The INFOMIX System for Advanced Integration of Incomplete and Inconsistent Data*

Nicola Leone
Dipartimento di Matematica
Università della Calabria
Rende, Italy
leone@mat.unical.it

Georg Gottlob
Inst. für Informationssysteme
Technische Universität Wien
Vienna, Austria
gottlob@dbai.tuwien.ac.at

Riccardo Rosati
Dip. di Informatica e Sistemistica
Università di Roma "La Sapienza"
Roma, Italy
rosati@dis.uniroma1.it

1. INTRODUCTION

The task of an *information integration system* is to combine data residing at different sources, providing the user with a unified view of them, called *global schema*. Users formulate queries over the global schema, and the system suitably queries the sources, providing an answer to the user, who is not obliged to have any information about the sources. Recent developments in IT such as the expansion of the Internet and the World Wide Web, have made available to users a huge number of information sources, generally autonomous, heterogeneous and widely distributed: as a consequence, information integration has emerged as a crucial issue in many application domains, e.g., distributed databases, cooperative information systems, data warehousing, or on-demand computing. Recent estimates view information integration to be a $10 Billion market by 2006 [14].

However, information integration is in general an extremely complex task: Indeed, "there is still a tremendous amount of research, engineering and development work needed to make the full information integration vision a reality" [14]. Both state-of-the-art commercial software solutions (e.g., [11]) and academic systems (see e.g. [10] for a survey) fulfill only partially the ambitious goal of integrating information in complex application scenarios. Moreover, comprehensive, formal methodologies and coherent tools for designing information integration systems are still missing.

INFOMIX is a novel system which supports powerful information integration, utilizing advanced reasoning capabilities. It implements recent results in database theory, which have been extended and specialized within the INFOMIX project [3, 4, 7, 5, 6, 8]. While INFOMIX is based on solid foundations, it is a user-friendly system, endowed with graphical user interfaces for the average database user and administrator, respectively. INFOMIX is built in cooperation with RODAN systems, a commercial DBMS developer.

The main features of the INFOMIX system are:

• **A comprehensive information model**, through which the knowledge about the integration domain can be declaratively specified. The possibility of defining expressive integrity constraints (ICs) over the global schema, the precise characterization of the relationship between global schema and the local data sources, the formal definition of the underlying semantics, as well as the use of a powerful query language, allow for the specification of complex integration applications. Then, logic-based methods for answering user queries, which are sound and complete with respect to the semantic of query answering, guarantee meaningful data integration.

• **Capability of dealing with data that may result incomplete and/or inconsistent** with respect to global ICs. We point out that declarative solutions to the problem of query answering for incomplete and inconsistent data (e.g., [2, 9]) have been recently proposed. Nonetheless, all approaches did not produce so far effective and scalable system implementations, mainly due to the high computational complexity of the problem. In this respect, the INFOMIX system is the first one that is capable of dealing with incomplete and inconsistent information in realistic applications.

• **Advanced information integration algorithms** [4], reducing query answering to cautious reasoning on (head cycle free) disjunctive datalog programs, allow to effectively compute the query results precisely, by using the state-of-the-art disjunctive datalog system DLV [13]. The formal query semantics is captured also in presence of incomplete and/or inconsistent data.

• **Sophisticated optimization techniques** [7, 5, 6] guarantee the effectiveness of query evaluation in INFOMIX. The novel optimization techniques, developed in INFOMIX, "localize" the computation and limit the inefficient (coNP) computation to a very small fragment of the input, obtaining fast query-answering, even in such a powerful data-integration framework (see Section 4).

• **A rich data acquisition and transformation framework** for accessing heterogeneous data in many formats (including relational, XML and HTML data), supporting different wrapper types. By LiXto wrappers and tools [1, 8], powerful data extraction and preprocessing is possible.

Further information, system documentation, and publications are available on the INFOMIX website: `www.mat.unical.it/infomix`.

SIGMOD '05 Baltimore, Maryland USA

2. DATA INTEGRATION FRAMEWORK

From an abstract point of view, the data integration system $\mathcal{I}$ in INFOMIX is a triple $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where $\mathcal{G}$ is the global schema, $\mathcal{S}$ is the source schema, which comprises the schemas of all the sources to be integrated, and $\mathcal{M}$ is the mapping establishing a relationship between $\mathcal{G}$ and $\mathcal{S}$. We assume that both $\mathcal{G}$ and $\mathcal{S}$ are specified in the relational model: actually, $\mathcal{S}$ stems from a subset of XML which can be transformed back and forth into relational data.

$\mathcal{G}$ may contain integrity constraints (ICs) of three kinds: Classical *Key Dependencies* (KDs), *Inclusion Dependencies* (IDs), expressing that (a projection of) a relation is included in (a projection of) another relation, and *Exclusion Dependencies* (EDs), expressing that two relations (or projections on them) are disjoint.

$\mathcal{M}$ is a *Global-As-View* (GAV) mapping [12], i.e., $\mathcal{M}$ is a set of logical implications

$$\forall x_1 \cdots \forall x_n . \Phi_{\mathcal{S}}(x_1, \ldots, x_n) \supset g_n(x_1, \ldots, x_n), \qquad (1)$$

where g_n is a relation from $\mathcal{G}$, n is the arity of g_n, $\Phi_{\mathcal{S}}$ is a conjunction of atoms on $\mathcal{S}$ and $x_1, \ldots, x_n$ are the free variables of $\Phi_{\mathcal{S}}$. Each global relation is thus associated with a *union of conjunctive queries* (UCQs).

Given a source database instance $\mathcal{D}$, the semantics of $\mathcal{I}$ w.r.t. $\mathcal{D}$, denoted $sem(\mathcal{I}, \mathcal{D})$, is the set of those database instances $\mathcal{B}$ for the global schema $\mathcal{G}$ that satisfy the ICs of $\mathcal{G}$ and maximize the satisfaction of the mapping $\mathcal{M}$: more formally, each such $\mathcal{B}$ contains as many tuples as possible that satisfy the instantiation of the variables $x_1, \ldots, x_n$ in the mapping assertions (1) of $\mathcal{M}$, provided that they satisfy the ICs of $\mathcal{G}$. This generalizes the classical sound semantics for data integration [12] and allows us to deal with data that are inconsistent w.r.t. to ICs on the global schema. For a more detailed description of the semantics, see [4].

User *queries* are specified over $\mathcal{G}$ (see Section 3.1 for the query language). The set of *certain answers* to a user query q, denoted $ans(q, \mathcal{I}, \mathcal{D})$, is the set of the tuples in its evaluation on each global database instance in $sem(\mathcal{I}, \mathcal{D})$.

Within the INFOMIX project we made a detailed decidability and complexity analysis on query answering [3, 4]. We considered different combinations of ICs for the global schema and user query languages, and singled out conditions which guarantee decidability of query answering, and at the same time allow for a rich and flexible specification. More precisely, in absence of IDs in the global schema, we allow the user to pose recursive datalog queries with aggregates and stratified negation. If IDs are present, in order to avoid undecidability, we have to restrict the query language to UCQs, and force IDs to be *non-key-conflicting* IDs (NKCIDs), a class of IDs which generalizes Foreign-Key constraints (and thus very useful in practice). The complexity for query answering in this setting was shown to be coNP-complete [3, 4], which calls for advanced reasoning tools for query answering.

3. INFOMIX ARCHITECTURE

The INFOMIX system supports two modes: a design and a query mode. In the first, the global schema, the source schema, and the mapping between them are specified. Furthermore, wrappers for the data sources are created or imported. In the query mode, the system provides query answering facilities at run time, including data acquisition, integration, answer computation, and presentation to the user.

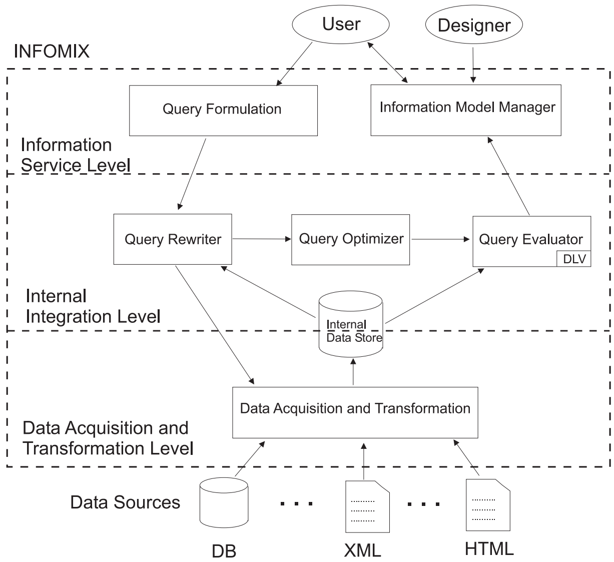


Figure 1: Conceptual INFOMIX Architecture

In both the design and query mode, INFOMIX is conceptually divided into three levels, as shown in Figure 1:

- **Information Service Level**. This level serves as a direct interface to the user (at run time) and the designer (at design time). It deals with global data, and provides the necessary interfaces (e.g., for global schema definition and for query formulation).
- **Internal Integration Level**. In this level, the actual integration of data is performed. It receives data from the sources, and exploits the mapping and the information of the global schema to effectively compute answers to the queries posed by the user at run time, relying on a computational logic system at its core.
- **Data Acquisition and Transformation Level**. Typically, information sources do not provide homogeneous data, which therefore can not be uniformly accessed. This level controls the way in which these data are acquired by the system, performs data transformations, alignment and cleaning, allowing upper levels to deal with source data in a uniform way.

3.1 Information Service Level

This level provides functionalities for specifying and manipulating the INFOMIX information model (design mode), and for query formulation (query mode). It comprises two modules: the Information Model Manager and the Query Formulation module. The *Information Model Manager* handles the definition of the global schema and the local schemas, as well as the mapping. It provides user-friendly interfaces for these tasks, including schema browsers. Furthermore, it gives automatic support for the verification of coherency, redundancy and adequacy of the application specification. In particular, if IDs are specified on the global schema, the module must check (and eventually alerting the user) wether the schema is non-key-conflicting, i.e., that all IDs are NKCIDs, thus avoiding cases in which query answering is undecidable [3]. Finally, this module presents query results in suitable form to the user. The *Query Formulation* module provides a graphical, user-friendly interface for query formulation over the global schema and query validation facilities; these check the interactions between user query and global ICs to guarantee that query answering is always decidable. For instance, if IDs are specified, the module must check whether the query is an UCQ; in absence of IDs, the module allows the user to exploit a richer query language than UCQs, with recursion, negation, aggregation functions and combinations thereof.

3.2 Internal Integration Level

The Internal Integration Level is based on computational logic and deductive database technology. It is composed by three modules, namely the Query Rewriter, the Query Optimizer and the Query Evaluator.

The *Query Rewriter* reformulates the user query according to global ICs. It makes use of a sub-module to verify data consistency: exploiting the mapping, the sub-module unfolds the user query over the source relations and activates the corresponding wrappers to retrieve relevant data; then it checks wether there are ICs violations. If no violations occur, the reformulation produced by the rewriter is a simple (disjunction free) Datalog program; otherwise, a suitable disjunctive Datalog program is generated that performs automatic "repair" of data, in a way such that cautious answers to this a program evaluated over the data sources correspond to the certain answers to the query. The *Query Optimizer* provides several optimizations strategies, which turned out to be crucial for the efficiency of the system; in particular, the module exploits some *focusing* techniques which are able to isolate the portion of the source database that is relevant to answer user query, by pushing constants in the query towards the sources. To this aim, an optimized (possibly disjunctive) Datalog program is generated by applying advanced binding propagation techniques à la Magic-Set [7, 5]. Finally, the optimized program is passed to the *Query Evaluator*. It first loads data from the Internal Data Store and then invokes DLV [13] in order to compute the consistent answers. The results are then sent to the *Information Model Manager* for suitable presentation to the user.

3.3 Data Acquisition & Transformation Level

This level provides uniform access to data sources. INFOMIX has an open architecture in this respect, which allows for the integration of heterogenous types of data sources, ranging from poorely structured data to standard data formats. The primary types are relational, XML, HTML, and object-oriented data sources (currently unsupported), but arbitrary other types of data sources can be incorporated easily. All data sources are conceptually transformed into a uniform source data format, which is a fragment of XML Schema, and can be browsed in this format.

The acquisition (and at the same time, transformation) of data is done by wrappers. A query plan for executing suitable wrappers is generated, which load data into the Internal Data Store. Constants are pushed to the wrappers (if possible) to reduce the amount of data retrieved. Currently, INFOMIX offers three classes of wrappers, which provide different levels of support for query formulation and wrapper code generation (see Table 1):

Code wrappers are basically a definition of an API and some code implementing it. The internals and characteristics of code wrappers are therefore inaccessible to INFOMIX. They serve mainly as a fallback, since one can provide a code wrapper for any type of source.

Wrapper Type	Query Formulation Support	Code Generation Support
Code Wrapper	none	none
Query Wrapper	none	automatic
Visual Wrapper	semi-automated	automatic

Table 1: Support in Wrapper Specification

Query wrappers ship queries to external data sources and treat the result as a logical data source. Prototypical query wrappers use the ODBC or JDBC interfaces to relational databases, but also wrappers using an XML query language are of this class. For this wrapper type, the system can give support for creation at design time, and could exploit its transparency for optimization by query modification.

Visual wrappers support full interactive development of wrappers at design time. Currently, there is support for developing LiXto wrappers [1] and pipes as well as for Rodan's Data Extractor. Thanks to LiXto Transformation Sever [8], powerful data extraction from web pages and further processing, including data cleaning and low-level data integration, is enabled. For example, web query forms can be wrapped easily which proves useful in practice.

4. APPLICATION AND EXPERIMENTS

4.1 Demo Scenario

We will demonstrate the INFOMIX prototype system by means of a real-life application scenario, in which data from various legacy databases and web sources must be integrated for a university information system. In particular, we built our information integration system on top of the data sources available at the University of Rome "La Sapienza".

The data sources comprise information on students, professors, curricula and exams in various faculties of the university. Currently, this data is dispersed over several databases in various (autonomous) administration offices and many webpages at different servers. Given this setting, we have devised a global schema of 14 relations,

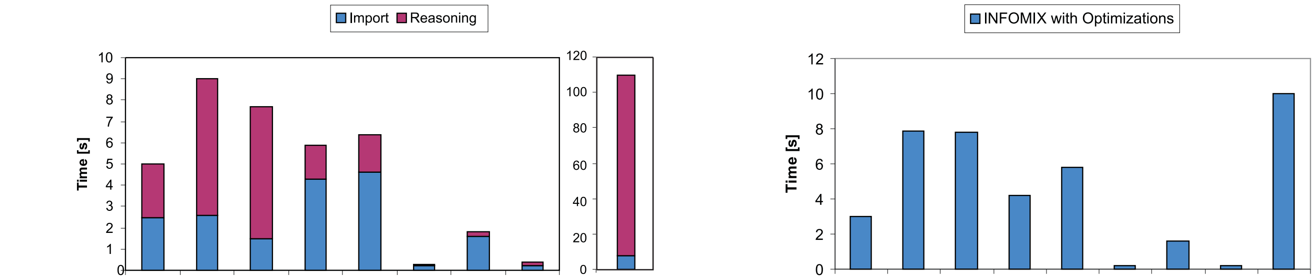
student(S_ID,FirstName,SecondName,CityOfResidence,
Address,Telephone,HighSchoolSpecialization)
enrollment(S_ID,FacultyName,Year)
course(C_Code, Description)
...

and 29 integrity constraints, comprising KDs, IDs, and EDs.

The demo scenario includes 3 legacy databases in relational format, comprising about 25 relations in total. The relation sizes range from a few hundred to tens of thousands of tuples (e.g., exam data). Besides these legacy databases, there are numerous web pages, which either provide information explicitly or through simple query interfaces (e.g., members of a department, phone numbers etc). We have developed a number of wrappers using LiXto tools, which extract information from these web sources. In total, there are about 35 data sources in the application scenario, which are mapped to the global relations with about 20 UCQs. Each UCQ joins up to three different logical data sources.

Finally, we have formulated 9 typical queries with peculiar characteristics, which model different use cases.

The demo scenario is non-trivial but, given its size, still comprehensible. It allows to present the features and usage of the INFOMIX system, both in the design mode and the query mode, in a nice way. Furthermore, it is open to modifications and experimentation by users.



	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
N° global relations	2	2	1	1	2	1	1	1	3
N° source relations	8	5	5	2	5	2	5	2	10
Size	161250	208828	143282	270047	381001	817	143282	817	531297

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
Relative Gain	40%	12%	-1%	29%	9%	27%	11%	45%	91%

Figure 2: Left: Execution time for Queries Q1,...,Q9. Right: Impact of Optimizations

4.2 Experiments

In the demo we will show a number of experiments in the scenario described in Section 4.1. Here we report performance results obtained on Pentium III machines running GNU/Linux with 256MB of memory. These demonstrate the feasibility of our approach, and the impact of optimization techniques.

The left part of Figure 2 shows the execution time for the 9 typical queries Q1, ...,Q9 without any optimization. For each query, both the *import time*, i.e., the time for importing all the data needed for producing the consistent answers into the main memory, and the *reasoning time*, i.e., the time for actually computing the answers in the computational logic environment, are shown. Moreover, for each query, also the total number of global relations involved, the total number of source relations accessed after query unfolding, and the total size of the data that must be retrieved from the sources for answering the query are shown. The import time for all queries is below 5 seconds, and for most queries also the reasoning time is in the same order of magnitude (we disregard wrapper execution times because of large variance in network performance). Query Q9, however, takes about 100 seconds of reasoning time. Inspection of the data revealed that for the global relations involved in Q9, there are about 50 conflicts (yielding about 2^{50} possible repairs), but only a small fraction of them is relevant to the query itself.

The right part of Figure 2 shows the total execution time for the optimized INFOMIX system, where in particular a magic-set technique is included. For many queries, we note a significant speed-up, especially for query Q9. We have verified that for Q9, the magic-set technique effectively prunes the unrelated conflicts, thus avoiding their repair. Note that for Q3 none of our optimizations applies; the overhead of the optimization methods (1%) is very lightweight, however.

5. ADDITIONAL AUTHORS

Thomas Eiter[1], Wolfgang Faber[1], Michael Fink[1], Gianluigi Greco[2], Giovambattista Ianni[2], Edyta Kalka[3], Domenico Lembo[4] Maurizio Lenzerini[4], Vincenzino Lio[2], Bartosz Nowicki[3], Marco Ruzzi[4], Witold Staniszkis[3], and Giorgio Terracina[2].

[1]Technische Universität Wien, Vienna, Austria
[2]Universitá della Calabria, Rende, Italy
[3]Rodan Systems S.A., Warsaw, Poland
[4]Università di Roma “La Sapienza”, Roma, Italy

6. REFERENCES

[1] R. Baumgartner, S. Flesca, and G. Gottlob. Visual web information extraction with LiXto. In *Proc. VLDB 2001*, pp. 119–128, 2001.

[2] L. Bravo and L. Bertossi. Logic programming for consistently querying data integration systems. In *Proc. IJCAI 2003*, pp. 10–15, 2003.

[3] A. Calì, D. Lembo, and R. Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *Proc. PODS 2003*.

[4] A. Calì, D. Lembo, and R. Rosati. Query rewriting and answering under constraints in data integration systems. In *Proc. IJCAI 2003*, pp. 16–21, 2003.

[5] C. Cumbo, W. Faber, G. Greco, and N. Leone. Enhancing the magic-set method for disjunctive datalog programs. In *Proc. ICLP 2004*, pp. 371–385.

[6] T. Eiter, M. Fink, G. Greco, and D. Lembo. Efficient evaluation of logic programs for querying data int egration systems. In *Proc. ICLP 2003*, pp. 163–177.

[7] W. Faber, G. Greco, and N. Leone. Magic sets and their application to data integration. In *Proc. ICDT 2005*, 2005. LNCS 3363, in press.

[8] G. Gottlob, C. Koch, R. Baumgartner, M. Herzog, and S. Flesca. The LiXto data extraction project - back and forth between theory and practice. In *Proc. PODS 2004*, pp. 1–12, 2004.

[9] G. Greco, S. Greco, and E. Zumpano. A logical framework for querying and repairing inconsistent databases. *IEEE TKDE*, 15(6):1389–1408, 2003.

[10] A. Y. Halevy. Data integration: A status report. In *Proc. BTW 2003*, pp. 24–29, 2003.

[11] H. Hayes and N. Mattos. Information on demand. *DB2 Magazine*, 8(3), 2003.

[12] M. Lenzerini. Data integration: A theoretical perspective. In *Proc. PODS 2002*, pp. 233–246, 2002.

[13] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The DLV System for Knowledge Representation and Reasoning. *ACM Trans. Comput. Log.*, 2004. To appear.

[14] N. M. Mattos. Integrating information for on demand computing. In *Proc. VLDB 2003*, pp. 8–14, 2003.