Querying a Polynomial Object-Relational Constraint Database in Model-Based Diagnosis

M.T. Gómez-López, R.M. Gasca, C. Del Valle, and F.T. de la Rosa

Departamento de Lenguajes y Sistemas Informáticos,
Escuela Técnica Superior de Ingeniería Informática, Universidad de Sevilla, Spain
{mayte, gasca, carmelo, ffrosat}@lsi.us.es

Abstract. Many papers related to Constraint Databases (CDBs) theories exist, including proposals that present frameworks for the treatment of constraints as a new data type. Our proposal presents a new way of storing and manipulating constraints as a usual data, and of making queries about the constraint variables derived from an Object-Relational Constraint Database (ORCDB). In this work, the constraints stored in an ORCDB are only polynomial equality constraints. The proposal is based on Gröbner bases, constraint consistency and constraint optimisation techniques. Most works in CDB use spatial-temporal data as a case study, however this work presents an emergent engineering domain, that of fault diagnosis.

1 Introduction

This work is based on the necessity of dicovering new ways of storing constraint information such as spatio-temporal, scientific, medical or engineering data. Current databases have limitations in storing constraint data, due to the finite size of the physical support. Very large databases have delays in retrieving and modifying information. This type of data makes it necessary to find another method to represent constraint data as discrete information.

The main objective of this paper is to present a way of storing and querying constraints and their variables. The constraints are stored as objects in an Object-Relational Constraint Database (ORCDB) using Oracle^{TM} 9.i. This function is indispensable for model-based diagnosis, due to the the lack of solutions creating equivalent systems which depend on the known variables. This paper proposes a solution to the problem of obtaining the constraints of a system, by means of asking about their variables.

In order to obtain new constraints inferred from an ORCDB, four different techniques are used: symbolic techniques, based on Gröbner Bases; constraint optimisation techniques; constraint consistency; and a combination of symbolic and constraint consistency techniques.

Constraint Databases (CDBs) have been specially used in the treatment of spatial-temporal data, however this work demonstrates that other engineering areas also can benefit from using CDBs. For this reason an emergent engineering domain is used, that of fault diagnosis.

This work is organised as follows: Section 2 analyses other previous works. Section 3 presents model-based diagnosis as a case of study. Section 4 analyses the most important techniques to develop the architecture. Section 5 shows the architecture and its most important modules. Sections 6 and 7 present the syntax and functionality for the creation of an ORCDB, insertion of new records into tables and for querying an ORCDB. Finally, some conclusions and future work are presented.

2 Background

Constraint Databases began their development in 1990 with the paper of Kuper, Kanellakis and Revesz [1], and grew out of the research on Datalog [2] and Constraint Logic Programming (CLP).

Many database applications have to deal with infinite concepts such as time and space. However, databases have a finite capacity. The basic idea is that constraints can be used to represent, in a compact way, data that could be very large, or even infinite.

There are other methods for implementing and building prototypes for CDBs, whose main objective is handling spatial-temporal data. The most important approaches are analysed:

- **MLPQ/PReSTO:** This proposal [3] presents a combination of MLPQ (Management of Linear Programming Queries) and PReSTO (Parametric Rectangle Spatio Temporal Object). MLPQ is a system for the management and linear programming query in CDBs. It allows Datalog queries and the addition of operators over linear functions. PREsTO facilitates the performance of relational algebra querying systems that change over time. Although both present similar SQL syntax, they actually use a plane file to store the information and a Datalog query transformation process.
- **DEDALE** [4] is one of the first implementations of CDBs based on linear constraint models. DEDALE provides a language to query CDBs, which allows information to be obtained and uses a graphical interface to show the results. In order to represent the constraints, DEDALE uses the object-oriented paradigm, a more appropriate way to represent complex data. In this approach all the information is stored as objects. The type of data used in DEDALE is the spatial data model and a special module is given for spatial queries.
- **CCUBE:** (Constraint Object-Oriented Database System) [5] is a constraint object-oriented database system. The CCUBE system is designed to be used for the implementation and optimisation of high-level constraint object-oriented query languages. The CCUBE data manipulation language (Constraint Comprehension Calculus) is an integration of constraint calculus for extensible constraint domains within monoid comprehension. CCUBE gives an optimisation-level language for object-oriented queries. The data model for the constraint calculus is based on constraint spatio-temporal (CST)

objects. CCUBE guarantees polynomial time data complexity whose implementation uses the linear programming package CPLEX.

3 Diagnosis: A Motivating Example

Fault detection and identification of faulty components are very important company strategies, due to the economic demand and environment conservation required to remain in competitive markets. Diagnosis allows us to determine why a correctly designed system does not work as is expected and is based here on the monitorization of a system using DX [6] approach [7]. These papers were proposed to identify the discrepancies between the observed and correct behaviour of systems.

In engineering applications the storage of these data and query processing are often overlooked. Other works such as [8] have improved the efficiency in some phases of the model-based diagnosis with CDBs.

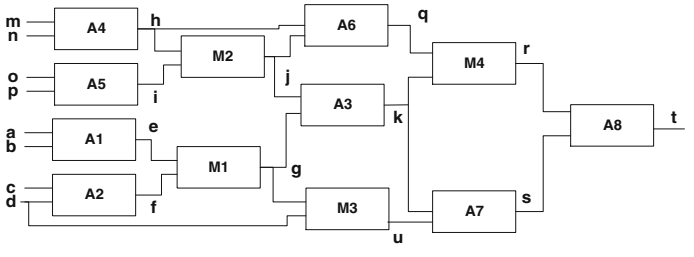

Fig. 1. Diagnosis Example

In this work, a new approach is proposed for querying ORCDBs in order to obtain equivalent systems which can be diagnosed such as the system shown in Figure 1. The example presents a group of components, multipliers (Mi) and adders(Ai), which work together. The use of SQL over constraints makes it possible to obtain several models. These models are compared to the real values in order to perform the diagnosis of the system. The location of sensors defines which variables are observable. Depending on the query, it is possible to know whether a part or the full system works correctly. It is also possible to obtain a group of equivalent constraints by replacing the non-observable variables.

4 Computational Techniques

In order to develop our architecture, four different tools are used. The first tool is the symbolic technique of Gröbner Bases, the second is the use of constraint consistency technique, the third is the constraint optimisation technique, and the last tool is a combination of symbolic and constraint consistency techniques.

4.1 Gröbner Bases

Gröbner bases theory [9] is the origin of many symbolic algorithms used to manipulate multiple variable polynomials. It is a generalisation of Gauss's elimination of multivariable lineal equations and of the Euclides algorithm for one-variable polynomial equations. Gröbner bases have better computational properties than the original system.

Gröbner bases transform a set of polynomial constraints into a standard form. By having the set of equality polynomial constraints in the form $P = 0$, Gröbner bases produce an equivalent system $G = 0$ which has the same solutions as the original.

For our work, there is a function called GröbnerBasis, which calculates Gröbner bases by means of a finite set of polynomial equations and a set of output variables, and those variables to be eliminated.

The signature of GröbnerBasis function is:

```
GröbnerBasis({Polynomials},{Output Variables},{Unwanted Variables})
```

4.2 Constraint Consistency and Constraint Optimisation Techniques

The previous problems of engineering can be modelled as Constraint Satisfaction Problems (CSP) [10]. A CSP consists of a finite set of variables, a domain of values for each variable and a set of constraints that restrict the combinations of values of the variables. The aim in a CSP is to determine a value for each variable so that all constraints in the problem are satisfied. Usually, a combination of search with consistency techniques is used to solve these problems. The consistency techniques remove inconsistent values from the domains of the variables during the search. Several local consistency and optimisation techniques have been proposed as ways of improving the efficiency of search algorithms.

The consistency techniques are used as a process to obtain the values of the unknown variables from the known variables, by avoiding the use of symbolic techniques that usually have a higher computational complexity. For the Constraint Consistency techniques (Subsection 7.2), the search is not necessary because the domain of the known variables in the queries has just one value. But constraint optimisation techniques (Subsection 7.3) can have several correct values, so it is necessary to define an objective and the search is necessary. Our proposal takes advantages of all these techniques and dynamically builds CSPs depending on the query. In this work the domain of the integer is the only type used.

5 The Architecture

The main objective of our work is to add an interface to make the use of constraints transparent, by handling *Constraint Type* as a usual type of data. To store constraints, the semantics of SQL has been modified, by changing as less as possible the syntax of the queries. Figure 2 shows the architecture of the system.

The interaction between the user and the system is through the interface CROQL (Constraint Relational Object Query Language). The *Lexical and Syntactic Analysis* module verifies that the query is correct. The *SQL Transformation* module obtains the necessary information to perform the query. The *TypeOfQuery* module decides which technique is necessary, in order to return the solution to the user. Depending on the query, one of the four modules (optimisation, consistency, symbolic or consistency/symbolic technique) is used. It is also possible to query the ORCBD without any modification.

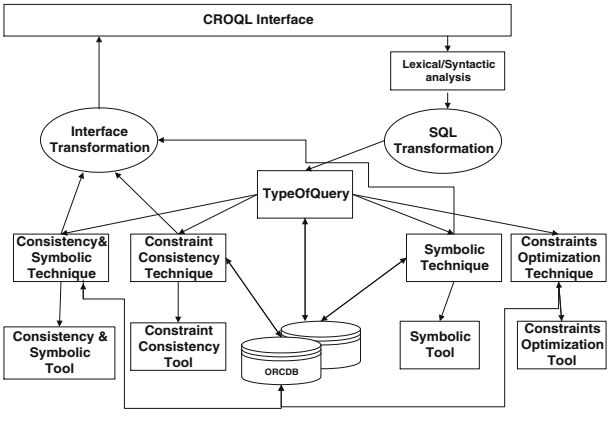

Fig. 2. Architecture of the system

The user of ORCDBs can ask, in a very easy way, about usual types, constraints or variables related to the constraints. The constraints are stored as objects indexed by the variables which contain them, in order to improve the execution time for obtaining the constraints related to some variables. In an ORCDB it is possible to store the same information because all the information is stored in a relational database. However when the user asks about information related to constraints it is necessary to develop some transformations. In order to clarify when it is necessary to use each part of the architecture, the example shown in Figure 1 is used.

6 Creating an ORCDB and Inserting Information

In this section, it is shown how it is possible to create and fill an ORCDB. Some implementation decisions have been accepted to improve the computational time in the queries, and to make the information versatile. One of the most important advantages of our proposal is to make the utilisation of constraints transparent to the user, therefore a very similar syntax of SQL is kept in CROQL.

6.1 Creating an ORCDB

In order to create an ORCDB the following sentence is used:

```
CREATE CONSTRAINTDATABASE <database_name>
```

In our model, when an ORCDB is created, the tables shown in Figure 3 are created too, in order to improve the computational time for obtaining the constraints related to some variables. These tables allow the identification of each constraint (table *Constraint*), each variable (table *Variable*) and to establish the relations between the constraints and the variables (*Constraint/Variable*), thereby avoiding the study of all constraints.

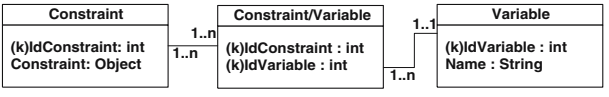

Fig. 3. Tables to index constraints and variables

6.2 Creating a Table in an ORCDB

As our proposal tries to modify SQL syntax as little as possible, the syntax is not modified at all to create a table. The unique change is that it is possible to create constraint fields, where $< \text{field\_type}_i >$ is *Constraint Type*. In our case, the possible type of constraints is polynomial equality constraints.

For the example shown in Figure 1, the sentence is:

```
CREATE TABLE Component (IdComponent Integer, Name String,
Behaviour Constraint)
```

6.3 Inserting New Information into the Tables

If a field of a table has been created as *Constraint Type*, it is possible to add constraint information. Our proposal adds the option to of handling the constraint as a usual type, such as *Integer*, *String*, *Date* ...

Therefore, if the user tries to add information of an incorrect type to a field, an error will be produced as in a relational database. The checking of the type works as for usual types.

When constraint data is added, indexes are created in the table *Constraint* in order to locate constraints more quickly and efficiently.

The users cannot see the indexes, and these are just used to speed up the queries. These indexes are necessary when constraints are added, and they are created and stored in an implicit way.

In order to store the example shown in Figure 1 in an ORCDB, one query could be:

```
INSERT INTO Component (IdComponent, Name, Behaviour)
VALUES (101, A1, "a + b = e" )
```

The ORCDB after inserting some components is shown in Figure 4.

Component		
Id	Name	Behaviour
1	A1	1
2	A2	2
3	M1	3
4	M2	4
5	A3	5
6	M3	6
...	...	...

Constraint	
Id	ConstraintObject
1	a+b=e
2	c+d=f
3	e*f=g
4	h*i=j
5	h+g=k
6	d+g=u
...	...

Variable	
Id	Name
1	a
2	b
3	e
4	d
5	c
6	f
...	...

Constraint Variable	
IdCo	IdVar
1	1
1	2
1	3
2	5
2	4
...	...

Fig. 4. Example of tables with stored constraints

7 Querying an ORCDB

This Section describes which part of the system is used depending on the query.

With our methodology, it is possible to query constraint variables, it means querying <**field_name**> or <**field_name**>**.VariableName**, where <**field_name**> is a *Constraint Type*.

Depending on the query, different parts of the architecture are used:

- If the query involves usual types or constraints stored directly in the ORCDB, but the query does not involve variables, the query is not transformed.
- If the query involves variables from constraints stored in the ORCDB and none one of the variables are instantiated, a symbolic tool is used, as explained in Subsection 7.1.
- If the query involves variables of constraints stored in the ORCDB, where the objective is defined in the query, a optimisation tool is used. This is explained in Subsection 7.3.
- If the query involves constraint variables stored in an ORCDB and some variables are instantiated, there are two possibilities: using consistency techniques, if all the variables are instantiated (Subsection 7.2); or using a combination of symbolic and consistency techniques, if only some variables are instantiated (Subsection 7.4).

7.1 Symbolic Techniques

This part of the architecture is used when none of the variables are instantiated in the query. For this reason, those variables which do not appear in the query but are related to constraints which contain the variables of the query, must be replaced by variables of the query.

An example of a query solved using the symbolic technique is:

```
SELECT Component.Behaviour.a, Component.Behaviour.b,
Component.Behaviour.c, Component.Behaviour.d,
Component.Behaviour.u FROM Component
```

The idea of this process is to determine each *group of related constraints*, which is defined as:

G is a group of related constraints if
$G \equiv \bigcup_i \{c_i\} \mid \forall\ c_i\ \text{VarNoQuery}(c_i) \subseteq \text{VarNoQuery}(G - c_i)$

where c_i is a constraint and VarNoQuery(C) are the variables of the constraints C that do not appear in the query

For the example of Figure 1, the information shown below represents the constraints related to the query, and the variables that do not appear in the query in this form {Component, {VarNoQuery(Component) }}. The actual information is the indexes of constraints and variables, however to show the idea clearly, the names of components and variables are used. The information for our example is: {A1,{e}},{A2,{f}}, {M3,{g}},{A7,{s,k}},{M1,{e,f,g}},{A3,{g,j,k}},{M4,{k,q,r}}

In this case, A1 participates in a *group of related constraints* if e is in another constraint, such as M1. M1 participates in a *group of related constraints* if f and g are in another constraint, such as A2 and M3 respectively. This means that {A1, A2, M3, M1} form a *group of related constraints*. The rest of the constraints do not participate in a *group of related constraints*

In order to use the Gröbner bases, Mathematica^{TM} $v.5$ is used. For the example, the call to this function is :

```
GroebnerBasis[{a+b-e, c+d-f, e*f-g, g*d-u}, {a, b, c, d, u}, {e, f, g}]
```

And the result is: {a*c*d + b*c*d + a*d^2 + b*d^2 - u = 0}

7.2 The Constraint Consistency Tool

When all the variables in the query are instantiated, it is possible to use the module of constraint consistency technique. In this case, all the *groups of related constraints* are instantiated. A constraint is instantiated if it has only one unknown variable which means that all *VarNoQuerys* can be instantiated.

An example of this type of query is:

```
SELECT Component.Behaviour.u FROM Component
WHERE Component.Behaviour.a=1 AND Component.Behaviour.b=3
AND Component.Behaviour.c=2 AND Component.Behaviour.d=1
```

Once it is known that this query generates instantiated constraints, a Constraint Satisfaction Problem (CSP) is created in order to infer the value of u. This CSP is created dynamically using the constraints obtained from the ORCDB, the *VarNoQuerys* and the instantiated variables. Figure 5.a shows the CSP for the example.

The result is u= 12. All *VarNoQuery* variables are instantiated, but only u is presented as the solution. The OPL Studio^{TM} [11] is used in order to instantiate the variables.

7.3 The Constraint Optimisation Tool

When a query has several solutions and the user wants to select from among the possible options, the module of optimisation is used. An example of a query that uses this module is:

```
SELECT MIN(Component.Behaviour.u) FROM Component
WHERE Component.Behaviour.a>=3 AND Component.Behaviour.b>= 1
AND Component.Behaviour.b< 5 AND Component.Behaviour.c>2
AND Component.Behaviour.d<=3 AND Component.Behaviour.d>=0
```

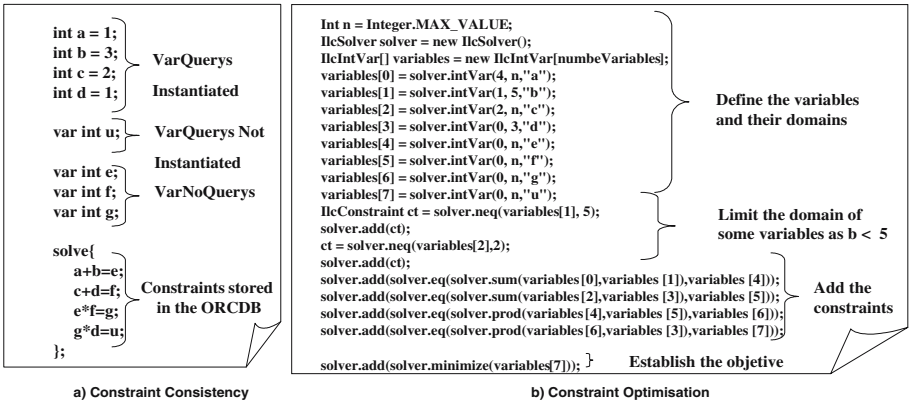

Fig. 5. Examples of CSP

Once it is known that this query generates instantiated constraints, a Constraint Satisfaction Problem (CSP) is created in order to infer the value of u. This CSP is dynamically created by using the constraints obtained from the ORCDB, *VarNoQuery* and the instantiated variables. Figure 5.b shows the CSP associated. JSolver^{TM} [12] is used in order to optimise the constraints.

7.4 The Constraint Consistency and Symbolic Tool

This module of the architecture is used when only some variables are instantiated in the query. In this case, a mixed tool is necessary, in order to propagate and to replace the variables in a symbolic way. This function also is from Mathematica^{TM} v.5. An example of this type of query can be:

```
SELECT Component.Behaviour.u FROM Component
WHERE Component.Behaviour.a=1 AND Component.Behaviour.d=2
```

In order to obtain the value of u, the system uses the syntax:

`Solve[{Constraints Related to the query},{Out Variables},{VarNoQuery}]`

For the example, the call would be:

`Solve[{`$a+b==e, c+d==f, e*f==g, g*d==u$`}, {u}, {e, f}]`

And the result would be: `{u = 2 (2 + 2 b + c + b c)}`

8 Conclusions and Future Work

This work extends the semantics of SQL, in order to store constraint information as a new data type. The constraints data are indexed in order to improve the computational time. The interaction between the user and the system is transparent to the constraint handler. The system allows the use of polynomial equality constraints, by using four techniques to perform the queries: symbolic, constraint consistency, constraint optimisation and symbolic/consistency techniques.

As future work, we propose extending the domains of variables and constraints. We also suggest an extension to incorporate different types of constraints, not only polynomial equalities. As for as SQL sentences are concerned, it is necessary to offer all the possibilities of standard SQL, such as UPDATE, REMOVE and other types of queries.

Acknowledgements

This work has been partially funded by the Ministerio de Ciencia y Tecnología of Spain (DPI2003-07146-C02-01) and European Regional Development Fund. (ERDF/FEDER).

References

1. G. M. Kuper P. C. Kanellakis and P. Z. Revesz. Constraint query languages. *Symposium on Principles of Database Systems*, pages 299–313, 1990.
2. P. Revesz. Datalog and constraints. *Constraint Databases,G. Kuper et al. eds., Springer-Verlag*, pages 155–170, 2000.
3. P. Z. Revesz, R. Chen, P. Kanjamala, Y. Li, Y. Liu, and 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*, page 601. ACM, 2000.
4. Stéphane Grumbach, Philippe Rigaux, and Luc Segoufin. The dedale system for complex spatial queries. In *SIGMOD Conference*, pages 213–224, 1998.
5. A. Brodsky, V. E. Segal, J. Chen, and 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*, pages 577–579. ACM Press, 1999.
6. R. Davis. Diagnostic reasoning based on structure and behavior. *In Artificial Intelligence 24*, pages 347–410, 1984.
7. R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence 32*, 1:57–96, 1987.
8. M. T. Gómez-López, R. Ceballos, R. M. Gasca, and C. Del Valle. Applying constraint databases in the determination of potential minimal conflicts to polynomial model-based diagnosis. In *CDB*, pages 75–89, 2004.
9. B. Buchberger. Gröbner bases: An algorithmic method in polynomial ideal theory. *Multidimensional Systems Theory, N. K. Bose, ed.*, pages 184–232, 1985.
10. Albert Croker and Vasant Dhar. A knowledge representation for constraint satisfaction problems. *IEEE Trans. Knowl. Data Eng.*, 5(5):740–752, 1993.
11. Reference Manual. Ilog opl studio 3.6. April, 2001.
12. Reference Manual. Jsolver 2.1. April, 2003.