IBIS: Semantic Data Integration at Work

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

[1] Università di Roma "La Sapienza"
Dipartimento 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. In this paper we present IBIS (*Internet-Based Information System*), a system for the semantic integration of heterogeneous data sources, which adopts innovative and state-of-the-art solutions to deal with all aspects of a complex data-integration environment, including query answering under integrity constraints and limitations on source access. IBIS is based on the global-as-view approach, using a relational mediated schema to query the data at the sources. Sources are wrapped so as to provide a relational view on them. A key issue is that the system allows the specification of integrity constraints (modeling constraints in the domain of interest) in the global schema. Since sources are autonomous, the extracted data in general do not satisfy the constraints. IBIS adapts and integrates the data extracted from the sources making use of the constraints in the global schema, so as to answer queries at best with the information available. IBIS deals with limitations in accessing data sources, and exploits techniques developed for querying sources with access limitations in order to retrieve the maximum set of answers. In particular, it may use integrity constraints available on the sources to improve the efficiency of the extraction process.

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 how data are structured at the sources and how they are to be reconciled in order to answer queries. 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. IBIS adopts innovative and state-of-the-art solutions to deal with all aspects of a complex data integration environment, including query answering under integrity constraints, limitations on source access, and source wrapping. Despite there are several mediation systems for data integration (see e.g., [7,6,11,18,22,21,12,9,1]), IBIS is the first system that fully exploits

J. Eder and M. Missikoff (Eds.): CAiSE 2003, LNCS 2681, pp. 79–94, 2003.

all available information (including integrity constraints) for query answering. Thus, to the best of our knowledge, IBIS is the first system actually devoted to semantic data integration.

The problem of designing effective data integration systems has been addressed by several research and development projects in the last years. Data integration systems 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, called *global-as-view* (or simply GAV) and *local-as-view* (or simply LAV) [19,10,14]. In the GAV approach, a view over the sources is associated to each element of the global schema, describing how to populate such an element using the data at the sources. Most data integration systems adopt the GAV approach, e.g., TSIMMIS [7], Garlic [6], COIN [9], *Squirrel* [23,22], and MOMIS [1].

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 (modeling constraints in the domain of interest) in the global schema. Since sources are autonomous and incomplete, the extracted data in general do not satisfy the constraints. To deal with this characteristic, IBIS adapts and integrates the data extracted from the sources making use of the constraints in the global schema, so as to answer queries at best with the information available. In this way, the intensional information in the constraints over the global schema allows one to obtain additional answers that would not be provided by the standard unfolding strategy associated with GAV data integration systems. Indeed, current 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 [19]. The reason why unfolding is sufficient in those systems is that the GAV mapping essentially specifies a single database conforming to the global schema. Instead, due to the presence of integrity constraints over the global schema there are several potential global databases conforming to the data in the sources, and hence query answering has to deal with a form of incomplete information [20,3,4].

A characterizing aspect of IBIS is the ability to deal with limitations in accessing data sources, and in particular Web sources, e.g., those requiring filling at least one field in a form. IBIS exploits and implements techniques developed for querying sources with binding patterns in order to retrieve the maximum set of answers [17,8,15,16]. Since the extraction process is the major bottleneck in the integration of data over the Web, specific optimization techniques have been developed. These allow one to take into account intentional knowledge holding

on the sources (in particular, integrity constraints) to limit the number of source accesses.

In this paper we give an overview of IBIS, showing how the recent theoretical results on query answering and optimization have been implemented in the system. In particular, we first illustrate the data integration framework adopted in IBIS. We then describe the query processing phase. After a brief overview of the system architecture, we give some details on the data extraction techniques that have been crucial for the actual deployment of the system. Finally, we discuss the mechanisms provided by IBIS for the user interaction, and we conclude the paper.

2 Framework for Data Integration in IBIS

The formal framework of 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. Given a database $\mathcal{DB}$ and a relation symbol r, we denote with $r^{\mathcal{DB}}$ the set of tuples associated to r in $\mathcal{DB}$. In IBIS, we deal with four kinds of constraints (the notion of satisfaction is the usual one for the first three):

1. *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.
2. *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.
3. *Functional dependencies.* A functional dependency over a relation r has the form $r : \mathbf{A} \rightarrow \mathbf{B}$, where $\mathbf{A}$ and $\mathbf{B}$ are subsets of the set of attributes of r.
4. *Simple full-width inclusion dependencies.* A simple full-width inclusion dependency between two relations r_1 and r_2 is denoted by $r_1 \subseteq r_2$; it is satisfied in a database $\mathcal{DB}$ if $r_1^{\mathcal{DB}} \subseteq r_2^{\mathcal{DB}}$.

A data integration application in IBIS is modeled through a triple $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where

- $\mathcal{G}$ is the *global schema*, expressed as a relational schema with key and foreign key constraints.
- $\mathcal{S}$ is the *source schema*, constituted by one relation for each source. The schema of each source relation is a relational schema with simple full-width inclusion dependencies and functional dependencies.
- $\mathcal{M}$ is the *mapping* between $\mathcal{G}$ and $\mathcal{S}$. 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. The query $\rho(r)$ is expressed in the language of union of conjunctive queries, and specifies how to retrieve the data satisfying r in terms of a view over the sources. In fact, such a query is also *annotated* by a description of

the additional processing to be carried out on the data retrieved in order not to violate the key constraint of r using a technique similar to that in [5]. That is, IBIS currently assumes that it is the responsibility of the designer to specify suitable data cleaning methods in such a way as to guarantee that the data retrieved for r satisfies its key constraint.

Finally, queries over the global schema are also unions 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. Although other assumptions are possible [14], the sound views assumption is usually considered the most natural in the context of data integration [10].

Given a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ 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 system 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, which indeed needs to deal with incomplete information [20]. In particular, IBIS aims at computing the *certain answers* of the query. 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 $\mathcal{D}$.

3 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 database, to produce the answer to the original query (see Section 5).

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. IBIS takes fully into account the integrity constraints over the global schema, which reflect the semantics of the application domain, and allows for retrieving *all* data that belong to the certain answer.

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 are 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, 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 in an automated way.

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. The expansion technique exploits the fact that foreign key constraints can be rewritten as Datalog programs with suitable Skolem functions in the head and is based on partial evaluation of logic programs, see [3,4] for details. 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]. Notably, the expanded query can be exponential in the original query and the foreign key constraints, however it can still be evaluated in polynomial time in the size of the data. As the construction of the retrieved global database is computationally costly, in IBIS it is not constructed explicitly. Instead, $exp_{\mathcal{G}}(q)$ is unfolded and the unfolded query $unf_{\mathcal{M}}(exp_{\mathcal{G}}(q))$

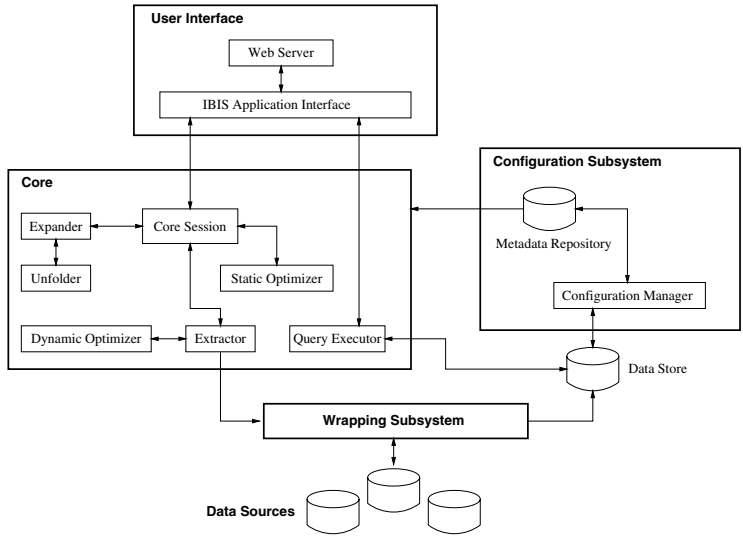

Fig. 1. Architecture of IBIS

is evaluated over the retrieved source database, whose data are extracted by an Extractor module (see next section). As shown in [4], this produces exactly the same results. Observe that in this way the query expansion is decoupled from the rest of the processing.

4 Architecture of IBIS

The system architecture of IBIS is shown in Figure 1. Four subsystems can be identified:

- the *wrapping subsystem*, which provides a uniform layer for all the data sources by presenting each source as a set of relations.
- the *configuration subsystem*, which supports system management and configuration of all the meta-data;
- the *IBIS core*, which implements the actual data integration algorithms and controls all the parts of the system;
- the *user interface*, which is divided in a Web interface and an application interface.

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. We detail below the wrapping subsystem and the IBIS core, which are the distinguishing elements of the IBIS architecture. The user interface and the interaction with the user are described in Section 6.

Wrapping Subsystem. The task of the wrapping subsystem is to provide a layer in which all data stored at the sources are presented to the other components of the system in a uniform way. Therefore, each component of IBIS sees the sources represented in the relational model. The wrappers in IBIS also take into account the limitations in accessing the sources; in fact, certain sources require a set of fields to be bound to constants in order to be accessed. A typical example is that of data accessible through Web forms, in which at least one field has to be filled with a value. Except for access limitations, wrappers do not need to expose any specific source behaviour. A set of properties, which can be configured by means of the configuration subsystem, allows the designer to specify the behavior of the wrapper according to source-dependent parameters such as throughput or reliability. Wrappers accept multiple requests which are buffered in a queue; thus the wrapping subsystem works asynchronously: each request is managed assigning it a wrapper taken from a pool. Several wrappers can work independently according to the capabilities of the source and of the server system.

IBIS Core. The IBIS Core is the set of components that take care at runtime of all the aspect of query processing. User queries are issued to the IBIS core by the application interface; the core performs evaluation of a query by (1) extracting data from the sources and (2) executing the query over such extracted data. *Data extraction*, which in IBIS is quite sophisticated because each source may present access limitations, is discussed in detail in Section 5. *Query processing* is performed according to the technique discussed in Section 3; an important feature is that the Expander module, which computes the expanded query, can operate independently from the Unfolder and the Executor modules, which respectively unfold the expanded query and evaluate it over the retrieved source database.

5 Data Extraction

The extraction of the data from the sources to build the retrieved source database for a given query 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. To improve efficiency in data extraction IBIS exploits specific techniques to deal with access limitations, and implements several types of optimizations to avoid accesses that would produce already retrieved data. In the following we describe in some detail these features of IBIS.

5.1 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 [17,8,16]. IBIS

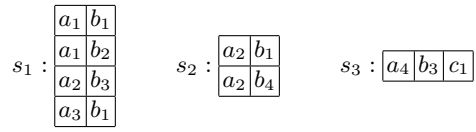

Fig. 2. Extension of sources of Example 1

exploits techniques developed specifically for dealing with access limitations [16], and extends them to make them deployable in practice. The extraction of data according to such techniques 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 in the data store.

In the following, without loss of generality, we assume that for each attribute of a relation an *abstract domain* with the same name is defined. An abstract domain is based on an underlying concrete domain, but represents information at a higher level of abstraction, e.g., to distinguish, strings representing person names from strings representing plate numbers. We call *binding tuple* a tuple of values that match with the attributes that must be bound to values; we call *binding values* the values of a binding tuple. For example, for a source $s(A^\star, B^\star, C)$, where the attributes that must be bound are starred, binding tuples are pairs of values (a, b), where a and b belong to the abstract domains D_A, D_B respectively, and D_A and D_B in turn characterize attributes A, B respectively.

Example 1. Consider the following source relations:

$$s_1(A^\star, B)$$
$$s_2(A, B^\star)$$
$$s_3(A, B^\star, C)$$

where, for the sake of simplicity, we have the same attribute names A, B, C for all attributes that belong to the abstract domains D_A, D_B, D_C respectively. Suppose we have the following conjunctive query over the sources:

$$q(C) \leftarrow s_1(a_1, B),\ s_3(A, B, C)$$

Now, assume the sources have the extension shown in Figure 2. Starting from a_1, the only constant in the query, we access s_1 getting the tuples (a_1, b_1) and (a_1, b_2). Now we have b_1 and b_2 with which to access s_2 and s_3; from s_2 we get (a_2, b_1), while from s_3 we get nothing. With the new constant a_2 we access s_1 getting (a_2, b_3). Finally, we access s_3 with b_3 getting (a_4, b_3, c_1) (with b_3 we do not get any tuple from s_2). At this point, we have populated the retrieved source database, on which we evaluate the query. The answer to q is therefore the tuple

(c_1). Observe that (a_3, b_1) and (a_2, b_4) could not be extracted from s_1 and s_2 respectively.

Although the extraction algorithm is straightforward, in order to make it efficient in practice, its implementation requires to take into account several technological aspects. The way the data extraction process is realized in IBIS is depicted in Figure 3, where the following elements can be identified:

- The *retrieved source database* (RSD) stores tuples retrieved during the data extraction for a certain query. It consists of physical tables, one for each source table defined in the source schema of IBIS.
- *Domain tables* store values that are used to produce binding tuples. There is one table for each abstract domain, containing all the values belonging to it. The values in the domain tables are contained in the set of values stored in the retrieved source database. The domain tables, although containing redundant data, are kept for efficiency reasons; indeed, experimental results have shown that the time needed for the generation of the binding tuples decreases significantly when domain tables are used.
- *Binding tables* are used to store binding tuples before submitting them to the sources; there is a binding table for each source with limitations.

To avoid wrappers to be overloaded with a number of binding tuples (i.e., access requests) that exceeds the capacity of the wrappers, they are fed with batches of binding tuples that do not exceed a prefixed maximum size. For each wrapper, according to its capabilities, the system manager assigns the maximum size 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 binding tuples 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 binding tuples for each wrapper.

Also, the new values in the tuples that are stored in the retrieved source database are not immediately "poured" in the domain tables, so as not to cause an excessive production of binding tuples: the transfer (see the arrow labeled with "Leaking" in the figure), controlled by the Core, is such that the values are

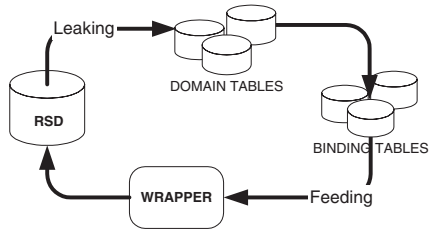

Fig. 3. Extraction process in IBIS

homogeneously distributed among the different abstract domains to which they belong.

The limitations in accessing the sources make the issue of data extraction inherently complex and costly. Our experimentations have shown that the time needed for the extraction of all obtainable tuples can be quite long. On the other hand, experiments have also shown that the system retrieves tuples (and values) that are significant for the answer in a time that is usually very short, compared to the total extraction time. This is due to the recursive nature of the extraction process, which obtains new values from the already retrieved ones; hence, a lower number of steps is required to obtain values extracted earlier, and these values have shown to be more likely part of the answer to the query.

5.2 Static Optimization

In general, having extracted a number of values at a certain point of the query answering process, and given a source s to be accessed using the values extracted so far as binding values, not all the possible accesses to s are necessary in order to calculate the answer to the query. This is illustrated in the following example.

Example 2. Let $\mathcal{S}$ be a source schema with $\mathcal{S} = \{s_1, s_2, s_3\}$; the sources are defined as follows:

$$s_1(A^\star, B) \qquad s_2(B^\star, C) \qquad s_3(C^\star, B)$$

For simplicity, suppose again we have distinct abstract domains D_A, D_B, D_C, one for each attribute name. Consider the following query:

$$q(C) \leftarrow s_1(a_0, B), s_2(B, C)$$

We easily observe that it is not useful to use the values obtained from s_2 to access s_3 in order to obtain new values of D_C with which to access s_2 again. In fact, due to the join condition between s_1 and s_2, the only tuples extracted from s_2 which can be used to construct a tuple of the answer to q are those obtained by binding the attribute B of s_2 with a binding value extracted from s_1.

In order to avoid unnecessary accesses, IBIS incorporates the optimization techniques presented in [2]. The optimization is as follows. Given an unfolded query on the sources, in order to optimise the query plan, information about the structure of the query and access limitations on the sources is encoded in a *dependency graph*. Intuitively, such a graph represents dependencies among sources, i.e., for any source s, it shows the sources that may provide binding values that are useful to access s. The dependency graph is pruned, taking into account the join conditions in the query, so that only necessary dependencies are left; the pruning procedure is performed in time polynomial in the size of the graph. From the pruned dependency graph, an optimized query plan is derived, which guarantees that only necessary accessed are performed during its execution.

Notice that the static optimisation of [2] is applicable for conjunctive queries, while an expanded and unfolded query in IBIS is a union of conjunctive queries. To this regard, IBIS offers the system manager two different strategies for processing a UCQ. The CQs can be either processed one by one, as if they were independent, or they can be *chained* in an ordered sequence, so that, in the extraction process for a CQ q, we can use as binding values the values extracted while processing the CQs preceding q in the chain.

5.3 Dynamic Optimization

The Dynamic Optimizer of 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 (see for example Amazon[1] or the ACM Sigmod Antology[2]). 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, simple full-width 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 simple full-width inclusion dependency $s \subseteq s'$. Note that the abstract domains of s and s' must match.

Simple full-width 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. We introduce the technique adopted in IBIS with an example.

Example 3. Consider two sources

$$s_1(\mathit{Code}, \mathit{Surname}, \mathit{City}^\star)$$
$$s_2(\mathit{Code}^\star, \mathit{Surname}^\star, \mathit{City})$$

where s_1 stores identification code, surname and city of birth of employees, and s_2 stores the same information about persons. Assume that the simple full-width inclusion dependency $s_1 \subseteq s_2$ holds and that the functional dependency $\mathit{Code} \rightarrow \mathit{Surname}, \mathit{City}$ holds on s_2.

Suppose that s_1 and s_2 have both the following extension:

Code	*Surname*	*City*
2	*brown*	*sidney*
5	*williams*	*london*
7	*yamakawa*	*tokyo*
9	*peretti*	*rome*

[1] `http://www.amazon.com/exec/obidos/ats-query-page/`

[2] `http://www.informatik.uni-trier.de/~ley/db/indices/query.html`

If our set of initial values is *rome* and *tokyo*, at the first step we access s_1 and we get the following tuples:

Code	*Surname*	*City*
7	*yamakawa*	*tokyo*
9	*peretti*	*rome*

Now we have four new values: the three codes 7 and 9, and the two surnames *yamakawa* and *peretti*. With these values we could access source s_1 to try and get new values. But we can easily observe that, because of the functional dependency on s_2, if we bind the *Code* attribute with one of the known values, we get a tuple we had already obtained from s_2. Therefore the access to s_1 is useless in this case. Instead, if we get 2 as a code and *brown* as a surname from another source, we could access s_1 and get new tuples.

To consider the general case, let $\mathbf{B}_1$ and $\mathbf{B}_2$ the set of attributes that must be bound in s_1 and s_2 respectively; let the dependency $s_1 \subseteq s_2$ hold. If a functional dependency $s_2 : \mathbf{C} \rightarrow \mathbf{D}$ holds, with $\mathbf{C} \subseteq \mathbf{B}_1$ and $\mathbf{D} \supseteq \mathbf{B}_2$, then if we access s_1 with a binding tuple ϑ such that $\vartheta = t[\mathbf{B}_1]$, where t is a tuple previously extracted from s_2, then the access with ϑ is useless, because it provides only tuples that have been already extracted from s_2 [2]. IBIS exploits this technique by selecting only the binding tuples that are potentially useful from the binding tables, just after their generation.

Another optimization is performed by IBIS when a key constraint holds on a source s. Let $\mathbf{K}$ be the key of s, with $\mathbf{K} \subseteq \mathbf{B}$, where $\mathbf{B}$ is the set of attributes of s that must be bound. Then, if we access s_1 with a binding tuple ϑ such that $\vartheta = t[\mathbf{B}]$, where t is a tuple previously extracted from s, then the access with ϑ is useless, because it provides only tuples that have been already extracted from s. This is again exploited by IBIS, by a suitable selection on the binding tuples.

6 Interaction with the User

IBIS is equipped with a user-friendly Web interface. In practice, the time required for answering a query may be significantly long; the bottleneck is constituted by the extraction phase, which has to cope with the usually very long response time of remote sources (Web sources and legacy systems) and with the intrinsic complexity of dealing with access limitations. Therefore, the traditional "submit-and-wait" interaction with Web-based systems is not suitable for IBIS. In order to offer the user a suitable form of interaction, IBIS has been designed with the following capabilities:

- the capability of incrementally presenting answers while they are computed;
- the capability of enhancing the query answering process by using additional data provided by the user together with the query;
- the capability of chaining queries to each other.

Incremental Generation of Answers. While one of the goals of IBIS is to provide the maximum set of answers, in practice this often requires an amount of time that could be unacceptable for a user operating in an interactive Web session. In order to cope with this problem, IBIS provides two strategies. The first one consists in showing tuples to the user as soon they are obtained, while the answering process is going on. In fact, the asynchronous extraction process allows evaluating the query over the source retrieved database before the end of the process itself. In this way, the user will see a continuous upgrade of the result set. Moreover, the user has the opportunity to stop the process at any time, when he is satisfied with the answers obtained so far. The second feature is the ability to continue the answering process also while a user is logged off, and present the obtained answers as soon as the user logs on again. E-mail and pager alerts are also available, to signal the user that a query has been completed.

Use of Domain-related Values. When a user query is processed, the set of constants appearing in the query is crucial, because at the beginning of data extraction such values represent the only way to access the sources. Therefore, adding values before starting the extraction process may significantly alter the extraction process itself. IBIS offers the user the possibility of expanding the set of initial values according to his knowledge of the domain of the global schema.

These constants influence the process in two ways: first, they may enlarge the set of tuples in the answer, because it is possible that the additional values lead to the generation of binding tuples for accessing the sources that would not be generated starting from the original set of values. Furthermore, in our experimentations of the system, the addition of domain-related values has shortened the time required for retrieving significant answers in most cases. This is due to the "proximity" that in many cases exists between the added values and the tuples in the answer. Obviously, the effectiveness of this feature depends on the user knowledge of the domain. Experiments have been carried out with non-expert users, unaware of the underlying sources, with data sources coming from the context of Government Institutions; the addition of initial values has proven to be useful in the majority of cases.

Chainable Queries. IBIS offers the possibility of using tuples extracted while answering a set of queries (the retrieved source databases of the queries) to answer another query related to the previous ones. When the freshness of data is not required by the user, the retrieved source databases obtained while answering previous queries can be seen as a cache for the new query. With this feature, queries can be *chained*, in the sense that each query uses the retrieved source databases of all the queries preceding it in the chain. IBIS is able to avoid producing binding tuples which have already been issued to the sources during the extraction of previous queries.

At the interface level, before submitting the query to the system, the user can choose if he wants to tie it to a particular set of already executed queries. At the end of the answering process he can also choose to save the extracted tuples or to discard them. This feature can also be used in a multi-user context:

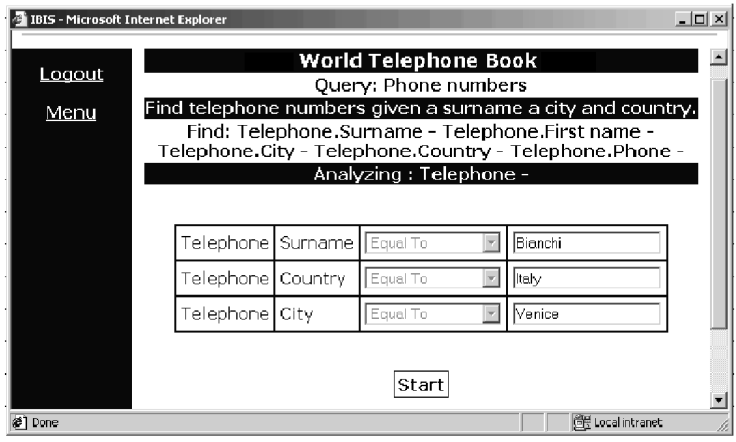

Fig. 4. Query interface in IBIS

the user who has issued a query q can allow a set of other users to use the data extracted from the sources while processing q.

Figure 4 shows a screen-shot of the IBIS Web interface to a stored query Phone numbers, and Figure 5 shows the result of evaluating such a query over a set of Web sources.

7 Conclusions

We have presented IBIS, a system for the semantic integration of heterogeneous data sources based on the GAV approach, adopting various innovative and state-of-the-art solutions to deal with source wrapping, source incompleteness, and limitations in accessing data sources. In particular, to the best of our knowledge, IBIS is the only data integration system capable of fully exploiting integrity constraints over the global schema and the sources in query answering.

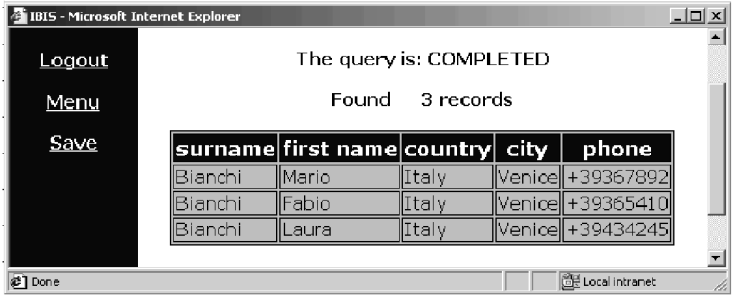

Fig. 5. Query result in IBIS

IBIS has been already released as a beta version and its final release is currently under active development. We are working on extending the system in various directions. In particular, we are studing techniques to deal with the problem of key constraint violations without requiring intervention of the designer. A first step in this direction is based on a weaker non-monotonic semantics for the mapping, based on suitable preference criteria in case of key constraint violations [13].

References

1. Sonia Bergamaschi, Silvana Castano, Maurizio Vincini, and Domenico Beneventano. Semantic integration of heterogeneous information sources. *Data and Knowledge Engineering*, 36(3):215–249, 2001.
2. Andrea Calì and Diego Calvanese. Optimized querying of integrated data over the Web. In *Proc. of the IFIP WG8.1 Working Conference on Engineering Information Systems in the Internet Context (EISIC 2002)*, pages 285–301. Kluwer Academic Publisher, 2002.
3. Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Accessing data integration systems through conceptual schemas. In *Proc. of the 20th Int. Conf. on Conceptual Modeling (ER 2001)*, pages 270–284, 2001.
4. Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Data integration under integrity constraints. In *Proc. of the 14th Conf. on Advanced Information Systems Engineering (CAiSE 2002)*, volume 2348 of *Lecture Notes in Computer Science*, pages 262–279. Springer, 2002.
5. Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Data integration in data warehousing. *Int. J. of Cooperative Information Systems*, 10(3):237–271, 2001.
6. M. J. Carey, L. M. Haas, P. M. Schwarz, M. Arya, W. F. Cody, R. Fagin, M. Flickner, A. Luniewski, W. Niblack, D. Petkovic, J. Thomas, J. H. Williams, and E. L. Wimmers. Towards heterogeneous multimedia information systems: The Garlic approach. In *Proc. of the 5th Int. Workshop on Research Issues in Data Engineering – Distributed Object Management (RIDE-DOM'95)*, pages 124–131. IEEE Computer Society Press, 1995.
7. Sudarshan S. Chawathe, Hector Garcia-Molina, Joachim Hammer, Kelly Ireland, Yannis Papakonstantinou, Jeffrey D. Ullman, and Jennifer Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *Proc. of the 10th Meeting of the Information Processing Society of Japan (IPSJ'94)*, pages 7–18, 1994.
8. Daniela Florescu, Alon Y. Levy, Ioana Manolescu, and Dan Suciu. Query optimization in the presence of limited access patterns. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 311–322, 1999.
9. Cheng Hian Goh, Stéphane Bressan, Stuart E. Madnick, and Michael D. Siegel. Context interchange: New features and formalisms for the intelligent integration of information. *ACM Trans. on Information Systems*, 17(3):270–293, 1999.
10. Alon Y. Halevy. Answering queries using views: A survey. *Very Large Database J.*, 10(4):270–294, 2001.
11. Joachim Hammer, Hector Garcia-Molina, Jennifer Widom, Wilburt Labio, and Yue Zhuge. The Stanford data warehousing project. *Bull. of the IEEE Computer Society Technical Committee on Data Engineering*, 18(2):41–48, 1995.

12. Matthias Jarke, Maurizio Lenzerini, Yannis Vassiliou, and Panos Vassiliadis, editors. *Fundamentals of Data Warehouses*. Springer, 1999.
13. Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Source inconsistency and incompleteness in data integration. In *Proc. of the 9th Int. Workshop on Knowledge Representation meets Databases (KRDB 2002)*. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-54/`, 2002.
14. Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002)*, pages 233–246, 2002.
15. Chen Li and Edward Chang. Query planning with limited source capabilities. In *Proc. of the 16th IEEE Int. Conf. on Data Engineering (ICDE 2000)*, pages 401–412, 2000.
16. Chen Li and Edward Chang. Answering queries with useful bindings. *ACM Trans. on Database Systems*, 26(3):313–343, 2001.
17. Chen Li, Ramana Yerneni, Vasilis Vassalos, Hector Garcia-Molina, Yannis Papakonstantinou, Jeffrey D. Ullman, and Murty Valiveti. Capability based mediation in TSIMMIS. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 564–566, 1998.
18. Yannis Papakonstantinou, Hector Garcia-Molina, and Jennifer Widom. Object exchange across heterogeneous information sources. In *Proc. of the 11th IEEE Int. Conf. on Data Engineering (ICDE'95)*, pages 251–260, 1995.
19. Jeffrey D. Ullman. Information integration using logical views. In *Proc. of the 6th Int. Conf. on Database Theory (ICDT'97)*, volume 1186 of *Lecture Notes in Computer Science*, pages 19–40. Springer, 1997.
20. Ron van der Meyden. Logical approaches to incomplete information. In Jan Chomicki and Günter Saake, editors, *Logics for Databases and Information Systems*, pages 307–356. Kluwer Academic Publisher, 1998.
21. Jennifer Widom (ed.). Special issue on materialized views and data warehousing. *Bull. of the IEEE Computer Society Technical Committee on Data Engineering*, 18(2), 1995.
22. Gang Zhou, Richard Hull, Roger King, and Jean-Claude Franchitti. Data integration and warehousing using H20. *Bull. of the IEEE Computer Society Technical Committee on Data Engineering*, 18(2):29–40, 1995.
23. Gang Zhou, Richard Hull, Roger King, and Jean-Claude Franchitti. Using object matching and materialization to integrate heterogeneous databases. In *Proc. of the 3rd Int. Conf. on Cooperative Information Systems (CoopIS'95)*, pages 4–18, 1995.