Query Resolution in Independent Databases by Partial Integration

Alejandro Botello C., Adolfo Guzmán A., and Renato Barrera
Centro de Investigación en Computación, Instituto Politécnico Nacional[1],
Centro de Ciencias Aplicadas y Desarrollo Tecnológico, Universidad Nacional Autónoma de México[2]
botello@cic.ipn.mx[1], a.guzman@acm.org[1], rntbarrera@yahoo.com[2]

Abstract

There is an abundance of information spread across the world, usually stored in databases or published as web pages. These data is not structured in a common way, because the sources were built to fill different purposes or they were made by different persons. Presently, if a user wants to exploit data coming simultaneously from diverse data sources and obtain useful results, he must combine all disperse pieces of relevant data and check the correspondence between them, regularly by hand.

In this research we propose the development of a database integration system (DaBIS) with the following features: 1) it allows many connections with remote and heterogeneous databases to extract their constitutive schemas; 2) it makes inferences about the data of each database, using matching algorithms to produce useful correspondences; and 3) it solves a user's global queries by means of a graphical interface, when the query involves diverse information sources. Our main idea is to combine in a semiautomatic way partial results obtained from distinct databases, and get relevant results, while, at the same time, simplify the formulation of difficult queries.

1. Introduction

Commercial enterprises, academic institutions and government bureaus, among other actors, daily manage large amounts of information published in the Internet either to enable data interchange, or simply, to make it publicly available. Such information, though, does not generally have a common structure that allows a straightforward solution of global queries. For example, suppose that a user want to know "the average age of high school students that live in the North of Mexico City, and have a parent working in some government's office". In order to solve this query, we have to decompose it into subqueries corresponding to each information source, i. e., we need to get first the students names that live in the North of Mexico City, then their parent's names, and the government's office where they work at, then verify the accuracy and correspondences between data (for example, that the Zip code from student's address corresponds to the northern area), and finally, combine all partial data into a useful result. We suppose that a) all the information sources are remote, and b) that we know nothing about their structure, or about which information sources have data relevant for our query, so that we can have an idea of results that can be obtained from each source. These hypotheses make the search and analysis of the information more complex.

In this work, we propose the development of an integrator (relational) database system, DaBIS, with the following characteristics:

a) It enables the connection and management of many heterogeneous databases, the acquisition of their internal structures, and the retrieval of partial results from subqueries;
b) It establishes, in a semi automatic manner, the logical correspondences among all database structures, by finding content mismatches, defining and applying data transformations, and inferring some mappings for each data structures;
c) It decomposes and solves queries formulated trough a user graphical interface, in which the global query are divided into several subqueries mapped to corresponding data sources, and
d) It merges the partial results into a final answer.

Figure 1 shows the structure of the DaBIS and the manner in which a sample query is solved. The method used in the solution will be explained in detail in Section 2, while a comparison with other systems is showed in Section 3. Finally, in Section 4 we comment some results from test, and give some conclusions.

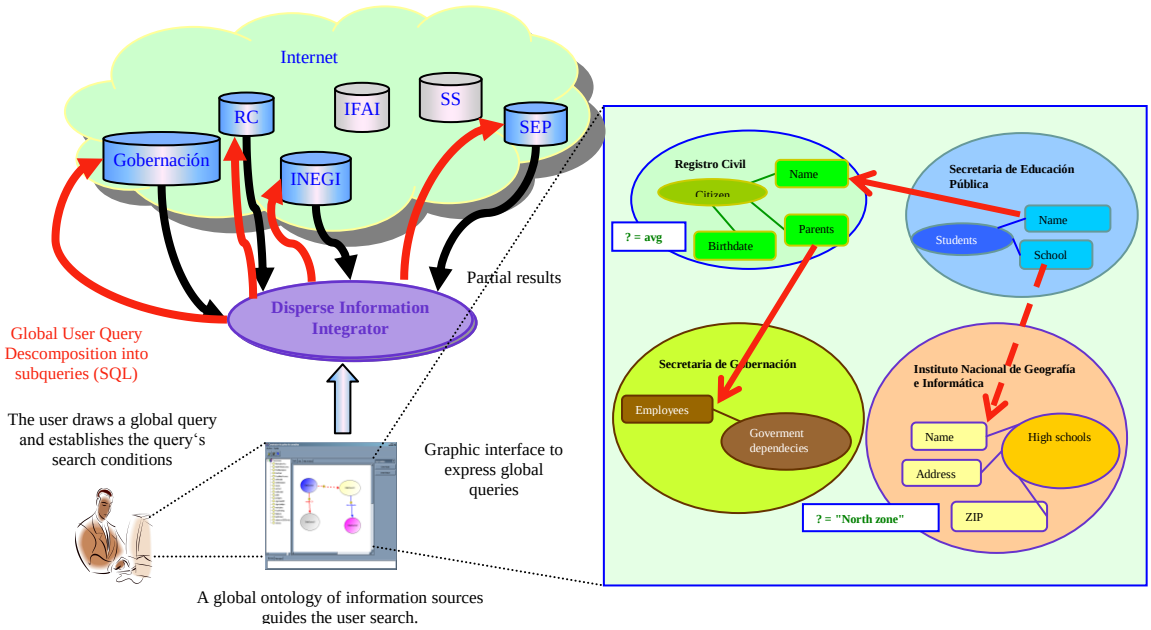

Figure 1. Solving global user queries over disperse data sources

2. Architecture of DaBIS

DaBIS has four constitutive modules:

- The Manual Linker (ML)
- The Graphical Query Builder (GQB),
- The Query Solver (QS), and
- The Data source Access Connector (DAC)

2.1 The Manual Linker and the Graphical Query Builder

The system manager incorporates the schemas (in Data Definition Language, DDLs) of each available data source in a panel (Figure 2), and draws directed lines (links) between pairs of data source to establish relationships between them. He can establish some restrictions among the data so linked, as ranges, aggregates, ordering, etc. For example, suppose that in one database the student name is "John C. Smith", while in the other is "Smith, John C."; both refer to the same person, but the orders of the first name, middle name and last name are different. In some cases, the system can detect some of these dissimilarities, based on a column's data type (integers, characters, dates or other), or on data restrictions (primary/foreign key, uniqueness, cardinality and participation) or on data content, e. g. formats (mm/dd/yyyy or dd-mm-yyyy), ranges (ages from 18 to 65 years), groups ('Manager', 'Analyst', 'Programmer'), conversions (from dollars to euros) or substitutions (N, S, W, E). In the case of data mismatches that can not be detected by the system, the system administrator can add new restrictions, written as data transformations functions to be included into

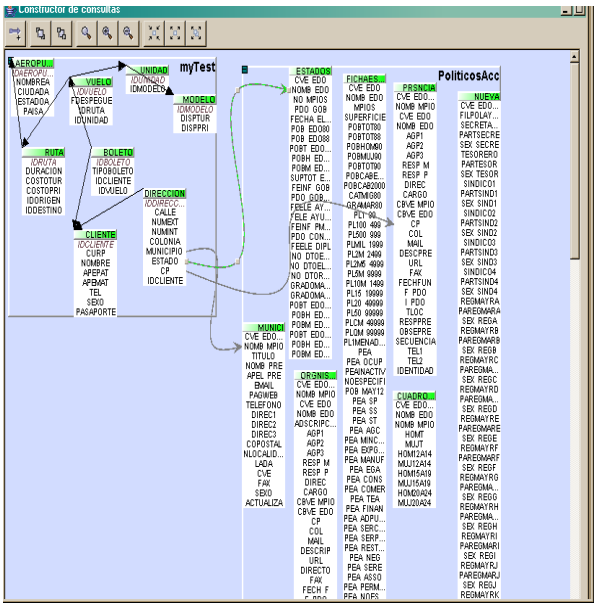

Figure 2. Panel where the system administrator adds links among relevant data items in disparate databases. These links are added only once, each time a new table is introduced

SQL queries. The system administrator has to perform this task (to add databases to the panel, through the ML) manually, only once for each new table.

2.2 The Query Solver

In this module, the system infer some relationships among the data sources involved in the query, e. g., if the site with a student's average age does not include the geographic orientation, we need to find some kind of relationship with another data source that was not initially considered, like cartographic maps of schools in Mexico City. For example, if one data source has the school's names in the North zone, and other data source has the government dependency's addresses, the system has to find some type of relationship between them, like the Zip code. If the system can not provide some inference, it asks the user for a possible alternative way.

Then, the Query Solver transforms each data source diagram into subqueries, one for each involved data base, written in SQL, to be solved by the correspondent "local" database management system (DBMS). Although databases are independent from each other, having different schemas, by combining parts of their data useful results can be obtained. Then, each subquery could be solved by a SQL sentence that returns only tuples involved in user constraints, and with the respective union attributes. The order in which partial queries are solved is relevant, because it is necessary to find the related attributes that permit to join data tables.

In the example of Fig. 3, if all information was contained in a single database, the SQL query that solves the problem would be like:

```
Select avg(s.age)
from GovDependencies g, Citizens c, Students
s, HigSchools h
where g.employee = c.parents
and c.name = s.name
and s.zip = h.zip
and h.zone = 'N' and h.city = 'Mexico'
```

Figure 3. Path or diagram drawn by the user over the panel of figure 2, to indicate the query "give me all the students that live in…". A path is drawn for each query

In a environment in which databases are disperse and all of them are independent, DaBIS need to solve this global query by rewriting it into several subqueries, one for each table included in the from clause, in the following way

```
select avg(t.age)
from temp4 t;

[temp4] ←
select getAge(t.birthdate) as age
from GovDependencies g, temp3 t
where g.employees = t.parents

[temp3] ←
select c.birthdate as birthdate,  c.parents
from citizen c, temp2 t
where c.name = t.name

[temp2] ←
select s.name
from highSchools h, temp1 t
where h.name = t.school
and h.zone = 'North' and h.city = 'Mexico'

[temp1] ←
select s.name, s.school
from students s
```

Although in each step there are partial results (from temp1 to temp4), these tuples are sent to the QS, which receives them and joins each one of them creating with the results a new temporal table (*temp1*). In our example, the QS follows this sequence: it first get the data corresponding to attributes *name* and *school* from the *students* table, to then forward this result to be joined with the high school's information (*highSchools* table), filtering these results by *zone* and *city;* afterwards, the results are joined with citizen's information, to be in turn joined with the government's data to produce a new partial result. The data obtained from each subquery is inserted into a local DBMS (hsqlbd)[8], which is a small relational engine built as Java classes, and easily transportable with the entire system.

2.3 The Data Source Access Connector

Each leaf node in the ontology's diagram represents a database that has tables, users, views, and other database objects. The information about these objects is extracted from the database's schemata and is stored

Figure 4. XML document for a data source connection.

locally as a XML document, as we shown in Figure 4. This file contains all related information about database structure, like catalog/schema owners, table and column's names, data types, primary/foreign keys relationships, functions/procedures supported, and also database connection properties (JDBC driver name and URL, user name and password).

When one partial result is joined with another data source, the DAC verifies whether there are some transformations to be applied to data, e. g. conversions, substitutions or validations (as mentioned in 2.1), stored in an XML document (mappings.xml). An XSLT template has been built to map these data sources, applying string compilations and probabilistic algorithms to determine the degrees of similitude among the sources; these algorithms are based on a dictionary built with common terms in the schema definition. If no exact or approximate correspondence exists, the system will then try to infer correspondences based on their content, exploring the values and restrictions, and looking for common patterns in data values. We have solved some data dissimilarities like naming and domains, and are presently working in other restrictions. These transformations are edited by the system administrator, who marks correspondences between nodes, and applies these changes to the data source's ontology. Final results of partial integration are stored as XML documents to be displayed on the graphical user interface.

Each DBMS in the system is accessed trough a Java JDBC driver, of type 1 or 4, and their schemata been extracted by the ResultSetMetaData JDBC API interface. The schemata obtained by the JDBC drivers are stored in a XML documents, one for each data access, this allows the system to know locally the schema of a data source even in the absence of a net connection. With such schema information, it is possible to draw a relational diagram showing the table names with their contained columns, and their relationships. If no explicit relationships are found by the ResultSetMetaData implementation, the inference engine tries to determine the most probable integrity restrictions using the primary/foreign key references; if this is not possible, the system will ask the user to mark the proper links between tables.

Applying XSLT templates to the metadata's XML document, we can obtain the SQL's DDL *create table* statements (see Figure 5) as well as the XML Schema. Finally we map each of the database transformations to generate the final result. If the user finds some inconsistencies, he can correct the model manually and the system will draw some inferences to apply them to the solution of the same problem in other similar contexts. For now, we not considerate do any kind of optimizations, but in a near future could be implemented.

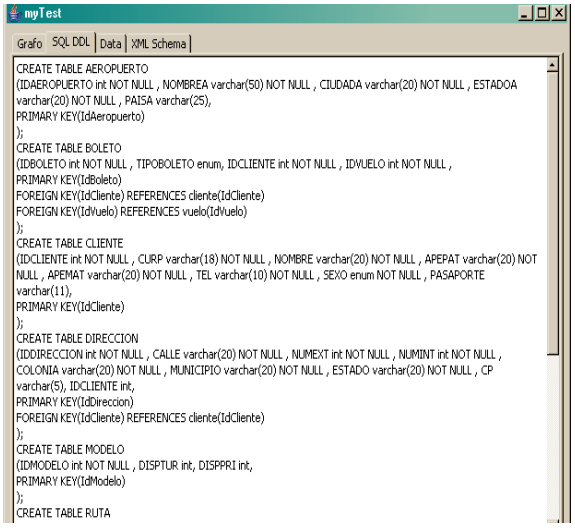

Figure 5. SQL DDL obtained from a data source connection.

3. Related Work

There is a plethora of work in the integration information area. Based in taxonomy described in [work], in which there are several main criteria to classify the different approaches, the most relevant systems to compare with DaBIS are TSIMMIS[7], SIMS[9], Garlic[10], and INDUS [INDUS]. While TSIMMIS and Garlic use a mediator agent, SIMS, INDUS and DaBIS use an ontology approach to integrate data sources, which abstracts user's models

from physical considerations of data sources. Although all of these systems enable the user interaction to select data source to be integrated, none offer a complete abstraction of query formulation, as we propose with a graphical user interface.

4. Results

We are using JGraph[7] to diagram queries on the user's interface. For now, this interface show the data sources as a graph, and user can connect many graph to express queries. The system solves this relationship with mappings, and permits the user to establish conditions and restrictions over the paths. When a path is solved, the results are integrated and shown in XML format.

With respect to database systems, we have tested our system with MySQL 4.0, MS Access 2003, SQLServer 2000 and Oracle 10g. In some cases, using JDBC:ODBC driver has some problems with metadata, specifically with the primary/foreign keys.

Acknowledgments. This work has been supported by CONACyT.

REFERENCES

[1] William Cohen. *"Integration of heterogeneous databases without common domains using queries based on textual similarity"*. In Proc. Of ACM SIGMOD Conf. On Management of Data, Seattle, WA, 1998.

[3] R.J. Miller, M.A. Hernandez, L.M. Haas, L.-L. Yan, C.T.H. Ho, R. Fagin, and L. Popa. "The Clio Project: Managing Heterogeneity". SIGMOD Record, 30(1):78-- 83, 2001.

[4] H.-H. Do and E. Rahm. COMA - A System for Flexible Combination of Schema Matching Approaches. In VLDB, 2002.

[5] E. Rahm, P.A. Bernstein. A survey of approaches to automatic schema matching. VLDB J. 10:4 (2001), pp. 334:350.

[6] Michael R. Genesereth, Arthur M. Keller, and Oliver Duschka, "Infomaster: An Information Integration System," in proceedings of 1997 ACM SIGMOD Conference, May 1997

[7] Sudarshan Chawathe, Hector Garcia-Molina, Joachim Hammer, Kelly Ireland, Yannis Papakonstantinou, Jeffrey Ullman, Jennifer Widom. The TSIMMIS Project, Integration of Heterogeneous Information Sources. Department of Computer Science, Stanford University, March 1997

[8] A. Kirk, A. Y. Levy, Y. Sagiv, and D. Srivastava. The Information Manifold. In AAAI Spring Symposium on Information Gathering. AAAI, 1995.

[9] Y. Arens, C. Chee, C. Hsu and C. Knoblock. Retrieving and Integrating Data from Multiple Information Sources. In Journal of Intelligent and Cooperative Information Systems, Vol. 2, June 1993.

[10]M.J. Carey et al. Towards heterogeneous multimedia information systems: The Garlic approach. Technical Report RJ 9911, IBM Almaden Research Center, 1994.

[11] Babu S., Bizarro P., Adaptive Query Processing in the Looking Glass. Proc. of the 2nd Biennial Conf. on Innovative Data Systems Research (CIDR), Jan. 2005.

[12] JGraph - The Java Open Source Graph Drawing Component, http://www.jgraph.com/jgraph.html

[13] HSQLDB – A Lightweight 100% Java SQL Database Engine, http://hsqldb.org/

[work] P. Ziegler. User-Specific Semantic Integration of Heterogeneous Data: What Remains to be Done? Technical Report ifi-2004.01, Department of Informatics, University of Zurich, 2004. http://www.ifi.unizh.ch/techreports/TR 2004.html.

[INDUS] J. Reinoso-Castillo, A. Silvescu, D. Caragea, J. Pathak, V. Honavar, Information Extraction and Integration from Heterogeneous, Distributed Autonomous: A Federated Ontology-Driven Query-Centric Approach. In 1st IEEE Intl. Conference on Information Integration and Reuse (IRI-2003).