Data Integration: A Challenging ASP Application

Nicola Leone[1], Thomas Eiter[2], Wolfgang Faber[2], Michael Fink[2], Georg Gottlob[2], Luigi Granata[1], Gianluigi Greco[1], Edyta Kałka[4], Giovambattista Ianni[1], Domenico Lembo[3], Maurizio Lenzerini[3], Vincenzino Lio[1], Bartosz Nowicki[4], Riccardo Rosati[3], Marco Ruzzi[3], Witold Staniszkis[4], and Giorgio Terracina[1]

[1] Dipartimento di Matematica, Università della Calabria, Rende, Italy
[2] Inst. für Informationssysteme, Technische Universität Wien, Vienna, Austria
[3] Dip. di Informatica e Sistemistica, Università "La Sapienza", Roma, Italy
[4] Rodan Systems S.A., Warsaw, Poland

Abstract. The paper presents INFOMIX a successful application of ASP technology to the domain of Data Integration. INFOMIX is a novel system which supports powerful information integration, utilizing the ASP system DLV. While INFOMIX is based on solid theoretical foundations, it is a user-friendly system, endowed with graphical user interfaces for the average database user and administrator, respectively. The main features of the INFOMIX system are: (i) a comprehensive information model, through which the knowledge about the integration domain can be declaratively specified, (ii) capability of dealing with data that may result incomplete and/or inconsistent with respect to global constraints, (iii) advanced information integration algorithms, which reduce (in a sound and complete way) query answering to cautious reasoning on disjunctive Datalog programs, (iv) sophisticated optimization techniques guaranteeing the effectiveness of query evaluation in INFOMIX, (v) a rich data acquisition and transformation framework for accessing heterogeneous data in many formats including relational, XML, and HTML data.

1 Data Integration Systems

The enormous amount of information even more and more dispersed over many data sources, often stored in different heterogeneous formats, has boosted in recent years the interest for data integration systems (see, e.g, [2,6,5,4]). Roughly speaking, a data integration system $\mathcal{I}$ provides transparent access to different data sources by suitably combining their data, and providing the user with a unified view of them, called *global schema*. Formally, $\mathcal{I}$ can be seen as a triple $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where $\mathcal{G}$ is the global schema, $\mathcal{S}$ is the source schema, constituted by the schemas of all sources, and $\mathcal{M}$ is the mapping between $\mathcal{G}$ and $\mathcal{S}$ that contains a view over $\mathcal{S}$ for each element of $\mathcal{G}$. The users formulate their queries over $\mathcal{G}$, and the system automatically provides the answers. We assume that $\mathcal{G}$ and $\mathcal{S}$ are relational schemas, and that $\mathcal{G}$ contains integrity constraints (ICs) of three kinds: *key dependencies* (KDs), *exclusion dependencies* (EDs), and *inclusion dependencies* (IDs). Views in the mapping are specified in Datalog.

Example 1. Consider the integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where (i) $\mathcal{G}$ comprises the relation predicates $HasTutor(student, tutor)$ and $Student(name)$, (ii) $\mathcal{S}$ comprises

C. Baral et al. (Eds.): LPNMR 2005, LNAI 3662, pp. 379–383, 2005.

the unary relation predicate s_1 and the binary predicates s_2 and s_3, (iii) $\mathcal{M}$ is defined by the Datalog program:

$$HasTutor(x, y) \leftarrow s_2(x, y) \vee s_3(x, y) \qquad Student(x) \leftarrow s_1(x).$$

ICs in $\mathcal{G}$ state that: (a) the key of $HasTutor$ is the attribute $student$ (KD); (b) a tutor cannot be a student (ED), and (c) each name in $Student$ must occur in $HasTutor$, i.e., each student has at least one tutor (ID). □

Data stored at the sources may violate global ICs when filtered through the mapping, since in general source data are not under the control of the data integration system. The standard approach to this problem is inherently procedural, and basically consists of explicitly modifying the data in order to eliminate violation of ICs (data cleaning). However, the explicit repair of data is not always convenient or possible. Therefore, when answering a user query, the system should be able to "virtually repair" relevant data in a declarative fashion (in the line of [1,2]) .

In this paper, we present INFOMIX, a novel system which supports powerful integration of inconsistent data by resorting to computational logic systems. The INFOMIX approach is based on the idea of reformulating a user query, taking into account the ICs of the global schema, in terms of a logic program Π, using disjunction, in a way such that the stable models of Π represent the repairs of the global database constructed by evaluating the mapping over the source extension. Then, answering a user query amounts to cautious reasoning over the logic program Π augmented by the facts retrieved from the sources. We point out that this reduction is possible since query answering in data integration systems is coNP-complete in data complexity in the above setting (and also in many other data integration frameworks [5,4]).

INFOMIX implements recent results of the database theory, which have been extended and specialized within the INFOMIX project [3,4,9,7,8,10]. While INFOMIX is based on solid theoretical foundations, it is a user-friendly system, endowed with graphical user interfaces for the average database user and administrator. INFOMIX is built in cooperation with RODAN systems, a commercial DBMS developer. Among the different features supported by the INFOMIX system,[1] we shall consider here the following two aspects:

- **Advanced information integration algorithms** [4] that reduce query answering to cautious reasoning on (head cycle free) disjunctive Datalog programs. This allows for effectively computing the query results (even in the presence of incomplete and/or inconsistent data) by using the state-of-the-art disjunctive Datalog system DLV [11]. The formal semantics of queries is captured also in the presence of incomplete and/or inconsistent data.
- **Sophisticated optimization techniques** [9,7,8] 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.

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

We have experimented the system on a real-life application scenario. The results we have conducted clearly indicate the impact of the optimizations techniques on the system performance, and, more generally, show that, despite its worst-case computational complexity, the semantic approach to data integration pursued by INFOMIX can be effectively realized.

2 Logic Programming Within the 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. In both the design and query mode, INFOMIX is conceptually divided into three levels, namely the *Information Service Level*, the *Internal Integration Level* and the *Data Acquisition and Transformation Level*. The most important level is the Internal Integration Level, which 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 taking into account global ICs. According to the notion of repair adopted in INFOMIX, the Query Rewriter can separately consider IDs and KDs and EDs in the reformulation process, as showed in [3,4]. More precisely, it makes use of a sub-module to verify data consistency w.r.t. KDs and EDs: 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 KD or ED violations. If no violations occur, the reformulation produced by the rewriter is a simple (disjunction free) Datalog program, constructed according to the global IDs; otherwise, a suitable disjunctive Datalog program is generated that performs automatic repair of data w.r.t. both EDs and KDs. Basically, the Datalog program encodes the user query, the views in the mapping, and the global ICs that involve relation predicates that are relevant to answer the user query, in a way such that cautious answers to this program evaluated over the data sources correspond to the answers to the user query.

Example 2. In our example, consider now the the user query $q(x) \leftarrow HasTutor(x, y)$. Wrapper activation populates the Internal Data Store of the system with a database $\mathcal{D}$ for the source schema $\mathcal{S}$. Let us assume that the consistency check finds out that the KD and the EDs in $\mathcal{G}$ are violated by data migrating from the sources to the global schema through the mapping. Then, the Query Rewriter module reformulates the query in the following program:

$$
\begin{aligned}
q(x) &:- HasTutor(x, y) \\
q(x) &:- Student(x) \\
HasTutor(x, y) &:- HasTutor_{\mathcal{D}}(x, y)\ ,\ not\ \overline{HasTutor}(x, y) \\
\overline{HasTutor}(x, y) \vee \overline{HasTutor}(x, z) &:- HasTutor_{\mathcal{D}}(x, y)\ ,\ HasTutor_{\mathcal{D}}(x, z)\ ,\ z \neq y \\
\overline{HasTutor}(x, y) \vee \overline{HasTutor}(y, z) &:- HasTutor_{\mathcal{D}}(x, y)\ ,\ HasTutor_{\mathcal{D}}(y, z) \\
Student(x) &:- Student_{\mathcal{D}}(x)\ ,\ not\ \overline{Student}(x) \\
\overline{HasTutor}(x, y) \vee \overline{Student}(y) &:- HasTutor_{\mathcal{D}}(x, y)\ ,\ Student_{\mathcal{D}}(y) \\
Student_{\mathcal{D}}(x) &:- s_1(x)
\end{aligned}
$$

$$HasTutor_{\mathcal{D}}(x, y) :- s_2(x, y)$$
$$HasTutor_{\mathcal{D}}(x, y) :- s_3(x, y)$$

Informally, for each global relation r, the above program contains *(i)* a relation $r_{\mathcal{D}}$ that represents the extension of r obtained by evaluating the associated view in the mapping over the source database $\mathcal{D}$; *(ii)* a relation r that represents a subset of such extension that is consistent with the KD and the EDs for r; *(iii)* an auxiliary relation $\overline{r}$. The first two rules encode in the user query the ID stating that each student must have a tutor (intuitively, in order to return all the students that have a tutor, the rewriting looks also in the predicate $Student$). The third and the fourth rule encode the KD on $HasTutor$. The fifth, the sixth, and the seventh rule encode the ED stating that students cannot be tutors, whereas the last three rules encode the mapping $\mathcal{M}$. □

The *Query Optimizer* provides several optimization 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 the 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 [9,7].

Finally, the optimized program is passed to the *Query Evaluator*. It first loads data from the Internal Data Store and then invokes DLV [11] in order to compute the answers. The results are then sent to the *Information Model Manager* for suitable presentation to the user.

3 Application and Experiments

We have tested the INFOMIX prototype system on 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 and 29 integrity constraints, comprising KDs, IDs, and EDs.

The application 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 [10], 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 through 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.

Figure 1 shows the execution time for the 9 typical queries Q1, . . . ,Q9. Results are obtained on Pentium III machines running GNU/Linux with 256MB of memory. These

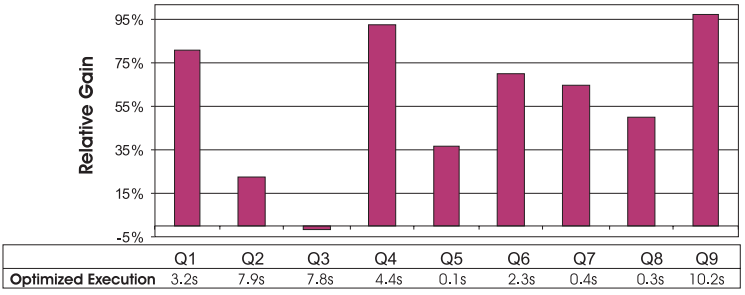

Fig. 1. Impact of Optimizations

demonstrate the feasibility of our approach, and the impact of optimization techniques. Actually, the figure shows the total execution time for the optimized INFOMIX system, where in particular a magic-set technique is included, and the relative gain w.r.t. the evaluation without any optimization. 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.

Acknowledgment. This work was partially funded by the Information Society Technologies programme of the European Commission, Future and Emerging Technologies under the IST-2001-33570 INFOMIX project.

References

1. M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *Proc. PODS 1999*, pages 68–79, 1999.
2. L. Bravo and L. Bertossi. Logic programming for consistently querying data integration systems. In *Proc. IJCAI 2003*, pages 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*, pages 260–271, 2003.
4. A. Calì, D. Lembo, and R. Rosati. Query rewriting and answering under constraints in data integration systems. In *Proc. IJCAI 2003*, pages 16–21, 2003.
5. J. Chomicki and J. Marcinkowski. Minimal-Change Integrity Maintenance Using Tuple Deletions. *Information and Computation*, 2004. 197(1-2): 90-121 (2005).
6. J. Chomicki, J. Marcinkowski, and S. Staworko. Computing consistent query answers using conflict hypergraphs. In *Proc. of CIKM 2004*, pages 417–426, 2004.
7. C. Cumbo, W. Faber, G. Greco, and N. Leone. Enhancing the magic-set method for disjunctive datalog programs. In *Proc. ICLP 2004*, pages 371–385, 2004.
8. T. Eiter, M. Fink, G. Greco, and D. Lembo. Efficient evaluation of logic programs for querying data int egration systems. In *Proc. ICLP 2003*, pages 163–177, 2003.
9. W. Faber, G. Greco, and N. Leone. Magic sets and their application to data integration. In *Proc. ICDT 2005*, 2005. LNCS 3363, in press.
10. 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*, pages 1–12, 2004.
11. 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. Available via http://www.arxiv.org/ps/cs.AI/0211004.