A Metadata Based Approach to Improving Query Responsiveness

Ling Liu
Department of Computing Science
University of Alberta
GSB 615, Edmonton, Alberta
T6G 2H1 Canada
email: lingliu@cs.ualberta.ca

Calton Pu
Dept. of Computer Science and Engineering
Oregon Graduate Institute
P.O.Box 91000 Portland, Oregon
97291-1000 USA
email: calton@cse.ogi.edu

Abstract

We describe a rule-based approach to semantic specification that can be used to dynamically establish semantic relevance between consumers' queries posed on the fly and producers' data sources available online. The semantic specification of user queries is independent of the content and capabilities of the data sources (*query-to-source independence*). The semantic specification of the content and capability of a data source is independent of the content and capability descriptions of other sources (*source-to-source independence*). Query processing techniques use these specifications along with the query modification and transformation routines to perform relevance reasoning and to guarantee semantically correct query answers. This work also examines the effect of changing metadata semantics, such as changes in the content and capabilitiy descriptions of data sources or the user query profile specifications. Methods are described for detecting these changes and for determining if the information sources can continue to supply meaningful data to answering consumers' queries. These methods and transformation routines are necessary for determining logical connectivity between information producers and information consumers and for establishing dynamic interconnection between a growing collection of data sources and a consumer's query.

1 Introduction

The rapid growth of World Wide Web (WWW) technology and the increase in network speed and bandwidth have challenged the way we think and the way we obtain and exchange information. Everyone today can publish information on the web independently at any time. The flexibility and autonomy of producing and sharing information on WWW is phenomenal. On the other hand, one has to learn to deal with the rapid increase of volume and diversity of online information and the constant changes of information sources in number, content, and location.

Thus, queries to the current WWW search tools are mostly specified by simply typing in the keywords, the search tools will handle the request and find the sources that match the given keywords. However, the scalability and the dynamic interconnection between producers and consumers are achieved at the price of effectiveness of queries, namely the quality and the responsiveness of the answers, for several reasons. First, responses returned by WWW search tools often contain too much irrelevant information (noise). Second, queries in network-centric information systems are more vulnerable to failure due to congestion of the networks, traffic at the intermediate sites and contention at the sources. Thus it

happens frequently that one needs information from multiple data sources and is unable to get and fuse the information from data sources in a timely fashion.

From the data management point of view, the main complications in providing quality access in an open environment such as Internet are the following:

1. Data sources are autonomous and heterogeneous in nature and the number of sources available online are constantly growing. For example, different book stores or publishing companies may use different data formats for their book databases. Many online virtual bookstores are available on the net.

2. Information stored in autonomous data sources are often interrelated and possibly replicated. For example, a book can be purchased from multiple bookstores or bookclubs.

3. Most data sources contain incomplete information. For example, very few book stores or publishers can provide all books on `Cancer`.

4. Many data sources are not full-featured database systems and can answer only a small set of queries over their data.

5. Over time, not only may new data sources or applications need to be added to an already heterogeneous mix, but existing data sources may also change the specifications of the data they provide, and consumers may change the requirements for the data they request.

To deal with these problems, we believe that a global information system must have the ability to allow consumers to pose queries on the fly in the sense that *users can issue queries without knowing the structure, location, or existence of requested data.* It is not anymore realistic to maintain a pre-defined integrated world (global) view of all data sources due to the dynamic nature of the data sources online and the diversified business objectives of information consumers. Therefore, to provide scalable and high quality query services for accessing heterogeneous data sources, a distributed interoperable information system must have (1) enough information about the content and query capability of the data sources available online, namely source capability profiles (see definition in Section 3), to understand and exploit the relationships between their contents, and (2) a good knowledge of the semantic scope of the user query domain and the context of what users want to receive from a query, namely user query profiles (see definition in Section 3), to find and relate the users' queries with the subset of data sources that actually contribute to answering the query.

Most importantly, we must capture the content and capability of each data sources independently of other sources, thus providing a source-to-source independent metadata management to obtain high scalability of the global information systems. Furthermore, we must allow information consumers to pose queries independently of the list of available data sources and their source-specific data representations, thus providing a query-to-source data independence, and an ability of incorporating new information sources into the answering of existing queries dynamically and seamlessnessly.

This research examines the specification and use of metadata in a simple consumer-producer model. Information consumers pose their queries without requiring any knowledge of data sources, and user query profiles are created and maintained for capturing the semantic annotation or changing requirements of the consumer queries. Information producers' data sources are described through the source capability profiles which capture the metadata information of the sources such as the content, catogory, query capability and access rights of the sources. We describe a rule-based representation language for

both the source capability profile specification and user query profile specification. We examine query processing strategies that use this semantic representation language to determine if the data source can provide the consumer query or application with meaningful data. In particular we describe the role of metadata in query routing, an important dynamic query processing technique in distributed open environments. The methods proposed for dynamic interconnection of consumers' queries with producers' sources allow for changes in the content, numbers, location, and connectivity of the sources or changes in the consumers's query requirements.

The rest of the paper is organized as follows. We first examine related work in the area of metadata management. Then we use a motivating example to describe problems of naive keyword based search and the role of metadata in improving query responsiveness. Following the background and motivation of our work, we introduce the metadata description model for describing user query profiles and source capability profiles. We describe the use of our metadata model and in particular, how the user query profiles are used in conjunction with the source capability profiles to determine the relevance of the sources to a particular user query and to identify and resolve the semantic conflicts between consumers' query representation and the data sources. We also examine the use of our metadata model in a dynamic environment where changes occur in either consumers' query specifications or producers' source capability descriptions. Finally we present our conclusion and describe the areas of future research including the use of metadata in semantic reconciliation and query result packaging and assembly.

2 Metadata

Metadata refers to data about the meaning, content, organization, or purpose of data. Metadata may be as simple as a relational schema or as complicated as class library or information describing the derivation, accurancy, and history of individual data items.

In [13], Sciore, Siegel and Rosenthal describe a metadata approach to facilitate interoperability among heterogeneous information systems. Their approach uses semantic values in the context of relational model and provide transparent context conversions and manipulations of metadata and a context mediator to capture the context-related meta information. In [11], McCarthy presents a metadata representation language. It allows the inclusion of a wide range of metadata accessible through a set of operators specially defined for metadata manipulation. [5] uses knowledge-based representation for metadata. However, neither of these methods discuss the practical means for defining comparable concepts and relating concepts at schema level and data level to facilitate the processing of queries over heterogeneous data sources.

We intend to use metadata to address the following questions:

1. How can we find the set of information sources that are relevant and are semantically meaningful to answering a user query in an open environment?
2. Can the way how a user query is posed be independent of changes in the information source content description?
3. Can the source content and capability description be independent of changes in the semantics of other information sources?

3 A Motivating Example

Example 1 Suppose we want to search for title, price, supplier, and review of books about `cancer`, published in 1996, looking at the reviews and compare the prices before making a purchase decision. The parameters of interest to this query are the price, year, title, the supplier name, and the book reviews. We may ask query Q: *get the title, price, the supplier, and the reviews of books about* `cancer` *that were published in 1996.* We may enter the query using a SQL-like on-line form (e.g., The DIOM Interface Query Language: IQL [10]) as follows:

```
SELECT  B.title, B.price, R.review, B.supplier
FROM    Book B, Review R
WHERE   B.description CONTAINS 'cancer' .AND. B.year = 1996
        .AND. R.booktitle = B.title;
```

Suppose we have access to the on-line information sources shown in Figure 1, among many others.

3.1 Problems With Naive Search

A straightforward way of answering this query is to contact all data sources available online. Neither user query profiles nor source query capability profiles are used. We refer to this approach as the naive search.

Due to the semantic conflict between what the user wants to query (such as books from book store or publisher suppliers) and the capability of the data sources, only four of the data sources can actually contribute to the answer of Q. They are Source 2, 4, 6, 8. The first problem is the overhead of contacting the rest of the data sources that do not contribute to the query. For instance, Source 1 will be contacted by most of the simple keyword-based search tools, even though it is actually not able to contribute to the answer of the query Q, because Source 1 contains reviews for books published during 1970 and 1980 whereas the query Q asks for books published in 1996.

The second problem is the risk of receiving too many irrelevant data items in the query result. For instance, Source 5 will be contacted and used to answer the query Q, because it is an online book store database. The results returned from Source 5 would be all reviews about books on subject `Cancer`, published after 1970, even though the query Q only asks for those books published in 1996. This is simply because Source 5 does not accept a particular year as the input of external queries, nor provides the year as the output parameter for external queries.

To avoid or reduce the overhead of contacting irrelevant sources at early stage of the query processing, and to guarantee semantically correct answers, it is quite clear that there is a need of a metadata specification language that can be used to describe the content and capability of data sources and the scope and context of user queries. There is also a need of transformation methods that can establish a dynamic interconnection between information consumers and producers.

3.2 Query Routing Using Source Capability and User Query Profiles

Suppose we annotate the example query Q in the user query profile (see Section 3.1) that the suppliers we are interested in include only book stores and publishers, and the description attribute of books may correspond to title, abstract, or subject of books. Also suppose we have captured the content and

Source 1: Reviews for books published during 1970 - 1980 **Category**: Book Review; **Content**: Reviews, $1970 \leq b\_publish\_year \leq 1980$; **Query Capabilities**: Accept as input b_title or $b\_authors$, and optionally $b\_publish\_year$, $reviewer\_name$. Output is the book reviews for that b_title or $b\_authors$, and optionally $b\_publish\_year$.
Source 2: Barnes & Nobel **Category**: Retail Book Store Company; **Content**: Books; **Query Capabilities**: Accept as input title or authors or category, and optionally year, price, publisher. Output is the qualifying books, with title, authors, year, price, or contact information.
Source 3: BookClubs **Category**: Book Club Database; **Content**: Book, BookClub; **Query Capabilities**: Accept as input title or authors, and optionally year or price or category or clubNo. Output is the qualifying books or bookclubs, with title, authors, year, price, clubName, or contact information.
Source 4: Morgan Kaufmann Publishers Inc. **Category**: Publisher; **Content**: Books, Journals; **Query Capabilities**: Accept as input title or authors, and optionally year, price for books, and title or editor-in-chief, and optionally year for journals. Output may include title, authors, price, year, ISBN, purchase contact information for books; or title, editor-in-chief, subjectDesc, or contact information for journals.
Source 5: Books For Sale Databases (books published after 1970) **Category**: Book Store Database; **Content**: Book, $year > 1970$; **Query Capabilities**: Accept as input title, authors, and optionally price, category. Output is the qualifying books, with title, authors, price, publisher, or contact information.
Source 6: Book Reviews Database **Category**: Book Review Database; **Content**: Reviews; **Query Capabilities**: Accept as input title, authors, and optionally year. Output is the book reviews for that title, authors and/or year(s).
Source 7: TechReport Database **Category**: Technical Report Database; **Content**: TechReport; **Query Capabilities**: Accept as input title or authors or organization, and optionally year. Output is the technical reports for that title or authors or organization or year.
Source 8: Book Stores Database **Category**: Book Stores; **Content**: Books, BookStores where $store\_location$ is within Portland; **Query Capabilities**: Accept as input title or authors, and optionally publisher, store_name. Output may include title, authors, year, price, publisher, store_id for qualifying books, or store_name, phone, address for qualifying book stores.

Figure 1: Example information sources

query capability of the data sources listed in Figure 1 in a rule-based representation language as shown in Figure 4.

Now we can immediately determine that Source 7 is not relevant to answering this query, because it has no information about books. We can also conclude that Source 1, 3, 5 are not able to contribute to the answer of Q. The reasoning here is more subtle: we are interested only in books published in 1996 and supplied by either book stores or publishers, whereas Source 1 has review information only on books published in between 1970 and 1980, Source 3 has only books supplied by book clubs, and Source 5 neither takes a particular year as its input argument nor provides year information on its books as output. We are left with sources 2, 4, 6, and 8, and three independent subquery execution plans:

1. Ask Source 2 for the title, authors, and price of books about `cancer`, published in 1996. Assign the source name `Barnes&Nobel` to the supplier field of qualifying books. For each book, obtain a review from the Source 6.

2. Ask Source 4 for title, authors, and price of books about `cancer` and published in 1996. Assign the source name `Morgan Kaufmann Publishers, Inc.` to the supplier field of qualifying books. For each book, obtain a review from the Source 6.

3. Ask Source 8 for the title, authors, price, year, and supplier of books about `cancer`. From the result tuples of type $(title, authors, price, year, supplier)$, select only those books published in 1996. For each selected book, obtain a review from the Source 6.

The query result is the union of the results returned by executing these three subqueries.

Note that Source 2, Source 4, and Source 8 are qualified for answering Q because (1) the output capabilities of Source 2, Source 4, and Source 8 meet the output requirements of the query Q, and (2) the output capabilities of Source 2, Source 4, and Source 8 are enough to cover the mandatory input requirements of Source 6 (i.e., either title or authors of the book). Had we added $ISBN$ into the list of outputs of Q, we would not be able to use Source 2 and Source 8 in obtaining the answers, because we are not able to obtain $ISBN$ according to the output capabilities of Source 2 and Source 8. Had the input constraints of Source 6 required more specific information about the book (e.g., publisher) to return a review, we would not be able to get the reviews for books returned from Source 2, because the output parameter list of Source 2 does not include publisher, thus if the output parameter review of Q cannot be absent (optional), Source 2 would not be able to contribute to the query Q.

From this example, we observe that it is useful and beneficial to create user query profiles that allow non-native users who know what they want to annotate and record the query-specific semantics of the query parameters. For instance, by capturing and utilizing the annotation to the `supplier` parameter of Q, i.e., the suppliers of books requested are either book stores or publishers, we can straightway decide that Source 3 and Source 7 are not relevant to the answer of Q. It is also important to capture the content and query capability information of the data sources, so we can further prune the data sources that are incapable of contributing to the query answer, either due to the restriction on the scope of query interest (e.g., Source 7 and Source 3 in our example), or due to constraints on the list of mandatory input or output arguments of the sources (e.g., Source 5 does not have `year` as input or output argument), or due to the conflict of query interest ("`year = 1996`") with the access constraints associated with the sources (e.g., Source 1 has reviews for only books published in between 1970 and 1980).

In what follows, we first introduce the metadata description model that is used for semantic specification of user query profiles and source content and capability profiles. Then we discuss the methods that use these metadata profiles to identify and locate the data sources that contain semantically correct answers. We call this functionality *query routing* in DIOM, a mediator-wrapper based system [9] for querying heterogeneous information sources.

Query routing is the first optimization step that constrains the search space for a query in open environments, in preparation for query execution planning. One of the main goals of query routing is to identify relevant data sources for a query as early as possible, thus reducing the overhead of contacting the data sources that do not contribute to the answer of the query.

4 Metadata Description in DIOM

4.1 User Query Profiles: Semantic Specification of User Queries

One of the objectives of creating a user query profile for each query posed on the fly is to capture the semantic scope and context of the query. For each consumer query Q, we create a user query profile which consists of two key components: the query scope description and the query capacity description.

The *query scope description* describes the synonyms (alternative descriptions) for each atom name used in the `FROM` clause of Q and how these synonyms participate in the alternation relationship with the term, namely the *alternation constraints*. Four different types of alternations are considered here: *total* & *disjoint*, *total* & *overlap*, *partial*& *disjoint*, *partial* & *overlap*. We model the query scope, denoted as Q_{scope}, using *scope records* of the form (VI, SY, AL), where VI is the name of a virtual Interface, SY denotes a set of synonyms of VI, and AL denotes the alternation constraint over SY.

The *query capacity description* describes the qualification requirements for the arguments of the query. We model the query capacity, denoted as Q_{capa}, using *capacity records* of the form $(Attr, SY, AL, Dtype, Mattr, IOtype, Bopt)$, where $Attr$ denote an attribute argument, SY denotes a set of synonyms of $Attr$, AL is the alternation constraint over the set of synonyms SY, $Dtype$ denotes the data type and $Mattr$ denotes the metadata properties of the attribute $Attr$, $IOtype$ indicates if the attribute is used as an input argument or output argument or both, and $Bopt$ denotes the binding option (*mandatory* or *optional*) for the argument $Attr$. Note that all the input arguments in a query need to be satisfied and thus their binding options are mandatory, but sometimes we may allow certain outputs to be absent [10] in the answer by indicating their $Bopt$ to be optional.

In our initial implementation, the user query profiles are created using an interactive dialog interface program. A dialog screen will be pop up whenever the user poses a query on the fly. For instance, when the query Q in Example 1 is posed, the user may annotate each virtual interface class in the `FROM` clause using the query scope description record as shown in Figure 2.

Virtual Interface	Synonyms	Alternation
$Book(b)$	`novel, textbook`	`partial` & `disjoint`
$Review(r)$	`book review`	partial & overlap

Figure 2: The query scope description fragment of the user query profile of the example query Q

The user who posed query Q may annotate each input or output attribute of the query Q by defining the synonyms, their alternation constraints, and the data type, the metadata properties, the input/output type and the bounding option. These semantic definitions will be captured in the user query capacity record as shown in Figure 3). Data are not in `this` font are entered by the system using default settings derived from the query statement of Q or inferred using available on-line ontology.

Attribute	Synonyms	Alternation	DType	Mattr	IOtype	Bopt
$title(b, t)$			String		out	`mandatory`
$description(b, d)$	`title, abstract, subject`	`partial` & `disjoint`	String		in	mandatory
$year(b, y)$			String		in	mandatory
$price(b, p)$	cost	partial & overlap	float	(`price_status = sale price, unit = US$`)	out	`mandatory`
$review(r, v)$			String		out	optional
$booktitle(r, bt)$	`title`		String		in	mandatory
$supplier(b, s)$	`book store, publisher`	`total` & `disjoint`	String		out	`mandatory`

Figure 3: The query capacity description fragment of the user query profile of the example query Q

Since we assume that the term *supplier* is restricted to the semantic context that only two types of suppliers are of interest: *book store*s and *publisher*s, we describe such context by indicating *book store* and *publisher* as synonyms of *supplier* with *total&disjoint* alternation constraint. The *total&disjoint* alternation constraint means that the given set of synonyms of supplier (*bookstore* and *publisher* in this case) presents a total and disjoint categorization of all supplier objects. The default alternation constraint is set to *partial & overlap*. Similarly, the default for all the non-numeric attributes is String type. The default for all the numeric attributes is the corresponding data type of their numeric values such as integer, float, double. The default for binding option of an output argument is optional.

Important to note is that, for each user query posed to the global information system, we create a virtual interface schema and a user query profile, *independentII* of the structure, the number, and the connectivity of data sources, or the existence of the data being requested. Since the collection of information sources available is large and frequently changing, the logical data independence as such allows us to add new data sources or incorporate changes seamlessly into the query processing system at any time without affecting the way how queries are posed and how answers are delivered, thus higher scalability is achieved.

4.2 Source Capability Profiles: Semantic Specification of Data Source Capabilities

A source capability profile tells what is in an information source and what types of services are provided about its content. It not only contains the content and query capability description but may also include, for instance, statistics on the local data (e.g., size of relations), availability of the source with respect to the access cost and access authorization, as well as update capabilities of the source. In this section we will focus only on the source category, content, and query capability descriptions, since they are the essential components of the source capability profile and are used extensively in each step of the query routing process.

4.2.1 Category and Content Description

The category and content description of a data source describe what is in the information source. The content description of an information source tells us what types of objects (or tuples) are in the source. The category description tells us what type of domain the data in the source are used for and is used for the *IsA* categorization of the source. The source category description S_{cat} often contains information that can be used to verify an input (selection) condition or fill in an output parameter of a query. Consider the query in Example 1. Source 2 does not have information in its content description that directly matches to the `supplier` attribute of Q or its synomyms, but using the source category description S_{cat} = *Retail Book Store Company*, we can easily infer that the source name "Barner & Nobel" of Source 2 is a book supplier. Similar analysis applies to Source 4.

We model the content of an information source in terms of the object types and the object access constraints that the source objects must satisfy. Each source object type is described by a unary relation. Each access constraint is described using a conjunction of built-in comparison atoms of the form $a\theta v$ where a is an attribute of a source type and v is a constant drawn from a domain that is compatible to the domain of a. Formally, given a source S_i of k relations, the content description of S_i, denoted as $S_{cnt}(S_i)$, is described by a set of *content records*: $\{(R_1(\vec{Y_1}), IC_1)$, $(R_2(\vec{Y_2}), IC_2)$ ..., $(R_t(\vec{Y_t}), IC_t)\}$, where R_i $(i = 1, ..., t)$ are relations in S_i and IC_i $(i = 1, ..., k)$ are access constraints over relations R_i. We may view a source content description as a collection of views defined over the source. Each R_i describes one type of source objects. Every object in R_i satisfies the insertion constraint IC_i.

4.2.2 Query Capability Description

The query capability description of a data source tells which types of queries the source can answer about its content. We model the query capabilities of an information source S using *capability records*, each is denoted by $(S_{in}, S_{out}, S_{cond})$. S_{in} denotes the set of permissible input arguments. S_{out} denotes the set of permissible output arguments. S_{cond} denotes the logical constraint ($\wedge$ or $\vee$) on the mandatory input arguments. Figure 4 shows the set of content records and the set of capability records of the information sources listed in Figure 1. From now on, we denote each information source by a triplet $(S_{cat}, S_{cnt}, S_{qpd})$ where S_{cat} denotes the text description of the category of the source, S_{cnt} is a set of source relations, each may be associated with some access constraints, i.e., $(R_i(\vec{Y_i}), IC_i)$, S_{qpd} denotes a set of query capability descriptions, each of the form $(S_{in}, S_{out}, S_{cond})$.

Source 1: Reviews for books (published during 1970 - 1980) **Category**: *Book Review;* **Content**: *Reviews(r);* $b\_publish\_year(r,y) \wedge 1970 \leq y \leq 1980$; **Query Capabilities**: $\{b\_title(r,bt), b\_authors(r,ba), b\_publish\_year(r,by), reviewer\_name(r,n)\}$, $\{review(r,v), (b\_title(r,bt), b\_authors(r,ba), b\_publish\_year(r,by),\ b\_title(r,bt) \vee b\_authors(r,ba)$.
Source 2: Barnes & Nobel **Category**: *Retail Book Store Company;* **Content**: *Books(b);* **Query Capabilities**: $\{title(b,t), authors(b,a), category(b,ct), year(b,y), price(b,p), publisher(b,s)\}$. $\{title(b,t), authors(b,a), year(b,y), price(b,p)\}$, $title(b,t) \vee authors(b,a) \vee category(b,ct)$.
Source 3: BookClubs **Category**: *Book Club Database;* **Content**: *Book(b), BookClub(cl);* **Query Capabilities**: $\{title(b,t), authors(b,a), clubName(cl,cln)\}$, $\{title(b,t), authors(b,a), price(b,p),$ $year(b,y), clubName(cl,cln), contact(cl,info)\}$, $(title(b,t) \vee authors(b,a)) \wedge clubName(cl,cln)$.
Source 4: Morgan Kaufmann Publishers Inc. **Category**: *Publisher;* **Content**: *Books(b), Journals(j);* **Query Capabilities**: $\{title(b,t), authors(b,a), year(b,y), price(b,p), j\_title(j,tl)\}$, $\{title(b,t), authors(b,a),$ $price(b,p), year(b,y), ISBN(b,isbn),\ j\_title(j,tl), j\_desc(j,d)\}$, $title(b,t) \vee authors(b,a)$.
Source 5: Books For Sale Databases (books published after 1970) **Category**: *Book Store Database;* **Content**: *Book(b),* $year(b,y) \wedge y > 1970$; **Query Capabilities**: $\{title(b,t), authors(b,a), price(b,p), category(b,c)\}$, $\{title(b,t), authors(b,a),$ $price(b,p), publisher(b,s), contact(b,i)\}$, $title(b,t) \wedge authors(b,a)$.
Source 6: Book Reviews Database **Category**: *Book Review Database;* **Content**: *Reviews(w);* **Query Capabilities**: $\{book\_title(w,bt), book\_authors(w,ba),\ book\_year(w,by)\}$, $\{review(w,r), book\_title(w,bt),$ $book\_authors(w,ba), book\_year(w,by)\}$, $book\_title(w,bt) \wedge book\_authors(w,ba)$.
Source 7: TechReport Database **Category**: *Technical Report Database;* **Content**: *TechReport(tr);* **Query Capabilities**: $\{title(tr,t), authors(tr,a), organization(tr,og), year(tr,y)\}$, $\{title(tr,t),$ $authors(tr,a), organization(tr,og), year(tr,y)\}$, $title(tr,t) \vee authors(tr,a)$.
Source 8: Book Stores Database (in Portland) **Category**: *Book Stores;* **Content**: *Book(b); BookStores(s),* $store\_location(s,l) \wedge l =$ *'Portland';* **Query Capabilities**: $\{title(b,t), authors(b,a), store\_name(s,n)\}$, $\{title(b,t), authors(b,a), year(b,y), price(b,p),$ $store\_id(b,id), store\_name(s,n), address(s,adds)\}$, $title(b,t) \vee authors(b,a)$.

Figure 4: The metadata description of example information sources

Example 2 Consider Source 5 in Figure 4.
S_{cat} = `Book Store Database`,
$S_{cnt} = \{(Book(b), year(b,y) \wedge y > 1970)\}$,
$S_{qpd} = (S_{in}, S_{out}, S_{cond})$ where
$S_{in} = \{title(b,t), authors(b,a), price(b,p), category(b,c)\}$,
$S_{out} = \{title(b,t), authors(b,a), price(b,p), publisher(b,s), contact(b,i)\}$,
$S_{cond} = title(b,t) \wedge authors(b,a)$.

From now on, we assume that for each information source registered, we will have its source content and capability profile created. Once queries are posed to the global information system, for each posed query, a virtual interface schema and a user query profile will be created to capture the structural and semantic context of the query and to receive its answer.

In the subsequent sections we describe the role of metadata profiles captured for the user queries and for the data sources in query routing, a first step in distributed query processing.

4.3 Reasoning of Semantic Relevance Through Query Routing

In this section we briefly discuss how we compare the semantic specification of user query profiles and the semantic specification of source content and capability profiles, and conduct two-levels of reasoning about the semantic relevance between a user query and a collection of data sources.

In DIOM, the semantic relevance reasoning is mainly conducted in the query routing stage, the first step in the DIOM dynamic query processing framework. The detailed routing strategies and algorithms are beyond the subject of this paper. Readers who are interested in further details may refer to [8].

For level-one relevance reasoning, we compare the scope specification of a user query Q with the content and category specificaton of the set of available data sources, say $S_1, ..., S_n$. This step serves as a first-round selection which rules out all the information sources that are irrelevant in terms of the query scope description and the source content descriptions. For our example query, level-one relevance reasoning will prune the information sources whose contents are not relevant to books or book reviews. Using the list of information sources in Figure 4, among many others, it will find that Source 7 is not relevant to the query answer. The main idea of level-one relevance reasoning is to prune out those data sources that have no source relations matching one of the interface names in the query scope description of Q.

For level-two relevance reasoning, we compare the capacity specification of a query Q with the query capability descriptions of data sources. During the level-two semantic relevance reasoning, we prune the data sources that have level-one relevance but do not offer enough query capability to contribute to the answer of Q, either due to the restriction on the scope of query parameters of Q, or the restriction on the list of input or output arguments of the sources, or due to the conflict of query interest with the access constraints associated with the sources. For our example query, level-two relevance pruning will further prune away Sources 1, 3, 5, because they are not capable of contributing to the query answer due to the restrictions on either source access or source category. The main idea of level-two relevance reasoning is to prune those data sources (1) that have no input arguments, which are relevant to the input arguments used in the user query, or (2) that have no output arguments, which are relevant to the output arguments used in the user query, or (3) that have conflict with the interest of the user query (such as the query selection conditions do not match the access constraints of the sources).

4.4 Metadata Change Management

Most data sources are not static, and just as the structure of data may change so may the meaning of the data. For instance, the currency of book price required in a query result may change from German Marks to European Currency Units (ECUs). Therefore, it is important that the methods for relevance reasoning during the query routing process allow for changes in data semantics of producers' data sources and consumers' query profiles.

The methods presented for semantic relevance reasoning during the query routing process can be used in such a dynamic environment.

- When changes are made in a source, they will be captured by the wrapper in the content and capability profile of the source. The metadata manager must reevaluate the corresponding source against those queries that use the source. The reevaluation process is performed by comparing the query profiles with the updated portions of the source capability profile.
- When changes are made in a user query profile, the metadata manager must reevaluate the selected sources to check if they may continue to contribute to the answer of the query. If new sources are available online, the metadata manager will extend the reevaluation process to perform the two levels of relevance pruning over these new sources.

5 Conclusion

Several approaches have been proposed for querying multiple heterogeneous information sources using mediator-based architecture, such as TSIMMIS [4], Information Manifold [7], DISCO [3], and the context interchange project [13]. The key aspect distinguishing DIOM from other systems is its strive for logical independence (e.g., query independence and source independence). We implement such notion by (1) identifying the key aspects of the user query profile that are useful for capturing the semantic context (and scope) of what the user wants in a specific query, (2) by supporting the content and capability description of an information source independently of the user queries and of other information sources, and (3) by offering mechanisms for dynamically interconnecting the information sources to a query through the query routing process. Another contribution of the paper is the semantic relevance reasoning strategies we use to discover and prune the information sources that do not contribute to a query. A novel aspect of our relevance reasoning methods is the step-wise utilization of the user query profile and the source capability descriptions to ensure that each step considerably reduces the number of candidate sources considered in the next step.

The first prototype implementation of DIOM distributed query processing system (DQS) [12] was developed and tested on Solaris platform using SunJDK version 1.1. The byte-code has been tested on the following platforms: Windows NT v.3.5.1 using Netscape Navigator v.2.01, and Sun OS v.4.1.4 using Netscape Navigator v.3.0.

Our future research will examine the role of source capability and query profile metadata in automating the query result packaging and assembly process. We believe that in an open environment, it is more benefical to resolve the hard heterogeneity problems by providing semantic attachment mechanisms to annotate the results returned from heterogeneous sources, rather than using the enforcement of a pre-defined integration approach.

Acknowledgement

The first author is partially supported by NSERC grant OGP-0172859, STR-0181014, and DARPA grant MDA972-97-1-0016. The second author is partially supported by NSF grant IRI-9510112, DARPA grant MDA972-97-1-0016, and grants from Intel and Hewlett-Packard. Our thanks are also due to the DIOM implementation team at University of Alberta, especially David Buttler, Yooshin Lee, Kirill Richine, and Wei Tang.

References

[1] Y. Arens and et al. Retrieving and integrating data from multiple information sources. *International Journal of Intelligent and Cooperative Information Systems*, 2(2):127–158, 1993.

[2] C. Collet, M. Huhns, and W. Shen. Obtaining complete answers from incomplete databases. In *IEEE Computer*, 1991.

[3] D. Florescu, L. Raschid, and P. Valduriez. Using heterogeneous equivalences for query rewriting in multidatabase systems. In *Proceedings of the International Conference on Cooperative Information Systems (CoopIS)*, 1995.

[4] H. Garcia-Molina and et al. The tsimmis approach to mediation: data models and languages (extended abstract). In *Technical Report, Stanford University*, 1994.

[5] P. Gray, G. Storrs, and J. de Boulay. Knowledge representations for database metadata. *Articial Intelligence Review*, (2):2–29, 1988.

[6] D. Konopnicki and O. Shmueli. W3qs: A query system for the www. In *Proceedings of the 21st International Conference on Very Large Data Bases*, 1996.

[7] A. Levy, A. Rajaraman, and J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proceedings of the 22nd International Conference on Very Large Data Bases*, 1996.

[8] L. Liu. Query routing in structured open environments. Technical report, TR97-10, Department of Computing Science, University of Alberta, Edmonton, Alberta, Feb. 1997.

[9] L. Liu and C. Pu. An adaptive object-oriented approach to integration and access of heterogeneous information sources. *DISTRIBUTED AND PARALLEL DATABASES: An International Journal*, 5(2), 1997.

[10] L. Liu, C. Pu, and Y. Lee. An adaptive approach to query mediation across heterogeneous databases. In *Proceedings of the International Conference on Coopertive Information Systems*, Brussels, June 19-21 1996.

[11] J. McCarthy. Metadata management for large statistical database. In *The International Conference on Very Large Data Bases*, pages 470–502, Mexico City, 1982.

[12] K. Richine. Distributed query scheduling in the context of diom: An experiment. MSc. Thesis, Department of Computer Science, University of Alberta, A demo version of the prototype is available at http://ugweb.cs.ualberta.ca/ diom/query/EQ.html, April, 1997.

[13] E. Sciore, M. Siegel, and A. Rosenthal. Using semantic values to facilitate interoperability among heterogeneous information systems. *ACM Transactions on Database Systems*, 19(2):254–290, 1994.

[14] G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer Magazine*, March 1992.