Query Planning in Infomaster

Oliver M. Duschka
duschka@cs.stanford.edu

Michael R. Genesereth
genesereth@cs.stanford.edu

Department of Computer Science
Stanford University
Stanford, CA 94305, USA

Abstract

Infomaster is an information integration system. It provides integrated access to distributed, heterogeneous information sources, thus giving its users the illusion of a centralized, homogeneous information system. Infomaster is the first such system that is able to handle arbitrary positive relational algebra user queries and database descriptions. It is able efficiently to use integrity constraints and local completeness information for optimization. The system has been deployed in a wide variety of application areas, including engineering, logistics, and electronic commerce. This article provides a much requested overview of the query processing method used by Infomaster.

1 Introduction

In recent years, there has been a dramatic growth in the number of publicly accessible databases on the Internet, and all indicators suggest that this growth should continue in the years to come. Unfortunately, retrieving information from these databases is not easy for several reasons.

The first complication is *distribution*. Not every query can be answered by the data in a single database. Useful relations may be broken into *fragments* that are distributed among distinct databases. In *horizontal* fragmentation, the rows of a database are split across multiple databases. In *vertical* fragmentation, the columns are split. Distributed databases can exhibit mixtures of these types of fragmentation.

A second complication in database integration is *heterogeneity*. This heterogeneity may be notational or conceptual. Notational heterogeneity concerns access language and protocol. One source may require SQL while another requires OQL and a third uses an ad hoc notation. This sort of heterogeneity can usually be handled through commercial products (such as the Sybase OpenServer). However, even if we assume that all databases use a standard language and protocol, there can still be conceptual heterogeneity, i.e. differences in relational schema and vocabulary. Distinct databases may use different words to refer to the same concept, and/or they may use the same word to refer to different concepts. Reassembling the distributed fragments of a database in the face of heterogeneity is doubly difficult.

Infomaster is an information integration tool that solves these problems. It provides integrated access to distributed, heterogeneous information sources, thus giving its users the illusion of a centralized, homogeneous information system.

Infomaster has been used in a variety of application areas, including engineering, logistics, and electronic commerce among others. The first application, a concurrent engineering system for the design of digital circuits, was completed in 1992. The most widely used application, the Stanford Information Network, has been in operation since 1995, providing public access to information about people, used goods, housing rentals, events, and so forth. Stanford University has recently embarked on a project aimed at expanding the use of Infomaster to integrate its thousands of campus databases. CommerceNet is investigating the use of Infomaster in integrating the catalogs of its 150+ member companies.

This article provides a much requested overview of Infomaster. Section 2 defines the description language used by the system; section 3 sketches the system's query processing algorithm; and section 4 presents results on guaranteed system behavior. The conclusion describes related work.

2 Language

In defining the language used by Infomaster, we start with a set of constants and a finite set of predicates. The *extension* of an n-ary predicate is the set of n-tuples of constants for which the predicate holds. Predicates can be divided into two types according to storage. *Extensional* predicates are those whose extensions are stored explicitly in at least one database. All other predicates are *intensional*. Predicates can also be classified according to definition. *Views* are predicates with definitions written in the definitional language described below. *Base predicates* are predicates without definitions. Note that, in much of database theory, views are intensional predicates. In our case, however, views are extensional while base relations are intensional.

We assume that all available data is stored in a finite

set of databases. A single database can store the extensions of multiple predicates. Moreover, the extension of a predicate can be stored in more than one database. Note that a database either stores a relation completely or does not store it at all. Fragmentation is dealt with by defining versions of the relations for each of the databases involved (hereafter called *site relations*).

We use a Datalog-like language to encode information about these relations. Database predicates and object constants are written in lower case; variables are written in upper case.

1. An *atom* is an expression of the form $\rho(\tau_1, \ldots, \tau_n)$, where ρ is an n-ary predicate and each τ_i is either a constant or a variable.

2. A *positive Boolean expression* is either an atom, an expression of the form $\phi \wedge \psi$ (i.e. a conjunction), or an expression of the form $\phi \vee \psi$ (i.e. a disjunction), where ϕ and ψ are positive Boolean expressions.

3. A *definition* is an expression of the form $\psi \equiv \phi$, where ψ is an atom and ϕ is a positive Boolean expression. ψ is called the *head* of the definition, and ϕ is called the *body*. Definitions are required to be *safe*, i.e. all variables appearing in the head must appear in the body. Variables appearing in the body but not in the head are assumed to be existentially quantified. The bodies of definitions are also required to be *uniform* with respect to the variables in the head, i.e. the disjuncts in every disjunction must contain the same head variables.

4. An *integrity constraint* is an expression of the form $\neg\phi$, where ϕ is a positive Boolean expression. Variables appearing in an integrity constraint are assumed to be universally quantified.

In Infomaster, we require that definitions be unique, i.e. if a predicate occurs in the head of one definition, it cannot occur in the head of any other definition. Note, however, that we allow Boolean expressions in the bodies of definitions, not just conjunctions as in Datalog.

We also require that definitions be non-recursive. If a predicate p occurs in the definition of q, then q cannot occur in the definition of p or any predicate used to define p and so forth.

These are strong restrictions. However, they still allow Infomaster to be used in a wide variety of applications. Most importantly, they allow us to give useful guarantees of system behavior that are impossible with more expressive languages.

The first use of this language is the definition of world relations in terms of other relations (regardless of which relations are extensional and which are intensional). For example, we can define the grandparent relation gp in terms of the parent relation p as shown below.

$$gp(X, Z) \equiv p(X, Y) \wedge p(Y, Z)$$

The second use of the language is to describe the contents of databases. As mentioned above, we do this by naming the relations actually stored within those databases (the site relations) and then defining those relations in terms of world relations in the same ways that views are defined. For example, assume that we have a database d_1 that stores those tuples of the unary relation r satisfying the unary relation s. We can capture this information by defining the site relation r_1 and writing the following definition for r_1.

$$r_1(X) \equiv r(X) \wedge s(X)$$

Partial information can be expressed by inventing new base relations (so-called *gensym* predicates) and using these new relations in the definitions of other relations. For example, we can express the fact that r_2 is contained in r by inventing a gensym predicate g and writing the following definition.

$$r_2(X) \equiv r(X) \wedge g(X)$$

By writing integrity constraints, we can describe what tuples are not present in a relation. For example, we can express the disjointness of relations p and q as follows.

$$\neg(p(X) \wedge q(X))$$

The expressiveness of Infomaster's description language allows us to define a wide range of horizontal and vertical fragments. One advantage of this approach is that different horizontal fragments can be described independently of other horizontal fragments, since each fragment can be described in separate rules.

3 Query Processing

Query processing in Infomaster consists of query planning and plan execution. The query planning algorithm takes as inputs a query Q and a collection Δ of rules and integrity constraints like those described in section 2. The output of the algorithm is a query processing plan suitable for input to the plan execution algorithm. The plan execution algorithm takes a query plan as input. It retrieves data from the available databases and merges this data as described in the plan. The output is a table of answers to the original query.

In this section, we present an informal sketch of the five steps in the query planning algorithm.

1. Reduction. In this step, the system rewrites each atom in the query using the definition of the associated predicate. It then repeats the process until it obtains an expression in terms of base relations (which, by definition, have no definitions of their own.) Termination is assured due to the non-recursive nature of the definitions.

2. Abduction. Given an expression Q_b in terms of base relations and a set of definitions, the abduction process produces the set R of all consistent and minimal conjunctions of retrievable atoms that can be shown from the definitions to be contained in Q_b.

3. Conjunctive Minimization. In this step, Infomaster eliminates any redundant conjunct, i.e. one that can be shown to contain the remaining conjuncts.

4. Disjunctive Minimization. In this step, Infomaster drops any disjunct that can be shown to be contained in the union of the remaining disjuncts.

5. Grouping. Finally, the conjuncts within each conjunction are grouped so that the atoms in each group all share a common provider.

For simplicity, we have described these steps as taking place sequentially. In the implementation, some of the steps are interleaved. For example, conjunctive minimization is interleaved with abduction. This saves time by curtailing further work on a conjunction once an inconsistency has been detected.

4 Guarantees

We say that a plan R is *semantically correct* with respect to a query Q if every answer to R is an answer to Q. R is *source-complete* if every retrievable answer to Q will be retrieved using R. A plan is *locally minimal* if it is not possible to drop any conjunct or disjunct without losing or gaining answers in some distribution of data compatible with the rules and integrity constraints in the knowledge base. The Infomaster planning algorithm produces plans that are semantically correct, source-complete, and locally minimal.

A plan is *globally minimal* if there is no semantically correct and source-complete plan with fewer groups. It is possible to extend Infomaster to produce globally minimal plans by iterating over possible plans; however, the cost of doing this is substantial, and its benefits are unclear in practice.

Given the restrictions on the form of definitions and integrity constraints, it is possible to show that the algorithm always terminates. Unfortunately, the problem of finding semantically correct and source-complete plans is at least NP-hard; and so the worst case processing time for a query can be exponential in terms of the number of atoms in the base expansion of the query.

5 Conclusion and Related Work

In this paper, we have presented the database description language and query planning algorithm used in Infomaster. The design of Infomaster builds upon extensive work in the field of information integration. Related efforts to integrate distributed information sources are the TSIMMIS project [CGMH+94], the Information Manifold project [LRO96], and the SIMS project [ACHK93].

TSIMMIS approaches the integration problem by defining a list of query-templates. For each template, a query plan expressing how to retrieve a query of this form is given. TSIMMIS finds predefined templates that match the user query, and executes the corresponding stored query plans. Obviously, query planning is very efficient in this approach, but the number of possible queries a user can ask is limited, and adding a new information source to the system requires recoding of all related query plans. In comparison, in Infomaster adding a new information source requires the addition of just one new description, viz. that of the newly added information source.

Like Infomaster, both SIMS and the Information Manifold utilize source descriptions and dynamically generate query plans for user queries. The description languages are more restricted though. In SIMS, for example, it is impossible to express the fact that an information source stores the grandparent relation *gp* if only the parent relation is a base relation. In the Information Manifold, it is possible to express the fact that an information source stores *gp*; but, if a user asks for *gp*, its query planning algorithm fails to retrieve *gp* anyway.

Qian presents an algorithm in [Qia96] to compute semantically correct and source-complete query plans. Qian's algorithm is restricted to conjunctive queries. The query planning algorithm used in Infomaster can be seen as a generalization of Qian's algorithm to arbitrary positive relational algebra queries. The authors generalize this result in [DG96] to recursive queries.

The problem of optimizing conjunctive queries was solved in [CM77], and the problem of optimizing unions of conjunctive queries was solved in [SY80]. However, the approaches given there are unable to handle integrity constraints. The Infomaster optimization algorithm fully incorporates integrity constraints stating that a positive relational algebra query has an empty answer. This is a broad class of integrity constraints, which includes disjointness, for example. Moreover, the Infomaster optimization algorithm makes use of local completeness information [EGW94, Dus96]. If an information source is known to store the *entire* extension of a predicate p, then no other information sources storing parts of p need to be queried.

References

[ACHK93] Yigal Arens, Chin Y. Chee, Chun-Nan Hsu, and Craig A. Knoblock. Retrieving and integrating data from multiple information sources. *International Journal of Intelligent & Cooperative Information Systems*, 2(2):127–58, 1993.

[CGMH+94] Sudarshan Chawathe, Hector Garcia-Molina, Joachim Hammer, Kelly Ireland, Yannis Papakonstantinou, Jeffrey Ullman, and Jennifer Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *Proceedings of the 100th Anniversary Meeting*, pages 7–18, 1994. Information Processing Society of Japan.

[CM77] Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proceedings of the Ninth Annual ACM Symposium on the Theory of Computing*, pages 77–90, 1977.

[DG96] Oliver M. Duschka and Michael R. Genesereth. Answering recursive queries using views. Technical Report 96-5, Logic Group, Department of Computer Science, Stanford University, 1996.

[Dus96] Oliver M. Duschka. Query optimization using local completeness. Technical Report 96-2, Logic Group, Department of Computer Science, Stanford University, 1996.

[EGW94] Oren Etzioni, Keith Golden, and Daniel Weld. Tractable closed world reasoning with updates. In *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning*, pages 178–189, 1994.

[LRO96] Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proceedings of the 22nd International Conference on Very Large Databases*, 1996.

[Qia96] Xiaolei Qian. Query folding. In *Proceedings of the 12th International Conference on Data Engineering*, pages 48–55, 1996.

[SY80] Yehoshua Sagiv and Mihalis Yannakakis. Equivalence among relational expressions with the union and difference operators. *J. ACM*, 27(4):633–655, 1980.