To appear in the Proceedings of the International Workshop on Intelligent Information Integration, Freiburg, Germany, September 1997.

Infomaster — An Information Integration Tool

Oliver M. Duschka
duschka@cs.stanford.edu
Department of Computer Science
Stanford University

Michael R. Genesereth
genesereth@cs.stanford.edu
Department of Computer Science
Stanford University

Abstract

We present the Infomaster system, an information integration tool developed and tested at Stanford University. The Infomaster system bridges differences in schemata and terminology between existing databases. This makes it possible to provide a uniform user interface to a collection of heterogeneous information sources. Information sources are described in an expressive language, the so-called knowledge interchange format (KIF). The crucial phase in the query planning process of the Infomaster system is based on a specialized model elimination theorem prover.

1 Introduction

Individuals, companies and governments store enormous amounts of information that is becoming available over the WWW, and over corporate and governmental networks. Possible usage of this information, however, remains limited as long as information is stored separately, without easy means to combine data from different sources. Clearly, individuals, companies and governments would benefit from an easy access to the world's knowledge and data, but this problem remains difficult due to several reasons.

The first complication is *distribution*. Not every query can be answered by the data available from a single information source. Useful information might be broken into fragments that are distributed among distinct sources. The second complication is *heterogeneity*. Different sources might require different access methods, like parsing of HTML pages, relational query languages like SQL, object-oriented query languages like OQL, or bibliography protocols like Z39.50. Moreover, there might be semantic mismatches between different sources. The same concept might be represented by different words. Even worse, the same word might refer to different concepts. Finally, the third complication is *instability*. New information sources appear every day, others disappear. Existing information sources change the format of their data, or change their content.

In an information world characterized by distribution, heterogeneity and instability, computer science researchers strive to provide programs that act as "intelligent agents". These agents should search for and find desired information, convert formats, translate different contexts, etc., in a fully automatic fashion. However, as the experience with today's WWW search engines shows, these ideal agents don't seem to be feasible yet. Considerable research in ontologies and natural language understanding seems to be required to fully automate the information integration task.

What is needed in the meantime are tools that make it easier to integrate information from distributed, heterogeneous information sources in an instable environment. The Infomaster system is such a tool. It provides users integrated access to distributed, heterogeneous information sources. Moreover, it makes it easy to manage evolving information sources, to add new information sources, and to remove outdated ones. In this paper, we give an overview of the Infomaster system.

2 Architecture

The Infomaster system is a generic information integration tool for integrating existing information sources. Information sources that can be integrated can vary from expressive SQL databases, over Z39.50 sources — a standard used for library information, to semistructured data that can be found on the WWW. Each type of information source requires a unique program, called *wrapper*, that translates between the language spoken by the information source and the language spoken by the core Infomaster system. The internal content language used by the Infomaster system is the so called *knowledge interchange format* (KIF), essentially a lisp like syntax for representing first order logic expressions.

3 Tested application areas

The Infomaster system was developed as a research project at Stanford University. It has been tested in a variety of application domains, some of which we are going to present here.

3.1 Newspaper classifieds

Several newspapers are published in the San Francisco Bay Area. All of them have rental and used car classified ads. We applied the Infomaster system to provide a uniform search interface to this information. For example, users can search for 2 bedroom apartments in Palo Alto, Menlo Park, or Portola Valley under $1000. The system then gathers the corresponding classifieds from all relevant newspapers.

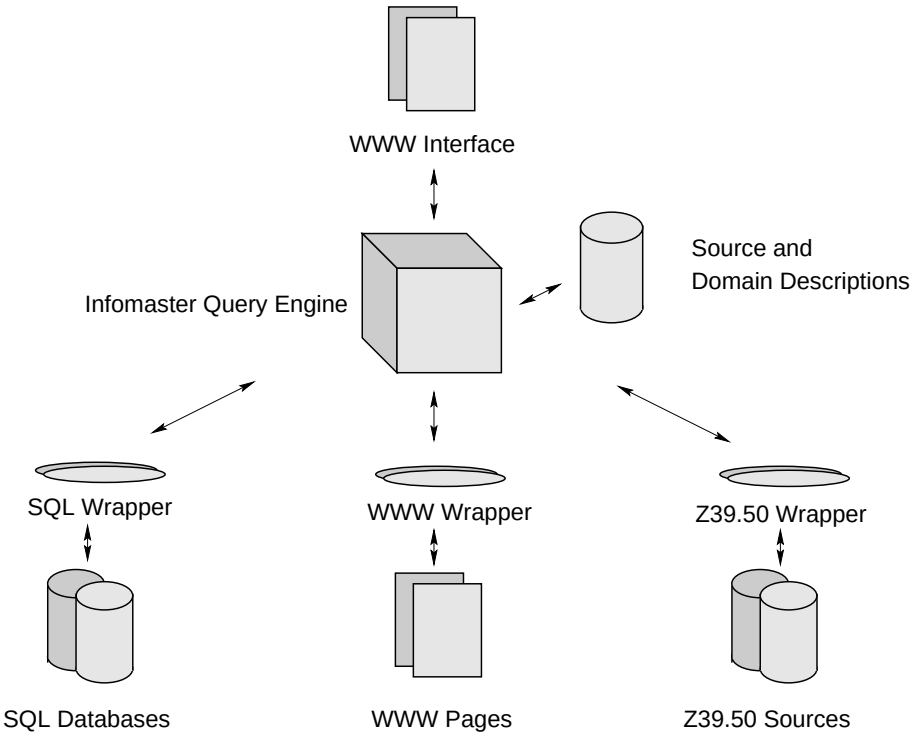

Figure 1: *Architecture of the Infomaster System*

3.2 Product catalogs

The Infomaster system has been deployed in integrating electronic product catalogs and catalogs for houseware items from several vendors. This application requires a lot of terminology translation. For example, one houseware items vendor refers to its version of Teflon as Maxalon X2000. Clearly, a user shouldn't be required to know all these details when searching for a non-sticking pan.

3.3 Campus databases

Stanford University itself has a wide collection of databases. We used Infomaster to provide a uniform interface to databases on people, courses, and library information.

4 Abstraction hierarchy

We model the user interface and the available information sources by a set of *relations*. The WWW forms that users can use to enter their queries are abstracted as so-called *interface relations*. Data available from an information source can also be seen as a relation, which we call *site relation*. The information integration problem can be reduced in this framework to the problem of relating the interface and the site relations in an appropriate way. Figure 2 shows an

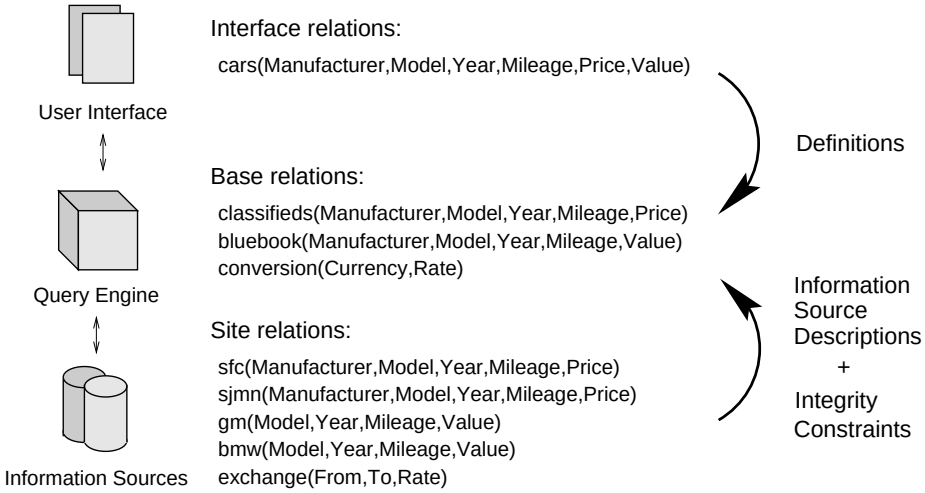

Figure 2: *The information integration problem is abstracted as describing the relationships between three kinds of relations: Interface relations conceptualize the interaction with a user through a WWW based user interface. Site relations represent the data that is actually stored in the available information sources. Base relations are used as means to describe both interface and site relations and are crucial in the query planning process.*

example of interface and site relations. The application domain in the example is the following: The San Francisco Chronicle and the San Jose Mercury News both contain a used car classifieds section in which cars are offered for sale. Moreover, we assume that we have access to information of car manufacturers General Motors and BMW. Both manufacturers provide data on the average market value of their cars for a given model, year, and mileage. Our goal is to provide a WWW interface to users that integrates all this information. Site relations sfc and $sjmn$ model the information parsed from the used car classifieds in the San Francisco Chronicle and the San Jose Mercury News respectively. Site relations gm and bmw represent the information available from the two car

manufacturers. Interface relation *cars* represents the WWW form presented to the user.

If we were sure that we would never add new information sources and that already integrated information sources never changed their content, then we could relate interface relations directly to site relations. However, we want to be more flexible in our design. For example, it is likely that we want to improve our service in the future by also including classifieds published in the Palo Alto Weekly, the Sacramento Bee, or the Los Angeles Times. In order to simplify adding new information sources and accommodating the changes in content of existing ones, we introduce a new level into our abstraction hierarchy. We call these new relations *base relations*. We will express both interface relations and site relations in the terms of base relations. This allows us to easily integrate new information sources. Also, we can easily change the user interface without having to integrate the information sources anew.

Base relations should be chosen to be the basic building blocks of the application domain at hand. In our example, there are basically two kinds of information: classified ads and general information on car models. We represent all classified ads by the base relation $classifieds$ and the market value information on car models by the base relation $bluebook$.

The example contains one further base relation called $conversion$ that provides the current exchange rate from a given currency into dollar. There is also an additional site relation $exchange$ that represents information provided by some currency exchange.

5 Descriptions of relationships

The previous chapter described how the information integration problem can be broken down into an abstraction hierarchy of site, base, and interface relations. Here we will show how these different abstraction levels can be related to each other. The descriptions of these relationships will be used by the query planner to translate user queries into queries that can be answered by the information sources, and to combine the resulting answers.

We describe both interface relations and site relations in terms of base relations. Interface relation *cars* combines information from classified ads with the average market value of the corresponding model from the *bluebook* base relation. This can be expressed in the following definition:

$$\begin{aligned} cars(Manufacturer, Model, Year, Mileage, Price, Value) \equiv \\ classifieds(Manufacturer, Model, Year, Mileage, Price) \\ \& \; bluebook(Manufacturer, Model, Year, Mileage, Value) \end{aligned}$$

Both the San Francisco Chronicle and the San Jose Mercury News provide classified ads. However, none of them publishes *all* classified ads. Therefore,

the sfc and $sjmn$ site relations are *contained* in (and not equivalent to) the $classifieds$ base relation.

$$sfc(Manufacturer, Model, Year, Mileage, Price) \Rightarrow$$
$$classifieds(Manufacturer, Model, Year, Mileage, Price)$$

$$sjmn(Manufacturer, Model, Year, Mileage, Price) \Rightarrow$$
$$classifieds(Manufacturer, Model, Year, Mileage, Price)$$

General Motors and BMW, on the other hand, do provide all information on their respective car models. Therefore, site relations gm and bmw are equivalent to the corresponding fragments of the $bluebook$ base relation. The distinction between containment and equivalence is important for the query optimization process. For example, even if there were another information source storing bluebook information for General Motors cars, it would be unneccessary to access both this information source and the original General Motors source because the original source is guaranteed to store all relevant information. On the other hand, a classified ad could be published in the San Francisco Chronicle or the San Jose Mercury News. None of the two information sources is complete.

$$gm(Model, Year, Mileage, Value) \equiv$$
$$bluebook(gm, Model, Year, Mileage, Value)$$

$$bmw(Model, Year, Mileage\_in\_km, Value\_in\_DM) \equiv$$
$$bluebook(bmw, Model, Year, Mileage, Value)$$
$$\& \; conversion(dm, Rate)$$
$$\& \; Mileage = Mileage\_in\_km * 1.6$$
$$\& \; Value = Value\_in\_DM * Rate$$

Information provided from the BMW information source is stored in km and DM instead of miles and Dollar. Therefore, the "mileage" in km has to be related to the mileage (in miles), and the market value in DM has to be related to the corresponding value in Dollar. For the second translation, the current exchange rate has to be obtained. Finally, the rule

$$exchange(From, dollar, Rate) \equiv$$
$$conversion(From, Rate)$$

expresses that the currency exchange provides data of exchange rates from a given currency into Dollar.

6 Query processing

Query processing in the Infomaster system is a three-step process. Assume the user asks a query q. This query is expressed in terms of interface relations. In a first step, query q is rewritten into a query in terms of base relations. We call

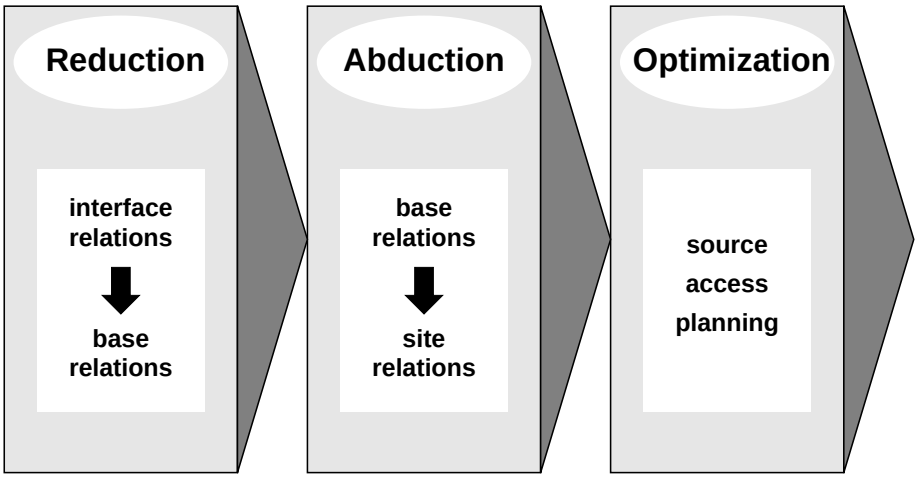

Figure 3: *The three steps of Infomaster's query planning process.*

this step *reduction*. In a second step, the descriptions of the site relations have to be used to translate the rewritten query into a query in terms of site relations. This second step is called *abduction*. The query in terms of site relations is an executable *query plan*, because it only refers to data that is actually available from the information sources. However, the generated query plan might be inefficient. Using the descriptions of the site relations, the query plan can be optimized.

As an example query, assume that a user asks for BMWs built in 1996 that are for sale for a price below their average market value:

$$q(Model, Mileage, Price) \equiv \\ cars(bmw, Model, 1996, Mileage, Price, Value) \\ \& \; Price < Value$$

We will discuss the three steps of the query planning process in the following.

6.1 Reduction

The reduction step in the query processing sequence is very simple. It is essentially a macro expansion. The user query is given in terms of interface relations, and interface relations are defined in terms of base relations. Therefore, the reduction step requires only to substitute interface relations by the corresponding definitions. In our example, the user query is rewritten to the following query in terms of base relations:

$$q(Model, Mileage, Price) \equiv$$
$$classifieds(bmw, Model, 1996, Mileage, Price)$$
$$\& \; bluebook(bmw, Model, 1996, Mileage, Value)$$
$$\& \; Price < Value$$

6.2 Abduction

The interesting step in the query processing sequence is the abduction step. It requires to translate a query in terms of base relations into a query in terms of site relations. This is more complicated than the reduction step, because site relations are expressed in terms of base relations and not vice versa[1]. This query rewriting problem is well known in the database literature as the problem of answering queries using views [LMSS95]. Several algorithms have been proposed to solve this problem [LRO96, Qia96, DG97a, DL97]. However, these algorithms are only designed to work for quite restricted query languages. In the following we are going to describe an algorithm for this problem based on the notion of *abduction*.

Definition 6.1 (**Abduction**) Given a logical theory Δ, a sentence ϕ, and a syntactical restriction r, the sentence ψ is an *abduct of* ϕ *under* Δ *and* r if it satisfies the following requirements:

(i) ψ satisfies the syntactic restriction r,

(ii) $\Delta \cup \{\psi\}$ is consistent, and

(iii) $\Delta \cup \{\psi\}$ logically implies ϕ.

□

The syntactic restriction that is needed for our application is that all relations in ψ are site relations. This is because only site relations are actually stored by the information sources. The logical theory Δ is the set of all descriptions of the site relations. Finally, ϕ is the rewritten user query after the reduction step.

Abduction is implemented in the Infomaster system using a standard model elimination theorem prover. In our example, abduction yields the following query plans:

$$q(Model, Mileage, Price) \equiv$$
$$sfc(bmw, Model, 1996, Mileage, Price)$$
$$\& \; bmw(Model, 1996, Mileage\_in\_km, Value\_in\_DM)$$
$$\& \; exchange(dm, dollar, Rate)$$

[1] It is of course possible to define base relations in terms of site relations. This would simplify the abduction step. However, adding an information source or accomodating the change in content of an information source then would become more difficult.

$$
\begin{aligned}
&\& \; Mileage = Mileage\_in\_km * 1.6 \\
&\& \; Value = Value\_in\_DM * Rate \\
&\& \; Price < Value
\end{aligned}
$$

$$
\begin{aligned}
q(Model, Mileage, Price) \equiv \\
&sjmn(bmw, Model, 1996, Mileage, Price) \\
&\& \; bmw(Model, 1996, Mileage, Value) \\
&\& \; exchange(dm, dollar, Rate) \\
&\& \; Mileage = Mileage\_in\_km * 1.6 \\
&\& \; Value = Value\_in\_DM * Rate \\
&\& \; Price < Value
\end{aligned}
$$

6.3 Optimization

Assume the San Francisco Chronicle would always be guaranteed to publish all classified ads that are published in the San Jose Mercury News. Then the second of the two query plans above would be redundant. In the optimization step, the Infomaster systems tries to eliminate redundant source accesses. In our example, the two query plans are already logically minimal. Before executing the plans, however, the Infomaster system will group the query plan in order to avoid querying the same information source twice. The resulting query plan is:

$$
\begin{aligned}
q(Model, Mileage, Price) \equiv \\
&(sfc(bmw, Model, 1996, Mileage, Price) \\
&\vee sjmn(bmw, Model, 1996, Mileage, Price)) \\
&\& \; bmw(Model, 1996, Mileage\_in\_km, Value\_in\_DM) \\
&\& \; exchange(dm, dollar, Rate) \\
&\& \; Mileage = Mileage\_in\_km * 1.6 \\
&\& \; Value = Value\_in\_DM * Rate \\
&\& \; Price < Value
\end{aligned}
$$

7 Conclusions and related work

We gave an overview of the Infomaster system, an information integration tool developed and tested at Stanford University. We presented the overall system architecture, some tested application areas, and showed how information sources can be described declaratively using an abstraction hierarchy of site, base, and interface relations. Finally, we explained the query planning process in Infomaster, consisting of a reduction, abduction, and optimization phase.

The use of abduction in the query planning process is very flexible in that it can handle an expressive description language, constraints, and background theories [DG97b]. The use of abduction can be streamlined, however, in the more restricted case that all descriptions are Horn clauses [DG97a]. In this case, query planning can be done in polynomial time, even if queries contain recursion.

Moreover, functional dependencies and restrictions on binding patterns can be taken into account easily [DL97]. The query optimization phase uses so-called local completeness information, which was introduced in [EGW94, EGW97], and applied to optimization in information integration in [Dus97].

The design of the Infomaster system builds upon extensive work in the field of information integration. Related efforts to integrate distributed information sources are the Information Manifold project [LRO96], the SIMS project [ACHK93], the Occam project [KW96], and the TSIMMIS project [CGMH+94]. The Information Manifold project and the SIMS project explore the use of descriptions logics for describing information sources. The Occam project uses general AI planning techniques to generate information gathering plans. Finally, the TSIMMIS project uses pattern matching techniques to match user queries against predefined queries with stored query plans.

Acknowledgments

We would like to express our thanks to the members of the Center for Information Technology at Stanford University for their continuous efforts in implementing, maintaining, and improving the Infomaster system.

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, June 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, Tokyo, Japan, October 1994. Information Processing Society of Japan.

[DG97a] Oliver M. Duschka and Michael R. Genesereth. Answering recursive queries using views. In *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS '97*, pages 109 – 116, Tucson, AZ, May 1997.

[DG97b] Oliver M. Duschka and Michael R. Genesereth. Query planning in Infomaster. In *Proceedings of the 1997 ACM Symposium on Applied Computing*, San Jose, CA, February 1997.

[DL97] Oliver M. Duschka and Alon Y. Levy. Recursive plans for information gathering. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, IJCAI*, Nagoya, Japan, August 1997.

[Dus97] Oliver M. Duschka. Query optimization using local completeness. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence, AAAI-97*, Providence, RI, July 1997.

[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, San Francisco, CA, June 1994.

[EGW97] Oren Etzioni, Keith Golden, and Daniel Weld. Sound and efficient closed-world reasoning for planning. *Journal of Artificial Intelligence*, 89(1–2):113–148, January 1997.

[KW96] Chung T. Kwok and Daniel S. Weld. Planning to gather information. In *Proceedings of the AAAI Thirteenth National Conference on Artificial Intelligence*, 1996.

[LMSS95] Alon Y. Levy, Alberto O. Mendelzon, Divesh Srivastava, and Yehoshua Sagiv. Answering queries using views. In *Proceedings of the 14th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, San Jose, CA, May 1995.

[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*, pages 251–262, Bombay, India, 1996.

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