IBIS: Data Integration at Work (extended abstract)

Andrea Calì[1], Diego Calvanese[1], Giuseppe De Giacomo[1], Maurizio Lenzerini[1], Paolo Naggar[2], Fabio Vernacotola[2]

[1]Università di Roma "La Sapienza"
Dip. di Informatica e Sistemistica
via Salaria 113, I-00198 Roma, Italy
lastname@dis.uniroma1.it

[2]CM Sistemi
via N. Sauro 1, I-00195 Roma, Italy
firstname.lastname@gruppocm.it

Abstract. We present IBIS (*Internet-Based Information System*), a system for the semantic integration of heterogeneous data sources. IBIS adopts innovative and state-of-the-art solutions to deal with all aspects of a complex data-integration environment, including source wrapping, limitations on source access, and query answering under integrity constraints.

1 Introduction

The goal of a Data Integration system is to provide a uniform access to a set of heterogeneous data sources, freeing the user from the knowledge about the data sources themselves. In this paper we present IBIS (*Internet-Based Information System*), a system for the semantic integration of heterogeneous data sources, studied and developed in the context of a collaboration between the University of Rome "La Sapienza" and CM Sistemi.

The problem of designing effective data integration systems has been addressed by several research and development projects in the last years. Most of the data integration systems described in the literature [7, 12, 16, 15, 8, 6, 1], are based on a unified view of data, called *mediated or global schema*, and on a software module, called *mediator* that collects and combines data extracted from the sources, according to the structure of the mediated schema. A crucial aspect in the design and the realization of mediators is the specification of the relation between the sources and the mediated schema. Two basic approaches have been proposed in the literature [14]. The first approach, called *global-as-view* (or simply GAV), focuses on the elements of the mediated schema, and associates to each of them a view over the sources. On the contrary, in the second approach, called *local-as-view* (or simply LAV), the focus is on the sources, in the sense that a view over the global schema is associated to each of them. Indeed, most data integration systems adopt the GAV approach. This is, for example, the case of TSIMMIS [7], Garlic [13], COIN [6], Squirrel [17, 16], and MOMIS [1].

IBIS adopts innovative and state-of-the-art solutions to deal with all aspects of a complex data integration environment, including source wrapping, limitations on source access, and query answering under integrity constraints. IBIS follows the GAV approach, using a relational mediated schema to query the data at the sources. The system is able to cope with a variety of heterogeneous data sources, including data sources on the Web, relational databases, and legacy sources. Each non relational source is wrapped to provide a relational view on it. Also, each source is considered incomplete, in the sense that its data contribute to the data integration system. A key issue is that the system allows the specification of integrity constraints in the global schema, derived from the domain of interest, and not from the sources. Since sources are autonomous and incomplete, the extracted data in general do not satisfy the constraints. To deal with this characteristic, IBIS adapts the information extracted from the sources, so as to answer queries at best with the data available. In this way, answers are obtained that would not be provided by the standard unfolding strategy associated with GAV data integration systems. Indeed, GAV data integration systems, such as the above mentioned ones, answer a query posed over the global schema by unfolding each atom of the query using the corresponding view [14]. The reason why unfolding is sufficient in those system is that the GAV mapping essentially specifies a single database conforming to the global schema. Since no integrity constraint is defined in the schema, such a database is always legal for the schema.

IBIS also deals with limitations in accessing data sources, e.g., those requiring filling at least one field in a Web form, and exploits techniques developed for querying sources with binding patterns in order to retrieve the maximum set of answers [11, 5, 9, 10]. Since such extraction process is expensive, new types of optimizations are performed, taking into account integrity constraints holding on the sources.

2 Modeling in IBIS

The modeling of a data integration application in IBIS is based on the relational model with integrity constraints. As usual, a *relational schema* is constituted by a set of relation symbols, each one with an associated arity, denoting the number of its attributes, and a set of integrity constraints. In IBIS, we deal with four kind of constraints (the notion of satisfaction for these types of constraints is the usual one): *Key constraints:* Given a relation r in the schema, a key constraint over r is expressed in the form $key(r) = \mathbf{X}$, where $\mathbf{X}$ is a set of attributes of r. *Foreign key constraints:* We express a foreign key constraint in the form $r_1[\mathbf{X}] \subseteq r_2[\mathbf{Y}]$, where r_1, r_2 are relations, $\mathbf{X}$ is a sequence of distinct attributes of r_1, and $\mathbf{Y}$ is a sequence formed by the distinct attributes forming the key of r_2. *Functional dependencies:* A functional dependency over a source s has the form $s : \mathbf{A} \rightarrow \mathbf{B}$, where $\mathbf{A}$ and $\mathbf{B}$ are subsets of the set of attributes of s. *Full inclusion dependencies:* A full incusion dependency between two sources s_1 and s_2 is denoted by $s_1 \subseteq s_2$.

A data integration application in IBIS is modeled through a triple $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where $\mathcal{S}$ is the *source schema*, $\mathcal{G}$ is the *global schema*, and $\mathcal{M}$ is the *mapping* between the two. The global schema $\mathcal{G}$ is expressed as a relational schema with key and foreign key constraints. The source schema $\mathcal{S}$ is a relational schema with full inclusion and functional dependencies. The mapping is of type GAV: to each relation r in the global schema, $\mathcal{M}$ associates a query $\rho(r)$ over the source schemas. Each query is a union of conjunctive queries (UCQ) plus a part, denoted in logic programming notation, which specifies an additional processing to be carried out on the data retrieved by the UCQ associated to $\rho(r)$ in order not to violate the key constraint of r. In the following, we will not delve into the detail of this part: we will assume that the key constraint of each relation in $\mathcal{G}$ is satisfied. Finally, queries over the global schema are also union of conjunctive queries.

In order to assign semantics to a data integration application $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, we start with the data at the sources, and specify which data satisfy the global schema. A *source database* $\mathcal{D}$ for $\mathcal{I}$ is a relational database constituted by one relation $r^{\mathcal{D}}$ for each source r in $\mathcal{S}$. A source database is said to be *legal* for $\mathcal{S}$ if it satisfies all the constraints in $\mathcal{S}$. A *global database* $\mathcal{B}$ for $\mathcal{I}$, or simply database for $\mathcal{I}$, is a database for $\mathcal{G}$. Given a legal source database $\mathcal{D}$ for $\mathcal{S}$, a global database $\mathcal{B}$ is said to be *legal* for $\mathcal{I}$ with respect to $\mathcal{D}$ if: $\mathcal{B}$ satisfies the integrity constraints of $\mathcal{G}$, and $\mathcal{B}$ satisfies the mapping $\mathcal{M}$, that is, for each relation r in $\mathcal{G}$, we have that the set of tuples $r^{\mathcal{B}}$ that $\mathcal{B}$ assigns to r contains the set of tuples $\rho(r)^{\mathcal{D}}$ that the query corresponding to r retrieves from the source database $\mathcal{D}$, i.e., $\rho(r)^{\mathcal{D}} \subseteq r^{\mathcal{B}}$. Observe that the previous assertion amounts to consider any view $\rho(r)$ over $\mathcal{S}$ as *sound*, i.e., the tuples provided by $\rho(r)$ are sound but not necessarily complete.

Given a data integration application $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ and a legal source database $\mathcal{D}$, the *semantics* of $\mathcal{I}$ is the set of global databases that are legal for $\mathcal{I}$ wrt $\mathcal{D}$. If such a set is not empty, the source database $\mathcal{D}$ is said to be *consistent* with $\mathcal{I}$.

The fact that the semantics of a data integration application needs to be defined in terms of a *set* of databases rather than a single one has a deep influence on the nature of query answering in IBIS. Given a query q over the global schema of a data integration application $\mathcal{I}$, and a legal source database $\mathcal{D}$, the *certain answers* $q^{\mathcal{I},\mathcal{D}}$ of q to $\mathcal{I}$ wrt $\mathcal{D}$ are the tuples that satisfy the query in every database that belongs to the *semantics* of $\mathcal{I}$, i.e., in every global database that is legal for $\mathcal{I}$ wrt $\mathcal{D}$. It follows that query processing in IBIS aims at computing the certain answers of the query.

3 Architecture of IBIS

The system architecture of IBIS is shown in Figure 1. Four subsystems can be identified: *Wrapping Subsystem:* Its task is to provide a uniform layer in which all data stored at the sources are presented in the relational model. The wrappers in IBIS also take into account the limitations in accessing the sources. *Configuration Subsystem:* It supports system management and configuration of all the meta-data, which are stored in a repository according to a proprietary information model. The main design tool of IBIS is the Configuration Manager. Through it, the

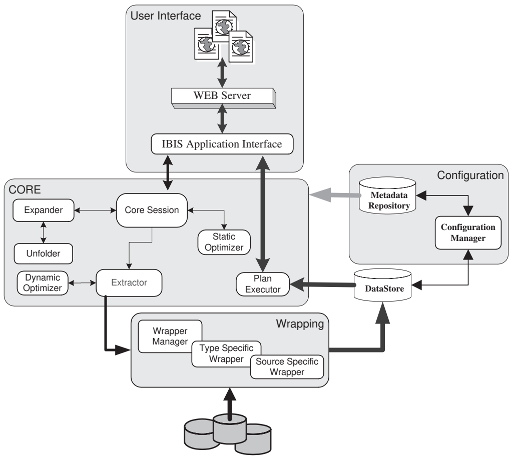

Fig. 1. Architecture of IBIS

designer is able to perform various configuration tasks, including schema design and the possibility to define properties of data sources. *IBIS Core:* It implements the actual data integration algorithms and controls all the parts of the system, taking care at runtime of all the aspects of query processing. User queries are issued to the IBIS core by the application interface; the core performs evaluation of a query by extracting data from the sources and executing the query over such extracted data. *Data extraction* and *query processing*, which are the main tasks of the IBIS Core, will be described in Sections 4 and 5, respectively. *User Interface:* Users of IBIS interact with the system by means of a Web interface. The Web Server dynamically generates HTML and XML documents by invoking the services provided by the application interface. The application interface provides functionalities such as authentication, browsing of the catalog of queries, setup of parametric queries, submitting a query etc., according to the information stored in the repository.

In addition to these subsystems, a *data store* is used to store temporary data which are used during query processing, and cached data extracted from the sources during the processing of previous queries.

4 Data Extraction

The extraction of the data from the sources is a key process in IBIS, and is complicated by the fact that limitations exist in accessing the sources. This is typical of Web data sources that are accessible through forms: usually a certain set of fields has to be filled with values in order to query the underlying database. Also, very often legacy databases have a limitation of this kind.

An interesting feature of IBIS is that the extraction process is decoupled from query processing. IBIS processes a query q by first expanding it to a query q'', and then obtains a query q' by unfolding q''. The resulting query q' is issued to the extractor. The extractor builds the so called *retrieved source database* $\mathcal{D}_q$ for q by retrieving all the tuples that may be used to answer q. Then, the query q' is evaluated over $\mathcal{D}_q$, providing the set of certain answers $q^{\mathcal{I},\mathcal{D}}$ to q.

4.1 Dealing with Access Limitations

In the presence of access limitations on the sources, simple unfolding is in general not sufficient to extract all obtainable answers from the sources [11, 5, 9, 10]. IBIS exploits techniques developed specifically for dealing with access limitations in the GAV approach, and extends the algorithm presented in [9] with suitable optimization techniques. The extraction of data according to this algorithm is performed as follows: starting from the set of initial values in the query, IBIS accesses as many sources as possible, according to their access limitations. The new values in the tuples obtained (if any), are used to access the sources again, getting new tuples, and so on, until there is no way of doing accesses with new values. At each step, the values obtained so far are stored.

The algorithm extracts all tuples obtainable respecting the access limitations, but there may be tuples in the sources that could contribute to the answer, but cannot be retrieved due to the access limitations. Although the extraction algorithm is straightforward, in order to make it efficient in practice, its implementation requires to take into account several technological aspects.

To avoid wrappers to be overloaded with a number of access requests that exceeds the capacity of the wrappers, they are fed with batches of requests that do not exceed a prefixed maximum size. For each wrapper, according to its capabilities, the system manager assigns the maximum dimension of the batches it can accept.

Furthermore, the extraction strategy of IBIS tries to keep working as many wrappers as possible. In order to do so, the IBIS Core constructs the requests to be sent to the wrappers independently from the order in which the values have been delivered to the retrieved source database. In doing so, it tries to generate the same amount of requests for each wrapper.

Also the new values in the tuples that are stored in the retrieved source database are not used to generate accesses immediately, but gradually and uniformly among sources. This strategy deals also with the problem of the large number of values extracted from sources without access limitations.

The limitations in accessing the sources make the issue of data extraction inherently complex and costly. Our experimentation have shown that the time needed for the extraction of all obtainable tuples can be quite long. On the other hand, the strategy of data extraction of IBIS, based on a concept of *proximity* of values to the tuples constituting the answer to the query, leads the system to retrieve tuples (and values) that are significant for the answer in a time that is usually very short, compared to the total extraction time. Since the results are presented to the user as soon as they are found, the user can decide that he is satisfied by the answers he has received so far, and stop the query answering process.

4.2 Static Optimization

Before starting the construction of the retrieved source database for a query, IBIS is capable of excluding from the extraction process those sources that are not useful for answering the query. In fact, in general not all sources are useful in the extraction process, in the sense that they can provide either tuples that contribute to the answer, or values used for the extraction of tuples that contribute to the answer. Moreover, some useless source may additionally degrade performance by providing values which generate bindings that cause more useless accesses. Techniques for selecting relevant sources have been proposed [9], however they are not applicable to IBIS because they cannot deal with the query language IBIS adopts, which is union of conjunctive queries. Instead, IBIS performs the following optimizations on the data extraction process.

First of all, IBIS checks whether the query that has to be issued to the sources is such that each of the values appearing in each atom covers all the attributes that have to be bound to a value in the corresponding source. In this case, the query can be answered without any complex extraction process, just as if there were no limitations in accessing the sources.

Then IBIS excludes a priori from the construction of the optimized query plan. all sources that are not queryable. A source is said to be *queryable* if it can be accessed at least once for at least one instance of at least one source database, starting from the values in the query. It can be shown that the method to calculate the queryable sources presented in [9] is still applicable in IBIS.

Finally, in order to exclude sources that are not useful for query processing, IBIS performs an optimization by exploiting its knowledge about which sources can provide values to which other sources. It starts from the sources appearing in the query, and, by using a dependency graph, it tracks back such information to identify the set of sources that potentially can provide values for the sources appearing in the query.

4.3 Dynamic Optimization

IBIS is capable of avoiding useless accesses to the sources by exploiting already extracted tuples and integrity constraints on the sources. Dynamic optimization based on integrity constraints comes into play when a data source is accessible in several ways, i.e., the same underlying data can be accessed with different limitations. The most relevant case is that of Web sources, where the same form can be submitted by filling in different sets of fields, but not by leaving all fields empty. The different access patterns for a source s are represented in IBIS as different sources $s_1, \ldots, s_n$ with different access limitations. To capture the fact that the sources $s_1, \ldots, s_n$ have the same extension, full inclusion dependencies $s_1 \subseteq s_2$, $s_2 \subseteq s_3$, $\ldots$, $s_{n-1} \subseteq s_n$, $s_n \subseteq s_1$ are used. More generally, the situation in which the extension of a source s is contained in that of another source s' is captured by the full inclusion dependency $s \subseteq s'$. Note that the abstract domains of s and s' must match.

Full inclusion dependencies, together with functional dependencies (which capture also key constraints), allow IBIS to performs runtime optimization during data extraction, taking into account the tuples already extracted from the sources [2].

5 Query Processing

Query processing in IBIS is separated in three phases: (1) the query is *expanded* to take into account the integrity constraints in the global schema; (2) the atoms in the expanded query are *unfolded* according to their definition in terms of the mapping, obtaining a query expressed over the sources; (3) the expanded and unfolded query is *executed* over the retrieved source databases, to produce the answer to the original query (see Section 4).

Query unfolding and execution are the standard steps of query processing in GAV data integration systems, while the *expansion* phase is the distinguishing feature of the IBIS query processing method. Indeed, to the best of our knowledge, no GAV data integration system exploits integrity constraints in order to overcome incompleteness of data at the sources. IBIS instead takes into account the integrity constraints over the global schema, which reflect the semantics of the application domain, and allows for the retrieving of data that could not be obtained in traditional data integration systems. This is done by encoding information about integrity constraints in the expanded query, so that the answers provided by evaluating the preprocessed queries are the best possible ones that can be obtained, given the available information.

In the following, let $\mathcal{I}$ be a data integration system and $\mathcal{D}$ a source database. In order to show how integrity constraints in the global schema can be taken into account, we make use of the notion of *retrieved global database* for a query q. Such a database is obtained by populating each relation r in the global schema according to the retrieved source database $\mathcal{D}_q$ for q and the mapping, i.e., by populating r with the tuples obtained by evaluating the associated query $\rho(r)$ on $\mathcal{D}_q$. Note that integrity constraints are not taken into account in such a construction.

In general, integrity constraints may be violated in the retrieved global database. Regarding key constraints, IBIS assumes, as mentioned before, that the query that the mapping associates to a global schema relation r is such that the data retrieved for r do not violate the key constraint of r. In other words, the management of key constraints is left to the designer.

On the other hand, the management of foreign key constraints cannot be left to the designer, since it is strongly related to the incompleteness of the sources. Moreover, since foreign keys are interrelation constraints, they cannot be dealt with in the GAV mapping, which, by definition, works on each global relation in isolation. Indeed, IBIS provides full support for handling foreign key constraints, which we now explain in detail.

The assumption of sound views asserts that the tuples retrieved for a relation r are a subset of the tuples that the system assigns to r; therefore, we may think of completing the retrieved global database by suitably adding tuples in order to satisfy foreign key constraints, while still conforming to the mapping. When a foreign key constraint is violated, there are several ways of adding tuples to the retrieved global database to satisfy such a constraint. In other words, in the presence of foreign key constraints in the global schema, the semantics of a data integration system must be formulated in terms of a *set* of databases, instead of a single one.

Since we are interested in the certain answers $q^{\mathcal{I},\mathcal{D}}$ to a query q, i.e., the tuples that satisfy q in all global databases that are legal for $\mathcal{I}$ wrt $\mathcal{D}$, the existence of several such databases complicates the task of query answering. To deal with this problem, IBIS expands the query q by taking into account the foreign key constraints on the global relations appearing in the atoms. For the details of the expansion process we refer to [4, 3]. The expansion $exp_{\mathcal{G}}(q)$ of q is a union of conjunctive queries, and it is possible to show that the evaluation of $exp_{\mathcal{G}}(q)$ over the retrieved source database produces exactly the set of certain answers of q to $\mathcal{I}$ wrt $\mathcal{D}$ [4]. As the construction of the retrieved global database is computationally costly, the IBIS Expander module does not construct it explicitly. Instead, it unfolds $exp_{\mathcal{G}}(q)$ and evaluates the unfolded query $unf_{\mathcal{M}}(exp_{\mathcal{G}}(q))$ over the retrieved source database, whose data are extracted by the Extractor module. As shown in [4], this produces exactly the same results.

6 Conclusions

To the best of our knowledge, IBIS is the only system for the semantic integration of heterogeneous data sources capable of exploiting integrity constraints over the global schema and the sources to improve query answering. IBIS has been released as a beta version and is under active development. We are working on extending the system in various directions, from the theoretical point of view and for the implementation.

References

1. S. Bergamaschi, S. Castano, M. Vincini, and D. Beneventano. Semantic integration of heterogeneous information sources. *Data and Knowledge Engineering*, 36(3):215–249, 2001.
2. A. Calì and D. Calvanese. Run-time optimization of query planning with limited source capabilities. In *Proc. of SEBD 2001*, pages 33–44, 2001.
3. A. Calì, D. Calvanese, G. De Giacomo, and M. Lenzerini. Accessing data integration systems through conceptual schemas (extended abstract). In *Proc. of SEBD 2002*, 2002.
4. A. Calì, D. Calvanese, G. De Giacomo, and M. Lenzerini. Data integration under integrity constraints. In *Proc. of CAiSE 2002*, 2002.
5. D. Florescu, A. Y. Levy, I. Manolescu, and D. Suciu. Query optimization in the presence of limited access patterns. In *Proc. of ACM SIGMOD*, pages 311–322, 1999.
6. C. H. Goh, S. Bressan, S. E. Madnick, and M. D. Siegel. Context interchange: New features and formalisms for the intelligent integration of information. *ACM Trans. on Information Systems*, 17(3):270–293, 1999.
7. J. Hammer, H. Garcia-Molina, J. Widom, W. Labio, and Y. Zhuge. The Stanford data warehousing project. *IEEE Bull. on Data Engineering*, 18(2):41–48, 1995.
8. M. Jarke, M. Lenzerini, Y. Vassiliou, and P. Vassiliadis, editors. *Fundamentals of Data Warehouses*. Springer, 1999.
9. C. Li and E. Chang. Query planning with limited source capabilities. In *Proc. of ICDE 2000*, pages 401–412, 2000.
10. C. Li and E. Chang. On answering queries in the presence of limited access patterns. In *Proc. of ICDT 2001*, pages 219–233, 2001.
11. C. Li, R. Yerneni, V. Vassalos, H. Garcia-Molina, Y. Papakonstantinou, J. D. Ullman, and M. Valiveti. Capability based mediation in TSIMMIS. In *Proc. of ACM SIGMOD*, pages 564–566, 1998.
12. Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object exchange across heterogeneous information sources. In *Proc. of ICDE'95*, pages 251–260, 1995.
13. M. Tork Roth, M. Arya, L. M. Haas, M. J. Carey, W. F. Cody, R. Fagin, P. M. Schwarz, J. T. II, and E. L. Wimmers. The Garlic project. In *Proc. of ACM SIGMOD*, page 557, 1996.
14. J. D. Ullman. Information integration using logical views. In *Proc. of ICDT'97*, volume 1186 of *LNCS*, pages 19–40. Springer, 1997.
15. J. Widom (ed.). Special issue on materialized views and data warehousing. *IEEE Bull. on Data Engineering*, 18(2), 1995.
16. G. Zhou, R. Hull, R. King, and J.-C. Franchitti. Data integration and warehousing using H20. *IEEE Bull. on Data Engineering*, 18(2):29–40, 1995.
17. G. Zhou, R. Hull, R. King, and J.-C. Franchitti. Using object matching and materialization to integrate heterogeneous databases. In *Proc. of CoopIS'95*, pages 4–18, 1995.