

# An Object-Oriented Approach to Interoperation of Heterogeneous Information Sources

Ling Liu<sup>1</sup> and Calton Pu<sup>2</sup>

<sup>1</sup>University of Alberta, Edmonton, Alberta T6G 2H1 Canada  
email: lingliu@cs.ualberta.ca

<sup>2</sup>Oregon Graduate Institute, Portland, Oregon 97291-1000 USA  
email: calton@cse.ogi.edu

**Abstract.** In the modern Internet environment, various types of data sources have become accessible, including multimedia data and web pages. Some of the fundamental assumptions in traditional databases, such as the existence of a global schema and data consistency maintained by a Data Base Administrator, are no longer true in many of the new data sources. We outline the DIOM [LP95b] object-oriented approach to build interoperable heterogeneous information systems despite the absence of global schema and the presence of data inconsistency. We describe the metadata catalog management component of DIOM, a key service in the support for interoperation among heterogeneous information sources. To support a flexible and customizable connection between information consumers and information producers, DIOM metadata catalog development utilizes the adaptive specification mechanisms, provided in DIOM interface description language, for explicit description of information consumers' domain query requirements and for object-oriented abstractions of information producers' information sources.

## 1 Introduction

One of the important contributions of database research is the carefully chosen set of assumptions made on the data in the database. For example, a relational database is described completely by the database schema, and data/schema consistency is maintained by the Data Base Administrator (DBA). Query processors and optimizers use database schema in critical ways. Similarly, atomic transactions [GR93] depend on data consistency assumption, since they are defined as programs that take a database from a consistent state to another consistent state. In the current generation of new information systems (or so called non-traditional database systems (NDBs) such as scientific data [BP93], multimedia databases, a variety of Internet data sources [LP95b], some of these important assumptions do not hold and classic database techniques may not apply directly. In this paper, we examine two decreasingly valid assumptions in nontraditional

databases that strive to be interoperable: (1) data is consistent in the database, and (2) database schema is reasonably stable. Our focus is on the interoperability problems aggravated by inconsistency and rapid evolution. We restrict our attention to these two assumptions due to the complexity of the issue. We acknowledge the importance of other traditional database assumptions and anticipate much more work to be done in this direction.

We have proposed the Distributed Interoperable Object Model (DIOM) [LP95b] to address the problem of data inconsistency and dynamic system evolution in the integration of heterogeneous information sources. The main idea of DIOM is to define interoperability as a flexible and voluntary interfacing of heterogeneous information consumers and information producers. This interfacing is defined explicitly, by using object-oriented abstractions to capture query requirements from the consumer side and the metadata of information sources from the producer side. Our main contribution in this paper is the design of a metadata catalog management system, a key service in DIOM for supporting transparent and customizable access across multiple disparate information sources. We demonstrate that, given the evolving and inconsistent nature of individual data sources, how object-orientation can be utilized in the DIOM metadata model construction to facilitate the integration and access of heterogeneous information sources.

## 2 Problems Analysis

Although obvious in retrospect, it is worthwhile to restate that traditional database assumptions do not necessarily hold simply because we call a data source a *database*. There is a strong temptation to apply database results to new data sources, since database theory and software are typically stronger than existing data access support. However, without examining the validity of traditional database assumptions we run the risk of introducing more problems than we solve. For example, if data is already inconsistent, then using atomic transactions will not produce a consistent database state. The inconsistency could be amplified by some transactions, for instance, one that makes multiplication involving a numerical field with inconsistency [RP95].

In this section we focus our analysis only on the traditional assumptions on data consistency and schema stability. By analyzing the validity of the data consistency and schema stability assumptions in traditional databases, we see that they are ensured by two basic factors: the central authority of DBAs, and the existence of a globally valid database schema. It is the role of DBA to run the programs that filter out inconsistent data,

and be the arbiter in deciding which data is good when conflicts and contradictions arise. Usually, the syntactic aspects of data consistency can be decided automatically using the database schema and the semantic aspects of data consistency depend on custom/application programming or DBA intervention when everything else fails.

## 2.1 Data Inconsistency and Heterogeneity

One of the most important challenges in the interoperable database systems of the future, especially in a rapidly growing environment such as the Internet, is the handling of inconsistency when individual and heterogeneous databases are integrated into an interoperable database system.

By analyzing inconsistency, we start with the inherent inconsistency in an individual application-specific database. In some information systems, the inconsistency in data is well recognized, although not necessarily explicitly described. For example, most of scientific observational data, including much of satellite data such as the Earth Observation System, contain uncertainties inherent in the sensors. Sometimes these uncertainties are included as part of data (e.g., error bars). Other times, these uncertainties are considered part of the sensor description (e.g., model number and brand of an electronic microscope for digital images), or included in the metadata, as part of experimental setup description (e.g., in a Crystallographic Information File description of protein structures [NSF93]).

To complicate matters, the inconsistency may take several forms or contain several dimensions. For example, some inconsistencies are given by absolute bounds [RP95], as in bank accounts that allow limited withdrawals from Automatic Teller Machines, where the inconsistency is bounded by the daily cash withdrawal limit. In contrast, other inconsistencies depend on statistical and probabilistic models, such as the error bars in scientific data [BP93]. Analogously, in addition to value inconsistency, there is a time dimension to most data [PTWY94]. For example, even if we do not know the current value of a data item, knowing the last update time may be helpful. Another example of inconsistency involves the issues known as data quality. Without a DBA certifying the correctness and up-to-dateness of a data item, the judgment of its quality is much harder.

As we attempt to integrate information from multiple autonomous and heterogeneous data sources through interoperation, inconsistency is compounded, and in many cases, in an unpredictable way. Without a good handle on the inconsistency problem, these interoperable databases will become unreliable sources of data from the consistency viewpoint.

## 2.2 Dynamic System Evolution

A schema is often used as an interface between a data repository and programs. Once a schema changes, three levels of inconsistency may occur: (1) Inconsistency in schema itself; (2) Inconsistency between existing database and the modified schema; (3) Incompatibility of application programs in accessing data.

In traditional database systems, schemas were assumed to be reasonably stable. It is mainly because the cost of maintaining database consistency and program compatibility in the presence of schema modification can be too high to be affordable.

As the number of databases available via networks increases swiftly, even when the changes in each individual component database schema are not frequent at all, the large number of component databases may add up to surprisingly frequent schema change events at the interoperable database system level. Assuming that there is only one change in two years for any component database schema, an interoperable system with two hundred databases as such will have to contend with one hundred changes every year, which corresponds to, on average, two changes every week!

## 3 Distributed Interoperable Object Model: An Overview

*Distributed interoperable objects* (DIOs) are objects that support a level of interoperability beyond the traditional object computing boundaries imposed by programming languages, data models, process address space, and network interface [Bet94].

Distributed Interoperable Object Model (DIOM) is developed mainly for incorporating system-wide scalability and evolution support as well as component composability into an interoperation framework. In an environment where multiple information sources are dynamically connected to the Internet, it is highly desirable for users and applications to be able to work with remote data sources and request services across networks as if they were working with a single database system. We have captured the requirements of these large-scale interoperable systems by USECA properties [LP95b] Uniform access to heterogeneous information sources, Scalability to the growing number of information sources, Evolution and Composability of software and information sources, and Autonomy of participants, both information consumers and information producers.

DIOM consists of an interface description language (IDL) and an interface query language (IQL). Rather than inventing yet another object-

oriented data model, DIOM adopts the ODMG-IDL and ODMG-OQL object database standard [Cea94, Kim94] as the base model and adds a number of interface abstractions to allow new interfaces to be constructed in terms of existing ones, and to provide better support for the USECA properties. These incremental interface construction facilities include: *import mechanism*, *interface aggregation*, *interface generalization*, and *interface specialization*.

DIOM introduces base interface and compound interface concepts to separate the interface descriptions for information producers' source models and the interface descriptions for information consumers' domain usage models (personal views). Such a separation serves dual purposes: (a) It enables the system to automate the specification of base interfaces. (b) It allows information consumers to build dynamic and personal views, rather than monolithic and integrated snapshots, over the growing number of information sources. Other important features of the DIOM object model include:

- the incremental design and construction of compound interoperation interfaces through recursive application of abstraction and composition operators on the existing interfaces,
- the deferment of conflict resolution to the later stage of query evaluation in contrast to classical approaches that enforce conflict resolution by schema integration or at query formulation time,
- the support for application designers to define their own interoperation interface schema, independently of any system-supplied global schema.

A DIOM database consists of a collection of DIOs, which are composed through dynamically binding data from multiple data repositories provided by the underlying component systems. The DIOM interface definition language (DIOM-IDL) is used to the relevant portions of the information sources through object-oriented abstractions.

Consider an enterprise system that deals with book inquiry, sale, and purchase business in a distributed and heterogeneous environment such as the Internet. For concreteness, we assume that the relevant information sources that are currently available include BookStore database (Oracle), book club database (Sybase), publisher database (Informix), library database (ObjectStore), and bibliography file (in LaTeX `.bib` format). We refer to these information sources as component information sources of the given enterprise system. Fragments of the export schemas of these information sources are given in Figure 1, where the primary keys are underlined. Obviously, some of these information sources have similar

data contents, but their representations are different. This is generally true for most of the information sources that are created and managed by separate organizations (information producers) with different business objectives or using different types of data repository managers.

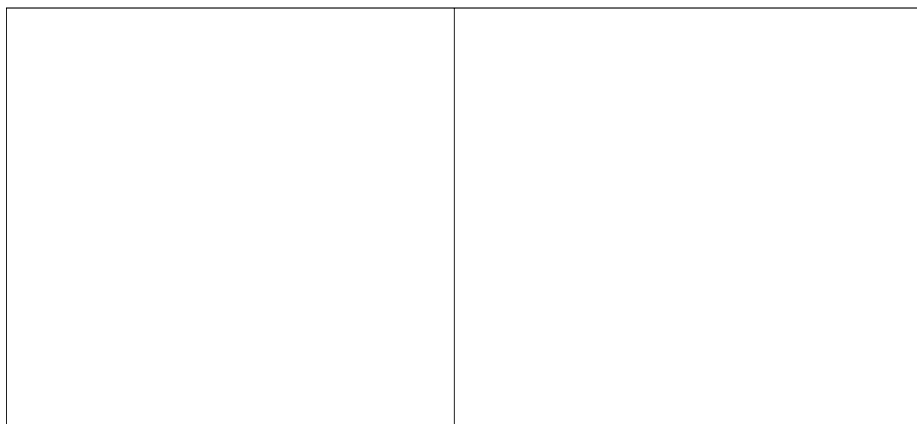


Figure 1: Fragments of the export schemas of example information sources

Suppose that a consumer wants to access the book information and the relevant supplier's information. She may describe her domain query model in IDL as follows:

```
CREATE INTERFACE Book
( extent books
  key   Book->ID )
{ GENERALIZATION OF
  select interface-name from Interface-Repository
  where description = '*Book' OR description = '*Books';
  ATTRIBUTES
    String ID,
    String title,
    Supplier supplier,
    String medium,
    Double price,
    Order orderNo;  }

CREATE INTERFACE Supplier
( extent suppliers
  key   Supplier->ID )
{ GENERALIZATION OF
  select interface-name from Interface-Repository
  where description CONTAINS
```

```

        ['BookStore', 'BookClub', 'Publisher', 'Supplier'];
ATTRIBUTES
    String ID,
    String name,
    String addr; }

```

All queries and request for data and services may be formulated using DIOM interface query language (DIOM-IQL) against some IDL interfaces. For example, the query  $Q$ : “find titles, prices and supplier names of all the books that are related to ‘SGML’ can be expressed in IQL as follows:

```

SELECT Book->title, Book->price, Supplier->name
FROM   Book, Supplier
WHERE  Book->title CONTAINS 'SGML';

```

Figure 2: An example query

In DIOM, to process this query we first select relevant sources that should be used to answer this query. Then we reformulate this IQL query into a number of subqueries, each targeting at one single source before translating each subquery to the source-specific query expression that can be executable at the source. The metadata catalog generated based on IDL and IQL specifications plays a critical role in each step of the DIOM query processing.

In the following section we discuss how we construct and maintain the distributed metadata catalog in DIOM and what roles the metadata catalog management play in each phase of the DIOM query mediation process and in supporting USECA properties within the DIOM interoperation architecture [LP95b]. Examples are provided to highlight the benefits of object-orientated approach to the design of DIOM interface description language. For further detail about DIOM-IDL and DIOM-IQL, readers may refer to [LP95a, LPL96].

## 4 Metadata Support

In the Diorama project, metadata extracted from information consumer’s interface schemas and information producers’ export schemas are managed and maintained in the DIOM metadata catalog. In the following sections we will discuss three most important components of the metadata catalog: information source repository, interface repository, and implementation repository. Examples are provided based on the *Book-Sale*

application scenario and the consumer’s query given in the previous section.

#### 4.1 Information Source Repository

*Information source repository* is created and maintained by the DIOM information source registration manager, which is responsible for recording all the available source producers’ information, such as the information producer’s name, the location pointer (e.g., the URL in WWW), the information source description (keywords), the export schema locator, the source type, and so on. A fragment of the DIOM information source repository is shown in Figure 3. Every application-specific mediator (App-mediator), once created, may also be registered as an information source to be available to the customer set of the DIOM system.

| SourceName | PointerToProducerSite   | SourceType  | Description             | ExportSchema      |
|------------|-------------------------|-------------|-------------------------|-------------------|
| Harvest    | harvest.cs.colorado.edu | Broker      | PCSoftware,CSTR,...     |                   |
| Yahoo      | yahoo.com               | Broker      | ...                     |                   |
| CForder    | diom.cs.ualberta.ca     | AppMediator | Claim,DiagnosisRep,...  | Diom_ClaimForder  |
| DR1        | dr1.com                 | RDB         | Books,BookStore,...     | DR1.BookstoreDB   |
| DR2        | dr2.com                 | RDB         | Books,BookClub,...      | DR2.BookClubDB    |
| DR3        | dr3.com                 | RDB         | Books,Publisher,...     | DR3.USPublisherDB |
| DR4        | dr4.com.de              | RDB         | Books,PurchaseOrder,... | DR4.GMPublisherDB |
| DR5        | dr5.org/                | OODB        | Books,Library,...       | DR5.LibraryDB     |
| DR6        | ftp.cs.ualberta.ca      | .bib file   | Book,Article,TechRep    | DR6.Bibliography  |
| ...        | ...                     | ...         | ...                     | ...               |

Figure 3: Fragment of DIOM information source repository

#### 4.2 Interface Repository

*Interface repository* is another basic component of the DIOM metadata catalog. It is created and maintained by the IDL preprocessor and the DIOM base interface generator. Whenever a new information consumer’s domain model is defined in DIOM-IDL, the interface repository will be extended. The first scan of the IDL preprocessor over a given IDL schema handles all the interfaces defined by means of the `import` and `hide` mechanisms. The base interface generator is invoked for each IDL interface defined using `import`. Figure 4 shows a fragment of the interface repository. For each consumer-defined (compound) interface or system-generated base interface, a pointer to the list of attributes, relationships, or methods associated with this interface definition is also maintained in the interface repository.

| interface-name | interface-type | description                           | source-producer |
|----------------|----------------|---------------------------------------|-----------------|
| Harvest        | base           | Security,PCSoftware,CS TechReport,... | Harvest         |
| Yahoo          | base           |                                       | Yahoo           |
| ClaimForder    | aggregation    | Claim,Image,DiagnosisRep,...          | Diom            |
| DR1.Books      | base           | Books                                 | DR1             |
| DR1.BookStore  | base           | BookStore,Supplier,...                | DR1             |
| DR1.Buyer      | base           | Buyer,Customer,...                    | DR1             |
| ...            | ...            | ...                                   | ...             |
| DR2.Books      | base           | Books                                 | DR2             |
| DR2.BookClub   | base           | BookClub,Supplier,...                 | DR2             |
| DR3.Publisher  | base           | Publisher,Supplier,...                | DR3             |
| ...            | ...            | ...                                   | ...             |
| Diom.Book      | generalization | Book, Books                           | Diom            |
| Diom.Supplier  | generalization | BookStore,BookClub,Publisher,...      | Diom            |
| Diom.Customer  | generalization | Buyer,Customer,Reader,...             | Diom            |
| Diom.Order     | generalization | Order,PurchaseOrder,...               | Diom            |
| ...            | ...            | ...                                   | ...             |

Figure 4: Fragment of DIOM interface repository

### 4.3 Implementation Repository

The DIOM servers (mediator level and wrapper level) provide a number of general services to facilitate the global query planning, the query reformulation, and the parallel execution of a multi-information source query. One of the most important services is the creation and maintenance of implementation repository metadata, which are used heavily by the DIOM query services provided by the mediators and their associated wrappers. The main issues to be addressed include: (1) how to establish a semantic correspondence between the types and relationships defined in the consumer's domain model with the classes and relations used in the producers' source models. (2) how to maintain the implementation metadata catalog in the presence of changes. The second issue will be addressed separately in Section 4.5.

The internal representation of relating an information source to a domain model for the **Book Sale and Purchase** application is illustrated in the Figure 5, Figure 6, and Figure 7. Each of these figures represents the attribute correspondence between mediated attributes defined in the consumer's domain model and source attributes defined in a particular information source model. The first column in each table contains the names of the mediated attributes and their type. The second column contains the corresponding information source level attributes, prefixed by the class or relation they belong to. For example, the attributes from database DR1, DR2, and DR3 are described in the second column of the table in Figure 5, Figure 6, and Figure 7 respectively. A null value in a source attribute column denotes the non-existence of the attribute in the

corresponding information source.

| Mediated-attributes | DR1               |
|---------------------|-------------------|
| Book→ID             | Books.book#       |
| Book→title          | Books.title       |
| Book→supplier       | Books.store#      |
| Book→orderNo        | Order.order#      |
| Book→medium         |                   |
| Book→price          | Order.price       |
| Order→ID            | Order.order#      |
| Order→quantity      | Order.quantity    |
| Order→customer      | Order.byerID      |
| Customer→ID         | Buyer.buyerID     |
| Customer→name       | Buyer.name        |
| Customer→addr       |                   |
| Supplier→ID         | BookStore.storeID |
| Supplier→name       | BookStore.name    |
| Supplier→addr       |                   |

Figure 5: Fragment of implementation repository maintained at the wrapper to DR1

| Mediated-attributes | DR2                 |
|---------------------|---------------------|
| Book→ID             | Books.bookID        |
| Book→title          | Books.description   |
| Book→supplier       | Books.clubID        |
| Book→orderNo        | Order.orderNo       |
| Book→medium         | Books.medium        |
| Book→price          | Order.price         |
| Order→ID            | Order.orderNo       |
| Order→quantity      | Order.qty           |
| Order→customer      | Order.customerID    |
| Customer→ID         | Customer.customerID |
| Customer→name       | Customer.name       |
| Customer→addr       | Customer.address    |
| Supplier→ID         | BookClub.clubID     |
| Supplier→name       | BookClub.name       |
| Supplier→addr       | BookClub.address    |

Figure 6: Fragment of implementation repository maintained at the wrapper to DR2

From the implementation point of view, the mediated attributes and their types are used as instruments to establish connections among similar attributes and their classes or relations defined in the relevant component information sources. These semantic links are central to the automation of global query planing, query refinement, query decomposition, and query result assembly (packaging).

Note that this type of metadata is mainly obtained by applying machine

| Mediated-attributes | DR3   |
|---------------------|---|
| Book→ID             | Books.bookID                                    |
| Book→title          | Books.title                                     |
| Book→supplier       | SaleDept.publisherID<br>PublishDept.publisherID |
| Book→orderNo        | Order.orderRef#                                 |
| Book→medium         | Books.medium                                    |
| Book→price          | Books.price                                     |
| Order→ID            | Order.orderRef#                                 |
| Order→quantity      | Order.qty                                       |
| Order→customer      | Order.customerID                                |
| Customer→ID         | Customer.customerID                             |
| Customer→name       | Customer.name                                   |
| Customer→addr       | Customer.address                                |
| Supplier→ID         | Publisher.publisherID                           |
| Supplier→name       | Publisher.name                                  |
| Supplier→addr       | Publisher.address                               |

Figure 7: Fragment of implementation repository maintained at the wrapper to DR3

learning techniques to the metadata catalog and user profile (if any). For example, the knowledge about connecting term **Publisher**, **BookClub**, and **BookStore** to the term **Supplier** can be learned directly from the interface definition of **Supplier**, while the connection between the federated attribute **Book→price** and the source attribute **Order.price** in DR1 can be learned by keyword-based partial matching. In order to build comprehensive domain-specific terminology matching rules and ontology, a sophisticated learning and knowledge discovery tool is needed. Appropriate interaction with the information consumer or domain expert may also be required. Our study on this topic will be reported in a forthcoming paper.

#### 4.4 Source-specific Metadata

In order to automate query translation and query result assembly, additional metadata are required. For example, for each attribute defined at an individual information source, the following information is needed: (1) the scale of the attributes, (2) the key constraint of the attributes (key, non-key, foreign key), and (3) the logical connection paths. The connection path assignment scheme varies for different types of information sources. For example,

- the connection path assignment scheme for the information sources managed by the relational systems is the following. A non-key attribute points to its key attribute(s). A key attribute points to

their foreign keys, and each foreign key points to its home key. If a relation has no foreign key, the pointer field is null. This simple path assignment schema ensures that all the necessary joins can be constructed when translating a consumer-level query into a group of subqueries at information producer's source level, each against a single information source.

- The connection path assignment scheme for object-based information sources is even simpler. Each *oid* field corresponds to the key field of a given object class. The rest are non-key fields. The navigational path for the same attribute type may vary from query to query, depending on the specific structural pointers (object references) specified in each query.

These metadata are mostly related to a single data source. They can be obtained directly from the export schema of the given information source, and incrementally maintained by the corresponding DIOM wrapper. More importantly, the translation of subqueries into the component query expressions should be carried out at the wrapper layer to prevent query processing bottleneck at the DIOM distributed object server layer. Figure 8 shows the internal representation of the metadata catalog maintained by the corresponding DIOM wrapper, the local DIOM agent to the DR3 information source. Note that (1) the designated navigational paths

| DR3.attribute         | DR3.domain      | DR3.scale | DR3.key | DR3.keyFD                             |
|-----------------------|-----------------|-----------|---------|---------------------------------------|
| Books.bookID          | none            | none      | key     |                                       |
| Books.title           | none            | none      | nonkey  | Books.bookID                          |
| Books.medium          | (CD-ROM, Paper) | none      | nonkey  | Books.bookID                          |
| Books.price           | none            | US\$      | nonkey  | Books.bookID                          |
| Books.deptID          | none            | none      | fkey    | SaleDept.deptID<br>PublishDept.deptID |
| SaleDept.deptID       | none            | none      | key     |                                       |
| PublishDept.deptID    | none            | none      | key     |                                       |
| Order.orderRef#       | none            | none      | key     |                                       |
| Order.bookID          | none            | none      | fkey    | Books.bookID                          |
| Order.customerID      | none            | none      | fkey    | Customer.customerID                   |
| Order.quantity        | none            | none      | nonkey  | Order.orderRef#                       |
| Customer.customerID   | none            | none      | key     |                                       |
| Customer.name         | none            | none      | nonkey  | Customer.customerID                   |
| Publisher.publisherID | none            | none      | key     |                                       |
| Publisher.name        | none            | none      | nonkey  | Publisher.publisherID                 |

Figure 8: Fragment of source-specific metadata extracted from the export schema of DR3

in the above table only specify the logical dependency between the non-key attributes and the key attributes of the corresponding information source which are involved in the MDB consumer's interface description.

During the subquery translation, these designated navigational paths will be re-examined based on the export schema in order to derive meaningful intra-database joins. (2) For each source attribute, a concrete structural pointer is derived at query execution time to replace the navigational path.

## 4.5 Maintaining Metadata Catalog in the Presence of Changes

When a new information source is registered, to guarantee this new information source will be used to respond a query, updates to the existing metadata catalog is needed. Two strategies can be used for maintaining the metadata catalog up to date: (1) Immediate update or (2) Deferred update.

With immediate update approach, the maintenance proceeds in three steps: First, the DIOM server triggers the metadata catalog manger to perform an incremental recompilation on all the existing domain interface schemas to which this new information source may be related. Second, for each relevant domain interface schema, the implementation repository manager will be invoked to relate object types used in this new information source to the object types defined in the given domain model. Third, the source-specific metadata catalog manager, associated with the wrapper to this new information source, is invoked. The source-specific implementation metadata is extracted from the export schema of this new information source.

With deferred update approach, the incremental recompilation of existing domain interface schemas is delayed to the time when a query is issued. By using deferred approach, instead propagating changes to all the domain interface schemas, only those domain interface schemas that are frequently used by queries are updated accordingly.

When a change is made to the information consumer's domain model or the information producer's source model, a modification to the corresponding metadata is performed accordingly and transparently. For example, modifying the domain of attribute `medium` by adding `Postscript` format in the relation `Books` of the information source `DR3` requires a change of the metadata, at the column of `DR3.domain` and the row of `Books.medium` in Figure 8, from `(CD-ROM,Paper)` to `(CD-ROM,Paper,PostScript)`. Obviously, this change propagation can be done transparently.

## 5 Discussion

DIOM is being implemented in the Diorama project. Diorama presents the applications with a number of distributed query services in addition to the interface description language IDL and interface query language IQL. Information consumers may query the information from multiple heterogeneous information sources through a unified object-oriented view in terms of IDL, and executes requests (e.g., queries, updates on object links, and method invocations) at the relevant information sources. Diorama's components conform to client/server standards such as CORBA [For93] and are dynamically bound to individual information sources to support independent component evolution. For instance, DIOM interfaces to C++ applications may be a set of C++ classes that act as "surrogates" for the corresponding definition of the DIOM interface schema. Information consumers may define their domain query requirement without having any precise knowledge of the available underlying information sources. New applications may define their interoperation interfaces by aggregation or generalization of existing interfaces, or by refinement of pre-defined interfaces, rather than starting from scratch. When a component source evolves into a new version, to incorporate the changes at the component level into the interoperation interface level, some modification to the DIOM distributed metadata library may be necessary. To ensure the subsequent queries are processed against the up-to-date information sources, the incremental IDL/IQL recompilation will be invoked transparently. As a result, no manual rewriting of consumer's query interface schema and application programs is required. Furthermore, when the number of information sources that participate in the interoperation increases, the existing legacy applications may continue to work without source code changes.

## 6 Related Work

Representative projects in the area of integration of heterogeneous information sources [Wie92, Wie93] include TSIMMIS [GMea94, PGMW95], which introduces an object exchange model and a query language for integrated information access. The Context Interchange approach [SSR94, GMS94] represents data semantics explicitly using a shared ontology. Other projects include work at USC [DS93], HP Labs [Sha93], and IBM Almaden Research Center [CHea94]. The entire area of distributed object management [ODV93] has also important impact on this line of research.

In addition, much of the work on heterogeneous databases are also relevant. However, our viewpoint is somewhat different from the previ-

ous research on the integration of schematic and heterogeneity, such as [KS91, SK92, SL90, SM91, VH91]. Instead of trying to minimize the differences and make them disappear, we accept the differences and try to characterize them and utilize them as semantic annotation in the DIOM query evaluation process.

## 7 Conclusion

We have looked into two of traditional database system assumptions in this position paper: (1) data is consistent in the database, and (2) database schema is reasonably stable. These assumptions are severely strained in distributed interoperable data sources, such as multimedia databases, object-oriented databases, web pages and other information sources on the Internet. Due to autonomy of component data sources, an interoperable information system will contain inherent inconsistency, as well as frequent schema changes. As a starting point for discussion, we have outlined the DIOM approach as a possible technique to handle the schema evolution and the data inconsistency incurred by representational and semantic heterogeneity of data.

The main idea of the DIOM approach is mandating explicit and voluntary declarations of what are information consumer wants and what information producers' supplies are of most interest. These declarations are stored in the metadata management system and used for deriving the relevant portions of producers' source data and for connecting information consumers with information producers in a flexible and dynamic way. We have described how the DIOM metadata catalog is constructed, managed and maintained in a DIOM environment. Three groups of metadata repositories are discussed: (1) metadata related to information sources (information producers' source information), (2) metadata related to consumer's interface descriptions (information consumer's domain query requirements), and (3) metadata related to query implementations (establishing a flexible connection between consumer's queries and producers' source data). We also highlight the benefits of building distributed metadata catalog based on the object-oriented design of IDL.

Our work on the DIOM distributed catalog management continues. The proposal for distributed metadata catalog management and the utilization of object-oriented abstractions to IDL design is still in its early stage of development. The metadata model and catalog management present only a first step towards finding solutions for the interoperability problem. No doubt other ideas and techniques will arise in response to the breakdown of these traditional database assumptions. We contend that in large networks such as the Internet, interoperability among heteroge-

neous information sources requires a systematic re-examination of many of the assumptions made by traditional databases. The investigation of which traditional techniques still apply under the relaxation of a particular set of assumptions become an increasingly challenging area for research and development. For example, techniques such as Epsilon Serializability (ESR) [PL91, WYP92, RP95] have been developed to bound the amount of inconsistency in individual transaction processing systems. We are currently studying the application of ESR algorithms to control inconsistency in the Internet environment.

## Acknowledgement

The work was supported partially by NSERC grant OGP-0172859 and NSERC grant STR-0181014 for the first author, and by ARPA under contract DABT63-95-C-0010 and NSF under grant IRI-9510112 for the second author.

## References

- [Bet94] Mark Betz. Interoperable objects: laying the foundation for distributed object computing. *Dr. Dobb's Journal: Software Tools for Professional Programmer*, (220), October 1994.
- [BP93] R. Barga and C. Pu. Accessing imprecise data: An interval approach. *IEEE Bulletin of the Technical Committee on Data Engineering*, 16(2):12–15, June 1993.
- [Cea94] R. Cattell and et al. *The Object Database Standard: ODMG-93 (Release 1.1)*. Morgan Kaufmann, 1994.
- [CHea94] M.J. Carey, L.M. Haas, and P.M. Schwarz et al. Towards heterogeneous multimedia information systems: the garlic approach. In *Technical Report, IBM Almaden Research Center*, 1994.
- [DS93] D.Mcleod D.Fang, S.Ghandeharizadeh and A. Si. The design, implementation, and evaluation of an object-based sharing mechanism for federated database systems. In *Proceedings of International Conference on Data Engineering*, Vienna Austria, 1993.
- [For93] Object Request Broker Task Force. *The Common Object Request Broker: Architecture and specification*. Object Management Group, 1993.
- [GMea94] Hector Garcia-Molina and et al. The tsimmis approach to mediation: data models and languages (extended abstract). In *Technical Report, Stanford University*, 1994.

- [GMS94] C. Goh, S. Madnick, and M. Siegel. Context interchange: overcoming the challenges of large-scale interoperable database systems in a dynamic environment. In *Proceedings of International Conference on Information and Knowledge Management*, pages 337–346, 1994.
- [GR93] J. Gray and A. Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann Publishers, 1993.
- [Kim94] Won Kim. Observations on the odm9-93 proposal. *ACM SIGMOD RECORD on Management of Data*, 23(1), March 1994.
- [KS91] W. Kim and J. Seo. Classifying schematic and data heterogeneity in multidatabase systems. *IEEE Computer Magazine*, 24(2):12–18, 1991.
- [LP95a] Ling Liu and Calton Pu. The diom approach to large-scale interoperable information systems. Technical report, TR95-16, Department of Computing Science, University of Alberta, Edmonton, Alberta, March 1995.
- [LP95b] Ling Liu and Calton Pu. The distributed interoperable object model and its application to large-scale interoperable database systems. In *ACM International Conference on Information and Knowledge Management (CIKM'95)*, Baltimore, Maryland, USA, November 1995.
- [LPL96] L. Liu, C. Pu, and Y. Lee. An adaptive approach to query mediation across heterogeneous databases. In *Proceedings of the International Conference on Cooperative Information Systems*, Brussels, June 19–21 1996.
- [NSF93] NSF with the cooperation of the Columbia University. *Proceedings of the CIF Tools Development Workshop*, Tarrytown, New York, November 1993.
- [ODV93] T. Ozsu, U. Dayal, and P. Valduriez. *Distributed Object Management*. Morgan Kaufmann, 1993.
- [PGMW95] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object exchange across heterogeneous information sources. In *Proceedings of International Conference on Data Engineering*, Taiwan, March 1995.
- [PL91] C. Pu and A. Leff. Replica control in distributed systems: An asynchronous approach. In *Proceedings of the 1991 ACM SIGMOD International Conference on Management of Data*, pages 377–386, Denver, May 1991.
- [PTWY94] C. Pu, M.K. Tsang, K.L. Wu, and P.S. Yu. Multiversion divergence control of time fuzziness. In *Proceedings of the Third International Conference on Information and Knowledge Management*, Gaithersburg, Maryland, November 1994.
- [RP95] K. Ramamrithan and C. Pu. A formal characterization of epsilon serializability. *IEEE Transactions on Knowledge and Data Engineering*, June 1995.

- [Sha93] M. Shan. Pegasus architecture and design principles. In *Proceedings of ACM/SIGMOD Annual Conference on Management of Data*, 1993.
- [SK92] A. Sheth and V. Kashyap. So far (schematically) yet so near (semantically). In *Proceeding of the IFIP WG2.6 Database Semantics*, pages 283–312, Victoria, Australia, 1992. North Holland.
- [SL90] A. Sheth and J.A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Transactions on Database Systems*, Vol. 22, No.3 1990.
- [SM91] M. Siegel and S. Madnick. A metadata approach to solving semantic conflicts. In *International Conference on Very Large Data Bases*, pages 133–145, 1991.
- [SSR94] E. Sciore, M. Siegel, and A. Rosenthal. Using semantic values to facilitate interoperability among heterogeneous information systems. *ACM Transactions on Database Systems*, Vol. 19, No. 2 June 1994.
- [VH91] V. Ventrone and S. Heiler. Semantic heterogeneity as a result of domain evolution. *ACM SIGMOD RECORD on Management of Data*, 20(4), 1991.
- [Wie92] Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer Magazine*, March 1992.
- [Wie93] Gio Wiederhold. Intelligent integration of information. In *Proceedings of ACM/SIGMOD Annual Conference on Management of Data*, 1993.
- [WYP92] K.L. Wu, P. S. Yu, and C. Pu. Divergence control for epsilon-serializability. In *Proceedings of Eighth International Conference on Data Engineering*, pages 506–515, Phoenix, February 1992. IEEE/Computer Society.