarse en las Bases de Datos Deductivas, fue necesario la definición de las equivalencias entre la terminología deductiva y la relacional. Estas equivalencias son las mostradas en la tabla 3.1.

Terminología Deductiva	Terminología relacional
Predicado	Relación
Argumento	Atributo
Hecho	Tupla
Regla	Vista
Fórmula	Expresión algebraica relacional

Tabla 3.1: Relación entre la Terminología Deductiva y la Relacional

Evaluación de Consultas

El Constraint Lifting Algorithm (CLA) consiste en una transformación donde el objetivo es mejorar los tiempos de evaluación de consultas. Esta transformación consiste en reescribir las cláusulas para que durante la evaluación sólo los hechos relevantes para la consulta sean analizados. La idea principal es manipular y simplificar las restricciones contenidas en la base de datos, de manera que no sea necesaria su manipulación en tiempo de evaluación, salvo para instanciar las variables en función de la consulta y comprobar si las restricciones son satisfactibles.

Un ejemplo fácil de transformación de las restricciones es el siguiente:

$$q(X) \leftarrow X + Y \leq 6 \wedge 2 \leq X \wedge p_1(X,Y) \wedge p_2(Y).$$

$$p_1(X,Y) \leftarrow 1 \leq X \wedge b_1(X,Y).$$

$$p_2(X) \leftarrow b_2(X,Y).$$

En una evaluación bottom-up, se generarían valores de X en p_1, por ejemplo $X = 1$, que no harían satisfactible q(X). Pero si las reglas se transformaran en:

$$q(X) \leftarrow X + Y \leq 6 \wedge 2 \leq X \wedge p_1'(X,Y) \wedge p_2'(Y).$$

$$p_1'(X) \leftarrow X + Y \leq 6 \wedge 2 \leq X \wedge 1 \leq X \wedge b_1(X,Y).$$

$$p_2'(X) \leftarrow X + Y \leq 6 \wedge 2 \leq X \wedge b_2(X,Y).$$

reduciendo p_1' de la forma:

$$q(X) \leftarrow X + Y \leq 6 \wedge 2 \leq X \wedge p_1'(X,Y) \wedge p_2'(Y).$$

$$p_1'(X) \leftarrow X + Y \leq 6 \wedge 2 \leq X \wedge b_1(X,Y).$$

$$p_2'(X) \leftarrow X + Y \leq 6 \wedge 2 \leq X \wedge b_2(X,Y).$$

evitaría la generación de muchos hechos que finalmente no serían satisfactibles. Para evaluaciones up-down, el proceso de reescritura conllevaría generar una nueva q' con las restricciones de p_1 de forma análoga.

Ejemplo

Un ejemplo de creación de tablas y consulta es:

```
open test/test // apertura de la base de datos deductiva
create scale(str, float, float, float, float) //creación de tabla
assert scale(str, 'sg', 0, 5000, 0, 0) //rellena la tabla scale
...
create persons(str, float) //crear la tabla persons
assert persons('Tom', 80000) //rellenar la tabla persons
...
create taxes(str, float, float) //crear la tabla taxes
assert persons(Kt, Sal, Taxes)← //rellenar la tabla taxes
  scale(Kt, Min, Max, X, Y) &//crea relaciones entre las tablas
  Sal>=Min & Sal<Max & //establece restricciones
  Taxes = X*Sal+Y //establece restricciones
```

Y un ejemplo de consulta podría ser: ¿qué impuestos deben pagar las personas de cualquier departamento? Cuya sentencia sería:

```
query taxes(Dep, Sal, Taxes) & persons(Pers, Sal)
```

Inconvenientes

DeCor al utilizar Datalog, comparte todos los inconvenientes comentados en DISCO y MLPQ/PReSTO. Junto a los inconvenientes propios a Datalog se puede añadir, al igual que ocurren en las demás propuestas, que no es posible inferir nuevas restricciones partiendo de las almacenadas.

3.6. Comparativas entre las Propuestas

Para realizar la comparación entre las distintas propuestas, y la que es la aportación de esta tesis (LORCDB), se utilizan seis parámetros que ayudarán a determinar las diferentes características que en este documento se han considerado importantes para las BDdR, y que se muestran en la tabla 3.2.

1. **Soporte Lógico de Almacenamiento:** Como se expuso en la sección 2.4, los datos pueden ser almacenados de forma diferente en función del soporte lógico que utilicen. Propuestas como MLPQ/PReSTO hacen referencia a la utilización de una base de datos, cuando en realidad utilizan un fichero de texto plano para el almacenamiento de las restricciones. Esto significa que no trabajan con bases de datos relacionales, desaprovechando su potencia, capacidad de modularidad, soporte masivo de datos, capacidad de definir referencias entre relaciones ... Las propuestas basadas en la orientación a objetos, como DEDALE y CCUBE, almacenan toda la información como objetos, no separando lo que es la información discreta de la restrictiva. Esta decisión obliga a tratar de forma similar tipos de datos de carácter diferente.

   Para contrarrestar los inconvenientes de las bases de datos orientadas a objetos, pero sin perder la capacidad expresiva de la orientación a objetos, en esta tesis se propone la utilización de Gestores de Bases de Datos Objeto-Relacionales. De esta forma se aprovechan todas las ventajas de las cualidades relacionales y la capacidad expresiva de la orientación a objetos.

2. **Programación Lógica:** Algunas propuestas de BDdR incorporan las Programación Lógica, permitiendo definir estructuras y lenguajes recursivos. Las propuestas que que permiten la Programación Lógica son las cercanas a Datalog y por lo tanto a las Bases de Datos Deductivas, combinando en las cláusulas tanto predicados como restricciones. Este aspecto no se trata en esta tesis, sólo permitiendo trabajar con Programación con Restricciones, sin añadir todas las características de la Programación Lógica.

3. **Lenguaje similar a SQL:** En muchas de estas propuestas, como DISCO y DeCoR, el lenguaje que se utiliza para la creación y consulta de las bases de datos está muy alejado del estándar SQL, siendo mucho más próximos a la sintaxis de Datalog. Esto obliga a que los usuarios deban aprender este lenguaje mucho menos extendido en las bases de datos que SQL. Se considera que el acercamiento de los lenguajes sobre bases de datos de restricciones a SQL es imprescindible, ya que al ser SQL un estándar de facto, facilita la incorporación de las BDdR al mercado. La propuesta LORCDB, mantiene toda la sintaxis y semántica de SQL, añadiendo funcionalidad relativa a restricciones. Esto hace posible que aplicaciones sobre bases de datos ya existentes fueran compatibles con su ampliación a BDdR.

4. **Tipos de Restricciones:** Pese a que en la teoría de BDdR se contempla la posibilidad de trabajar con restricciones polinómicas, la mayoría de las soluciones, al trabajar con problemas espacio-temporales, definen operaciones exclusivamente para restricciones lineales. Pese a que en muchos casos será suficiente con utilizar las restricciones lineales ya que pueden aproximar la solución, en general las restricciones lineales no son suficientes. Por esta razón se propone ampliar los tipos de restricciones también a polinómicas.

5. **Independiente de la Aplicación:** Los ejemplos para los cuales se han desarrollado estos prototipos son mayoritariamente los datos espacio-temporales, como MLPQ/PReSTO o CQA/CDB. Eso enfoca la sintaxis y la semántica de las consultas a la aplicación, no siendo soluciones generales sino dependiente del problema. Por esta razón en esta tesis se propone la definición de un lenguaje independiente de la aplicación, pero que se pueda adecuar a ella.

6. **Definición del tipo Variable de Restricción:** La definición de BDdR radica en describir la relaciones de una base de datos como una restricción mediante los operadores booleanos de conjunción y disyunción. Propuestas como CQA/CDB apuntan a la definición de dos tipos de atributos, clásicos y restricción. Sin embargo, en esta tesis se propone definir tres tipos de atributos: clásicos, restricciones y variables de restricciones. Ninguna de las propuestas existentes hasta el momento describe el álgebra relacional para estos tres tipos de atributos, como el álgebra relacional hace con los tipos clásicos. Todas las propuestas enmarcan las operaciones sobre un conjunto de restricciones definidas sobre las mismas variables, no analizando la semántica de los operadores cuando las restricciones se definen sobre múltiples variables distintas. Esto permite inferir nuevas restricciones partiendo de las almacenadas, utilizando sustitución simbólica de variables.

3.7. Resumen

En este capítulo se muestran los formalismos que se han establecido para definir las Bases de Datos de Restricciones. Este análisis va desde la definición de qué es una *Restricción* utilizando la lógica de primer orden, hasta la combinación de las restricciones para dar origen a las BDdR. Para definir el nuevo lenguaje de consultas sobre dichas bases

	Modelo Lógico	Programación Lógica	Similar a SQL	Tipo de Restricción	Independiente de la Aplicación	Variable de Restricción
MLPQ/ PReSTO	Archivo	Sí(Datalog)	Sí	Lineales	No	No
DISCO	Archivo	Sí(Datalog)	No	Lineales	No	No
DEDALE	BDOO	No	Sí	Lineales	No	No
CCUBE	BDOO	No	Sí	Lineales	No	No
CQA/CDB	¿?	No	Sí	Lineales	No	No
DeCor	Archivo	Sí(Datalog)	No	Lineales	Sí	No
LORCDB	BDOR	No	Sí	Lineales y Polinómicas	Sí	Sí

Tabla 3.2: Características de los prototipos de Bases de Datos de Restricciones

de datos, se propuso una extensión del álgebra relacional adaptada a las nuevas necesidades, todo ello haciendo posible realizar la evaluación de consultas sobre restricciones. Basándose en esas definiciones, se ha originado un conjunto de propuestas con distintas características: en función del modelo lógico de almacenamiento; si utilizan Programación Lógica o sólo Programación con Restricciones; tipo de lenguaje de consulta y si estos permiten trabajar sobre un dominio de aplicación genérico; tipos de restricciones a las que dan soporte (lineal o polinómica); y definición de operaciones sobre múltiples variables de restricciones. Todas estas características se presentan en una comparativas de las distintas propuestas, detectando las deficiencias que se mitigarán en la propuesta LORCDB, la que se detalla en esta memoria de tesis.

Capítulo 4

Propuesta de Modelado para las Bases de Datos de Restricciones

En este capítulo se reformulan algunas de las definiciones relacionadas con las Bases de Datos de Restricciones, con el objetivo de diferenciar los datos extensivos de los intensivos, de forma que la Base de Datos permita tratar y almacenar datos heterogéneos de una manera homogénea. Esto deriva en la distinción de tres tipos de atributos: los tipos clásicos, el tipo restricción y el tipo variable de restricción. Esto conlleva también la necesidad de redefinir las operaciones primitivas del álgebra relacional para estos nuevos tipos, describiendo la sintaxis y la semántica de las distintas operaciones cuando los nuevos tipos están involucrados. Al ampliarse los tipos de atributos de la bases de datos, será necesario utilizar nuevas técnicas para la evaluación de las consultas, las cuales también se presentan en este capítulo.

4.1. Introducción

La propuesta que se presenta en esta tesis cambia la definición de BDdR, introduciendo nuevos conceptos fundamentados en la idea de que una relación formada por un conjunto de tuplas puede ser representada como una combinación de restricciones y atributos clásicos (de dominios atómicos). Esta idea es la presentada en la figura 4.1, que utiliza un ejemplo que muestra la equivalencia semántica entre un conjunto de tuplas de una relación y una restricción (4.1.a y 4.1.b), cuya transformación se explicó en la sección 3.2.

La redefinición de las BDdR aborda el tratamiento de las tuplas como atributos de valores extensivos (aquellos donde las variables se puedan representar mediante restricciones de igualdad entre variables y valores constantes) y atributos intensivos, que representarán relaciones entre variables en forma de restricción. Esto da origen a relaciones cuyos atributos pueden ser de dos tipos, siendo equivalentes a una relación extendida o a una relación restrictiva, como se presenta en la figura 4.1.c.

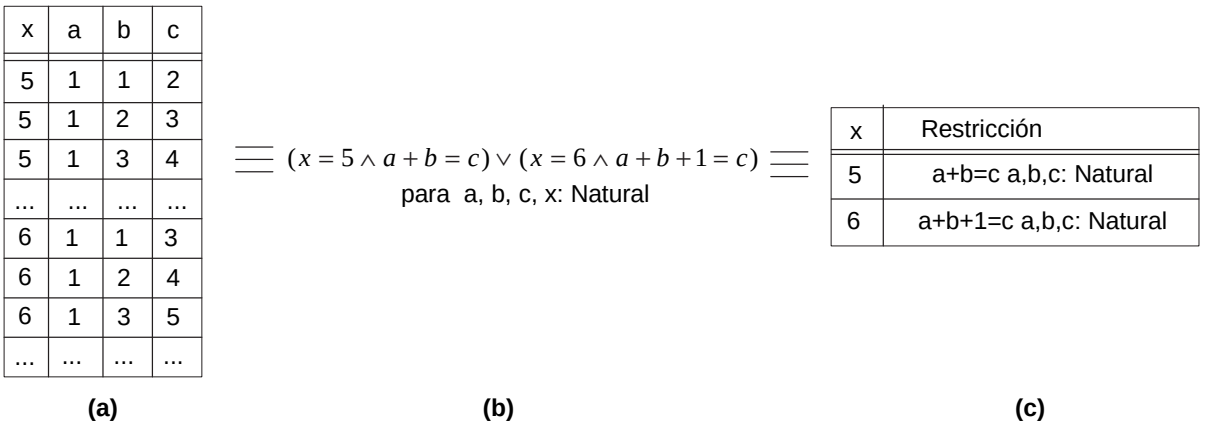

Figura 4.1: Equivalencia entre una relación extendida, una relación restrictiva y una relación con atributos clásicos y restricciones

La nueva definición de las BDdR radica en la característica de dualidad de los datos, extensivos e intensivos, para que ambos tipos puedan formar parte de una misma relación. Aquellos que pueden ser representados como atributos clásicos no deben participar en la restricción, mientras que los que toman un número de valores muy grandes o incluso infinitos serán representados mediante variables pertenecientes a restricciones. Esta diferencia viene promovida porque el tratamiento de los datos intensivos, pese a su capacidad expresiva, es más complejo que en el caso de los extensivos. Utilizando las dos formas de representación será posible minimizar el tamaño de la base de datos haciendo posible que la información sea completa, a la vez que se pueden aprovechar todas las ventajas del álgebra relacional respecto a los atributos que toman un único valor, manteniendo la primera forma normal que establece que los dominios de los atributos deben ser atómicos [32].

Una relación de una BDdR está formada por dos tipos de atributos: atributos clásicos, que tienen una relación de igualdad con un valor determinado, por lo que el atributo es a su vez una variable (como x en el ejemplo de la

figura 4.1.c); y atributos que describen la relación entre un conjunto de variables mediante una fórmula (como el atributo *Restricción* de dicha figura). El atributo *Restricción* representa a su vez una relación, pero en este caso una relación restrictiva.

El nombre de la variable que tenga una relación de igualdad con una constante, corresponderá al nombre del atributo, mientras que el nombre del atributo restricción tendrá una relación semántica con las variables, no sintáctica. Por ejemplo, si se representa mediante una restricción el recorrido de un vehículo utilizando las variables x, y y t, se le puede dar el nombre al atributo *Recorrido*, que es independiente de las variables que lo representan. Las variables que forman una restricción no pueden formar parte de los atributos clásicos de dicha relación, por ejemplo la variable x del atributo *Recorrido*, no puede ser a su vez un atributo clásico llamado x.

Esta nueva definición de equivalencias entre relaciones restrictivas y relaciones con atributos clásicos y restricciones, se origina para encontrar un nivel intermedio entre expresividad y optimización en la evaluación de consultas, donde no toda la información sea representada de forma extensiva, ni toda de forma intensiva como una restricción. Esta es la misma filosofía que originó las Bases de Datos Objeto-Relacionales, para poder almacenar datos más complejos sin perder las ventajas de los avances en el área de las bases de datos relacionales.

4.2. Formalización

Para redefinir las BDdR, junto a las diferentes operaciones sobre las mismas, es necesario reformular algunas de las definiciones dadas en el capítulo 3 relativas a *restricción*, *k-tupla restricción*, *relación restrictiva* y *Bases de Datos de Restricciones*.

4.2.1. Restricción

Una restricción es una relación entre variables definidas sobre un dominio [42], donde existe una limitación sobre el espacio de posibilidades o valores que pueden tomar dichas variables. El tipo Restricción definido en esta tesis será: Restricción numérica definida como una combinación conjuntiva de ecuaciones e inecuaciones, lineales y polinómicas.

La definición de las restricciones se basa en la propuesta desarrollada en el capítulo 2 de [130] con algunas variantes, utilizando una gramática donde se definen las variables y constantes de un dominio (Natural, Flotante o Entero). Una restricción para la nueva definición de DBdR será de la forma:

```
restricción := restricción_atómica '∧' restricción
      | restricción_atómica
      ;
```

Donde una restricción_atómica es:

```
restricción_atómica :=expresión PREDICADO expresión
      ;
expresión := término OPERADOR_BINARIO expresión
      | término
      | '−'término
      ;
término := Variable//representando el atributo variable de restricción
      | Constante
      ;
```

De forma general, los predicados y los símbolos de los operadores son:

```
PREDICADO := '=' | '<' | '≤' | '>' | '≥'
      ;
OPERADOR_BINARIO := '+' | '−' | '*'
      ;
```

La redefinición de las Bases de Datos de Restricciones que aquí se propone se basa en cuatro tipos de restricciones, también analizadas en [57]: restricciones lineales de igualdad, polinómicas de igualdad, inecuaciones lineales e inecuaciones polinómicas. Dependiendo del tipo de restricción se utilizarán técnicas y algoritmos diferentes para un tratamiento más eficiente, por lo que es necesario reformular la definición de `PREDICADO`, `expresión` y `OPERADOR_BINARIO` para detectar a cuál de estos cuatro tipos pertenece:

```
PREDICADO := PREDICADO_ECUACIÓN
      | PREDICADO_INECUACIÓN
      ;
PREDICADO_ECUACIÓN := '='
      ;
PREDICADO_INECUACIÓN := '<' | '≤' | '>' | '≥'
      ;
```

Entrando en los detalles para diferenciar las restricciones lineales y polinómicas:

```
expresión := expresión_Lineal
      | expresión_Polinómica
      ;
expresión_Lineal := término_Lineal SÍMBOLO_LINEAL expresión_Lineal
      | término_Lineal
      | '−'término_Lineal
      ;
expresión_Polinómica := término OPERADOR_BINARIO expresión_Polinómica
      | término
      | '−'término
      ;
término := Variable
      | Constante
      ;
```

Donde los término y los símbolos para las restricciones lineales son:

```
SÍMBOLO_LINEAL := '+' | '−'
      ;
término_Lineal := Constantes '*' Variable '*' Constantes
      | Constantes '*' Variable
      | Variable '*' Constantes
      | Variable
      ;
Constantes := Constante '*' Constantes
      | Constante
      ;
```

```
Constante := ℝ | ℕ | ℤ
      ;
Variable := ℝ | ℕ | ℤ
      ;
```

Basadas en estos tipos de expresiones y predicados, se definen los cuatro tipos de restricciones soportadas:

- **Restricciones lineales de Igualdad:** Las que están definidas sobre `expresión_Lineal` y `PREDICADO_ECUACIÓN`
- **Restricciones sobre inecuaciones lineales:** Las que están definidas sobre `expresión_Lineal` y `PREDICADO_INECUACIÓN`
- **Restricciones polinómicas de igualdad:** Las que están definidas sobre `expresión_Polinómica` y `PREDICADO_ECUACIÓN`
- **Restricciones sobre inecuaciones polinómicas:** Las que están definidas sobre `expresión_Polinómica` y `PREDICADO_INECUACIÓN`

4.2.2. Bases de Datos de Restricciones

Las aportaciones a las BDdR en esta tesis se apoyan en dos aspectos, uno de ellos es la redefinición de sus características acercándolas a las bases de datos relacionales clásicas aprovechando todos los avances en este área. Por otra parte, se hacen propuestas relativas a la implementación basadas en el modelo teórico, un aspecto que se ha dejado en algunas ocasiones en un segundo plano en las soluciones existentes. En este capítulo se analiza esta nueva definición, mientras que los detalles de implementación serán abordados en los siguientes capítulos.

4.2.3. Nueva Definición del Modelo de Bases de Datos de Restricciones

El nuevo modelo de BDdR sigue una definición paralela a la original de las BDdR (definición 3.8 del capítulo 3), pero en este caso diferenciando las partes de la restricción

que se representan mediante igualdad entre variables y constantes (atributos clásicos o univaluados), y la información representada mediante restricciones (atributos restricción).

- Una *k-tupla restricción* con las variables $x_1, \ldots, x_k$ es una conjunción finita de la forma $\varphi_1 \wedge \ldots \wedge \varphi_N$ donde cada φ_i, para $1 \leq i \leq N$, es una restricción de la forma $\{x_j$=*Constante*$\}$ donde $x_j \in \{x_1, \ldots, x_k\}$ a los que se denominará *Atributo Clásico o Univaluado*, o una relación representada mediante una restricción, siguiendo la gramática de la sección 4.2.1, entre las variables $x_1, \ldots, x_k$ no pertenecientes a un atributo univaluado, a los que se denominará *Atributo Restricción*.

- Una *relación restrictiva* está definida como un conjunto de *Atributos Univaluados* y un conjunto de *Atributos Restricción*. Una *relación restrictiva de aridad k*, es un conjunto finito $r = \{\psi_1, \ldots, \psi_M\}$, donde cada ψ_j para $1 \leq j \leq M$ es una k-tupla restricción sobre las variables $x_1, \ldots, x_k$. La fórmula correspondiente es la disjunción de la forma $\psi_1 \vee \ldots \vee \psi_M$, tal que $\psi_j = \varphi_1 \wedge \ldots \wedge \varphi_N$ donde cada φ_i para $1 \leq i \leq N$ es una *k-tupla restricción*. Si existe en cada $\psi_j \in r$ una φ_i de la forma $\{x$=*Constante*$\}$, donde x sea la misma variable en todos los φ_i pertenecientes a distintos ψ_j y x no aparece en el resto de los φ_i de una misma ψ_j, se dirá que x es un **atributo univaluado**, mientas que el resto de variables formarán parte en uno o varios **Atributos Restricción**.

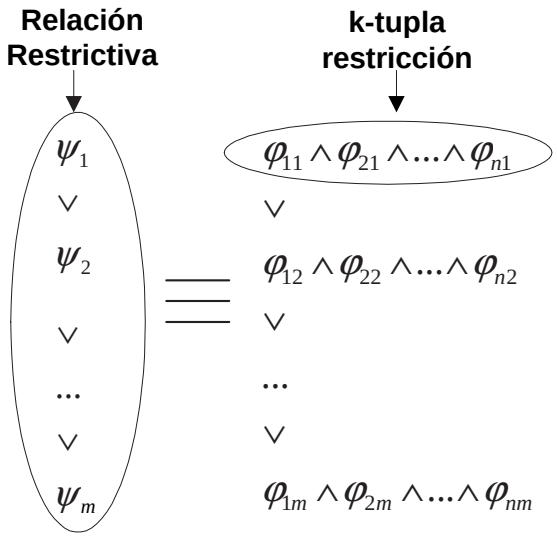

Figura 4.2: Representación de k-tuplas restricción y relaciones restrictivas

En la figura 4.2 se presenta la equivalencia entre una relación restrictiva y las k-tuplas restrición que la forman. Para que en dicha relación existan tanto atributos univaluados como atributos restricción, tiene que cumplirse que:

Siendo φ_{ij} una $\varphi_i \in \varphi_1 \wedge \ldots \wedge \varphi_N$ y $\psi_j \in \psi_1 \vee \ldots \vee \psi_M$, tal que $\psi_j = \varphi_{1j} \wedge \ldots \wedge \varphi_{Nj}$, una relación restrictiva tendrá atributos univaluados (x) si:

$\exists \varphi_{ij} \forall j \in 1..M \mid i \in 1..N$, $\{\varphi_{ij} \equiv x = c_j\}$ donde c_j es una constante

$\wedge\ \forall t \in 1..N \wedge t \neq i \wedge \varphi_{tj}(x_1, \ldots, x_k) \wedge x \notin \{x_1, \ldots, x_k\}$

Lo que significa que si en todas las k-tuplas restricción existe una relación de igualdad entre una variable y una constante (la misma variable en todas las tuplas), y dicha variable no aparece en el resto de la k-tupla restricción, se dirá que dicha variable es un *atributo univaluado* o *atributo clásico*, ya que corresponde con la definición de atributo en el álgebra relacional de Codd.

Un ejemplo es el presentado en la figura 4.3, donde la relación de la Base de Datos de Restricciones está formada por un atributo restricción y dos atributos univaluados. En el ejemplo existen dos variables x e y que aparecen en todas las tuplas con una relación de igualdad con una constante, y que no participan en el resto de la relación, por lo que son atributos univaluados.

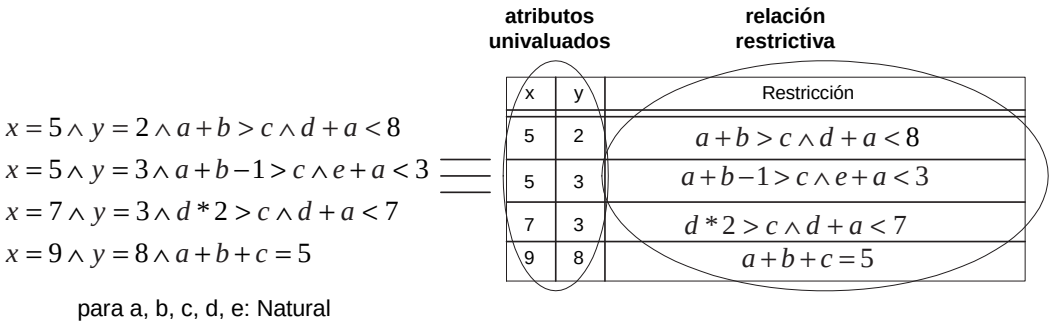

Figura 4.3: Ejemplo de representación de k-tuplas restricción y relaciones restrictivas

- Por lo tanto, una *Base de Datos de Restricciones* es una colección finita de relaciones restrictivas formadas por **atributos univaluados** y **atributos restricción**.

Un aspecto que aún queda abierto es definir si todas las restricciones irán en un mismo atributo restricción o en diferentes. Aunque a lo largo de este trabajo se tratará este tema

otra vez, ahora es oportuno hacer algunas puntualizaciones. Una de las razones por la cual un conjunto de restricciones debe pertenecer al mismo atributo restricción puede ser por las variables que las componen. Un ejemplo puede ser describir características de personas como nombre, edad, ... (atributos univaluados), junto a la situación geográfica de la parcela que posee sobre las variables (x, y), que representan la latitud y longitud de los distintos puntos que la forman. En este caso se utilizará un atributo restricción para almacenar todas las parcelas, ya que todas las restricciones están definidas sobre las mismas variables y describen una misma característica. Sin embargo existen otros dos casos de análisis relativos a este tema:

- **Restricciones definidas sobre las mismas variables pero que describen distintas características**. Un ejemplo puede ser: sean dos tipos de restricciones, ambas sobre (x, y), una que describe las parcelas y otra las casas. Pese a estar definidas sobre las mismas variables, describen dos características de las posesiones de la persona y desde el punto de vista de las bases de datos relacionales clásicas, no sería conveniente almacenarlas en un mismo atributo ya que no tienen relación semántica.
- **Restricciones definidas sobre distintas variables pero que describen la misma característica**. Un ejemplo de esto es el que precisamente ocurre en la diagnosis de fallos basada en modelos, que es uno de los casos de estudio analizados en esta tesis. En la diagnosis basada en modelos cada componente puede ser descrito mediante una restricción, que estará definida sobre variables que no tiene que ser iguales a las variables de las restricciones. De hecho, lo más usual es que los componentes tengan distintas entradas y salidas para conformar los sistemas, pero que a nivel semántico las restricciones describen una misma característica: el comportamiento de los componentes. En este caso las restricciones de los componentes pertenecerán a un mismo atributo.

4.2.4. Atributos en las Bases de Datos de Restricciones

Derivado de las definiciones anteriores, aparece una nueva clasificación de atributos que permitirá crear, actualizar y consultar las BDdR en concordancia con sus características. En esta nueva definición existen tres tipos distintos de atributos:

- **Atributo Clásico** ($\texttt{at}_i$): Representa un atributo perteneciente a una k-tupla de una relación, donde at_i es de un tipo permitido por el estándar SQL, como por ejemplo *Numérico*, *Cadena*, *Fecha* ... Este tipo de atributo es el definido como atributo univaluado.

- **Atributo Restricción** ($\texttt{at}_i^c$): Representa un atributo perteneciente a una k-tupla de una relación, donde at_i^c es una restricción (atributo restricción), por lo que a su vez describe una tupla de variables. Dicho atributo restricción tendrá un nombre, al igual que los atributos univaluados, pero no relacionado con la sintaxis de las variables, sino con su semántica.

- **Atributo Variable de Restricción** ($\texttt{at}_i^c.\texttt{v}_j$): Representa un atributo que será una variable $\text{v}_j \in \{v_1, \ldots, v_k\}$ que participa en el atributo restricción at_i^c. No es necesario que la variable v_j pertenezca a todas las restricciones de las distintas tuplas que forman la relación, pero sí al menos a una. Por ejemplo en la figura 4.4 se presenta de manera extensiva la relación de la figura 4.3 donde se puede hacer referencia a un atributo univaluado (x o y), a una restricción completa (*Asociación*), a una variable que está en todas las restricciones (*Asociación.a*), o a una variable que sólo está en una restricción del atributo restricción (*Asociación.e*).

4.3. Consultas a Bases de Datos de Restricciones

Al tener nuevos tipos distintos de atributos, ahora las consultas a las BDdR serán más complejas. Al igual que se presentó en el capítulo 3, será necesaria la eliminación de cuantificadores para inferir nuevo conocimiento derivado de las restricciones almacenadas, junto a nuevas estrategias que resuelvan la evaluación de consultas para estos nuevos tipos de atributos.

Antes de comenzar con la redefinición del álgebra relacional para las nuevas BDdR, es conveniente comparar algunas propiedades de las bases de datos clásicas y su relación con las BDdR:

- Cada instancia de los atributos tienen un único valor, a lo que se le denomina dominio atómico. Esto no ocurre en las BDdR a nivel de atributo restricción, ya que el atributo de una relación puede ser una restricción que represente muchos valores

x	y	Asociación
5	2	a b c d 1 1 1 6 2 1 1 5 3 2 4 3 4 2 4 3 ... ...
5	3	a b c e 1 2 1 2 2 1 1 0 1 3 2 1 2 2 2 0 ... ...
7	3	a c d 1 1 5 1 1 6 1 3 2 2 2 2 ... ...
9	8	a b c 1 1 3 1 2 2 1 3 1 2 1 2 ... ...

Figura 4.4: Representación extendida de la relación de la figura 4.3

para un conjunto de variables. Sin embargo analizado desde el punto de vista de que un atributo restricción es a su vez una relación de variables, se podría definir como una referencia a una relación. Esto mantendría conceptualmente la primera forma normal, a la vez que permite la inclusión del atributo restricción.

- Los valores de los atributos tienen que pertenecer a un determinado dominio. Sobre esto es necesario hacer alguna aclaración para la nueva definición de las BDdR, ya que un atributo restricción puede estar definido sobre distintas variables, aunque sí es necesario que todas las variables de un mismo atributo restricción estén definidas sobre el mismo dominio (Naturales, Enteros o Flotantes).

- Cada relación tiene un nombre diferente para poder diferenciar una de otra, lo que

también ocurre en las BDdR, a lo que en la teoría de las BDdR se le llama esquema de la base de datos.

- Cada atributo de una relación tiene un nombre único dentro de dicha relación, lo que también ocurre en las BDdR. La diferencia es que sí es posible que la misma variable pertenezca a atributos restricción diferentes, de forma que dos restricciones puedan definir relaciones sobre la misma variable.

- Cada tupla es diferente, no pudiendo estar duplicadas. Esto también tiene algunos detalles en las BDdR, donde hay que evitar que dos restricciones que estén representadas de una manera sintácticamente diferente no representen el mismo conjunto de valores.

- El orden de los atributos y de las tuplas no es importante, ni en las bases de datos relacionales ni en las BDdR.

Para definir las consultas es necesario apoyarse en las cinco operaciones primitivas del álgebra relacional, con las cuales se pueden definir las restantes. Dichas operaciones son: selección (σ), proyección (π), producto cartesiano ($\times$), unión de conjuntos ($\cup$) y diferencia de conjuntos ($-$). Además del tipo de operación, hay que tener en cuenta qué tipos de atributos están involucrados en dicha operación, lo que dependerá de cada caso.

- **Operador de Selección.** Su signatura es $R_1 = \sigma_{predicado}(R_2)$, donde R_1 y R_2 son relaciones de una base de datos, y el *predicado* una condición que involucra atributos de la relación R_2. La relación R_1 es un subconjunto horizontal de la relación R_2 que representa aquellas tuplas que satisfacen la condición del predicado. Un predicado es una expresión booleana cuyos componentes son operadores lógicos $\chi = \{\wedge, \vee\}$ y operadores de comparación $\theta = \{<, \leq, >, \geq, =, <>\}$. Los operandos de estos operadores son atributos o constantes pertenecientes al dominio del atributo. Esto significa que el predicado tendrá la forma:

  $$a_1\theta c_1 \ \chi \ a_2\theta c_2 \ \chi \ \ldots \ \chi \ a_n \ \theta \ c_n$$

  donde a_i representa un atributo (de cualquier tipo de los tres propuestos) y c_i constantes del mismo tipo que a_i u otro atributo. También se pueden definir las condiciones mediante negaciones $\neg$, de la forma $\neg(a_i\theta c_i)$.

  Cuando nos referimos a esta operación en las BDdR, es necesario replantear la parte del *predicado* del operador de selección, ya que podría hacer referencia a

atributos clásicos, atributos restricción o a atributos variable de restricción. Los detalles relativos a las BDdR se muestran en la sección 4.5.

- **Operador de Proyección.** Su signatura es $R_1 = \pi_{<a_1,\ldots,a_n>}(R_2)$, donde R_1 y R_2 son relaciones y $a_1, \ldots, a_n$ son atributos de la relación R_2. La relación R_1 es un subconjunto vertical de la relación R_2, donde sólo aparecen los atributos a_1, ..., a_n y sin tuplas duplicadas.

  Para esta operación en las BDdR, es necesario replantear la obtención de los atributos, ya que si se pretende obtener un conjunto de atributos del tipo variable de restricción, sería necesario eliminar de la solución aquellas variables de las restricciones que no aparecieran en la proyección. Los detalles relativos a la proyección para las BDdR se muestran en la Sección 4.6.

- **Producto Cartesiano.** Su signatura es $R_3 = R_1 \times R_2$, donde la relación R_3 define la unión horizontal (con el operador $\wedge$) de cada tupla de la relación R_1 con cada tupla de la relación R_2. De esta forma, la operación une horizontalmente dos relaciones obteniendo una nueva con todos los pares posibles entre las tuplas de R_1 y R_2. Si una relación tiene I tuplas y N atributos y la otra tiene J tuplas y M atributos, la relación R_3 tendrá $I * J$ tuplas (cardinalidad) con $N + M$ atributos (grado). Para este operador no es necesario describir una nueva semántica operacional para los atributos restricción ni variables de restricción, ya que los operandos de esta operación son relaciones, lo que no permite diferenciar entre los distintos tipos de atributos. Su descripción es igual a la definida en el álgebra relacional clásica pero con atributos que pueden ser restricciones.

- **Unión de Conjuntos.** Su signatura es $R_3 = R_1 \cup R_2$, donde se define la relación R_3 como la unión vertical de todas las tuplas de R_1 y R_2, sin tuplas duplicadas. R_1 y R_2 tienen que ser compatibles con respecto a la unión, lo que significa que tienen que tener el mismo grado, y que para todos los atributos de R_1, su atributo i-ésimo tiene que ser del mismo tipo que el atributo i-ésimo de R_2.

  Pese a que al igual que en el producto cartesiano, los parámetros de entrada de la unión de conjuntos son relaciones, sí es interesante destacar que será necesario el análisis de la relación de salida. Como se ha comentado, la unión de conjuntos no permite tuplas duplicadas, así que será necesario cerciorarse de que dos tuplas con atributos restricción no representan el mismo conjunto de valores, o que los valores

de una de las tuplas no están incluidos en otra. Los detalles relativos a las BDdR se muestran en la sección 4.7.

- **Diferencia de Conjuntos.** Su signatura es $R_3 = R_1 - R_2$, donde R_3 es una relación formada por las tuplas de R_1 que no pertenecen a R_2. Las relaciones R_1 y R_2 también tienen que ser compatibles en el mismo aspecto que en la unión. También para esta operación, en el caso de los atributos que sean restricciones, será necesario analizar cuándo una restricción es o no igual a otra, lo que se estudia en la sección 4.8.

4.4. Resolución de Problemas con Restricciones

Antes de comenzar la exposición sobre cómo evaluar las consultas para las distintas primitivas presentadas, es necesario introducir dos tipos de técnicas para resolución de problemas: los problemas de satisfacción y optimización de restricciones; y la eliminación y sustitución de variable mediante técnicas simbólicas. La utilización de estas familias de técnicas dependerá del tipo de consulta que se realice.

- **Problemas de Satisfacción u Optimización de Restricciones:** Esto es necesario ya que la evaluación de las consultas relacionadas con algunos de estos operadores se basan en la construcción y resolución de problemas de satisfacción de restricciones. Los problemas de satisfacción de restricciones ayudarán a: evaluar si se cumple el predicado de la operación de selección; obtener valores de las variables en la operación de proyección; y conocer si dos restricciones comparten las mismas soluciones en las operaciones de unión y diferencia de conjuntos. Un ejemplo para la selección podría ser obtener todas las tuplas de una relación donde las soluciones de un atributo restricción At estén incluidas en otra restricción C. En este caso se construirá un problema de satisfacción de restricciones con cada restricción del atributo At y C para conocer si existe alguna solución de At que no sea solución de C.

  En el caso de los problemas de optimización de restricciones, se puede encontrar un ejemplo en la proyección para obtener el mayor o el menor valor que puede tomar una variable contenida en las restricciones de un atributo restricción para una determinada relación.

- **Eliminación o Sustitución de Variables:** Estas técnicas se utilizarán en el operador de proyección, para obtener de manera simbólica nuevas restricciones eliminando un conjunto de las variables de las restricciones almacenadas en la base de datos. Este tipo de técnicas se basa en la sustitución de variables, apoyándose en las Bases de Gröbner, y la eliminación de variables utilizando la Descomposición Algebraica Cilíndrica.

4.4.1. Problemas de Satisfacción de Restricciones

Muchos problemas de ingeniería han sido modelados mediante problemas de satisfacción de restricciones [56][111][37]. Un problema de satisfacción de restricciones (Constraint Satisfaction Problem - CSP) es una representación de un conjunto de variables, dominios y restricciones, ligado a uno o varios esquema de razonamiento para resolverlo. Formalmente, está definido por la tripleta <X, D, C> donde, $\mathrm{X} = \{x_1, x_2, \ldots, x_n\}$ es un conjunto finito de variables, $\mathrm{D} = \{d(x_1), d(x_2), \ldots, d(x_n)\}$ es el conjunto de dominios de cada una de las variables, y $\mathrm{C} = \{\mathrm{C}_1, \mathrm{C}_2, \ldots, \mathrm{C}_m\}$ un conjunto de restricciones. Cada restricción C_i está definida como una relación R de un conjunto de variables $\mathrm{V} = \{x_i, x_j, \ldots, x_k\}$ a lo que se denomina *ámbito de la restricción*. R puede ser representada como un subconjunto de productos cartesianos de la forma $d(x_i) \times d(x_j) \times \ldots \times d(x_k)$, una restricción $\mathrm{C}_i = (\mathrm{V}_i, \mathrm{R}_i)$ limita los valores simultáneos que pueden tomar las variables para que satisfagan las restricciones donde están involucradas. Sea $\mathrm{V}_k = \{x_{k_1}, x_{k_2}, \ldots, x_{k_l}\}$ un subconjunto de X, una l-tupla $(x_{k_1}, x_{k_2}, \ldots, x_{k_l})$ de $d(x_{k_1}), d(x_{k_2}), \ldots, d(x_{k_l})$ se denomina *instanciación* de variables en V_k. Una instanciación es *consistente* si son satisfactibles todas sus restricciones para los valores de V_k. Una instanciación consistente de todas las variables en X es una *solución*, por lo que el objetivo de la resolución de un CSP consiste en encontrar una o más de una *solución*. Aunque la resolución o evaluación de un CSP también puede ser *verdadero* o *falso* cuando sólo se quiere conocer si un conjunto de valores de variables es una solución o no del problema.

Un CSP puede ser resuelto mediante diferentes técnicas de búsqueda y consistencia, como algoritmos de vuelta atrás (backtracking) [62][15]. Sin embargo, cuando un CSP tiene un dominio continuo, no se puede garantizar que se encuentre una solución utilizando únicamente búsqueda, ya que no es posible obtener todos los valores de las variables cuando su dominio es muy grande, pudiendo llegar a ser infinito. De todas formas, los CSP con dominios infinitos pueden ser resueltos aplicando relajación de consistencias antes o

durante la búsqueda, de manera que se realiza eliminación local de valores inconsistentes para las variables del dominio [14][102]. En teoría, si un CSP tiene una única solución, aplicando un determinado grado de consistencia se puede reducir el dominio de las variables a un solo valor; si un CSP tiene más de una solución, o incluso infinitas, todos los valores que permanezcan en el dominio después de aplicar la consistencia serán parte de la solución. En la práctica, una resolución eficiente de un CSP consiste en aplicar varios niveles de consistencia como la arco consistencia [103][112][113] a cada nodo del árbol de búsqueda. Normalmente la resolución de problemas de satisfacción de restricciones se realiza utilizando combinaciones de búsqueda y técnicas de consistencia. Esta búsqueda también se ha analizado con técnicas de búsqueda local, con el objetivo de utilizar técnicas más eficientes [122][121][29]. La búsqueda exhaustiva no siempre trabaja bien con problemas complejos, ni siquiera incluyendo técnicas de ramificación y acotación o vuelta atrás más inteligentes. Para los problemas de gran tamaño, en algunas ocasiones es más eficiente usar búsqueda local donde no se analizan todas las soluciones, pero se pueden encontrar algunas en un tiempo menor.

4.4.2. Problemas de Optimización con Restricciones

Un problema de optimización con restricciones (Constraint Optimization Problem - COP) es un problema de satisfacción de restricciones definido por<X, D, C> junto a una función objetivo *f* que debe ser optimizada:

f: $\{d(x_1) \times d(x_2) \times \ldots \times d(x_n)\} \rightarrow \mathcal{S}_t$ donde $(\mathcal{S}_t, \leq)$ definen el orden total.

Para un problema de optimización de restricciones que pretende maximizar un objetivo, $\mathcal{A}$ es una solución para el COP si y sólo si $\mathcal{A}$ es una solución del CSP y no existe un $\mathcal{A}'$ para el cual $f(\mathcal{A}') > f(\mathcal{A})$. De esta forma sólo las soluciones del CSP que maximicen o minimicen el objetivo, serán consideradas soluciones del COP [38][76].

4.4.3. Bases de Gröbner

Las Bases de Gröbner se basan en la sustitución simbólica de variables para restricciones polinómicas de igualdad. La teoría de las bases de Gröbner, desarrollada por Buchberger [23], fue el origen de muchos de los algoritmos de sustitución simbólica para

múltiples variables en polinomios de igualdad. Dicha teoría se plantea como una generalización del método de eliminación de Gauss para múltiples variables en ecuaciones lineales, y el algoritmo de Euclides para ecuaciones polinómicas de una variable.

Mediante las bases de Gröbner se transforma un conjunto de restricciones polinómicas en otro conjunto equivalente que tiene las mismas soluciones que el original, pero eliminado unas determinadas variables. Partiendo de un conjunto de polinomios de igualdad de la forma $P = 0$, y las variables que se desean eliminar, se obtiene un sistema equivalente $(G = 0)$ pero sin dichas variables.

Un ejemplo de obtención de las bases de Gröbner puede ser: siendo el conjunto de polinomios $\{a + b = c, d + e = f, c * f = g\}$, eliminar las variables $\{c, f\}$. En esta caso se obtendrá la nueva restricción:

$$\{f = (a + b) * (d + e)\}$$

Poder obtener nuevas restricciones dependerá de las variables que se pretenden eliminar, ya que no todas las posibilidades son matemáticamente viables. Sólo será posible eliminar variables que pueden ser sustituidas por otras que sí forman parte de la solución. Por ejemplo, para el sistema de ecuaciones anterior no podría ser eliminada la variable$a$.

El algoritmo de Buchberger fue el primero para la obtención de las Bases de Gröbner Bases, aunque a éste lo siguieron algunas mejoras [49][48].

En general, la complejidad de los algoritmos para la obtención de las Bases de Gröbner es alta, exponencial en función del número de variables involucradas en el peor de los casos.

4.4.4. Descomposición Algebraica Cilíndrica

En el caso de restricciones representadas mediante inecuaciones polinómicas, la eliminación de variables se basa en la Descomposición Algebraica Cilíndrica. Al igual que las bases de Gröbner, la Descomposición Algebraica Cilíndrica es una técnica simbólica de eliminación de variables, pero en este caso permite trabajar con inecuaciones polinómicas dentro del dominio de los reales. Ésta es una técnica clásicamente usada en el estudio de problemas de propiedades topológicas y algebraicas, aunque es posible encontrar más detalles sobre otras utilidades de esta técnica [5][6]. En esta tesis, la aportación de la Descomposición Algebraica Cilíndrica radica en la eliminación de las variables no pertenecientes

a una sentencia con el operador de proyección cuando la sustitución simbólica no es suficiente.

La Descomposición Algebraica Cilíndrica de $\mathbb{R}^k$ es una secuencia $S_1 \ldots S_k$ donde, para cada $1 \leq \mathrm{i} \leq \mathrm{k}$, S_i es una partición finita de $\mathbb{R}^i$ en subconjuntos semi-algebraicos, llamados *celdas de nivel i*. La Descomposición Algebraica Cilíndrica devolverá inecuaciones cuyos bordes implican a funciones algebraicas [22][126]. Un ejemplo es el mostrado en la figura 4.5, que muestra las restricciones:

$(x^2 + y^2 < 1)$ y $((x+1)^2 + y^2 < 1)$.

La primera restricción representa una circunferencia centrada en $(0, 0)$ de radio 1, y la segunda otra circunferencia también de radio 1 pero centrada en -1.

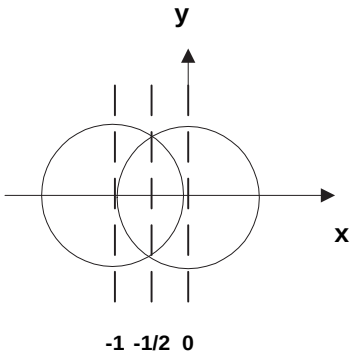

Figura 4.5: Ejemplo de Descomposición Algebraica Cilíndrica

Al realizar la Descomposición Algebraica Cilíndrica sobre la variable x, se crearán las celdas mostradas con líneas discontinuas marcadas es dicha figura. Dichas divisiones se generan según los valores de x donde ambas restricciones se cortan entre si, lo que significará que son los puntos donde hay que analizar su comportamiento conjunto. En este caso se obtendrá la restricción:

$$-1 < x < -1/2 \wedge -\sqrt{1 - x^2} < y < \sqrt{1 - x^2}$$

$$\vee$$

$$-1/2 < x < 0 \wedge -\sqrt{-2 - x^2} < y < \sqrt{-2x - x^2}$$

La cual describe el área que forma la intersección entre ellas. La eliminación de la variable y significará obtener sólo la parte de la restricción que contiene a x, obteniendo finalmente:

$$-1 < x < 0$$

El mayor problema de la Descomposición Algebraica Cilíndrica es la complejidad, ya que el algoritmo para realizarla es doblemente exponencial en función del número de celdas. Algunos trabajos han analizado dicha complejidad para mejorar los tiempos de eliminación de cuantificadores [35][144]. Otra limitación es la ya comentada sobre que no se puede asegurar la eliminación de variables para restricciones con variables del dominio de los enteros o los naturales.

4.5. Operador Selección para BDdR

Como se ha expuesto anteriormente, el operador selección obtiene un subconjunto horizontal de las tuplas de la relación de entrada en función de las condiciones que se definen en un predicado. En el predicado para la selección sobre las BDdR pueden aparecer los tres tipos de atributos:

- **Atributo Clásico o Univaluado** ($\mathtt{at}$): Para atributos de tipo clásico o univaluados no es necesario redefinir el operador de selección, ni a nivel sintáctico ni semántico, ya que describirá la misma semántica operacional que en el álgebra relacional clásica.
- **Atributo Restricción** ($\mathtt{at}_i^c$): Este tipo de atributo representa una columna de la relación que es de tipo restricción, lo que significa que el tipo de los atributos del predicado serán del tipo atributo restricción. En este caso es necesario redefinir las operaciones de comparación $<$, $\leq$, $>$, $\geq$, $=$, $<>$ para restricciones numéricas. Este análisis se presenta en la subsección 4.5.1 de este capítulo, junto a la ampliación del predicado con nuevas operaciones de comparación entre restricciones.
- **Atributo Variable de Restricción** ($\mathtt{at}_i^c.\mathtt{v}_j$): Este tipo de atributo representa a una variable de un atributo restricción, lo que necesita ser estudiado. Los predicados con este tipo de atributo involucrado se analizarán con detalle en la subsección 4.5.2

4.5.1. Operador de Selección para Atributos de tipo Restricción

En esta subsección se analiza el operador de selección cuando en el predicado hay involucrados atributos de tipo restricción de una determinada relación. Para esto es necesario conocer qué tipos de condiciones se pueden establecer sobre el tipo restricción. La

comparación entre atributos restricción que se proponen hacen referencia a la relación entre las soluciones que pueden tomar las dos restricciones involucradas en la comparación. Entre los operadores, se definirán los analizados en el capítulo 2 de [107] con respecto a que las soluciones que hacen satisfactible una restricción están incluidas en las soluciones de otra ($\subset$, $\subseteq$, $\supset$, $\supseteq$), o que tienen al menos una solución en común (&). Además se definirán, para atributos restricción, la semántica de los operadores de comparación ya existentes en el álgebra relacional clásica (θ =($<$, $\leq$, $>$, $\geq$, $=$, $<>$)). Estos operadores informarán de si todas las soluciones de una restricción son mayores, menores, iguales o distintas de las soluciones de otra restricción. Un ejemplo de selección para atributos de tipo restricción puede ser:

$$R' = \sigma_{(C_x \subseteq a^2+b^2<1)}(R)$$

donde C_x representa el nombre de un atributo restricción de la relación R. La relación de salida R' estará formada por el subconjunto de las tuplas de R donde todas las soluciones que hacen satisfactible las restricciones almacenadas en el atributo restricciónC_x también sean solución de la restricción $\{a^2 + b^2 < 1\}$.

Para conocer todas las tuplas de R que cumplen la condición, se tendrán que construir problemas de satisfacción de restricciones con cada una de las restricciones del atributo $\{R.C_x\}$ y la restricción del predicado $\{a^2 + b^2 < 1\}$, con el objetivo de conocer qué tuplas formarán parte de R'. Sólo podrán cumplir la condición aquellas restricciones que tengan las mismas variables que la restricción del predicado, a y b para este ejemplo.

Antes de comenzar con las operaciones de comparación sobre restricciones, es necesario incluir una nueva operación sobre variables, la igualdad sintáctica entre variables de distintas restricciones:

Definición de Igualdad Sintáctica: Sean C_x y C_y dos restricciones donde $X = \{x_1, x_2, \ldots, x_n\}$ representa las variables de C_x, e $Y = \{y_1, y_2, \ldots, y_m\}$ las variables de C_y. Si $x_i \in X$, e $y_j \in Y$, entonces $x_i \equiv_{\mathcal{S}} y_j$ es cierto si ambas variables son sintácticamente iguales, o lo que es lo mismo, tienen el mismo nombre. Que dos variables pertenecientes a distintas restricciones sean sintácticamente iguales, no implica que sean la misma variable, o lo que es lo mismo, que los valores que puedan tomar sean los mismos. Esto es debido a que al pertenecer a distintas restricciones, las soluciones que hacen satisfactible una o otra restricción no tienen porqué ser las mismas. Dependiendo de la condición que relacione ambas restricciones, se establecerá o no esta igualdad sintáctica como igualdad semántica. Por ejemplo, sea una relación con un atributo restricción donde cada tupla

representa el recorrido de un vehículo, todas las restricciones estarán definidas sobre las variables x, y y t. Pese a esto, los valores de dichas variables que hacen satisfactible cada restricción no tienen que ser los mismos, lo que significa que son sintácticamente iguales pero no semánticamente iguales. Sin embargo, si se estableciera como condición obtener los vehículos que se encuentran en su recorrido en un punto $(x,\ y)$ para un instante de tiempo t, se estaría definiendo la igualdad semántica entre dichas variables, ya que necesariamente tienen que compartir soluciones.

Definición de las operaciones de comparación permitidas entre restricciones:

- $C_x < C_y$ es cierto si para todas las $x_i \equiv_{\mathcal{S}} y_j$, el máximo valor que puede tomar x_i siempre es menor que el mínimo valor que puede tomar y_j, y para todas las variables de C_x existe una sintácticamente igual en C_y y viceversa. Esta definición es extensible a $C_x > C_y$.

  Para conocer si $C_x < C_y$, se construye un CSP donde todas las variables x_i de C_x, que también pertenezcan a C_y (y_j) de forma que $(x_i \equiv_{\mathcal{S}} y_j)$, serán renombradas como x'_i construyendo una nueva restricción C'_x. Esto es necesario ya que las soluciones de ambas restricciones tienen que poder ser tomar distintos valores con respecto a las variables compartidas, y si se mantienen los mismos nombres en el CSP, nunca podrá ocurrir que los valores que tome x_i en C_x sean distintos a los que toma en C_y. Para conocer si todas las soluciones de C_x para las variables $x_i \equiv_{\mathcal{S}} y_j$ son menores que las soluciones que hacen satisfactible C_y, se construye un CSP donde se buscará si existe una solución donde esto no ocurra. Si se encuentra alguna solución, entonces la salida de la evaluación del predicado $C_x < C_y$ será *falso*, y *cierto* en caso contrario. Por lo tanto la fórmula:

  $$\forall x_i \in X,\ \forall y_j \in Y \mid x_i \equiv_{\mathcal{S}} y_j \Rightarrow (x_i < y_j)$$

  será transformada en:

  $$\neg(\exists x_i \in X,\ \exists y_j \in Y \mid x_i \equiv_{\mathcal{S}} y_j \Rightarrow x_i > y_j \vee x_i = y_j)$$

  Esta trasformación evita tener que analizar todo el espacio de búsqueda de las soluciones del CSP, sólo teniendo que resolver un CSP con el objetivo de encontrar

un valor donde se cumple que $x_i \geq y_j$. El CSP se construye de la forma:

$X' = \{x'_1,\ x'_2,\ \ldots,\ x'_n\}$
$Y = \{y_1,\ y_2,\ \ldots,\ y_n\}$
C'_x
C_y
$\forall x_i \in X\ \forall y_j \in Y \mid x_i \equiv_{\mathcal{S}} y_j \Rightarrow$ `añadir al CSP` $\{(x'_i > y_j \vee x'_i = y_j)\}$

Si existe alguna solución para el CSP, entonces $C_x < C_y$ devolverá *falso*, porque significará que no todos los valores de las variables de C_x son menores que en C_y ya que es posible encontrar un valor que cumple lo contrario. Si no existe ninguna solución para el CSP, la comparación $C_x < C_y$ devolverá *cierto*. Un ejemplo es el mostrado en la figura 4.6, donde $C_x < C_y$ es cierto.

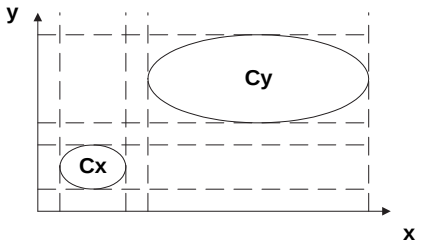

Figura 4.6: Ejemplo donde $C_x < C_y$

Este tipo de selección puede ser utilizada en las consultas sobre sistemas de información geográfica, por ejemplo para conocer qué parcelas están al suroeste de otra.

- $C_x \leq C_y$ es cierto si para todas las variables $x_i \equiv_{\mathcal{S}} y_j$, el máximo valor de x_i siempre es menor o igual que el menor valor que puede tomar y_j. Esta definición puede ser extendida a $C_x \geq C_y$. La construcción del CSP es muy similar a la del caso anterior:

$X' = \{x'_1,\ x'_2,\ \ldots,\ x'_n\}$
$Y = \{y_1,\ y_2,\ \ldots,\ y_m\}$
C'_x
C_y
$\forall x_i \in X\ \forall y_j \in Y \mid x_i \equiv_{\mathcal{S}} y_j \Rightarrow$ `añadir al CSP` $\{(x'_i > y_j)\}$

Si existe una solución al CSP, entonces $C_x \leq C_y$ devolverá *falso*, y *cierto* en caso contrario.

- $C_x = C_y$ es cierto si todas las soluciones de C_x también lo son de C_y y viceversa. En este caso no simplemente consiste en comparar ambas restricciones sintácticamente, ya que se puede representar el mismo conjunto de valores con dos restricciones morfológicamente diferentes. En la construcción del CSP no se renombran las variables, ya que precisamente lo que se busca es conocer si todas las soluciones que hacen satisfactible a C_x también hacen satisfactible a C_y. En este caso, se creará el siguiente CSP:

  $X = \{x_1,\ x_2,\ \ldots,\ x_n\}$
  $Y = \{y_1,\ y_2,\ \ldots,\ y_n\}$
  $(C_x \wedge \neg C_y) \vee (C_y \wedge \neg C_x)$

  Si existe alguna solución para el CSP, entonces C_x=C_y devolverá *falso*, y *cierto* en caso contrario. Para el operador <>, que describe si dos restricciones no son iguales, se puede utilizar esta misma forma de construir el CSP. La diferencia radica que si existe alguna solución devolverá *cierto*, y *falso* en caso contrario.

  Aquí no acaba la complicación de estos operadores de comparación, ya que la operación de negación para los reales tiene una problemática añadida. En los problemas de satisfacción de restricciones sobre reales, el número de soluciones es muy grande, teóricamente infinita, lo que hace que la búsqueda de soluciones sea muy costosa. Esto ha generado soluciones que abordan el problema mediante aproximaciones con una determinada precision utilizando intervalos que contienen las soluciones. Esta utilización de intervalos provoca que una restricción y su negación compartan soluciones en la práctica, aunque de forma teórica no tendrían que compartirlas [12][64]. Para evitar esto, la creación de CSP que se propone en esta tesis se apoya en la reescritura de la negación de las restricciones, cuando éstas están definidas sobre reales, construyendo una nueva restricción. Para obtener de manera práctica, no exclusivamente teórica la operación de negación, será necesario que dicha negación no tome los valores derivados de la aproximación, construyendo la negación con la separación de una constante ε. El valor de ε dependerá de la aproximación que se haga, siendo de un orden mayor que la aproximación que utilice el resolutor de CSP, por ejemplo si la aproximación llega hasta reales de 10^{-4}, ε será 10^{-3}. La reescritura para una restricción de forma $\texttt{término}_1\ \theta\ \texttt{término}_2$, donde θ puede ser uno de los operadores $<$, $\leq$, $>$, $\geq$ o $=$, será de la forma:

  - $\texttt{término}_1 < \texttt{término}_2 \Rightarrow \texttt{término}_1 \geq \texttt{término}_2 + \varepsilon$

- $\texttt{término}_1 \leq \texttt{término}_2 \Rightarrow \texttt{término}_1 > \texttt{término}_2 + \varepsilon$
- $\texttt{término}_1 > \texttt{término}_2 \Rightarrow \texttt{término}_1 + \varepsilon \leq \texttt{término}_2$
- $\texttt{término}_1 \geq \texttt{término}_2 \Rightarrow \texttt{término}_1 + \varepsilon < \texttt{término}_2$
- $\texttt{término}_1 = \texttt{término}_2 \Rightarrow \texttt{término}_1 + \varepsilon < \texttt{término}_2 \vee \texttt{término}_1 - \varepsilon > \texttt{término}_2$

En la figura 4.7 se muestra gráficamente un ejemplo donde partiendo de la restricción $(x-5.5)^2+(y-5.5)^2 \leq 4$, se construye su negación obteniendo la nueva restricción $(x-5.5)^2+(y-5.5)^2 > 4+10^{-3}$.

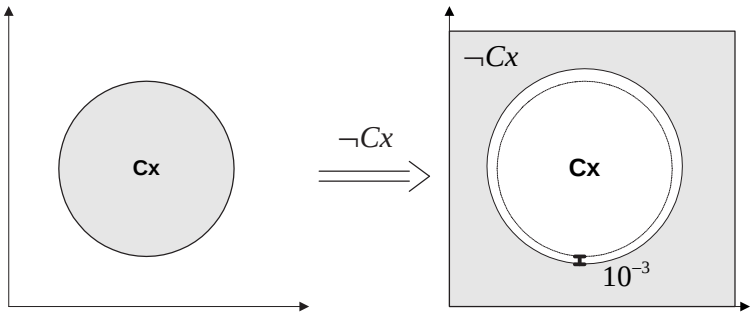

Figura 4.7: Ejemplo de reescritura de la negación

Sólo cabe aclarar que si las restricciones están definidas sobre el dominio de los enteros o los naturales, no será necesario incluir el factor de aproximación ε en la negación de las mismas.

- C_x & C_y será cierto si existe alguna solución que haga satisfactible ambas restricciones, siendo todas las variables sintácticamente iguales entre ambas. En este caso el CSP será de la forma:

$X = \{x_1, x_2, \ldots, x_n\}$
C_x
C_y

Si se encuentra una solución para este CSP, la operación C_x & C_y devolverá *cierto*, y *falso* si no encuentra solución.

- $C_x < (C_y$ FOR $\{y_d, \ldots, y_h\})$ es una variante para la operación $<$ que devolverá *cierto* si para todas las variables $x_i \equiv_{\mathcal{S}} y_j$ tal que $y_j \in \{y_d, \ldots, y_h\}$ el máximo valor de x_i es menor que el mínimo valor de y_j. Esta operación realiza una proyección

sobre las variables que aparecen en el predicado $\{y_d, \ldots, y_h\}$, lo que conlleva que las variables de C_x que no pertenecen a $\{y_d, \ldots, y_h\}$ no son relevantes en esta operación. En este caso, las restricciones C_x y C_y no tienen que tener todas las variables sintácticamente iguales entre ellas, y sólo se incluirán comparaciones entre aquellas variables que aparezcan en la condición. El CSP que se construye es de la forma:

$X' = \{x'_1,\ x_2,\ \ldots,\ x'_n\}$
$Y = \{y_1,\ y_2,\ \ldots,\ y_m\}$
C'_x
C_y
$\forall x_i \in X\ \ \forall y_j \in \{y_d, \ldots, y_h\}\ \mid\ x_i \equiv_{\mathcal{S}} y_j$
$\Rightarrow$ añadir al CSP $\{(x'_i > y_j\ \vee\ x'_i = y_j)\}$

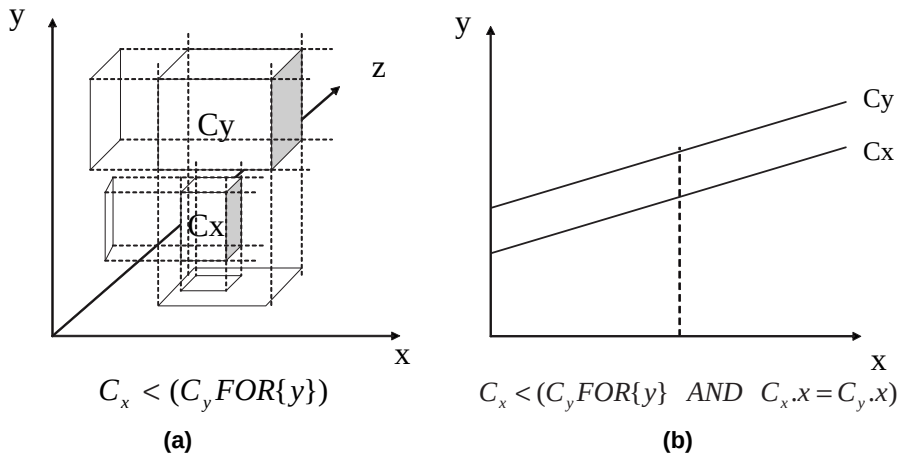

Figura 4.8: Ejemplo de proyección sobre variables

Si existe alguna solución para el CSP, la comparación devolverá *falso*, y *cierto* en caso contrario. Un ejemplo para esto es el que se muestra en la figura 4.8.a donde la proyección se hace sobre la variable y, y el predicado es $C_x{<}(C_y$ FOR $\{y\})$. Dicho predicado pregunta si todos los valores de las soluciones de y que hacen satisfactible C_x son menores que los que hacen satisfactible C_y, independientemente de los valores de las variables $\{x, z\}$. Sin embargo, para el predicado $C_x{<}(C_y$ FOR $\{x\})$ el mismo ejemplo devolvería falso. Esta definición es extensible a las operaciones $\leq$, $>$, $\geq$ y &.

Este tipo selección puede ser utilizada en las consultas sobre sistemas de información geográfica, por ejemplo para conocer si una nube está encima de una región.

Otra variante de este tipo de comparación es la que se basa en conocer si las soluciones de una restricción son menores que las de otra, para un valor establecido de una o varias variables. En la figura 4.8.b se muestra un ejemplo, donde para los valores de x donde C_x y C_y son satisfatibles, siempre el valor de y que hace satisfactible C_x es menor que el valor de y que hace satisfactible C_y. La condición para este ejemplo es:

C_x < (C_y FOR {y} AND C_x.x=C_y.x)

De forma que la sintaxis genérica para establecer igualdad semántica será:

C_x < (C_y FOR {y_d,...,y_h} AND C_x.$\mathtt{v}_1$=C_y.$\mathtt{v}_1$ AND ... AND C_x.$\mathtt{v}_i$=C_y.$\mathtt{v}_i$

AND ... AND C_x.$\mathtt{v}_k$=C_y.$\mathtt{v}_k$), donde $\mathtt{v}_i \notin \{y_d,\ldots,y_h\}$

Lo que significará que se construirá y resolverá un CSP de la forma:

$X' = \{x'_1,\ x_2,\ \ldots,\ x'_n\}$
$Y = \{y_1,\ y_2,\ \ldots,\ y_m\}$
C'_x
C_y
$\forall x_i \in X\ \forall y_j \in \{y_d,\ldots,y_h\} \mid x_i \equiv_{\mathcal{S}} y_j$
  $\Rightarrow$ añadir al CSP $\{(x'_i > y_j \vee x'_i = y_j)\}$
$\forall v \in \{v_1,\ldots,v_k\} \mid v \equiv_{\mathcal{S}} x'_i \wedge v \equiv_{\mathcal{S}} y_j$
  $\Rightarrow$ añadir al CSP $\{x'_i = y_j\}$

- $C_x \subseteq C_y$ será cierto si todas las soluciones de C_x lo son también de C_y. Para la inclusión, es necesario que las restricciones estén definidas sobre las mismas variables, de forma que sean C_x y C_y dos restricciones donde $X = \{x_1,\ x_2,\ \ldots,\ x_n\}$ son las variables de C_x y C_y, $C_x \subseteq C_y$ es igual que la implicación $(C_x \rightarrow C_y)$ [107]. Esta operación determina si todas las soluciones de C_x son también soluciones de C_y, aunque es posible que C_y tenga soluciones que no pertenezcan a C_x.

  Para evitar analizar todas las soluciones de C_x, comprobando que éstas también lo son de C_y, se buscará una solución donde esto no ocurra, de forma que la evaluación de la condición $C_x \subseteq C_y$ corresponde con la fórmula:

  $\neg(\exists_{x_i \in X}(C_x \wedge \neg C_y))$

Y el CSP que se creará es:

$$X = \{x_1, x_2, \ldots, x_n\}$$
$$C_x \wedge \neg C_y$$

Si se encuentra alguna solución para el CSP devolverá *falso*, y *cierto* si no encuentra ninguna. En este caso también es necesario la misma reescritura de la negación de restricciones que se ha explicado para la operación '$\models$'. En la figura 4.9 hay un ejemplo donde la restricción C_x está incluida en la restricción C_y. Esta definición puede ser extendida a $C_x \supseteq C_y$.

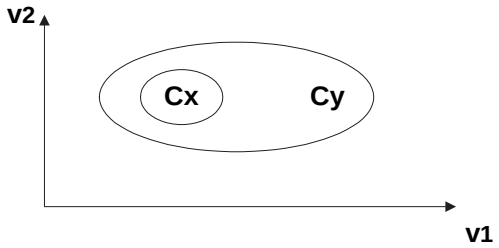

Figura 4.9: Ejemplo de $C_x \subseteq C_y$

- $C_x \subset C_y$ devolverá *cierto* si todas las soluciones de C_x también son de C_y, y además obligatoriamente C_y tiene al menos una solución que no pertenecen a C_x. Como el operador anterior, ambas restricciones tienen que estar definidas sobre las mismas variables.

  En este caso, se construye un CSP como el anterior para saber que $C_x \subseteq C_y$, y otro para saber que existe una solución para C_y que no pertenece a C_x:

  $$X = \{x_1, x_2, \ldots, x_n\}$$
  $$C_y \wedge \neg C_x$$

  Si existe alguna solución para el CSP, entonces $C_x \subset C_y$ devolverá *cierto* y *falso* en otro caso. Esta definición es extensible a $C_x \supset C_y$.

4.5.2. Operador de Selección para Atributos de tipo Variable de Restricción

Los atributos variable de restricción también pueden participar en el predicado de una selección. El atributo $at_i^c.\mathrm{v}_j$ se podrá comparar con una constante u otro atributo que

pertenezca al mismo dominio que la variable v_j, lo que implica dos tipos de comparación:

- Sea x_i una variable perteneciente a las variables $X = \{x_1, x_2, \ldots, x_n\}$ de un atributo restricción C_x y $\{C_x.x_i\ \theta\ c\}$ una parte del predicado de la selección, donde c es un valor del dominio de x_i, y $\theta = \{<, >, \leq, \geq, =\}$. Se cumplirá la condición para aquellas restricciones donde la variable x_i puede tomar valores relativos a c en función de θ. Para realizar dicha selección, se proponen dos alternativas que se analizarán más en profundidad en el siguiente capítulo. Una posibilidad es crear un CSP, o ampliar el que se construya con el resto del predicado, incluyendo la restricción $\{x_i\ \theta\ c\}$. Si el CSP encuentra alguna solución, significará que la variable puede tomar algún valor relativo a c que cumpla la condición. La otra opción es almacenar en la base de datos los valores máximos y mínimos para cada variable en cada restricción, de manera que en la selección primero se comprobará si el valor c puede cumplir la condición, en función del máximo y el mínimo de la variable $C_x.x_i$ y lo establecido por θ. Si esto es posible, se construirá un CSP para asegurar que realmente existe un valor que cumple la condición. Si no está dentro de dichos límites, se podrá asegurar que la restricción C_x no cumple la condición del predicado.

- Sea x_i una variable que pertenece a las variables $X = \{x_1, x_2, \ldots, x_n\}$ de la restricción C_x, y $\{C_x.x_i\ \theta\ C_y.y_j\}$ perteneciente al predicado de la selección, donde y_j es una variable del atributo restricción C_y. La variable y_j puede ser sintácticamente igual o diferente a $C_x.x_i$, aunque obligatoriamente del mismo dominio, y $\theta = \{<, >, \leq, \geq, =\}$. Se seleccionarán aquellas tuplas donde la variable x_i de C_x puede tomar los valores relacionados con la variable y_j de C_y en función de θ. Una vez seleccionadas la tuplas con restricciones cuyas variables son las que aparecen en el predicado, será necesario crear o modificar el CSP que se construye para la selección, por ejemplo con la restricción $\{C_x.x_i\ \theta\ C_y.y_j\}$. Si el CSP encuentra una solución, significará que las variables x_i e y_j pueden tomas los valores cuya relación viene determinada por θ.

  Un ejemplo de selección con variables de restricción en el predicado puede ser $\sigma_{Relacin.c>Relacin.d}(R)$, donde R es la relación de la figura 4.10. Como salida de la selección sólo se obtendrá la primera tupla $\{x = 5 \wedge y = 2 \wedge a + b > c \wedge d + a < 8\}$ de la relación. Esto es debido a que en la segunda y en la cuarta tupla no aparece d, por lo que no se puede definir la relación entre variables, y en la tercera aparece $d > c * c + 1$, que nunca cumplirá la condición.

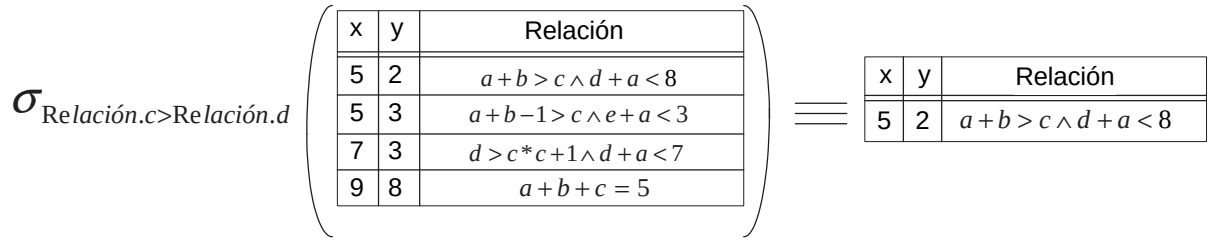

Figura 4.10: Ejemplo de selección con condición de atributos variable de restricción

4.6. Operador Proyección para BDdR

El operador de proyección realiza una selección vertical de una relación, devolviendo un subconjunto de los atributos que la conforman. Como en la nueva definición de las BDdR existen tres tipos de atributos, existirán tres tipos de selección vertical o proyección:

- Para atributos univaluados o clásicos (at) no es necesario redefinir el operador de proyección, ni a nivel sintáctico ni semántico, ya que es el descrito en el álgebra relacional clásica.
- Para atributos restricción (at_i^c), pese a que a su vez representan una columna que es una relación, no conllevará mayor problema ya que será obtener las restricciones tal como están almacenadas. El único proceso que se tendría que realizar sería comprobar que no existen tuplas duplicadas, como se explicará en la sección 4.7 para la unión de conjuntos que tiene la misma problemática.
- Los atributos variable de restricción ($at_i^c.\mathrm{v}_j$), hacen referencia a una columna de una columna de la relación. Esta proyección se analizará en la subsección 4.6.1, ya que la evaluación de este atributo para el operador de proyección necesita profundizar en diferentes aspectos.

4.6.1. Operador de Proyección para Atributos Variables de Restricción

Cuando se quieren obtener las variables de un atributo restricción de una relación, es posible obtener dos tipos de información. Una de las posibilidades es obtener los valores extensivos (o numéricos) de las variables (como se muestra en la figura 4.4), y la otra

es obtener una representación intensiva (o simbólica), mediante restricciones que describa todos los valores que pueden tomar las variables involucradas en la proyección. La sentencia para la proyección sobre variables de restricción será de la forma:

$\pi_{(at_i^c.v_j,\ldots,at_k^c.v_l)}(R)$

Y la evaluación de dicha operación puede obtener información intensiva o extensiva:

- **Información Intensiva:** En este caso, se obtendrán nuevas restricciones que describen el conjunto de valores que pueden tomar las variables $(at_i^c.v_j, \ldots, at_k^c.v_l)$.

  La forma de la sentencia será:

  $\pi_{CONSTRAINTS(at_i^c.v_j,\ldots,at_k^c.v_l)}(R)$

  donde la palabra *CONSTRAINTS* se utiliza para denotar que se quiere obtener una representación simbólica de los valores de las variables $(at_i^c.v_j, \ldots, at_k^c.v_l)$.

  Estas restricciones, equivalentes a los valores extensivos de las variables, se pueden obtener utilizando diferentes técnicas, sustitución simbólica de variables, o eliminación de cuantificadores:

  - **Sustitución simbólica de variables:** Esta técnica se puede utilizar para restricciones definidas sobre cualquier dominio de variables, ya que consisten en eliminar un conjunto de variables sustituyéndolas en función de otras pero sin perder información. La limitación de la sustitución simbólica viene dada porque sólo está definida para restricciones de igualdad, tanto lineales como polinómicas, ya que hace uso de la teoría de las Bases de Gröbner presentada en la sección 4.4.3.
  - **Eliminación simbólica de cuantificadores:** Existen casos de consultas sobre BDdR donde para obtener un subconjunto de variables de forma intensiva no es suficiente con la sustitución simbólica, bien por el tipo de las restricciones involucradas, o porque no existan suficientes variables para sustituir las no deseadas. En estos casos, será necesario recurrir a la eliminación de cuantificadores. La eliminación de los cuantificadores ($\exists$ y $\forall$) [101][152] se basa en la Descomposición Algebraica Cilíndrica [110] orientada a la proyección sobre aquellas variables que formarán parte de la solución, tal como se presenta en

la sección 4.4.4. Desafortunadamente, el peor caso de complejidad para la eliminación de cuantificadores reales es doblemente exponencial en función del número de bloques de cuantificadores [153][39][27].

Los tipos de restricciones con los que se trabaja en esta tesis, sobre las cuales se puede asegurar que se obtendrá una solución a la eliminación de cuantificadores se presentan en la tabla 4.1. En dicha tabla se puede observar que sólo para las soluciones sobre el dominio de los reales es posible asegurar dicha eliminación, por lo que es posible que la proyección devuelva una relación vacía, para mantener la propiedad de cierre, cuando dicha eliminación no sea posible.

Restricción	Dominio	Tiene eliminación de cuantificadores
Lineal	Real	Sí
Lineal	Natural	No
Lineal	Entero	No
Polinómica	Real	Sí
Polinómica	Natural	No
Polinómica	Entero	No

Tabla 4.1: Eliminación de Cuantificadores en función del tipo de restricción y dominio

- **Información Extensiva:** Cuando se realiza la proyección sobre las variables, también se ofrecerá la posibilidad de obtener los valores extensivos de las variables, no su representación simbólica. Este caso también es de proyección en función de las variables, pero no será necesario eliminar cuantificadores, ya que la salida será extensiva. Para obtener dichos valores se construirá un CSP con todas las restricciones y variables involucradas, aunque sólo se mostrará como evaluación de la proyección aquellas variables definidas en la sentencia del operador de proyección.

  La forma de la sentencia será:

  $$\pi_{VALUES[N](at_i^c.v_j,\ldots,at_k^c.v_l)}(R)$$

  donde de utiliza la palabra reservada *VALUES* para describir que se quieren obtener valores instaciados de las variables, y N representa el número de tuplas que se quieren obtener, ya que puede ser que las posibilidades sean infinitas. Si no se establece en valor de N, por defecto se presentará una sola tupla.

Siendo C_1, ..., C_n las restricciones de la relación R involucradas en la proyección, el CSP que se creará para obtener el valor de las variables será de la forma:

```
añadir al CSP {X_1 = {x_{1,1}, x_{1,2}, ..., x_{1,m} ∈ C_1}}
...
añadir al CSP {X_n = {x_{n,1}, x_{n,2}, ..., x_{n,m} ∈ C_n}}
C_1 ∨ ... ∨ C_n
```

Cuando un problema de satisfacción de restricciones no sólo pretende encontrar una solución, sino que quiere encontrar la mejor solución en función de un objetivo definido, se utilizará un problema de optimización de restricciones. Esto tiene relación con la bases de datos clásicas, ya que en las bases de datos relacionales, no sólo es posible obtener todas los valores de un conjunto de atributos, también es posible extender el álgebra añadiendo agregados como $COUNT$, SUM, AVG, MAX y MIN que permite realizar a su vez operaciones con dichos atributos. Para el caso de las variables de restricciones, en esta tesis se amplían la utilización de las operaciones MAX y MIN. Estas operaciones en las bases de datos relacionales se pueden realizar sobre atributos numéricos, por lo que también se podrán utilizar para variables de restricciones que son también atributos numéricos, pudiendo obtener sus máximos y mínimos valores. Para obtener la evaluación de la proyección será necesario crear un COP con las restricciones relacionadas con la variable a maximizar o minimizar. Sean C_1, C_2, ..., C_n las restricciones de la relación involucradas en la proyección, sea X_i las variables de C_i, y x la variable a minimizar o maximizar, entonces el COP se construye de la forma:

```
X_1 = {x_{1,1}, x_{1,2}, ..., x_{1,m} ∈ C_1}
...
X_n = {x_{n,1}, x_{n,2}, ..., x_{n,m} ∈ C_n}
Maximizar(x) ∨ Minimizar(x)
C_1 ∨ C_2 ∨ ... ∨ C_n//para las C_i donde x ∈ X_i
```

La figura 4.11 resume lo presentado en esta sección, donde se muestra la división del tipo de proyección en función de la salida que se quiere obtener (intensiva o extensiva). En el caso de la extensiva o numérica, se construirán problemas de satisfacción u optimización de restricciones para obtener valores concretos de las variables. En el caso de la salida intensiva, representado mediante restricciones, se eliminarán las variables no deseadas.

Si dichas variables pueden ser sustituidas por otras y las restricciones son polinómicas o lineales de igualdad, se recurrirá a la obtención de las Bases de Gröbner. Si esto no es posible, se necesitará eliminar cuantificadores utilizando la Descomposición Algebraica Cilíndrica, cuya salida sólo se puede asegurar para restricciones definidas sobre el dominio de los reales. En el caso de la proyección extensiva, se obtendrán tuplas con los valores concretos que pueden tomar las variables, como atributos univaluados. En el caso de la obtención extensiva no existen limitaciones con respecto al tipo de restricción o dominio de sus variables.

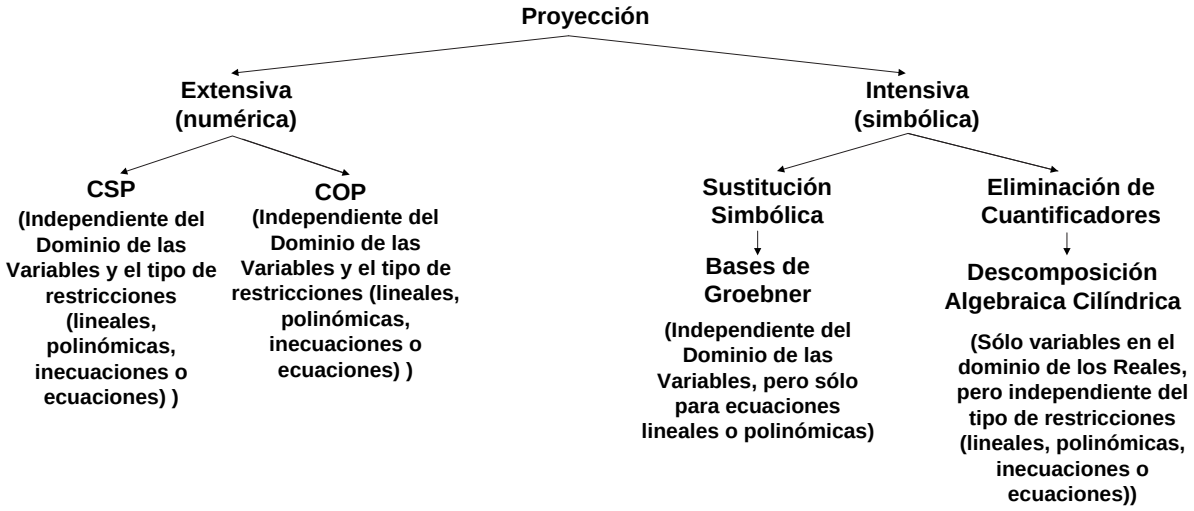

Figura 4.11: Tipos de Proyección en función de la salida

4.7. Operador de Unión de Conjuntos para BDdR

Otra de las operaciones primitivas del álgebra relacional es la unión de conjuntos ($R_1 \cup R_2$), la cual obtiene para dos relaciones de entrada, la unión vertical de las tuplas de ambas sin repetición. La problemática de este operador radica en que una tupla es una restricción definida como una combinación conjuntiva de atributos univaluados y atributos restricción. En función de las soluciones que compartan dichas tuplas, se obtendrán distintas relaciones de salida. Analizando la unión entre dos restricciones definidas sobre las mismas variables, se pueden encontrar los cuatro casos, como se muestra en la figura 4.12, donde C_x y C_y representan las restricciones:

- Si $C_x \nsubseteq C_y$ y $C_y \nsubseteq C_x$, entonces la unión de ambas restricciones será una nueva restricción $\{C_x \vee C_y\}$. Esto es lo que se muestra en las figuras 4.12.a y 4.12.b.
- Si se cumple que $C_x \subseteq C_y$ o su equivalente $C_y \subseteq C_x$, entonces la unión de ambas restricciones será $\{C_y\}$ o $\{C_x\}$ respectivamente. Esto es lo que se muestra en las figuras 4.12.c y 4.12.d.

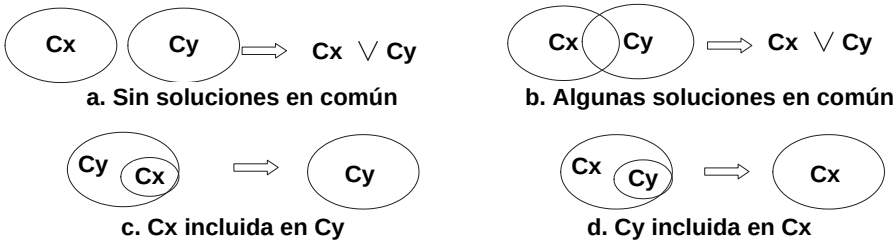

Figura 4.12: Tipos de Unión entre restricciones

Para obtener la relación de salida para la operación de unión, hay que tener en cuenta que las relaciones pueden estar formadas por varios atributos clásicos y restricción. De esta forma, se compara cada tupla $t_1 \in R_1$, con cada tupla $t_2 \in R_2$, analizando si dichas tuplas son iguales o comparten soluciones. Sean las tuplas t_1 y t_2, donde x_i representa atributos univaluados, $const_i$ los valores de los atributos univaluados, y C_i atributos restricción:

$t_1 \equiv \{x_1 = const_1 \wedge \ldots \wedge x_n = const_n \wedge C_1 \wedge \ldots \wedge C_m\}$

$t_2 \equiv \{x_1 = const'_1 \wedge \ldots \wedge x_n = const'_n \wedge C'_1 \wedge \ldots \wedge C'_m\}$

Si $\exists_{x_i} \mid const_i \neq const'_i \Rightarrow t_1 \cup t_2 \equiv t_1 \vee t_2$

En otro caso:

Si $\exists_{C_i} \mid C_i \nsubseteq C'_i \vee \exists_{C'_i} \mid C'_i \nsubseteq C_i$

$\Rightarrow t_1 \cup t_2 \equiv t_1 \vee t_2$ (Figuras 4.12.a y 4.12.b)

En otro caso: $(\forall_{C_i, C'_i}\ (C_i \subseteq C'_i \vee C'_i \subseteq C_i))$

$\Rightarrow t_1 \cup t_2 \equiv \{x_1 = const_1 \wedge \ldots \wedge x_n = const_n \wedge C''_1 \wedge \ldots \wedge C''_m\}$

Si $C'_i \subseteq C_i \Rightarrow C''_i \equiv C_i$ (Figura 4.12.d)

En otro caso: $(C_i \subseteq C'_i) \Rightarrow C''_i \equiv C'_i$ (Figura 4.12.c)

4.8. Operador Diferencia de Conjuntos para BDdR

Por último, queda por analizar la diferencia de conjuntos ($R_1 - R_2$), la cual obtiene para dos relaciones de entrada las tuplas de R_1 que no pertenecen a R_2. La problemática de este operador también radica en que una tupla es una restricción definida como una combinación conjuntiva de atributos univaluados y atributos restricción. En función de las soluciones que compartan dichas tuplas, se obtendrán distintas relaciones de salida. Analizando la diferencia entre dos restricciones, se pueden diferenciar cuatro casos, presentados en la figura 4.13, donde C_x y C_y representan restricciones:

- Si C_x y C_y no tienen ninguna solución en común, entonces la diferencia de las restricciones será C_x, como se presenta en la figura 4.13.a.
- Si se cumple que C_x y C_y tienen alguna solución en común pero C_x tiene soluciones que no pertenecen a C_y, entonces la diferencia de las restricciones será $\{C_x \wedge \neg C_y\}$. Estos casos son los presentados en las figuras 4.13.b y 4.13.c.
- Si se cumple que $C_x \subseteq C_y$, entonces la diferencia entre las restricciones será la restricción vacía, como se muestra en la figura 4.13.d.

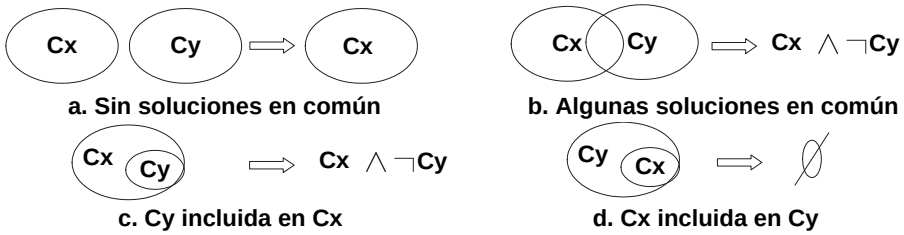

Figura 4.13: Tipos de Diferencias entre restricciones

La negación de las restricciones también conllevará la reescritura de las mismas en los términos explicados en la sección 4.5.1.

Para obtener la relación de salida para la operación de diferencia, hay que tener en cuenta que las relaciones pueden estar formadas por varios atributos clásicos y restricción. De esta forma, se compara cada tupla $t_1 \in R_1$, con cada tupla $t_2 \in R_2$, analizando si dichas tuplas son iguales o comparten soluciones. Sean las tuplas t_1 y t_2, donde x_i

representa atributos univaluados, $const_i$ los valores de los atributos univaluados, y C_i atributos restricción:

$t_1 \equiv \{x_1 = const_1 \wedge \ldots \wedge x_n = const_n \wedge C_1 \wedge \ldots \wedge C_m\}$

$t_2 \equiv \{x_1 = const'_1 \wedge \ldots \wedge x_n = const'_n \wedge C'_1 \wedge \ldots \wedge C'_m\}$

Si $\exists_{x_i} \mid const_i \neq const'_i$

$\Rightarrow t_1 - t_2 \equiv t_1$

En otro caso:

Si $\exists_{C_i} \mid C_i \wedge \neg C'_i$

$\Rightarrow t_1 - t_2 \equiv \{x_1 = const_1 \wedge \ldots \wedge x_n = const_n \wedge C''_1 \wedge \ldots \wedge C''_m\}$

Si $\exists$ una solución de $C_i \mid C_i \wedge C'_i \Rightarrow C''_i \equiv C_i \wedge \neg C'_i$ (Figuras 4.13.b y 4.13.c)

En otro caso: $C''_i \equiv C_i$ (Figura 4.13.a)

En otro caso: $(\forall_{C_i}\ C_i \subseteq C'_i)$:

$t_1 - t_2 \equiv \phi$ //(Figura 4.13.d) t_1 no pertenece a la relación de salida

4.9. Resumen

En este capítulo se reformula la definición de Bases de Datos de Restricciones para diferenciar los atributos clásicos o univaluados y los atributos restricción. Esta división genera tres diferentes tipos de atributos: atributo clásico, atributo restricción y atributo variable de restricción. Esto hace necesario que las operaciones primitivas del álgebra relacional tengan que ampliar tanto su sintaxis como su semántica. Las operaciones que se revisan en este capítulo son:

- **La selección**, ya que la condición del predicado puede incluir restricciones y variables de restricciones.
- **La proyección**, ya que se podrá realizar proyección sobre las variables de los atributos restricción, tanto de forma extensiva como intensiva.
- **La unión y diferencia de conjuntos**, siendo necesario analizar las relaciones entre las tuplas formadas por atributos clásicos y atributos restricción.

Estas operaciones se pueden redefinir gracias a dos familias de técnicas: por una parte, la construcción y resolución de problemas de satisfacción y optimización de restricciones para el tratamiento numérico de las restricciones; y por otra, la sustitución de variables y eliminación de cuantificadores para el tratamiento simbólico de la información. En la tabla 4.2 se presenta un resumen del tratamiento de los distintos tipos de atributos en función de las distintas operaciones. En dicha tabla se refleja cuándo se utiliza el álgebra relacional clásica (AR), cuándo es necesario crear y resolver un CSP o un COP, o en qué circunstancias hay que tratar de forma simbólica las restricciones.

<table>
<tr><th></th><th>at_i</th><th>at_i^c</th><th colspan="3">$at_i^c.v_j$</th></tr>
<tr><td>σ</td><td>Igual que AR</td><td>Creación de CSP</td><td colspan="3">Creación de CSP</td></tr>
<tr><td>π</td><td>Igual que AR</td><td>Igual que AR</td><td>Creación de CSP</td><td>Creación de COP</td><td>Tratamiento Simbólico</td></tr>
<tr><td>$\times$</td><td>Igual que AR</td><td>Igual que AR</td><td colspan="3">No aplicable a esta Operación</td></tr>
<tr><td>$\cup$</td><td>Igual que AR</td><td>Creación de CSP</td><td colspan="3">No aplicable a esta Operación</td></tr>
<tr><td>$-$</td><td>Igual que AR</td><td>Creación de CSP</td><td colspan="3">No aplicable a esta Operación</td></tr>
</table>

Tabla 4.2: Resumen de las Operaciones y Técnicas para evaluar consultas

Capítulo 5

Arquitectura LORCDB. Un Gestor para BDdR

En base al nuevo concepto de Base de Datos de Restricciones introducido en el capítulo anterior, en este capítulo se especifica una arquitectura que permite dar soporte a las características descritas. La utilización de las BDdR ha ampliado sin duda la capacidad expresiva de los gestores de bases de datos, pero a su vez también complica el tratamiento de la información, ya que su representación es intensiva y se necesitan técnicas de inferencia diferentes a las propias de las bases de datos extensivas. Esto hace necesario desarrollar una arquitectura que facilite dicho procesamiento, obteniendo un gestor de base de datos con capacidad expresiva pero con métodos de evaluación de consultas computacionalmente eficientes. Para el tratamiento de datos de tipo restricción será necesario definir un nuevo lenguaje de consulta, que se presenta como una ampliación de la gramática de SQL. El nuevo lenguaje mantendrá toda la funcionalidad del estándar, junto a la posibilidad de realizar las mismas operaciones para datos de tipo restricción y variable de restricción.

5.1. Introducción

La utilización de restricciones permite una mayor expresividad, pero también implica procesos más complejos para su tratamiento. Por esta razón, en esta memoria de tesis se propone una arquitectura que permita que los datos en forma de restricción sean fácilmente almacenables, y por supuesto se puedan recuperar de una manera lo más eficiente posible.

Como también se ha presentado en esta tesis, las restricciones se almacenarán como un tipo nuevo (atributo restricción), de manera que el álgebra relacional es extendida para dichas restricciones y las variables que las componen. Aunque existen trabajos [30] donde se analizan las variables que forman las restricciones, no abordan el problema de las variables como un atributo nuevo que puede participar en las operaciones de selección y proyección, siendo necesario enriquecer el lenguaje de consulta para este nuevo tipo de atributo.

Pese a ampliar el modelo relacional con las restricciones, la propuesta aquí presentada mantiene las reglas relacionales descritas por Codd [33]. La idea principal es mantener todas las ventajas del álgebra relacional junto a la expresividad que permiten las restricciones. La combinación de ambas áreas es beneficioso tanto para las bases de datos como para las construcción y resolución de problemas de satisfacción de restricciones, ya que las ventajas de una representan las deficiencias de otra. En la tabla 5.1 se muestran las ventajas e inconvenientes de cada uno de los paradigmas utilizados, la combinación de ambos dará como resultado una nueva arquitectura para gestores de Bases de Datos de Restricciones.

	Ventajas	**Desventajas**
BD Relacionales	Definición asentada desde 1970 Gran capacidad el almacenamiento Optimización de Consultas Estándar aceptado por la comunidad Solución Genérica	Representación extensiva Almacenamiento finito Consultas por extracción
Restricciones	Gran expresividad Datos infinitos en espacio finito Inferencia de nueva información Representación Intensiva Técnicas de Inferencia	Conocimiento experto Falta de estándar Soluciones a medida Complejidad de resolución

Tabla 5.1: Ventajas e Inconvenientes de Restricciones y BD Relacionales

La teoría de bases de datos relacionales es aceptada tanto por la comunidad de investigadores como por las empresas que desarrollan soluciones comerciales genéricas e independientes de las aplicaciones que las utilizan, siendo la mejor solución para el almacenamiento de gran cantidad de datos siempre que éstos puedan ser representados de

forma finita. Para los datos clásicos se han desarrollado diferentes algoritmos, minimizando los tiempos de acceso a los registros. Por otro lado, la programación con restricciones tiene gran capacidad expresiva, pero entre sus inconvenientes se encuentran que no existen soluciones generales que permitan crear problemas de satisfacción u optimización de restricciones de forma automática, lo que significa que cada modelo para resolver un problema tendrá que ser construido a medida de las necesidades. Sin embargo, la construcción y resolución de problemas de satisfacción de restricciones necesita de conocimientos más expertos y complejos, no existiendo un lenguaje estándar. Si ambos paradigmas se combinaran, se obtendría un sistema con gran capacidad de almacenamiento y gran capacidad expresiva, donde los modelos se puedan construir partiendo de la información almacenada de una manera fácil mediante un lenguaje sencillo y conocido como es SQL. Para el tratamiento de la información almacenable de forma finita se utilizarán las mejoras relacionadas con el álgebra relacional, mientras que el tratamiento de restricciones se hará utilizando técnicas y algoritmos de la programación con restricciones y eliminación simbólica. El siguiente paso es definir una forma de almacenar las restricciones en una base de datos relacional, por lo que se propone utilizar Bases de Datos Objeto-Relacionales. Como se muestra en el capítulo 2, las Bases de Datos Objeto-Relacionales permiten almacenar objetos manteniendo las características del álgebra relacional, por lo que será la alternativa de soporte lógico utilizada en esta tesis.

5.2. Las Bases de Datos Objeto-Relacionales y las Restricciones

Las restricciones que se tratarán son las expresadas en la gramática mostrada en la sección 4.2, por lo que presentan una estructura compleja que no se puede almacenar en una base de datos relacional clásica. Por dicha razón, se utilizan las Bases de Datos Objeto-Relacionales para dar soporte a las restricciones numéricas. De esta forma, las restricciones se describen en base a objetos como árboles sintácticos, donde los nodos internos son los operadores booleanos, operadores de comparación y operadores numéricos, mientras que las hojas serán las variables y las constantes que forman cada restricción. De todas formas, en las siguientes secciones se presenta más extensamente cómo son estos objetos, y cómo se harán persistentes en la base de datos. Sobre dichos objetos se construirá un conjunto de métodos para su obtención, tratamiento y modificación, produciendo una ampliación

del gestor de bases de datos clásico añadiéndoles la capacidad de trabajar con restricciones y las variables que las forman.

5.2.1. Representación de las Restricciones como Objetos usando UML

Las Bases de Datos Objeto-Relacionales analizadas por Stonebraker [143] son una ampliación de las bases de datos relacionales para soportar el almacenamiento de objetos. Estas bases de datos utilizan el lenguaje de consulta SQL2003, el cual permite la inclusión de la definición de nuevos tipo de datos y procedimientos, lo que resulta necesario para la arquitectura que se presenta en esta tesis dando soporte a las BDdR.

La elección de las Bases de Datos Objeto-Relacionales es novedosa en el mundo de las BDdR, ya que ninguna otra propuesta la ha tenido en cuenta para almacenar las restricciones. Entre las características de estas bases de datos, es relevante destacar aspectos como el uso de un lenguaje de programación de más alto nivel, lo que hará posible el tratamiento de las restricciones de una forma más fácil al representarlas como objetos, pudiendo combinar la potencia de las bases de datos relacionales con la expresividad de la orientación a objetos. La definición de este *objeto-restricción* está basada en la notación extendida de UML propuesta en [106] y mostrada en la figura 5.1.

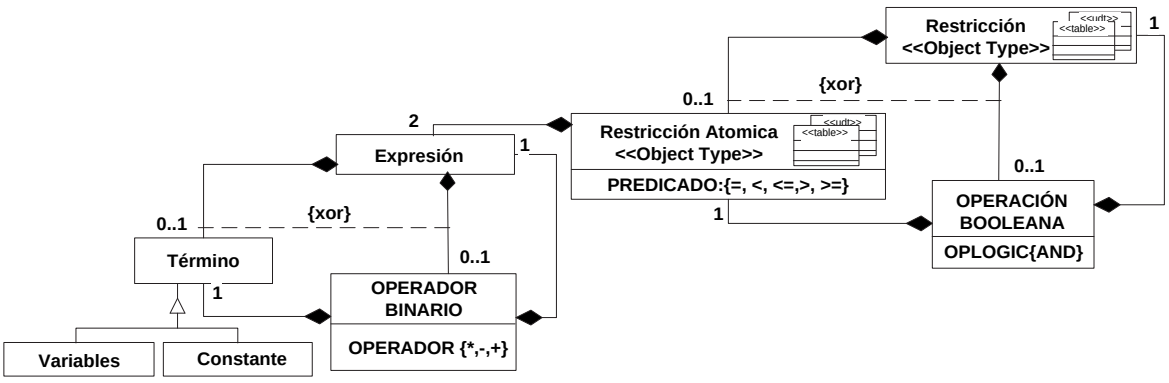

Figura 5.1: Representación del objeto-restricción con UML

En esta figura se representa una *Restricción* como una *Restricción Atómica*, o una conjunción lógica de restricciones atómicas. A su vez, una *Restricción Atómica* es la comparación de dos *Expresiones* usando los comparadores ($=$, $<$, $>$, $\leq$, $\geq$). Cada *Expresión*

es una combinación de operadores aritméticos de variables y constantes definidas sobre enteros, naturales o flotantes. En la figura 5.2 se muestra un ejemplo de cómo se representa la restricción $\{a+b>c \wedge d+a<8\}$ como un objeto en forma de árbol sintáctico. De esta forma, a una restricción representada como un objeto mediante un árbol sintáctico se le denominará *objeto-restricción*.

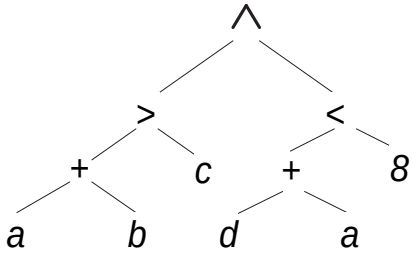

Figura 5.2: Ejemplo de árbol sintáctico de una restricción (objeto-restricción)

Un árbol sintáctico es una estructura recursiva que viene definida por una raíz y dos árboles sintácticos (hijo izquierdo e hijo derecho) si es un operador binario, o una raíz y un árbol sintáctico si es unario. Cada uno de estos árboles se representarán mediante objetos que representan las distintas restricciones, donde cada uno de estos objetos tiene un identificador, OID (Object Identification), que lo hace único y diferenciable de todos los demás.

El tipo de restricciones a las cuales se les da soporte en esta nueva arquitectura son las restricciones lineales de igualdad, las inecuaciones lineales, las polinómicas de igualdad y las inecuaciones polinómicas, todas ellas sobre el dominio de los naturales, enteros y los flotantes, las cuales se describieron en la sección 4.2.1. Este soporte debe ser ofrecido de una manera eficiente, y con una implementación modular y fácil de adaptar a nuevos requisitos. Todo esto, presentando una arquitectura independiente de la naturaleza de los datos que se estén tratando, pudiendo crear modelos en función de las consultas y de los datos almacenados en la base de datos.

Como ya se ha comentado en el capítulo anterior, la forma de evaluar una consulta dependerá de qué quiere obtener el usuario (información intensiva o extensiva), del tipo de restricciones involucradas y del dominio de las variables que las constituyen. En función del tipo de las restricciones se utilizará un motor de inferencia u otro, por lo que cuando una restricción es insertada, se analizará y se etiquetará de una de las cuatro formas: 'Lineal

de Igualdad' (`LinEQ`), 'Inecuación Lineal' (`LinIN`), 'Polinómica de Igualdad' (`PolEQ`) o 'Inecuación Polinómica' (`PolIN`).

Dos de los problemas de las Bases de Datos Objeto-Relacionales son la complejidad que conlleva el almacenamiento y tratamiento de objetos, y el coste computacional que ello provoca. Para paliar dichos problemas, se propone una arquitectura con ciertas características que ayuden a que el proceso de evaluación de las consultas sea más fácil y eficiente. Las sentencias sobre las BDdR más costosas, computacionalmente hablando, son las relacionadas con las restricciones y sus variables, ya que será necesario trabajar con objetos en lugar de con información exclusivamente extensiva acorde con el álgebra relacional clásica. En el caso de las restricciones, se almacenará información adicional relacionada con ellas, con el objetivo de facilitar su posterior tratamiento. Este análisis se realizará cuando las restricciones son añadidas a la base de datos (en tiempo de compilación), mejorando así los tiempos de la evaluación de consulta (en tiempo de ejecución). Pese a que este tratamiento hará más lenta la inserción de restricciones, se compensa con la ganancia en las consultas, que son procesos que se suelen repetir más en una base de datos. Para cada restricción se almacenarán distintas características:

- **Rango de las variables (envolvente):** Para cada variable de cada restricción se almacenarán los valores máximos y mínimos que puede tomar. Dichos valores son conocidos bien porque dicha información ha sido descrita al añadir la restricción, por las características de la propia restricción en la que está la variable, o por los valores máximos y mínimos del resto de variables de la restricción a la que pertenece. De esta forma se puede utilizar la aritmética intervalar [11] definida sobre las operaciones de la gramática descrita:

  $$[a,b] + [c,d] = [\lfloor a + c \rfloor, \lceil b + d \rceil]$$

  $$[a,b] - [c,d] = [\lfloor a - d \rfloor, \lceil b - c \rceil]$$

  $$[a,b] \times [c,d] = [min(\lfloor ac \rfloor, \lfloor ad \rfloor, \lfloor bc \rfloor, \lfloor bd \rfloor), max(\lceil ac \rceil, \lceil ad \rceil, \lceil bc \rceil, \lceil bd \rceil)]$$

  Los valores máximos y mínimos que toma cada variable para cada restricción serán almacenados en la base de datos para hacer más eficiente la evaluación de consultas. Simplemente con el análisis de dichos valores, se puede conocer si es posible que dos restricciones compartan alguna solución, si las soluciones de una restricción pueden estar incluida en otra ... Esta información también ayudará a utilizar como

heurística de resolución de problemas de satisfacción de restricciones, comenzar la búsqueda por las variables con tamaños de dominios más pequeños.

- **Tipo de la restricción:** Ya que en función de si son ecuaciones o inecuaciones la resolución simbólica se realizará con diferentes técnicas, se almacenará qué tipo de restricción es. Para la resolución numérica el tipo de restricción puede ayudar a utilizar como heurística resolver primero las variables que están en restricciones lineales y luego las que pertenecen a restricciones polinómicas.
- **Relación entre variables y restricciones:** Se mantendrá una tabla donde se almacenará la relación entre las restricciones y las variables que tienen. Esta indexación hará más eficiente la búsqueda de variables que tiene una restricción y en qué restricciones está una determinada variable. También esta información ayudará a utilizar heurísticas relacionadas con el grado de las variables, o lo que es lo mismo, el número de restricciones en las que cada variable participa para comenzar la búsqueda por dichas variables.

Estas características se reflejan en las tres tablas, mostradas en la figura 5.3, para mantener la segunda forma normal con respecto a las variables y las restricciones.

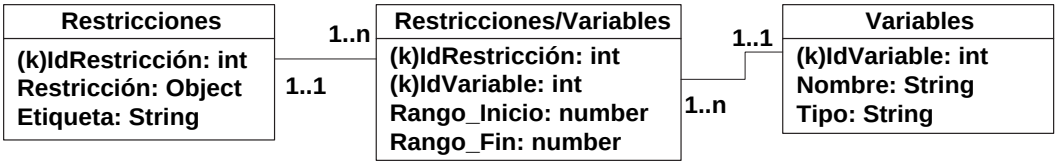

Figura 5.3: Indexación entre variables y restricciones

Cuando se crea una Base de Datos de Restricciones se crean también estas tres tablas (*Restricciones*, *Variables* y *Restricciones/Variables*), que mantienen las relaciones entre las variables y las restricciones. Estas tablas disminuyen la complejidad computacional de las consultas ya que con una sola consulta se puede conocer qué restricciones tienen qué variables y viceversa. Estas tablas permiten identificar cada restricción (tabla *Restricciones*), cada variable (tabla *Variables*) y establecer la relación entre ambas (tabla *Restricciones/Variables*), lo que ayuda a detectar las restricciones relacionadas con una consulta sin necesidad de tratarlas todas. Estas tablas no son visibles ni accesibles por el usuario de una forma directa, pero sí de una manera implícita cuando se añaden, modifican, borran o consultan restricciones de la base de datos. La tabla *Restricciones* almacena

el *idRestricción* que corresponde al identificador del objeto (OID) generado por el sistema, y la *Etiqueta* acorde con el tipo de restricción que representa. La tabla *Variables* almacena, para cada variable, su identificador, su nombre y el dominio (Natural, Entero o Flotante) al que pertenece. La tabla *Restricciones/Variables* almacena la relación entre las variables y las restricciones, y sobre qué rango (*Rango_Inicio*, *Rango_Fin*) están definidas cada una de las variables para cada una de las restricciones en las que participan.

A su vez, las restricciones necesitarán una tabla donde los objetos (árboles sintácticos) se almacenarán de forma persistente, la cual tendrá los campos presentados en la figura 5.4. Dicha figura presenta una tabla que permite serializar la información relativa al objeto-restricción para poderla hacer persistente en una base de datos basado en el teorema de Cayley y Prüfer revisado en [108]. Un objeto-restricción se almacenará como un conjunto de tuplas con el mismo **Id_Restricción**. Cada una de estas tuplas representa un subárbol sintáctico cuyo identificador se almacena en el atributo **Id_Árbol**. Cada uno de estos árboles tiene como atributos: una **Raíz**, que puede tomar como valor uno de los operadores ($\wedge$, $<$, $>$, $\leq$, $\geq$, $=$, $+$, $*$, $-$); el **Hijo_Izquierdo**, que puede ser a su vez un subárbol con un determinado **Id_Árbol**, un identificador de variable o una constante; y el **Hijo_Derecho** con la misma semántica que el hijo Izquierdo. Los atributos **Hoja_Izquierdo** y **Hoja_Derecho** describirán si los subárboles, Hijo_Izquierdo e Hijo_Derecho respectivamente, representan un subárbol (si toma el valor 0), el identificador de una variable (si toma el valor 1) o una constante (si toma el valor 2). De esta manera, pese a que los árboles sintácticos son por definición una estructura recursiva, no es necesario hacer consultas recursivas para obtener un objeto-restricción completo, ya que todas las tuplas de los subárboles que lo forman se caracterizan por tener un mismo **Id_Restricción**.

Árbol Sintáctico Restricción
Id_Restricción: Entero Identificador de la Restricción
Id_Árbol: Entero Identificador del árbol
Raiz: Cadena con la raiz del árbol
Hijo_Izquerdo: Entero Identificador del árbol Izquierdo, variable o constante
Hoja_Izquerdo: 0, 1 ó 2 dependiendo de si es árbol, variable o constante
Hijo_Derecho: Entero Identificador del árbol Derecho, variable o constante
Hoja_Derecho: 0, 1 ó 2 dependiendo de si es árbol, variable o constante

Figura 5.4: Tabla de almacenamiento de restricciones para la serialización de los objetos

Para la restricción anterior $\{a + b > c \wedge d + a < 8\}$, una forma de serializarla para almacenarla en una tabla es la mostrada en la figura 5.5. Aunque en el ejemplo se muestran los nombres de las variables, en realidad se almacenará el identificador de cada una de ellas establecido en la tabla interna *Variables*, lo que facilitará el cambio de nombre y dominio de las variables, ya que sólo se almacenan en un lugar, cumpliendo así la segunda regla de Codd sobre el acceso garantizado [33][51].

Árbol Sintáctico Restricción						
Id_Restricción	Id_Árbol	Raiz	Hijo Izquerdo	Hoja Izquerdo	Hijo Derecho	Hoja Derecho
1	1	AND	2	0	3	0
1	2	>	4	0	c	1
1	3	<	5	0	8	2
1	4	+	a	1	b	1
1	5	+	d	1	a	1
...	...	...	...	...	...	...

Figura 5.5: Ejemplo de Serialización de Objetos-Restricción

Las restricciones serán tratadas de diferente manera en función de cómo se especifique en las sentencias de consulta a la base de datos, dependiendo de si su tratamiento es numérico o simbólico. Al representarse la información como un objeto, se permite definir un conjunto de métodos pertenecientes a dicho *objeto-restricción* que hará posible crear diferentes modelos dependiendo de las restricciones involucradas y de la evaluación de la consulta necesaria.

5.3. Arquitectura del Sistema Gestor de BDdR

En este documento se propone una arquitectura que trata la optimización temporal de las consultas cuando hay involucradas restricciones en ellas. Dicha arquitectura está basada en la utilización de un gestor de Bases de Datos Objeto-Relacional, junto a la construcción de las tres tablas internas comentadas y la definición del objeto-restricción. El principal objetivo es añadir una capa que haga lo más transparente posible el uso de restricciones por parte del usuario, ampliando la sintaxis y la semántica de SQL, cambiando

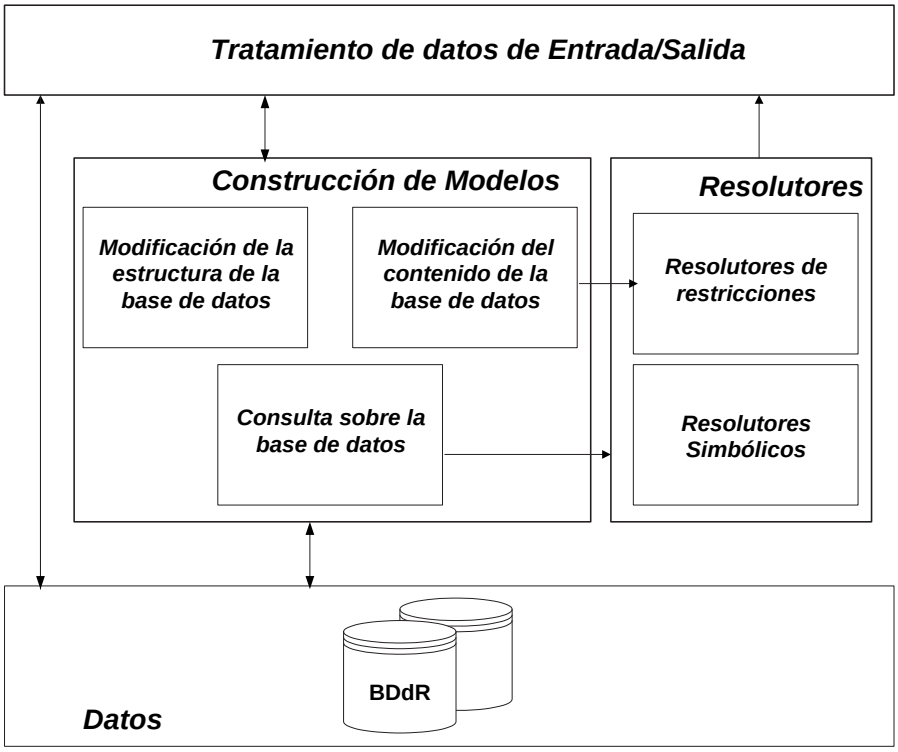

Figura 5.6: Arquitectura de la Base de Datos de Restricciones

su sintaxis lo menos posible y siguiendo la filosofía del estándar. Dicha ampliación conlleva mantener toda la semántica de SQL de forma que si no existiesen atributos del tipo restricción, sería un gestor de bases de datos relacional clásico, lo que facilita la adaptación progresiva de bases de datos ya existentes a una Base de Datos de Restricciones. Esto significa que, con respecto al tratamiento de los datos clásicos y no del tipo restricción, ni la semántica ni la sintaxis de SQL sufrirán ningún cambio, sólo se ampliarán para el tratamiento de las restricciones y las variables que las forman. La arquitectura propuesta toma como nombre LORCDB, derivado de algunas de sus características expresadas en inglés (**Labelled Object-Relational Constraint Database**).

La arquitectura, esquematizada en la figura 5.6, se divide principalmente en cuatro capas:

- **Tratamiento de datos de Entrada/Salida:** Se encarga de la relación con el usuario, analizando las sentencias de entrada y mostrando el resultado de la evaluación de las consultas. Mediante un *Análisis léxico y sintáctico* se estudia si es

necesario el tratamiento de restricciones para la evaluación de la consulta. Si no tiene restricciones involucradas, la consulta será tratada por el gestor de Base de Datos Objeto-Relacional y devolverá la información obtenida al usuario sin necesidad de un tratamiento especial. En caso de que la consulta implique el tratamiento de restricciones, el módulo *Construcción de Modelos* construirá el modelo necesario para tratar la consulta.

- **Construcción de Modelos:** Alguna de las partes que componen este módulo se explicarán con más detalle en el siguiente capítulo, ya que es la capa más compleja e importante de la arquitectura. Dependiendo del tipo de sentencia (*Creación de la BDdR*, *Creación de Tablas*, *Consultas*, ...) se utilizarán distintas técnicas y herramientas junto a un conjunto de algoritmos y heurísticas para resolverlas. Esta capa está a su vez dividida en tres, las cuales representan los tipos de acciones que se pueden realizar sobre una base de datos:

  - **Modificación de la estructura de la base de datos:** Este módulo sólo necesita comunicarse con la base de datos, e integra la *Creación de una DBdR* con las tablas internas descritas en la sección anterior, *Creación de Tablas* cuando tienen algún campo *restricción*, y *Modificación de la Estructura de Tablas* cuando éstas tienen algún campo de tipo *restricción*.
  - **Modificación del contenido de las tablas de la base de datos:** Este módulo se comunicará con la base de datos y necesitará construir problemas de satisfacción y optimización de restricciones. Entre las acciones que integra se encuentra la *Inserción de Tuplas en una Tabla* y la *Modificación del Contenido de una Tabla*, tal como se muestra en la figura 5.7. La *Inserción de Tuplas en una Tabla* añade una tupla con restricciones en una tabla determinada, construyendo el árbol sintáctico, obteniendo la envolvente (Rango_Inicio, Rango_Fin) con un COP para conocer los valores máximos y mínimos que pueden tomar cada una de las variables, y etiquetando la restricción en función de su tipo. También es necesario comprobar la integridad de la restricción (que tiene al menos una solución), ya que no tendría que permitirse introducir restricciones como por ejemplo $\{x < y \wedge y < x\}$, ya que no existe ningún valor de x e y donde dicha restricción se cumpla. Por otro lado, el submódulo de *Modificación de Contenido de una Tabla* se encarga de modificar aquellas tuplas de una relación que cumplen una determinada condición, por lo que en caso de que

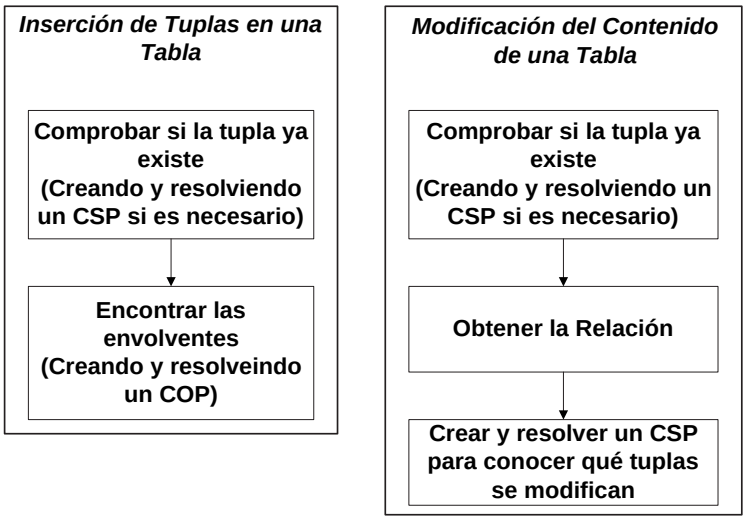

Figura 5.7: Modificación del contenido de una Bases de Datos de Restricciones

existan restricciones involucradas, necesitará construir y resolver un CSP tal como se ha explicado en la sección 4.5 para la selección de tuplas que cumplan una condición.

- **Consulta sobre la base de datos:** Este módulo gestiona el tratamiento de la sentencia cuando se realiza una consulta sobre tablas con atributos restricción, del tipo selección, proyección, producto cartesiano, unión o diferencia de conjuntos. Los detalles de este módulo serán descritos más ampliamente en la subsección 5.3.1.

- **Datos:** Representa el sistema gestor de Bases de Datos Objeto-Relacional, con la estructura de tablas internas explicadas en la sección anterior. Como ejemplo de gestor de Base de Datos Objeto-Relacional se ha considerado utilizar Oracle^{TM} 9.i, aunque se podría extender a cualquier otro gestor que permitiera el modelo lógico objeto-relacional.

- **Resolutores:** Está formado por las herramientas que resolverán, siempre que la sentencia lo requiera, los modelos construidos en la capa *Construcción de Modelos*. Como ejemplo de herramientas en esta tesis se han utilizado JSolver^{TM} [78] y Mathematica^{TM}v.5 [128], donde JSolver^{TM} evaluará las sentencias de forma numérica y Mathematica^{TM}v.5 lo hará de manera simbólica, aunque podrían ser sustituidas por otras herramientas equivalentes.

Al ser una arquitectura modular, permite el intercambio de los resolutores de modelos, de esta forma provee de mayor capacidad de adaptación a nuevos tipos de restricciones, nuevos dominios de variables, añadir otras operaciones definidas para la consulta, y ampliar las herramientas que resuelvan los modelos. Esto es posible ya que se utiliza*el patrón de diseño Estrategia*, pudiendo decidir en tiempo de ejecución qué herramienta realizará la evaluación de la consulta.

5.3.1. Consulta sobre la base de datos

Este módulo, presentado en la figura 5.8, da soporte a los cinco tipos de operaciones primitivas del álgebra relacional pero adaptadas al tratamiento de las restricciones, lo cual ha derivado en la definición de los dos nuevos tipos de atributos.

Cualquiera de las relaciones de salida puede ser a su vez una relación de entrada que participe en otra operación, manteniendo así la propiedad de cierre o clausura del álgebra relacional. Dicha característica se observa en la figura 5.8 entre el módulo *Consulta* y los restantes módulos definidos para las distintas operaciones. Esto hace posible que una consulta pueda estar formada por varias operaciones anidadas, por ejemplo una proyección que se hace sobre la relación obtenida como salida en una selección.

Aunque en el siguiente capítulo se analizan todos los detalles de implementación relativos a cada una de las operaciones, aquí se presentan algunos detalles de interés:

- **Selección:** La operación de selección se realiza sobre una relación que puede ser una tabla de la base de datos o la relación de salida de otra operación, lo cual se hace en *Obtener las relaciones involucradas*. De entre las tuplas de dicha relación, formarán parte de la salida de la operación de selección aquellas que cumplan una determinada *condición*. En el submódulo *Seleccionar las restricciones* se obtendrán aquellas tuplas que cumplan la condición en función del álgebra relacional sobre los atributos clásicos, que puedan cumplir la condición en función de los valores máximos y mínimos que puedan tomar las variables (envolvente), y que tengan variables relacionadas con el predicado. Para conocer cuáles de estas tuplas cumplen el predicado, siempre que estén involucrados atributos restricciones, será necesario construir un CSP de la forma que se ha explicado en el capítulo anterior (sección 4.5).

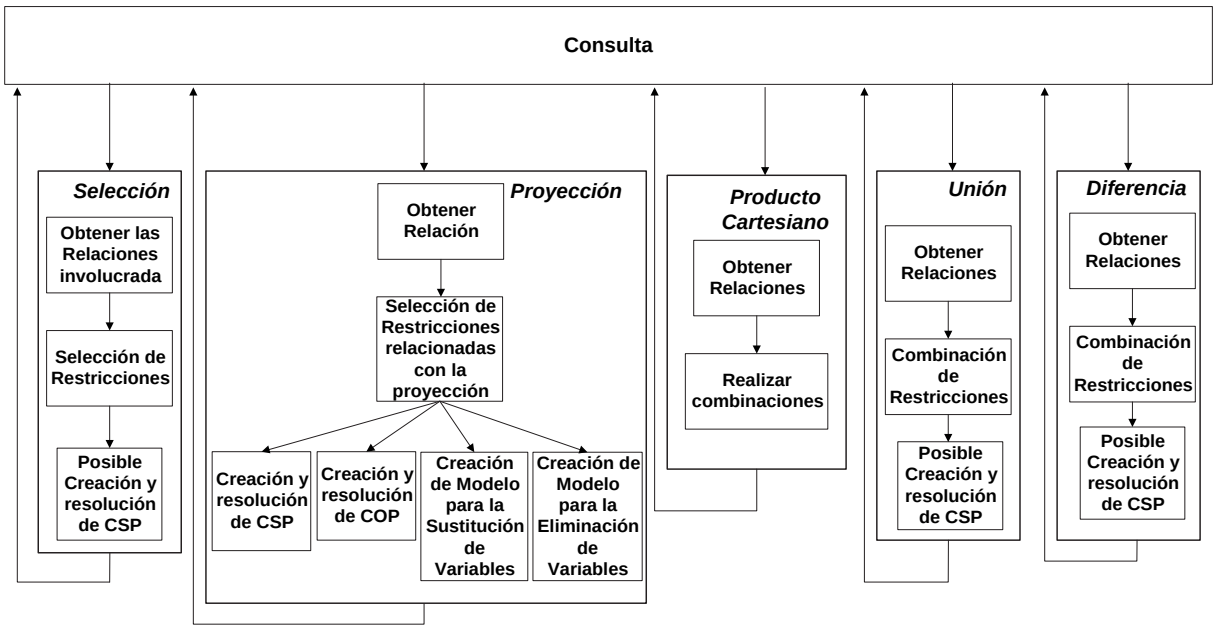

Figura 5.8: Consulta sobre la Bases de Datos de Restricciones

- **Proyección:** Esta es la operación más compleja, ya que la proyección es una selección horizontal de los atributos que forman la relación, por lo que también pueden ser restricciones o variables de restricción. La mayor problemática la establece la obtención de las variables de restricción, que pueden ser conjuntos de valores instanciados de las variables (obtenidos mediante CSP o COP), o nuevas restricciones inferidas de las originales por la eliminación o sustitución de variables, como se explicó en el capítulo anterior (sección 4.6). Al igual que en la selección, se tendrá que obtener primero la relación de entrada de la operación (tabla de la base de datos o relación de salida de otra operación) y los atributos restricción relacionados con la proyección que se pretende realizar.

- **Producto Cartesiano:** Como se ha analizado en el capítulo anterior, no se necesita un tratamiento especial aunque existan atributos restricción en las relaciones sobre las que se realiza el producto cartesiano. Al igual que en el resto de operadores, las relaciones de entrada de dicho operador pueden ser tablas de la base de datos o relaciones de salida de otras operaciones.

- **Unión:** En el caso de que las tuplas contengan atributos restricción hay que cerciorarse de que no existan dos tuplas repetidas en la relación de salida como se analizó

en la sección 4.7. Si los atributos clásicos toman valores diferentes, o las envolventes de las distintas restricciones no tienen valores en común, no será necesario crear y resolver un CSP.

- **Diferencia:** Tiene las mismas características que la unión, analizando si las restricciones no tienen valores en común o las soluciones de una están incluidas en la de otra (sección 4.8), por lo que también en algunos casos será necesario crear y resolver un CSP.

5.4. CORQL: Ampliación de SQL para BDdR

Una vez introducidos los distintos módulos de la arquitectura, esta sección presenta la sintaxis y la semántica de las sentencias que hacen uso de ellos. Este conjunto de sentencias definirán una ampliación de SQL para Bases de Datos de Restricciones que recibirá el nombre de **CORQL (Constraint Object-Relational Query Language)**.

5.4.1. Creación de una LORCDB

Para la creación de una LORCDB se utiliza la siguiente sintaxis:

```
CREATE LORCDB <nombre_base_de_datos>
```

Como se ha explicado en este capítulo, las restricciones se almacenarán como objetos en una Base de Datos Objeto-Relacional, pero con una estructura de tablas internas que permite indexar las variables y almacenar los valores máximos y mínimos de cada variable para hacer las consultas más eficientes. Esto significa que cuando una LORCDB es creada con la sentencia anterior, también estas tres tablas son creadas. Estas tablas estarán protegidas ante el acceso de los usuarios de forma explícita, sólo permitiendo que se modifiquen por el sistema cuando las restricciones son añadidas, modificadas o borradas.

5.4.2. Creación de tablas en una LORCDB

Siguiendo la filosofía de modificar la sintaxis de SQL lo menos posible, la creación de las tablas para las LORCDB es similar a la de las bases de datos relacionales clásicas,

con la única diferencia de poder crear columnas (atributos) de tipo restricción (Atributo Restricción).

Un ejemplo de creación de tablas puede ser:

```
CREATE TABLE Parcelas(IdPropietario Integer,
  Nombre String, Parcela Constraint)
```

Aunque también es posible crear una tabla con varios atributos restricción. Por ejemplo, las parcelas y las casas que contiene las parcelas:

```
CREATE TABLE ParcelasYCasas(IdPropietario Integer,
  NombrePropietario String, Parcela Constraint, Casa Constraint)
```

5.4.3. Inserción de tuplas en la tabla

Si un atributo es de tipo restricción, será posible añadir restricciones en dicha columna. Al igual que el resto de tipos, se realizará la comprobación de que el tipo del elemento insertado es conforme con el tipo de la columna donde se añade la información.

Cuando se añade una restricción a la base de datos, se actualiza la tabla*Restricciones*, junto a la tabla *Variables* si se añade alguna variable nueva, y *Variables/Restricción* para crear la relación entre las variables y la restricción añadida. Las variables de las restricciones tienen que estar definidas sobre un tipo (*Natural*, *Entero* y *Flotante*); si no se especifica, el dominio por defecto será *Flotante*. También es posible definir el rango de las variables para un determinado dominio. La sintaxis de las restricciones insertadas tiene que seguir la gramática definida en la subsección 4.2.1 para restricciones.

La sintaxis de la inserción de la información será de la forma:

```
INSERT INTO <nombre de la tabla> ( <lista de atributos> )
VALUES ( <lista de valores> )
```

Si un atributo se ha creado como atributo restricción, el valor que se le da tendrá la sintaxis:

```
'{'<cadena restricción>[,'('Dominio [Rango] nombre_variable[,Dominio [Rango]
nombre_variable ... ]')]}'
```

Un ejemplo de inserción de restricciones en la tabla anterior puede ser:

```
INSERT INTO ParcelasYCasas(IdPropietario, Nombre, Parcela, Casa)
VALUES
(101, 'Jesús Pineda', {'x^2 + y^2 <= 49'(Float -10..10 x, Float y)},
{'x > 0 AND x <= 5 AND y > 0 AND y <= 6.5'})
```

Como se muestra en el ejemplo, es posible definir el dominio de cada variable, teniendo en cuenta que si una variable participa en varias restricciones, es necesario que en todas esté definida sobre el mismo tipo, aunque el rango en cada restricción pueda tomar diferentes valores. También es posible acotar los posibles valores que puede tomar una variable utilizando la sintaxis `Rango_Inicio..Rango_Fin`, que en caso de no especificarse será $-\infty..+\infty$ que corresponderá por defecto con {Integer.MIN_VALUE..Integer.MAX_VALUE} para las variables de Enteros, {0..Integer.MAX_VALUE} para las variables de Naturales, y {Float.MIN_VALUE..Float.MAX_VALUE} para las variables de Flotantes. Estos valores son los definidos en el lenguaje $Java^{TM}$ usado para la implementación del lenguaje CORQL. Cuando una restricción se añade a la base de datos, se comprueba su consistencia, ya que si no existe ninguna solución para ella no se realizará su inserción, por ejemplo $x^2 > 2x^2$.

Resumiendo, los pasos para almacenar una tupla con atributos restricción en una LORCDB son:

1. Comprobar que la sintaxis de la sentencia es correcta.
2. Comprobar que ninguna de las variables de la restricción de entrada está ya en la base de datos con otro tipo. En caso de que esto ocurra se mostrará un error y no se realizará la inserción.
3. Comprobar que dicha tupla no está ya almacenada.
4. Crear un objeto con la restricción y almacenarlo en la tabla *Restricciones*.
5. Rellenar el campo de la etiqueta en la tabla *Restricciones* en función del tipo de restricción que sea (`LinEQ, LinIN, PolEQ, PolIN`).
6. Si la variable ya está en la base de datos no se volverá a incluir a la tabla *Variables*, sí se incluirá en caso contrario.

7. Crear tuplas en la tabla *Restricciones/Variables* para la nueva restricción y todas las variables, tanto las nuevas como las que estaban.

8. Crear un COP, al que se le incluirá el rango definido en la inserción, para conocer para cada variable de la restricción su valor mayor y menor, y almacenarlos en los campos `Rango_Inicio` y `Rango_Fin` de la tabla *Restricciones/Variables*.

Tras realizar estos pasos, las tablas para la inserción anteriormente mostradas quedarán como se muestra en la figura 5.9, donde la tabla *ParcelasYCasas* tiene dos atributos de tipo restricción. Los atributos restricción contienen el identificador del objeto que se almacena en la tabla *Restricciones*, a la vez que sus variables son almacenadas en la tabla *Variables* para definir la indexación con las restricciones mediante la tabla *Restricciones/Variables*. Pese a que el rango de la variable `x` definida en la sentencia es $-10..10$, el máximo valor que puede tomar para que la restricción $\{x^2 + y^2 \leq 49\}$ sea satisfactible es $-7..7$, que serán los almacenados en la tabla *Restricciones/Variables*. Con respecto a la etiqueta, en este caso las restricciones son inecuaciones polinómicas (PolIN) y lineales (LinIN). Estas etiquetas ayudarán a escoger qué herramienta es más adecuada para evaluar las distintas consultas sobre las diferentes tuplas.

ParcelasYCasas

Id	Nombre	Parcela	Casa
101	J. Pineda	1	2
...	...	...	...

Restricciones

Id	Restricción	Etiqueta
1	$x^2+y^2=<49$	PolIN
2	x>0 & x=<5 & y>0 & y=<6.5	LinIN
...	...	...

Restricciones/Variables

IdRes	IdVar	Rango_Ini	Rango_Fin
1	1	-7	7
1	2	-7	7
2	1	0	5
2	2	0	6.5
...	...	...	...

Variables

Id	Nombre	Tipo
1	x	Float
2	y	Float
...	...	...

Figura 5.9: Ejemplo de tablas rellenas

5.4.4. Consulta a una LORCDB

Una consulta a una Base de Datos de Datos de Restricciones es una combinación de las operaciones primitivas del álgebra relacional presentadas en el capítulo anterior en la sección 4.3.

La sintaxis de las consultas serán de la forma:

```
consulta := SELECT <lista_de_atributos(t1) [,lista_de_atributos(t2), ...]>
            FROM <lista_de_tablas(t1, t2, ...)>
            WHERE predicado
          [{UNION|MINUS} consulta]
```

El parámetro <lista_de_atributos(t_i)> representa diferentes tipos de proyección realizada sobre los atributos pertenecientes en la tabla t_i, que pueden ser atributos clásicos, restricciones o variables de restricciones. Si se quieren obtener atributos de distintas tablas (t_1, t_2, ...) conllevará la realización del producto cartesiano entre la relación resultante de cada tabla. El parámetro `predicado` hace referencia al operador de selección, definiendo las condiciones que tienen que cumplir los atributos de la relación. Por último `UNION` y `MINUS` serán los operadores de unión y diferencia de conjuntos.

El orden de prioridad al resolver las operaciones serán:

1. Selección (`WHERE predicado`)
2. Proyección para cada tabla (`SELECT` <lista_de_atributos(t_i)>)
3. Producto cartesiano, uniendo todas las relaciones
4. Unión y diferencia, ambas con la misma prioridad

Con respecto a la lista de atributos, donde `at` representa atributos clásicos, $\texttt{at}_c$ atributos restricción, y $\texttt{at}_c$`.variable` atributos variables de restricción, la sintaxis será:

```
lista_de_atributos := atributo ',' lista_de_atributos
      | atributo
      ;
atributo := ['MIN' | 'MAX' | 'AVG' | 'COUNT' ...] at
      | atc
      | [VERTICAL | HORIZONTAL] atributo_Variable
      ;
atributo_Variable := 'VALUES' '('listaAtVariables')'
```

```
        | 'CONSTRAINTS' '('listaAtVariables')'
        | 'FULL CONSTRAINTS' '('listaAtVariables')'
        | 'VALUES'[n] '('listaAtVariables')'
        | ['MIN' | 'MAX'] 'VALUE' '('at_c.variable')'
        | 'BOOLEAN VALUE' '('at_c')'
        ;
listaAtVariables := at_c'.'variable ',' listaAtVariables
        | at_c'.'variable
        ;
```

Cuando la proyección se realiza sobre atributos variables de restricción, existen diferentes alternativas en función de cómo y qué tipo de información se quiere obtener:

- Para determinar qué restricciones de la relación está involucradas en la operación de proyección, se utilizan palabras reservadas del lenguaje `VERTICAL` y `HORIZONTAL`. Dichas palabras representan si la relación entre las variables se pueden hacer entre restricciones de la misma tupla (`VERTICAL`), o del mismo atributo pero entre distintas tuplas (`HORIZONTAL`). Si no se especifica, la opción por defecto es `HORIZONTAL`.
- Cuando se quiere obtener como salida otra restricción, se utiliza la palabra reservada `CONSTRAINTS`, pero cuando se quiere especificar que se quiere hacer sustitución simbólica y no simplemente eliminación, se utiliza la sentencia `FULL CONSTRAINTS`.
- Cuando se quiere hacer una proyección sobre atributos variable de restricción, pero obteniendo valores instanciados de las variables, se utiliza la palabra `VALUES`, lo que devolverá una tupla con valores de las variables. En caso de querer obtener más de una tupla, se utilizará `VALUES` [n], donde n representa el número de tuplas que se quieren obtener. Si se quiere obtener sólo un valor de una variable, el mínimo o el máximo, será ['MIN' | 'MAX'] `'VALUE'` la sentencia utilizada.
- Cuando se quiere conocer si existe un valor que hace satisfactible cada una de las restricciones de un atributo restricción, se utiliza la sentencia `BOOLEAN VALUE`, donde la salida será `true` o `false`.

Con respecto al predicado, y siguiendo la definición introducida en 4.5, la sintaxis será de la forma:

```
predicado := condición
    | condición ['AND' | 'OR'] predicado
    ;
condición := at ['<' | '≤' | '>' | '≥' | '=' | '<>']
   [at | constante]
  | at_c ['<' | '≤' | '>' | '≥' | '=' | '<>' | '&' | '⊂' | '⊆' | '⊃' | '⊇']
   [at_c | restricción]
  | at_c ['<' | '≤' | '>' | '≥' | '=' | '<>' | '&' | '⊂' | '⊆' | '⊃' | '⊇']
   '(' [at_c | restricción] 'FOR' '{'listaVariables'}'
   ['AND' listaAtVarIgual]')'
  | atributo_Variable ['<' | '≤' | '>' | '≥' | '=' | '<>']
   [at_c'.'variable | constante]
  ;
listaVariables := Variable
  | Variable ',' listaVariables
  ;
listaAtVarIgual := at_c'.'Variable '=' at_c'.'Variable
  | at_c'.'Variable '=' at_c'.'Variable 'AND' listaAtVarIgual
  ;
```

5.4.5. Modificación de la estructura de una tabla

Existen dos acciones de modificación de la estructura de una tabla, una de ellas es la modificación de la estructura, añadiendo una nueva columna con la sintaxis:

```
ALTER TABLE <nombre_de_la_tabla> ADD <nombre_de_la_columna>
```

```
<tipo_de_la_columna>
```

donde la columna añadida puede ser un atributo restricción.

La otra opción es borrar una columna que también puede ser un atributo restricción. En ese caso, se eliminarán todas las restricciones que estén en la tabla *Restricciones*, las variables que no pertenezcan a otras restricciones de *Variables*, y por supuesto las entradas que relacionen las variables y restricciones de *Restricciones/Variables*. La sintaxis sería:

```
ALTER TABLE <nombre_de_la_tabla> DROP <nombre_de_la_columna>
```

5.4.6. Modificación del contenido de una tabla

Con respecto a la modificación de contenido, se puede modificar la información ya almacenada o eliminar tuplas completas. La sintaxis para la modificación de las tuplas es:

```
UPDATE <nombre_de_la_tabla> SET
  <nombre_del_atributo> = <nuevo_valor>
WHERE predicado
```

donde la columna de la sentencia tiene que estar involucrada en el predicado. Con esta operación será posible cambiar el contenido de un atributo clásico, un atributo restricción, o el nombre de una variable de una restricción. Cambiar el nombre de una variable no es más que modificar un registro de la tabla *Variables*.

Con respecto al borrado de una tupla, la sintaxis será:

```
DELETE FROM <nombre_de_la_tabla> WHERE predicado
```

en este caso, en el predicado tienen que estar involucrados algunos de los atributos de la tabla sobre la cual se va a realizar el borrado. El predicado seguirá la sintaxis y la semántica explicada en la sección 4.5.1 del capítulo anterior.

5.5. Resumen

En este capítulo se presenta la arquitectura que da soporte a la nueva definición de Base de Datos de Restricciones. La arquitectura LORCDB tiene características que ayudarán a que la evaluación de consultas sea más fácil y eficiente, almacenando información

sobre las restricciones. La información que almacena está relacionada con el rango del dominio donde está definida, el tipo de restricción y la relación entre restricciones y las variables que las conforman. Para trabajar con atributos del tipo restricción, se ha definido una extensión de SQL llamado CORQL, definiendo la gramática relativa a las distintas operaciones que se pueden realizar sobre una Base de Datos de Restricciones, como su creación, creación de tablas, inserción de tuplas, consultas, modificaciones y borrados. Para ello se ha utilizado la redefinición de las cinco primitivas del álgebra relacional, teniendo en cuenta los atributos restricción y atributos variable de restricción, propuestos en el capítulo anterior.

Capítulo 6

Implementación del lenguaje CORQL

Una vez definida la arquitectura que da soporte a las Bases de Datos de Restricciones y el nuevo lenguaje CORQL presentado como extensión de SQL, en este capítulo se presentará cómo dicha arquitectura junto a un conjunto de algoritmos, hacen más práctica y eficiente la evaluación de las distintas operaciones relacionadas con la consulta. Para facilitar la presentación de las diferentes estrategias de evaluación de consultas, se introduce un ejemplo hipotético basado en los impuestos a pagar cuando se realiza la venta de productos en función de la provincia o el país involucrado.

6.1. Ejemplo de caso de estudio

Como ejemplo para hacer más claros los pasos que conforman los distintos algoritmos relacionados con la implementación de cada una de las operaciones del álgebra relacional para BDdR, se utiliza la base de datos mostrada en la figura 6.1. Dicha base de datos almacena las provincias (tabla *Provincia*) de fabricación de distintos productos (tabla *Producto*). Cuando se realiza la venta de dichos productos será necesario pagar unos impuestos provinciales si se venden dentro de su misma provincia (tabla *ImpuestosProvinciales*), impuesto nacionales si se venden a otra provincia del mismo país, donde se pagarán los impuestos en función de la provincia donde se realice la venta (tabla *ImpuestosNacionales*), o impuestos internacionales si se venden a otro país (tabla *ImpuestosInternacionales*) lo que dependerá del país donde se haga dicha venta.

Algunas de estas tablas tienen atributos de tipo *Restricción*, donde se definirá una relación entre variables. En el caso de la tabla *Provincia* se establecerá la relación entre las

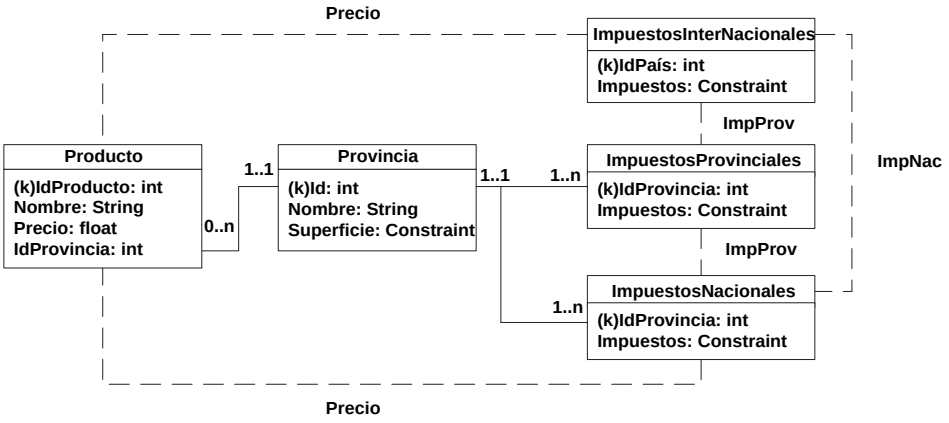

Figura 6.1: Diagrama de Tablas del ejemplo

variables espaciales x e y en el atributo restricción *Superficie*. Algunos ejemplos concretos de estas restricciones son los presentos en la figura 6.2.

Provincias		
Id	**Nombre**	**Superficie**
1	Sevilla	-27 x +39 y < 5682 AND 8x+67 y < 14656 AND 75 x +19 y < 15575
2	Cádiz	28 x +9 y < 6053 AND 5 x -42 y < -3149 AND -33 x +33 y < -1683
3	Huelva	34 x +80 y < 14050 AND 16 x -13 y < 3421 AND -41 x -67 y < -13481 AND -9 x < -1665
4	Córdoba	-24x +10y < -7298 AND 28x +30 y < 12806 AND 18x -24 y < 4986 AND -22x -16 y < -8678
5	Málaga	7 x +36 y < 20488 AND 39 x -22 y < -10270 AND -46 x -14 y < -8660
6	Jaen	88 y < 50776 AND 57 x < 7980 AND -39 x -88 y < -51220 AND -18 x < -936
7	Granada	39 x +88 y < 51220 AND -110 y < -57200 AND -39 x +22 y < 10270
...	...	...

Figura 6.2: Ejemplo de tabla Provincias

Otro ejemplo de atributo restricción es *ImpuestosNacionales.Impuestos*, que relaciona las variables *ImpNac, ImpProv y Precio*. Al igual que en las bases de datos relacionales clásicas, se pueden definir relaciones entre los atributos univaluados de distintas tablas mediante la definición de claves ajenas, como la que existe entre *Provincias.id* e *ImpuestosNacionales.IdProvincia*. Aunque en las BDdR se crean relaciones entre las variables de las restricciones de forma implícita, por ejemplo entre los atributos restricción *Impuestos* para las tablas *ImpuestosProvinciales*, *ImpuestosNacionales* e *ImpuestosInter-*

nacionales. Dicha relación se representa en la figura 6.1 con líneas discontinuas junto al nombre de las variables. Cabe destacar el atributo *Precio*, que puede representar tanto un atributo clásico, cuando se denota como *Producto.Precio*, como un atributo variable de restricción *ImpuestosNacionales.Impuestos.Precio*. Un ejemplo del contenido de las tablas más relevantes, ya que tienen atributos de tipo restricción, se muestra en la figura 6.3. Como se puede observar, las restricciones tienen variables que las relacionan, las cuales estarán almacenadas e indexadas en las tablas *Restricciones*, *Restricciones/Variables* y *Variables*, por lo que el cambio de las restricciones no implicará cambios estructurales en el diseño de la base de datos, sólo cambios en el contenido de las tablas de indexación.

ImpuestosProvinciales

IdProvincia	Impuestos
1	0<Precio<3.000 AND ImpProv=0.05*Precio+50
1	3.000<=Precio<6.000 AND ImpProv=0.06*Precio+53
1	12.000<=Precio AND ImpProv=0.07*Precio+61
2	0<Precio<24.000 AND ImpProv=0.09*Precio
3	0<Precio<38.000 AND ImpProv=0.10*Precio+50
4	0<Precio<12.000 AND ImpProv=0.02*Precio+50
4	12.000<=Precio<41.000 AND ImpProv=0.13*Precio+50
...	...

ImpuestosNacionales

IdProvincia	Impuestos
1	0<=ImpProv<500 AND ImpNac=ImpProv + 0.02Precio
1	500<ImpProv AND ImpNac=ImpProv + 0.01Precio
2	0<=ImpProv<200 AND ImpNac=Precio*Precio*0.05
2	200<=ImpProv AND ImpNac=Precio*0.07
3	ImpNac=ImpProv*ImpProv+99
4	0<=Precio<2000 AND ImpNac=ImpProv + 0.01Precio
4	2000<=Precio<=5000 AND ImpNac=0.11*Precio + 54
...	...

ImpuestosInternacionales

IdPaís	Impuestos
1	ImpInterNac=ImpNac + 10
2	ImpInterNac=ImpNac+12
3	ImpInterNac=ImpNac*1.02
4	ImpInterNac=ImpProv + 100
5	ImpInterNac=ImpProv + 0.01Precio
6	ImpInterNac=ImpProv + 0.02Precio
7	ImpInterNac=ImpNac*1.20
8	ImpInterNac=ImpNac*1.0545
9	ImpInterNac=ImpProv
10	ImpInterNac=ImpProv + 26
11	ImpInterNac=ImpProv + 98
12	ImpInterNac=Precio*0.025
13	ImpInterNac=Iva*Valor*0.025
...	...

Figura 6.3: Ejemplo de tablas que almacenan Restricciones

Sobre estas tablas se presentarán distintos ejemplos para las cinco operaciones primitivas del álgebra relacional, donde se reflejará cómo se incorporan distintas estrategias para el tratamiento de restricciones, junto a los detalles de cada uno de los algoritmos necesarios para que la evaluación de dichas operaciones sea eficiente.

6.2. Implementación de la Selección

Para que la implementación de la selección sea eficiente, se define una estrategia para tratar, en un determinado orden, cada una de las condiciones del predicado. La estrategia de evaluación de la selección consistirá en, partiendo de una relación de entrada (R), obtener un subconjunto horizontal R_{Salida}. Para obtener la salida, se ejecutará una secuencia de pasos donde en cada uno de ellos se eliminarán las tuplas que no cumplan una parte de

la condición. Esto vendrá determinado por el tipo de atributo involucrado en el predicado, tal como se muestra en la figura 6.5.

1. **Atributos Univaluados:** Partiendo de una relación R de entrada, sobre la que se quieren obtener aquellas tuplas que cumplen un predicado, primero se seleccionarán aquellas cuyos atributos clásicos o univaluados cumplan el predicado en función del álgebra relacional clásica, obteniendo una nueva relación R'. Esto es posible gracias al nuevo modelo de BDdR introducido en esta tesis, ya que en el resto de propuestas no existía esta división entre los datos univaluados y los datos restricción.

2. **Atributos Variable de Restricción:** El segundo tratamiento estará relacionado con los atributos variable de restricción, donde dichos atributos representan variables numéricas cuyos valores máximos y mínimos están almacenados en las tablas de indexación de la base de datos. Los predicados relacionados con ellos, pueden definir una comparación entre dos variables, una variable y un atributo clásico numérico, o una variable y una constante. Estos tres tipos de comparaciones se tratarán mediante el análisis de los límites (máximo y mínimo) entre los dos atributos involucrados en la comparación del predicado. Por ejemplo, si las tuplas de la relación de salida tienen que cumplir que la variable `a` es menor que el valor 5, y el dominio de dicha variable para una restricción es 10..15, significará que la tupla a la que pertenece dicha variable no formará parte de la salida. En algunos casos, esta eliminación de tuplas se podrá realizar sin necesidad de crear ningún problema de satisfacción de restricciones, tan sólo siendo necesario una consulta clásica, ya que los valores máximos y mínimos de cada variable están almacenados en atributos clásicos. Sin embargo, para estar seguros que la variable puede tomar ese valor, será necesario crear un CSP.

3. **Atributos Restricción:** Para los atributos restricción se harán dos tipo de tratamientos. En principio se analizarán los rangos de las variables (envolventes), para conocer si se cumple el predicado involucrado con los atributos restricción. Si con este análisis no es suficiente, será necesario crear y resolver problemas de satisfacción de restricciones.

   - **Análisis de Rangos de las Variables:** Analizando el mínimo y máximo valor que puede tomar una variable, es posible inferir si las restricciones involucradas

en la comparación cumplen o no una condición. Por ejemplo si se comparan dos restricciones C_x y C_y, ambas definidas sobre las variables v_1 e v_2, donde los rangos son $C_x(v_1{:}[5..15],\ v_2{:}[20..30])$ y $C_y(v_1{:}[20..25],\ v_2{:}[40..55])$. Si el predicado fuera $C_x{<}C_y$, se puede asegurar que la tupla a la que pertenecen ambas restricciones formará parte de la relación de salida, mientras que si el predicado fuera $C_x{\subseteq}C_y$, la tupla a la que pertenecen las restricciones no formaría parte de la relación de salida.

- **Construcción de CSP:** Aunque existen casos donde es posible saber si una tupla formará parte de la relación de salida sin necesidad de construir y resolver un CSP, también existen casos donde no es posible saberlo con un simple análisis de los dominios de las variables. Volviendo al ejemplo anterior, si los dominios de las variables fueran $C_x(v_1{:}[5..15],\ v_2{:}[20..30])$ y $C_y(v_1{:}[2..25],\ v_2{:}[10..55])$, y el predicado $C_x{\subseteq}C_y$, pese a que los dominios de las variables en C_x están incluidas en los dominio de las variables en C_y, no se puede asegurar que se cumple el predicado para las restricciones. Un ejemplo donde los rangos de las variables están incluidos dentro de otra, pero no sus soluciones, es el mostrado en la figura 6.4. En esos casos será necesario crear y resolver un problema de satisfacción de restricciones con ambas restricciones de la forma presentada en la sección 4.5.1.

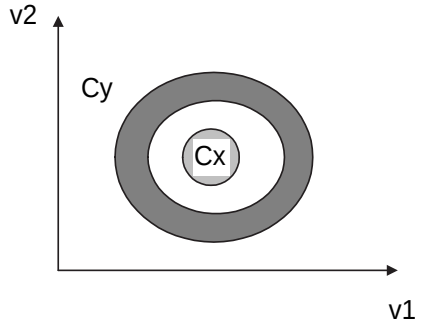

Figura 6.4: Ejemplo de $C_x \nsubseteq C_y$

Un ejemplo de consulta con selección sobre las tablas de la figura 6.3, y siguiendo los pasos descritos anteriormente puede ser:

*Obtener los impuestos provinciales, impuestos nacionales e impuestos internacionales que tienen que pagar los productos de la provincia con identificador igual a*1. *Además se*

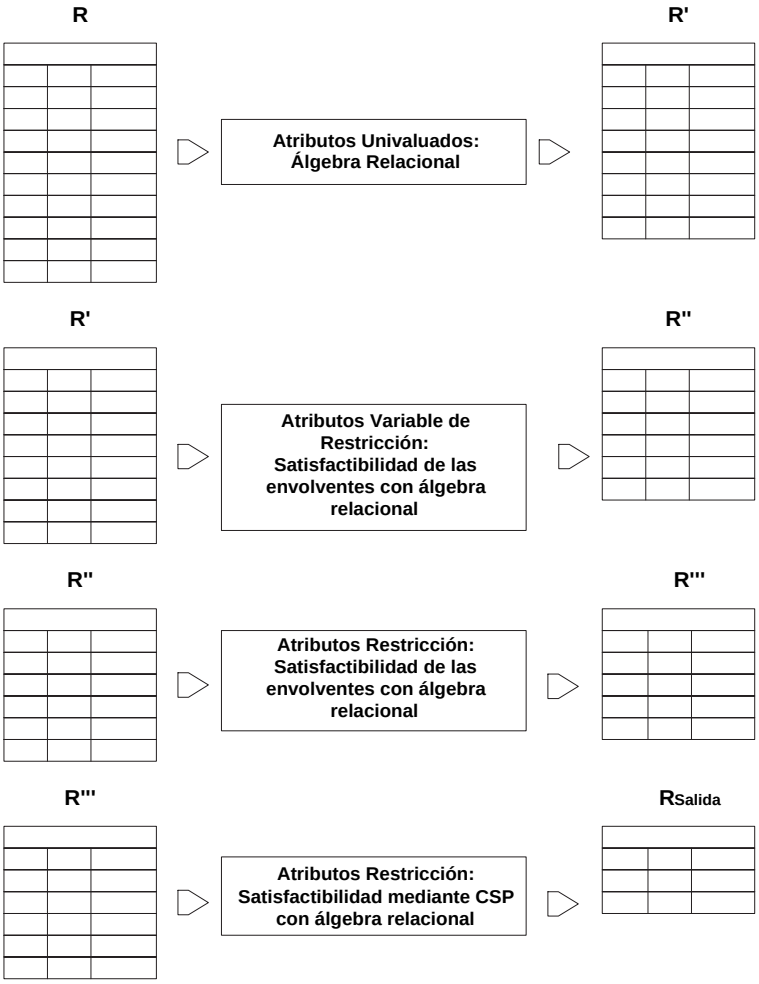

Figura 6.5: Pasos de la operación de Selección

desea que de entre todas las posibilidades de impuestos provinciales, sólo presentará los impuestos para los productos cuyos precios son mayores de 4.000 *y menores de* 5.000, *y que para cada impuesto nacional (ImpNac) el impuesto internacional siempre sea menor que* $ImpNac * 1.25$.

La sintaxis de consulta para el ejemplo será de la forma:

```
SELECT *

FROM ImpuestosProvincial, ImpuestosNacionales, ImpuestosInternacionales

WHERE ImpuestoProvinciales.idProvincia = 1 AND

  ImpuestoNacionales.idProvincia = 1 AND
```

```
ImpuestoProvincial.Impuestos.Precio > 4000 AND
ImpuestoProvincial.Impuestos.Precio < 5000 AND
ImpuestosInternacionales.Impuestos < (ImpInterNac = ImpNac * 1,25
  FOR {ImpInterNac} AND
  ImpuestosInternacionales.Impuestos.ImpNac=ImpNac)
```

Este ejemplo muestra un predicado donde hay involucrados atributos clásicos, como es el caso de `ImpuestoProvinciales.id` = 1, atributos restricción como `ImpuestosInternacionales.Impuestos` $<$ $(ImpInterNac = ImpNac * 1{,}25)$, y variables de restricción como `ImpuestoProvincial.Impuestos.Precio` $>$ 4000. Todos estos tipos de atributos ayudarán a explicar la problemática del tratamiento de los datos de tipo restricción y cómo son evaluados en la selección.

Para cada una de las relaciones (tablas) que aparezcan en el predicado de la consulta se resolverán de forma independiente, al menos que existan condiciones que comparen dos relaciones. El orden de resolución será el mostrado en la figura 6.5, que para este ejemplo serán:

1. **Atributos Clásicos (univaluados)**. *Eliminar todas aquellas tuplas que no cumplan las condiciones relativas a los atributos univaluados*

   Esta eliminación es la que hacen las bases de datos relacionales clásicas, evitando el tratamiento de dichas tuplas mediante problemas de satisfacción de restricciones para saber si se cumple una condición. Si no se almacenaran los atributos restricción y los univaluados de forma independiente esto no sería posible, con el consecuente gasto computacional.

   En el caso del ejemplo, no pertenecerán a la relación de salida aquellas tuplas de la tabla *ImpuestoProvinciales*, cuyo atributo *id* sea distinto de 1, junto a las tuplas de *ImpuestosNacionales* cuyo *idProvincia* no sea igual a 1. De esta forma sólo quedarán las tuplas mostradas en la figura 6.6.

2. **Atributos Variable de Restricción**. *Eliminar todas aquellas tuplas relativas a variables de atributos restricción cuya envolvente no cumpla la parte del predicado relacionada con ellas.*

ImpuestosProvinciales	
IdProvincia	Impuestos
1	0<Precio<3.000 AND ImpProv=0.05*Precio+50
1	3.000<=Precio<6.000 AND ImpProv=0.06*Precio+53
1	12.000<=Precio AND ImpProv=0.07*Precio+61

ImpuestosNacionales	
IdProvincia	Impuestos
1	0<=ImpProv<500 AND ImpNac=ImpProv + 0.02Precio
1	500<ImpProv AND ImpNac=ImpProv + 0.01Precio

ImpuestosInternacionales	
IdPaís	Impuestos
1	ImpInterNac=ImpNac +10
2	ImpInterNac=ImpNac+12
3	ImpInterNac=ImpNac*1.02
4	ImpInterNac=ImpProv + 100
5	ImpInterNac=ImpProv + 0.01Precio
6	ImpInterNac=ImpProv + 0.02Precio
7	ImpInterNac=ImpNac*1.20
8	ImpInterNac=ImpNac*1.0545
9	ImpInterNac=ImpProv
10	ImpInterNac=ImpProv + 26
11	ImpInterNac=ImpProv + 98
12	ImpInterNac=Precio*0.025
13	ImpInterNac=Iva*Valor*0.025

Figura 6.6: Relaciones tras la eliminación por atributos univaluados (Paso I)

Esto también se transformará en una consulta clásica sobre la base de datos, ya que la envolvente está almacenada como dos atributos numéricos de la relación*Restricciones/Variables*. Esto provocará que el tiempo de evaluación de las consultas sea menor, ya que se eliminarán todas aquellas tuplas cuyas restricciones relacionadas con la condición, no puedan cumplir el predicado por su envolvente. Más aún, se facilitará la resolución de los CSP, si su construcción fuera necesaria, añadiendo para cada variable de cada restricción sus valores máximos y mínimos (Rango_Inicio, Rango_Fin). Esta aportación evitará crear problemas de satisfacción de restricciones que de todas formas no van a ser satisfactibles.

En el ejemplo, servirá para eliminar todas las tuplas de la tabla de*ImpuestosProvinciales* cuyo valor de la variable *Precio* no contenga valores entre 4.000 y 5.000. De forma que finalmente podrán formar parte de la relación de entrada las tuplas que contienen la restricción del atributo *ImpuestosProvinciales.Impuesto*: $3.000 \leq Precio < 6.000\ AND\ ImpProv = 0.06 * Precio + 53$. Tras esta eliminación, las tuplas que hasta el momento estarán involucradas en la consulta son las mostradas en la figura 6.7.

3. **Atributos Restricción**. *Analizar la parte del predicado de la selección donde se establece la comparación entre dos restricciones*.

   Sólo se analizarán aquellas tuplas que tengan las mismas variables que la restricción del predicado. Para el caso del ejemplo, donde se quieren obtener todas las tuplas tal que *ImpuestosInternacionales.Impuestos* < *ImpInterNac*=*ImpNac*∗1.25 sólo se analizarán aquellas cuyo *IdPaís* es 1, 2, 3, 7 y 8, ya que las demás no relacionan el *ImpNac* con el *ImpInterNac*. Una vez detectado este subconjunto de tuplas, será

ImpuestosProvinciales	
IdProvincia	Impuestos
1	0<Precio<3.000 AND ImpProv=0.05*Precio+50

ImpuestosNacionales	
IdProvincia	Impuestos
1	0<=ImpProv<500 AND ImpNac=ImpProv + 0.02Precio
1	500<ImpProv AND ImpNac=ImpProv + 0.01Precio

ImpuestosInternacionales	
IdPaís	Impuestos
1	ImpInterNac=ImpNac + 10
2	ImpInterNac=ImpNac+12
3	ImpInterNac=ImpNac*1.02
4	ImpInterNac=ImpProv + 100
5	ImpInterNac=ImpProv + 0.01Precio
6	ImpInterNac=ImpProv + 0.02Precio
7	ImpInterNac=ImpNac*1.20
8	ImpInterNac=ImpNac*1.0545
9	ImpInterNac=ImpProv
10	ImpInterNac=ImpProv + 26
11	ImpInterNac=ImpProv + 98
12	ImpInterNac=Precio*0.025
13	ImpInterNac=Iva*Valor*0.025

Figura 6.7: Relaciones tras la eliminación por atributos variables de restricción (Paso II)

necesario analizar las restricciones que la forman. También las envolventes de las variables, donde se aproxima su comportamiento con los valores máximos y mínimos que pueden tomar, ayudan a conocer si una tupla con restricciones va a formar o no parte de la solución.

Para dos restricciones C_x y C_y, donde al menos una de ellas formará parte de una relación de la base de datos, la relación del rango entre una de las variables que tienen en común puede ser de diferentes formas, como se muestra en la figura 6.8, lo que puede ayudar a conocer si cumplen o no un predicado. En el caso de que ambas restricciones formen parte de una relación, el análisis se realiza comparando cada una de las restricciones de una de las relaciones con todas las restricciones de la otra, lo que conllevará $n*m$ combinaciones, donde n y m representan el número de tuplas de cada una de las relaciones respectivamente. Un ejemplo podría ser un predicado que incluyera entre sus condiciones $ImpuestosProvinciales.Impuestos \subseteq ImpuestosNaciones.Impuestos$, donde se tendrían que comparar todas las restricciones de los impuestos provinciales con cada una de las restricciones de los impuestos nacionales.

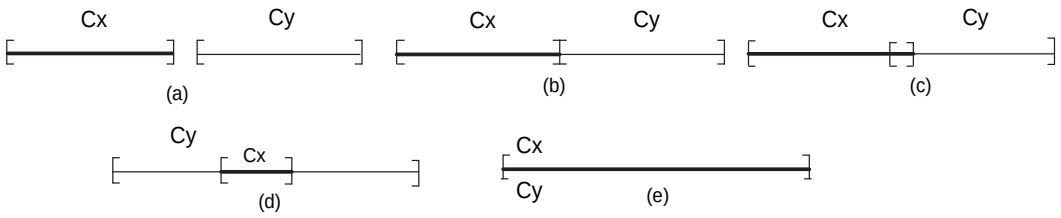

Figura 6.8: Tipos de relaciones entre los rangos de las variables de dos restricciones

En el trabajo de P. Veltri [150] se introdujo esta idea de aproximación mediante cajas de consistencia, para mejorar los tiempos de evaluación de consulta en las BDdR. Pero dicho trabajo tiene ciertas limitaciones, ya que sólo define las operaciones y tipos de relación entre restricciones lineales que representan polígonos en dos dimensiones. Esta tesis se basa en dicha propuesta, pero extendiendo su uso a todo tipo de restricción polinómica, que no tienen que representar un polígono, y para un número indeterminado de dimensiones. Además, dicho trabajo añade las restricciones que describen la caja de consistencia a la restricción que acota, mientras que una de la propuestas de esta tesis radica en no almacenar las envolventes como restricción, sino como datos numéricos clásicos con características relacionales, evitando así tener que construir y resolver el problema de satisfacción de restricciones. Las opciones de relación entre dos restricciones con respecto a sus envolventes, y para cada una de sus variables (v), las mostradas en figura 6.8, son:

(a) Todos los valores que la variable v puede tomar para hacer satisfactible C_x son siempre menores que los que puede tomar para hacer satisfactible C_y.

(b) El mayor valor que la variable v puede tomar para hacer satisfactible C_x es igual que el menor que puede tomar para hacer satisfactible C_y.

(c) La variable v puede tomar un conjunto de valores que hacen satisfactible tanto la restricción C_x como la C_y, y los que exclusivamente hacen que C_x sea satisfactible son menores que los que pueden hacer C_y satisfactible.

(d) Para la variable v, todos los valores que puede tomar en C_x también son solución en C_y, aunque en C_y puede tomar más valores.

(e) Todos los valores que v puede tomar para hacer satisfactible C_x también son soluciones de C_y, y viceversa.

Estas relaciones entre dominios ayudarán a conocer las relaciones entre restricciones descritas con las operaciones definidas en la sección 4.5, evitando en algunos casos construir y resolver problemas de satisfacción de restricciones. Generalizando dicha relación para n variables, siendo $v_1 \ldots v_n$ las variables de las restricciones C_x y C_y, y sea $\text{Rango\_Inicio}_{C_x}(v_1)$, ..., $\text{Rango\_Inicio}_{C_x}(v_n)$ los menores valores que pueden tomar las variables para hacer satisfactible la restricción C_x, $\text{Rango\_Fin}_{C_x}(v_1)$, ..., $\text{Rango\_Fin}_{C_x}(v_n)$ los valores mayores que puede tomar las variables para hacer satisfactible la restricción C_x, $\text{Rango\_Inicio}_{C_y}(v_1)$, ..., $\text{Rango\_Inicio}_{C_y}(v_n)$ los valores menores que puede tomar las variables para hacer satisfactible la restricción C_y, y

Rango_Fin$_{C_y}(v_1)$, ..., Rango_Fin$_{C_y}(v_n)$ los valores mayores que pueden tomar las variables para hacer satisfactible la restricción C_y. En los casos donde se añade la cláusula `FOR` en la condición, para determinar sobre qué conjunto de variables se realiza la comparación, sólo se analizarán dichas variables. Mientras que si también se utiliza la palabra reservada `AND` junto a `FOR`, se definirá la igualdad semántica entre las variables involucradas en la condición. Los distintos casos dependiendo de la operación entre las restricciones C_x y C_y, y los rangos de sus variables son:

- Si $\forall v \in v_1, \ldots, v_n$ o $\forall v \in v_i, \ldots, v_j$ $(i \in 1 \ldots n,\ j \in 1 \ldots n$ y $i \neq j)$ en caso de que se utilice la cláusula *FOR* $\{v_i, \ldots, v_j\}$, se cumple que:

  Rango_Fin$_{C_x}(v)$ < **Rango_Inicio**$_{C_y}(v)$ **(Figura 6.8.a)**, entonces:

  - C_x<C_y y C_x≤C_y: cierto
  - C_x>C_y y C_x≥C_y: falso
  - C_x=C_y: falso
  - C_x<>C_y: cierto
  - C_x&C_y: falso
  - C_x⊂C_y y C_x⊆C_y: falso
  - C_x⊃C_y y C_x⊇C_y: falso

  En la figura 6.9 se muestran ejemplos de relación entre las cajas de satisfacción, resaltadas con sombreado, para dos restricciones. En la figura 6.9.a se presenta un ejemplo donde C_x<C_y, lo que significa que todos los valores de v_1 y v_2 que hacen satisfactible C_x, son menores que los valores de dichas variables que hacen satisfactibe C_y. En la figura 6.9.b se presenta un ejemplo donde C_x<C_y FOR $\{v_2\}$, lo que significa que todos los valores de v_2 que hacen satisfactible C_x son menores que los valores de dicha variable que hacen satisfactible C_y.

- Si $\forall v \in v_1, \ldots, v_n$ o $\forall v \in v_i, \ldots, v_j$ $(i \in 1 \ldots n,\ j \in 1 \ldots n$ y $i \neq j)$ en caso de que se utilice la cláusula *FOR* $\{v_i, \ldots, v_j\}$, se cumple que:

  Rango_Fin$_{C_x}(v)$ = **Rango_Inicio**$_{C_y}(v)$ **(Figura 6.8.b)**, entonces:

  - C_x<C_y: falso
  - C_x≤C_y: cierto
  - C_x>C_y y C_x≥C_y: falso
  - C_x=C_y: falso

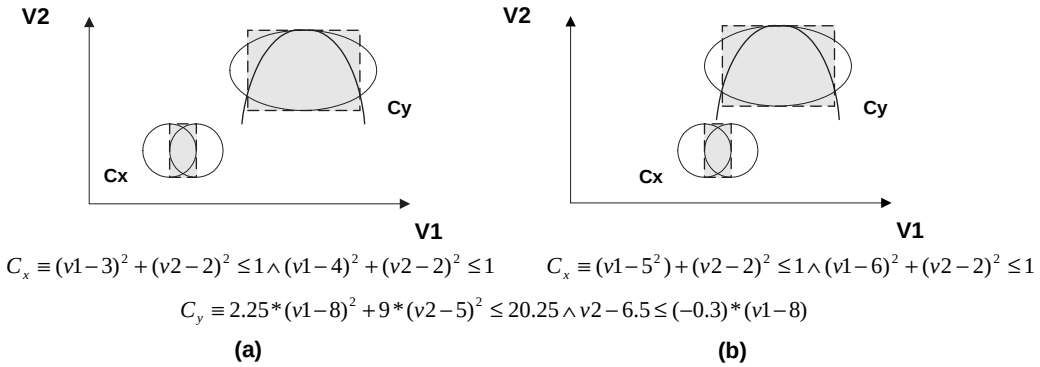

Figura 6.9: Ejemplo de relación entre restricciones en función de sus envolventes

- C_x<>C_y: cierto
- C_x&C_y: No se tiene suficiente información, hay que crear un CSP tal como se explicó en la subsección 4.5.1
- C_x⊂C_y y C_x⊆C_y: falso
- C_x⊃C_y y C_x⊇C_y: falso

▪ Si $\forall v \in v_1 \ldots v_n$ o $\forall v \in v_i \ldots v_j$ ($i \in 1 \ldots n$, $j \in 1 \ldots n$ y $i \neq j$), en caso de que se utilice la cláusula *FOR* $\{v_i, \ldots, v_j\}$, se cumple que:

Rango_Fin$_{C_x}(v)$ > **Rango_Inicio**$_{C_y}(v)$ **y** **Rango_Inicio**$_{C_x}(v)$ < **Rango_Inicio**$_{C_y}(v)$ **(Figura 6.8.c)**, entonces:

- C_x<C_y y C_x≤C_y: falso
- C_x>C_y y C_x≥C_y: falso
- C_x=C_y: falso
- C_x<>C_y: cierto
- C_x&C_y: No se tiene suficiente información, hay que crear un CSP tal como se explicó en la subsección 4.5.1
- C_x⊂C_y y C_x⊆C_y: falso
- C_x⊃C_y y C_x⊇C_y: falso

▪ Si $\forall v \in v_i, \ldots, v_j$ ($i \in 1 \ldots n$, $j \in 1 \ldots n$ y $i \neq j$) en caso de que se utilice la cláusula (*FOR* $\{v_i, \ldots, v_j\}$ AND C_x.var=C_y.var AND ...), se cumple que:

Rango_Fin$_{C_x}(v)$ > **Rango_Inicio**$_{C_y}(v)$ **y** **Rango_Inicio**$_{C_x}(v)$ < **Rango_Inicio**$_{C_y}(v)$ **(Figura 6.8.c)**, entonces:

- C_x<C_y *FOR* $\{v_i, \ldots, v_j\}$ y $C_x \leq C_y$ *FOR* $\{v_i, \ldots, v_j\}$: No se tiene suficiente información, hay que crear un CSP tal como se explicó en la subsección 4.5.1
- C_x>C_y *FOR* $\{v_i, \ldots, v_j\}$ AND ... y $C_x \geq C_y$ *FOR* $\{v_i, \ldots, v_j\}$ AND ...: falso
- C_x=C_y *FOR* $\{v_i, \ldots, v_j\}$ AND ...: falso
- C_x<>C_y *FOR* $\{v_i, \ldots, v_j\}$ AND ...: cierto
- $C_x \subset C_y$ *FOR* $\{v_i, \ldots, v_j\}$ AND ... y $C_x \subseteq C_y$ *FOR* $\{v_i, \ldots, v_j\}$ AND ...: falso
- $C_x \supset C_y$ *FOR* $\{v_i, \ldots, v_j\}$ AND ... y $C_x \supseteq C_y$ *FOR* $\{v_i, \ldots, v_j\}$ AND ...: falso

Para los ejemplos mostrados en la figura 6.10, la sentencia $C_x < (C_y \quad FOR \quad \{v_2\})$ será cierta exclusivamente para el caso 6.10.a, mientras que la sentencia $C_x < (C_y \quad FOR \quad \{v_2\} \quad AND \quad C_x.v_1 = C_y.v_1)$ será cierto para los casos a y b de dicha figura. El problema es que para poder asegurar que $C_x < (C_y \quad FOR \quad \{v_2\} \quad AND \quad C_x.v_1 = C_y.v_1)$ es cierto para el ejemplo 6.10.b, será necesario crear y resolver un CSP, no siendo suficiente con analizar las envolventes (marcada con línea discontinua en la figura).

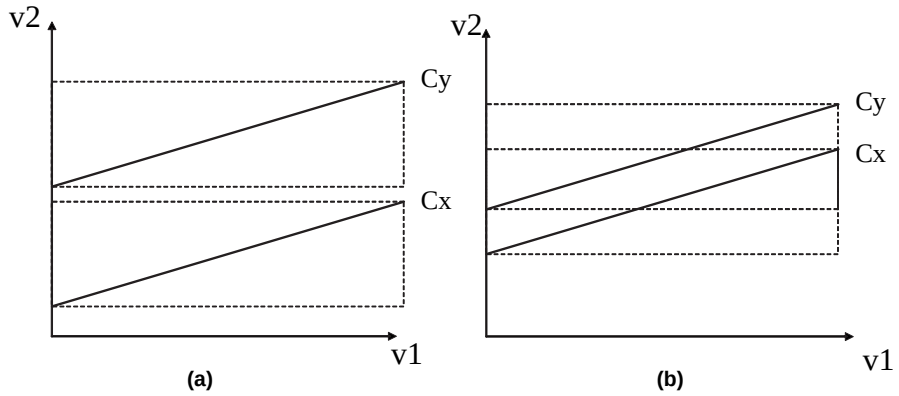

Figura 6.10: Ejemplo de comparación entre restricciones

- Si $\forall v \in v_1, \ldots, v_n$ o $\forall v \in v_i, \ldots, v_j$ ($i \in 1 \ldots n$, $j \in 1 \ldots n$ y $i \neq j$) en caso de que se utilice la cláusula *FOR* $\{v_i, \ldots, v_j\}$, se cumple que:

  Rango_Inicio$_{C_x}(v)$ > **Rango_Inicio**$_{C_y}(v)$ **y** **Rango_Fin**$_{C_x}(v)$ < **Rango_Fin**$_{C_y}(v)$**, lo que significa que el dominio de** C_x **está incluido en el** C_y **(Figura 6.8.d)**, entonces:

- C_x<C_y y C_x≤C_y: falso
- C_x>C_y y C_x≥C_y: falso
- C_x=C_y: falso
- C_x<>C_y: cierto
- C_x&C_y: No se tiene suficiente información, hay que crear un CSP tal como se explicó en la subsección 4.5.1
- C_x⊂C_y y C_x⊆C_y: No se tiene suficiente información, hay que crear un CSP tal como se explicó en la subsección 4.5.1
- C_x⊃C_y y C_x⊇C_y: No se tiene suficiente información, hay que crear un CSP tal como se explicó en la subsección 4.5.1

- Si $\forall v \in v_1, \ldots, v_n$ o $\forall v \in v_i, \ldots, v_j$ ($i \in 1 \ldots n$, $j \in 1 \ldots n$ y $i \neq j$) en caso de que se utilice la cláusula *FOR* $\{v_i, \ldots, v_j\}$, se cumple que:

  Rango_Inicio$_{C_x}(v)$ = **Rango_Inicio**$_{C_y}(v)$ **y** **Rango_Fin**$_{C_x}(v)$ = **Rango_Fin**$_{C_y}(v)$ **(Figura 6.8.e)**, entonces:

  - C_x<C_y y C_x≤C_y: falso
  - C_x>C_y y C_x≥C_y: falso
  - C_x=C_y y C_x<>C_y No se tiene suficiente información, hay que crear un CSP tal como se explicó en la subsección 4.5.1
  - C_x⊂C_y y C_x⊆C_y: No se tiene suficiente información, hay que crear un CSP tal como se explicó en la subsección 4.5.1
  - C_x⊃C_y y C_x⊇C_y: No se tiene suficiente información, hay que crear un CSP tal como se explicó en la subsección 4.5.1

  En la figura 6.11.a se presenta un ejemplo donde la caja de valores satisfactibles de C_x está incluida en la de C_y y realmente todas las soluciones de C_x lo son también de C_y. Por otro lado en la figura 6.11.b se presenta un ejemplo donde, pese a que el rango de C_x es igual que el de C_y, no todas las soluciones de C_x lo son de C_y. En el caso de las operaciones de igualdad e inclusión, en función de los rangos será posible asegurar que dichas comparaciones no se cumplen, pero para asegurar que se cumplen siempre será necesario crear y resolver un problema de satisfacción de restricciones.

- Si no se cumple ninguna de las opciones mencionadas sobre los rangos de las variables, la salida será *falso*. Esto se tendrá que hacer para todas las variables

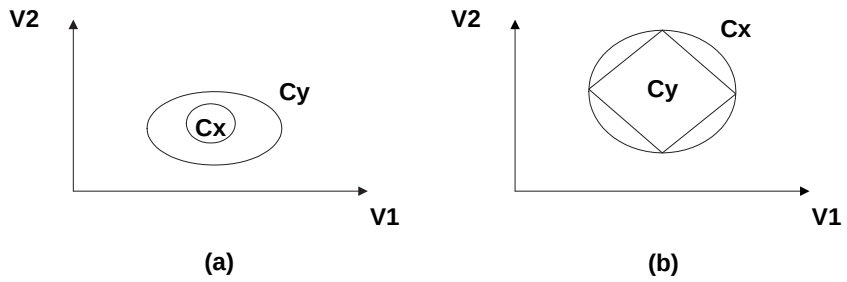

Figura 6.11: Ejemplo de restricciones

de las restricciones involucradas en el predicado. Si no existe la variante *FOR* se hará para todas las variables, hasta que se encuentra una con la que la evaluación devuelva *falso*, o se hayan estudiado todas las variables involucradas en las restricciones. En caso de que finalmente devuelva *cierto*, significa que las tuplas a las que pertenecen las restricciones formarán parte de la relación de salida.

Continuando con el ejemplo de la figura 6.7, se analizarán las tuplas de la relación *ImpuestosInternacionales* con *IdPaís* igual a 1, 2, 3, 7 y 8, ya que son los únicos que comparten las variables con la restricción del predicado. En la figura 6.12 se muestra gráficamente el contenido del atributo restricción *Impuestos* de la relación *ImpuestosInternacionales*. Analizar dichas restricciones para seleccionar aquellas que cumplan *ImpuestosInternacionales.Impuestos* $<$ *(*$ImpInterNac = ImpNac * 1{,}25$ *FOR {ImpInterNac} AND ImpuestosInternacionales.Impuestos.ImpNac=ImpNac)*, significa encontrar las restricciones donde para un valor de la variable *ImpNac*, siempre su impuesto internacional *ImpInterNac* será menor que el que pueda tomar en la restricción $\{ImpInterNac = ImpNac * 1{,}25\}$. En este caso es posible discriminar las restricciones de las tuplas con *idPaís* 1 y 2, ya que el mínimo valor que puede tomar la variable *ImpInterNac* en ambas, es mayor que el que puede tomar la misma variable en la restricción de entrada $\{ImpInterNac = ImpNac * 1{,}25\}$. Para las restantes restricciones, no será suficiente analizar exclusivamente las envolventes, siendo necesario construir un CSP comparando cada una de las restricciones de *ImpuestosInternacionales.Impuestos* con $\{ImpInterNac = ImpNac * 1{,}25\}$, para conocer las tuplas donde se cumple la condición.

La resolución de los CSP informará que las restricciones que cumplen la condición son las que pertenecen a las tuplas con *idPaís* igual a 3, 7 y 8, que son las mostradas en la figura 6.13.

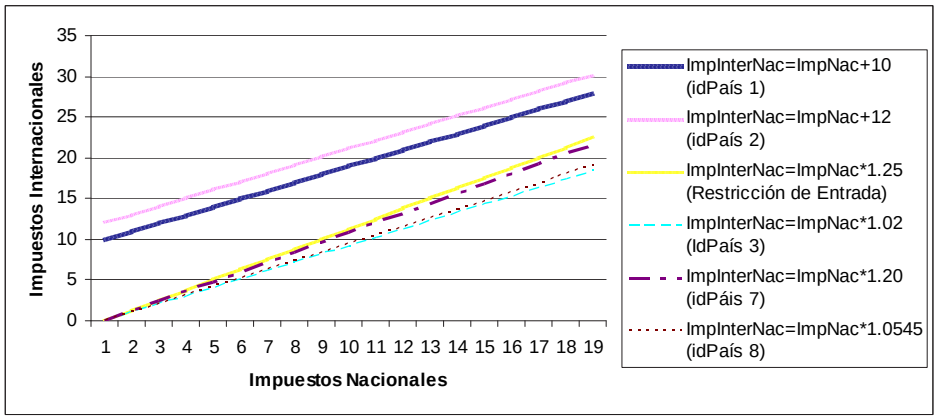

Figura 6.12: Representación de restricciones

ImpuestosProvinciales	
IdProvincia	Impuestos
1	0<Precio<3.000 AND ImpProv=0.05*Precio+50

ImpuestosNacionales	
IdProvincia	Impuestos
1	0<=ImpProv<500 AND ImpNac=ImpProv + 0.02Precio
1	500<ImpProv AND ImpNac=ImpProv + 0.01Precio

ImpuestosInternacionales	
IdPaís	Impuestos
3	ImpInterNac=ImpNac*1.02
7	ImpInterNac=ImpNac*1.20
8	ImpInterNac=ImpNac*1.0545

Figura 6.13: Relaciones tras la eliminación por construcción de CSP (Paso III y IV)

6.3. Implementación del Producto Cartesiano

El producto cartesiano entre dos relaciones devuelve una nueva relación formada por la unión conjuntiva de las tuplas de cada una de las relaciones de entrada. Por lo que la implementación de esta operación no necesita de un tratamiento especial pese a que las relaciones contengan atributos restricción. La salida de la consulta propuesta en la sección 6.2 obtendrá las tres relaciones mostradas en la figura 6.13, para realizar el producto cartesiano de ellas, será necesario efectuar dos productos cartesianos para obtener una única relación de salida. Siendo *ImpuestosProvinciales* (R_1), *ImpuestosNacionales* (R_2) e *ImpuestosInternacionales* (R_3), se tendrán que realizar dos veces el producto cartesiano, por ejemplo $((R_1 \times R_2) \times R_3)$ o $(R_1 \times (R_2 \times R_3))$, ambos equivalentes. Pese a que estas relaciones están formadas por atributos restricción, se hará como en el álgebra relacional clásica, concatenando horizontalmente cada tupla de una relación con las tuplas de la otra relación. Para el ejemplo finalmente se obtendrá una nueva relación, presentada en la figura 6.14.

ImpuestosProvinciales. idProvincia	ImpuestosProvinciales. Impuestos	ImpuestosNacionales. idProvincia	ImpuestosNacionales. Impuestos	ImpuestosInterNacionales idPais	ImpuestosInterNacionales. Impuestos
1	0<=Precio<3.000 AND ImpProv=0.05*Precio+50	1	0<=ImpProv<500 AND ImpNac=ImpProv + 0.02Precio	3	ImpInterNac=ImpNac*1.02
1	0<=Precio<3.000 AND ImpProv=0.05*Precio+50	1	0<=ImpProv<500 AND ImpNac=ImpProv + 0.02Precio	7	ImpInterNac=ImpNac*1.20
1	0<=Precio<3.000 AND ImpProv=0.05*Precio+50	1	0<=ImpProv<500 AND ImpNac=ImpProv + 0.02Precio	8	ImpInterNac=ImpNac*1.0545
1	0<=Precio<3.000 AND ImpProv=0.05*Precio+50	1	500<ImpProv AND ImpNac=ImpProv + 0.01Precio	3	ImpInterNac=ImpNac*1.02
1	0<=Precio<3.000 AND ImpProv=0.05*Precio+50	1	500<ImpProv AND ImpNac=ImpProv + 0.01Precio	7	ImpInterNac=ImpNac*1.20
1	0<=Precio<3.000 AND ImpProv=0.05*Precio+50	1	500<ImpProv AND ImpNac=ImpProv + 0.01Precio	8	ImpInterNac=ImpNac*1.0545

Figura 6.14: Salida del ejemplo tras el Producto Cartesiano

6.4. Implementación de la Proyección

La operación de proyección obtiene un subconjunto vertical de la relación de entrada, lo que significa un subconjunto de los atributos de la relación. Dependiendo del tipo de atributo sobre el cual se realice la proyección (clásicos (univaluados), atributos restricción o variable de restricción), se obtendrán diferentes salidas y se tratarán de diferente manera. En el caso de la proyección sobre atributos de tipo restricción o de tipo clásico, el comportamiento de la proyección será igual que en el álgebra relacional clásica, la diferencia fundamental está cuando la proyección hace referencia a las variables atributos restricción. Para obtener un subconjunto de las variables que aparecen en las restricciones, no sólo es necesario tratar las restricciones en las que dichas variables aparecen, también es necesario conocer las restricciones con variables que a su vez están relacionadas con ellas. Volviendo al ejemplo de la figura 6.14, si se quiere obtener el *Impuesto Internacional* que se paga en función del *Precio*, significará hacer una proyección donde se obtienen las variables *ImpuestosProvinciales.Impuestos.Precio* e *ImpuestosInternacionales.Impuestos.ImpInterNac*. Obtener la relación simbólica entre estas variables significa tener que eliminar las variables *ImpNac* e *ImpProv*, las cuales aparecen en el atributo restricción *ImpuestosNacionales.Impuesto*. Si en lugar de una proyección simbólica se quisiera obtener valores numéricos de las variables, también se tendrían que tener en cuenta las restricciones del atributo restricción *ImpuestosNacionales.Impuesto*, ya que restringen los valores que puede tomar la variable *ImpProv* y por lo tanto los que pueden tomar *ImpNac* e *ImpInterNac*.

Dependiendo de si la salida de la evaluación de la proyección es intensiva o extensiva (simbólica o numérica), se utilizará uno de los dos algoritmos que se proponen en esta sección con respecto a la búsqueda de las restricciones relacionadas con los atributos de la proyección. En el caso de la obtención de las instancias de las variables mediante

resolución numérica, será necesario encontrar todas las restricciones que estén involucradas con las variables de la proyección directa o indirectamente. Mientras que para la proyección simbólica se buscarán los conjuntos de restricciones que permitan realizar eliminación o sustitución de variables, lo que significa que será posible devolver nuevas restricciones donde sólo las variables de la proyección estén involucradas.

En esta tesis se definen dos tipos de proyección con respecto a cómo se realiza esta búsqueda de restricciones relacionadas con los atributos variable de restricción:**Proyección Horizontal** y **Proyección Vertical**. Dependiendo de si la búsqueda de restricciones relacionadas por variables se realiza por tuplas o por columnas, se utilizará un caso u otro. Se denomina **Proyección Horizontal** a la búsqueda de restricciones relacionadas con la proyección cuando las restricciones pertenezcan a la misma tupla (6.15.a). Para la proyección horizontal se realizará una búsqueda de restricciones relacionadas por cada $G_i \in G_1 \ldots G_n$. Por otra parte, la **Proyección Vertical** realizará la búsqueda de restricciones relacionadas entre las distintas tuplas de un mismo atributo restricción (6.15.b). Para la proyección vertical se realizará una búsqueda de restricciones relacionadas por cada$G_i \in G_1 \ldots G_m$.

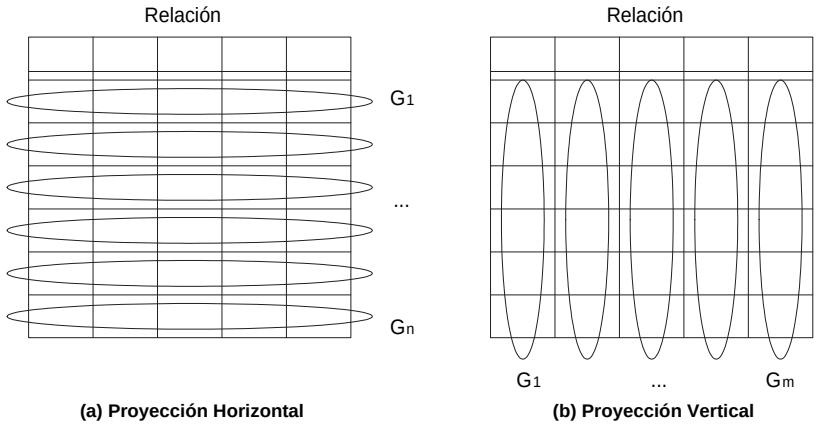

Figura 6.15: Tipos de Proyección: Horizontal y Vertical

Los atributos variable de restricción que participen en la proyección, definirán si la operación se realiza con las restricciones de una misma tupla (horizontal), o con las restricciones de una misma columna (vertical). Si todas las variables pertenecen a un mismo atributo restricción, la proyección será vertical por defecto, mientras que si pertenecen a más de un atributo restricción se realizará la proyección horizontal por defecto. En el caso de que aparezca una sola variable como atributo en la proyección, se definirá hori-

zontal por defecto. Estas definiciones por defecto se pueden variar utilizando las palabras reservadas `HORIZONTAL` y `VERTICAL`.

El objetivo primero en la proyección será obtener todos los conjuntos de restricciones que contengan las variables que aparezcan en la consulta, o que estén relacionadas con dichas variables. La definición formal de qué es un *Conjunto de Restricciones Relacionadas con la Proyección* dependerá de si la evaluación es simbólica o numérica, y de si la proyección es horizontal o vertical.

- **Restricciones para la Proyección Horizontal:** Conjunto de restricciones pertenecientes a una misma tupla. Una relación con atributos restricción tendrá $G_1 \ldots G_i \ldots G_n$ conjuntos de restricciones, donde cada G_i está formado por todas las restricciones de la tupla i-ésima de la relación, para sus n tuplas.
- **Restricciones para la Proyección Vertical:** Conjunto de restricciones pertenecientes a una misma columna (atributo restricción). Una relación con atributos restricción tendrá $G_1 \ldots G_i \ldots G_m$, donde cada G_i está formado por todas las restricciones del i-ésimo atributo restricción de la relación, para sus m atributos restricción.

6.4.1. Detección de grupos de restricciones relacionadas para la Proyección Numérica

Cuando la proyección es numérica, todas las restricciones que tengan variables relacionadas de forma directa o indirecta con las variables que aparezcan en la proyección participarán en el problema de satisfacción u optimización de restricciones que se creará para su evaluación. Que una restricción esté relacionada de forma directa con una variable significa que dicha variable pertenece a la restricción, mientras que si está relacionada de forma indirecta significará que la restricción tienen una variable en común con otra restricción que está relacionada de forma directa o indirecta con la variable de la proyección. El caso de la representación simbólica no es igual, precisamente por la representación simbólica del valor de las variables mediante restricciones. Para ver más claramente la diferencia entre las restricciones relacionadas para los dos tipos de proyección se utiliza el ejemplo de la figura 6.16. En el caso de obtener la representación simbólica de las variables $R_2.ImpProv$, $R_2.Precio$ y $R_3.ImpInterNac$, sólo será necesario realizar la eliminación

simbólica de la variable $R_2.ImpNac$ trabajando con las restricciones R_2 y R_3, obteniendo una nueva restricción $\{-49.0196{*}ImpInterNac + 50 * ImpProv + Precio = 0\}$, en este caso utilizando las Bases de Gröbner. Pero si se quiere conocer los valores instanciados de dichas variables es necesario incluir R_1 entre las restricciones relacionadas, ya que la variable *ImpNac* depende de la variable *ImpProv*, cuyos valores no sólo están restringidos por la restricción R_2, también por R_1.

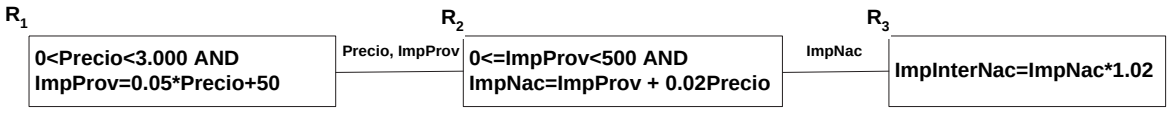

Figura 6.16: Ejemplo de conjunto de restricciones

El algoritmo propuesto para encontrar todas las restricciones relacionadas con las variables de una proyección numérica, está basado en la representación de cada conjunto de restricciones G_i, independientemente de que sea proyección vertical y horizontal, como un grafo. Las restricciones se representan como nodos y las variables como aristas, de forma que existirá una arista entre dos nodos si las restricciones que representan dichos nodos tienen una variable en común, y dicha variables no está instanciada en el predicado. De esta idea de deriva la definición:

Definición 6.1: Conjunto de Restricciones Relacionadas por Variables (CR-RV) para la Proyección Numérica. Máximo conjunto de restricciones donde al menos una variable de cada restricción afecta directa o indirectamente a las variables de la proyección. O lo que es lo mismo, una componente conexa del grafo en los términos anteriormente expuestos.

6.4.2. Detección de grupos de restricciones relacionadas para la Proyección Simbólica

Como se comenta en el capítulo 5, la proyección puede obtener datos intensivos mediante técnicas de sustitución o eliminación de variables. Independientemente de la técnica que se utilice será necesario conocer las relaciones entre las restricciones en función de las variables que las forman y las que se establecen en la proyección, lo que también dependerá de si la proyección es vertical u horizontal. La única diferencia entre la proyección horizontal vertical y horizontal son las restricciones involucradas en dicha búsqueda.

Antes de definir qué es un grupo de restricciones relacionadas por variables en una proyección simbólica, son necesarias algunas notaciones previas. Siendo$c_1, \ldots, c_n$ todas las restricciones de un conjunto G_i, e independientemente de que dicho grupo esté formado por restricciones de la misma tupla o del mismo atributo restricción, y $c_i, \ldots, c_j$ un subconjunto de dichas restricciones:

- **Variables($c_i, \ldots, c_j$).** Conjunto de las variables de las restricciones $c_i, \ldots, c_j$ sin repetición.
- **Variables(Q, $\{c_i, \ldots, c_j\}$).** Conjunto de las variables sin repetición de las restricciones $c_i, \ldots, c_j$ que aparecen en la consulta Q.

Partiendo de dicha notación, se buscará un conjunto de restricciones que estén relacionadas entre sí por compartir variables que tengan que ser eliminadas. Para un conjunto de restricciones G_i formado por todas las restricciones de una tupla $\{c_1, \ldots, c_n\}$, un subconjunto G_i' formada por las restricciones $\{c_i, \ldots, c_j\} \subseteq \{c_1, \ldots, c_n\}$, se le denominará **Conjunto de Restricciones Relacionadas por Variables (CRRV) para la Proyección Simbólica** si, cada variable de las restricciones de G_i': aparece en la proyección; o está al menos en dos restricciones de G_i'; o sólo pertenece a una restricción de G_i. Además este conjunto debe ser mínimo, lo que significa que no puede existir ningún CRRV incluido dentro de otro.

Definición 6.2: Conjunto de Restricciones Relacionadas por Variables (CRRV) para la Proyección Simbólica. G_i' será un Conjunto de Restricciones Relacionadas por Variables si:

Siendo G_i'$\equiv \{c_i, \ldots, c_j\} \mid (G_i \equiv \{c_1, \ldots, c_n\} \wedge G_i' \subseteq G_i)$
$\forall\, c_k \in G_i$'
  $\forall\, v \in$ Variables(c_k) $\mid v \notin$ Variables(Q, c_k) $\Rightarrow$
    ($v \in$ Variables(G_i'$-\{c_k\}$)) $\vee$ ($\nexists\, c_l \in (G_i - \{c_k\}) \mid v \in$ Variables(c_l))
$\wedge$
Variables(Q,$\{c_i, \ldots, c_j\}$) $\neq \emptyset$
$\wedge$
$\nexists\, G$"$\subset G_i$' $\mid G$" sea un CRRV

Ejemplo de Proyección Horizontal

Este tipo de proyección se hará cuando los atributos de la proyección pertenecen a distintos atributos restricción, lo que significa que se realizará la búsqueda de restricciones relacionadas entre las restricciones de una misma tupla.

Un ejemplo de proyección simbólica horizontal sobre la relación de la figura 6.14 puede ser:

```
SELECT ImpuestoInternacionales.id,
  CONSTRAINTS (ImpuestosProvinciales.Impuestos.ImpInterNac,
  ImpuestosInternacionales.Impuestos.Precio)
FROM ImpuestosProvinciales, ImpuestosInternacionales
```

Lo que significa que un CRRV estará formado por un conjunto de restricciones de una misma tupla, donde sus variables pertenecen a las variables de la proyección Q, o aparecen en otra restricción de dicho conjunto, o no aparecen en ninguna restricción de dicha tupla. El algoritmo que obtendrá los CRRV para la proyección simbólica también se basa en la representación de cada conjunto de restricciones G_i como un grafo, donde las restricciones se representan como nodos y las variables como aristas, de forma que existirá una arista entre dos nodos si las restricciones que representan dichos nodos tienen una variable en común que no pertenezca a la proyección. Para buscar la similitud con la teoría de grafos, al igual que para la proyección numérica, primero se buscarán las componentes conexas pero con la diferencia de que las variables de la proyección no se representarán como aristas. El segundo paso será buscar los subconjuntos mínimos de componentes que permitan eliminar las variables que no aparecen en la proyección. En la figura 6.17 se muestran los conjuntos de restricciones para cada una de las tuplas del ejemplo de la figura 6.14, junto a la relación entre las distintas restricciones y las variables que las relacionan.

Los CRRV del ejemplo para las variables de la proyección *Precio* e *ImpInterNac* serán: $\{R_1, R_7, R_{13}\}$, $\{R_2, R_8, R_{14}\}$, $\{R_3, R_9, R_{15}\}$, $\{R_4, R_{10}, R_{16}\}$, $\{R_5, R_{11}, R_{17}\}$ y $\{R_6, R_{12}, R_{18}\}$, ya que forman un conjunto de restricciones donde todas las variables que no aparecen en la consulta pertenecen a otra restricción de dicho conjunto. Los detalles relativos al algoritmo están descritos en el apéndice A.

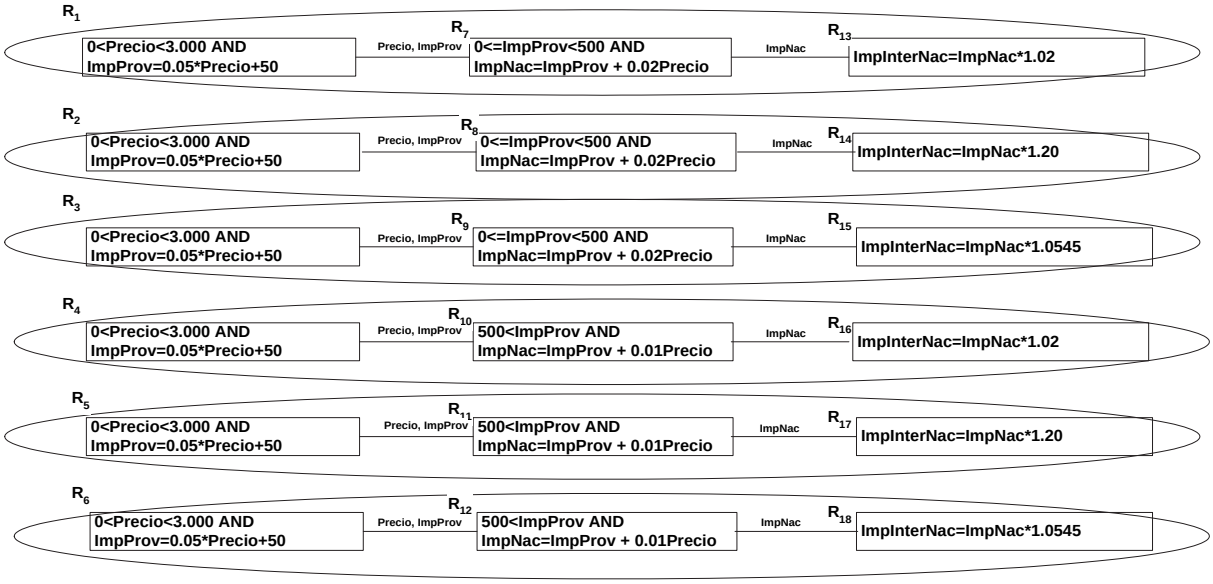

Figura 6.17: Grafo de Restricciones de la Relación para Proyección Horizontal

Ejemplo de Proyección Vertical

Una vez explicada la proyección horizontal, es fácil introducir la proyección vertical, ya que en lo único que se diferencian es que en la proyección vertical las restricciones sobre las que se realiza la búsqueda pertenecen a un mismo atributo restricción.

En la figura 6.18 se presenta un ejemplo muy similar al anterior con respecto a las restricciones, con la diferencia de que todas las restricciones están en una misma columna. En este caso se realizará una proyección vertical, donde las restricciones han sido nombradas para que sea más fácil la explicación de las relaciones entre variables y restricciones.

	Impuestos
R_1	0<=Precio<3.000 AND ImpProv=0.05*Precio+50
R_2	0<=ImpProv<500 AND ImpNac=ImpProv + 0.02Precio
R_3	500<ImpProv AND ImpNac=ImpProv + 0.01Precio
R_4	ImpInterNac=ImpNac*1.02
R_5	ImpInterNac=ImpNac*1.0354

Figura 6.18: Ejemplo para realizar proyección Vertical

Un ejemplo de proyección sobre la relación de la figura 6.18 puede ser:

```
SELECT CONSTRAINTS (ImpuestosInternacionales.Impuestos.ImpInterNac,
  ImpuestosInternacionales.Impuestos.Precio)
FROM ImpuestosInternacionales
```

En la figura 6.19 se muestra el grafo donde se representan las restricciones como nodos y las variables que las relacionan como aristas.

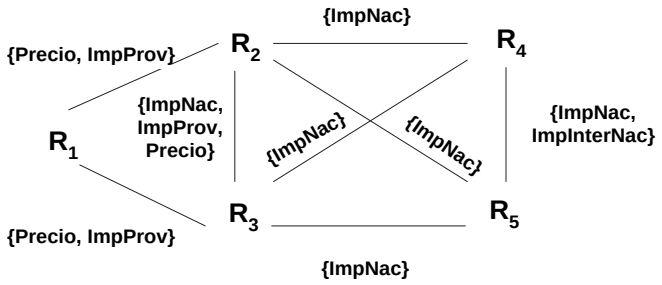

Figura 6.19: Grafo del Ejemplo para la proyección Vertical

Pese a que sólo hay una componente conexa en este grafo, existen diferentes CRRV, ya que cumplen la definición 6.2. Los CRRV para este ejemplo son: {R_1, R_2, R_4}, {R_1, R_2, R_5}, {R_1, R_3, R_4} y {R_1, R_3, R_5}.

Eliminación o Sustitución Simbólicas de Variables

Cuando un CRRV tiene una variable que sólo aparece en una restricción del conjunto G_i, y no pertenece a la proyección, no será posible realizar sustitución simbólica, ya que dicha variable no puede ser representada mediante otras de la proyección. Para los casos donde sólo se quieran obtener nuevas restricciones utilizando sustitución de variables, los CRRV no podrán estar formados con este tipo de variables. A este nuevo CRRV se le denominará **Conjunto Completo de Restricciones Relacionadas por Variable (CCRRV)**, y será de la forma:

Definición 6.3: Conjunto Completo de Restricciones Relacionadas por Variables (CCRRV). G' será un Conjunto Completo de Restricciones Relacionadas por Variables si:

Siendo G_i'$\equiv \{c_i, \ldots, c_j\} \wedge G_i \equiv \{c_1, \ldots, c_n\} \wedge G_i' \subseteq G_i$
$\forall\ c_k \in G_i$'

$$\forall\, v \in \text{Variables}(c_k) \mid v \notin \text{Variables}(Q, c_k) \Rightarrow$$
$$(v \in \text{Variables}(\text{G}_i\text{'} - \{c_k\}))$$
$$\wedge$$
$$\text{Variables}(Q, \{c_i, \ldots, c_j\}) \neq \emptyset$$
$$\wedge$$
$$\nexists\ \text{G''} \subset \text{G}_i\text{'} \mid \text{G''} \text{ sea un CRRV}$$

La sintaxis de este tipo de proyección es:

```
SELECT FULL CONSTRAINTS (<lista de Variables>)
FROM relación WHERE predicado
```

Esta nueva definición permite diferenciar al usuario cuándo quiere hacer eliminación de variables, y cuándo quiere hacer sustitución de variables siempre que sea posible. Si aparece la palabra reservada `FULL` y se obtienen los CCRRV, significará que se quiere hacer sustitución de variables, ya que todas las variables que no aparecen en la proyección pueden ser sustituidas por variables de la proyección. Mientras que si no aparece dicha palabra reservada, significará que al usuario no le importa obtener nuevas restricciones simplemente haciendo eliminación de variables, utilizando Descomposición Algebraica Cilíndrica y eliminación de cuantificadores.

Esta operación es especialmente interesante en casos de estudios como la diagnosis basada en modelos, el cual se explicará con detalle en el siguiente capítulo.

6.4.3. Selección de la técnica para evaluar la proyección

Una vez conocidos los Conjuntos de Restricciones Relacionadas por Variables, es necesario construir un modelo y seleccionar la técnica que lo resolverá. Un tipo de proyección u otra, dependerá de la sintaxis que utilice el usuario cuando realice la consulta. Dentro de esta división (intensiva y extensiva) se utilizarán distintas técnicas en función de los conjuntos de restricciones relacionadas con las variables que se quieran obtener. Recordando que la sintaxis de la proyección es de la forma:

```
SELECT [ [FULL] CONSTRAINTS | VALUES | MIN VALUE | MAX VALUE]
  (<lista_de_atributos>)
FROM relación WHERE predicado
```

Las palabras reservadas `MIN` y `MAX`, denotarán la construcción de un problema de optimización de restricciónes, mientras que la palabra reservada `VALUES` definirá el deseo del usuario de conocer los valores instanciados de las variables. Por otra parte, la palabra reservada `CONSTRAINTS` describe la necesidad de utilizar técnicas simbólicas para mostrar el resultado en forma de restricción. Para conocer qué técnica es necesaria, y cómo construir el modelo con más detalle, se utiliza el árbol de decisión presentado en la figura 6.20, en función de la proyección de entrada y a los CRRV obtenidos en la sección 6.4.1 y 6.4.2.

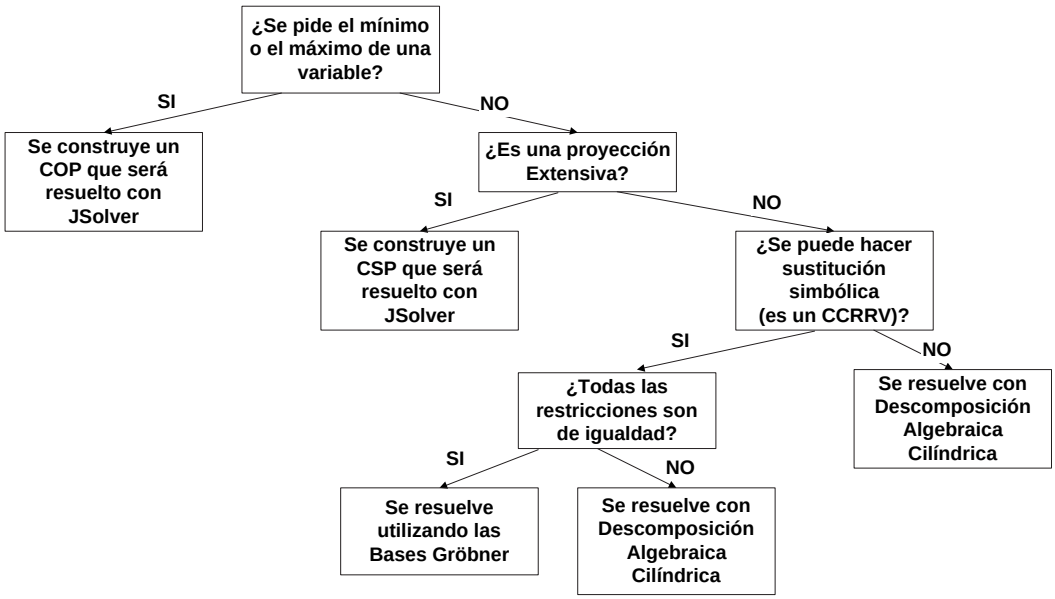

Figura 6.20: Árbol de decisión para construir el modelo y decidir qué técnica lo resolverá

La secuencia de decisiones que se reflejan en dicho árbol son:

- **¿Se pide el mínimo o el máximo de una variable en la consulta?**La respuesta a esta pregunta será *sí*, cuando en la proyección aparezcan las palabras `MIN VALUE` o `MAX VALUE`. Este tipo de problema se resuelve contruyendo un COP.
- **¿Es una proyección Extensiva?** La respuesta a esta pregunta será *sí*, cuando en la proyección aparezca la palabra reservada `VALUES`, lo que significará que se buscan valores concretos de las variables, lo que se resolverá con la construcción de un CSP. Si aparecen la palabra reservada `CONSTRAINTS` la respuesta será *no*, y se resolverá la proyección de manera simbólica.
- **¿Se puede hacer sustitución simbólica?** La respuesta a esta pregunta será *sí* cuando el conjunto de restricciones relacionadas por variables es completo (CCRRV),

lo que significa que todas las variables o están al menos en dos restricciones o pertenecen a la proyección y pueden participar en la solución. Si se utiliza la palabra reservada `FULL` sólo los CCRRV serán tratados, y los CRRV serán ignorados. Si no aparece dicha palabra, se tratarán tanto los CRRV como los CCRRV.

- **¿Todas las restricciones son de igualdad?** Si todas las restricciones están etiquetadas como `LinEq` o `PolEq`, significa que se pueden utilizar las Bases de Gröbner para obtener la nueva restricción como resultado, en caso contrario se utilizará la Descomposición Algebraica Cilíndrica junto a la eliminación de cuantificadores.

Construcción de Modelos (Hojas del árbol de decisión)

Una vez conocidos los CRRV, o los CCRRV, y qué tipo de modelo hay que construir, es el momento de construir dicho modelo. Para la problemática propuesta existen cuatro tipos: Problemas de Optimización de Restricciones, Problemas de Satisfacción de Restricciones, sustitución simbólica con Gröbner, o eliminación de cuantificadores con la Descomposición Algebraica Cilíndrica.

1. **Construcción de Problemas de Optimización de Restricciones**

   Como ya se ha mencionado en este mismo capítulo, la sintaxis de este tipo de proyección será de la forma:

   ```
   SELECT [MIN VALUE | MAX VALUE] (<atributo_variable_de_restricción>)
   FROM relación WHERE predicado
   ```

   Definiéndose *Lista_CRRV* como la lista con los CRRV obtenidos en la sección 6.4.1 ($\text{CRRV}_1 \ldots \text{CRRV}_n$), y teniendo en cuenta si existen variables instanciadas en el predicado de la forma `relación.AtribRestricción.variable=constante` o `relación.AtribRestricción.variable=relación.AtribRestricción.variable`, se construirá un COP. Se nombrarán las restricciones de cada CRRV de la lista de la forma CRRV_1.Restricciones ... CRRV_n.Restricciones. El COP que se construye será una combinación disyuntiva de cada CRRV, mientras que entre las restricciones de un mismos CRRV existirá una relación conjuntiva. Esto hará posible obtener el

máximo o mínimo valor de la variable analizando todos los CRRV. La estructura del COP será de la forma:

```
añadir al COP las variables
  {Variables(CRRV_1.Restricciones) ... Variables(CRRV_n.Restricciones)}
añadir al COP
  (Maximize(atributo variable de restricción)
      ∨
  Minimize(atributo variable de restricción))
añadir al COP las restricciones
  {CRRV_1.Restricciones ∨ ... ∨ CRRV_n.Restricciones}
relación.AtribRestricción.variable=constante
  ...
relación.AtribRestricción.variable=relación.AtribRestricción.variable
```

Este COP se resolverá con JSolver^{TM} [78], aunque como se ha comentado en el capítulo anterior se podría utilizar otro resolutor. La salida que se mostrará al usuario será sólo el valor de la variable objetivo de la proyección.

2. **Construcción de Problemas de Satisfacción de Restricciones**

Para obtener valores instanciados de un conjunto de variables se construirá un problema de satisfacción de restricciones, que será similar a la construcción de problemas de optimización de restricciones con respecto al CRRV, pero sin incluir el objetivo de maximizar o minimizar.

Como ya se ha mencionado en este mismo capítulo, la sintaxis de este tipo de proyección será de la forma:

```
SELECT VALUES (<lista_de_Variables>) FROM relación WHERE predicado
```

Al igual que en la optimización de variables, *Lista_CRRV* representa la lista de CRRV obtenidos en la sección 6.4.1. También es necesario tener en cuenta si existen variables instanciadas en el predicado de la forma `relación.AtribRestricción.variable=constante` o

relación.AtribRestricción.variable=relación.AtribRestricción.variable

Con toda esta información, se construirá un CSP de la forma:

```
añadir al CSP las variables
  {Variables(CRRV_1.Restricciones) ... Variables(CRRV_n.Restricciones)}
añadir al CSP las restricciones
  {CRRV_1.Restricciones ∨ ... ∨ CRRV_n.Restricciones}
relación.AtribRestricción.variable=constante
  ...
relación.AtribRestricción.variable=relación.AtribRestricción.variable
```

Para la arquitectura que se propone en esta memoria de tesis, este CSP se resolverá con JSolver^{TM} [78] y la salida que se mostrará al usuario será el valor de las variables que pertenecen a la `lista de Variables`. Como es posible que el número de soluciones sea muy grande o incluso infinito, por defecto se presentará sólo una solución, aunque se podrá modificar en la misma sentencia para N tuplas, de la forma:

```
SELECT VALUES[N](<lista_de_Variables>) FROM relación WHERE predicado
```

3. **Construcción de Problemas de Eliminación de Variables**

Este tipo de problemas se construirá cuando se quiere obtener una relación simbólica entre variables en lugar de sus valores concretos. Se aplicará cuando los CRRV no son completos, o existe al menos una inecuación polinómica entre las restricciones del CRRV. La sintaxis de este tipo de proyección será de la forma:

```
SELECT CONSTRAINTS(<lista_de_Variables>) FROM relación WHERE predicado
```

Siendo *Lista_CRRV* el conjunto de restricciones relacionadas obtenido en la sección 6.4.2 en función de la lista de variables de la proyección, se construirá una función que será resuelta con Mathematica v.5^{TM}. El prototipo de la función es el siguiente:

```
Reduce [
  Exists[{Variables que se quieren eliminar}, restricciones],
  {lista de todas las Variables}
]
```

En este caso, para cada $\text{CRRV}_i \in \{\text{CRRV}_1, \ldots, \text{CRRV}_n\}$, los cuales son los CRRV que forman Lista_CRRV, se construirá:

```
Reduce [
Exists
 [{Variables(CRRV_i.Restricciones)−Variables(Q, CRRV_i.Restricciones)},
  CRRV_i.Restricciones], {Variables(CRRV_i)}
]
```

La construcción de esta sentencia se genera de forma automática y transparente para el usuario, como en el resto de los modelos. La salida que se mostrará al usuario será el valor de salida de dicha función.

4. **Construcción de Problemas de Sustitución Simbólica con Gröbner**

En este caso, todas las variables que aparecen en una sola restricción también aparecen en la consulta, por lo que estamos seguros de poder hacer sustitución simbólica sin perder información de las variables de dicha consulta, lo que significa que los CRRV son completos. Además las restricciones tienen que estar formadas por ecuaciones lineales o polinómicas, siendo la sintaxis de la sentencia de la forma:

```
SELECT FULL CONSTRAINTS (<lista_de_Variables>) FROM relación
```

La sintaxis de la función, que también será resuelta por Mathematica v.5^{TM}, es:

```
GroebnerBasis({Restricciones polinómicas de igualdad},
  {Variables de salida}, {Variables a eliminar})
```

Tras obtener los CCRRV, se construirá de forma automática un modelo para cada $\text{CCRRV}_i \in \textit{Lista\_CRRV}$ de la forma:

```
GröbnerBasis({CRRV_i.Restricciones},
  {Variables(Q, CRRV_i.Restricciones)},
  {Variables(CRRV_i.Restricciones)−Variables(Q, CRRV_i.Restricciones)})
```

La salida que se mostrará al usuario será el valor de salida de dicha función para cada uno de los CCRRV, ya a los CRRV que no sean completos no se les puede hacer sustitución de variables.

6.5. Implementación de la Operación de Unión

Esta operación tiene dos relaciones como parámetros de entrada (R_1, R_2) donde ambas relaciones tienen que ser equivalente con respecto a la unión. La relación que se obtiene como salida tendrá todas las tuplas de R_1 y R_2 sin repetición. En el caso de que las relaciones que se vayan a unir tengan atributos restricción, será necesario una comprobación especial de dichos atributos para que no aparezcan tuplas duplicadas en la relación de salida. Para hace más eficiente la implementación, se separará el tratamiento de los atributos univaluados de los atributos restricción. Los pasos de dicha comprobación se realizará entre todos los pares de tuplas $t_1 \in R_1$ y $t_2 \in R_2$, y serán:

1. Si todos los atributos clásicos entre dos tuplas son iguales, se pasa al paso2, en caso contrario ambas tuplas formarán parte de la solución. Esta comprobación se hará mediante el álgebra relacional clásica.

2. Si todas las restricciones de ambas tuplas tienen las mismas variables, se pasará al paso 3, en caso contrario ambas tuplas formarán parte de la solución. Esta comprobación se hará utilizando exclusivamente el álgebra relacional clásica, ya que dicha información está almacenada en la tabla *Restricciones/Variables*.

3. Entre las tuplas se comparan sus atributos restricción como se ha explicado en la sección 4.7. En dicha sección se analiza la unión de tuplas utilizando la operación de inclusión entre restricciones ($\subseteq$), lo que gracias a tener almacenado el mínimo y máximo valor de cada variable se puede hacer de una manera más eficiente, evitando en algunos casos la creación de CSP. Si existe más de un atributo restricción en las relaciones R_1 y R_2 se comparará cada i-ésimo atributo restricción de t_1 con el atributo restricción i-ésimo de t_2, existiendo dos casos:

   - Si existe alguna restricción perteneciente a t_1 con soluciones que no estén en el atributo restricción de t_2, y viceversa (los casos a y b de la figura 6.21): t_1 y t_2 formarán parte de la relación de salida
   - En caso contrario, significará que todas las relaciones serán de la forma c o d de la figura 6.21, por lo tanto una nueva tupla formará parte de la relación de salida. Dicha tupla tendrá como atributos clásicos los mismos que t_1 y t_2, y como atributos restricción la restricción de t_2 (caso d) o de t_1 (caso c).

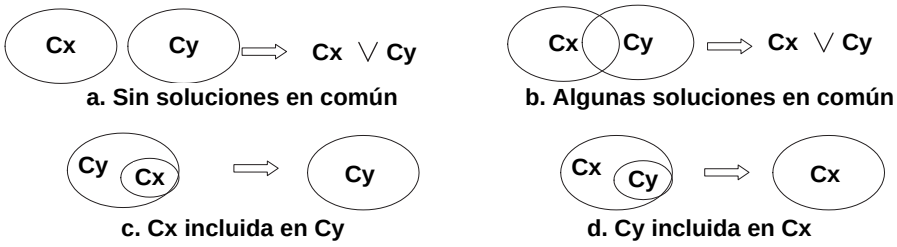

Figura 6.21: Tipos de unión entre restricciones

6.6. Implementación de la Operación de Diferencia

En el caso de la diferencia sobre los relaciones (R_1, R_2), donde ambas relaciones tienen que ser equivalente con respecto a la diferencia, la relación que se obtiene como salida tendrá todas las tuplas de R_1 que no pertenezcan a R_2. En el caso de que las relaciones que se vayan a unir tengan atributos restricción, será necesario comprobar las soluciones que tienen en común cada una de las restricciones de R_1 con respecto a las de R_2. Para hace más eficiente la implementación, se separará el tratamiento de los atributos univaluados de los atributos restricción. Los pasos de dicha comprobación se realizará entre todos los pares de tuplas $t_1 \in R_1$ y $t_2 \in R_2$, y serán:

1. Si todos los atributos clásicos entre las dos tuplas son iguales, se pasa al paso2, en caso contrario la tupla t_1 formarán parte de la solución. Esta comprobación se hará mediante el álgebra relacional clásica.

2. Si todas las restricciones de ambas tuplas tienen las mismas variables, se pasará al paso 3, en caso contrario la tupla t_1 formará parte de la solución. Esta comprobación se hará mediante el álgebra relacional clásica gracias a tener las variables indexadas en la tabla *Restricciones/Variables*.

3. Entre las tuplas se comparan sus atributos restricción como se ha explicado en la sección 4.8. En dicha sección se estudia la diferencia de tuplas analizando si existen alguna solución para dos restricciones, equivalente al operado & explicado, lo que gracias a tener almacenado el mínimo y máximo valor de cada variable se puede hacer de una manera más eficiente, evitando en algunos casos la creación de CSP. Si existe más de un atributo restricción en las relaciones R_1 y R_2 se comparará cada

i-ésimo atributo restricción de t_1 con el atributo restricción i-ésimo de t_2, existiendo dos casos:

- Si existe alguna restricción del atributo restricción de t_1 (C_x) que tenga una solución no perteneciente al atributo restricción de t_2 (C_y)(los casos a, b y c de la figura 6.22):

  Una nueva tupla formará parte de la relación de salida. Dicha tupla tendrá como atributos clásicos los mismos que t_1 y t_2, y como atributos restricción la restricción de t_1 (caso a) o una nueva restricción con las soluciones de t_1 que no pertenecen a t_2 (caso b y c).

- En caso contrario, significará que todas las relaciones serán de la forma$d$ de la figura 6.22: La tupla t_1 no formará parte de la salida.

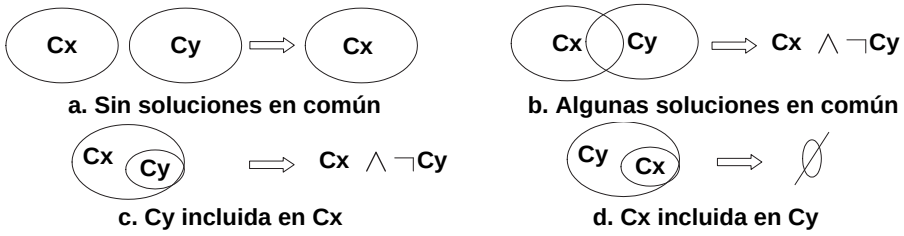

Figura 6.22: Tipos de diferencias entre restricciones

6.7. Resumen

En este capítulo se presenta, para cada una de las cinco operaciones primitivas del álgebra relacional, los detalles de implementación necesarios para evaluarlas de una manera eficiente y acorde con la arquitectura propuesta.

Para la selección se propone utilizar el álgebra relacional clásica para los atributos univaluados, las envolventes para la parte del predicado relacionada con atributos variables de restricción y para los atributo restricción, y por último construir un CSP en los casos en que sea necesario, dada la elevada complejidad de resolución éstos. Todas estas decisiones son tomadas con el objetivo de conocer las tuplas que cumplen un determinado predicado con el menor gasto computacional posible.

Para la proyección se divide el problema en dos etapas: búsqueda de restricciones de manera horizontal o vertical; y búsqueda de restricciones relacionadas por variables en función de si la evaluación es simbólica o numérica. Para cada uno de estos casos, se describe cómo se crearán cada uno de los modelos dependiendo del tipo de restricciones involucradas y el tipo de consulta.

Por último se presenta la diferencia y la unión, minimizando el número de CSP que se tienen que construir para la evaluación gracias a: el análisis de las envolventes de las variables de cada una de las restricciones; aprovechando el álgebra relacional clásica para los atributos univaluados; y analizando los rangos de las variables para conocer las soluciones que comparten las restricciones.

Capítulo 7

La Diagnosis de Fallos Basada en Modelos: Un caso de estudio

Aunque ha sido el campo de los sistemas geográficos el más utilizado como caso de estudio en las BDdR, también se pueden encontrar innovadores estudios en el campo de la genética [124], o en el campo de *data mining* [97]. La diagnosis de fallos se considera una área donde también las BDdR pueden aportar grandes ventajas. En este capítulo se introducirá el problema de la diagnosis de fallos basada en modelos como otro ejemplo de utilización de las Bases de Datos de Restricciones. El proceso de diagnosis detecta qué componentes fallan en función de los valores obtenidos de un conjunto de sensores incluidos en el sistema. Cuando el comportamiento de los componentes que conforman el sistema se puede representar con restricciones polinómicas, se puede utilizar una LORCDB para el tratamiento y evaluación de la información. Se ha utilizado la diagnosis de fallos como un caso de uso en las BDdR debido a que tiene características idóneas, ya que los valores de las variables no son únicos, dependen de las entradas, y su funcionamiento y configuración del sistema puede ser descrito mediante restricciones. Entre las razones que han llevado a la utilización de la diagnosis de fallos como caso de estudio se encuentran:

- Un sistema puede estar formado por muchos componentes, por lo que una base de datos es una estructura ideal para almacenar toda la información con el objetivo de agilizar su tratamiento.
- Los distintos componentes se relacionan entre ellos mediante variables, a lo cual da soporte las LORCDB mediante la indexación entre restricciones y variables.

- En la diagnosis de sistemas sólo se conocen los valores de las variables que tienen sensores, de forma que pese a que el sistema puede ser el mismo, las restricciones que representan su funcionamiento dependen de la localización de dichos sensores. Esta es la misma idea desarrollada en la proyección sobre atributos variable de restricción, donde sobre una misma relación de entrada es posible realizar diferentes proyecciones dependiendo de los atributos que se quieran obtener.
- Dependiendo del tipo de restricción que describa el comportamiento del componente y del tipo de información que se quiera obtener, es necesario utilizar diferentes técnicas. La acción de decidir y ejecutar dichas técnicas se ve facilitada por la utilización de la arquitectura LORCDB y el lenguaje CORQL. Esto significa que un proceso tan complejo como es el de la diagnosis de fallos, se puede realizar simplemente mediante un conjunto de consultas a la base de datos.

Para introducir la problemática de la diagnosis basada en modelos, en principio se utiliza un sistema sencillo y muy difundido en el área de la diagnosis de fallos. Este ejemplo ayudará a explicar las definiciones relativas al paradigma de la diagnosis basada en modelos. Una vez que el problema y los pasos de la diagnosis han sido introducidos, se presenta un sistema más complejo, tanto en el número de componentes, como en el tipo de restricciones que lo describen y por lo tanto el tipo de consultas que se pueden realizar. Sobre este ejemplo más complejo, se introducirá cómo las distintas técnicas basadas en problemas de satisfacción de restricciones y eliminación de variables hacen posible el proceso de la diagnosis de fallos.

7.1. Conceptos sobre Diagnosis de Fallos

El área de la diagnosis de fallos es un campo emergente desde el punto de vista de las empresas, ya que un sistema que falla puede conllevar grandes pérdidas económicas. La diagnosis basada en modelos permite determinar, basándose en su monitorización, por qué un sistema correctamente diseñado no funciona como se espera. De esta forma abarca la detección de los componentes cuyo comportamiento no corresponde a los esperados por el diseñador del sistema.

Los sistemas a diagnosticar mediante modelos son especialmente idóneos para su tratamiento mediante DBdR, ya que se pueden representar mediante un conjunto de res-

tricciones (componentes) relacionadas mediante variables (uniones entre componentes). Analizando un sistema de componentes como un conjunto de datos relacionados mediante las variables que contiene, sería posible obtener el comportamiento de una parte del sistema en función de sus variables, o saber qué valores de salida tendría el sistema para una entrada determinada, facilitando la labor del ingeniero. Heredando toda la potencia de las bases de datos relacionales, y utilizando BDdR también sería posible alterar o ampliar el sistema almacenado con pocas modificaciones, facilitando el acceso a sistemas formados por gran cantidad de componentes.

Antes de continuar con los detalles que relacionan las BDdR y la diagnosis de fallos, es necesario hacer una introducción a las dos propuestas de diagnosis en las que se basan los ejemplos utilizados en este capítulo, FDI y DX. FDI (Fault Detection and Isolation) [80][115] consiste en el análisis de un modelo para identificar las relaciones entre los componentes que lo forman, determinando qué componentes del sistema afectan a cada variable observable (con sensores), utilizando el análisis estructural [142][123]. De esta forma, FDI define una firma de fallo para cada uno de los posible errores del sistema, sólo teniendo en cuenta los fallos simples, lo que significa que sólo puede fallar un componente del sistema al mismo tiempo. FDI realiza dicho análisis estructural dependiendo de la configuración del sistema, por lo que dicho proceso se hará fuera de línea (información precompilada). DX, al contrario que FDI, da cabida a la detección de fallos múltiples cuyos primeros desarrollos se publicaron en los trabajos [40][127][89]. Estos trabajos tratan de descubrir discrepancias entre el comportamiento observado de un sistema y el comportamiento esperado según un modelo. La propuesta DX se caracteriza por la diagnosis de un sistema definiendo un conjunto mínimo de componentes que pueden fallar en función del comportamiento observado (síntomas), por lo que será un proceso en línea. Sin duda fueron los trabajos de De Kleer y Williams [90] y Reiter [127] los que formalizaron el problema de la diagnosis de fallos, junto a la definición de propuestas para hacer el proceso más eficiente evitando búsquedas innecesarias. El tipo de diagnosis sobre la que se basan los ejemplos de las consultas que se describirán en este capítulo, utiliza una combinación de ambas técnicas (DX y FDI) [55] seleccionando las características más ventajosas de ambas.

También hay que tener en cuenta que entre las soluciones dadas para la diagnosis, se suele pasar por alto la importancia del almacenamiento del sistema a diagnosticar. El inconveniente es que no existen aportaciones que permitan tratar las restricciones y sus variables como un dato consultable en una base de datos relacional, por lo que la creación

y la diagnosis de sistemas de gran tamaño resulta engorroso y complicado. El uso de las BDdR en el campo de la diagnosis hace los sistemas más versátiles, conlleva persistencia, facilita la composición de los sistema, disminuye la utilización de memoria y hace más eficiente el proceso de diagnosis ya que agiliza el acceso a la información.

Como se ha expuesto con anterioridad, el uso de SQL sobre restricciones hace posible obtener diferentes modelos en función de la consulta, en este capítulo se presenta cómo la arquitectura LORCDB puede almacenar las restricciones que describen el comportamiento y la relación entre los componentes de un sistema. Junto a esto se presentan qué consultas ayudan a realizar la diagnosis sin necesidad de crear una aplicación específica para su tratamiento, ya que todas la técnicas y lógica de inferencia de conocimiento están incluidas en la arquitectura. La localización de los sensores definirá qué variables son observables y cuáles deben ser inferidas en función del resto (no observables). La arquitectura LORCDB permite consultar estos sistemas equivalentes, simulando la colocación de los sensores en lugares diferentes e infiriendo nuevas restricciones. La colocación de los sensores definirá qué parte del sistema se conoce, o lo que es lo mismo, qué variables se quieren obtener mediante la proyección.

7.2. Definiciones y Notación

Para la comprensión clara del caso de estudio son necesarias algunas definiciones. Las definiciones y la notación usadas están basadas en los conceptos desarrollados por la comunidad de diagnosis, combinando la lógica DX y las firmas de fallos (FDI).

Para facilitar la comprensión de las definiciones, se utilizará un ejemplo sencillo conocido en el campo de la diagnosis mostrado en la figura 7.1. En este ejemplo los componentes A_i representan sumadores, mientras que los componentes M_i representan multiplicadores.

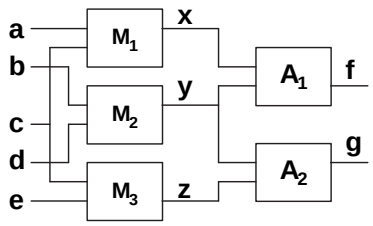

Figura 7.1: Ejemplo introductorio

Definición 7.1. Modelo del Sistema Polinómico - MSP (System Polynomial Model - SPM). Puede ser definido como un conjunto finito P de restricciones polinómicas o lineales las cuales describen el comportamiento del sistema. Para diagnosticar dicho sistema es necesario definir qué variables serán no observables (V_{nob}) y cuales observables (V_{ob}), que serán las que tengan sensores. De esta forma, un MSP está formado por la tupla $\{P, V_{ob}, V_{nob}\}$.

Para el ejemplo de la figura 7.1, las restricciones serán las descritas en la tabla 7.1, donde las variables observables son {a, b, c, d, e, f, g}, y las no observables {x, y, z}.

Nombre	restricción
M_1	$a * c = x$
M_2	$b * d = y$
M_3	$c * e = z$
A_1	$x + y = f$
A_2	$y + z = g$

Tabla 7.1: Modelo del Sistema

Definición 7.2. Contexto. Cualquier subconjunto de componentes que componen el sistema. El número de los posibles contextos es $2^{nComp} - 1$, donde $nComp$ es el número de componentes del sistema.

Definición 7.3. Red de Contextos (RC). Un grafo formado por todos los Contextos del sistema, acorde con ATMS [88]. Los nodos del grafo son los Contextos y existirá una arista entre dos Contextos si todos los componentes de uno de ellos están contenidos en el otro, y el contexto menor tiene sólo un elemento menos que el otro contexto. La creación de estas aristas facilitará la diagnosis final. Para el ejemplo la RC es la mostrada en la figura 7.2, donde se muestra un grafo donde los nodos son todos los Contextos.

Definición 7.4. Restricciones de Contextos con Redundancia Analítica. Estas restricciones son más conocidas por sus nombre en inglés (Context Analytical Redundancy Constraint - CARC), y definen restricciones derivadas de las restricciones originales del sistema. Estas restricciones se pueden obtener utilizando técnicas simbólicas para la eliminación de las variables no observables. Las restricciones de contexto con redundancia analítica son restricciones equivalentes a las originales, pero que sólo relacionan las variables definidas como observables.

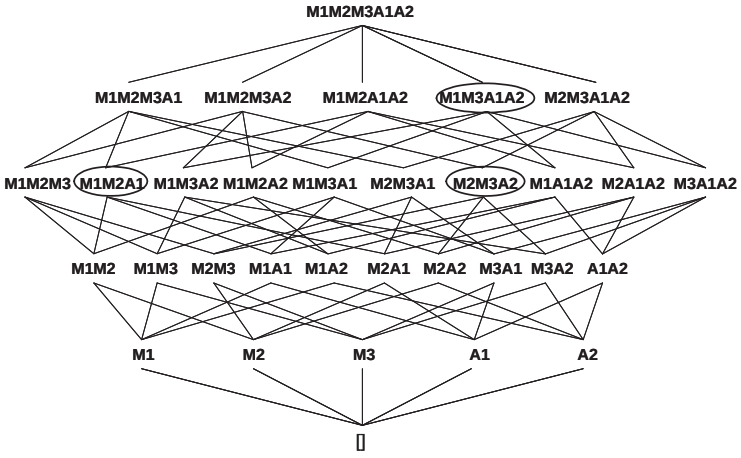

Figura 7.2: Red de Contextos

Definición 7.5. Posible Contexto Conflictivo Mínimo (PCCM). Cuando es posible obtener un CARC con las restricciones asociadas a los componentes de un Contexto, se dirá que dicho Contexto es un Posible Contexto Conflictivo Mínimo. Que sea mínimo significa que para que un Contexto sea un PCCM, ninguno de sus subcontextos pueden tener asociados las mismas CARC que él.

Los Posibles Contextos Conflictivos Mínimos (PCCM) obtenidos, y los CARC para el ejemplo de la tabla 7.1, son los siguientes:

- CARC_1 $\{f = a * c + b * d\}$: Generado con las restricciones de los componentes $\{M_1, M_2, A_1\}$
- CARC_2 $\{g = b * d + c * e\}$: Generado con las restricciones de los componentes $\{M_2, M_3, A_2\}$
- CARC_3 $\{f - g = a * c - c * e\}$: Generado con las restricciones de los componentes $\{M_1, M_3, A_1, A_2\}$

Donde las restricciones asociadas a dichos PCCM se obtendrán mediante la eliminación de las variables no observables, utilizando en este caso las Bases de Gröbner por ser restricciones polinómicas de igualdad. Los PCCM están marcados en la figura 7.2 mediante círculos, donde se puede observar que por ejemplo {M2, M3, A2} $\subset$ {M2, M3, A1, A2}, ambos tienen la CARC_2 asociada, pero se selecciona {M2, M3, A2} como PCCM por ser mínimo.

Aquellos contextos cuyas Restricciones de Contextos con Redundancia Analítica no sean vacías, y no estén en otro contexto menor, serán los Posibles Contextos Conflictivos Mínimos. Para obtener los PCCM no es necesario estudiar las $2^{nComp} - 1$ combinaciones, sólo aquellas de las cuales se puedan obtener las restricciones de contextos con redundancia analítica [55].

Definición 7.6. Modelo Observable (MO). Conjunto de valores definidos para un subconjunto de las variables del sistema. Las variables cuyo valor es conocido, se definirán como variables observables. Estas variables son las encargadas de mostrar los síntomas de un sistema, y en función de sus valores será posible realizar la diagnosis.

Definición 7.7. Contexto Conflictivo Mínimo (CCM). Cuando tras aplicar un modelo observable alguna de las CARC no es satisfactibles, se dice que el o los Posibles Contextos Conflictivos Mínimos a los que están asociados dichas CARC son Contextos Conflictivos Mínimos. La diagnosis del sistema se basará en los componentes asociados a los Contextos Conflictivos Mínimos.

Definición 7.8. Hitting Set (HS). Sea una colección de conjuntos de componentes $\mathcal{C}$ los Contextos Conflictivos Mínimos para un modelo observable, un hitting set es un conjunto $\mathcal{H} \subseteq \bigcup_{\mathcal{S} \in \mathcal{C}} \mathcal{S}$ tal que $\mathcal{H}$ contiene al menos un elemento de cada $\mathcal{S} \in \mathcal{C}$. Un HS de $\mathcal{C}$ es mínimo si y sólo si ningún subconjunto propio es un hitting set de$\mathcal{C}$. Se le llama *single minimal hitting set* a un hitting set formado por un solo componente.

Siendo los distintos hitting set minimales $\{\mathcal{HS}_1, \ldots, \mathcal{HS}_i, \ldots, \mathcal{HS}_n\}$, la cardinalidad de $\mathcal{HS}_i$ ($|\mathcal{HS}_i|$) es el número de componentes que forman $\mathcal{HS}_i$.

Reiter [127] presentó una de las primeras soluciones para la determinación de los hitting sets mínimos utilizando los llamados *HS-trees*. Este trabajo fue mejorado por Greiner [65] en el llamado *HS-DAG*, evitando que el tamaño del árbol de combinaciones creciera exponencialmente con el número de componentes. Otras soluciones relativas a la obtención de hitting sets minimales fueron *BHS-trees* mediante la utilización de árboles binarios, o los *HST-trees* de Wotawa [155]. Aunque también existen las soluciones que utilizan programación con restricciones en lugar de árboles [41], basadas en encontrar el máximo conjunto de restricciones satisfactibles para una modelo observable determinado.

Definición 7.9. Diagnosis mínima. Está formada por los hitting sets mínimos del sistema para un modelo observable. En caso de que se utilicen distintos modelos observables, la diagnosis mínima será la intersección entre todos los hitting sets mínimos obtenidos para cada modelo observable.

Por ejemplo, para el modelo observable {a=1, b=2, c=2, d=3, e=2, f=9, g=11} no son satisfactibles las Restricciones de Contextos con Redundancia Analítica $CARC_1$ y $CARC_2$. Lo que significa que los Contextos Conflictivos Mínimos para este modelo observable serán: $\{M_1, M_2, A_1\}$ y $\{M_2, M_3, A_2\}$. En este caso M_2 será un *single minimal hitting set*.

Para estos Contextos Conflictivos Mínimos los hitting sets mínimos obtenidos son: $\{M_2\}$, $\{M_1, M_3\}$, $\{M_1, A_2\}$, $\{A_1, M_3\}$, $\{A_1, A_2\}$.

7.3. La Diagnosis de Fallos y las BDdR

Una vez mostrados todas las definiciones relativas a la diagnosis de fallos basada en modelos, a continuación se presenta qué sentencias se realizarán para almacenar la información relativa al sistema, y qué consultas son necesarias para obtener la diagnosis de fallos cuando éste está almacenado en una LORCDB. Para mostrar las valores que se obtienen se continuará utilizando el ejemplo de la figura 7.1

7.3.1. Inserción de Componentes en la BDdR

Para almacenar el comportamiento de cada uno de los componentes del sistema es necesario crear una tabla con un atributo restricción. Un ejemplo de dicha tabla puede ser la creada de la forma:

```
CREATE TABLE COMPONENTES(Id Number, Nombre String, Comportamiento
Constraint)
```

Una vez creada la tabla para almacenar los componentes, es el momento de insertar la información relativa a cada uno de ellos, como es el caso del componente M_1 mostrado a continuación, junto al resto de componentes mostrados en la tabla 7.1. Cada vez que se inserta una restricción se actualizan las tablas *Restricciones*, *Variables* y *Restricciones/Variables* para que las futuras búsquedas de restricciones relacionadas por variables sean más eficientes.

```
INSERT INTO COMPONENTES(Id, Nombre, Comportamiento)
  VALUES
(3, 'M1', {'a * c = x'(Float -10..10 a, Float -10..10 c, Float x)})
```

En este caso, pese a que no se establece el valor mínimo y máximo que puede tomar la variable `x` de forma explícita, se puede inferir de los rangos de `a` y `c`, obteniendo los valores $-100..100$. Esto es posible gracias a utilizar la gramática intervalar [11].

7.3.2. Obtención de la Red de Contextos

Para obtener la Red de Contextos es necesario realizar todas las combinaciones de los componentes del sistema. Para realizarlo mediante consultas a la base de datos, una forma utilizando exclusivamente consultas, es realizar una secuencia de productos cartesianos anidados sobre relaciones de booleanos, tantos como componentes tenga el sistema.

De forma general será:

$$(((\mathtt{C}_1 \times \mathtt{C}_2)\ldots) \times \mathtt{C}_i) \times \ldots) \times \mathtt{C}_n)$$

donde $\mathrm{C}_1 \ldots \mathrm{C}_i \ldots \mathrm{C}_n$ representan n relaciones, una por componente. Cada una de estas relaciones estará formada por un atributo que representará al componente i-ésimo del sistema y dos tuplas, una con el valor *true* y otro *false* como las mostradas en la figura 7.3. Tras realizar los $n-1$ productos cartesianos entre las relaciones y eliminando la primera tupla que tendrá todos los componentes a *false*, se obtendrá una relación con n atributos y $2^{nComp} - 1$ tuplas que representarán los componentes que forman cada Contexto.

A1
False
True

×

A2
False
True

=

A1	A2
False	False
False	True
True	False
True	True

A1	A2
False	False
False	True
True	False
True	True

×

A3
False
True

=

A1	A2	A3
False	False	False
False	False	True
False	True	False
False	True	True
True	False	False
True	False	True
True	True	False
True	True	True

Figura 7.3: Producto Cartesiano entre Componentes

7.3.3. Obtención de las CARCs y los PCCM

Para la obtención tanto de los Posibles Contextos Conflictos Mínimos (PCCM), junto a las restricciones asociadas a ellos (CARC), hay que definir las variables donde están los sensores colocados, los cuales determinarán qué variables son observables. Como ejemplo se propone obtener los nombres de los componentes de los PCCM y las CARCs asociados para las variables observables *a, b, c, d, e, f, g*. Para obtener unas nuevas restricciones donde sólo aparezcan relacionadas las variables observables, será necesario realizar una proyección simbólica, lo que conlleva utilizar la palabra reservada`CONSTRAINTS`. Debido al diseño de la tabla *Componentes*, todas las restricciones se almacenan en un mismo atributo restricción llamado *Comportamiento*, por lo que la proyección será vertical, aunque no será necesario utilizar la palabra reservada `VERTICAL` ya que todas las variables de la proyección pertenecen al mismo atributo restricción. Al realizar la diagnosis es más conveniente realizar sustitución de variables en lugar de eliminación, así será posible obtener las salidas en función de las entradas, por lo que se utilizará la palabra reservada`FULL`. Uniendo todas estas características, se construirá la consulta:

$R_1 =$
```
SELECT NOMBRE, FULL CONSTRAINTS
  (COMPORTAMIENTO.a, COMPORTAMIENTO.b, COMPORTAMIENTO.c,
  COMPORTAMIENTO.d, COMPORTAMIENTO.e, COMPORTAMIENTO.f, COMPORTAMIENTO.g)
FROM COMPONENTES
```

Nombre	Comportamiento(a, b, c, d, e, f, g)
M1	$-(a*c)-b*d+f=0$
A1	$-(a*c)-b*d+f=0$
M2	$-(a*c)-b*d+f=0$
M1	$-(a*c)+c*e+f-g=0$
A1	$-(a*c)+c*e+f-g=0$
A2	$-(a*c)+c*e+f-g=0$
M3	$-(a*c)+c*e+f-g=0$
M2	$-(b*d)-c*e+g=0$
A2	$-(b*d)-c*e+g=0$
M3	$-(b*d)-c*e+g=0$

Tabla 7.2: Obtención de CARC para el ejemplo introductorio

Donde se obtendrá como salida la tabla 7.2. Sobre la relación de salida R_1 se puede obtener por una parte los componentes que forman el PCCM, y por otra parte cada CARC para cada PCCM. Para ello se crearán dos tablas, *PCCM* y *CARCS*, de la forma:

```
CREATE TABLE CARCS(idPCCM Autonumérico, carc Constraint)
```

La tabla *CARCS* almacenará cada una de las restricciones derivadas en el atributo *carc* como el conjunto de componentes que forman parte del PCCM con el identificador almacenado en el atributo *idPCCM*. Dicho identificador se generará de forma automática con un campo autonumérico. Para almacenar las CARCs en dicha tabla se realizará la sentencia sobre la relación R_1 anteriormente obtenida de la forma:

```
INSERT INTO CARCS (carc) VALUES (R1)
```

Como las tuplas no pueden obtenerse repetidas, se obtendrá la tabla mostrada en figura 7.4.a.

IdPCCM	CARC
1	$-(a*c)-b*d+f=0$
2	$-(a*c)+c*e+f-g=0$
3	$-(b*d)-c*e+g=0$

(a) Tabla CARCs

IdPCCM	Componente
1	M1
1	A1
1	M2
2	M1
2	A1
2	A2
2	M3
3	M2
3	A2
3	M3

(b) Tabla PCCM

Figura 7.4: Tablas de las CARCs y los PCCM para el ejemplo introductorio

Y para obtener los componentes del PCCM se creará una tabla donde serán almacenados. La creación de la tabla será de la forma:

```
CREATE TABLE PCCM(idPCCM Integer, Componente String)
```

Y rellenándola con la información de la consulta R_1 de la forma:

```
INSERT INTO PCCM(idPCCM, Componente) VALUES
 (Select CARCS.idPCCM, R1.Nombre from CARCS, R1
 WHERE CARCS.carc=R1.Comportamiento)
```

Tras la inserción de esta información, la tabla PCCM quedará como se muestra en la figura 7.4.b.

7.3.4. Obtención de los CCM para un Modelo Observable

Para obtener los Contextos Conflictivos Mínimos es necesario saber qué restricciones son satisfactibles para el conjunto de los valores de los sensores, lo que se establece en el predicado de la consulta (la condición del WHERE). Para conocer si cada una de las restricciones (CARC) almacenadas en la tabla son satisfactibles para cada uno de los valores de los sensores, se utilizará las palabras reservadas `BOOLEAN VALUE`, que devuelve *true* o *false*. Dicha operación se realiza sobre restricciones, siendo equivalente a `VALUES`, pero con la única diferencia que devuelve *true* si encuentra un valor de las variables que hace satisfactible la restricción, renombrando dicho atributo con el nombre *Satisfactible*. La sentencia será de la forma:

R_2 = `SELECT idPCCM, CARC, BOOLEAN VALUE (CARC)` **as Satisfactible**
`FROM CARCS WHERE (CARC.a=1 AND CARC.b=2 AND CARC.c=2`
`AND CARC.d=3 AND CARC.e=2 AND CARC.f=9 AND CARC.g=11)`

Obteniendo la relación mostrada en la tabla 7.3, donde se añade un nuevo atributo que indica si cada CARC es o no satisfactible en función de la instanciación de un conjunto de variables.

IdPCCM	CARC	Satisfactible
1	$-(a*c)-b*d+f=0$	false
2	$-(a*c)+c*e+f-g=0$	true
3	$-(b*d)-c*e+g=0$	false

Tabla 7.3: Obtención de los CCM (I)

Para realizar la diagnosis mínima habrá que obtener sólo aquellos conjuntos de componentes asociados a las CARCs no satisfactibles, los cuales se obtendrán realizando la consulta:

`SELECT IdPCCM, CARC FROM` R_2 `WHERE Satisfactible=false`

Obteniendo la tabla 7.4.

IdPCCM	CARC
1	$-(a*c)-b*d+f=0$
3	$-(b*d)-c*e+g=0$

Tabla 7.4: Obtención de los CCM (II)

7.3.5. Obtención de los Hitting Sets Mínimos

Partiendo de los componentes que forman los CCM conseguidos en el apartado anterior, se obtendrán los hitting sets y con ellos la diagnosis mínima del sistema para el modelo observable. Los hitting set deben estar formados por al menos un componente de cada CCM, y para que sea mínimo, sólo podrá estar formado por un componente de cada CCM. La cardinalidad máxima de los hitting sets será el número de CCM diferentes que se hallan obtenido, 2 en el caso del ejemplo que se está desarrollando.

Y con la consulta, combinado la tabla CCM y PCCM que contiene los componentes de cada hitting set:

```
SELECT PCCM.Componente, PCCM_1.Componente
FROM PCCM, PCCM AS PCCM_1
WHERE PCCM.idPCCM=1 AND PCCM.idPCCM_1.id=3
```

Se obtendrá una tupla por cada hitting set mínimo, en este caso las representadas en la tabla 7.5. O lo que es lo mismo, la tabla 7.6.

En la tabla se observan hitting sets mínimos de uno y dos componentes, que corresponde con el número de CCM. El caso de $M2$ es un *single minimal hitting set*, ya que el fallo del componente $M2$ explicaría el comportamiento del sistema para el modelo observable descrito. Las tuplas restantes representan *hitting sets mínimos* de dos componentes $\{M1, M2\}$, $\{A1, M2\}$, $\{M1, A3\}$, $\{A1, A3\}$, ...

7.4. Más ejemplos de Consultas

En las secciones anteriores se ha presentado un ejemplo y las consultas que hay que realizar para obtener la diagnosis de un sistema. El ejemplo estaba formado por cinco componentes, todos ellos con restricciones polinómicas de igualdad, por lo que la eliminación de variables se realiza utilizando la teoría de las Bases de Gröbner. En esta sección

PCCM.Componente	PCCM_1.Componente
M1	M2
A1	M2
M2	M2
M1	A2
A1	A2
M2	A2
M1	M3
A1	M3
M2	M3

Tabla 7.5: Obtención de los Hitting Sets Mínimos

Hitting Set	A1	A2	M1	M2	M3
HS_1	0	0	1	1	0
HS_2	1	0	0	1	0
HS_3	0	0	0	1	0
HS_4	0	1	1	0	0
HS_5	1	1	0	0	0
HS_6	0	0	1	0	1
HS_7	0	1	0	1	0
HS_8	1	0	0	0	1
HS_9	0	0	0	1	1

Tabla 7.6: Obtención de los Hitting Sets Mínimos (II)

se analiza un ejemplo más complejo, tanto en el número de componentes, que pasarán a ser 38, como en el tipo de restricciones, que serán tanto ecuaciones como inecuaciones polinómicas. También se presentan nuevos tipos de consultas relacionadas con la proyección numérica a parte de la simbólica. El conjunto de componentes del sistema y sus relaciones es el presentado en la figura 7.5, donde Mi representa multiplicadores, Ai sumadores y Ri restadores.

El sistema se ha dividido en cuatro partes en función del tipo de restricción

que la forman. La parte del sistema etiquetada con **(1)** tiene restricciones de la forma {variable$_1$ operación variable$_2$ <= variable$_3$ $\wedge$ variable$_1$ operación variable$_2$ >= variable$_3$ operación constante}. La parte del sistema etiquetada con **(2)** tiene restricciones de la forma {variable$_1$ operación variable$_2$ <= variable$_3$}, mientras que los componentes etiquetados con **(3)** y **(4)** son de la forma: {variable$_1$ operación variable$_2$ = variable$_3$}. Los detalles de todos los componentes se presentan en la sección B.1 del apéndice B.

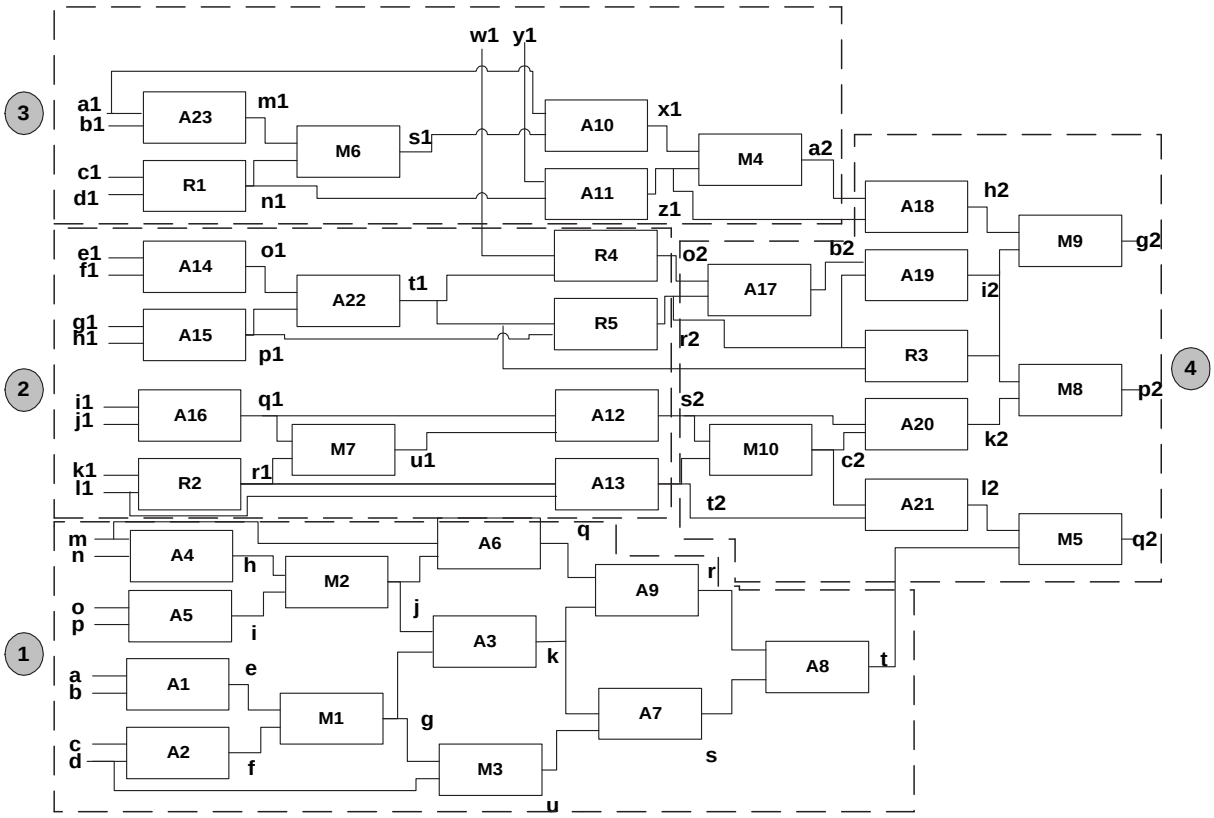

Figura 7.5: Ejemplo de Diagnosis

Los componentes están almacenados en una tabla creada de la forma:

```
CREATE TABLE COMPONENTES (Identificador Integer,
  Nombre String, Comportamiento Constraint, Subsistema Integer)
```

Para insertar información en la tabla se harán sentencias de la forma:

```
INSERT INTO TABLE COMPONENTES
  (IDENTIFICADOR, NOMBRE, COMPORTAMIENTO, SUBSISTEMA)
VALUES
  (1, 'A1', {'a+b<=e AND e-1<=a+b' (Float 5..10 a,
  Float 5..15 b, Float e)}, 1)
```

7.5. Ejemplos de Consultas en la Diagnosis Basada en Modelos

Como se ha presentado en capítulos anteriores, el uso de las BDdR amplía la semántica de las operaciones del álgebra relacional. En esta sección se muestran algunos ejemplos de selección y proyección utilizando la tabla *Componentes* sobre los tres tipos de atributos definidos en las BDdR (at_i, at_i^c, $\mathrm{at}_i^c.\mathrm{v}_j$). Estos tres tipos de atributos permiten ampliar los tipos de consultas.

Con la combinación del operador y selección se pueden realizar gran cantidad de consultas relativas a la diagnosis basada en modelos. La combinación natural de las operaciones es de la forma: $R_1 = \pi_{a_1,\ldots,a_n}(\sigma_{predicate}(R_2))$, que con los distintos atributos que pueden participar en ellas, mostrados en la tabla 7.7, se obtendrán nueve variantes.

π atributos	σ atributos relativos al predicado
$\mathtt{at}_i$	$\mathtt{at}_i$
$\mathtt{at}^c{}_i$	$\mathtt{at}^c{}_i$
$\mathtt{at}^c{}_i.\mathtt{v}_j$	$\mathtt{at}^c{}_i.\mathtt{v}_j$

Tabla 7.7: Posibles combinaciones de atributos y operaciones π y σ

Algunos de los ejemplos de estas combinaciones relacionadas con la diagnosis basada en modelos pueden ser:

1. $\pi_{at_1\ldots at_m}(\sigma_{(at_1\ldots at_n)})$. ¿Cuál es el identificador del componente cuyo nombre es $A1$? En esta consulta sólo hay relacionados atributos clásicos, por lo que sólo se hará uso del álgebra relacional clásica.

2. $\pi_{at_1 \ldots at_m}(\sigma_{(at_1^c \ldots at_n^c)})$. ¿Qué nombres tienen los componentes cuyo contenido del atributo *Comportamiento* es igual a la restricción $x + y = z$? Esta consulta obtendrá como salida todos los nombres de los componentes cuya restricción de comportamiento sea igual a dicha restricción. Esta operación de comparación es una de las definidas entre restricciones en el capítulo 4, donde se describe que dos restricciones son iguales si comparten las mismas soluciones.

3. $\pi_{at_1 \ldots at_m}(\sigma_{(at_1^c.v_1 \ldots at_1^c.v_k \ldots at_n^c.v_1 \ldots at_n^c.v_k)})$. ¿En qué componentes la variable c puede tomar el valor 5? En este caso se buscarán las restricciones con dicha variable comprobando que el valor 5 está incluido entre los valores de las envolventes, y con aquellos que cumplan dicha condición se creará un problema de satisfacción de restricciones.

4. $\pi_{(at_1^c \ldots at_m^c)}(\sigma_{(at_1 \ldots at_n)})$. ¿Qué restricción es la que describe el comportamiento del componente cuyo nombre es $A1$? Es una selección clásica, pese a que obtenga un atributo restricción.

5. $\pi_{(at_1^c \ldots at_m^c)}(\sigma_{(at_1^c \ldots at_n^c)})$. ¿Qué restricciones son distintas a $a + b = e$? Lo que implica compara las soluciones de cada restricción con $a + b = e$, y devolver aquellas que no son iguales.

6. $\pi_{(at_1^c \ldots at_m^c)}(\sigma_{(at_1^c.v_1 \ldots at_1^c.v_k \ldots at_n^c.v_1 \ldots at_n^c.v_k)})$. ¿Cuáles son las restricciones si se conocen los valores de las variables $\{a = 5, b = 3\}$? La evaluación de esta consulta consiste en sustituir las variables por constantes con los valores establecidos.

7. $\pi_{(at_1^c.v_1 \ldots at_1^c.v_k \ldots at_m^c.v_1 \ldots at_n^c.v_k)}(\sigma_{(at_1 \ldots at_n)})$. ¿Cuál es la relación simbólica de las variables *a, b, c, d, u* pertenecientes a los componentes $A1$, $A2$, $M1$ y $M3$? En este caso es necesario recurrir a la eliminación de cuantificadores, para que en la solución sólo aparezcan las variables establecidas en la proyección (*a, b, c, d, u*).

8. $\pi_{(at_1^c.v_1 \ldots at_1^c.v_k \ldots at_m^c.v_1 \ldots at_n^c.v_k)}(\sigma_{(at_1^c \ldots at_n^c)})$. ¿Qué valor máximo puede tomar la variable c en el atributo restricción *Comportamiento*? La evaluación de esta consulta se realizará con la creación de un problema de optimización de restricciones con las restricciones relacionadas con esta proyección numérica.

9. $\pi_{(at_1^c.v_1 \ldots at_1^c.v_k \ldots at_m^c.v_1 \ldots at_n^c.v_k)}(\sigma_{(at_1^c.v_1 \ldots at_1^c.v_k \ldots at_n^c.v_1 \ldots \ at_n^c.v_k)})$. ¿Cuáles son los valores de las variables a, b y c si $d = 5$ y $e = 6$? La solución a esta consulta se obtiene con la construcción y resolución de un problema de satisfacción de restricciones.

Aunque en esta sección se han incluido ejemplos generales sobre consultas relacionadas con la diagnosis de fallos, en las siguientes subsecciones se plantearán consultas concretas con las acciones a realizar necesarias para evaluarlas.

7.5.1. Ejemplos de Selección

En esta subsección se presentan tres ejemplos, cada uno de ellos correspondiente a un tipo de atributo.

- **Relativo a atributos clásicos:** Seleccionar las tuplas de la tabla *Componentes* cuyo atributo *Nombre* sea *A1*.

  `SELECT * FROM COMPONENTES WHERE NOMBRE = 'A1'`

  Obteniendo la tupla mostrada en la tabla 7.8.

Identificador	Nombre	Comportamiento
1	A1	$a + b = e$

Tabla 7.8: Salida consulta con Selección (I)

- **Relativo a atributos restricción:** Seleccionar las tuplas de la tabla *Componentes* cuyo campo *Comportamiento* sea igual a $\{c + d - g + 5 = 5\}$.

  `SELECT * FROM COMPONENTES WHERE COMPORTAMIENTO = "c + d - g + 5 = 5"`

  Obteniendo la tupla mostrada en la tabla 7.9.

Identificador	Nombre	Comportamiento
2	A2	$c + d = f$

Tabla 7.9: Salida consulta con Selección (II)

Para evaluar esta consulta se creará de forma automática sólo un problema de satisfacción de restricciones, ya que sólo se realiza la comparación entre restricciones que comparten las mismas variables.

- **Relativo a atributos variable de restricción:** Seleccionar las tuplas de la tabla *Componentes* cuya variable *Comportamiento.j* sea mayor que 5.

  ```
  SELECT * FROM COMPONENTES WHERE COMPORTAMIENTO.j>5
  ```

  Obteniendo la tupla mostrada en la tabla 7.10.

Identificador	Nombre	Comportamiento
5	M2	$h * i = j$
7	A6	$j + m = q$
8	A3	$j * g = k$

Tabla 7.10: Salida consulta con Selección (III)

Para evaluar esta consulta primero se analiza si la variable de la selección, en este caso `j`, puede tomar valores mayores que 5 en función de su envolvente almacenada en las tablas de indexación de la base de datos. Luego se construirá y resolverá un CSP para asegurar que esto es posible. Para el ejemplo propuesto, esto ocurre en todas las restricciones que contienen a `j`.

En caso de que la comparación fuera entre dos variables en lugar de entre una variable y una constante, se analizaría también en función de sus envolventes estudiando si existe algún valor donde la comparación se cumpla. Un ejemplo puede ser para la restricción del componente A6, si en el predicado apareciera la condición `Comportamiento.j>Comportamiento.q`, y el dominio de las variables `j` y `m` fueran positivos, dicha tupla no formaría parte de la salida ya que la condición nunca se podría cumplir.

7.5.2. Ejemplos de Proyección

Sobre una relación se puede obtener un subconjunto de sus atributos, que pueden ser de los tres tipos comentados. A continuación se muestran ejemplos para cada uno de los tipos de atributos:

- **Relativo a atributos clásicos:** Obtener el atributo *Nombre* de los componentes de la relación *Componentes*.

```
SELECT NOMBRE FROM COMPONENTES
```

Obteniendo la tupla mostrada en la tabla 7.11.

Nombre
A1
A2
A4
A5
M2
M1
A6
...

Tabla 7.11: Salida consulta con Proyección (I)

- **Relativo a atributos restricción:** Obtener el comportamiento de las tuplas de la relación *Componentes*.

```
SELECT COMPORTAMIENTO FROM COMPONENTES
```

Obteniendo la tupla mostrada en la tabla 7.12.

Comportamiento
$a + b = e$
$c + d = f$
$m + n = h$
$o + p = i$
$h * i = j$
...

Tabla 7.12: Salida consulta con Proyección (II)

En este caso para evaluar esta consulta sólo se utilizará el álgebra relacional clásica, ya que no existen dos restricciones con las mismas variables, lo que significa que son diferentes. En otros casos, cuando dos o más restricciones comparten las mismas variables será necesario realizar la comparación de dichas restricciones para

comprobar que no son iguales mediante la creación y resolución de un problema de satisfacción de restricciones.

- **Relativo a atributos variable de restricción:** Ésta es sin duda la proyección que implica más complicación algorítmica y computacional, ya que un atributo variable de restricción se puede obtener de forma simbólica o numérica. Siguiendo la sintaxis y la semántica de la extensión de SQL propuesta en esta tesis y mostrada en el capítulo 4, se presentan distintos ejemplos:

  - **Proyección numérica para obtener tuplas de valores:** Obtener 10 valores de las variables *c1, d1, b1, a1, y1, a2.*

    ```
    SELECT VALUES[10](COMPORTAMIENTO.c1, COMPORTAMIENTO.d1,
    COMPORTAMIENTO.b1, COMPORTAMIENTO.a1, COMPORTAMIENTO.y1,
    COMPORTAMIENTO.a2) FROM COMPONENTES
    ```

    Obteniendo la tupla mostrada en la tabla 7.13.

c1	d1	b1	a1	y1	a2
9	9	0	5	20	100
9	10	0	5	1	20
9	10	0	5	2	30
9	10	0	5	3	40
9	10	0	5	4	50
9	10	0	5	5	60
9	10	0	5	6	70
9	10	0	5	7	80
9	10	0	5	8	90
9	10	0	5	9	100

Tabla 7.13: Salida consulta con Proyección (III)

Para evaluar esta consulta se obtiene los conjuntos de restricciones relacionadas con las variables que aparecen en la proyección, con lo que obtiene las restricciones del subconjunto de componentes $\{A23, M6, R1, A10, M4, A11\}$. En ese caso la proyección será vertical por defecto, ya que todas las variables pertenecen a un mismo atributo restricción. Con esas restricciones se construye

un problema de satisfacción de restricciones, devolviendo los10 primeros valores obtenidos al resolver dicho problema.

- **Proyección numérica para obtener valores máximos o mínimos:** Obtener el mayor valor que puede tomar la variable t para los componentes {$A1$, $M1$, $A2$, $M3$, $A7$, $A3$, $A8$, $M2$, $A9$, $A4$, $A5$, $A6$}.

  `SELECT VERTICAL MAX VALUE(COMPORTAMIENTO.t) FROM COMPONENTES WHERE SUBSISTEMA` $= 1$

  Obteniendo la tupla mostrada en la tabla 7.14.

t
8063

Tabla 7.14: Salida consulta con Proyección Numérica Vertical (IV)

Para evaluar esta consulta se creará un COP con las restricciones de los componentes {$A1$, $M1$, $A2$, $M3$, $A7$, $A3$, $A8$, $M2$, $A9$, $A4$, $A5$, $A6$}, estableciendo t como objetivo a maximizar. Con esas restricciones se construye y resuelve un problema de optimización de restricciones donde se buscará el valor máximo de la variable `t`.

- **Proyección simbólica:** Obtener las restricciones de Contextos con Redundancia Analítica (CARC) y sus componentes asociados cuando los sensores están en las variables {a1, b1, c1, d1, y1, a2}. Lo que significa obtener restricciones donde sólo las variables de la proyección pueden aparecer. Como sólo se puede conocer la información de las variables con sensores, relacionando entradas con salidas del sistemas, se utilizará la palabra reservada `FULL`, para que sólo se analicen los conjuntos completos de restricciones relacionadas por variables.

  `SELECT NOMBRE, FULL CONSTRAINTS (COMPORTAMIENTO.a1, COMPORTAMIENTO.b1, COMPORTAMIENTO.c1, COMPORTAMIENTO.d1, COMPORTAMIENTO.y1, COMPORTAMIENTO.a2) FROM COMPONENTES WHERE SUBSISTEMA` $= 3$

  O utilizando la secuencia de consultas descritas en la sección 7.3.4, obteniendo en una tabla las CARCs y en otra los componentes del PCCM, lo que se muestra en la figura 7.6.

IdPCCM	Componente
1	A23
1	M6
1	R1
1	A10
1	M4
1	A11

(b) Tabla PCCM

IdPCCM	Comportamiento(a1, b1, c1, d1, y1, a2)
1	a2-a1*c1-a1*c1^2-b1*c1^2 +a1*d1+2*a1*c1*d1+2b1*c1d1^2 -a1*d1^2 -b1*d1-a1*y1-a1*c1*y1-b1*c1*y1+a1*d1*y1+b1*d1y1=0

(a) Tabla CARCs

Figura 7.6: Salida consulta con Proyección (V)

En este caso sólo se obtiene un conjunto de restricciones relacionadas por variables formada por las restricciones de los componentes {$A23$, $M6$, $R1$, $A10$, $M4$, $A11$}. Esta nueva restricción se utiliza en la diagnosis para, en función del Modelo Observable, conocer si la restricción es satisfactible y por lo tanto todos los componentes funcionan de forma correcta, o si por el contrario no se cumple y algún componente está fallando. En este caso la obtención de la nueva restricción se realiza utilizando las bases de Gröbner, ya que todas son polinómicas de igualdad.

- Otro ejemplo de proyección simbólica que también utiliza las Bases de Gröbner pero sobre varios conjuntos de restricciones relacionadas, es obtener las restricciones de Contextos con Redundancia Analítica (CARC) y sus componentes asociados cuando los sensores están en las variables {a2, z1, b2, r2, t1, s2, c2, g2, p2}. Lo que significa obtener restricciones donde sólo las variables de la proyección pueden aparecer. Como sólo se puede conocer la información relativa a los sensores se utilizará la palabra reservada `FULL`, para que sólo se analicen los CRRV completos.

```
SELECT NOMBRE, FULL CONSTRAINTS (COMPORTAMIENTO.a2,
COMPORTAMIENTO.z1, COMPORTAMIENTO.b2, COMPORTAMIENTO.r2,
COMPORTAMIENTO.t1, COMPORTAMIENTO.s2, COMPORTAMIENTO.c2,
COMPORTAMIENTO.g2, COMPORTAMIENTO.p2) FROM COMPONENTES WHERE
SUBSISTEMA = 4
```

O utilizando la secuencia de consultas descritas en la sección 7.3.4, obteniendo en una tabla las CARCs y en otra los componentes del PCCM, lo que se muestra en la figura 7.7.

IdPCCM	Comportamiento(a2, z1, b2, r2, t1, s2, c2, g2, p2)
1	$a2 * b2 + b2^2 + g2 - a2 * r2 - b2 * r2 = 0$
2	g2-a2*r2-b2*r2+a2*t1+b2*t1=0
3	c2*g2-a2*p2-b2*p2+g2*s2=0
4	-b2+t1=0
5	-(b2*c2)-p2+c2*r2-b2*s2+r2*s2=0
6	p2-c2*r2-r2*s2+c2*t1+s2*t1=0

(a) Tabla CARCs

IdPCCM	Componente
1	A18
1	M9
1	A19
2	A18
2	M9
2	R3
3	A18
3	M9
3	M8
3	A20
4	A19
4	R3
5	M8
5	A19
5	A20
6	R3
6	M8
6	A20

(b) Tabla PCCM

Figura 7.7: Salida consulta con Proyección (VI)

En este caso se obtienen seis conjuntos de restricciones relacionadas por variables {A18, M9, A19}, {A18, M9, R3} y {A18, M9, M8, A20}, {A19, R3}, {A19, M8, A20}, {R3, M8, A20} cada uno de ellos con una restricción asociada. Dichos conjuntos se obtienen de una manera eficiente utilizando el algoritmo de proyección simbólica presentado en el apéndice A.

- Otro ejemplo de proyección simbólica, pero que utiliza la Descomposición Algebraica Cilíndrica por tratar inecuaciones, se basa en obtener nuevas restricciones y los componentes asociados cuando los sensores están en las variables {a, b, c, d, u}. Se utilizará la palabra reservada `FULL`, para que exclusivamente se analicen los CCRRV. La consulta en este caso será:

  ```
  SELECT NOMBRE, FULL CONSTRAINTS(COMPORTAMIENTO.i1,
  COMPORTAMIENTO.j1, COMPORTAMIENTO.k1, COMPORTAMIENTO.l1,
  COMPORTAMIENTO.s2) FROM COMPONENTES WHERE SUBSISTEMA = 2
  ```

  También se puede obtener dicha información por separado en las tablas $CARCS$ y $PCCM$, utilizando la secuencia de consultas descritas en la sección 7.3.4, lo que se muestra en la figura 7.8.

IdPCCM	Componente
1	A16
1	M7
1	R2
1	A12

(b) Tabla PCCM

IdPCCM	Comportamiento(i1, j1, k1, l1, s2)
1	j1<-i1∨j1=-i1∧(l1≥1+k1∧s2≥0∨l1>1+k1) ∨j1>-i1∧(l1<1+k1∧s2≥i1+j1+i1*k1+j1*k1-i1*l1-j1*l1 ∨l1=1+k1∧s2≥0∨l1>1+k1)

(a) Tabla CARCs

Figura 7.8: Salida consulta con Proyección (VII)

Como el operador OR no puede aparecer dentro de un atributo restricción será necesario transformar dicha restricción a DNF (Disjunctive Normal Form) para representar cada una de las partes de la restricción en una tupla diferente, ya que las relaciones entre las distintas tuplas es precisamente disyuntiva. La nueva relación es por lo tanto la que se muestra en la figura 7.9.

7.6. Resumen

En este capítulo se presentan distintas definiciones relacionadas con la diagnosis de fallos basada en modelos. Este tipo de problema puede ser tratado con las BDdR, ya que el comportamiento de los componentes puede ser representado con restricciones polinómicas de igualdad o desigualdad. Esto hace posible que utilizando CORQL y LORCDB, se pueda realizar un proceso tan complejo como es la diagnosis de fallos exclusivamente mediante un conjunto de consultas. También se presenta, para un ejemplo más grande, distintos tipos de consultas relacionados con las operaciones de selección y proyección. En este capítulo se puede apreciar cómo la sintaxis y la semántica de CORQL es válida tanto para ejemplos de uso tan dispares como el económico, mostrado en el capítulo anterior, y la diagnosis de fallos, junto al tratamiento de sistemas de información geográfica que se presentará en el siguiente capítulo. Esto facilita el desarrollo de cualquier aplicación que necesite usar restricciones, ya que la arquitectura ofrece gran parte de la lógica de inferencia y las técnicas que necesite utilizar. Además cabe destacar la transparencia de cara al usuario con respecto al tipo de restricciones que se ven involucradas en la consulta, y de la técnica necesaria para evaluarlas.

IdPCCM	Comportamiento(i1, j1, k1, l1, s2)
1	$j1 < -i1$
2	$j1 = -i1 \wedge l1 \geq 1 + k1 \wedge s2 \geq 0$
3	$j1 = -i1 \wedge l1 > 1 + k1$
4	$j1 > -i1 \wedge l1 < 1 + k1 \wedge s2 \geq i1 + j1 + i1 * k1 + j1 * k1 - i1 * l1 - j1 * l1$
5	$j1 > -i1 \wedge l1 = 1 + k1 \wedge s2 \geq 0$
6	$j1 > -i1 \wedge l1 > 1 + k1$

(a) Tabla CARCs

IdPCCM	Componente
1	A16
1	M7
1	R2
1	A12
2	A16
2	M7
2	R2
2	A12
3	A16
3	M7
3	R2
3	A12
4	A16
4	M7
4	R2
4	A12
5	A16
5	M7
5	R2
5	A12
6	A16
6	M7
6	R2
6	A12

(b) Tabla PCCM

Figura 7.9: Salida consulta con Proyección (VIII)

Capítulo 8

Optimización y Pruebas en la Evaluación de Consultas en BDdR

En los capítulos anteriores se han explicado las distintas decisiones tomadas para la definición de un nuevo concepto y diseño de las Bases de Datos de Restricciones. Dichas decisiones tienen como objetivos dar capacidad expresiva y hacer la evaluación de consultas lo más eficiente posible. En este capítulo, se presentan distintas pruebas empíricas para mostrar cómo dichas decisiones mejoran los tiempos de la evaluación para las distintas operaciones. Todas estas pruebas se basarán en dos tipos de casos de estudio: por una parte la diagnosis basada en modelos, especialmente adecuada para la Proyección sobre variables; y los sistemas de información geográfica para el resto de operadores (Selección, Unión y Diferencia). Como ya se ha comentado en capítulos anteriores, el Producto Cartesiano no conlleva ninguna modificación del tratamiento de los datos con respecto a los tipos básicos ya definidos en SQL, por lo que no se muestran pruebas para él. Para que la evaluación de las consultas sea eficiente en tiempo, se proponen distintas estrategias de resolución para cada una de las operaciones, definiendo el orden del tratamiento de datos, junto a un conjunto de heurísticas.

8.1. Introducción

Uno de los campos más analizados en el área de las bases de datos durante toda su historia, es la optimización de la evaluación de consultas, entendida como la mejora de

los tiempos de respuesta de un sistema gestor de bases de datos. Mejorar la evaluación de las consultas también implica la mejora de las operaciones de borrado y modificación, ya que ambas operaciones consisten en seleccionar un conjunto de tuplas de una relación que tienen que ser modificadas o borradas. Por esta razón, en este capítulo se presenta cómo las decisiones de diseño de la arquitectura LORCDB han influido en la mejora de los tiempos de evaluación de consultas.

En función del tipo de operación que se ejecute en la base de datos, se recurrirá a las distintas mejoras que se exponen a continuación. Las decisiones de implementación que están involucradas con la mejora computacional de los tiempos de evaluación de consultas son:

- **Estrategia de Optimización 1. Utilización del álgebra relacional:** El modelo de Base de Datos de Restricciones propuesto, separa en atributos distintos aquellos que representan valores únicos, a los que se les ha denominado atributos univaluados, y los campos que describen restricciones. Esto permite que la **Selección** se realice en dos pasos, primero la obtención de las tuplas que cumplen las condiciones relacionadas con el álgebra relacional clásica, y luego las relacionadas con las restricciones en sí. También la utilización del álgebra relacional influye en las operaciones de **Unión** y **Diferencia**, ya que ambas operaciones necesitan analizar si las tuplas de las relaciones involucradas son o no iguales. Para que dos tuplas sean iguales, lo tienen que ser sus atributos clásicos y sus atributos restricción, siendo más eficiente empezar la comparación por los atributos clásicos en lugar de por las restricciones, lo que implica la construcción y resolución de menos problemas de satisfacción de restricciones.

- **Estrategia de Optimización 2. Indexación de variables:** La arquitectura propuesta almacena las relaciones entre las variables y las restricciones a las que pertenecen. Esto permite realizar la búsqueda del Conjunto de Restricciones Relacionadas por Variables (CRRV) necesaria en la operación de **Proyección Simbólica y Numérica** sobre atributos variable de restricción. Si no se hiciera ningún tipo de tratamiento para conocer las restricciones inferidas tras la eliminación para la proyección simbólica, sería necesario analizar las 2^n-1 combinaciones posibles de conjuntos de restricciones, siendo n el número de restricciones implicadas. Esta indexación también permite conocer, en operaciones como la **Selección**, **Unión** y la **Diferencia**, si dos restricciones tienen o no las mismas variables.

- **Estrategia de Optimización 3. Almacemiento de envolventes de variables:** Cada vez que una restricción se almacena en una Base de Datos de Restricciones, se almacena también el valor máximo y mínimo que pueden tomar cada una de sus variables. Esto permite conocer el grado de consistencia entre un conjunto de restricciones simplemente comparando dichas envolventes. De esta forma, simplemente con una consulta sobre atributos numéricos y utilizando el álgebra relacional clásica, se puede conocer si es posible que dos restricciones compartan alguna solución, si todos los valores que toman un conjunto de variables para hacer satisfactible una restricción son mayores o menores que los que hacen satisfactible la otra restricción, o si ambas restricciones pueden compartir las mismas soluciones. Dicho conocimiento es muy importante, con respecto a la eficiencia computacional, en operaciones como la **Selección**, **Unión** y **Diferencia**.

  Otro tipo de operación que es mejorable, en aspectos de eficiencia, gracias a tener almacenado los valores máximos y mínimos de las variables, es la proyección numérica para maximizar o minimizar una variable. En este tipo de operación es necesario construir y resolver un problema de optimización de restricciones, lo que puede conllevar el análisis de todo el espacio de búsqueda para conocer cuál es el valor mayor o menor de dicha variable. Gracias a tener almacenado los valores máximos y mínimos de cada variable para cada restricción, se puede utilizar como heurística empezar la búsqueda instanciando la variable que se quiere maximizar o minimizar con el máximo o el mínimo valor respectivamente. Utilizando esta estrategia, se transformará el análisis de todos el espacio de búsqueda por la búsqueda de la primera solución.

- **Estrategia de Optimización 4. Etiquetado de restricciones:**Se han definido cuatro tipos de etiquetas de restricciones, para conocer si son inecuaciones o ecuaciones, o si son lineales o polinómicas. El etiquetado relativo a si son o no restricciones de igualdad ayuda a conocer qué técnica tiene que ser utilizada en la proyección simbólica sobre variables de restricción, mientras que el etiquetado que describen si son o no lineales ayuda a utilizar distintas heurísticas de resolución de problemas de satisfacción de restricciones [85][77][151]. En este documento se analizan tres tipos de heurísticas: por el dominio de las variables, por el tipo de restricción a la que pertenece (lineal o polinómica), y por el grado de las variables (número de restricciones en las que participa una variable).

Estas cuatro estrategias se utilizarán de diferente manera para la evaluación de las

distintas operaciones. Para mostrar los distintos tiempos de evaluación, en el caso del operador de *Proyección* se utilizará la diagnosis basada en modelos presentada en el capítulo 7. Esta elección es debida a que los ejemplos de sistemas de componentes a diagnosticar involucran muchas variables, enriqueciendo los tipos de proyección que se pueden hacer sobre las distintas variables. La búsqueda de las restricciones relacionadas por variables encuentra su caso más desfavorable y computacionalmente complejo, cuando hay involucradas gran cantidad de variables. Para mostrar el estudio empírico de los operadores de selección, unión y diferencia, es mucho más apropiado utilizar como caso de estudio los sistemas de información geográfica. Los SIG presentan su caso más complejo y desfavorable cuando las restricciones involucradas en la operación comparten las mismas variables, lo que puede implicar la construcción y resolución de problemas de satisfacción de restricciones. Además, el tratamiento de los sistemas de información geográfica es un área de gran importancia y en auge en el campo de las BDdR. De esta forma, cada operación se probará sobre los casos más desfavorables, ya que la proyección sobre atributos variable de restricción con las mismas variables no necesita obtener los CRRV, y la selección, la unión y la diferencia sobre restricciones con distintas variables, no necesita la creación de problemas de satisfacción de restricciones para resolverlas. En ambos casos de estudio se utilizará el dominio de los *Flotante*, ya que de forma general es computacionalmente más complejo que los *Enteros* y los *Naturales*.

Las pruebas temporales que se presentan en este documento sobre la evaluación se han realizado en una máquina *AMD Athlon* 64 *X*2 *Dual Core a* 2,21 *Hz con* 2 *GB de Memoria*. Cada uno de los tiempos que aparecen en este capítulo han sido obtenidos con 100 pruebas para obtener el tiempo medio de ejecución de ellas, siempre presentado en milisegundos.

8.2. Pruebas para el Operador de Proyección

La proyección sobre una relación permite obtener un subgrupo vertical de los atributos que la forman. En el caso de las Bases de Datos de Restricciones esto significa obtener atributos clásicos, atributos restricción o atributos variable de restricción. El único caso que necesita de un tratamiento diferente al que hacen los gestores de bases de datos relacionales clásicas, es la obtención de atributos variable de restricción. Serán los distintos

tipos de proyección en función de la salida y las restricciones involucradas, el objetivo de los ejemplos relacionados con este operador.

8.2.1. Ejemplo de Diagnosis Basada en Modelos

Como en el capítulo 7 ya se introdujo el problema de la diagnosis basada en modelos y por qué se utiliza como caso de estudio en esta tesis, sólo queda recordar el problema ya presentado que se vuelve a mostrar en la figura 8.1. Este sistema se divide a su vez en cuatro subsistemas etiquetados en la figura desde 1 hasta 4.

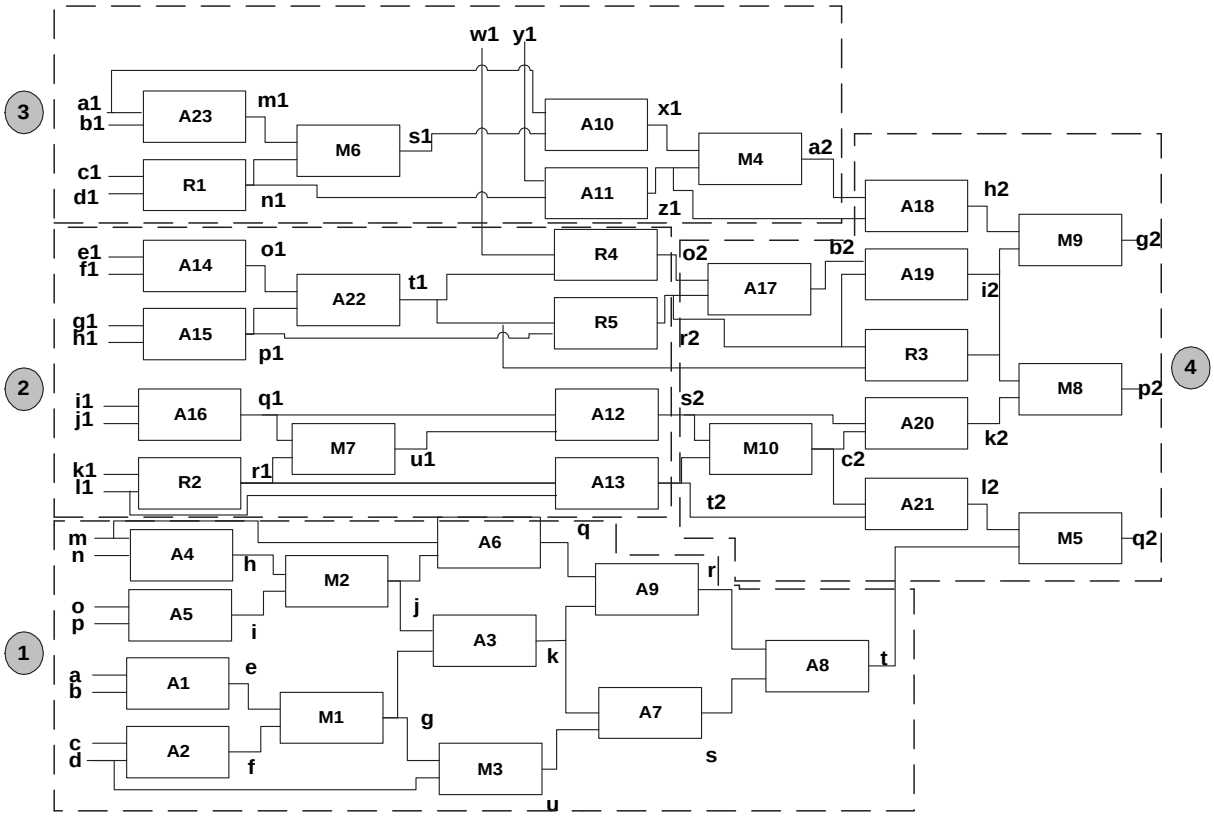

Figura 8.1: Ejemplo de Diagnosis

Las restricciones se almacenarán en una tabla llamada *Componentes* con los campos *Identificador*, *Nombre*, *Subsistema* y *Comportamiento*, donde *Comportamiento* es un atributo restricción. El contenido exacto de esta tabla se muestran en las figuras B.1, B.2, B.3 y B.4 del Apéndice B, donde cada tabla representa una parte de la tabla *Componentes* para cada uno de sus subsistemas. Las proyecciones que se realizan sobre dicho sistema

contemplan los tres tipos posibles: simbólicas, numéricas y numérica con maximización o minimización de variables.

8.2.2. Mejoras en la Evaluación de la Proyección Simbólica

La **Proyección Simbólica** sobre una relación, es la obtención de nuevas restricciones donde sólo estén involucradas las variables definidas en la proyección. Si no se utilizara la arquitectura LORCDB, sería necesario analizar toda las combinaciones de restricciones, mientras que utilizando la indexación propuesta se realizará la búsqueda de CRRV. Se presentan la comparación de ambas estrategias, sin utilizar los CRRV y con CRRV para demostrar empíricamente cómo se mejora la eficiencia computacional.

- **Analizar todas las combinaciones de las restricciones:** Este sería el caso donde la base de datos tiene almacenadas las restricciones sin información adicional sobre qué variables relacionan qué restricciones. Para inferir todas las restricciones que se pueden obtener, en función de un conjunto de variables dadas en la proyección, es necesario probar las $2^n - 1$ combinaciones, y que las técnicas de eliminación de cuantificadores obtuvieran las nuevas restricciones, en caso de que fuera posible.
- **Obtener los CRRV:** Esto es posible gracias a la **Estrategia 2**, donde se propone la indexación entre las restricciones y las variables, junto a un algoritmo de búsqueda que permite conocer a priori sólo las combinaciones de restricciones que inferirán nuevas restricciones relacionando sólo un conjunto de variables.

8.2.3. Ejemplos de Pruebas usando Proyección Simbólica

Para mostrar distintos usos de la proyección simbólica para las distintas técnicas de eliminación simbólica, se ha almacenado en la Base de Datos de Restricciones el ejemplo de la figura 8.1 que consta de diferentes tipos de restricciones. Los subsistemas1 y 2 con inecuaciones polinómicas, y los subsistemas 3 y 4 con ecuaciones polinómicas. Los diez primeros ejemplos con operaciones de proyección simbólica se harán sobre los subsistemas 3 y 4, por lo que se utilizarán las Bases de Gröbner, mientras que los diez últimos ejemplos se harán sobre los subsistemas 1 y 2, por lo que se utilizará de Descomposición Algebraica Cilíndrica para la eliminación de cuantificadores.

Las medidas temporales que aparecen en las tablas sólo tienen en cuenta el tiempo que tarda en realizarse la operación de proyección, no la selección que será analizada más adelante.

Las consulta con proyección simbólica para este ejemplo tienen la sintaxis:

```
SELECT FULL CONSTRAINTS (Comportamiento.v1, Comportamiento.v2, ...)
  FROM Componentes WHERE subsistema = n
```

donde n es el identificador del subsistema y v_1, v_2, ... son las variables que se quieren obtener en la proyección pertenecientes al atributo *Comportamiento*. Se presentan algunos ejemplos de consultas, utilizando como variables los casos más desfavorables, los cuales son aquellos donde están relacionadas las variables de entrada y salida de cada uno de los subsistemas, lo que implica una búsqueda de CRRV más compleja. Estos primero diez ejemplos son para eliminación simbólica de variables donde las restricciones involucradas son ecuaciones polinómicas, y por lo tanto utilizando las Bases de Gröbner:

1. Inferir las restricciones del subsistema 4 únicamente con las variables $s2$, $t2$, t, $q2$.
2. Inferir las restricciones del subsistema 4 únicamente con las variables $a2$, $z1$, $o2$, $r2$, $g2$.
3. Inferir las restricciones del subsistema 4 únicamente con las variables $r2$, $t1$, $s2$, $t2$, $p2$.
4. Inferir las restricciones del subsistema 4 únicamente con las variables $a2$, $z1$, $o2$, $r2$, $t1$, $s2$, $t2$, $g2$, $p2$.
5. Inferir las restricciones del subsistema 4 únicamente con las variables $s2$, $t2$, t, $r2$, $t1$, $p2$, $q2$.
6. Inferir las restricciones del subsistema 4 únicamente con las variables $s2$, $t2$, t, $a2$, $z1$, $o2$, $r2$, $g2$, $q2$.
7. Inferir las restricciones del subsistema 4 únicamente con las variables $s2$, $t2$, t, $a2$, $z1$, $o2$, $r2$, $t1$, $g2$, $q2$, $p2$.
8. Inferir las restricciones del subsistema 3 únicamente con las variables $a1$, $b1$, $c1$, $d1$, $y1$, $a2$.

9. Inferir las restricciones del subsistema 3 únicamente con las variables $c1$, $d1$, $y1$, $z1$.

10. Inferir las restricciones del subsistema 3 únicamente con las variables $a1$, $b1$, $c1$, $d1$, $y1$, $a2$, $z1$.

Los tiempos medios de evaluación de cada una de estas consultas se presentan en la tabla 8.1. Dicha tabla muestra el número de veces que se hace uso de las Bases de Gröbner, y el tiempo medio de ejecución en milisegundos para cada una de las consultas. El número de veces que se hace uso de las Bases de Gröbner será $2^n - 1$ si se analizan todas las posibilidades, o el número de CRRV dependiente del subsistema y de las variables que se quieran obtener.

Consulta	Todas las Combinaciones		Sólo CRRV	
	Número de Combinaciones	Tiempo (ms)	Número de CRRV	Tiempo (ms)
(1)	$2^{10} - 1 = 1023$	$1{,}5045 * 10^5$	1	$1{,}2871 * 10^2$
(2)	$2^{10} - 1 = 1023$	$1{,}2358 * 10^5$	1	$1{,}2831 * 10^2$
(3)	$2^{10} - 1 = 1023$	$1{,}2386 * 10^5$	1	$1{,}2910 * 10^2$
(4)	$2^{10} - 1 = 1023$	$1{,}3486 * 10^5$	3	$4{,}5416 * 10^2$
(5)	$2^{10} - 1 = 1023$	$1{,}2385 * 10^5$	3	$2{,}3815 * 10^2$
(6)	$2^{10} - 1 = 1023$	$1{,}2880 * 10^5$	2	$4{,}2740 * 10^2$
(7)	$2^{10} - 1 = 1023$	$1{,}2399 * 10^5$	5	$5{,}6727 * 10^2$
(8)	$2^6 - 1 = 63$	$8{,}9338 * 10^3$	1	$1{,}3592 * 10^2$
(9)	$2^6 - 1 = 63$	$8{,}2281 * 10^3$	1	$1{,}9431 * 10^1$
(10)	$2^6 - 1 = 63$	$8{,}2281 * 10^3$	1	$2{,}4359 * 10^2$

Tabla 8.1: Comparación para la evaluación de Proyecciones Simbólicas utilizando las Bases de Gröbner

De una forma similar se realiza la comparación entre la utilización o no de los CRRV cuando las restricciones involucradas en la relación son inecuaciones polinómicas. En este caso los ejemplos de proyección son:

1. Inferir las restricciones del subsistema 2 únicamente con las variables $w1$, $e1$, $f1$, $g1$, $h1$, $o2$.

2. Inferir las restricciones del subsistema 2 únicamente con las variables $e1$, $f1$, $g1$, $h1$, $r2$.

3. Inferir las restricciones del subsistema 2 únicamente con las variables $i1$, $j1$, $k1$, $l1$, $s2$.

4. Inferir las restricciones del subsistema 2 únicamente con las variables $k1$, $l1$, $t2$.

5. Inferir las restricciones del subsistema 2 únicamente con las variables $w1$, $e1$, $f1$, $g1$, $h1$, $o2$, $r2$.

6. Inferir las restricciones del subsistema 2 únicamente con las variables $e1$, $f1$, $g1$, $h1$, $i1$, $j1$, $k1$, $l1$, $r2$, $s2$.

7. Inferir las restricciones del subsistema 2 únicamente con las variables $i1$, $j1$, $k1$, $l1$, $s2$, $t2$.

8. Inferir las restricciones del subsistema 1 únicamente con las variables a, b, c, d, m, n, o, p, r.

9. Inferir las restricciones del subsistema 1 únicamente con las variables a, b, c, d, m, n, o, p, t.

10. Inferir las restricciones del subsistema 1 únicamente con las variables a, b, c, d, m, n, o, p, r, t.

Los tiempos medios de evaluación de cada una de estas consultas se presenta en la tabla 8.2. Dicha tabla muestra el número de veces que se hace uso de la Descomposición Algebraica Cilíndrica, y el tiempo medio de ejecución en milisegundos para cada una de las consultas. Al igual que para los sistemas con inecuaciones polinómica, el número de veces que se hace uso de la Descomposición Algebraica Cilíndrica será $2^n - 1$ si se analizan todas las posibilidades, o el número de CRRV dependiente del subsistema y de las variables que se quieran obtener.

8.2.4. Conclusiones para el operador de Proyección Simbólica

Para mostrar gráficamente las diferencias con respecto al tiempo de evaluación de la proyección, analizando todas las combinaciones o sólo los CRRV, se presentan las figuras 8.2 y 8.3. La figura 8.2 muestra los datos presentados en la tabla 8.1 relativos al tiempo de evaluación, sin ningún tipo de mejora (-♦-) o sólo analizando los CRRV (-■-). La

Consulta	Todas las Combinaciones		Sólo CRRV	
	Número de Combinaciones	Tiempo (ms)	Número de CRRV	Tiempo (ms)
(1)	$2^{10} - 1 = 1023$	$1{,}3147 * 10^6$	1	$1{,}4657 * 10^3$
(2)	$2^{10} - 1 = 1023$	$1{,}3329 * 10^6$	1	$1{,}4424 * 10^3$
(3)	$2^{10} - 1 = 1023$	$1{,}3247 * 10^6$	1	$1{,}5754 * 10^3$
(4)	$2^{10} - 1 = 1023$	$1{,}4320 * 10^6$	1	$1{,}4273 * 10^3$
(5)	$2^{10} - 1 = 1023$	$1{,}5847 * 10^6$	3	$3{,}9177 * 10^3$
(6)	$2^{10} - 1 = 1023$	$1{,}3250 * 10^6$	2	$2{,}8363 * 10^3$
(7)	$2^{10} - 1 = 1023$	$1{,}5522 * 10^6$	2	$2{,}9097 * 10^3$
(8)	$2^{12} - 1 = 4095$	$3{,}6122 * 10^6$	1	$1{,}2838 * 10^4$
(9)	$2^{12} - 1 = 4095$	$3{,}4882 * 10^6$	1	$2{,}9334 * 10^4$
(10)	$2^{12} - 1 = 4095$	$3{,}8114 * 10^6$	1	$8{,}9334 * 10^4$

Tabla 8.2: Comparación para la evaluación de Proyecciones Simbólicas utilizando Descomposición Algebraica Cilíndrica

figura 8.3 presenta la comparación de los tiempos de evaluación presentados en la tabla 8.2, sin ninguna mejora (-♦-) y sólo analizando los CRRV (-■-). Ambas figuras utilizan la escala logarítmica para facilitar la presentación de los datos, pudiendo así observar la mejora cuantitativa utilizando la búsqueda de los CRRV, que puede llegar a ser hasta 1000 veces más rápida. Dicha mejora se hace más notable para los sistemas con muchos componentes, ya que el número de combinaciones crece exponencialmente en función del número de restricciones.

8.2.5. Mejoras en la Evaluación de la Proyección Numérica

Cuando la proyección sobre una relación con restricciones requiere la obtención de un conjunto de los valores instanciados de las variables definidas en dicha proyección, se construirá y resolverá un problema de satisfacción de restricciones. Dicho problema de satisfacción de restricciones se creará con todas las restricciones involucradas en la relación como se explicó en la sección 6.4.1. Para resolver el problema de satisfacción de restricciones y obtener dichos valores, existe un conjunto de heurísticas para mejorar los tiempos, basadas en la ordenación de variables y valores, lo cual tiene un significativo impacto en el tamaño del espacio de búsqueda [38][43][52][53][83]. En este capítulo se

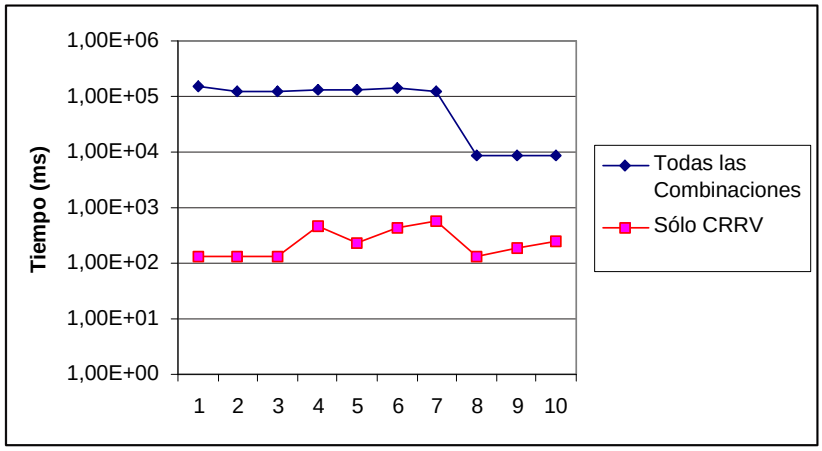

Figura 8.2: Tiempos de Evaluación para Proyección Simbólica para Ecuaciones Polinómicas

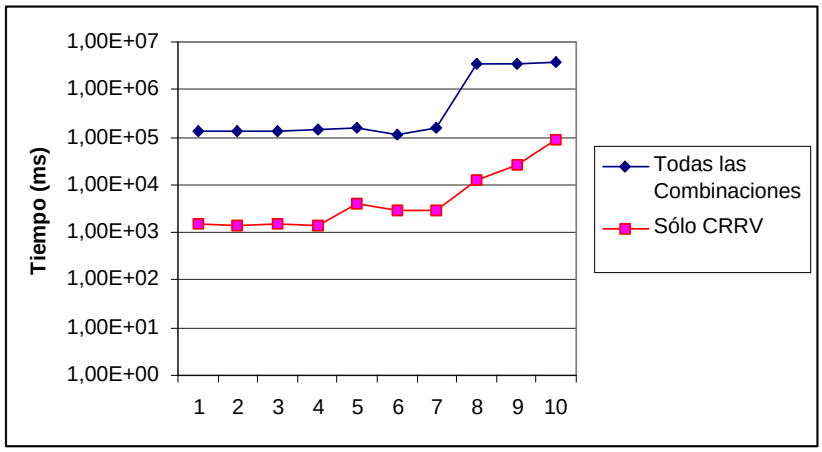

Figura 8.3: Tiempos de Evaluación para Proyección Simbólica para Inecuaciones Polinómicas

van a utilizar algunas de estas heurísticas, comparando los tiempos de evaluación con la resolución del problema sin utilizar ninguna de ellas. Las pruebas relativas a las heurísticas son:

- **Sin heurísticas:** Construcción de CSP sin heurísticas, realizando la búsqueda en el orden en el que la base de datos devuelve las restricciones y las variables.
- **Por el dominio de las variables:** Comenzar la búsqueda con las variables con el dominio más pequeño, lo que se puede hacer gracias a que la arquitectura LORCBD almacena las envolventes de cada variable (**Estrategia 3**).

- **Por tipo de restricción:** Comenzar por las variables que se encuentran en restricciones lineales, analizando primero las restricciones lineales y luego las polinómicas (**Estrategia 4**).
- **Por grado de las variables:** Comenzar por las variables de mayor grado, lo que se define como el número de restricciones en las que participa una variables. Gracias a utilizar las tablas de indexación entre variables y restricciones, es muy sencillo conocer el grado de las variables (**Estrategia 2**).

Precisamente por ser heurísticas, los tiempos de evaluación dependerán del tipo de problema [148], no siendo posible definir una heurística como *la mejor* de forma global.

8.2.6. Ejemplos de Pruebas usando la Proyección Numérica

Para todos los ejemplos se obtendrán los 10 primeros valores que encuentre que hagan las restricciones involucradas satisfactibles. El número de tuplas que se quieren obtener,10 en este caso, será el mismo para todas las heurísticas para que no influya en la comparativa.

La proyección numérica para este ejemplo tienen la sintaxis:

```
SELECT VALUES[10] (Comportamiento.v1, Comportamiento.v2, ...)
  FROM Componentes WHERE subsistema = n
```

donde n es el identificador del subsistema y v_1, v_2, ... son las variables que se quieren obtener en la proyección numérica pertenecientes al atributo restricción *Comportamiento*. Se ha realizado un ejemplo de proyección en cada subsistema, ya que pese a que sólo se muestra un conjunto de variables, se tiene que encontrar una solución para las restantes aunque no formen parte de la salida. Esto implica que el tiempo de resolución de un problema de satisfacción de restricciones será independiente de las variables que se presenten como solución. Los ejemplos utilizados son:

1. Obtener 10 tuplas de valores del subsistema 4 para las variables $a2$, $z1$, $b2$, $r2$, $t1$, $g2$.
2. Obtener 10 tuplas de valores del subsistema 3 para las variables $n1$, $m1$, $a1$, $y1$, $a2$.
3. Obtener 10 tuplas de valores del subsistema 2 para las variables $e1$, $f1$, $g1$, $h1$, $t1$.

4. Obtener 10 tuplas de valores del subsistema 1 para las variables a, b, c, d, u.

La comparación utilizando las diferentes heurísticas, presentada en milisegundos, es la expuesta en la tabla 8.3.

Consulta	Sin heurística Tiempo (ms)	Por dominio Tiempo (ms)	Por tipo de restricción Tiempo (ms)	Por grado Tiempo (ms)
(1)	49,884	30,408	29,883	32,142
(2)	34,157	30,957	28,311	27,145
(3)	31,981	31,985	31,457	31,452
(4)	38,797	37,748	25,690	28,1312

Tabla 8.3: Comparación para la evaluación de Proyecciones Numéricas

8.2.7. Conclusiones para el operador de Proyección Numérica

En la figura 8.4 se presentan los tiempos para los cuatro ejemplos de proyección mostrados en la tabla 8.3. En dicha figura se presenta gráficamente la comparación utilizando las cuatro posibilidades descritas para resolver el problema de satisfacción de restricciones: sin heurística (-♦-); comenzando por las variables de menor dominio (-■-); comenzando por variables que se encuentran en restricciones lineales (-▲-); y comenzando por las variables de mayor grado (-×-). Pese a que los tiempos obtenidos con las heurísticas son prometedores, no se puede generalizar y definir una heurística como la mejor. Lo que sí cabe destacar es que la arquitectura LORCBD facilita llevar a cabo dichas heurísticas, pudiendo hacerse de forma transparente para el usuario.

8.2.8. Mejoras en la Evaluación de la Proyección Numérica con Optimización de Variable

Cuando la proyección sobre una relación con restricciones requiere de la obtención del mayor o el menor valor que puede tomar una variable, será necesario construir y resolver un problema de optimización de restricciones. Dicho problema de optimización de restricciones se creará con todas las restricciones involucradas en la relación como se explicó en la sección 6.4.1. En este caso se realizará la comparación entre estudiar todo el

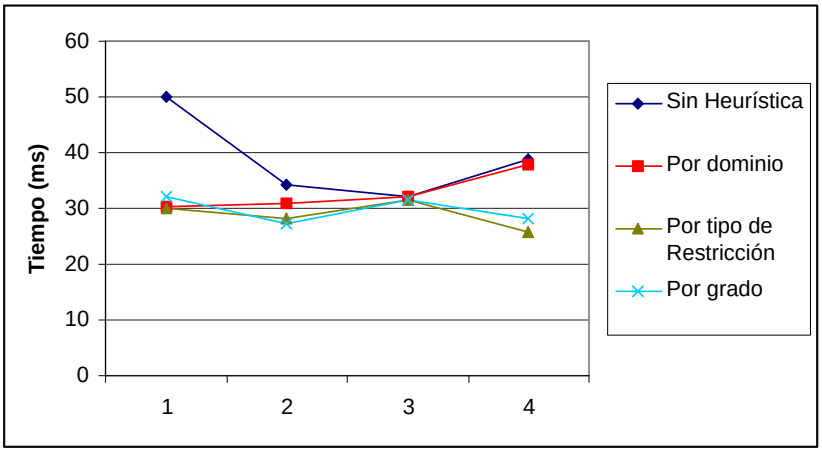

Figura 8.4: Tiempos de Evaluación para Proyección Numérica

espacio de búsqueda, o utilizar una estrategia derivada de las envolventes de las variables. La comparación de los distintos tiempos de ejecución se hará de la forma:

- **Estudiar todo el espacio de búsqueda:** Con el objetivo de maximizar o minimizar la variable definida en la proyección.
- **Comenzar la búsqueda por la variable objetivo con un valor de terminado (el máximo o el mínimo):** De esta forma se comienza la búsqueda instanciando la variable a maximizar por su mayor valor, o en el caso de la minimización por el menor valor de dicha variable (**Estrategia 3**). Esto implica transformar el análisis de todo el espacio de búsqueda en la búsqueda de la primera solución.

8.2.9. Ejemplos de Pruebas usando la Proyección Numérica con Optimización

En este caso se presentan diez ejemplos de proyección donde se quieren obtener el máximo valor que puede tomar una variable. Si no se utiliza ningún tipo de mejora, el tiempo dependerá de las restricciones que forman el COP, mientras que si se realiza la transformación del análisis de todo el espacio de búsqueda en la búsqueda de una solución, el tiempo se reducirá drásticamente.

La sintaxis de los ejemplos de proyección para este caso es:

```
SELECT MAX VALUE(Comportamiento.v)
  FROM Componentes WHERE subsistema = n
```

donde n es el identificador del subsistema y v la variable a maximizar perteneciente al atributo *Comportamiento*. Los ejemplos de consulta para la proyección numérica con maximización propuestos son:

1. Obtener el máximo valor que puede tomar la variable u para el subsistema 1.
2. Obtener el máximo valor que puede tomar la variable t para el subsistema 1.
3. Obtener el máximo valor que puede tomar la variable q para el subsistema 1.
4. Obtener el máximo valor que puede tomar la variable $t1$ para el subsistema 2.
5. Obtener el máximo valor que puede tomar la variable $s2$ para el subsistema 2.
6. Obtener el máximo valor que puede tomar la variable $a2$ para el subsistema 3.
7. Obtener el máximo valor que puede tomar la variable $y1$ para el subsistema 3.
8. Obtener el máximo valor que puede tomar la variable $g2$ para el subsistema 4.
9. Obtener el máximo valor que puede tomar la variable $q2$ para el subsistema 4.
10. Obtener el máximo valor que puede tomar la variable $p2$ para el subsistema 4.

Cuyos tiempos de evaluación, utilizando las dos estrategias explicadas, se presentan en la tabla 8.4.

8.2.10. Conclusiones para el operador de Proyección Maximizando valores de Variables

En la figura 8.5 se presenta, utilizando la escala logarítmica, la comparativa de todos los tiempos en milisegundos para las diez pruebas. Por una parte se analiza todo el espacio de búsqueda (-♦-), y por otra se utiliza como estrategia comenzar la búsqueda por la variable objetivo (-■-). En dicha gráfica se puede observar que la mejora es cuantitativamente muy superior, pudiendo obtener hasta una solución 10.000 veces mejor en algunos casos, ya

Consulta	Todo el Espacio de Búsqueda	Por Variable Objetivo
(1)	$6{,}8607 * 10^5$	$7{,}5497 * 10^2$
(2)	$6{,}8607 * 10^5$	$7{,}9927 * 10^2$
(3)	$6{,}8607 * 10^5$	$8{,}1328 * 10^2$
(4)	$1{,}5090 * 10^6$	$7{,}5943 * 10^2$
(5)	$1{,}5090 * 10^6$	$8{,}2365 * 10^2$
(6)	$9{,}7087 * 10^5$	$7{,}3478 * 10^2$
(7)	$9{,}7087 * 10^5$	$7{,}5647 * 10^2$
(8)	$1{,}1539 * 10^6$	$8{,}4934 * 10^2$
(9)	$1{,}1539 * 10^6$	$7{,}6414 * 10^2$
(10)	$1{,}1539 * 10^6$	$1{,}2801 * 10^2$

Tabla 8.4: Comparación para la Evaluación de Proyecciones Numérica con Optimización

que la opción del análisis de todo el espacio de búsqueda es muy costosa en tiempo. Esto es posible gracias a almacenar el máximo y mínimo valor de cada variable en la base de datos.

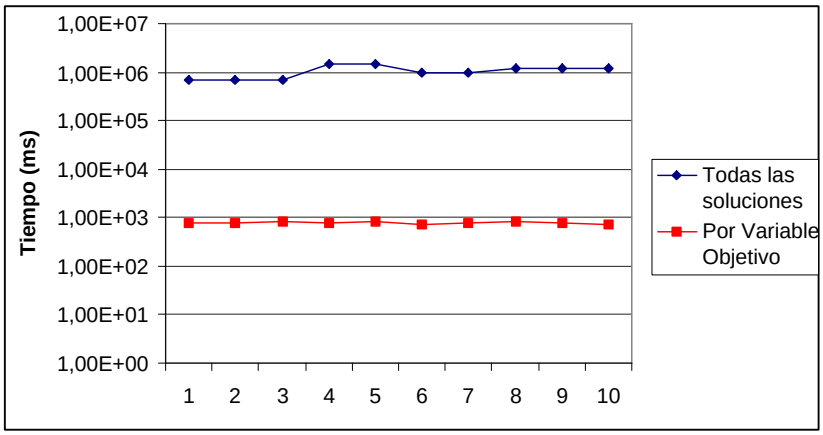

Figura 8.5: Tiempos de Evaluación para Proyección Numérica con Optimización

8.3. Los Sistemas de Información Geográfica y LORCDB

El área de los sistemas de información geográfica ha sido el caso de estudio más analizado en el tratamiento de las Bases de Datos de Restricciones, ya que las restricciones son una forma natural de representar elementos geométricos, propios de los sistemas de información geográfica. Es por esta razón, por la que en esta memoria de tesis se presenta cómo pueden ser tratados con la arquitectura LORCDB, y qué ventajas añade la utilización de dicha arquitectura. Las aplicaciones comerciales para el tratamiento de sistemas de información geográfica como Oracle Spatial^{TM}, SigPac^{TM} o ArcGis^{TM} utilizan dos formas para almacenar dichos datos: mediante mallas o celdas de bits (raster); o de forma vectorial (puntos, líneas y polígonos). Esta representación tiene tres claros inconvenientes: la saturación de la base de datos con gran cantidad de información; la pérdida de precisión, ya que se tendrá que aproximar la información con líneas; y la falta de expresividad, ya que las ecuaciones no se almacenan como tal. Derivado de estas desventajas, existe la problemática de una mayor complicación en las operación de modificación de la base de datos, ya que si para almacenar un determinado objeto son necesarios más registros, también más registros tendrán que modificarse si dicho objeto cambia o es eliminado. Con el objetivo de paliar dichos inconvenientes, las BDdR proponen sustituir las mallas y los vectores por restricciones.

Otra de las razones para abordar los sistemas de información geográfica en esta tesis, es la realización de las distintas pruebas con los operadores de selección, unión y diferencia. Dichos operadores son más complejos cuando en las relaciones sobre las que se opera hay involucradas restricciones con las mismas variables. Mientras que los sistemas para realizar la diagnosis basada en modelos están formados por muchas variables, los sistemas de información geográfica están formadas por muchas restricciones definidas sobre las mismas variables. Esto implica que los sistemas de información geográfica son un caso complejo de evaluación de la selección, unión y diferencia, ya que las comparaciones entre las restricciones de las distintas tuplas conllevará, en algunos casos, construir y resolver problemas de satisfacción de restricciones.

Lo primero para poder hacer consultas sobre un conjunto de restricciones de una Base de Datos de Restricciones es almacenarlas. Para construir un conjunto de tablas relacionadas con información geográfica, sobre las que hacer las consultas, se ha desarrollado

e implementado una aplicación que ayuda a la captura de puntos de un mapa con el objetivo de obtener las restricciones y almacenarlas en la Base de Datos de Restricciones. Toda la información sobre la aplicación se encuentra en la Sección B.2 del Apéndice B. Dicha aplicación utiliza el método de Lagrange para la aproximación mediante polinomios a un conjunto de puntos, dicho método utiliza la interpolación mediante mínimos cuadrados [54][7]. La utilización de polinomios también conlleva una aproximación de la realidad, pero que se acerca más a la información real y con menor número de restricciones que si se utilizan sólo líneas.

8.3.1. Información almacenada en la LORCDB

Para definir diversos ejemplos donde la selección, la unión y la diferencia se pueden ver involucradas, se crean cuatro tablas diferentes. Dichas tablas tendrán atributos restricción para almacenar la posición de un grupo de *Residenciales*, *Calles*, *Zonas de Censo* y *Obras* que se están realizando en la zona.

En la figura 8.6 se muestra la vista híbrida obtenida con *Google Maps* [105] de una zona de Sevilla, donde se pueden observar las distintas calles con sus nombres. Las restricciones concretas de estas 26 calles se encuentran en la figura B.6 del Apéndice B. Las calles se almacenarán en la tabla *Calles* creada con la sentencia:

```
CREATE TABLE Calles(Id Integer, Nombre String, Situacion Constraint, Tipo String)
```

Como atributos a destacar están el campo *Situacion* que almacena las restricciones que describe cada calle, y el *tipo* que definirá si es *'calle'* o *'avenida'*, tomando estas dos cadenas como posibles valores.

Para enriquecer las consultas, también se ha incluido la tabla *Residenciales* con los residenciales de la zona, creada mediante la sentencia:

```
CREATE TABLE Residenciales(Id Integer, Nombre String, Situacion Constraint)
```

En esta tabla, como atributo a destacar está el atributo restricción *Situacion* que almacena una combinación conjuntiva de inecuaciones lineales y polinómicas. El contorno de los residenciales se muestra en la figura 8.7, mientras que las restricciones de los 27 residenciales concretos se muestran en la figura B.7 del Apéndice B, y las envolventes de las variables en la figura B.8 del mismo apéndice.

Figura 8.6: Vista del Mapa Híbrida

También se creará en la base de datos otra tabla para almacenar los censos electorales de la zona, que se muestran en la figura 8.8, y cuyas restricciones concretas se encuentran en la figura B.9 del apéndice B. La tabla *Censos* será creada mediante la sentencia:

```
CREATE TABLE Censos(Id Integer, Nombre String, Situacion Constraint)
```

Cada una de las tres tablas creadas representa una capa sobre las variables (`x`, `y`), permitiendo almacenar, en poco espacio y manteniendo la simbología, la información relativa a calles, residenciales y censos.

Queda por describir la última tabla *Obras*, que se detallará cuando se analicen las operaciones de unión y diferencia, ya que sólo se utilizan en consultas de ese tipo.

8.4. Pruebas para el Operador de Selección

Partiendo de una relación donde hay involucradas restricciones, la selección devolverá un subconjunto horizontal de dicha relación con aquellas tuplas que cumplan una condición dada. Dicha condición debe estar relacionada con las restricciones que conforman la

Figura 8.7: Residenciales de la zona

tupla, utilizando los operadores $<$, $\leq$, $>$, $\geq$, $=$, $<>$ &, $\subseteq$, $\subset$, $\supseteq$ y $\supset$, presentados en el capítulo 4. En esta sección se presenta un conjunto de consultas con estos operadores de comparación, y las mejoras relacionadas con la evaluación del operador de selección. Los casos que se compararán son:

- **Sin Mejora:** Esto significa crear todos los CSP necesarios para la comparación de las restricciones. Si las restricciones estuvieran almacenadas sin información relativa a las envolventes de las variables, sería necesario crear un problema de satisfacción de restricciones para cada tupla con restricciones. Si la comparación se hiciera entre dos atributos restricción, eso conllevará realizar $n * m$ problemas de satisfacción de restricciones, donde n y m representan el número de tuplas de cada una de las relaciones.

- **Con Álgebra Relacional:** Lo que significa crear los CSP sólo con las tuplas que cumplan las condiciones establecidas con el álgebra relacional sobre los atributos clásicos. Si además de condiciones relativas a un atributo restricción existen condiciones relativas a atributos univaluados, no será necesario crear problemas de satisfacción de restricciones con aquellas tuplas que no cumplan las condiciones del

Figura 8.8: Censos de la zona

álgebra relacional clásica (**Estrategia 1**). Esta selección la realizará el gestor de bases de datos relacionales.

- **Analizando Envolventes:** Lo que implica crear los CSP sólo con las restricciones que cumplan las envolventes, derivado de la **Estrategia 3**. Analizando las envolventes de las restricciones se puede obtener mucha información sobre ella, y sólo para algunos casos será necesario crear un problema de satisfacción de restricciones. Estos casos ya han sido enumerados en la sección 6.2.

- **Añadiendo Envolventes:** Incluyendo los dominios de las variables en los CSP para acotar más la búsqueda, derivado también de la **Estrategia 3**. Ya que se tienen almacenadas las envolventes de las variables, éstas pueden ser incluidas en los problemas de satisfacción de restricciones para acotar más la búsqueda.

8.4.1. Ejemplos de Pruebas usando Selección

Se propone un conjunto de diez consultas con distintas operaciones de comparación entre restricciones, utilizando las tres tablas incluidas en la base de datos (*Residenciales*,

Calles y *Censos*). Se utilizarán cuatro estrategias diferentes, anotando el número de CSP que se tienen que crear y resolver para evaluar la selección, y el tiempo de ejecución de cada consulta. Dichos ejemplos utilizan condiciones de atributos clásicos combinados con atributos restricción. Entre las operaciones de comparación entre restricciones se encuentran todos los operadores de comparación descritos, junto a ejemplos donde la comparación se realiza con restricciones añadidas en la propia consulta, no pertenecientes a las relaciones almacenadas.

1. **¿Qué residenciales están al este de la Avenida de las Ciencias?** La descripción detallada de esta consulta será: obtener todos los residenciales `r` tal que para todo (`x`, `y`) que sea solución de la Avenida de las Ciencias y para todo (`x'`, `y`) que sea solución de `r`, siempre `x` $\leq$ `x'`. La consulta será:

```
SELECT Residenciales.Nombre FROM Residenciales, Calles
  WHERE Residenciales.Situacion >= (Calles.Situacion FOR x
  AND Calles.Situacion.y = Residenciales.Situacion.y)
  AND Calle.Nombre = 'Avd de las Ciencias'
```

2. **¿Qué residenciales están al oeste de la calle Flor de Albahaca?** La descripción detallada de esta consulta será: obtener todos los residenciales `r` tal que para todo (`x`, `y`) que sea solución de Flor de Albahaca y para todo (`x'`, `y`) que sea solución de `r`, siempre `x` $\geq$ `x'`. La consulta será:

```
SELECT Residenciales.Nombre FROM Residenciales, Calles
  WHERE Residenciales.Situacion <= (Calles.Situacion FOR x AND
  Calles.Situacion.y = Residenciales.Situacion.y)
  AND Calles.Nombre = 'Flor de Albahaca'
```

3. **¿Qué residenciales están al norte de la calle Flor de Salvia?** La descripción detallada de esta consulta será: obtener todos los residenciales `r` tal que para todo (`x`, `y`) que sea solución de Flor de Salvia y para todo (`x`, `y'`) que sea solución de `r`, siempre `y'` $\geq$ `y`. La consulta será:

```
SELECT Residenciales.Nombre FROM Residenciales, Calles
  WHERE Calles.Situacion <= (Residenciales.Situacion FOR y
```

```
AND Calles.Situacion.x = Residenciales.Situacion.x)
AND Calles.Nombre = 'Flor de Salvia'
```

4. **¿Qué residenciales están más al norte (tienen mayor latitud) que el residencial 'Entre Parques'?** La descripción detallada será: obtener todos los residenciales `r` tal que para todo (`x`, `y`) que sea solución de Entre Parques y para todo (`x'`, `y'`) que sea solución de `r`, siempre `y' > y`. La consulta será:

```
SELECT r.Nombre FROM Residenciales as r, Residenciales as r1
  WHERE r1.Nombre = 'Entre Parques'
  AND r.Situacion > (r1.Situacion FOR y)
```

5. **¿Qué residenciales están más al noreste que el residencial Galileo (están a más latitud y longitud)?** La descripción detallada será: obtener todos los residenciales `r` tal que para todo (`x`, `y`) que sea solución de Galileo y para todo (`x'`, `y'`) que sea solución de `r`, siempre `y' > y` y `x' > x`. La consulta será:

```
SELECT r1.Nombre FROM Residenciales as r, Residenciales as r1
  WHERE r.Nombre = 'Galileo'
  AND r.Situacion < r1.Situacion
```

6. **¿Qué calles están exclusivamente en el censo 2?**La descripción detallada será: obtener todas las calles `c` tal que para todo (`x`, `y`) que sea solución de `c` también lo es del censo 2. La consulta será:

```
SELECT Calles.Nombre FROM Censos, Calles
  WHERE Censos.Nombre = 'Censo 2'
  AND Calles.Situacion ⊆ Censos.Situacion
```

7. **¿Qué residenciales pertenecen al censo 3?** La descripción detallada será: obtener todos los residenciales `r` tal que para todo (`x`, `y`) que sea solución de `r` también lo es del censo 3. La consulta será:

```
SELECT Residenciales.Nombre FROM Residenciales, Censo
  WHERE Censos.Nombre = 'Censo 3'
  AND Censos.Situacion ⊇ Residenciales.Situacion
```

8. **¿Qué avenidas pertenecen al censo 2?** La descripción detallada será: obtener todos las calles `c` que sean de tipo avenida, donde existe un valor de (`x`, `y`) que es solución de `c` y también lo es del censo 2. La consulta será:

```
SELECT Calles.Nombre FROM Censo, Calles
  WHERE Censos.Nombre='Censo 2'
  AND Censos.Situacion & Calles.Situacion
  AND Calles.Tipo = 'avenida'
```

9. **¿Qué residenciales están más al norte (a mayor latitud) que la rotonda** $(x-177)^2+(y-288)^2 \leq 22^2$**?** Este es un ejemplo de inserción de restricciones en la consulta que no están previamente almacenadas en la base de datos. La consulta será:

```
SELECT Residenciales.Nombre FROM Residenciales
  WHERE Residenciales.Situacion > ((x-177)^2+(y-288)^2 <= 22^2 FOR y)
```

10. **¿Qué residenciales están más al este que la rotonda** $(x-177)^2+(y-288)^2 \leq 22^2$**?** La consulta será:

```
SELECT Residenciales.Nombre FROM Residenciales
  WHERE Residenciales.Situacion > ((x-177)^2+(y-288)^2 <= 22^2 FOR x)
```

Los datos relativos al tiempo y al número de CSP que se crean para cada selección se muestran en la tabla 8.5. En esta tabla se presentan los cuatro tipos de medias para el caso sin mejoras, y cómo el número de problemas de satisfacción de restricciones y el tiempo se van decrementando cada vez que se va añadiendo cada una de las tres mejoras. Esto significa que la última columna contiene los resultados cuando se utilizan las tres mejoras para la evaluación de la selección.

8.4.2. Conclusiones para el operador de Selección

Tras realizar todas las medidas temporales y crear los CSP necesarios, en la figura 8.9 se presenta gráficamente toda la información relativa a los tiempos de ejecución. Cada una de las medidias, representadas con distintos colores en el gráfico, muestra cómo decrementan los tiempos de ejecución cuando las mejoras se van incorporando. Los datos se presentan en

Consulta	Sin Mejora		Con Álgebra Relacional		Analizando Envolventes		Añadiendo Envolventes	
	Número de CSP	Tiempo (ms)	Número de CSP	Tiempo (ms)	Número de CSP	Tiempo (ms)	Número de CSP	Tiempo (ms)
(1)	26∗27	$5{,}0745{*}10^3$	27	$1{,}4755{*}10^3$	20	$7{,}6475{*}10^2$	20	$7{,}5334{*}10^2$
(2)	26∗27	$3{,}2044{*}10^3$	27	$1{,}4466{*}10^3$	3	$7{,}1346{*}10^2$	3	$6{,}9327{*}10^2$
(3)	26∗27	$4{,}3165{*}10^3$	27	$1{,}6655{*}10^3$	2	$6{,}8030{*}10^2$	2	$6{,}6032{*}10^2$
(4)	26∗27	$4{,}2456{*}10^3$	27	$9{,}2444{*}10^2$	0	2,0991	0	2,0991
(5)	26∗27	$8{,}5145{*}10^3$	27	$2{,}0255{*}10^3$	0	2,0971	0	2,0971
(6)	26∗3	$1{,}5598{*}10^3$	27	$1{,}4467{*}10^3$	12	$7{,}9477{*}10^2$	12	$7{,}9453{*}10^2$
(7)	27∗3	$1{,}6373{*}10^3$	27	$9{,}0247{*}10^2$	7	$7{,}9627{*}10^2$	7	$7{,}8622{*}10^2$
(8)	26∗3	$9{,}0265{*}10^2$	2	$7{,}0245{*}10^2$	2	$7{,}0236{*}10^2$	2	$7{,}0277{*}10^2$
(9)	27	$1{,}6808{*}10^3$	27	$1{,}6808{*}10^3$	0	4,1943	0	4,1943
(10)	27	$7{,}3295{*}10^2$	27	$7{,}3295{*}10^2$	0	3,1457	0	3,1457

Tabla 8.5: Comparación para la Evaluación de la Selección

milisegundos y utilizando la escala logarítmica para mostrar más claramente las diferencias temporales. Estas mejoras vienen derivadas de la reducción del número de problemas de satisfacción de restricciones, mostrada en la figura 8.10, también presentada utilizando la escala logarítmica. Como se puede observar en la gráfica de la figura 8.9, existen consultas donde se obtienen evaluaciones del orden de hasta1000 veces más eficientes que sin utilizar ninguna mejora. En casos como las consultas3, 4, 9, y 10, no es necesario construir ningún CSP ya que el simple análisis de la envolvente, que no es más que una consulta clásica sobre atributos numéricos, es suficiente. También se puede observar que siempre el número de CSP del tercer (-▲-) y cuarto (-×-) caso son iguales, y el tiempo de evaluación muy similar, ya que la cuarta mejora sólo implica incluir las envolventes, no reducir el número de CSP. Si no se hiciera referencia a atributos clásicos en la condición, el tiempo y el número de los CSP construidos comparando la primera y segunda prueba para cada ejemplo, serían iguales.

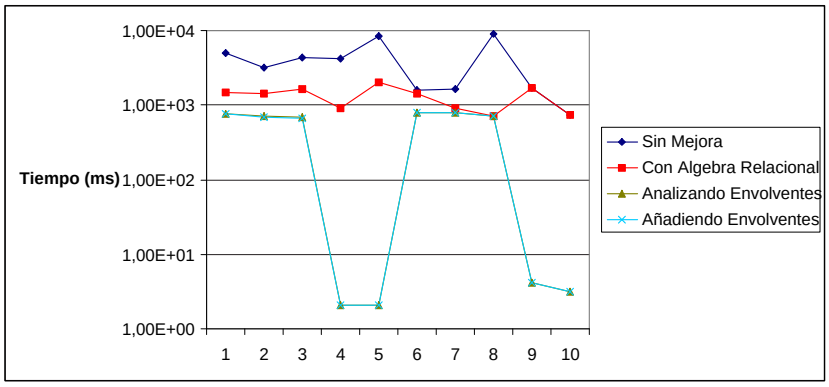

Figura 8.9: Tiempos de Evaluación para la Selección

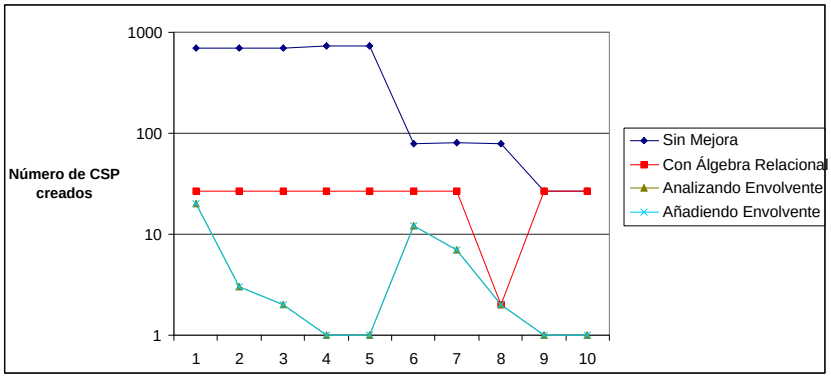

Figura 8.10: Número de CSP creados para la Selección

8.5. Pruebas para el Operador de Unión

La unión entre dos relaciones devolverá una nueva relación con las tuplas de ambas sin repetición. Cuando en las tuplas involucradas en la operación de unión hay atributos de tipo restricción, es necesario analizar dichos atributos para cerciorarse de que las tuplas con dichas restricciones no son iguales. La comparación entre dos restricciones no pueden ser exclusivamente morfológica, ya que dos restricciones pueden tener las mismas soluciones y estar representadas de forma distinta, por ejemplo $2y = 2x + 2$ y $y = x + 1$. Cómo comparar dos tuplas con atributos restricción de una manera eficiente se ha explicado en la sección 6.5.

Para esta operación se proponen 10 ejemplos, donde se muestra la comparativa añadiendo cada una de las estrategias de optimización. Las comparaciones se realizarán sobre los casos:

- **Sin Mejora:** Creando todos los posibles CSP entre dos relaciones R_1, R_2 con restricciones, siendo necesario crear un CSP con una restricción de R_1 y otra de R_2 para conocer si comparten todas las soluciones. Esto significa que existen $n * m$ posibilidades donde m y n son el número de tuplas de R_1 y R_2 respectivamente.
- **Con Álgebra Relacional:** Creando CSP con las tuplas que tengan todos los atributos clásicos iguales, utilizando la **Estrategia 1**. Si una tupla de R_1 y otra de R_2 tienen al menos uno de sus atributos clásicos o univaluados distintos, no sería necesario crear un CSP para esa opción ya que independientemente de las restricciones, dichas tuplas son distintas y ambas formarán parte de la relación de salida.
- **Analizando las Envolventes:** Creando los CSP sólo con las restricciones que cumplan las envolventes, como se explica en la sección 6.5. La envolvente de las variables de las restricciones da información sobre si las restricciones pueden compartir o no soluciones. Si los valores definidos por las envolventes de las restricciones no tienen ningún valor en común o sólo algunos, no será necesario crear un CSP con esas restricciones ya que seguro que las restricciones no son iguales. Sin embargo, si las envolventes de las variables de una restricción están incluidas en la otra, será necesario comprobar mediante un CSP, si realmente una restricción está incluida dentro de otra aunque lo estén sus envolventes.

Para facilitar las consultas y enriquecer los ejemplos se añade una nueva tabla *Obras* a la Base de Datos de Restricciones. La tabla *Obras* almacenará 25 zonas donde se van a realizar obras en los próximos años, como se muestra en la figura 8.11. Los detalles de cada una de las restricciones almacenadas en la tabla *Obras* se muestran en la figura B.10 del Apéndice B.

8.5.1. Ejemplos de Pruebas usando la Unión

En el análisis de tiempos para los próximos ejemplos, sólo tendrán en cuenta el tiempo de la evaluación de la operación de unión, una vez que las relaciones de entrada se han obtenido.

1. Obtener la situación (atributo `Situacion`) de las calles y residenciales. La consulta será:

Figura 8.11: Obras de la zona

```
SELECT Calles.Situacion FROM Calles
UNION
SELECT Residenciales.Situacion FROM Residenciales
```

2. Para los residenciales y la calles, obtener los identificadores de censos a los que pertenecen y su situación. La consulta será:

```
SELECT Censos.Id, Calles.Situacion FROM Censos, Calles
  WHERE Calles.Situacion ⊆ Censos.Situacion
UNION
SELECT Censos.Id, Residenciales.Situacion FROM Censos, Residenciales
  WHERE Residenciales.Situacion ⊆ Censos.Situacion
```

3. Obtener los identificadores de obras y las situaciones de las calles y residenciales, ambas del censo 1, que les afecte alguna obra. La consulta será:

```
SELECT Obras.Id, Calles.Situacion FROM Obras, Calles, Censos
  WHERE Calles.Situacion & Obras.Situacion
  AND Calles.Situacion ⊆ Censos.Situacion AND Censoa.Id = 1
```

```
UNION
SELECT Obras.Id, Residenciales.Situacion
  FROM Obras, Residenciales, Censos
  WHERE Residenciales.Situacion & Obras.Situacion
  AND Residenciales.Situacion ⊆ Censos.Situacion AND Censos.Id = 1
```

4. Obtener los identificadores de obras y las situaciones de las calles y residenciales que les afecte alguna obra. La consulta será:

```
SELECT Obras.Id, Calles.Situacion FROM Obras, Calles
  WHERE Calles.Situacion & Obras.Situacion
UNION
SELECT Obras.Id, Residenciales.Situacion FROM Obras, Residenciales
  WHERE Residenciales.Situacion & Obras.Situacion
```

5. Obtener los identificadores de censo y las situaciones de las obras y las calles. La consulta será:

```
SELECT Censos.Id, Obras.Situacion FROM Obras, Censos
  WHERE Obras.Situacion & Censos.Situacion
UNION
SELECT Censos.Id, Calles.Situacion FROM Calles, Censos
  WHERE Calles.Situacion & Censos.Situacion
```

6. Obtener los identificadores de censo y las situaciones de las obras y los residenciales. La consulta será:

```
SELECT Censos.Id, Obras.Situacion FROM Obras, Censos
  WHERE Obras.Situacion & Censos.Situacion
UNION
SELECT Censos.Id, Residenciales.Situacion FROM Residenciales, Censos
  WHERE Residenciales.Situacion & Censos.Situacion
```

7. Obtener los identificadores de censo y las situaciones de las obras del censo1 y de las calles de cualquier censo. La consulta será:

```
SELECT Censos.Id, Obras.Situacion FROM Obras, Censos
  WHERE Obras.Situacion & Censos.Situacion AND Censos.Id= 1
UNION
SELECT Censos.Id, Calles.Situacion FROM Calles, Censos
  WHERE Calles.Situacion & Censos.Situacion
```

8. Obtener los identificadores de censo y las situaciones de las obras del censo2 y de los residenciales de cualquier censo. La consulta será:

```
SELECT Censos.Id, Obras.Situacion FROM Obras, censo
  WHERE Obras.Situacion & Censos.Situacion AND Censos.Id = 2
UNION
SELECT Censos.Id, Residenciales.Situacion FROM residenciales, censo
  WHERE Residenciales.Situacion & Censos.Situacion
```

9. Obtener los identificadores de censo y las situaciones de las obras y las calles incluidas en el censo 2. La consulta será:

```
SELECT Censos.Id, Obras.Situacion FROM obras, censo
  WHERE Obras.Situacion & Censos.Situacion
UNION
SELECT Censos.Id, Calles.Situaciont FROM Calles, Censos
  WHERE Calles.Situacion ⊆ Censos.Situacion AND Censos.Id = 2
```

10. Obtener los identificadores de censo y las situaciones de las obras de cualquier censo y de los residenciales incluidos en el censo 3. La consulta será:

```
SELECT Censos.Id, Obras.Situacion FROM Obras, censo
  WHERE Obras.Situacion & Censos.Situacion
UNION
SELECT Censos.Id, Residenciales.Situacion FROM residenciales, censo
  WHERE Residenciales.Situacion ⊆ Censos.Situacion AND Censos.Id = 3
```

Toda la información relativa a los tiempos medios de ejecución y el número de CSP que se crean para cada uno de los ejemplos se muestra en la tabla 8.6.

Consulta	Sin Mejora		Con Álgebra Relacional		Analizando Envolventes	
	Número de CSP	Tiempo (ms)	Número de CSP	Tiempo (ms)	Número de CSP	Tiempo (ms)
(1)	702	$5{,}6953*10^3$	702	$5{,}6953*10^3$	28	$9{,}2012*10^2$
(2)	702	$6{,}0534*10^3$	215	$2{,}3855*10^3$	11	$8{,}7556*10^2$
(3)	49	$1{,}0286*10^3$	22	$8{,}7765*10^2$	3	$7{,}2142*10^2$
(4)	336	$2{,}4798*10^3$	24	$9{,}4371*10^2$	11	$1{,}4239*10^2$
(5)	650	$4{,}8486*10^3$	146	$1{,}6053*10^3$	7	$7{,}9691*10^2$
(6)	650	$4{,}9985*10^3$	158	$1{,}7028*10^3$	4	$7{,}9062*10^2$
(7)	182	$2{,}4264*10^3$	49	$9{,}9509*10^2$	4	$7{,}4448*10^2$
(8)	104	$1{,}4627*10^3$	32	$1{,}0255*10^3$	0	$5{,}1799*10^1$
(9)	300	$2{,}6361*10^3$	48	$1{,}0684*10^3$	2	$7{,}0182*10^2$
(10)	275	$3{,}2747*10^3$	77	$1{,}2572*10^3$	0	$5{,}3372*10^1$

Tabla 8.6: Comparación para la evaluación de la Unión

8.5.2. Conclusiones para el operador de Unión

La información contenida en la tabla 8.6 se presenta de forma gráfica en las figuras 8.12 y 8.13. El tiempo de evaluación del operador de unión para cada ejemplo (presentado en la figura 8.6) está estrechamente ligado con el número de CSP necesarios para resolverlos (presentado en la figura 8.13), ambas presentadas utilizando la escala logarítmica. Dichas gráficas muestran en distintos colores los datos relativos a cada una de las mejoras propuestas: sin ninguna mejora creando todos los CSP (-♦-); creando CSP sólo cuando los atributos clásicos son iguales (-■-); y creando CSP sólo en aquellos casos donde la envolvente de una de las restricciones está incluida en la envolvente de otra (-▲-).

Cabe destacar casos como el del ejemplo 1, donde el uso del álgebra relacional no decrementa el número de CSP que se crean y por lo tanto tampoco el tiempo de evaluación. Esto ocurre cuando las relaciones no tienen atributos univaluados o todos ellos son iguales en las distintas tuplas. En la figura 8.12 se puede observar cómo para esta

operación la **Estrategia 1** puede realizar la evaluación hasta tres veces más rápido que sin mejoras. Sin embargo, para este ejemplo la mejora más sustancial es el Análisis de Envolventes (derivada de la **Estrategia 3**) que en los ejemplos, y combinándola con el álgebra relacional, ha llegado a ser hasta sesenta veces más eficiente en tiempo.

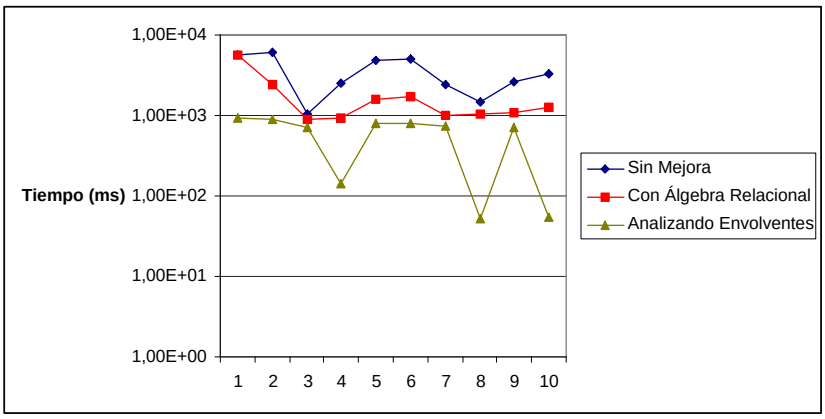

Figura 8.12: Tiempos de evaluación para la Unión

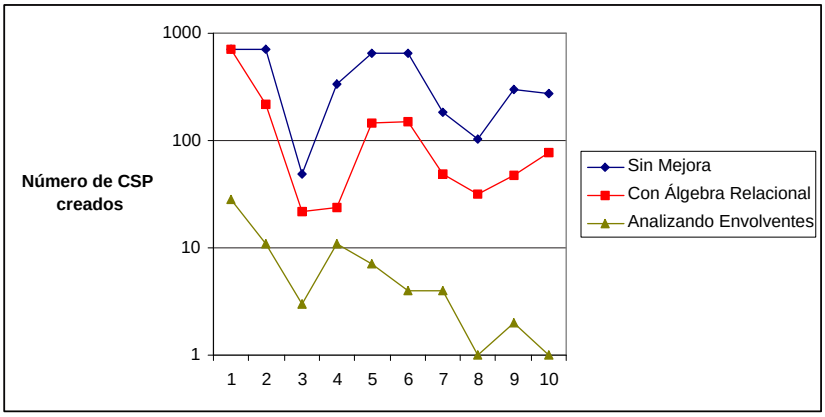

Figura 8.13: Número de CSP creados para la Unión

8.6. Pruebas para el Operador de Diferencia

La diferencia entre dos relaciones R_1 y R_2 devolverá una nueva relación con las tuplas de R_1 que no estén en R_2. Cuando en las tuplas involucradas en la operación de diferencia hay atributos de tipo restricción, hay que hacer un análisis específico de ellas para cerciorarse que las tuplas con dichas restricciones no comparten soluciones, para no incluir en la

salida tuplas de R_2. Cómo realizar la evaluación de la diferencia de una manera eficiente se ha explicado en la sección 6.6.

Para esta operación se proponen 10 ejemplos, donde se muestra la comparativa añadiendo cada una de las mejoras, iguales a las de la operación de unión. Las comparaciones se realizarán sobre los casos:

- **Sin Mejora:** Creando todos los posibles CSP, lo que al igual que en la Unión significa $n * m$ posibilidades, donde m y n son el número de tuplas de R_1 y R_2 respectivamente.
- **Con Álgebra Relacional:** Creando CSP sólo con las tuplas que tengan todos los atributos clásicos iguales, derivado de la **Estrategia 1**. Si una tupla de R_1 y otra de R_2 tienen al menos unos de sus atributos clásicos o univaluados distintos, no sería necesario crear un CSP para esa opción, ya que independientemente de las restricciones, dichas tuplas son distintas, lo que significa que la tupla deR_1 formarán parte de la relación de salida.
- **Analizando las Envolventes:** Creando sólo CSP con las restricciones que cumplan las envolventes cuyos detalles se explicaron en la sección 6.6, derivado de la **Estrategia 3**.

8.6.1. Ejemplos de Pruebas usando la Diferencia

Los ejemplos para la operación de la Diferencia también se realizarán sobre las tablas *Residenciales*, *Calles*, *Censos* y *Obras*. Los tiempos presentados para los próximos ejemplos sólo tienen en cuenta el tiempo de ejecución de la diferencia, una vez de las relaciones de entrada se han obtenido.

1. Obtener la situación de las obras incluidas en el censo 1 menos la situación de las calles, también del censo 1, donde hay una obra. La consulta será:

```
SELECT Obras.Id, Obras.Situacion FROM Obras, Censos
  WHERE Obras.Situacion ⊆ censos.Situacion AND Censos.Id = 1
MINUS
SELECT Obras.Id, Calles.Situacion FROM Obras, Censos, Calles
```

```
WHERE Calles.Situacion ⊆censos.Situacion AND
Calles.Situacion ⊆ Obras.Situacion AND Censos.Id = 1
```

2. Obtener el identificador y la situación de las obras incluidas en el censo 3 que no estén incluidas en la situación del censo 2. La consulta será:

```
SELECT Obras.Id, Obras.Situacion FROM Obras, Censos
  WHERE Obras.Situacion ⊆ Censos.Situacion AND Censos.Id = 3
MINUS
SELECT Obras.Id, Obras.Situacion FROM Obras, Censos
  WHERE Obras.Situacion ⊆ Censos.Situacion AND Censos.Id = 2
```

3. Obtener el identificador del censo y la situación de las obras incluidas en el censo 1, menos la situación de los residenciales incluidas en cualquier censo. La consulta será:

```
SELECT Censos.Id, Obras.Situacion FROM Obras, Censos
  WHERE Obras.Situacion ⊆ Censos.Situacion AND Censos.Id = 1
MINUS
SELECT Censos.Id, Residenciales.Situacion FROM Censos, residenciales
  WHERE Residenciales.Situacion ⊆ Censos.Situacion
```

4. Obtener el censo y la situación de las obras incluidas en el censo2, menos la situación de los residenciales de cualquier censo. La consulta será:

```
SELECT Censos.Id, Obras.Situacion FROM Obras, Censos
  WHERE Obras.Situacion ⊆ Censos.Situacion AND Censos.Id = 2
MINUS
SELECT Censos.Id, Residenciales.Situacion FROM Censos, residenciales
  WHERE Residenciales.Situacion ⊆ Censos.Situacion
```

5. Obtener el identificador del censo y la situación de los residenciales incluidos en el censo 1, menos la situación de las obras de cualquier censo. La consulta será:

```
SELECT Censos.Id, Residenciales.Situacion FROM Censos, residenciales
```

```
  WHERE Residenciales.Situacion ⊆ Censos.Situacion AND Censos.Id = 1
MINUS
SELECT Censos.Id, Obras.Situacion FROM Obras, Censos
  WHERE Obras.Situacion ⊆ Censos.Situacion
```

6. Obtener el identificador del censo y la situación de las obras incluidas en el censo1, menos la situación de las calles incluidas en cualquier censo. La consulta será:

```
SELECT Censos.Id, Obras.Situacion FROM Obras, Censos
  WHERE Obras.Situacion ⊆ Censos.Situacion AND Censos.Id = 1
MINUS
SELECT Censos.Id, Calles.Situacion FROM Censos, Calles
  WHERE Calles.Situacion ⊆ Censos.Situacion
```

7. Obtener el identificador del censo y la situación de las obras incluidas en el censo2, menos la situación de las calles incluidas en cualquier censo. La consulta será:

```
SELECT Censos.Id, Obras.Situacion FROM Obras, Censos
  WHERE Obras.Situacion ⊆ Censos.Situacion AND Censos.Id = 2
MINUS
SELECT Censos.Id, Calles.Situacion FROM Censos, Calles
  WHERE Calles.Situacion ⊆ Censos.Situacion
```

8. Obtener el identificador del censo y la situación de las calles incluidas en el censo2, menos la situación de las obras incluidas en cualquier censo. La consulta será:

```
SELECT Censos.Id, Calles.Situacion FROM Censos, Calles
  WHERE Calles.Situacion ⊆ Censos.Situacion AND Censos.Id = 2
MINUS
SELECT Censos.Id, Obras.Situacion FROM Obras, Censos
  WHERE Obras.Situacion ⊆ Censos.Situacion
```

9. Obtener el identificador del censo y la situación de las calles, menos la situación de las obras incluidas en cualquier censo. La consulta será:

```
SELECT Censos.Id, Calles.Situacion FROM Censos, Calles
  WHERE Calles.Situacion ⊆ Censos.Situacion
MINUS
SELECT Censos.Id, Obras.Situacion FROM Obras, Censos
  WHERE Obras.Situacion ⊆ Censos.Situacion
```

10. Obtener el identificador del censo y la situación de los residenciales, menos la situación de las obras incluidas en cualquier censos. La consulta será:

```
SELECT Censos.Id, Residenciales.Situacion FROM Censos, residenciales
  WHERE Residenciales.Situacion ⊆ Censos.Situacion
MINUS
SELECT Censos.Id, Obras.Situacion FROM Obras, Censos
  WHERE Obras.Situacion ⊆ Censos.Situacion
```

Toda la información relativa a los tiempos medios de ejecución y el número de CSP que se crean para cada uno de los ejemplos se muestra en la tabla 8.7.

8.6.2. Conclusiones para el operador de Diferencia

Para mostrar de forma gráfica los datos presentados en la tabla 8.7 para los10 ejemplos, se han creado las figuras 8.14 y 8.15, ambas utilizando la escala logarítmica. En cada una de estas tablas se utilizan diferentes colores para presentar cómo las distintas mejoras van decrementando el número de CSP que se generan, y por lo tanto el tiempo de evaluación: creando todos los CSP (-♦-); sólo creando CSP cuando los atributos clásicos son iguales (-■-); y sólo creando CSP con aquellos casos donde las cajas de consistencia definidas por las envolventes tienen alguna solución en común (-▲-). Para los ejemplos y con la utilización del álgebra relacional, se obtienen mejoras el doble de eficientes que sin mejoras. Con respecto al análisis de envolventes, se obtienen evaluaciones de consultas casi tres veces más rápidas que sin utilizar ninguna estrategia de optimización.

Consulta	Sin Mejora		Con Álgebra Relacional		Analizando Envolventes	
	Número de CSP	Tiempo (ms)	Número de CSP	Tiempo (ms)	Número de CSP	Tiempo (ms)
(1)	35	$1{,}4040*10^3$	4	$7{,}2142*10^2$	4	$7{,}2142*10^2$
(2)	80	$8{,}4934*10^2$	3	$7{,}3610*10^2$	3	$7{,}2771*10^2$
(3)	240	$1{,}9314*10^3$	64	$8{,}9851*10^2$	15	$7{,}8433*10^2$
(4)	240	$1{,}3004*10^3$	64	$9{,}1654*10^2$	14	$7{,}6751*10^2$
(5)	300	$1{,}9980*10^3$	140	$1{,}0863*10^3$	29	$8{,}7034*10^2$
(6)	240	$1{,}7909*10^3$	72	$8{,}7665*10^2$	24	$7{,}8852*10^2$
(7)	240	$1{,}1709*10^3$	109	$9{,}45099*10^2$	40	$8{,}4751*10^2$
(8)	403	$2{,}2668*10^3$	104	$9{,}2458*10^2$	40	$8{,}0264*10^2$
(9)	930	$2{,}3651*10^3$	256	$1{,}1557*10^3$	92	$8{,}8234*10^2$
(10)	930	$3{,}4534*10^3$	268	$1{,}3179*10^3$	58	$9{,}0085*10^2$

Tabla 8.7: Comparación para la evaluación de la Diferencia

8.7. Resumen

En este capítulo se presentan un conjunto de ejemplos para las operaciones primitivas del álgebra relacional redefinidas en esta tesis. Dichas pruebas utilizan dos casos de estudio, la diagnosis basada en modelos para la proyección simbólica y numérica, y los sistemas de información geográfica para la selección, unión y diferencia. Se han utilizado casos de estudio diferentes para analizar los casos computacionalmente más complejos en cada operación, muchas variables para la proyección, y pocas variables en el resto de los casos. Para cada una de las operaciones se ha realizado la comparativa añadiendo las distintas mejoras propuestas derivadas del diseño de la arquitectura LORCDB: utilización del álgebra relacional, indexación de variables, almacenar las envolventes de las variables, y etiquetado de restricciones.

Entre las cuatro operaciones, se han realizado176 pruebas, cada una de ellas ejecutadas 100 veces para obtener los tiempos medios de evaluación. El análisis de las operaciones se ha realizado de manera separada, lo que significa que al combinar las distintas operaciones en una misma consulta, también se mejorará el tiempo de evaluación global.

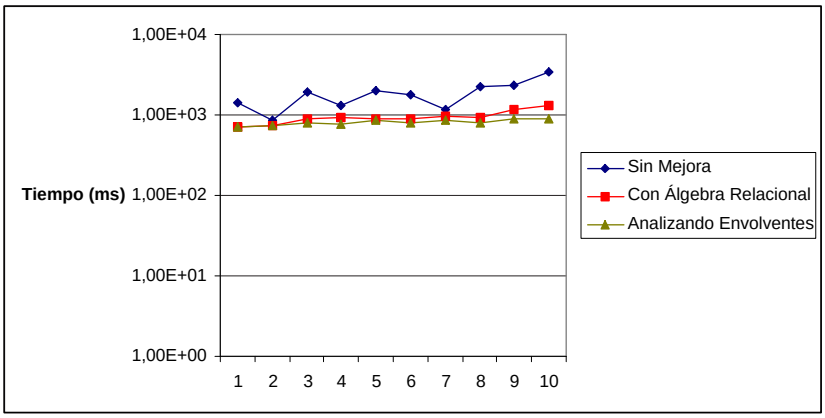

Figura 8.14: Tiempos de evaluación de la Diferencia

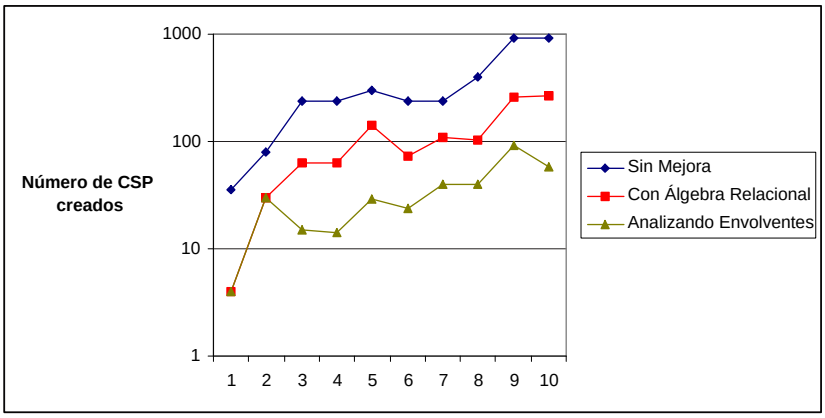

Figura 8.15: Número de CSP creados para la Diferencia

Cabe destacar que la operación que más decrementa el tiempo de evaluación es la proyección, tanto simbólica como numérica. Esto es posible gracias a la utilización del algoritmo para la detección de los Conjuntos de Restricciones Relacionadas por Variables en el caso simbólico, y el uso de las distintas heurísticas en el caso de la proyección numérica.

Capítulo 9

Conclusiones y Trabajos Futuros

En esta tesis se han analizado las necesidades existentes en las bases de datos para el tratamiento de información intensiva, mediante su representación utilizando restricciones. Se han analizado las soluciones existentes hasta el momento, especialmente en el ámbito de las Bases de Datos de Restricciones. Dicho análisis ha dado origen a la detección de un conjunto de deficiencias en las propuestas existentes hasta el momento, y un conjunto de propuestas para mejorar sus inconvenientes. En este capítulo se realiza un resumen de los objetivos cumplidos en esta tesis, qué publicaciones se han derivado de ellos, y las líneas de investigación abiertas que aún quedan por cumplir.

9.1. Análisis de la Consecución de los Objetivos

El principal objetivo definido en esta tesis es:

Ampliar los Gestores de Bases de Datos para el tratamiento de Información compleja representada mediante Restricciones

Como consecuencia de este objetivo, se derivan un conjunto de subobjetivos para conseguirlo:

- **Analizar las necesidades de las Bases de Datos actuales y sus deficiencias con respecto al tratamiento de datos complejos**

  Las bases de datos relacionales tienen cualidades para el tratamiento de datos extensivos, pero existen grandes deficiencias en lo concerniente a los datos intensivos.

De esta forma, se analizan los distintos gestores de bases de datos para dar cabida a las aplicaciones que necesitan representación intensiva de la información.

- **Analizar las Bases de Datos de Restricciones y sus deficiencias**

  Una de las formas de tratar dichos datos intensivos es utilizando las Bases de Datos de Restricciones, que presentan el centro de estudio de esta memoria. Este área de investigación ha dado como origen un conjunto de propuestas, analizadas en esta tesis con el objetivo de detectar qué características pueden ser mejoradas. El objetivo de esta tesis ha sido encontrar y subsanar dichos inconvenientes, definiendo un Gestos de Bases de Datos no dependiente de la aplicación.

- **Definir un nuevo modelo de Base de Datos con Restricciones**

  Derivado de los defectos detectados, se define un nuevo modelo de datos para la redefinición de las Bases de Datos de Restricciones. Dicha definición implica la separación de la información intensiva de la extensiva en dos tipos de atributos distintos, con formas de representación y tratamientos diferentes.

- **Ampliar la sintaxis y la semántica de SQL para tratar Restricciones**

  Al tener diferentes tipos de datos, se modifica la sintaxis y la semántica de SQL para incluir las restricciones y las variables de restricciones como nuevos atributos que pueden participar en las consultas. Para ello se ha redefinido la semántica de algunos de los operadores de comparación ya existentes, y se han incluido nuevos operadores para definir las operaciones sobre restricciones. Uno de los defectos de las otras propuestas de Bases de Datos de Restricciones radica en que las soluciones estaban diseñadas en función de la aplicación a la que daban soporte, los sistemas de información geográfica mayoritariamente. Sin embargo, en esta tesis se ha propuesto un lenguaje de consulta independiente de la aplicación, que se puede utilizar siempre que los datos se puedan representar como restricciones polinómicas, utilizando inecuaciones o ecuaciones.

- **Diseñar una arquitectura para las Bases de Datos con Restricciones**

  Para que la propuesta no se quede sólo a nivel teórico, se propone una arquitectura para las Bases de Datos de Restricciones utilizando el paradigma objeto-relacional, junto a un conjunto de tablas de indexación para facilitar el tratamiento de las

restricciones almacenadas. Dentro del Sistema Gestor de Bases de Datos de Restricciones, se incluyen un conjunto de herramientas y estrategias de resolución de consultas, para que la funcionalidad relativa a las restricciones sea resuelta por el gestor. Esto provoca que las aplicaciones queden liberadas de la complejidad y tratamiento de las restricciones.

- **Implementación del lenguaje de Consulta**

  Gracias a la arquitectura propuesta y un conjunto de algoritmos, se han implementado de una manera eficiente cada una de las operaciones primitivas del álgebra relacional orientadas a restricciones. Para mostrar de forma empírica cómo las decisiones de diseño e implementación reducen el tiempo de evaluación de las consultas, se utilizan diferentes casos de estudios y usos de operadores, mostrando los tiempos medios de evaluación para cada uno de ellos.

9.2. Principales Aportaciones

Cabe destacar un subconjunto de las propuestas de esta tesis, como las aportaciones más relevantes en al área de los Gestores de Bases de Datos.

9.2.1. Redefinición del modelo de Bases de Datos de Restricciones

Utilizando la capacidad de extracción del álgebra relacional, y la capacidad expresiva y de inferencia de las programación con restricciones, se crea un nuevo modelo con tres tipos de atributos: clásico o univaluado, restricción, y atributo de restricción. Esto provoca una redefinición del modelo de Bases de Datos de Restricciones, junto a las operaciones de selección, proyección, producto cartesiano, unión y diferencia. Estas consultas modifican necesariamente la sintaxis de SQL, intentando mantener su sintaxis y su semántica en lo posible, facilitando así adaptación de los usuarios que conocen SQL. Este nuevo lenguaje llamado CORQL, junto a los detalles relativos al nuevo modelo, se presentan en el capítulo 4.

9.2.2. Creación de un nuevo Gestor de Bases de Datos de Restricciones

En el capítulo 5 se presenta la arquitectura LORCDB, la cual permite mayor expresividad que las bases de datos relacionales al utilizar datos intensivos, haciendo a la vez la evaluación de consultas más eficiente. Esto es gracias a ciertas decisiones de diseño, como la indexación de variables, almacenado de los límite de los valores de satisfactibilidad, y etiquetado de las restricciones en función de su tipo. El gestor crea de forma automática y transparente al usuario, los modelos necesarios para evaluar las consultas. La arquitectura incluye las herramientas necesarias para resolver dichos modelos, y presentar la salida al usuario.

9.2.3. Implementación eficiente del lenguaje CORQL

Para la mejora de la eficiencia cuando se tratan datos intensivos y extensivos, se proponen un conjunto de algoritmos y estrategias de evaluación de las consultas. La filosofía de las estrategias propuestas radica en posponer el uso de técnicas simbólicas o resolución de problemas de satisfacción de restricciones. Esto es posible gracias los distintos algoritmos para la detección de restricciones relacionadas por variables, y al análisis de envolventes para conocer la satisfactibilidad de un conjunto de restricciones, tal como se analiza en el capítulo 6.

9.2.4. Gestor válido para aplicaciones de naturaleza diferente

Tres son los ejemplos de aplicaciones utilizados en esta memoria, para mostrar cómo la misma arquitectura y el mismo lenguaje son válidos para distintos usos. Dichas aplicaciones son los sistemas de información geográfica, gestión económica, y la diagnosis de fallos. Cabe destacar la diagnosis de fallos basada en modelos como ejemplo novedoso. La diagnosis basada en modelos nunca había sido utilizada antes como caso de estudio en las Bases de Datos de Restricciones, siendo un problema emergente y de gran importancia en el mundo de la ingeniería. El uso de este caso de estudio hace posible ampliar tipos de consultas, especialmente orientadas a los atributos variable de restricción, no definidos hasta el momento. Esto es debido a que los sistemas a diagnosticar suelen ser complejos y están formados por gran cantidad de variables, tal como se explica en el capítulo 7.

9.3. Publicaciones

Durante el desarrollo de esta tesis se han publicado trabajos tanto en el área de la bases de datos, de la diagnosis de fallos, como específicamente en el campo de las Bases de Datos de Restricciones. Dichas aportaciones presentadas cronológicamente son:

[**2003**]**:** El objetivo cubierto en este año se basó en la creación de una base de datos relacional que permitieran almacenar restricciones. En función de la consulta a la base de datos, se construían y resolvían CSP en tiempo de ejecución. Las restricciones eran almacenadas como cadenas en la base de datos, lo que significa que se podían hacer consultas con respecto a las restricciones, pero no a sus variables. En estos primeros trabajos, sólo era posible trabajar con restricciones lineales, lo que venía determinado por el tipo de resolutor que se utilizaba.

- **Título:** Arquitectura para la Consulta a Bases de Datos Restrictivas.

  Autores: María Teresa Gómez, Rafael M. Gasca, Carmelo Del Valle y Rafael Ceballos

  Publicado en: Jornadas de Ingeniería del Software y Bases de Datos, páginas: 593-602, 2003

  Índice de Aceptación: 24.6 %

- **Título:** CSP y Bases de Datos Restrictivas.

  Autores: María Teresa Gómez, Rafael M. Gasca, Carmelo Del Valle y Rafael Ceballos

  Publicado en: Revista Iberoamericana de Inteligencia Artificial, Volumen20, páginas: 150-163, 2003

- **Título:** CSP aplicados a la diagnosis basada en modelos.

  Autores: Rafael Ceballos, Carmelo Del Valle, María Teresa Gómez y Rafael M. Gasca.

  Publicado en: Revista Iberoamericana de Inteligencia Artificial, Volumen20, páginas: 137-150, 2003

- **Título:** Modelo de CSP para consultas a Bases de Datos Restrictivas

Autores: María Teresa Gómez, Rafael M. Gasca, Carmelo Del Valle y Antonio Márquez

Publicado en: X Conferencia de la Asociación Española Para la Inteligencia Artificial (CAEPIA), Volumen I, páginas: 469-472, 2003

[**2004**]**:** Los trabajos desarrollados durante este año se centran en resolver el problema de la diagnosis de componentes basada en modelos, almacenando toda la información en Bases de Datos de Restricciones. Se ofrece una arquitectura que permite conocer los componentes que fallan en un sistema cuando éste está almacenado como restricciones en una base de datos relacional. Se aportan los algoritmos para mejorar la eficiencia de la proyección para atributos variable de restricción, e incluyendo técnicas simbólicas al proceso.

- **Título:** Determination of Possible Minimal Conflict Sets Using Constraint Databases Technology and Clustering.

  Autores: M. T. Gómez-López, R. Ceballos, R. M. Gasca y S. Pozo.

  Publicado en: IBERAMIA, Lecture Notes in Computer Science, Springer, Volumen: 3315, páginas: 942-952, 2004.

  Índice de Aceptación: 31 %

- **Título:** Applying Constraint Databases in the Determination of Potential Minimal Conflicts to Polynomial Model-based Diagnosis.

  Autores: María Teresa Gómez, R. Ceballos, R. M. Gasca y C. Del Valle

  Publicado en: Constraint Databases (CDB), Lecture Notes in Computer Science, Springer, Volumen: 3074, páginas: 74-87, 2004.

  Índice de Aceptación: 34 %

- **Título:** Constraint Databases Technology for Polynomial Models Diagnosis

  Autores: María Teresa Gómez-López, Rafael Ceballos, Rafael M. Gasca y Carmelo Del Valle

  Publicado en: 15th International Workshop on Principles of Diagnosis (DX'04), páginas: 215-220, 2004

- **Título:** Determination of Possible Minimal Conflict Sets Using Components Clusters and Gröbner Bases.

  Autores: Rafael Ceballos, María Teresa Gómez-López, Rafael M. Gasca y S. Pozo

  Publicado en: 15th International Workshop on Principles of Diagnosis (DX'04), páginas: 21-26, 2004

[**2005**]**:** Durante este año se crea la primera versión de la arquitectura usando el paradigma objeto-relacional. Dicha arquitectura permite crear problemas de satisfacción de restricciones e inferir nuevas restricciones partiendo de las almacenadas en la base de datos utilizando técnicas simbólicas. El caso de estudio continúa siendo la diagnosis basada en modelos, incorporando todas la mejoras obtenidas en el año anterior. En estos trabajos las restricciones son almacenadas como objetos indexados en una base de datos relacional, para mejorar la eficiencia de las consultas.

- **Título:** Improving the determination of Minimal Hitting Sets in Model-Based Diagnosis.

  Autores: María Teresa Gómez-López, Rafael Ceballos, Rafael M. Gasca y Carmelo Del Valle

  Publicado en: 16th International Workshop on Principles of Diagnosis, (DX'05) Monterey (California), páginas: 61-66, 2005

- **Título:** Algoritmo para la búsqueda de restricciones en una base de datos relacional orientada a objetos con restricciones polinómicas.

  Autores: María Teresa Gómez-López, Rafael M. Gasca y Carmelo Del Valle

  Publicado en: Revista CUORE. Vivat Academia, Octubre 2005, número 12, páginas: 3-16

- **Título:** ORCDB: Arquitectura para la extensión de la semántica de SQL en bases de datos restrictivas orientadas a objetos con restricciones polinómicas de igualdad.

  Autores: María Teresa Gómez-López, Rafael M. Gasca, Carmelo Del Valle y Victor Cejudo

  Publicado en: Jornadas de Ingeniería del Software y Bases de Datos, páginas 221-229, 2005.

  Índice de Aceptación: 31,7 %

- **Título:** Querying a Polynomial Constraint Object-Relational Database in Model-based Diagnosis.

  Autores: María Teresa Gómez-López, Rafael M. Gasca, Carmelo Del Valle y F. Fernando de la Rosa T.

  Publicado en: International Conference on Database and Expert Systems Applications (DEXA), Lecture Notes in Computer Science, Springer, volumen3588, páginas 848-857, 2005.

- **Título:** A model integration of DX and FDI techniques for automatic determination of minimal diagnosis.

  Autores: Rafael Ceballos, María Teresa Gómez-López, Rafael. M. Gasca y Carmelo Del Valle

  Publicado en: International Joint Conference on Artificial Intelligence (IJCAI'05), 2nd MONET Workshop on Model-Based Systems, Edimburgo, Scotland, Julio2005

[**2006**]**:** Una vez que la arquitectura está desarrollada, se completa la sintaxis y la semántica de la ampliación de SQL. Como caso de estudio se utiliza la diagnosis basada en modelos para la obtención de los hitting sets, junto a ejemplos económicos. También se plantea el primer trabajo de cómo las Bases de Datos de Restricciones pueden ser ampliadas para un tratamiento distribuido de la información.

- **Título:** Improving the determination of Minimal Hitting Sets in Model-Based Diagnosis using Constraint Databases.

  Autores: María Teresa Gómez-López, Rafael M. Gasca and Carmelo Del Valle

  Publicado en: 6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, SAFEPROCESS 2006

- **Título:** Ampliación de la sintaxis y la semántica de SQL para el tratamiento de datos tipo restricción.

  Autores: María Teresa Gómez-López y Rafael M. Gasca

  Publicado en: XI Jornadas de Ingeniería del Software y Bases de Datos, Sitges, España, páginas: 471-476, 2006

- **Título:** Distributed Model-Based Diagnosis using Object-Relational Constraint Databases.

  Autores: María Teresa Gómez-López, Rafael M. Gasca, Carmelo Del Valle y Sergio Pozo

  Publicado en: IEEE The 20th International Conference on Advanced Information Networking and Applications, Vienna, Austria, vol 2. páginas: 866-870,2006

- **Título:** Integración de técnicas basadas en modelos para la determinación de la diagnosis minima de un sistema.

  Autores: Rafael Ceballos, María Teresa Gómez-López, Rafael M. Gasca y Carmelo Del Valle

  Publicado en: Revista Iberoamericana de Inteligencia Artificial, Volumen31, páginas: 41-52, 2006

[**2007**]**:** La utilización de las Bases de Datos de Restricciones es beneficioso tanto desde el punto de vista del almacenamiento de datos, como desde el punto de vista de la construcción y resolución de problemas de satisfacción de restricciones. Con esta idea se han creado trabajos, tanto en el caso de la diagnosis basada en modelos utilizando la indexación de variables, como la resolución de problemas sobre restringidos utilizando las envolventes de las variables.

- **Título:** A compiled model for faults diagnosis based on different techniques.

  Autores: Rafael Ceballos, María Teresa Gómez-López, Rafael M. Gasca y Carmelo Del Valle

  Publicado en: AI Communications, vol. 20, páginas 6-17, 2007

  Índice de Impacto: 0.46

- **Título:** Developing a Labelled Object-Relational Constraint Database Architecture for Projection Operator.

  Autores: María Teresa Gómez-López, Rafael Ceballos, Rafael M. Gasca y Carmelo Del Valle

  Publicado en: En la segunda fase del proceso de revisión en Data & Knowledge Engineering. Ed. Elsevier

  Índice de Impacto: 1.36

- **Título:** NMUS: Structural Analysis for Improving the Derivation of All MUSes in Overconstrained Numeric CSPs

  Autores: Rafael M. Gasca y Carmelo Del Valle, María Teresa Gómez-López y Rafael Ceballos

  Publicado en: XII Conferencia de la Asociación Española Para la Inteligencia Artificial (CAEPIA), Lecture Notes in Artificial Intelligence, Springer, volumen4788, páginas 160-169, 2007

  Índice de Aceptación: 20,1

- **Título:** Bases de Datos de Restricciones para el tratamiento de CSP

  Autores: María Teresa Gómez-López, Rafael M. Gasca y C. Del Valle

  Publicado en: Workshop on Planning, Scheduling and Constraint Satisfaction, 2007

- **Título:** Supervised and Distributed Model-Based Diagnosis

  Autores: Diana Borrego, María Teresa Gómez-López y Rafael M. Gasca

  Publicado en: II Workshop on Industrial Applications of Distributed Intelligent Systems, 2007

9.4. Líneas Futuras de Investigación

Una vez definido el nuevo modelo de Bases de Datos de Restricciones, existen líneas claras para continuar con la investigación, ya que el objetivo siempre es tener todas las ventajas de las bases de datos relaciones clásicas, pero añadiendo la utilización de restricciones como tipo de dato. Con esta idea, se proponen las siguientes líneas futuras de investigación.

9.4.1. Incluir la Programación Lógica

Incluir capacidades lógicas, no exclusivamente numéricas en la Base de Datos de Restricciones. Esto significa añadir las cualidades de la programación lógica con restricciones, no sólo programación con restricciones numéricas como se tiene hasta el momento.

9.4.2. Restricciones de Integridad

Incluir restricciones de integridad en las Bases de Datos de Restricciones sobre atributos restricción o variable de restricción. Por ejemplo, que las soluciones de las restricciones de un atributo tienen que estar incluidas en las soluciones de otro atributo restricción.

9.4.3. Ampliar el tipo Restricción

Con respecto al tratamiento de restricciones, sería muy interesante definir el tipo *Restricción* utilizando más operadores como $\vee$, $\neg$, $\rightarrow$ o $\leftrightarrow$. Pese a que en operaciones como la unión y la diferencia se utiliza el operador de negación ($\neg$) en la obtención de nuevas restricciones, no se permite añadir nuevas restricciones a la base de datos con dicho operador. Si se incluyeran dichos operadores, la construcción de problemas de satisfacción de restricciones o de optimización con restricciones no conllevaría demasiada problemática, pero la eliminación simbólica de variables sería mucho más compleja.

9.4.4. Bases de Datos Distribuidas

Cuando la información de una Base de Datos de Restricciones no está centralizada, será necesario ampliar la arquitectura para el tratamiento de dicha información. Esto también significa crear los problemas de satisfacción de restricciones y modelos de eliminación simbólica de forma distribuida. Aunque ya se han publicado trabajos que amplían la arquitectura LORCDB para hacer distribuida, aún quedan muchos aspectos que analizar y mejorar.

9.4.5. Analizar más casos de estudio

Aunque en esta tesis se presentan tres casos de estudios (económico, diagnosis basada en modelos y sistemas de información geográfica), existen mucho otros como la planificación, optimización, y los sistemas CAD/CAM. Estos casos de estudio son especialmente complejos ya que pueden utilizar estructuras recursivas, por lo que presenta interés analizar cómo el uso de Bases de Datos de Restricciones podrían facilitar su desarrollo, o cómo se puede ampliar la arquitectura propuesta para dichos casos.

Apéndice A

Algoritmo para la Proyección Simbólica

En esta sección se presenta con detalle el algoritmo para la proyección simbólica sobre atributos variable de restricción, tanto para búsqueda horizontal como vertical. Para obtener el conjunto de restricciones relacionadas por variables (CRRV) se propone un algoritmo, mostrando las estructuras de datos necesarias, una traza del mismo, y las propiedades que cumple.

A.1. Algoritmo para la Obtención de los CRRV

Como se expuso en el capítulo 6, para la búsqueda de las restricciones relacionadas con los atributos variable de restricción, existen dos variantes, la proyección horizontal (figura A.1.a) y la proyección vertical (figura A.1.b). En función de qué tipo de proyección se realice, la búsqueda involucrará restricciones de la misma tupla (G_1, ..., G_n), o del mismo atributo restricción (G_1, ..., G_m).

Partiendo de la notación introducida en el capítulo 6:

- **Variables**($c_i, \ldots, c_j$)**.** Conjunto de las variables de las restricciones c_i, ..., c_j sin repetición.
- **Variables**(Q, $\{c_i, \ldots, c_j\}$)**.** Conjunto de variables sin repetición de las restricciones c_i, ..., c_j que aparecen en la consulta Q.

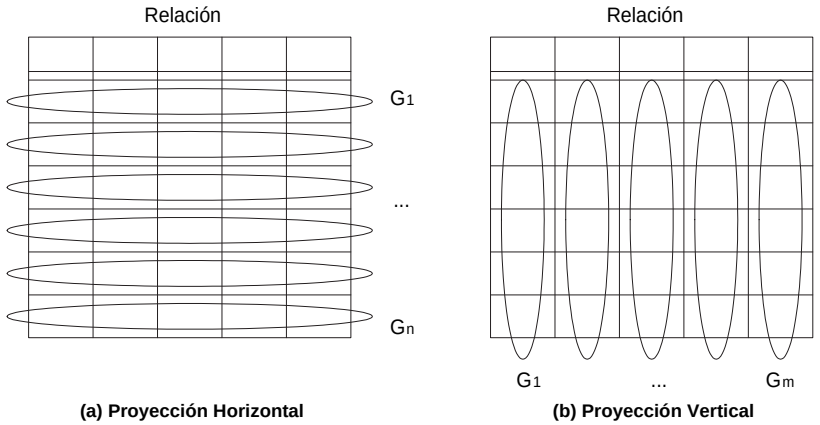

Figura A.1: Tipos de Proyección: Horizontal y Vertical

Y de la definición de CRRV para la proyección simbólica:

Conjunto de Restricciones Relacionadas por Variables (CRRV) para la Proyección Simbólica. G_i' será un Conjunto de Restricciones Relacionadas por Variables si:

$$
\begin{aligned}
&\text{Siendo } G_i\text{'} \equiv \{c_i, \ldots, c_j\} \mid (G_i \equiv \{c_1, \ldots, c_n\} \wedge G_i\text{'} \subseteq G_i) \\
&\forall\, c_k \in G_i\text{'} \\
&\quad \forall\, v \in \text{Variables}(c_k) \mid v \notin \text{Variables}(Q, c_k) \Rightarrow \\
&\qquad (v \in \text{Variables}(G_i\text{'} - \{c_k\})) \vee (\nexists\, c_l \in (G_i - \{c_k\}) \mid v \in \text{Variables}(c_l)) \\
&\wedge \\
&\text{Variables}(Q, \{c_i, \ldots, c_j\}) \neq \emptyset \\
&\wedge \\
&\nexists\, G\text{''} \subset \text{G}_i\text{'} \mid G\text{''} \text{ sea un CRRV}
\end{aligned}
$$

A.2. Estructuras de Datos

En este apéndice se describen los detalles del algoritmo para obtener los CRRV para cada uno de los G_i, y las variables de la proyección definidas en la consulta Q. La idea del algoritmo es agrupar las restricciones que tengan variables en común que no pertenezcan a la consulta. Antes de describir el algoritmo, cabe destacar los tipos de relación entre dos restricciones a nivel de CRRV. Dos restricciones tendrán una relación `AND` si son las únicas que tienen una variable que no pertenece a la consulta, lo que significa que si una de las restricciones forma parte de un CRRV, la otra restricción también pertenecerá al

CRRV. Mientras que dos restricciones en un CRRV tendrán una relación `OR` si tienen una variable no perteneciente a la proyección en común pero que también está en otras restricciones de G_i, lo que significa que dichas restricciones pueden pertenecer al mismo CRRV o no. Basado en esta idea se crea un grafo donde cada nodo estará formado por una restricción sola, o un conjunto de restricciones con una relación `AND` entre ellas dos a dos, lo que provocará que el grafo tenga menos nodos, y por lo tanto el recorrido de él sea más eficiente.

Cada uno de los nodos del grafo (*Nodo-Restricciones*) estará formado por una estructura con los siguientes atributos:

- *Conjunto Restricciones*: Conjunto de las restricciones que forman el nodo, que tienen una relación `AND` entre ellas. Es posible que este conjunto sólo tenga una restricción.
- *Conjunto VariablesResueltas*: Conjunto con las variables de las restricciones que pertenecen a la proyección, o que están en dos restricciones del conjunto de *Restricciones* derivado de que tengan una relación `AND`.
- *Conjunto VariablesNoResueltas*: Conjunto con las variables de las restricciones del conjunto *Restricciones*, que no pertenecen a las *VariablesResueltas*.

Entre dos nodos del grafo (*Nodo-Restricciones*) existirá una arista si comparten al menos una variable no resuelta (*VariablesNoResueltas*).

En la figura A.2 se muestra un ejemplo donde las restricciones $R1$ y $R2$ tienen una relación `AND`, ya que la variable k sólo aparece en esas dos restricciones, por lo que si una pertenece a un CRRV, la otra también tendrá que pertenecer. El resto de restricciones tienen una relación `OR`.

Una vez formado dicho grafo, el recorrido del grafo consiste en, partiendo de cada uno de los *Nodo-Restricciones* con variables pertenecientes a la consulta Q, obtener todos los conjuntos de *Nodo-Restricciones* que cumplan la definición de CRRV. Para ello se utiliza una estructura llamada *Estructura-CRRV*, a la que se le irán añadiendo las distintas restricciones a lo largo del recorrido del grafo tal como se explicará en el algoritmo 1. La *Estructura-CRRV* tiene los atributos:

- *Conjunto Restricciones*: Conjunto de los identificadores de las restricciones que hasta el momento forman el CRRV.

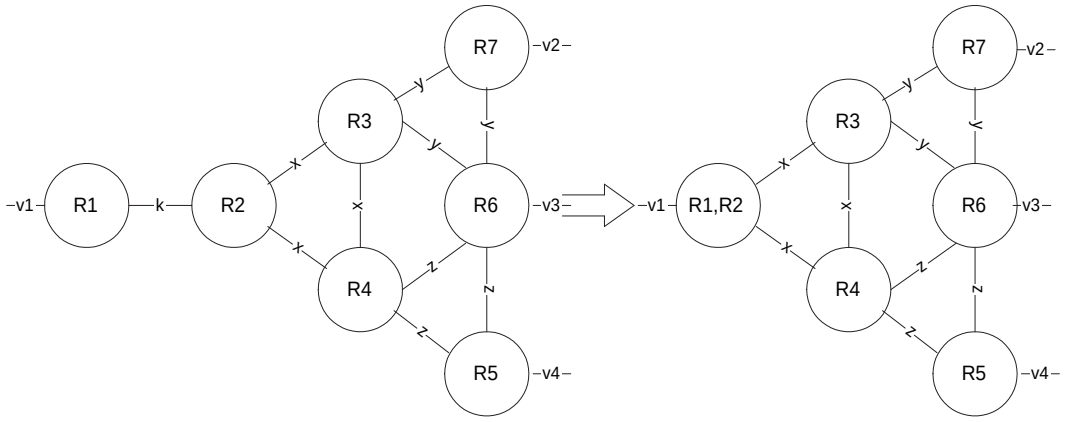

Figura A.2: Ejemplo transformación grafo *Nodo-Restricciones*

- *Conjunto VariablesAnalizadas*: Conjunto con las variables que ya se han catalogado dentro de los tres tipos definidos para los CRRV: (a) variables que aparecen en la proyección; (b) variables que están al menos en dos restricciones del conjunto *Restricciones* de la misma *Estructura-CRRV*; (c) variables que sólo están en una restricción de las pertenecientes al grupo de restricciones G_i sobre la que se está haciendo la búsqueda, y no pertenecen a las variables de la proyección.

- *Conjunto VariablesNoAnalizadas*: Conjunto con las variables de las restricciones del conjunto *Restricciones* que no pertenecen aún a *VariablesAnalizadas*.

- *Conjunto VariablesNoResueltas*: Conjunto con las variables no pertenecientes a la proyección, que sólo aparecen en una restricción de las pertenecientes al grupo de restricciones G_i sobre el que se está realizando el recorrido.

- *Conjunto RestCiclo*: Este conjunto de restricciones evita ciclos cuando existen relaciones `OR` entre restricciones, y se pueden obtener CRRV repetidos resolviendo la variable con dos o más restricciones diferentes. Un ejemplo basado en la figura A.2 es obtener dos CRRV por la relación `OR` existente entre ($R_1 - R_2$, R_3) y ($R_1 - R_2$, R_4), por lo que realizando el recorrido por ambas ramas se obtendrán dos CRRV: $\{R_1 - R_2,\ R_3,\ R_6,\ R_4\}$ y $\{R_1 - R_2,\ R_4,\ R_6,\ R_3\}$. Estos CRRV son los mismos, ya que el orden de las restricciones no es determinante. Para evitar estas duplicaciones, al comenzar un recorrido `OR` se añadirá dicha restricción a *RestCiclo*, para que no pueda participar en otro recorrido. Un ejemplo de la traza completa se mostrará más adelante.

La idea del algoritmo consiste en, inicializando una *Estructura-CRRV* con un *Nodo-Restricciones* con variables de la proyección, ir añadiendo a la *Estructura-CRRV* nuevas restricciones que tenga *VariablesNoResueltas* en común con las *VariablesNoAnalizadas* de la *Estructura-CRRV* que se esté tratando. Cuando se añade a una *Estructura-CRRV* llamada por ejemplo `crrv`, un *Nodo-Restricciones* `rr` (`crrv.añadir(rr)`) por existir una variable que pertenezca a las *variablesNoAnalizadas* de `crrv`, y a las *variablesNoResueltas* de `rr`, se hará:

1. Añadir las restricciones de `rr` a `crrv`:

   `crrv.restricciones.añadir(rr.restricciones)`

2. Poner como analizadas aquellas variables no analizadas de `crrv` que aparezcan como no resueltas en `rr`, ya que son variables que no pertenecen a la proyección que están al menos en dos restricciones, una de las restricciones ya incluidas en `crrv` y otra de `rr`:

   `crrv.variablesAnalizadas.añadir(crrv.variablesNoAnalizadas` $\cap$ `rr.variablesNoResueltas)`

3. Añadir a las variables no analizadas de `crrv`, las no resueltas de `rr` que no estén ya en las analizadas:

   `crrv.variablesNoAnalizadas.añadir(rr.VariablesNoResueltas`$-$`crrv.VariablesAnalizadas)`

Cuando una *Estructura-CRRV* no tiene ninguna variable en las *VariablesNoAnalizadas* y el conjunto es mínimo, se almacenará en una lista llamada *ListaCRRV*. Para conocer si un CRRV obtenido mediante el algoritmo es mínimo, se utilizará la función *esMínimo(CRRV)*, la cual comprueba si para cada una de las restricciones del CRRV de entrada, existe al menos una variable que sólo aparece en dos restricciones del conjunto. Esto significará que dicha restricción es necesaria para forma el CRRV, y que si ella no formará parte del conjunto, dejaría de ser un CRRV. Con el algoritmo propuesto, esta comprobación sólo será necesaria si existen restricciones que tengan más de una variable en común. Como se muestra en la figura A.3, donde el conjunto de restricciones mostrado no forma un CRRV, ya que por ejemplo todas las variables de R_2 (x, y) aparecen en otras dos restricciones.

Si el conjunto *VariablesNoResueltas* está vacío, el CRRV será un CCRRV (Conjunto Completo de Restricciones Relacionadas por Variables).

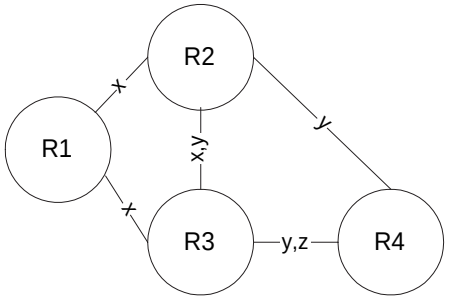

Figura A.3: Ejemplo de CRRV no mínimo

A.3. Descripción del Algoritmo

Para cada $G_i \in G$, independientemente de que sea $G = (G_1, \ldots, G_n)$ o $G = (G_1, \ldots, G_m)$, se construirá un grafo como se ha explicado. El procedimiento que obtiene todos los CRRV es el mostrado en el algoritmo 1.

Algoritmo 1: Obtención de CRRV de la Proyección Simbólica para G

Entrada: Q
Salida: *Lista* listaCRRV
Lista G_i
Lista listaCRRV := Lista()
Lista G := ObtenerGrafosRestricciones(Q)
Para cada $G_i \in G$ **hacer**
    Lista restProyección := ObtenerRestriccionesProyección(G_i)
    ObtenerCRRV(G_i, restProyección, listaCRRV)

Dependiendo de la consulta Q, se obtendrán los conjuntos de restricciones $G_1, \ldots, G_n$, donde cada G_i es una lista de *Nodo-Restricciones*, obtenido mediante la función *ObtenerGrafosRestricciones*. Para cada G_i, será necesario conocer los *Nodo-Restricciones* que contienen las variables que participan en la proyección, las cuales se obtendrán con la función *ObtenerRestriccionesProyección*. La función *ObtenerCRRV* obtiene, partiendo de las restricciones de *restProyección*, todos los CRRV para cada G_i en la variable *listaCRRV*.

ObtenerCRRV(Lista G_i, Lista restProyección, Lista listaCRRV): Este procedimiento realizará un recorrido de las restricciones de G_i, comenzando por las restricciones pertenecientes a la lista *restProyección*. Cuando se comienza la búsqueda por un *Nodo-Restricciones*, el algoritmo obtiene todos los CRRV con dicha restricción. Para evitar que

al comenzar la búsqueda por otro *Nodo-Restricciones* con variables de la proyección, se vuelvan a encontrar dichos CRRV, dichas restricciones se almacenarán en*restriccionesVisitadas*. Para cada una de estas *Estructuras-CRRV*, se analizarán sus *VariablesNoAnalizadas* para unir aquellas restricciones que tengan variables en común que no pertenezcan a las variables de la proyección. La función *ObtenerCRRV* es la mostrada en el algoritmo 2.

Algoritmo 2: Función ObtenerCRRV

Entrada: *Lista* G_i, *Lista* restProyección, *Lista* listaCRRV
Salida: *Lista* listaCRRV
Estructura-CRRV crrv
Lista restricionesVisitadas
Para cada $r \in$ *restProyección* $\mid r \notin$ *restriccionesVisitadas* **hacer**
    restriccionesVisitadas.añadir(r)
    crrv := Estructura-CRRV();
    crrv.restricciones.añadir(r)
    crrv.variablesAnalizadas.añadir(r.VariablesResueltas)
    crrv.variablesNoAnalizadas.añadir(r.VariablesNoResueltas)
    crrv.RestCiclo := Lista()
    crrv.VariablesNoResueltas := Lista()
    recorridoGrafoCRRV(crrv, listaCRRV, restriccionesVisitadas)

El procedimiento *RecorridoGrafoCRRV* es recursivo (mostrado en el algoritmo 3), y se basa en ir encontrando restricciones que tengan en común variables no analizadas de la *Estructura-CRRV* que se pasa como parámetro de entrada. Cuando no quedan variables por analizar, se añade a *listaCRRV*. Para evitar recorridos duplicados, se utiliza el conjunto *restriccionesVisitadas*, que también será parámetro de entrada. Volviendo a la figura A.2, un ejemplo de los recorridos duplicados que esta lista de *restriccionesVisitadas* evitados puede ser: al comenzar la búsqueda por el nodo R_1-R_2, se creará un CRRV formado por $\{R_1$-R_2, R_4, $R_5\}$, si R_1-R_2 no se añadiera a la lista de *restriccionesVisitadas*, al comenzar la búsqueda por R_5 si fuera una restricción con variables de la proyección, se volvería a construir el mismo CRRV.

Algoritmo 3: Procedimiento recorridoGrafoCRRV

```
Entrada: Estructura-CRRV crrv, Lista listaCRRV, Lista restriccionesVisitadas
Salida: Lista listaCRRV
si crrv.variablesNoAnalizadas.esVacía() entonces
    si esMínimo(crrv) entonces
        listaCRRV.añadir(crrv)
sino
    Para v ∈ CRRV.VariablesNoAnalizadas
    crrv.VariablesNoAnalizadas.borrar(v)
    crrv.VariablesAnalizadas.añadir(v)
    Lista restRelacionadas := ObtenerRestrYVariables(crrv, v,
        G_i, restriccionesVisitadas)
    //obtiene los Nodo-Restricciones de G_i que tengan a v, pero no estén ya
    //incluidos en crrv.RestCiclo ni restriccionesVisitadas
    si restRelacionadas.esVacía() entonces
        si v ∉ Variables(G_i − crrv.restricciones) entonces
            crrv.varNoResueltas.añadir(v)
    sino
        Para cada rr ∈ restRelacionadas hacer
            crrv.restriccionesCiclo.añadir(rr)
            //evita obtener el mismo CRRV pero recorriendo las
            //restricciones en distinto orden
            Estructura-CRRV copiaCrrv := crrv.clonar()
            copiaCrrv.añadir(rr)
            recorridoGrafoCRRV(copiaCrrv, listaCRRV, restriccionesVisitadas)
```

A.4. Ejemplo de Traza

Para entender mejor el algoritmo, se presenta la traza para el ejemplo de la figura A.2 donde las variables de la proyección son v_1, v_2, v_3:

- Se inicializa crrv con algún Nodo-Restricciones que contenga variables de la proyección, crrv:{Restricciones: R_1-R_2; VariablesAnalizadas: {k}; VariablesNoAnalizadas:

{x}; VariablesNoResueltas: {}; restCiclo: {}}, RestriccionesVisitadas:R_1-R_2.

Para x: $\{R_3, R_4\}$:= ObtenerRestYVariables(..., x, ...)

- crrv:{Restricciones: R_1-R_2, R_3; VariablesAnalizadas: {k, x}; VariablesNoAnalizadas: {y}; VariablesNoResueltas: {}; restCiclo: {R_3}}, RestriccionesVisitadas: R_1-R_2.

  Para y: $\{R_6, R_7\}$ = ObtenerRestYVariables(..., y, ...)

  - crrv:{Restricciones: R_1-R_2, R_3, R_6; VariablesAnalizadas: {k, x, y}; VariablesNoAnalizadas: {z}; VariablesNoResueltas: {}; restCiclo: {R_3, R_6}}, RestriccionesVisitadas: R_1-R_2.

    Para y: $\{R_4, R_5\}$ = ObtenerRestYVariables(..., z, ...)

    - crrv:{Restricciones: R_1-R_2, R_3, R_6, R_4; VariablesAnalizadas: {k, x, y, z}; VariablesNoAnalizadas: {};VariablesNoResueltas: {}; restCiclo: {R_3, R_6, R_4}}, RestriccionesVisitadas: R_1-R_2.

      Como no quedan variables por analizar, **{R_1-R_2, R_3, R_6, R_4} es un CRRV**, y como no tiene variablesNoResueltas también será un CCRRV.

    - crrv:{Restricciones: R_1-R_2, R_3, R_6, R_5; VariablesAnalizadas: {k, x, y, z}; VariablesNoAnalizadas: {};VariablesNoResueltas: {}; restCiclo: {R_3, R_6, R_4, R_5}}, RestriccionesVisitadas: R_1-R_2.

      Como no quedan variables por analizar, **{R_1-R_2, R_3, R_6, R_5} es un CRRV**, y como no tiene variablesNoResueltas también será un CCRRV.

  - crrv:{Restricciones: R_1-R_2, R_3, R_7; VariablesAnalizadas: {k, x, y}; VariablesNoAnalizadas: {}; VariablesNoResueltas: {}; restCiclo: R_3, R_6, R_7}, RestriccionesVisitadas: R_1-R_2.

    Como no quedan variables por analizar, **{R_1-R_2, R_3, R_7} es un CRRV**, y como no tiene variablesNoResueltas también será un CCRRV.

- crrv:{Restricciones: R_1-R_2, R_4; VariablesAnalizadas: {k, x}; VariablesNoAnalizadas: {z}; VariablesNoResueltas: {}; restCiclo: {R_3, R_4}}, RestriccionesVisitadas: R_1-R_2.

  Para z: $\{R_5, R_6\}$ = ObtenerRestYVariables(..., z, ...)

- crrv:{Restricciones: R_1-R_2, R_4, R_5; VariablesAnalizadas: {k, x, z}; VariablesNoAnalizadas: {};VariablesNoResueltas: {v4}; restCiclo: {R_3, R_4}}, RestriccionesVisitadas: R_1-R_2.
  Como no quedan variables por analizar, {R_1-R_2, R_4, R_5} es un CRRV, pero como tiene variables no resueltas, no es un CCRRV.
- crrv:{Restricciones: R_1-R_2, R_4, R_6; VariablesAnalizadas: {k, x, z}; VarNoAnalizadas: {y};VariablesNoResueltas: {v4}; restCiclo: {R_3, R_4}}, RestriccionesVisitadas: R_1-R_2.
  Para y: {R_7} = ObtenerRestYVariables(. . ., y, . . .). No devuelve R_3 porque está en restCiclos, y así se evita que la listaCRRV contenga CRRV duplicados.
  - crrv:{Restricciones: R_1-R_2, R_4, R_6, R_7; VariablesAnalizadas: {k, x, z, y}; VariablesNoAnalizadas: {};VariablesNoResueltas: {v4}; restCiclo: {R_3, R_4}}, RestriccionesVisitadas: R_1-R_2.
    Como no quedan variables por analizar, {R_1-R_2, R_4, R_6, R_7} es un CRRV, pero como tiene variables no resueltas, no es un CCRRV.

- crrv:{Restricciones: R_5; VariablesAnalizadas: {}; VarNoAnalizadas: {z}; VariablesNoResueltas: {}; restCiclo: {}}, RestriccionesVisitadas:R_1-R_2, R_5.

  . . .

  Ningún CRRV obtenido desde la inicialización conR_5 incluirá al Nodo-Restricciones R_1-R_2, ya que todos los CRRV con R_1-R_2 se han obtenido con la primera inicialización de la traza, y se ha incluido en *RestriccionesVisitadas*.

Un ejemplo de cómo se va construyendo la *Estructura-CRRV* en los distintos pasos de la traza presentada, se muestra en la figura A.4 en forma de árbol. Los diferentes CRRV serán todos los posibles caminos desde la raíz hasta las hojas de cada uno de dichos árboles.

A.5. Propiedades del Algoritmo

Para analizar las distintas propiedades del algoritmo, se comenzará estudiando cómo cambia el estado de la *Estructura-CRRV*, la *listaCRRV* y *restriccionesVisitadas*, que son los parámetros de entrada del algoritmo recursivo *recorridoGrafoCRRV*.

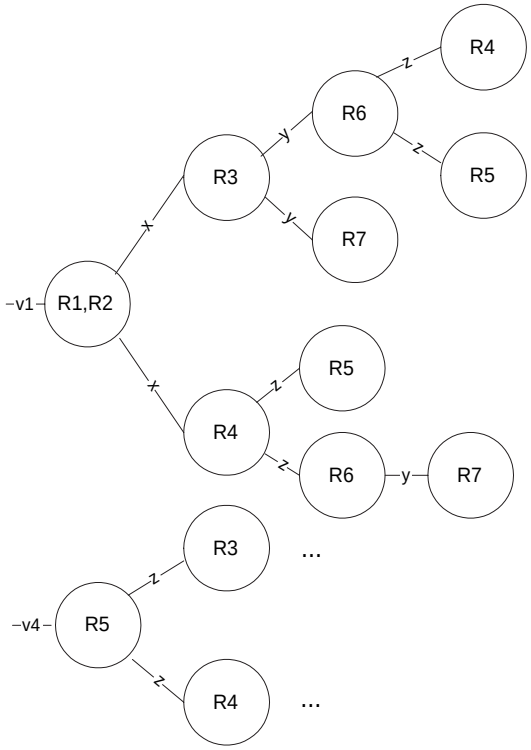

Figura A.4: Ejemplo de búsqueda de restricciones relacionadas por variables

A.5.1. Transición entre estados de la Estructura-CRRV

Antes de comenzar la descripción de la complejidad del algoritmo y las distintas propiedades que tiene, es necesario analizar cómo cambian los distintos parámetros de entrada del método recursivo *recorridoGrafoCRRV* (algoritmo 3) para obtener todos los CRRV: Estructura-CRRV, listaCRRV y restriccionesVisitas:

Estructura-CRRV (crrv):
  restricciones $r_1, \ldots, r_n$
  VariablesAnalizadas $vA_1, \ldots, vA_m$
  VariablesNoAnalizadas $vNA_1, vNA_2, \ldots, vNA_k$
  VariablesNoResueltas $vNR_1, \ldots, vNR_p$
  restriccionesCiclo $rC_1, \ldots, rC_q$
listaCRRV $lcrrv_1, \ldots, lcrrv_r$
restriccionesVisitadas $rV_1, \ldots, rV_s$

Los estados a donde puede cambiar dependerán de las variables no analizadas, de lo que se derivan cuatro tipos de transiciones:

1. Si $\{vNA_1, vNA_2, \ldots, vNA_k\} = \emptyset$ y el conjunto de restricciones es mínimo, la variable crrv formará un CRRV y será añadido a listaCRRV:

   listaCRRV $lcrrv_1, \ldots, lcrrv_r$, **crrv**

   El recorrido continuará con otro CRRV derivado de la utilización de la recursividad.

2. Si vNA_1 sólo está en una restricción, ya añadida a crrv, lo que implica que (vNA_1 $\notin$Variables(G_i−crrv.restricciones)), se añadirá a las variables no resueltas:

   Estructura-CRRV (crrv):
   restricciones $r_1, \ldots, r_n$
   VariablesAnalizadas $vA_1, \ldots, vA_m$
   VariablesNoAnalizadas $vNA_2, \ldots, vNA_k$
   VariablesNoResueltas $vNR_1, \ldots, vNR_p$, $\mathbf{vNA}_1$
   restriccionesCiclo $rC_1, \ldots, rC_q$
   listaCRRV $lcrrv_1, \ldots, lcrrv_r$
   restriccionesVisitadas $rV_1, \ldots, rV_s$

3. Si todas las restricciones que tienen a la variable vNA_1, son aquellas ya incluidas en el crrv, en las restriccionesVisitadas, o en las restriccionesCiclo ($\bigcup_{r \in G_i}$ | Variables(r) $\cap$ $vNA_1 \neq \emptyset$) − (crrv.restricciones $\cup$ restriccionesCiclo $\cup$ restriccionesVisitadas) = $\emptyset$), significará que el crrv no formará parte de listaCRRV.

   listaCRRV $lcrrv_1 \ldots lcrrv_r$
   restriccionesVisitadas $rV_1 \ldots rV_s$

   El recorrido continuará con otro CRRV derivado de la utilización de la recursividad.

4. En el caso de que la variable no analizada vNA_1 esté en otras restricciones que no impliquen CRRV duplicados, para ($\bigcup_{r \in G_i}$ | Variables(r)$\cap$ $vNA_1 \neq \emptyset$) − (crrv.restricciones $\cup$ restriccionesCiclo $\cup$ restriccionesVisitas) = $\{rr_1 \ldots rr_t\}$), donde $\{rr_1 \ldots rr_t\} \neq \emptyset$, se construirán t CRRV, con las distintas restricciones $\{rr_1 \ldots rr_t\}$, resueltos de forma recursiva con el procedimiento recorridoGrafoCRRV.

   Estructura-CRRV (crrv):
   restricciones $r_1, \ldots, r_n$, $\mathbf{rr}_1$
   VariablesAnalizadas $\{vA_1, \ldots, vA_m\} \cup$

$\{rr_1.VariablesNoResueltas \cap \{vNA_1, \ldots, vNA_k\}\}$
VariablesNoAnalizadas $(\{vNA_1, \ldots, vNA_k\} \cup$
$rr_1.VariablesNoResueltas)-$VariablesAnalizadas
VariablesNoResueltas $vNR_1, \ldots, vNR_p$
restriccionesCiclo $rC_1, \ldots, rC_q$, $\mathbf{rr}_1$
listaCRRV $lcrrv_1, \ldots, lcrrv_r$
restriccionesVisitadas $rV_1, \ldots, rV_s$

...

Estructura-CRRV (crrv):
restricciones $r_1, \ldots, r_n$, $\mathbf{rr}_t$
VariablesAnalizadas $\{vA_1, \ldots, vA_m\} \cup$
$\{rrt.VaraiblesNoAnalizadas \cap \{vNA_1, \ldots, vNA_k\}\}$
VariablesNoAnalizadas $(\{vNA_1, \ldots, vNA_k\} \cup$
$rr_t.VariablesNoResueltas)-$VariablesAnalizadas
VariablesNoResueltas $vNR_1, \ldots, vNR_p$
restriccionesCiclo $rC_1, \ldots, rC_q$, $\mathbf{rr}_1$, ..., $\mathbf{rr}_t$
listaCRRV $lcrrv_1, \ldots, lcrrv_r$
restriccionesVisitadas $rV_1, \ldots, rV_s$

A.5.2. Complejidad

Para cada Nodo-Restricciones con alguna variable de la consulta, se hace un recorrido del grafo en función de las variables no analizadas que tengan los *Nodo-Restricciones*. Lo que significa que los *Nodo-Restricciones* de cada G_i se recorrerán como mucho $m * (n^2)$, donde m es el número de *Nodo-Restricciones* con variables de la proyección, y n el número de *Nodo-Restricciones* del problema. Extendido para $G \equiv \{G_1, \ldots, G_k\}$ será: $m_1 * (n_1^2) + \ldots + m_i * (n_i^2) + \ldots + m_k * (n_k^2)$, donde m_i será el número de Nodo-Restricciones con variables de la consulta en G_i, y n_i el número de Nodos-Restricciones en G_i.

A.5.3. Completitud

En esta propiedad será necesario demostrar que: *partiendo de un conjunto de grafos* $G = \{G_1, \ldots, G_n\}$, *el algoritmo obtiene todos los CRRV para dicho conjunta*

El algoritmo analiza cada uno de los $\{G_1, \ldots, G_n\}$ en función de la consulta. Como para cada G_i se comienza la búsqueda desde cada Nodo-Restricciones con una variable de la proyección, se obtendrán todos los CRRV para unG_i. Lo que queda por analizar es que inicializando una *Estructura-CRRV* con un Nodo-Restricciones (r_1) que contenga una variable de la consulta, se obtendrán todos los crrv que contengan dicho Nodo-Restricciones.

Si las restricciones $\{r_1, \ldots, r_{n-1}, r_n\}$ forman un CRRV significa que la estructura con las restricciones $\{r_1, \ldots, r_{n-1}\}$ no será un crrv, y tendrá al menos una variable no analizada en común con r_n. Esto significa que existirá una transición tipo 4 entre el crrv $\{r_1, \ldots, r_{n-1}\}$ y $\{r_1, \ldots, r_{n-1}, r_n\}$ tal como se explicó en la sección anterior. Esta idea se puede extender a que si las restricciones $\{r_1, \ldots, r_{n-1}\}$ forman parte de una *Estructura-CRRV*, significa que existe una variable no analizada en $\{r_1, \ldots, r_{n-2}\}$ en común con la restricción r_{n-1} ... Hasta que finalmente la *Estructura-CRRV* contenga tan sólo a $\{r_1\}$, que es un estado inicial si $\{r_1\}$ contienen variables de la proyección.

Que no se obtengan todos los CRRV con el algoritmo, significa que existe un CRRV que no pertenece a la *listaCRRV* ($\exists crrv \notin \{lcrrv_1, \ldots, lcrrv_r\}$), donde crrv.restricciones $= \{r_1, \ldots, r_n\}$. Si crrv es un CRRV, por la definición tiene que $\exists r \in \{r_1, \ldots, r_n\}$ que pertenezca a la proyección (Variables(Q,$\{c_i$, ..., $c_j\}$) $\neq \emptyset$). Como se obtienen todos los crrv con una restricción con variables de la proyección, no es posible que exista un CRRV que no pertenezca a *listaCRRV*.

A.5.4. Corrección

Para demostrar que el algoritmo es correcto será necesario demostrar que: *todas las Estructuras-CRRV añadidas a listaCRRV son CRRV, lo que también implica que sean mínimos y existan CRRV duplicados*.

Todos los elementos de la listaCRRV son CRRV:

Derivado de los cuatro tipos de transiciones de estados presentados, se puede demostrar que todas los elementos de ListaCRRV forman un conjunto de restricciones $\{c_i, \ldots, c_j\}$ donde:

$\forall\ c_k \in \{c_i, \ldots, c_j\}$
  $\forall\ v \in$ Variables(c_k) | $v \notin$ Variables(Q, c_k) $\Rightarrow$
    ($v \in$ Variables(G$_i$'$-\{c_k\}$)) $\vee$ ($\nexists\ c_l \in \{$G$_i - c_k\}$ | $v \in$ Variables(c_l))

$\wedge$
Variables(Q,$\{c_i, \ldots, c_j\}$) $\neq \emptyset$

Una Estructura-CRRV se almacenará en la listaCRRV si no le quedan variablesNoAnalizadas (Transición 1). Para eliminar las variables de la lista de VariablesNoAnalizadas se pueden dar dos casos: que sea una variable no resuelta (Transición 2), por lo que cumple la condición de la definición $\nexists\ c_l \in \{\mathrm{G}_i{-}c_k\} \mid v \in \mathrm{Variables}(c_l)$); o que esté en otra restricción (Transición 4), por lo que cumple la condición de la definición ($v \in$ Variables(G$_i$'$-\{c_k\}$). Si no existe ninguna restricción que no esté incluida ya en *restriccionesCiclo* o *restriccionesVisitadas*, el crrv no formará parte de la solución (Transición 3).

La cláusula de la definición relativa a las variables de la proyección (Variables(Q,$\{c_i$, $\ldots$, $c_j\}$) $\neq \emptyset$), también se cumple porque siempre la búsqueda de CRRV se comienza inicializando una Estructura-CRRV con restricciones con variables de la proyección.

Los CRRV obtenidos son mínimos:

Si una Estructura-CRRV almacenada en la listaCRRV no fuera mínima, significaría que para un CRRV de salida del algoritmo con las restricciones $\{r_1, \ldots, r_{i-1}, r_i, r_{i+1}, \ldots, r_n\}$:

$\exists r_i \in \{r_1, \ldots, r_{i-1}, r_i, r_{i+1}, \ldots, r_n\} \mid \{r_1, \ldots, r_{i-1}, r_{i+1}, \ldots, r_n\}$ es un CRRV.

Si $\{r_1, \ldots, r_{i-1}, r_{i+1}, \ldots, r_n\}$ cumple la definición de CRRV, significa que todas las variables que no sean de la proyección y que estén en más de una restricción de G_i, están al menos en dos restricciones del CRRV. Por lo que el conjunto de restricciones $\{r_1, \ldots, r_{i-1}, r_i, r_{i+1}, \ldots, r_n\}$ no sería un CRRV ya que la función *esMínimo(CRRV)* habría detectado que no todas las restricciones son necesarias para formas el CRRV.

No existen CRRV duplicados en la lista listaCRRV

Gracias a utilizar las estructuras auxiliares *restriccionesVisitadas* y *restCiclo* no pueden existir dos CRRV duplicados en la *listaCRRV*. La lista *restriccionesVisitas* incluye a las restricciones con variables de la proyección para las cuales ya se han encontrado todos los CRRV donde está involucrada.

Para que dos CRRV formados por las mismas restricciones, pero añadidas a la *Estructura-CRRV* en diferente orden ($\{r_1, \ldots, r_{i-1}, r_i, \ldots, r_j, \ldots, r_n\}$, $\{r_1, \ldots, r_{i-1}, r_j, \ldots, r_i, \ldots, r_n\}$), formarán parte de la *listaCRRV* de salida, tendría que

ocurrir: r_i y r_j tienen una relación `OR` con r_{i-1}, por lo que por la Transición 4, se incluiría r_i en el atributo *restCiclo* de los siguientes recorridos y r_i no podrá formar parte de la *Estructura-CRRV* con las restricciones $\{r_1, \ldots, r_{i-1}, r_j\}$.

Apéndice B

Ejemplos de Tablas de Restricciones para el Análisis de Eficiencia

En este apéndice se muestran las restricciones usadas para todos los ejemplos del capítulo 8. Dicho capítulo utiliza el ejemplo de la diagnosis basada en modelos y los sistemas de información geográfica. También se presenta la aplicación desarrollada para facilitar la captura de restricciones necesarias para realizar las consultas sobre los sistemas de información geográfica.

B.1. Ejemplo para la diagnosis basada en modelos

El caso de estudio de la diagnosis basada en modelos utiliza un conjunto de componentes distribuidos en cuatro subsistemas. Las restricciones concretas se muestran en las figuras: B.1, B.2, B.3 y B.4.

B.2. Aplicación para la captura de restricciones sobre SIG

Lo primero para poder hacer consultas sobre un conjunto de restricciones de una Base de Datos de Restricciones, será obtener y almacenar dichas restricciones. En esta tesis se ha desarrollado e implementado una aplicación que ayuda a la captura de puntos de

NOMBRE	COMPORTAMIENTO
A1	a+b<=e AND e-1<=a+b
A2	c+d<=f AND f-1<=c+d
M1	e+f<=g AND g-1<=e+f
M3	g*d<=u AND u-1<=g*d
A4	m+n<=h AND h-1<=m+n
A5	o+p<=i AND i-1<=o+p
M2	h*i<=j AND j-1<=h*i
A3	j+g<=k AND k-1<=j+g
A6	m+j<=q AND q-1<=m+j
A7	k+u<=s AND s-1<=k+u
A9	q+k<=r AND r-1<=q+k
A8	r+s<=t AND t-1<=r+s

Figura B.1: Contenido de la tabla *Componentes* para el subsistema 1

NOMBRE	COMPORTAMIENTO
A14	e1+f1<=o1
A15	g1+h1<=p1
A16	i1+j1<=q1
R2	k1-l1<=r1
A22	o1+p1<=t1
M7	q1*r1<=u1
R4	w1-t1<=o2
R5	t1-p1<=r2
A12	q1+u1<=s2
A13	r1+l1<=t2

Figura B.2: Contenido de la tabla *Componentes* para el subsistema 2

un mapa con el objetivo de obtener las restricciones y almacenarlas en la Base de Datos de Restricciones, cuya interfaz se muestra en la figura B.5. Dicha aplicación utiliza el método de Lagrange para la aproximación mediante polinomios a un conjunto de puntos, dicho método utiliza la interpolación mediante mínimos cuadrados [54][7]. La utilización de polinomios también conlleva una aproximación de la realidad, pero que se acerca más a la información real que si sólo se utilizan rectas, a la vez que necesita menos restricciones para aproximar una superficie curva.

NOMBRE	COMPORTAMIENTO
A23	a1+b1=m1
R1	c1-d1=n1
M6	m1*n1=s1
A10	a1+s1=x1
A11	y1+n1=z1
M4	x1*z1=a2

Figura B.3: Contenido de la tabla *Componentes* para el subsistema 3

NOMBRE	COMPORTAMIENTO
A17	o2+r2=b2
M10	s2*t2=c2
A18	a2+b2=h2
A19	b2+i2=r2
R3	r2-t1=i2
A20	s2+c2=k2
A21	c2+t2=l2
M9	h2*i2=g2
M8	i2*k2=p2
M5	l2*t=q2

Figura B.4: Contenido de la tabla *Componentes* para el subsistema 4

B.2.1. Pasos para la obtención de restricciones

1. Marcando algunos de los puntos del contorno de la figura que se quiera obtener su restricción.

2. Incrementar el grado del polinomio hasta que se acerque lo más posible a lo que se quiere representar.

3. Determinar si es un polinomio de igualdad (Equals), si el polinomio es el límite inferior (Up) o es el límite superior (Down) de la superficie marcada.

4. Determinar el nombre de dicha figura en la zona de nombre y salvar (Save).

5. Si la zona a representar tiene que estar representada por varias restricciones, con una relación conjuntiva, se continúan añadiendo restricciones manteniendo el nombre. En caso contrario, se utiliza el botón *Clear* para comenzar con una nueva restricción.

La aplicación también muestra el coeficiente de determinación que se está utilizando, y el valor residual de la suma de los cuadrados de la aproximación. Cuando una restricción se almacena en la base de datos también se almacenan las envolventes para cada una de las variables, que son directamente obtenidos de los puntos añadidos por el usuario.

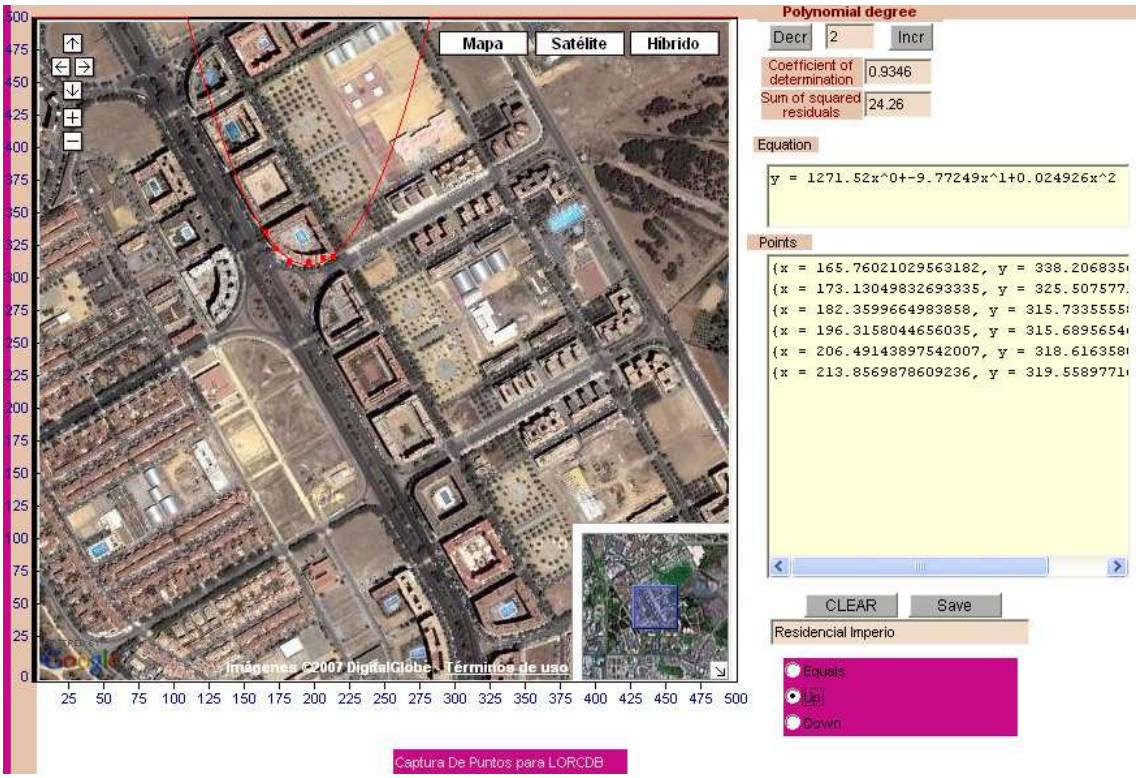

Figura B.5: Aplicación para el almacenamiento de Restricciones

B.3. Contenido de las tablas de Sistemas de Información Geográfica

Para realizar las distintas pruebas sobre las operaciones de Selección, Unión y Diferencia, se han creado cuatro tablas con atributos clásicos y atributos restricción. Dichas tablas son *Calles* (figura B.6), *Residenciales* (figura B.7), *Censos* (figura B.9) y *Obras*

(figura B.10). En la figura B.8 se muestran las envolventes de las variables x e y para las distintas restricciones de la tabla *Residenciales*, información que se almacena en la tabla *Restricciones/Variables*. En realidad, no se utiliza el nombre de la variable, sino su identificador almacenado en la tabla *Variables*, pero en esta figura se utiliza el nombre de las variables para facilitar su entendimiento.

NOMBRE	SITUACION	TIPO
Avd de las Ciencias	y=593.988-1.8736*x	avd
Flor de Papel	y=688.373-1.71818*x	calle
Flor de Salvia	y=0.22116+0.59277*x	calle
Flor de Pascua	y=0.59472*x-39.22334	calle
Flor de Albahaca	y=718.871-1.81785*x	calle
Flor de Tomillo	y=40.3822+0.57528*x	calle
Flor de Adelfa	y=169.234+0.52905*x	calle
Flor de Gitanilla	y=185.743+0.59616*x	calle
Flor de Retama	y=988.143-1.84247*x	calle
Avd del Deporte	y=1106.33-1.87302*x	avd
de Fernanda Calado	y=727.008-1.8387*x	calle
de Doctor Cortella	y=234.374+0.5431*x	calle
Doctor Madrazo	y=356.684+0.6005*x	calle
Caminos	y=336.045+3.52716*x+0.00018223*x*x*x-0.05533*x*x	calle
Argos	y=482.821-1.77833*x	calle
Casiodoro	y=492.882-1.85667*x	calle
Horizontes	y=424.6-1.75632*x	calle
Telémaco	y=414.046-1.76143*x	calle
Juan Pérez	y=418.069-1.81935*x	calle
Italica	y=38.4621+0.5707*x	calle
Homero	y=61.1124+0.56942*x	calle
laerte	y=111.206+0.63107*x	calle
Eolo	y=145.418+0.56082*x	calle
Ulises	y=186.98+0.57267*x	calle
Invierno	y=225.763+0.65323*x	calle
Proverbios	y=253.746+0.58928*x	calle

Figura B.6: Contenido de la tabla Calles

ID	NOMBRE	SITUACION
1	parque Rey	y>=798.63-2.35377*x AND y>=1324.45+0.011515*x*x-7.70909*x AND y<=638.436-1.61572*x AND y<=0.55883*x-114.10925
2	Albahaca	y>=661.554-1.91149*x AND y>=0.53002*x-95.81503 AND y<=708.065-1.80784*x AND y<=0.58864*x-65.70084
3	Azahares	y>=751.931-2.2404*x AND y>=0.7069*x-98.65191 AND y<=821.356-2.1736*x AND y<=0.59148*x-10.80666
4	Flores I	y>=574.472-1.57327*x AND y>=28.7305+0.51503*x AND y<=743.231-1.92326*x AND y<=30.1147+0.69344*x
5	Flores II	y>=662.294-1.92981*x AND y>=65.4749+0.59046*x AND y<=683.237-1.70075*x AND y<=117.151+0.57688*x
6	Flores III	y>=648.587-1.87108*x AND y<=7.72613*x-1293.75942 AND y<=82.0971+1.02014*x AND y<=798.652-2.16849*x AND y>=167.397+0.41206*x
7	Imperio	y>=1154.23+0.021644*x*x-8.53416*x AND y<=698.904-1.73947*x AND y<=251.249+0.53045*x
8	Palacio I	y>=644.717-1.88521*x AND y>=260.087+0.55861*x AND y<=753.542-2.06502*x AND y<=286.981+0.64113*x
9	Palacio II	y>=629.816-1.76586*x AND y>=297.388+0.64172*x AND y<=751.252-2.06111*x AND y<=344.903+0.62514*x
10	Bequer	y<=6.99408*x-317.76581 AND y>=617.263-1.61746*x AND y>=341.04+0.71547*x AND y<=701.496-1.72402*x AND y<=291.114+1.52933*x
11	Montesierra	y>=1023.49-3.91711*x AND y>=1.56411-0.00208*x AND y<=1495.43-5.04115*x AND y<=0.73781*x-156.84585
12	Albeniz	y>=545.248-1.99379*x AND y>=0.44282*x-73.12717 AND y<=597.696-1.91742*x AND y<=0.55765*x-65.77279
13	Galileo	y>=501.502-1.83301*x AND y>=171.022+0.76506*x AND y>=4.0179*x-292.29032 AND y<=663.418-2.4635*x AND y<=245.738+0.57854*x
14	Entre Parques	y>=528.236-2.03932*x AND y>=253.158+0.60383*x AND y<=573.651-1.74672*x AND y<=288.781+0.6278*x
15	Estrella Este	y>=503.713-1.74919*x AND y>=320.117+0.40117*x AND y<=593.791-2.0301*x AND y<=335.85+0.62979*x
16	Nueva Europa	y>=781.622-1.72354*x AND y>=0.59249*x-34.35807 AND y<=931.135-1.73801*x AND y<=0.60543*x-16.92547
17	Gitanilla I	y>=1009.73-2.39601*x AND y>=42.3518+0.49521*x AND y<=1236.44-2.96877*x AND y<=38.7733+0.56649*x
18	Gitanilla II	y>=726.372-1.49326*x AND y>=147.772+0.60913*x AND y<=863.995-1.72691*x AND y<=206.603+0.48227*x
19	Gitanilla III	y>=204.825+0.55405*x AND y>=710.96-1.3951*x AND y<=879.086-1.78843*x AND y<=203.93+0.64958*x
20	Gitanilla IV	y>=34.6539*x-5855.91605+0.04822*x*x AND y<=24.8521*x-4097.508+0.03437*x*x AND y>=952.302-1.70646*x AND y>=1.01896*x-22.76475
21	Gitanilla V	y>=1046.54-2.00222*x AND y>=210.684+0.5551*x AND y<=1143.14-2.02261*x AND y<=229.918+0.6269*x
22	Las Perlas	y>=738.927-1.83882*x AND y>=366.748+0.58911*x AND y<=864.74-2.16301*x AND y<=407.83+0.56307*x
23	Italica	y>=281.65-1.71925*x AND y>=0.52942*x-34.79724 AND y<=477.357-2.14978*x AND y<=2.08717+0.50255*x
24	Eliseo	y>=294.26-1.82841*x AND y>=0.64626*x-5.12689 AND y<=406.122-1.78733*x AND y<=39.4045+0.54061*x
25	Ciudad Verde	y>=295.512-1.84166*x AND y>=145.72+0.55885*x AND y<=440.555-2.04365*x AND y<=172.54+0.5581*x
26	Ciudad Blanca	y>=311.12-2.28052*x AND y>=191.15+0.58889*x AND y<=392.302-1.58437*x AND y<=222.117+0.52884*x
27	Caminos	y>=429.515-1.78383*x AND y>=258.822+0.57015*x AND y<=513.794-2.11745*x AND y<=291.153+0.56996*x

Figura B.7: Contenido de la tabla Residenciales

IDRESTRICCION	VARIABLE	RANGOINICIO	RANGOFIN
1	x	277.7758832854868	395.38826222467617
1	y	35.33053138942838	80.42598396383768
2	x	289.8013072860666	344.46615014679895
2	y	68.60899902565419	125.49217106931403
3	x	266.6914655183901	319.49194264273217
3	y	104.87990534646492	166.6481062375199
4	x	239.864095597758	292.61665790449985
4	y	163.70152782153056	217.59823173552616
5	x	216.69221223102636	268.50934067656914
5	y	204.89283237073636	259.7810842534029
6	x	194.48670775279064	243.49464149094482
6	y	245.08316955103206	330.3496198523122
7	x	150.90420734097688	241.62972703797243
7	y	313.7215565062881	358.8066832943434
8	x	140.72105884576757	190.73842618015095
8	y	348.0256875542157	399.9891724892255
9	x	118.4868565593635	167.59324975343694
9	y	385.32816234362093	420.5847256480299
10	x	83.34858917226916	148.19568878230442
10	y	418.6491748297344	485.3406464042011
11	x	249.08896601397714	298.1913921218703
11	y	.936911029336809	52.968355726419794
12	x	199.07670020179032	292.63517816621834
12	y	36.271180575623504	51.91429816350369
13	x	96.28969000146076	162.0624725420813
13	y	267.6796450370108	326.4247910748183
14	x	38.900475776590476	103.72944620107222

Figura B.8: Envolventes de las variables de Residenciales

ID	NOMBRE	SITUACION
1	Censo 1	y<=419.58+0.54445*x AND y>=185.99+0.61551*x AND y<=970.294-1.48215*x
2	Censo 2	y>=77.7246-1.45753*x AND y>=0.60054*x-4.29414 AND y<=1097.29-1.83763*x AND y<=182.185+0.61363*x
3	Censo 3	y<=0.61709*x-7.16005 AND y<=1045.76-1.74412*x AND y>=117.43-2.11156*x AND y>=0.55757*x-157.58308

Figura B.9: Contenido de la tabla de Censos

ID	SITUACION
1	y>=4844.9+0.023536*x^x-20.45744*x AND y<=464.78-0.06257*x
2	y>=2382.49-4.80871*x AND y>=468.259-0.06235*x AND y<=3454.67-7.21221*x AND y<=544.074-0.20781*x
3	y>=8216.56+0.03966*x^x-35.35461*x AND y<=402.092+0.011931*x
4	y>=1593.41-2.92636*x AND y>=354.845+0.035308*x AND y<=1309.21-2.10086*x AND y<=403.171-0.04307*x
5	y>=17930.8+0.08198*x^x-76.13863*x AND y<=301.571+0.050463*x
6	y>=975.987-1.67069*x AND y>=0.87921*x-140.92149 AND y<=1232.24-2.04651*x AND y<=33.7898+0.58749*x
7	y>=1049.94-1.81351*x AND y>=179.462+0.64301*x AND y<=1012.13-1.56996*x AND y<=310.149+0.41158*x
8	y>=767.101-1.91679*x AND y>=0.59435*x-100.41688 AND y<=602.197-1.3137*x AND y<=0.50279*x-36.22524
9	y>=767.775-1.56448*x AND y>=0.60298*x-111.6633 AND y<=958.725-1.7392*x AND y<=0.59265*x-64.96071
10	y>=655.526-1.47708*x AND y>=0.54682*x-62.06288 AND y<=759.818-1.43119*x AND y<=0.5639*x-7.39079
11	y>=861.506-1.94681*x AND y>=236.334+0.52113*x AND y<=859.177-1.49367*x AND y<=268.139+0.5379*x
12	y>=737.996-1.69971*x AND y>=172.744+0.58671*x AND y<=825.567-1.49447*x AND y<=261.993+0.48238*x
13	y>=952.795-1.95945*x AND y>=53.598+0.6232*x AND y<=890.215-1.60517*x AND y<=87.2253+0.73446*x
14	y>=849.398-1.90691*x AND y>=80.7159+0.53152*x AND y<=877.421-1.83025*x AND y<=108.928+0.67242*x
15	y>=630.297-2.57582*x AND y>=344.869+0.59751*x AND y<=629.61-1.67897*x AND y<=380.778+0.61216*x
16	y>=574.343-1.96574*x AND y>=193.423+0.48652*x AND y<=597.711-1.76501*x AND y<=201.182+0.62568*x
17	y>=573.594-1.88651*x AND y>=0.60246*x-157.0355 AND y<=607.131-1.8064*x AND y<=0.67617*x-146.56117
18	y>=542.266-1.92627*x AND y>=0.40778*x-101.70734 AND y<=685.895-1.81987*x AND y<=407.003+0.45526*x
19	y>=509.099-1.93267*x AND y>=25.0237+0.76126*x AND y<=588.549-1.70133*x AND y<=193.217+0.48819*x
20	y>=349.498-1.80081*x AND y>=315.97+0.5128*x AND y<=416.698-1.72243*x AND y<=329.189+0.6146*x
21	y>=331.837-1.58392*x AND y>=287.411+0.7362*x AND y<=401.627-1.64561*x AND y<=308.139+0.70721*x
22	y>=289.659-1.78442*x AND y>=230.581+0.55965*x AND y<=364.935-1.2045*x AND y<=247.884+0.46049*x
23	y>=242.291-2.01519*x AND y>=174.341+0.47695*x AND y<=392.032-1.55539*x AND y<=195.643+0.60965*x
24	y>=197.138-1.64146*x AND y>=33.9391+0.52863*x AND y<=414.679-1.93396*x AND y<=43.8982+0.6432*x
25	y>=177.953-1.30592*x AND y>=0.57141*x-49.70892 AND y<=436.893-1.97675*x AND y<=0.5414*x-28.2277

Figura B.10: Contenido de la tabla de Obras

Bibliografía

[1] J. M. Almendros-Jiménez, A. Becerra-Terón, A Safe Relational Calculus for Functional Logic Deductive Databases, Electronic Notes on Theoretical Computer Science 86 (3).

[2] W. G. Aref, H. Samet, Extending a DBMS with Spatial Operations, in: SSD '91: Proceedings of the Second International Symposium on Advances in Spatial Databases, Springer-Verlag, London, UK, 1991.

[3] M. P. Atkinson, F. Bancilhon, D. J. DeWitt, K. R. Dittrich, D. Maier, S. B. Zdonik, The Object-Oriented Database System Manifesto., in: the First International Conference on Deductive and Object-Oriented Databases, 1989.

[4] F. Bancilhon, S. Khoshafian, A Calculus for Complex Objects., in: PODS, 1986.

[5] S. Basu, Algorithms in semi-algebraic geometry, ph D thesis (1996).

[6] S. Basu, R. Dhandapani, R. Pollack, On the Realizable Weaving Patterns of Polynomial Curves in R3, in: Symposium on Graph Drawing, vol. 3383 of Lecture Notes in Computer Science, Springer, 2004.

[7] B. Beckermann, E. Saff, the sensitivity of least squares polynomial approximation (1998).

[8] N. Beckmann, H.-P. Kriegel, R. Schneider, B. Seeger, The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles., in: SIGMOD Conference, 1990.

[9] A. Belussi, E. Bertino, B. Catania, Manipulating Spatial Data in Constraint Databases, in: M. Scholl, A. Voisard (eds.), Proceedings of the 5th International Symposium on Large Spatial Databases (SSD), vol. 1262, Springer-Verlag, 1997.

[10] A. Belussi, E. Bertino, B. Catania, An Extended Algebra for Constraint Databases, Knowledge and Data Engineering 10 (5) (1998) 686–705.

[11] F. Benhamou, L. Granvilliers, Continuous and interval constraints, in: F. Rossi, P. van Beek, T. Walsh (eds.), Handbook of Constraint Programming, Foundations of Artificial Intelligence, chap. 16, Elsevier Science Publishers, Amsterdam, The Netherlands, 2006.

[12] F. Benhamou, P. V. Hentenryck, Introduction to the Special Issue on Interval Constraints., Constraints 2 (2) (1997) 107–112.

[13] M. Berler, J. Eastman, D. Jordan, C. Russell, O. Schadow, T. Stanienda, F. Velez, The object data standard: ODMG 3.0, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.

[14] C. Bessière, J.-C. Régin, Arc Consistency for General Constraint Networks: Preliminary Results., in: IJCAI (1), 1997.

[15] J. Bitner, E. M. Reingold, Backtracking programming techniques, Comm of the ACM 18 (11) (1975) 651–656.

[16] R. E. Bixby, E. A. Boyd, R. Z. Ríos-Mercado (eds.), Integer Programming and Combinatorial Optimization, 6th International IPCO Conference, Houston, Texas, USA, June 22-24, 1998, Proceedings, vol. 1412 of Lecture Notes in Computer Science, Springer, 1998 (1998).

[17] A. Brodsky, V. S. andJ. Chen, P. Exarkhopoulo, The CCUBE Constraint Object-Oriented Database System, International Conference on Management of Data Proceding, Philadelphia (1999) 577–579.

[18] A. Brodsky, Y. Kornatzky, The $L_{yli}c$ language: Querying constraint object, In *Int. Conf. ACM SIGMOD, on Management of Data*

[19] A. Brodsky, V. E. Segal, J. Chen, P. A. Exarkhopoulo, The CCUBE Constraint Object-Oriented Database System., Constraints 2 (3/4) (1997) 245–277.

[20] A. Brodsky, V. E. Segal, J. Chen, P. A. Exarkhopoulo, The CCUBE Constraint Object-Oriented Database System, in: SIGMOD '99: Proceedings of the 1999 ACM SIGMOD international conference on Management of data, ACM Press, 1999.

[21] A. Brodsky, X. S. Wan, On approximation-based query evaluation, expensive predicates and constraint objects, in: ILPS95 Workshop on Constraints, Databases and Logic Programming, Portland, 1995.

[22] C. W. Brown, Improved projection for Cylindrical Algebraic Decomposition, J. Symb. Comput. 32 (5) (2001) 447–465.

[23] B. Buchberger, Gröbner bases: An algorithmic method in polynomial ideal theory, Multidimensional Systems Theory, N. K. Bose, ed. (1985) 184–232.

[24] J.-H. Byon, P. Z. Revesz, DISCO: A Constraint Database System with Sets., in: CDB, 1995.

[25] Y. Caseau, P.-Y. Guillo, E. Levenez, A Deductive and Object-Oriented Approach to a Complex Scheduling Problem, Springer, Berlin, Heidelberg, 1993.

[26] R. G. G. Cattell, The Object Database Standard: ODMG-93 (Release 1.1), Morgan Kaufmann, 1994.

[27] B. Caviness, J. e. Johnson, Quantifier Elimination and Cylindrical Algebraic Decomposition, Springer-Verlag.

[28] S. Ceri, G. Gottlob, L. Tanca, What you always wanted to know about Datalog (and never dared to ask), vol. 1, IEEE Educational Activities Department, Piscataway, NJ, USA, 1989.

[29] L. Chang, A. K. Mackworth, Local Consistency in Junction Graphs for Constraint-Based Inference., in: AAAI, 2006.

[30] J. Chomicki, D. Q. Goldin, G. M. Kuper, D. Toman, Variable Independence in Constraint Databases., IEEE Trans. Knowl. Data Eng. 15 (6) (2003) 1422–1436.

[31] J. Chomicki, J. Marcinkowski, On the Computational Complexity of Consistent Query Answers, CoRR cs.DB/0204010.

[32] E. F. Codd, A Relational Model of Data for Large Shared Data Banks, Commun. ACM 13 (6) (1970) 377–387.

[33] E. F. Codd, Relational database: a practical foundation for productivity, Commun. ACM 25 (2) (1982) 109–117.

[34] J. Cohen, Constraint Logic Programming Languages., Commun. ACM 33 (7) (1990) 52–68.

[35] G. E. Collins, J. R. Johnson, W. Krandick, Interval arithmetic in Cylindrical Algebraic Decomposition, Journal of Symbolic Computation 34 (2) (2002) 145–157.

[36] R. M. Colomb, Deductive Databases and Their Applications, Taylor & Francis, Inc., Bristol, PA, USA, 1998.

[37] A. Croker, V. Dhar, A Knowledge Representation for Constraint Satisfaction Problems, IEEE Trans. Knowl. Data Eng. 5 (5) (1993) 740–752.

[38] P. Dasgupta, P. P. Chakrabarti, A. Dey, S. Ghose, W. Bibel, Solving Constraint Optimization Problems from CLP-Style Specifications Using Heuristic Search Techniques, IEEE Transactions on Knowledge and Data Engineering 14 (2) (2002) 353–368.

[39] J. Davenport, J. Heintz, Real Quantifier Elimination Is Doubly Exponential, J. Symb. Comput, 5 (1988) 29–35.

[40] R. Davis, Diagnostic reasoning based on structure and behavior, In Artificial Intelligence 24 (1984) 347–410.

[41] M. G. de la Banda, P. J. Stuckey, J. Wazny, Finding all minimal unsatisfiable subsets, in: PPDP '03: Proceedings of the 5th ACM SIGPLAN international conference on Principles and practice of declaritive programming, ACM Press, 2003.

[42] R. Dechter, Constraint Processing (The Morgan Kaufmann Series in Artificial Intelligence), Morgan Kaufmann, 2003.

[43] R. Dechter, I. Meiri, Experimental evaluation of preprocessing algorithms for constraint satisfaction problems, Artif. Intell. 68 (2) (1994) 211–241.

[44] J. V. den Bussche, Çonstraint Databases, Queries, and Query Languages", Constraint Databases, G. Kuper and L. Libkin and J. Paredaes, Springer, 2000.

[45] R. S. Devarakonda, Object-Relational Database Systems - The Road Ahead, ACM Electronic Magazine (Agosto).

[46] F. Dumortier, M. Gyssens, L. Vandeurzen, D. V. Gucht, On the Decidability of Semi-Linearity of Semi-Algebraic Sets and its Implications for Spatial Databases, in: Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 12-14, 1997, Tucson, Arizona, ACM Press, 1997.

[47] R. Fagin, Normal forms and relational database operators, in: SIGMOD '79: Proceedings of the 1979 ACM SIGMOD international conference on Management of data, ACM Press, New York, NY, USA, 1979.

[48] J. Faugère, A new efficient algorithm for computing Gröbner bases without reduction to zero (f5), ACM Press, New York, NY, USA, 2002.

[49] J.-C. Faug'ere, A new efficient algorithm for computing gröbner bases (f_4), Journal of Pure and Applied Algebra 139 (1-3) (1999) 61–88.

[50] L. Fergaras, D. Maier, Towards an effective calculus for object query processing, In Proc ACM SIGMOD Conf. on Management of Data, 1995.

[51] R. Finkelstein, Is your database relational? Part 2 of Dr. Codd's rules for relational databases, Data Based Advis. 5 (6) (1987) 66–68.

[52] P. Flener, B. Hnich, Z. Kiziltan, A meta-heuristic for subset problems, in: PADL '01: Proceedings of the Third International Symposium on Practical Aspects of Declarative Languages, Springer-Verlag, London, UK, 2001.

[53] E. C. Freuder, A sufficient condition for backtrack-free search, J. ACM 29 (1) (1982) 24–32.

[54] E. Fuchs, K. Donner, Fast Least-Squares Polynomial Approximation in Moving Time Windows, in: ICASSP '97: Proceedings of the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '97)-Volume 3, IEEE Computer Society, Washington, DC, USA, 1997.

[55] R. Gasca, C. D. Valle, R. Ceballos, M. Toro, An Integration of FDI and DX approaches to polinomial models, 14th International Workshop on principles of Diagnosis - DX.

[56] R. M. Gasca, J. A. Ortega, M. Toro, A framework for semiqualitative reasoning in engineering applications., Applied Artificial Intelligence 16 (3) (2002) 173–197.

[57] F. Geerts, B. Kuijpers, On the decidability of termination of query evaluation in transitive-closure logics for polynomial constraint databases., Theor. Comput. Sci. 336 (1) (2005) 125–151.

[58] D. Goldin, P. Kanellakis, Constraint Query Algebras Constraints, Journal E. Freuder editor.

[59] D. Goldin, A. Kutlu, M. Song, Extending the Constraint Database framework, in: PCK50: Proceedings of the Paris C. Kanellakis memorial workshop on Principles of computing & knowledge, ACM Press, New York, NY, USA, 2003.

[60] D. Q. Goldin, Taking Constraints out of Constraint Databases., in: Constraint DataBases, vol. 3074 of Lecture Notes in Computer Science, Springer, 2004.

[61] D. Q. Goldin, A. Kutlu, M. Song, F. Yang, The Constraint Database Framework: Lessons Learned from CQA/CDB., in: ICDE, 2003.

[62] S. W. Golomb, L. D. Baumert, Backtrack programming., J. ACM 12 (4) (1965) 516–524.

[63] J. C. González-Moreno, M. T. Hortalá-González, M. Rodríguez-Artalejo, Semantics and Types in Functional Logic Programming., in: Fuji International Symposium on Functional and Logic Programming, 1999.

[64] L. Granvilliers, F. Goualard, F. Benhamou, Box Consistency through Weak Box Consistency, in: ICTAI '99: Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence, IEEE Computer Society, Washington, DC, USA, 1999.

[65] R. Greiner, B. A. Smith, R. W. Wilkerson, A correction to the algorithm in Reiter's theory of diagnosis, Artificial Intelligence 41 (1) (1989) 79–88.

[66] R. Gross, R. W. Marti, DeCor a Deductive Constraint Database System., in: WLP, 1994.

[67] R. Gross, R. W. Marti, Managing Temporal Knowledge using a Deductive Constraint Database System., in: BTW, 1995.

[68] R. Gross-Brunschwiler, Implementation of Constraint Database Systems Using a Commpiler-Time Rewrite Approach, Ph.D. thesis, Swis Federal Institute of Technology , Zürich (May 1996).

[69] S. Grumbach, P. Rigaux, M. Scholl, L. Segoufin, DEDALE, A Spatial Constraint Database., in: Workshop on Database Programming Languages, 1997.

[70] S. Grumbach, P. Rigaux, L. Segoufin, The DEDALE System for Complex Spatial Queries., in: SIGMOD Conference, 1998.

[71] M. Hadjieleftheriou, G. Kollios, P. Bakalov, V. J. Tsotras, Complex Spatio-Temporal Pattern Queries., in: VLDB, 2005.

[72] A. Y. Halevy, I. S. Mumick, Y. Sagiv, O. Shmueli, Static analysis in Datalog extensions, J. ACM 48 (5) (2001) 971–1012.

[73] J. Han, Constraint-Based Query Evaluation in Deductive Databases, Knowledge and Data Engineering 6 (1) (1994) 96–107.

[74] J. Han, Y. Fu, W. Wang, K. Koperski, O. Zaiane, Dmql: A data mining query language for relational databases, in: Proc.oftheSIG-MOD'96 DKMD Workshop, 1996.

[75] W. Hodges, Some Strange Quantifiers., in: Structures in Logic and Computer Science, 1997.

[76] X. Huang, A General Extension of Constraint Propagation for Constraint Optimization., in: CP, 2004.

[77] T. Hulubei, B. O'Sullivan, Search Heuristics and Heavy-Tailed Behaviour., in: CP, 2005.

[78] ILOG, Jsolver 2.1, Reference Manual April.

[79] Ilog, Inc., Solver CPLEX, http://www.ilog.fr/products/cplex/.

[80] R. Isermann, Supervision, fault-detection and fault-diagnosis methods an introduction, Control Engineering Practice 5 (1997) 639–652.

[81] J. Jaffar, J.-L. Lassez, Constraint Logic Programming., in: POPL, 1987.

[82] J. Jaffar, M. J. Maher, Constraint Logic Programming: A Survey., J. Log. Program. 19/20 (1994) 503–581.

[83] P. W. P. Jr., Search rearrangement backtracking and polynomial average time., Artif. Intell. 21 (1-2) (1983) 117–133.

[84] P. C. Kanellakis, G. M. Kuper, P. Revesz, Constraint Query Languages, Symposium on Principles of Database Systems (1990) 299–313.

[85] K. Kask, New Search Heuristics for Max-CSP, in: Principles and Practice of Constraint Programming, 2000.

[86] S. Khoshafian, R. Abnous, Object-Orientation: Concepts, Languages, Databases, User Interfaces, John Wiley & Sons, New Jersey, 1990.

[87] M. Kifer, W. Kim, Y. Sagiv, Querying Object-Oriented Databases., in: SIGMOD Conference, 1992.

[88] J. D. Kleer, An Assumption-based Truth Maintenance System, Artificial Intelligence 28 2 (1986) 127–161.

[89] J. D. Kleer, A. Mackworth, R. Reiter, Characterizing diagnoses and systems, Artificial Intelligence 56 2-3 (1992) 197–222.

[90] J. D. Kleer, B. Williams, Diagnosing multiple faults, Art. Int.

[91] M. Koubarakis, S. Skiadopoulos, Tractable Query Answering in Indefinite Constraint Databases: Basic Results and Applications to Querying Spatiotemporal Information, in: Spatio-Temporal Database Management, 1999.

[92] T. Kudrass, M. Conrad, Management of XML Documents in Object-Relational Databases., in: EDBT Workshops, 2002.

[93] G. Kuper, L. Libkin, J. Paredaes, Constraint Databases, Springer, 1998.

[94] G. Kuper, L. Libkin, J. Paredaes, "Introduction", Constraint Databases, G. Kuper and L. Libkin and J. Paredaes, Springer, 2000.

[95] G. Kuper, L. Libkin, J. Paredaes, "The DISCO SYSTEM", Constraint Databases, G. Kuper and L. Libkin and J. Paredaes, Springer, 2000.

[96] H.-J. Lee, S.-W. Lee, H.-J. Kim, Design and implementation of an extended relationship semantics in an ODMG-compliant OODBMS, J. Syst. Softw. 76 (2) (2005) 171–180.

[97] C. K.-S. Leung, Dynamic FP-Tree Based Mining of Frequent Patterns Satisfying Succinct Constraints., in: CDB, 2004.

[98] L. Libkin, Variable independence for first-order definable constraints, ACM Trans. Comput. Logic 4 (4) (2003) 431–451.

[99] A. Lomonosov, Graph and combinatorial algorithms for geometric constraint solving, Ph.D. thesis, chair-Meera Sitharam (2004).

[100] S. Lucas, Context-sensitive Computations in Functional and Functional Logic Programs., Journal of Functional and Logic Programming 1998 (1).

[101] A. Macintyre, K. McKenna, L. van den Dries, Elimination of quantifiers in algebraic structures, Adv. Math. 47 (1) (1983) 74–87.

[102] A. K. Mackworth, Consistency in networks of relations., Artif. Intell. 8 (1) (1977) 99–118.

[103] A. K. Mackworth, E. C. Freuder, The Complexity of Constraint Satisfaction Revisited, Artificial Intelligence 59 (1-2) (1993) 57–62.

[104] D. Maier, J. B. Cushing, Treating Programs as Objects: The Computational Proxy Experience, Springer, Berlin, Heidelberg, 1993.

[105] G. Maps, http://maps.google.es/maps (2007).

[106] E. Marcos, B. Vela, J. M. Cavero, A methodological approach for Object-Relational Database design using uml., Inform., Forsch. Entwickl. 18 (3-4) (2004) 152–164.

[107] K. Marriott, P. J. Stuckey, "Programming with Constraints. An introduction", Simplification, Optimization and Implication, The Mitt Press, 1998.

[108] J. L. Martin, V. Reiner, Factorizations of some weighted spanning tree enumerators, Journal of Combinatorial Theory, Series A 104 (2) (2003) 287–300.

[109] E. Mayol, E. Teniente, Consistency preserving updates in Deductive Databases, vol. 47, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, 2003.

[110] S. McCallum, Solving Polynomial Strict Inequalities using Cylindrical Algebraic Decomposition., Comput. J. 36 (5) (1993) 432–438.

[111] U. Montanari, Networks of constraints: Fundamental properties and applications to picture processing., Inf. Sci. 7 (1974) 95–132.

[112] B. A. Nadel, Tree search and ARC consistency in constraint satisfaction algorithms, Springer-Verlag (1988) 287–342.

[113] B. A. Nadel, Constraint satisfaction algorithms, Comput. Intell. 5 (4) (1990) 188–224.

[114] S.Ñieva, F. Sáenz-Pérez, J. Sánchez, Towards a Constraint Deductive Database Language based on Hereditary Harrop Formulas, in: P. Lucio, F. Orejas (eds.), Sextas Jornadas de Programación y Lenguajes, PROLE, 2006.

[115] M.Ñyberg, M. Krysander, Combining AI, FDI, and statistical hypothesis-testing in a framework for diagnosis, in: Proceedings of IFAC Safeprocess'03, Washington, USA, 2003.

[116] A. Olivé, Integrity Constraints Checking In Deductive Databases, in: G. M. Lohman, A. Sernadas, R. Camps (eds.), 17th International Conference on Very Large Data Bases, September 3-6, 1991, Barcelona, Catalonia, Spain, Proceedings, Morgan Kaufmann, 1991.

[117] E. Oomoto, M. Kamitani, T. Takamatsu, Path existence constraints in Object-Oriented Databases (2000) 45–58.

[118] B. O'Sullivan, P. van Beek, Introduction to the Special Issue on Principles and Practice of Constraint Programming (cp 2005)., Constraints 11 (2-3) (2006) 83–84.

[119] E. Pardede, J. W. Rahayu, D. Taniar, On using collection for aggregation and association relationships in XML object-relational storage, in: SAC '04: Proceedings of the 2004 ACM symposium on Applied computing, ACM Press, New York, NY, USA, 2004.

[120] K. Parsaye, M. Chignell, Intelligent database tools & applications, John Wiley & Sons, Inc., New York, NY, USA, 1993.

[121] S. Prestwich, Stochastic local search in constrained spaces (2000) 27–39.

[122] S. Prestwich, Local Search and Backtracking vs Non-Systematic Backtracking Technical Report FS-01-04 (2001) 109–115.

[123] B. Pulido, C. Alonso, Possible conflicts, ARRs, and conflicts (2002).

[124] V. Ramanathan, P. Z. Revesz, Constraint Database Solutions to the Genome Map Assembly Problem., in: Constraint DataBases, vol. 3074 of Lecture Notes in Computer Science, Springer, 2004.

[125] S. Rasetic, J. Sander, J. Elding, M. A. Nascimento, A Trajectory Splitting Model for Efficient Spatio-Temporal Indexing., in: VLDB, 2005.

[126] S. Ratschan, Approximate quantified constraint solving by cylindrical box decomposition (2002).

[127] R. Reiter, A theory of diagnosis from first principles, Artificial Intelligence 32 1 (1987) 57–96.

[128] W. Research, Mathematica 5, Reference Manual April.

[129] P. Revesz, Datalog and Constraints, Constraint Databases (2000) 155–170.

[130] P. Revesz, Introduction to Constraint Databases, Springer, 2001.

[131] P. Z. Revesz, Problem Solving in the DISCO Constraint Database System, in: V. Gaede, A. Brodsky, O. Gunter, D. Srivastava, V. Vianu, M. Wallace (eds.), Proceedings of the 2nd Workshop on Constraint Databases and Applications, vol. 1191, Springer-Verlag, 1997.

[132] P. Z. Revesz, MLPQ/PReSTO User Manual, http:// www.cse.unl.edu/ ∼revesz/MLPQ/mlpq.htm.

[133] P. Z. Revesz, R. Chen, P. Kanjamala, Y. Li, Y. Liu, Y. Wang, The MLPQ/GIS Constraint Database System, in: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA, ACM, 2000.

[134] P. Rigaux, M. Scholl, L. Segoufin, S. Grumbach, Building a constraint-based spatial database system: model, languages, and implementation., Inf. Syst. 28 (6) (2003) 563–595.

[135] O. W. Salomons, F. van Slooten, F. J. A. M. van Houten, H. J. J. Kals, Conceptual graphs in constraint based re-design, in: SMA '95: Proceedings of the third ACM symposium on Solid modeling and applications, ACM Press, New York, NY, USA, 1995.

[136] S. Sarawagi, S. Thomas, R. Agrawal, Integrating Association Rule Mining with Relational Database Systems: Alternatives and Implications, Data Min. Knowl. Discov. 4 (2-3) (2000) 89–125.

[137] R. Selje, A new method for integrity constraint checking in Deductive Databases, Data Knowl. Eng. 15 (1) (1995) 63–102.

[138] F. G. Sener, New Module for Damage Probabilistic Calculations with efficient user interface and including the latest classification society rules, internal Report (2005).

[139] M. Shakeri, Implementation of an automated operation planning and optimum operation sequencing and tool selection algorithms, Comput. Ind. 54 (3) (2004) 223–236.

[140] A. Soffer, H. Samet, Two data organizations for Storing Symbolic Images in a Relational Database System, in: DS-8: Proceedings of the IFIP TC2/WG2.6 Eighth Working Conference on Database Semantics- Semantic Issues in Multimedia Systems, Kluwer, B.V., Deventer, The Netherlands, The Netherlands, 1998.

[141] M. Spivey, An introduction to logic programming through Prolog, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

[142] M. Staroswiecki, P. Declerk., Analytical redundancy in non linear interconnected systems by means of structural analysis., in: IFAC Advanced Information Processing in Automatic Control (AIPAC-89), Nancy, France, 1989.

[143] M. Stonebraker, D. Moore, P. Brown, Object-Relational DBMSs: Tracking the Next Great Wave, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998.

[144] A. W. Strzebonski, Cylindrical Algebraic Decomposition using validated numerics, Journal of Symbolic Computation 49 (9) (2006) 1021–1038.

[145] T. Sturm, Reasoning over Networks by Symbolic Methods, Applicable Algebra in Engineering, Communication and Computing 10 (1) (1999) 79–96.

[146] V. Telerman, Using Constraint Solvers in CAD/CAM Systems, in: PSI '02: Revised Papers from the 4th International Andrei Ershov Memorial Conference on Perspectives of System Informatics, Springer-Verlag, London, UK, 2001.

[147] D. Toman, Constraint Databases and Program Analysis Using Abstract Interpretation, in: Constraint Databases and Applications, 1997.

[148] E. Tsang, J. Borrett, A. Kwan, An attempt to map the performance of a range of algorithm and heuristic combinations (1995).

[149] J. Ullman, Assigning an appropriate meaning to database logic with negation (1994).

[150] P. Veltri, Constraint database query evaluation with approximation, in: ITCC '01: Proceedings of the International Conference on Information Technology: Coding and Computing, IEEE Computer Society, Washington, DC, USA, 2001.

[151] R. Wallace, Factor Analytic Studies of CSP Heuristics., in: CP, 2005.

[152] V. Weispfenning, Aspects of quantifier elimination in algebra, in: Universal algebra and its links with logic, algebra, combinatorics and computer science, vol. 4 of R & E Res. Exp. Math., Heldermann, Berlin, 1984.

[153] V. Weispfenning, The complexity of linear problems in fields, Journal of Symbolic Computation (1988) 3 – 27.

[154] P. T. Whelan, CAD/CAM database management system requirements for mechanical parts, ph D thesis (1989).

[155] F. Wotawa, A variant of reiter's hitting-set algorithm., Inf. Process. Lett. 79 (1) (2001) 45–51.

[156] S. B. Zdonik, D. Maier (eds.), Readings in Object-Oriented Database Systems, Morgan Kaufmann, 1990.