CSP and Restricted Databases

M. T. Gómez, R. M. Gasca, C. Del Valle, R. Ceballos

Depto de Lenguajes y Sistemas,
Universidad de Sevilla, Sevilla, España,

e-mail: {mayte, gasca, carmelo, ceballosg}@lsi.us.es

Constraint Databases were proposed because it was necessary to represent infinite relations in a more modular and compact way. In this way, Constraint Databases were used to handle continuous data, like spatio-temporal, and to enrich both the data model and the queries with constraints. It allows us to handle these constraints and it makes easier to construct and model the Constraint Satisfaction Problems (CSP) when we want to evaluate these queries.

In this paper, we carry out a study of the different reasons, methodologies and tools that have helped to the development of Constraint Databases. Also, we present a study of some defects and how to improve them. To help us in the CSP construction, we show a modular framework, with all the advantages that it implies. Finally, we show an example to understand better the reasons that have helped to the development of our system.

Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No.20 (2003)
ISSN: 1137-3601. © AEPIA (http://www.aepia.org/revista) pp 151-163.

CSP y Bases de Datos Restrictivas

M. T. Gómez, R. M. Gasca, C. Del Valle, R. Ceballos

Depto de Lenguajes y Sistemas, Universidad de Sevilla,
Sevilla, España,
{mayte, gasca, carmelo, ceballos}@lsi.us.es

Resumen

Las Bases de Datos Restrictivas se originaron ante la necesidad de representar de forma más compacta y modular datos de gran tamaño. De esta forma, y como medio para tratar datos continuos como es el caso de los espacio-temporales, se optó por tratar la información como restricciones almacenadas en una base de datos. Gracias a esta forma de tratar las restricciones, se facilita la construcción y el modelado de problemas de satisfacción de restricciones (CSP) y su posterior resolución. En este artículo, se realiza un recorrido por las distintas razones, metodologías y herramientas que han ayudado al desarrollo de las Bases de Datos Restrictivas. Junto a dicho estudio, se lleva a cabo un análisis de sus deficiencias y de los posibles aspectos a mejorar. Para aumentar la habilidad en la construcción de modelos, y ayudando a la resolución de problemas de satisfacción de restricciones (CSP), se ofrece una arquitectura de implementación modular, con las ventajas que eso conlleva. Para finalizar, se presenta un ejemplo que aclara las razones que han movido al desarrollo de nuestra propuesta.

Palabras clave: Bases de Datos Restrictivas, CSP, Álgebra Relacional, Base de Datos Relacional, Lenguajes de Consulta.

1. Introducción

Para el desarrollo de ciertas aplicaciones, es necesario la utilización de datos continuos, para los que las bases de datos convencionales presentan considerables limitaciones. Por ejemplo, ante la necesidad de almacenar y tratar información espacio-temporal, científica, médica, biológica..., aparece el problema del carácter físico de los soportes, que obliga a un tamaño finito, junto al inconveniente que genera el almacenamiento masivo de datos, tanto para su tratamiento, como a la hora de inferir conocimiento partiendo de dichos datos. Pese a las limitaciones que en principio parecen presentar las tecnologías relacionales para el tratamiento de las datos actuales, no se debe olvidar al potencia que ofrecen a la hora de deducir conocimiento de registros relacionales.

Partiendo de este planteamiento, las Bases de Datos Restrictivas, pretenden resolver los problemas aprovechando tanto la versatilidad de las bases de datos relacionales, como la potencia de la programación con restricciones para representar información continua. Combinando ambos campos, será posible almacenar la información de gran tamaño en un número reducido de registros.
Un ejemplo donde se tienen datos que no son convencionales, en lo que respecta a su forma de almacenamiento, podría ser un sistema donde se encuentran un conjunto de parcelas, las cuales tienen al menos un propietario. La información que es necesaria almacenar, es de tipo discreto en el caso de los datos de los propietarios (nombre, dirección...), pero de carácter geográfico para poder localizar las parcelas. En ejemplos de este tipo, sería oportuno la utilización de Bases de Datos Relacionales, ya que se puede integrar las bases

de datos relacionales, con la versatilidad que ello conlleva, y los problemas de satisfacción de restricciones para poder resolver consultas como por ejemplo 'quiénes son los propietarios de las parcelas por las que pasa una carretera'.
En este artículo, se pretende repasar los orígenes históricos de las Bases de Datos Restrictivas, gracias a las aplicaciones que ofrece, y la situación actual. Como aportación, se muestra una nueva arquitectura para la construcción de CSP ayudado por Bases de Datos Restrictivas, junto a un ejemplo para analizar su funcionalidad.

2. Introducción Histórica

El campo de las Bases de Datos Restrictivas [1, 5] tuvo su origen a principios de 1990 en el artículo de Kanellakis, Kuper y Revesz [3], cuyo objetivo era definir una versión de bases de datos para la *Programación Lógica con Restricciones (CLP)*, continuando la misma línea de la definición de Datalog, como una nueva versión de base de datos Prolog.
El objetivo inicial de las Bases de Datos Restrictivas, fue abarcar aplicaciones donde los datos podían ser representados como un gran conjunto de restricciones, combinando las bases de datos con la Programación Lógica con Restricciones. Los desarrollos en este campo, se incentivaron aún más ante la necesidad de nuevas formas de almacenamiento, ocasionado por la existencia de gran cantidad de datos de carácter continuo. La idea clave fue la noción de que una tupla en una base de datos relacional, podía ser sustituida por un conjunto de restricciones, y que muchas de las características del modelo relacional podían ser extendidas al campo de la lógica con restricciones. De esta manera un lenguaje de consulta, permitirá a los usuarios obtener la información almacenada en una base de datos convencional, de una forma trasparente. Para abordar esta propuesta, se partió de un modelo de datos ya introducido [1], fundamentado en considerar una tupla en una base de datos, como un conjunto de restricciones lineales de igualdad o desigualdad en los atributos de dicha tupla, ya que las restricciones son un mecanismo natural de especificar las consultas de similitud en un conjunto de datos.

2.1. Aplicaciones de las Bases de Datos Restrictivas

Considerando las limitaciones que las bases de datos convencionales, presentan ante las necesidades que plantea la inferencia de conocimiento en algunas de las aplicaciones actuales, se abre el campo de la tecnología en Bases de Datos Restrictivas, que originalmente involucran datos espaciales [10]. Muchos de los datos espaciales pueden ser descritos utilizando restricciones lineales o polinómicas. Mientras que las restricciones polinómicas son más generales, también tienen más coste de implementación y ejecución. Los sistemas de restricciones lineales son más eficientes a la hora de resolverse, aunque más limitados en el tipo de datos que pueden representar. Pese a ésto, las restricciones lineales son suficientes en muchas de las aplicaciones, incluyendo a los Sistemas de Información Geográfica (GIS). En la práctica, los datos espaciales en un GIS consisten en segmentos lineales, de forma que las restricciones lineales son particularmente apropiadas para representar estos datos.
La principal diferencia entre las Bases de Datos Restrictivas Lineales y la mayoría de las sistemas geográficos, radica en que las Bases de Datos Restrictivas están basadas en un modelo relacional, suministrando una estructura uniforme para el tratamiento de los datos tanto espacio-temporales, como con otros tipos de información. Ésto provoca que uno de los mayores problemas de los GIS, haya sido la difícil integración con otros tipos datos, lo que es mejorable combinando la información espacio-temporal con una estructura uniforme resultante de un lenguaje de consulta, y facilitando su uso.
La expresividad de las Bases de Datos Relacionales puede ser también aplicadas a problemas de configuración o en sistemas dinámicos, que han sido resueltos mediante otras técnicas en trabajos anteriores [2, 7].

2.2. Herramientas y Prototipos para Bases de Datos Restrictivas

En la actualidad existen varias propuestas en la implementación y construcción de prototipos para Bases de Datos Restrictivas, cuyo principal objetivo es el tratamiento de sistemas espacio-temporales. A continuación, se hace un recorrido por las de mayor transcendencia.

2.2.1. MLPQ

MLPQ (Management of Linear Programming Queries) es un sistema desarrollado en el Departamento de Ciencia de la Computación e Ingeniería de la Universidad de Nebraska [12], para la gestión de consultas con programación lineal para una Base de Datos Restrictiva, que permite consultas Datalog, y añadir operadores sobre funciones lineales.
Hay dos áreas principales de desarrollo para el sistema MLPQ. La primera es en la investigación de operadores cuando los datos disponibles en una base de datos, necesitan ser modificados por alguna consulta, suministrando en este caso MLPQ de la flexibilidad necesaria. La segunda, y principal aplicación de MLPQ, es para el tratamiento de información espacio-temporal, gracias a un entorno gráfico que permite trabajar en 2 ó 3 dimensiones, mediante la interacción del usuario con el conocimiento almacenado, todo ello mediante consultas Datalog.

2.2.2. PReSTO

PReSTO (Parametric Rectangle Spatiotemporal Objects [5, 13]) es un sistema de base de datos espacio-temporales, que al igual que MLPQ, fue desarrollado en el Departamento de Ciencia de la Computación e Ingeniería de la Universidad de Nebraska. Dicho sistema permite consultas del álgebra relacional a sistemas que cambian en función del tiempo. El formato de fichero de entrada en PReSTO es de la forma:

$$
\begin{array}{l}
begin\,\%STDB\,\% \\
\quad r_1 \\
\quad r_2 \\
\quad \vdots \\
\quad r_n \\
end\,\%STDB\,\%
\end{array}
$$

Donde cada r_i es una regla o un parámetro para la representación de periodicidad. Un ejemplo de reglas, ofrecido por la herramienta, podrían ser:

$$
\begin{array}{l}
clouds(h) : -i = 1, \\
\quad x1 - t = 80, y1 - 0{,}5t = 100, \\
\quad x2 - 1{,}1t = 220, y2 - 0{,}6t = 200, \\
\quad t >= 0, t <= 300, \\
\quad p = -1, s = 0, \\
\quad h = 0. \\
clouds(h) : -i = 1, \\
\quad x1 - t = 105, y1 - 0{,}5t = 200, \\
\quad x2 - t = 111, y2 - 0{,}6t = 205, \\
\quad t >= 0, t <= 300, \\
\quad p = -1, s = 0, \\
\quad h = 0. \\
clouds(h) : -i = 1, \\
\quad x1 - t = 111, y1 - 0{,}5t = 200, \\
\quad x2 - 1{,}1t = 225, y2 - 0{,}6t = 210, \\
\quad t >= 0, t <= 300, \\
\quad p = -1, s = 0, \\
\quad h = 0.
\end{array}
$$

En este caso se muestra un ejemplo donde un elemento, en este caso una nube (cloud), cambia su posición en función del tiempo (t).

2.2.3. MLPQ/PReSTO

El sistema MLPQ/Presto [5, 13] proviene de la combinación de los sistemas MLPQ y PReSTO. Un ejemplo, ofrecido por la herramienta, de los ficheros que son tratados y almacenados podría ser:

$$
\begin{array}{l}
AK(1, x, y) : - \\
\quad -27x + 39y < 5682, \\
\quad 8x + 67y < 14656, \\
\quad 75x + 19y < 15575, \\
\quad -33x + 33y > -1683, \\
\quad -53x + 57y > -2335, \\
\quad 59x + 43y > 6751, \\
\quad -83x + 8y < -2785.
\end{array}
$$

$$
\begin{array}{l}
AK(1, x, y) : - \\
\quad 28x + 9y < 6053, \\
\quad -5x + 42y > 3149, \\
\quad -33x + 33y < -1683.
\end{array}
$$

$$
\begin{array}{l}
AK(1, x, y) : - \\
\quad 34x + 80y < 14050, \\
\quad -16x + 13y > -3421, \\
\quad 41x + 67y > 13481, \\
\quad 9x > 1665.
\end{array}
$$

$$
\begin{array}{l}
HI(2, x, y) : - \\
\quad -24x + 10y < -7298, \\
\quad 28x + 30y < 12806, \\
\quad -18x + 24y > -4986, \\
\quad 22x + 16y > 8678.
\end{array}
$$

$$
\begin{array}{l}
WA(3, x, y) : - \\
\quad 7x + 36y < 20488, \\
\quad -39x + 22y > 10270, \\
\quad 46x + 14y > 8660.
\end{array}
$$

```
/* the capital cities */
Capital( 1, name) :- name="Juneau".
Capital( 2, name) :- name="Honolulu".
Capital( 3, name) :- name="Seattle".
```

En este ejemplo, se muestra la situación geográfica de un conjunto de estados americanos, y las capitales de cada uno de ellos.

2.2.4. DEDALE

DEDALE [15] es una implementación realizada en 1999 por miembros de centro de investigación CNAM y del instituto francés para la investigación en informática (INRIA). El objetivo principal de este sistema se basa en aplicaciones de carácter geométrico, basándose en un modelo de restricciones lineales con bases de datos, dando solución a los GIS, y a los sistemas espacio-temporales. DEDALE ofrece un lenguaje para la consulta a bases de datos de conocimiento mediante un interfaz gráfico, que permite obtener información de la base de datos, y mostrar los resultados. Para la representación de las restricciones se utiliza orientación a objeto, más apropiado para estructura de datos complejos. Un ejemplo de los datos que permite DEDALE [18] puede ser de la forma:

PEOPLE={[
name : string,
category : string,
activity : {name : string, time : $\mathbb{Q}$},
trajectory : {space : $\mathbb{Q}^2$, time : $\mathbb{Q}$}
]}

En este ejemplo, se observa el tratamiento de datos espacio-temporales, como es el caso de la *trajectoria*. Una posible instancia de este ejemplo sería la descrita en la tabla 1 [18].

2.2.5. CCUBE

CCUBE (Constraint Object-Oriented Database System) es la primera implementación de una Base de Datos Restrictiva orientada a objetos [16], desarrollada por el grupo de investigación en Programación con restricciones y Bases de Datos de la Universidad de George Mason en 1999. El sistema se basa en dar uniformidad a la heterogeneidad de los datos existentes en la actualidad, como es el caso de las trayectorias en 4 dimensiones, interconexión entre distintos sistemas coordinados entre si o GIS. Aportando principalmente, además de las ventajas de la combinación de las restricciones con las bases de datos, la implementación a alto nivel mediante la orientación a objetos, lo que provee de mayor versatilidad a la hora de construir un sistema software.

2.2.6. Inconvenientes de las Herramientas

Los inconvenientes de la utilización de algunas de las herramientas actuales para el tratamiento de Bases de Datos Restrictivas, radica en varios puntos:

1. Pese a incluir un interfaz gráfico, necesitan que el usuario conozca lenguajes de consulta como SQL o Datalog, ya que la interacción del usuario con la información se realiza con la ayuda de ellos.
2. Tratamiento de restricciones únicamente lineales, que pese a ser más eficientes no reflejan todo el dominio de las aplicaciones existentes en la actualidad.
3. Poca modularidad en la forma de almacenar datos, lo que dificulta la adaptación con sistemas ya existentes.
4. Tratamiento con grandes cantidades de información, ya que la capacidad del interfaz gráfico para mostrar datos es limitada. Si los rangos de los dominios de los datos que se presentan, son muy diferentes, es difícil mostrar la información el sistema con claridad y de forma fidedigna.
5. Dificultad en el almacenamiento masivo de datos, ya que en sistemas como MLPQ/PReSTO se utilizan ficheros de texto (Datalog) como medio para el almacenamiento. Si se utilizaran bases de datos relacionales, sería posible tratar gran volumen de información y hacer uso de las ventajas que ofrecen las bases de datos relacionales.

3. Consultas a Bases de Datos Restrictivas

Una consulta es una fórmula en una determinada lógica, donde tanto la entrada como la salida

Name	Category	Activity	Trajectory
Jonh	Tourist	(name='Sleep' $\wedge 2 < t < 10$) $\vee$ (name='Eat' $\wedge 10 < t < 11$) $\vee$ (name='Ski' $\wedge 11 < t < 15$)) ...	($x = 1230 \wedge y = 134$ $\wedge 1 < t < 11$) $\vee (3x - 2y \leq 49$ $\wedge 23x + 2y \geq 134 \ldots$ $\wedge 11 < t < 16$) ...

Tabla 1: Ejemplo de DEDALE para una tupla.

son un conjunto de relaciones. En el caso del tratamiento de las Bases de Datos Restrictivas, se pueden utilizar diferentes lenguajes de consulta, como es el caso del álgebra relacional o el lenguaje SQL.
Una vez construidas las consultas, éstas son 'evaluadas' 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 disjunta (DNF).
Las propiedades que constituyen las consultas a bases de datos relacionales descrita abajo, son entre otras las que contribuyen al éxito del modelo relacional:
Clausura: Propiedad de los lenguajes cuya álgebra relacional es cerrada, lo que significa que la entrada son un conjunto de relaciones y la salida es de la misma forma. La clausura es una propiedad muy útil, por ejemplo que se pueda descomponer una consulta para optimizar cierto propósito, y atender cada parte de la relación como una consulta en ella misma.
Eficiencia de Evaluación: Las consultas pueden evaluarse eficientemente usando las nociones de complejidad de dato, que consiste en evaluar una consulta en términos del tamaño de la base de datos.
Cálculo/Álgebra: Gracias a la existencia de equivalencias entre el álgebra relacional y los lenguajes de consulta a más alto nivel, es posible que el usuario construya consultas en lenguaje declarativo, mientras que el sistema utilice un lenguaje procedural para la obtención de resultados.
La unión de lenguajes de consulta con restricciones (CQL) [3], se originó ante la búsqueda de mayor potencia y la posibilidad de utilizar estructura de datos más complejas. De esta forma en [3] se propone un una sintaxis del CQL, que es la unión de los lenguajes de consulta convencionales y la lógica.
Un ejemplo para aclarar el funcionamiento de un CQL se puede encontrar al representar dos superficie rectangulares, cada una de ellas descrita mediante 4 puntos. Sobre esta información almacenada en una base de datos, se pueden realizar consultas como si las superficies comparten espacio, o a qué superficie pertenece un determinado punto.
Al tener almacenada la información mediante los puntos que definen la superficie, es necesario realizar una transformación de los datos a forma de restricción. Por ejemplo, la zona de intersección entra las superficies $(1,2)(3,2)(1,4)(3,4)$, y $(2,3)(6,3)(2,7)(6,7)$ vendrá representada por la restricción:

$$\{1 \leq x \leq 3 \wedge 2 \leq y \leq 4 \wedge 2 \leq x \leq 6 \wedge 3 \leq y \leq 7\}$$

4. Generación de CSP consultando a Bases de Datos Restrictivas

Como principales objetivos en nuestra propuesta, está unir la potencia de consulta de las bases de datos relacionales, y el almacenamiento de datos restrictivos. En principio, no será objetivo de nuestros estudios cómo se resolverán las restricciones, sino ofrecer un mecanismo de construcción dinámica de CSP en función de la información almacenada en una base de datos relacional. También es interesante destacar, la posibilidad que se ofrece de construir distintos CSP, en función del motor de inferencia que se encargará de su resolución, sin que ello intervenga en la arquitectura de la base de datos y de los datos allí almacenados.
Entre otra de las ventajas de la utilización de bases de datos relacionales, en lugar de ficheros de texto, se encuentra que tan sólo se construirá el CSP con los datos involucrados en nuestro problema no con toda la base de conocimiento. No obstante, las ventajas que ofrece la utilización de

bases de datos relacionales se reflejará con mayor claridad en 5.
Pese a ser la construcción de CSP la parte clave de la arquitectura que se propone, no hay que olvidar que no siempre será necesario construir y resolver un CSP, ya que las consultas donde no se involucran datos restrictivos, se pueden resolver exclusivamente con consultas SQL convencionales.

4.1. Formas de almacenar las Restricciones

Para poder tratar la información de una forma rápida y adecuada, es necesario que se busque la manera más correcta de almacenar los datos continuos, abordándolo desde distintas opciones para encontrar la mejor elección.

4.1.1. Representación Mediante Puntos

Para sistemas dinámicos, una de las técnicas que ha proliferado mucho en el tratamiento de datos continuos, radica en simulación el sistema y la discretización de los resultados obtenidos. Para tener una idea más tangible del tipo de conocimiento con el que se puede tratar, se propone el sistema dinámico que describe el comportamiento de un muelle:

$$\frac{d}{dt}v = -kx - \mu v$$

y donde los valor de k se encuentran en el rango $[k'..k'']$, por ejemplo [1,8 . . . 1,9].

Esta técnica se basa en la obtención de conocimiento mediante la simulación de los sistemas y discretizando la información continua, lo que transforma un problema de carácter infinito, en uno finito. La información obtenida tras la discretización será la almacenada en la base de datos, y de donde se inferirán futuros conocimientos [11]. Como se muestra la figura 1, se obtiene el comportamiento del sistema para un conjunto de los valores que puede tomar k, como por ejemplo 1.81, 1.82, 1.83...

Esta opción, llevada a cabo en trabajos previos [6, 11], tiene entre sus inconvenientes:

1. Pérdida de información al realizar la obligada discretización, con la consecuente poda de datos ante la imposibilidad de almacenar un número de valores infinitos.

2. Saturación innecesaria de la base de conocimiento, que pasará a contener gran número de registros, cuya cantidad será proporcional a la exactitud con la que se pretenda representar el sistema simulado.

3. Necesidad de decidir la cantidad de datos a almacenar, lo que necesitará de un estudio anterior a la simulación del sistema, para valorar el nivel entre veracidad de la información y velocidad de acceso, lo que también provocará pérdida en tiempo y recursos.

4. Pérdida de simbología, con la consecuente dificultad en el tratamiento y aprendizaje del conocimiento.

4.1.2. Representación de los Términos del Sistema

La segunda opción se encuentra almacenando las variables que sean necesarias para conocer el comportamiento del sistema. Por ejemplo, en el caso de almacenar una restricción polinómica, se guardarán los distintos términos que lo constituyen:

x	*Indep*	Relación
3	-2	$>$
...	...	...

$\equiv$ 3x - 2 > y

O para mayor número de términos:

x^2	x	*Indep*	Relación
3	0	5	$<$
...	...	...	...

$\equiv \quad 3x^2 + 5 < y$

Esta propuesta ofrece importantes ventajas, ya que permite almacenar información continua con poco gasto de espacio y no perder información en la discretización. También entre sus ventajas se encuentra, el tener una manera fácil de realizar modificaciones en los datos ya almacenados. Por ejemplo, cambiar todo el dominio que representa una restricción C por su complementario C, significa cambiar tan sólo una desigualdad, lo que

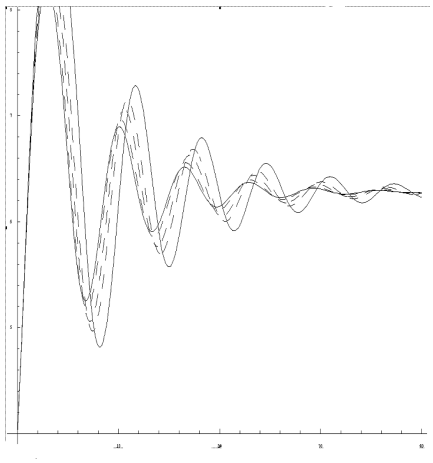

Figura 1: Ejemplo de simulación de sistemas dinámicos

implica modificar un solo registro. De esta manera, pasar de representar un determinado área a su opuesta, será posible con tan sólo cambiar $\leq$ *por* $>$.

Otra de las ventajas, que mejora la primera de las soluciones (discretización de datos continuos), es la no necesidad de un estudio previo del sistema para decidir cómo se va a realizar la poda de información, ya que dicha poda no es necesaria cuando almacenamos la restricción en si en lugar de datos discretos.
Pese a las comentadas ventajas, también presenta el inconveniente de no permitir incorporar fácilmente el comportamiento de características heterogéneas, debido a la pérdida de simbología. Cada vez que se pretenda incluir en la base de datos un comportamiento con distinto número de parámetros, se deberá crear una nueva tabla con los campos necesarios, provocando ante un sistema cambiante, constantes modificaciones en la arquitectura de la base de datos.

4.1.3. Almacenamiento de las Restricciones como Cadenas:

Ante las estudiadas desventajas que presentan sistemas convencionales, en este trabajo se propone una tercera solución, donde se abordan las Bases de Datos Restrictivas, almacenando las restricciones sin ningún tipo de formato, tan sólo como un campo más. Esta solución aprovecha lo aportado por la segunda solución (Representación de los términos), pero mejorando sus deficiencias. Con esta nueva forma, es posible almacenar información continua en muy poco espacio, evitando también la simulación y el pretratamiento de datos, junto a una fácil modificación de la información, ya que tan sólo es necesario modificar un registro. A su vez, resuelve el problema que presentaba la segunda solución, ante la incorporación de sistemas con distinto número de parámetros, ya que en esta propuesta la restricción se guarda como una cadena, sin estudiar en principio las partes de la sentencia.
Esta aportación, ofrece tratar las Bases de Datos Restrictivas a un alto nivel de abstracción, siendo posible almacenar en una base de datos relacional cualquier información, donde algunos de sus registros tienen carácter continuo, sin que ésto intervenga en la arquitectura de la base de datos. Ésta situación originó la necesidad de la búsqueda de una nueva arquitectura que aprovechara tanto la potencia de las bases de datos relacionales y las consultas, como los avances en el campo de Satisfacción de Restricciones.
La aportación que se describe a continuación, propone abordar el razonamiento mediante bases de datos restrictivas almacenando las restricciones sin ningún tipo de formato, en una base de datos relacional convencional, siendo posible representar el conocimiento de una manera simbólica evitando así el pretratamiento de datos, junto a la facilidad la modificación de la información y el tratamiento cognoscitivo. Esta aportación, ofrece tratar las bases de datos con restricciones a un alto nivel de abstracción, ya es posible almacenar en una base de datos relacional cualquier información, donde algunos de sus registros tienen carácter continuo, sean o no lineales, y sin que ésto intervenga en la arquitectura de la base de datos ni al tratamiento del conocimiento.
Basándose en esta idea, se desarrolla una arquitectura en el punto 4.3, que permite aprovechar al máximo las características de las Bases de Datos

Restrictivas.

4.2. Modelo de Datos

Para la resolución del problema se plantea el almacenamiento de las restricciones en forma de cadena, dentro de un determinado campo. Para obtener las restricciones, sólo se tendrá que acceder al registro adecuado. La gramática del Modelo de Datos que se utilizará para representar las restricciones (c) que pueden aparecer en las tuplas, será de la forma:

$$
\begin{array}{rcl@{\qquad}rcl}
c & : & expr = expr & expr & : & constante \\
 & | & expr \geq expr & & | & Variable \\
 & | & expr \leq expr & & | & f(expr) \\
 & | & expr > expr & & | & expr + expr \\
 & | & expr < expr & & | & expr - expr \\
 & | & si(c_1) \Rightarrow c_2 & & | & expr * expr \\
 & | & c_1 \vee c_2 & & | & expr/expr \\
 & | & c_1 \wedge c_2 & & | & \frac{d}{dt} Variable \\
 & | & \neg c_1 & & &
\end{array}
$$

Las consultas que se pueden realizar a una Base de Datos Restrictiva, pueden ser de la misma forma que a cualquier base de datos relacional, sólo que hay que tener en cuenta que se puede tanto obtener, como modificar restricciones, almacenadas en forma de cadena, además de cualquier tipo de datos que soporte la base de datos relacional en concreto.

4.3. Arquitectura del Sistema

Una vez encontrada una mejor forma de almacenar las restricciones para que el tratamiento y modificación del conocimiento sea lo más versátil posible, es necesario abordar la manera de extraer la información para adquirir la mayor potencia, tanto de la base de datos en sí, como de las restricciones almacenadas en ella. Para ello, se propone la arquitectura de la figura 2. Esta arquitectura propone la consulta a una base de datos, con datos restrictivos, para la construcción de problemas de satisfacción de restricciones. La construcción de los problemas se realizarán mediante las restricciones almacenadas en la base de datos, obtenidas mediante consultas, y *Restricciones Externas* que son datos restrictivos introducidos por el usuario que ayudan a la construcción del CSP final.

Las partes de la arquitectura se dividen principalmente en los módulos:

Interfaz de Usuario: Interfaz gráfico que facilita la relación de un usuario no experto en restricciones y en lenguajes de consulta, con el sistema. Gracias a un entorno mediante el cual puede definir sus necesidades, y marque al sistema cómo tiene que construir el CSP e introducir *Restricciones Externas*.
Construcción de Consultas: Una vez definidas las necesidades del usuario, el sistema construye las consultas SQL necesarias para obtener de la bases de datos la información concreta, tanto restricciones como datos discretos.
Bases de Datos Restrictivas: Implementada con cualquier tipo de base de datos relacional, que será seleccionada en función de las necesidades del sistema, como puede ser la velocidad de acceso, paralelismo de consultas, soporte de gran cantidad de registros,distribución de la información...
Obtención del Conocimiento: En el módulo de la obtención de conocimiento se decide si es necesario construir un CSP o ya se tiene toda la información necesaria. En el caso de que los datos sean restrictivos y haya que obtener más información, será necesario construir un CSP para que sea resulto por el motor de inferencia, en caso contrario, la información se mostrará por la interfaz de usuario.
Construcción del CSP: Mediante la información conseguida del módulo de *Obtención de Conocimiento* y las *Restricciones Externas*, se construye el o los **CSP**. De esta forma, se ayuda a enriquecer más aún la implementación del problema, y es posible minimizar el tiempo de búsqueda de información deductiva por parte del motor de inferencia, pudiendo hacer tratamiento del conocimiento de diversos tipos para la satisfacción de restricciones:

1. Sin que sufran ningún tipo de modificación, insertándolas directamente en el modelo.
2. Combinándolas con otras restricciones (de la propia base de datos o externas) mediante operaciones del álgebra Booleana como AND, OR, NOT... Al tener almacenadas las restricciones directamente, resultará mucho más fácil combinar distintas restricciones e inferir nuevos conocimientos, que si la información estuviera discretizada.

Motores de Inferencia: Son los encargados que resolver propiamente los CSP, pudiendo dar como

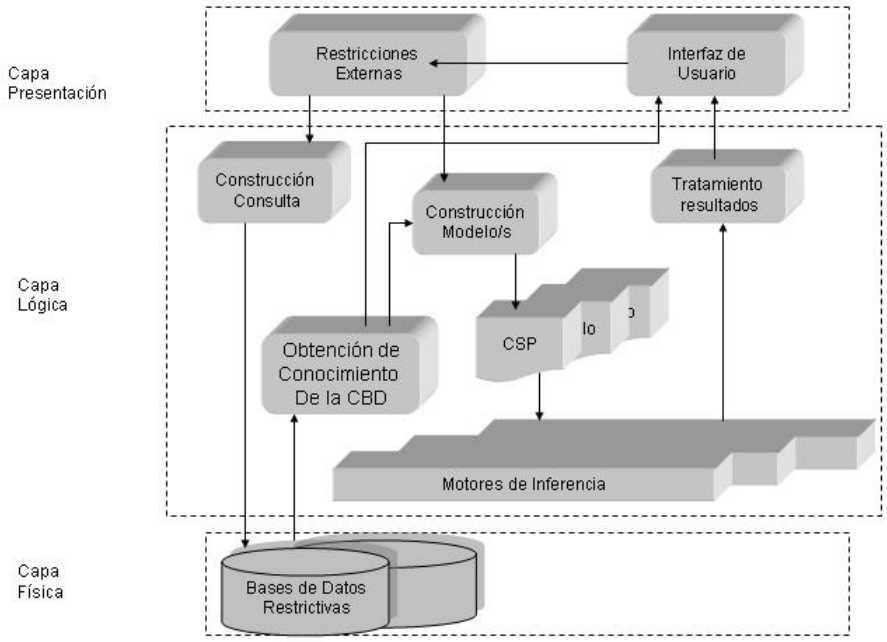

Figura 2: Arquitectura del sistema

solución valores concretos u otra restricción.
Tratamiento de Resultados: Procedimiento mediante el cual se formatean los datos para que sean presentados al usuario mediante el Interfaz.

Con diferentes formas de construir el CSP, es posible inferir distintos tipos de conocimientos, aunque en todos los casos se parta de la misma base de conocimiento. De esta manera, es posible construir CSP para resolver problemas de la forma:

1. Número de CSP que cumplan una serie de características.
2. Obtener los CSP que cumplan...
3. Existe algún CSP que cumpla...
4. Combinación de restricciones con operaciones del álgebra Booleana como: $\vee, \wedge, \neg, \Rightarrow$ ...

4.4. Mejoras Obtenidas

Entre las mejoras más significativas que aporta la solución que se propone están:

1. Tratamiento de sistemas cuyo comportamiento cambia en función del tiempo (sistemas dinámicos) evitando la discretización del conocimiento, como se hacía en trabajos anteriores [6].
2. Construcción de Modelos en tiempo de ejecución, donde las restricciones sólo tengan que ser almacenadas una vez. Esta característica es muy importante en sistemas con amplio campo cognoscitivo y que cambien frecuentemente. Además cabe destacar que gracias a estar almacenadas en una base de datos relacional, podemos tener acceso a la información de forma dinámica mediante selección sobre la base de conocimiento.
3. Facilidad al ampliar el campo cognoscitivo, ya que no hay que cambiar la arquitectura de las base de datos, y ampliar las restricciones ya almacenadas con la gramática propuesta en la sección 4.2, dado que la combinación de restricciones genera una nueva restricción.
4. Al almacenar las restricciones directamente y no los puntos obtenidos en una discretización, es mucho más fácil realizar comparaciones entre restricciones.

5. Incorporar datos no puramente restrictivos que ayuden a la inferencia de nuevos conocimientos.

6. Al proponer una arquitectura modularizada en tres capas, se puede intercambiar elementos sin que afecte a todo el sistema

7. Reutilización de Bases de Datos relacionales ya existente, teniendo sólo que incorporar más información pero sin reestructurarla por completo.

8. Gran flexibilidad del sistema, al ser posible utilizar distintas bases de datos y motores de inferencia haciendo pocas modificaciones en el sistema, y reutilizando información.

9. La posibilidad de utilizar distintos tipos de restricciones, no exclusivamente lineales, como ocurre en la mayoría de los sistema desarrollados hasta el momento.

4.5. Inconvenientes

Pese a las ventajas, también hay inconvenientes a mejorar que se deben tener en cuenta:

1. la vez que la capa de presentación ayuda a la interacción de usuarios no expertos con el sistema, reduce las posibilidades de la funcionalidad. Esto significa que el usuario no puede realizar consultas no reflejadas anteriormente por el sistema.

2. Tratamiento de las restricciones obtenidas de la consulta a la Base de Datos Restrictiva, para que sea código correcto y entendible por el motor de inferencia encargado de la resolución del CSP.

4.6. Implementación

La implementación de la arquitectura se realiza en tres partes:
Capa Física: las bases de datos Restrictivas se pueden construir con cualquier tipo de base de datos relacional, la selección de una u otra dependerá de las necesidades de capacidad de almacenamiento, velocidad de acceso, distribución de la información...
Capa Lógica: la implementación de la capa lógica está realizada en Java, para facilitar la conexión a las bases de datos restrictivas de forma trasparente mediante JDBC, y las consultas con SQL estándar. Gracias a la realización de las consultas se obtiene la información para construir el o los modelos [9], y desde el propio código Java se recurre a la ejecución de la herramienta OPL Studio [8], como motor de inferencia.
Capa Presentación: también está implementada en Java, pero se podría utilizar cualquier otro lenguaje, ya que tan sólo se encarga de facilitar la relación del usuario con el sistema.

5. Ejemplo Exploratorio

Para entender todos los conceptos y las razones que han incentivado a la creación de la arquitectura anteriormente descrita, se propone un problema donde se construye un sistema de almacenamiento como el que se refleja en el diagrama de entidad-relación de la figura 3. La base de datos relacional donde se almacena la información, refleja una estructura que contiene un conjunto de parcelas, las cuales tienen 1 o varios propietarios, a la vez que un propietario lo puede ser de más de una parcela. A su vez, las parcelas pertenecen a un municipio, y cada municipio a una provincia. Un campo de especial interés es la *Situación* de una *Parcela*, que refleja el lugar geográfico de dicha parcela como una restricción. Otro de los campos que representa la situación geográfica es la *Localización*, tanto en la tabla *Municipios*, como en *Provincias*.

Gracias a utilizar Bases de Datos Restricticas y a la arquitectuta descrita en 4.3 es posible:

1. Almacenar en la base de datos relacional tantas *Parcelas*, *Municipios*, *Propietarios* y *Provincias* necesarias, aprovechando la potencia de acceso a datos del álgebra relacional.

2. Desarrollar un interfaz de usuario donde sea posible introducir nuevas restricciones, lo que ha sido denominado como *Restricciones Externas*.

3. Crear un modelo de datos con las restricciones obtenidas de las base de datos y las Externas para mejorar el tratamiento de la información por el motor de Inferencia.

4. Presentar los resultados.

Un ejemplo de la tabla *Parcelas* podría ser la descrita en la tabla 2.

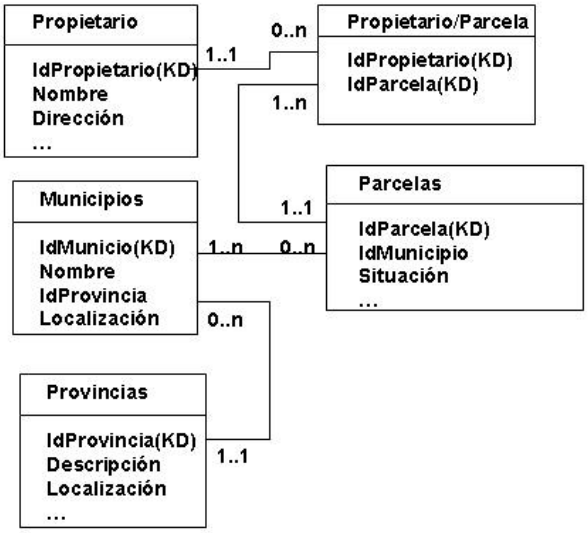

Figura 3: Diagrama de Entidad Relación (KD): Clave primaria

IdParcela	IdMunicipio	Situación
100	51	$y \leq x+2 \wedge y \geq x \wedge y \geq -x+2 \wedge y \leq -x+6$
101	51	$y \leq x \wedge y \geq x-2 \wedge y \geq -x+2 \wedge y \leq -x+6$
102	43	$y \leq x-2 \wedge y = 0 \wedge y \leq -x+6$
103	78	$y \leq x-6 \wedge y = 0 \wedge y \leq -x+10$
104	78	$y \leq x-2 \wedge y \geq x-4 \wedge y \geq -x+6 \wedge y \leq -x+10$
105	55	$y \geq x-6 \wedge y \leq x-4 \wedge y \geq -x+6 \wedge y \leq -x+10$

Tabla 2: Ejemplo de Tabla de Parcelas.

5.1. Consultas a la Base de Datos

En este apartado se proponen algunos ejemplos de consultas que se pueden realizar sobre la base de datos.

Consulta 1: Comprobar si un determinado camino pasa por una Parcela

Supongamos que un camino está expresado por la restricción externa $x = y$, y se desea conocer si pasa por la parcela cuyo IdParcela es 100. En este caso será necesario realizar una consulta como:

SELECT SITUACIÓN FROM PARCELAS
WHERE IDPARCELAS=100;

De donde se obtendrá la restricción:

$$y \leq x+2 \wedge y \geq x \wedge y \geq -x+2 \wedge y \leq -x+6$$

Si al unir ambas restricciones en una sola, existe alguna solución para los valores de x e y, significará que el camino pasa por la parcela. Para el ejemplo, la nueva restricción será:

$$y \leq x+2 \wedge y \geq x \wedge y \geq -x+2 \wedge y \leq -x+6$$

$$\wedge \quad x = y$$

Consulta 2: Comprobar si un determinado camino pasa por un Municipio

Esta consulta sería una extensión de la Consulta 1, ya que para que un camino pase por un municipio, tendrá que pasar por alguna de las parcelas que lo constituyen. Por ejemplo, para un municipio cuyo IdMunicipio sea 78, obtendremos la *Situación* de todas sus parcelas con la consulta:

SELECT SITUACIÓN FROM PARCELAS
WHERE IDMUNICIPIO=78;

Para el ejemplo propuesto en la tabla 2, se obtendrán las parcelas cuyo IdParcelas son el 103 y el

104. La nueva restricción que se construiría para obtener los resultados deseados para el mismo camino($x = y$) será:

$$(y \leq x - 6 \wedge y = 0 \wedge y \leq -x + 10 \quad \wedge \quad x = y) \quad \vee$$

$$(y \leq x - 2 \wedge y \geq x - 4 \wedge y \geq -x + 6 \wedge y \leq -x + 10 \quad \wedge \quad x = y)$$

Si una de las dos partes de la restricción, separadas por $\vee$, da valores posibles para la x y la y, significará que el camino pasa por el municipio.

Consulta 3: Nombre de los Propietarios de las Parcelas por las que pasa un camino Para obtener los nombres de los propietarios, primero será necesario saber cuáles son las parcelas que son atravesadas por el camino. Obtendremos la *Situación* de todas las parcelas con la consulta:

```
SELECT IDPARCELA, SITUACIÓN FROM
PARCELAS;
```

Con cada una de las restricciones obtenidas se llevará a cabo la misma operación que en la Consulta 1, con lo que se sabrá si hay valores en común entre el camino y cada una de las parcelas. Una vez conocidas las parcelas, y mediante cada uno de los IdParcela obtenidos en la consulta, se obtendrán los nombres de los propietarios de cada una de las parcelas(Id):

```
SELECT PROPIETARIO.NOMBRE FROM
PROPIETARIO prop,
PROPIETARIO/PARCELAS pp, PARCELAS par
WHERE par.IDPARCELA=Id
AND pp.IDPARCELAS=par.IDPARCELAS AND
pp.IDPROPIETARIO=prop.IDPROPPIETARIO;
```

Una vez obtenidos los nombres, solamente quedará presentar la información mediante la interfaz de usuario.

Estas consultas, son algunos de los ejemplos de los tipos de información que se pueden obtener de una Base de Datos Restrictiva. Si los registros almacenados son ampliados o modificados, no cambiará de ninguna forma las consultas, sólo la información obtenida, y por lo tanto el CSP que se creará. La ventaja es que al construir el CSP de forma dinámica, en función de lo contenido en la base de datos, las modificaciones serán transparente al usuarios, no necesitando transformación del programa para reflejar la nueva situación.

6. Conclusiones y trabajos futuros

Tras analizar las soluciones que existían hasta el momento, se propone una nueva arquitectura para el tratamiento de información a partir del conocimiento almacenado en Bases de Datos Restrictivas. Las Bases de Datos Restrictivas, permiten trabajar con datos de carácter infinito, con la consecuente exactitud y optimización en la búsqueda de soluciones, todo ello gracias a que su forma de almacenamiento es más declarativa e intuitiva, por lo que provee de una fácil forma de modificar y deducir conocimiento a partir del almacenado en una base de datos relacional. Como mejora a las herramientas aquí descritas, se aporta la utilización de bases de datos relacionales, en lugar de fichero de texto con información en Datalog. Otra de las grandes ventajas de esta arquitectura es que utiliza SQL como lenguaje de consulta, sin necesidad de desarrollar un nuevo lenguaje, y de forma transparente al usuario. También es posible reutilizar bases de datos relacionales existentes, simplemente ampliando su contenido, sin tener que rehacer su arquitectura desde el principio.
Además de lo aquí expuesto, es destacable la posibilidad de utilizar los conocimientos y las herramientas ya desarrolladas tanto en el campo de la satisfacción de restricciones, como en el mundo de las bases de datos. Se ofrece una capa para unir todas las herramientas, delegando tanto las resoluciones de las restricciones como las consultas a las bases de datos a entornos ya desarrollados.
Como trabajos futuros se plantea un tratamiento más exhaustivo del conocimiento almacenado, ampliando el lenguaje de consulta, pudiendo deducir más información y buscando nuevas técnicas de optimización y seguridad en las consultas de las Bases de Datos Restrictivas.

7. Agradecimiento

Este trabajo ha sido parcialmente financiado por la Comisión Internacional de Ciencia y Tecnología (DPI2000-0666-C02-02).

Referencias

[1] G. Kuper, L. Libkin and J. Paredaes: Constraint Databases. ISBN:3-540-66151-4 Springer, 1998.

[2] R. M. Gasca, J. A. Ortega, and M. Toro: Structual Constraint-Based Modeling and Reasoning with Basic Configuration Cells. 7th International Conference, pp. 595-599. 2001.

[3] P. C. Kanellakis, G. M. Kuper and P. Z. Revesz: Constraint Query Languages. Symposium on Principles of Database Systems, pp. 299-313. 1990.

[4] R. Helm, K. Marriott and M. Odersky: Constraint-based query optimization for spatial databases. In Tenth ACM Symposium on the Principles of Database Systems, pages 181-191, Denver, CO, May 1991.

[5] P. Revesz: Introduction to Constraint Databases. ISBN:0-387-98729-0 Springer, 2001.

[6] J. A. Ortega, R. M. Gasca y M. Toro: Obtención de Patrones de Comportamiento de Modelos Semicualitativos. Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial, vol. 9 II/00, pag: 66-75, 2000

[7] J. A. Ortega, R. M. Gasca, M. Toro y J. Torres: A Semiquantitative approach to study semiqualitative systems Advances in Artificial Intelligence. IBERAMIA2002, Lecture Notes in Artificial Intelligence núm 2527 pag 303-312, 2002.

[8] Component Libraries. User's Manual. ILOG OPL Studio 3.6, April 2001.

[9] Reference Manual. ILOG OPL Studio 3.6, April 2001.

[10] S. Grumbach, P. Rigaux, M. Scholl, and L. Segoun. Spatial constraint database. In S. Cluet and R. Hull, editors,Proceedings of Database Programming Languages (DBPL'97), volume 1369 of Lecture Notes in Computer Science, pages 38-59. Springer-Verlag, 1998.

[11] P. J. Abad, A. J. Suárez, J. A. Ortega, and R. M. Gasca. Diagnosis en Fases Tempranas de Sistemas Dinámicos. Computación y Sistemas, volume 6 No.2, Revista Iberoamericana de Computación, pages 116-129. ISBN 1405-5546, Octubre 2002.

[12] P. Revesz, R. Chen, P. Kanjamala, Y. Li, Y. Liu and Y. Wang. The MLPQ/GIS Constraint Database System ACM SIGMOD Record,vol. 29, no. 2, 2000.

[13] P. Revesz. Datalog and Constraints Constraint Databases,G. Kuper et al. eds., Springer-Verlag, pp. 155-170, 2000.

[14] P. Revesz. The DISCO Constraint Database System Constraint Databases, G. Kuper et al. eds.Springer-Verlag, pp. 383-389, 2000.

[15] S. Grumbach, P. Rigaux, M. Scholl and L. Segoufin DEDALE, A Spatial Constraint Database Workshop on Database Programming Languages, pp. 38-59, 1997.

[16] A. Brodsky, V. E. Segal, J. Chen and P. A. Exarkhopoulo. The CCUBE Constraint Object-Oriented Database System SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3,USA, eds. ACM Press, pp. 577-579, 1999.

[17] M. Cisterna Métodos de optimización de consultas para el lenguaje SQL. Tesis Doctoral, Universidad de Santiago de Chile 2002.

[18] S. Grumbach, P. Rigaux, M. Scholl, L. Segoufin. The Design and Implementation of DEDALE. Technical Report, November 30, 1999