Using Reasoning of Description Logics for Query Processing in Multidatabase Systems

Alfredo Goñi and **Jesús Bermúdez** and **José M. Blanco** and **Arantza Illarramendi**
Facultad de Informática, Universidad del País Vasco. Spain.
e-mail: jipileca@si.ehu.es
fax: + 34 43 219306

Abstract. Nowadays, the interest to work simultaneously with data stored in several databases is growing. Multidatabase Systems (MDBS) have been proposed as a solution to work with different pre-existing autonomous databases. Federated Database Systems (FDBS) are a special type of MDBS where an integrated schema is provided. This integrated schema is the result of an integration process among the schemata of the pre-existing autonomous databases. In our case we have built a FDBS that integrates several heterogeneous relational databases by using a particular type of Knowledge Representation system based on Description Logics (DL system). The integrated schema is represented as a terminology formed by a set of classes and attributes. Although there has been a lot of research about the problems of translation and integration of schemata to obtain integrated ones, the problem of query processing against these integrated schemata has not been treated so much. In this paper we present an overview of the query processing of our FDBS and then we point out the features of DL systems used in the query processing.

1 INTRODUCTION

Nowadays, the interest to work simultaneously with data stored in several databases is growing. MultiDataBase Systems (MDBS) have been proposed as a solution to work with different pre-existing autonomous databases. Federated DataBase Systems (FDBS) are a special type of MDBS where an integrated schema is provided. This integrated schema is the result of an integration process among the schemata of the pre-existing autonomous databases. There exist many distinct approaches for building a FDBS, namely the Entity-Relationship model approach, the Object-Oriented approach, and the Knowledge Representation Systems (KRS) approach. In our case we have built a FDBS that integrates several heterogeneous relational databases by using a particular type of KRS based on Description Logics (DL system). The integrated schema, built upon the different relational database schemata, is viewed as a terminology, formed by a set of classes and attributes, and the extension of the terminology (the instances of the classes and attribute values) is in fact in the underlying databases. When a query is formulated over that terminology, the answer must be obtained from the different databases. In [Blanco *et al.*,1994a; 1994b] we show the advantages of using a DL system for building a Federated Relational Database System.

Three different types of problems are involved when building a FDBS: *translation* of the underlying database schemata into schemata expressed in a canonical model, *integration* of the translated schemata into an integrated schema and *query processing* of the user-formulated queries over the integrated schema by accessing the underlying databases. Although there has been a lot of research about the problems of translation and integration of schemata to obtain integrated ones ([Bertino *et al.*,1989; Larson *et al.*,1989; Navathe *et al.*,1989; Collet *et al.*,1991; Sull and Kashyap,1992; Spaccapietra *et al.*,1992; Qutaishat *et al.*,1992; Pitoura *et al.*,1995]), the problem of query processing against these integrated schemata has not been treated so much.

In general the query processing problem is a hot topic in many database research areas. In particular, query processing in distributed database systems has been studied in detail ([Ceri and Pelagatti,1984; Ozsu and Valduriez,1991]) but the solutions proposed are not the same as the solutions needed for multidatabase systems because of the autonomy and heterogeneity of the component databases ([Dayal,1985; Ozsu and Valduriez,1991; Schild,1991; Du *et al.*,1992; Lu *et al.*,1992; Du *et al.*,1995; Kim,1995]), and therefore more research is needed to solve all the problems that appear in the multidatabase systems. With this aim, we have defined a new query processing strategy.

Concerning the related works, we mention only those works that use a DL system in connection with information sources that may be in a multidatabase context or not. In [Devanbu,1993] Devanbu explains how translators from DL queries to database queries can be built. Borgida and Brachman [Borgida and Brachman,1993] present an efficient translator from DL querie s to SQL queries and also present some problems when loading data into the DL terminology. In the previous cases only one database is connected to the DL terminology and the whole DL terminology is loaded at the beginning of the session.

Therefore, user formulated queries are answered without accessing to the database using for that the query processing features provided by the DL system. In our case several databases are connected to the DL terminology base and only some of their data are loaded in the cache memory. In [Arens *et al.*,1993] Arens et al. show the SIMS system, that integrates data from several heterogeneous and distributed information sources (databases and LOOM knowledge bases) by using the LOOM DL system. From the query processing point of view they reformulate the queries by selecting the appropriate information sources, create plans to answer the query, perform semantic optimization of the plan by exploiting knowledge about the domain and the information sources and execute the optimized plan. They also point out in [Arens and Knoblock,1994] that during the query processing, there can exist classes that are worth caching, and give several principles that describe the interesting data to cache. We have stated some principles of caching and defined a cost model in order to evaluate the benefit and cost of having the objects in the cache memory, with which the optimal set of objects worth caching can be calculated. We also distinguish between two replacement strategies: the *static*, applied between sessions, and the *dynamic*, applied during the query processing inside a session. In [Levy *et al.*,1995], Levy et al. present an information system that englobes different information systems as databases, object-oriented knowledge bases and structured files. The unified view of the information sources is expressed by using the CLASSIC DL system. They have designed a query optimization strategy that minimizes the number of information sources accessed during the query processing, but they do not consider the response times for queries nor incorporate a cache memory to improve the efficiency.

The goal of this paper is to present an overview of the query processing in a FDBS built by using BACK [Peltason *et al.*,1989], a particular DL system, but first we show an example of an integrated terminology used throughout the paper.

2 EXAMPLE OF AN INTEGRATED TERMINOLOGY

Let us suppose that there are two very simple exported schemata, namely *db1* and *db2*, from two databases with information about teachers, students, courses and which teachers teach what courses and which students attend what courses.

Let us suppose that the integrated terminology obtained after the integration process and the mapping information that relates the classes and attributes of the terminology and the data elements in the database schemata appear respectively in figures 2 and 3. Notice that there is *replication of data* because the instances of the class *teaching_assistant* can be obtained from three different ways: a) accessing only to *db1*, b) accessing only to *db2* or c) accessing both *db1* and *db2* and doing the intersection. As it can be shown, the derived relations that appear in these mapping informations are multidatabase ERA expressions because they contain aggregate functions ($\mathcal{F}_{\text{count()}}$) and attributes and relations are from different databases (e.g. *db1.student, db2.teacher*).

db1
student(id,name,address) studies(s_id,course#)
db2
teacher(id,name,address,title,degree) course(c#,name,depart,creds) teaches(t_id,c#)

Figure 1: Simplified schemata

CLASSES
person :< *anything* *student* :< *person* *teacher* :< *person* *course* :< *anything* *teaching_assistant* := *teacher and student* *super_student* := *student and atleast(10,studies)* *studies: "A450"* *lucky_teacher* := *teacher and atmost(0,teaches)*
ATTRIBUTES (only some of them appear here)
name :< *domain(person) and range(string)* *title* :< *domain(teacher) and range(string)* *teaches* :< *domain(teacher) and range(course)* *teaches_to* :< *domain(teacher) and range(student)* *studies* :< *domain(student) and range(course)*

Figure 2: Integrated terminology

CLASS <attr,rel>
person: <id,db1.student $\bigcup$(id) db2.teacher>
student: <id,db1.student>
teacher: <id,db2.teacher>
teaching_assistant: <id,db1.student $\bigcap$(id) db2.teacher>
teaching_assistant: <id,σ cat="grad_stud" (db2.teacher)>
teaching_assistant: <id,σ course#$\geq$"A600" (db1.student)>
super_student: <s_id,σ new_attr$\geq$10 ((idp $\mathcal{F}$ count(course#) db1.studies $\bowtie$ σ course#="A450" (db1.studies))>
lucky_teacher: <s_id,db2.teacher $-$(id=t_id) db2.teaches>
ATTRIBUTE: <attr_inst,attr_attr,rel>
studies: <s_id,course#,db1.studies>
title: <id,title,db2.teacher>

Figure 3: Mapping Information

3 OVERVIEW OF THE QUERY PROCESSING

There are five stages in the Query Processing of our FDBS: parsing of the query, semantic transformation and decomposition, query processing in the underlying databases, loading of the answers brought from the databases in the cache memory and query processing in the cache memory. In figure 4, the architecture of the Query Processor is shown.

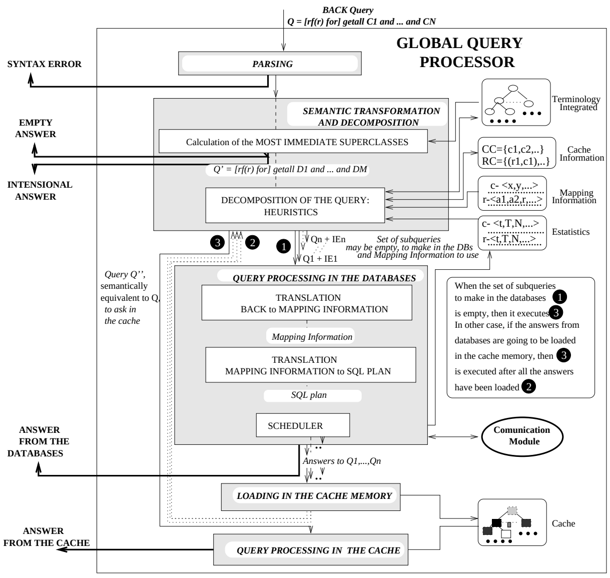

Figure 4: Architecture of the Query Processor.

Parsing

In this stage lexical and syntactical errors are detected. For example, *getall* **professors**, or *getall teacher* **adn** *student*. In our case this is achieved by using the parser of the DL system.

Semantic Transformation and Decomposition

In the semantic transformation and decomposition stage, different situations may happen: a) the query is detected as inconsistent and therefore the process ends, b) the query is answered from the cache memory but first, if needed some subqueries are asked in the underlying databases or c) the query is answered from the underlying databases and nothing is loaded in the cache memory.

There are two tasks performed in this stage: 1) calculation of the Most Immediate Superclasses in order to obtain a semantically equivalent query by using DL reasoning capabilities and 2) decomposition of that query by using a set of heuristics.

1. Set of Most Immediate Superclasses (MIS)

The set of MIS corresponding to a query is formed by all the classes and attribute restrictions that belong to the terminology or to the query that satisfy the following conditions: a) all of them subsume the class description of the query b) none of elements in the MIS set subsumes another element in MIS and c) if there is some class or some restriction that subsumes the class description of the query, then it belongs to the MIS set or there is already an element in MIS to which it subsumes. The query formed by the conjunction of all the elements in the MIS set is semantically equivalent to the class description of the query. The reasoning that DL systems perform to calculate subsumption relationships between descriptions is used in order to get the MIS set.

If the query is

getall teacher and person and student and atleast(20,studies) and atmost(0,teaches) and studies: "A450" and atmost(25,studies)

then the MIS set is

{*teaching_assistant, lucky_teacher, super_student, atleast(20,studies), atmost(25,studies)*}

By calculating the MIS set some more specific classes that do not appear in the initial query may be detected (in the example *teaching_assistant, lucky_teacher* and *super_student*) and other redundant classes and attribute restrictions are eliminated (in the example *person, atmost(0,teaches)* and *studies: "A450"*). In some cases, it is better to use this semantically equivalent query but this is not always true.

Furthermore, in this stage the initial query may be detected as inconsistent and also intensional answers may be given to the user.

It is detected that the query is inconsistent when *getall nothing* is a semantically equivalent query to the user query, that is, when *nothing* is in the set of Most Immediate Superclasses.

Two different types of intensional answers may be given to the user: *Most Specific Formulation* of the query and *Extended Formulation* of the query. The first one is formed by the elements of the MIS set corresponding to the query and the second one is formed by recursively substituting the classes in the initial query by their definitions.

2. Decomposition of the query: heuristics

In order to know if the initial query may be answered from the cache memory, the semantically equivalent query is used. The query is cached if all the classes in the set of MIS and all the attributes that appear in restrictions of the MIS are cached. When the query cannot be answered from the cache memory, then the semantically equivalent query has to be decomposed in a set of subqueries to ask in the underlying databases. Due to analyzing all the possible combinations of subqueries is very complex we have defined a set of heuristics that try to reduce such number of combinations. These heuristics take into account if some parts of the query are cached, domain information about the integrated terminology and statistics about queries previously formulated. The goals of these heuristics are:

- Goal 1: To avoid that the answer sent from each database node is too large.
- Goal 2: To try that the computation cost in each database node is small, unless it is needed to reach the goal 1.
- Goal 3: To try not to bring parts that are already cached, unless it is needed to reach the previous

goals.

- Goal 4: To try to send subqueries that are executed in parallel in different database nodes, if these subqueries satisfy the two first goals.

The heuristics are presented in the following table:

H.	Description
H1	Substitute a non-cached defined class without alternative mapping information by its most specific definition
H2	Maintain a non-cached defined class with alternative mapping
H3	Substitute a class by its only non-cached subclass
H4	Use an attribute to project if it is cached
H5	Reduce the size of a subquery within a database node
H6	Reduce the size of subqueries among database nodes
H7	Substitute restrictions over an attribute by its projection

We show some examples for the heuristics. Let us suppose that the MIS set is the previous one:

{*teaching_assistant, lucky_teacher, super_student, atleast(20,studies), atmost(25,studies)*}

Heuristic H1

If *super_student* is not cached then it is better to substitute it by its definition:

student and atleast(10,studies) and studies:"A450"

Then the redundant restriction *atleast(10,studies)* is eliminated because it subsumes *atleast(20,studies)*. The reason of why *super_student* is not interesting in this case is that the mapping information of *super_student* has been obtained from its definition, and it includes the mapping for *atleast(10,studies)*. Therefore it is more inefficient.

Heuristic H2

If *teaching_assistant* is not cached then it is not substituted by its definition *teacher and student* because it does have two alternative mapping informations:

$$<id, \sigma_{\text{cat="grad\_stud"}}\ (db2.teacher)>$$
$$<id, \sigma_{\text{course\#}\geq\text{"A600"}}\ (db1.student)>$$

each one in a different database that are better than the mapping information based on the definition *teacher and student*.

Heuristic H5

Let us suppose that, after having applied heuristics H1 and H2, the non-cached elements are

{*atmost(0,teaches), studies: "A450", teaching_assistant, atleast(20,studies), atmost(25,studies)*}

It can be noticed that *atmost(0,teaches)* is the only one that can only be brought from *db2*, and that *teaching_assistant* can be brought indistinctly from *db1* or from *db2*. Heuristic H5 says that it is better to send

Q1: getall teaching_assistant and atmost(0,teaches)

to *db2* instead of

getall atmost(0,teaches)

because the answer for this last one may be too big.

On the other hand, applying the same heuristic H5, it is decided that the query

Q2: getall teaching_assistant and atleast(20,studies) and atmost(25,studies) and studies: "A450"

has to be answered from *db2*.

Heuristic H6

If the size of the answer in some database node is big then it is decided to send queries that have to be answered from different databases. For example, if the answer to

Q1: getall teaching_assistant and atmost(0,teaches)

is considered to be too big then the previous two queries, Q1 and Q2, obtained after applying heuristic H5 are merged into only one. The query

Q3: getall teaching_assistant and atmost(0,teaches) and atleast(20,studies) and atmost(25,studies) and studies: "A450"

has to be answered from the underlying databases.

Query processing in the underlying databases

The set of subqueries that have been selected in the previous stage have to be asked in the underlying databases. For each DL subquery, its corresponding optimal mapping information is generated and then it is translated into an optimal SQL plan to be executed in the underlying databases. An SQL plan is a set of SQL queries to answer in the database nodes along with the communication operations sending intermediate results between the nodes.

Optimization of the mapping information

Taking into account that the mapping information is already defined for all the classes and attributes of the terminology, the mapping information for all the constructors: *atleast(n,attr), atmost(n,attr), all(attr,class_type), attr: value, attr: close(value)* and for combinations of classes has to be defined. This mapping can be optimized when some information about the underlying databases is known: *functional, inclusion and exclusion dependencies, ranges of values for attributes, information about null values*, and when some combinations of constructors happen.

For example, the mapping information for the query:

getall atleast(20,studies) and atmost(25,studies)

is the next one:

$$<s\_id, \sigma_{20\leq\text{new\_attr}\leq 25}\ ({}_{\text{s\_id}}\mathcal{F}_{\text{count(course\#)}}\ db1.studies)>$$

However, the mapping information for the query:

getall atleast(20,title) and atmost(25,title)

is ϕ, because the functional dependency `teacher.id` $\rightarrow$ `teacher.title` exists.

Generation of an optimal SQL plan

The previously generated mapping information has to be translated into an SQL plan. When a Multidatabase SQL is available then in the translation process it has to be avoided generating too many views. If that MSQL does not exist then every SQL query has to be sent to each corresponding database node and the optimal combination of intermediate results calculated.

The SQL plan corresponding to

$<s\_id, \sigma_{\ 2\leq \text{new\_attr} \leq 5}\ ({}_{\text{s\_id}}\mathcal{F}_{\text{count(course\#)}}\ db1.studies)>$

would be a simple SQL sentence to execute in *db1* node.

```
select s_id
from studies
group by s_id
having count(distinct course#)>= 20
and count(distinct course#)<= 25
```

Loading in the cache memory

If the query is going to be answered from the cache memory, then the answers to the queries are loaded in the corresponding classes and attributes of the terminology[1]. In order to avoid that the DL system classifies the instances a solution like the presented in [Borgida and Brachman,1993] may be applied.

Query processing in the cache memory

In the last stage it is possible to answer from the cache memory, once all the subqueries have been made in the underlying databases and their answers loaded in the cache memory.

4 CONCLUSION

In [Blanco *et al.*,1994a; Goñi *et al.*,1995] we presented the advantages of using DL systems to build FDBS and to define a cache memory. It is our belief that reasoning mechanisms from DL are useful to perform query processing and in particular semantic query optimization, as said in [Beneventano *et al.*,1994]. However, only by using those reasoning mechanisms the optimal plans cannot be obtained and other techniques need to be applied. For example, most specific classes may be useful during the query processing but it is not always better to use them. Furthermore, we have defined a set of heuristics to obtain an optimal set of subqueries to process in the underlying databases and an optimization process of the mapping information and SQL plan for these DL queries.

DL systems have been also shown as appropriate to offer intensional answers to the users. We have incorporated this feature to our FDBS.

Lastly, we also consider interesting to incorporate a metalevel mechanism to DL systems in order to permit a different query processing strategy [Bermúdez *et al.*,1995].

References

[Arens and Knoblock, 1994] Y. Arens and C. A. Knoblock. Intelligent caching: Selecting, representing and reusing data in an information server. In *Proceedings of the Third International Conference on Information and Knowledge Management CIKM*, 1994.

[Arens *et al.*, 1993] Y. Arens, C.Y. Chee, C. Hsu, and C.A. Knoblock. Retrieving and integrating data from multiple information sources. *International Journal of Intelligent and Cooperative Information Systems*, 2(2):127–158, 1993.

[Beneventano *et al.*, 1994] D. Beneventano, S. Bergamaschi, and C. Sartori. Subsumption for semantic query optimization in OODB. In *Proc. of the International Workshop on Description Logics. Bonn. Germany*, 1994.

[Bermúdez *et al.*, 1995] J. Bermúdez, A. Illarramendi, J.M. Blanco, and A. Goñi. Metalevel for an efficient query answering. In *Proc. of the International Workshop on Description Logics. Roma. Italy*, June 1995.

[Bertino *et al.*, 1989] E. Bertino, M. Negri, G. Pelagatti, and L. Sbatella. Integration of heterogeneous database applications through an object-oriented interface. *Information Systems*, 14(5), 1989.

[Blanco *et al.*, 1994a] J.M. Blanco, A. Illarramendi, and A. Goñi. Building a federated database system: an approach using a knowledge based system. *International Journal of Intelligent and Cooperative Information Systems*, 3(4):415–455, December 1994.

[Blanco *et al.*, 1994b] J.M. Blanco, A. Illarramendi, A. Goñi, and J. Bermúdez. Advantages of using a terminological system for integrating databases. In *Proc. of the International Workshop on Description Logics. Bonn. Germany*, 1994.

[Borgida and Brachman, 1993] A. Borgida and R. J. Brachman. Loading data into description reasoners. In *Proceedings of the ACM SIGMOD Conference*, 1993.

[Ceri and Pelagatti, 1984] S. Ceri and G. Pelagatti. *Distributed Databases: Principles and Systems.* Mac Graw Hill, 1984.

[Collet *et al.*, 1991] C. Collet, M. N. Huhns, and W. Shen. Resource integration using a large knowledge base in CARNOT. *IEEE Computer*, pages 55–62, December 1991.

[Dayal, 1985] U. Dayal. *Query Processing in a Multidatabase System*, pages 81–108. Springer-Verlag, 1985.

[Devanbu, 1993] P.T. Devanbu. Translating description logics to information server queries. In *Proceedings of the ISMM International Conference on Information and Knowledge Management CIKM*, 1993.

[Du *et al.*, 1992] W. Du, R. Krishnamurthy, and M. Shan. Query optimization in heterogeneous DBMS. In *Proc. of the 18th VLDB Conference*, 1992.

[Du *et al.*, 1995] W.D. Du, M. Shan, and U. Dayal. Reducing multidatabase query response time by tree balancing. In *Proc. of 1995 ACM SIGMOD, May*, 1995.

[1] The cache memory is the extension corresponding to the integrated terminology that is maintained in main memory, and that has to be small enough to be efficient.

[Goñi *et al.*, 1995] A. Goñi, A. Illarramendi, and J.M. Blanco. Caching on multidatabase systems based on DL. In *Proc. of the International Workshop on Description Logics. Roma. Italy*, June 1995.

[Kim, 1995] W. Kim. *Modern Database Systems.* ACM press, 1995.

[Larson *et al.*, 1989] J. A. Larson, S. B. Navathe, and R. Elmasri. A theory of attribute equivalence in databases with application to schema integration. *IEEE TOSE*, SE-15(4), April 1989.

[Levy *et al.*, 1995] A.Y. Levy, D. Srivastava, and T. Kirk. Data model and query evaluation in global information systems. *Journal of Intelligent Information Systems*, 5(2):121–143, September 1995.

[Lu *et al.*, 1992] H. Lu, B. Ooi, and C. Goh. On global multidatabase query optimization. *SIGMOD RECORD*, 21(4):6–11, December 1992.

[Navathe *et al.*, 1989] S. Navathe, S. K. Gala, and S. Geum. Federated information bases: A loose-coupled integration of databases systems and application subsystems. In *Proc. of the 4th. Database Symposium*, 1989.

[Ozsu and Valduriez, 1991] M. T. Ozsu and P. Valduriez. *Distributed Databases: Principles and Systems.* Prentice Hall, 1991.

[Peltason *et al.*, 1989] C. Peltason, A. Schmiedel, C. Kindermann, and J. Quantz. The BACK system revisited. Technical University Berlin. KIT-Report 75, September 1989.

[Pitoura *et al.*, 1995] E. Pitoura, O. Bukhres, and A. Elmagarmid. Object orientation in multidatabase systems. *ACM Computing Surveys*, 27(2):141–195, June 1995.

[Qutaishat *et al.*, 1992] M.A. Qutaishat, N.J. Fiddian, and W.A. Gray. Association merging in a schema meta-integration system for a heterogeneous object-oriented database environment. In *Lecture Notes in Computer Science Proc. 10th British National Conference on Databases*, 1992.

[Schild, 1991] K. Schild. A correspondence theory for terminological logics – preliminary report. In *Prooceedings IJCAI'91, Sidney, Australia*, August 1991.

[Spaccapietra *et al.*, 1992] S. Spaccapietra, C. Parent, and Y. Dupont. Model independent assertions for integration of heterogeneous schemas. *VLDB*, 1:81–126, 1992.

[Sull and Kashyap, 1992] W. Sull and R. L. Kashyap. A self-organizing knowledge representation scheme for extensible heterogeneous information environment. *IEEE Transactions on Knowledge and Data Engineering*, 4(2):185–191, April 1992.