Determining Schema Interdependencies in Object-Oriented Multidatabase Systems

Jian Yang
Dept of Computer Science
University College, UNSW
Australian Defence Force Academy
Canberra, ACT 2600
Australia

Mike P. Papazoglou
School of Information systems
Queensland University of Technology
GPO Box 2434
Brisbane QLD 4001
Australia

Abstract

Schema analysis is the process of determining and capturing how individual schema items in diverse databases are related to each other. During this process different types of semantic relationships between schema items in different databases are identified and meaningful cross-links are established. The work presented in this paper concentrates on the issue of determining several types of interdependencies among schema items in object-oriented multidatabase systems in terms of their semantic context. A methodology is proposed for semi-automated schema analysis whereby knowledge from schema analysts may be applied in an interactive manner to refine and fully determine potential schema item interdependencies.

keywords: multi-database systems, object-oriented DBs, schema analysis and integration, semantic and structural discrepancies, inter-type assertions, analogical reasoning.

1 INTRODUCTION

A multidatabase system (MDBS) consists of a collection of independently developed and separately administered component database systems (CDBSs). The purpose of an MDBS is to support controlled sharing of system-wide data sources and provide coordinated control among a collection of autonomous and possibly heterogeneous CDBSs. Corporate data may reside anywhere in the CDBS network and can be operated upon transparently by both users and applications. To achieve this functionality an MDBS provides a homogenizing layer on top of the existing CDBSs, thus gives users the illusion of a single unified homogeneous database system.

In order for sharing to take place among heterogeneous CDBSs, some common data model must be used for describing corporate data and for coordinating distributed control. This model must be semantically expressive in order to capture the intended meanings of CDBS schemas which may use different data formats, obey different data modeling conventions and may more importantly model different aspects of the same concepts of some common domain. It is generally accepted that object-oriented models (especially those which include active database capabilities) can be considered as a key core technology for achieving integration and interoperation between multiple autonomous data sources along with providing the capacity to perform distributed object management functions [1], [2], [3], [4].

Successful integration of multiple CDBS schemas requires the identification and subsequent reconciliation of structural and semantic disparities which occur between the individual data elements within the CDBSs. For the purposes of this research it is convenient to distinguish between two complementary sets of such problems: structural conflicts and incompatibilities, and semantic inconsistencies and incompleteness [5]. By structural conflicts and incompatibilities we mean that the objects, object properties represented in the component databases are either conflicting or they present domain mismatches with respect to one another - even though they may model to the same 'real-world' concepts. Examples include problems in instance matches, units of measurement, contextual and structural mismatches, and so forth.

Semantic inconsistencies involve different interpretations of the meanings of schema entities in the individual CDBSs. For example, consider a CDBS which refers to its students as undergraduate and postgraduate students, whereas a second may choose to refer to them as first, second, third year, Master's and PhD students. The issue of semantic incompleteness relates to the situation where meaningful inter-object and inter-schema relationships, required for the purposes of schema merging and interoperation, are missing from the component schemas. For example, if one data source deals with information regarding commercial enterprises, it is quite unlikely to include all the information relating to a university lecturer with respect to his employment as company consultant. Conversely, a university database may not contain information concerning the consulting record of its lecturers. Only by establishing meaningful inter-schema relationship(s) between Lectures in the university database and Employees in the company database and combining information regarding these two views will the corporate information source be able to form an overall picture about employees who are lecturers at the same time.

The issues of structural conflicts and incompatibilities have been extensively researched. There exist various proposals for solving these and related problems [5, 7, 8, 9, 10, 11, 12]. In the area of structural incompatibilities, techniques were proposed for structural and domain mismatches whereby conflicting attributes are mapped to common domains by means of applying extended relational operators [13]. It is only recently that has been realized that tremendous potential lies in the use of sophisticated semi-automated tools for data analysis and integration. Bellcore's Schema Design and Integration Toolkit (BERDI) [14] provides intelligent support to improve the quality and reduce the time required to perform schema analysis and integration of non-trivial schemas in industrial set ups. These capabilities built on and enhance previous attempts to provide interactive


support for schema integration purposes [10].

As a general rule most of the above proposals are based on fairly high-level and intuitive understanding rather than a detailed classification scheme. Recently Kim and Seo [15] attempted to resolve this problem in a relational multi-database environment by proposing a systematic classification scheme. However, the issue of a general formal methodology for specifying useful inter-schema relationships and their characteristics during the process of schema analysis has yet to be addressed.

Semantic issues and in particular the problem of semantic incompleteness have either been marginally treated or received no attention thus far. Most notable is the work of Kent [16] where the author describes a great variety of ways to represent a given fact - some of which relate to our perception of semantic inconsistencies. The paper, however, does not propose any solution for resolving semantic inconsistencies. In a more recent paper [17] Kent discusses and proposes solutions to another semantic problem, that of ontological correspondences between entities created in different CDBSs. The author proposes a solution to this problem in terms of spheres of knowledge which act as repositories of abstracted knowledge describing CDB information. Semantic incompleteness has yet to receive appropriate attention, it only received tangential treatment as a by-product of the schema merging activities [18], [19]. The main thrust of these research activities has been towards using a knowledge representation scheme for automating the process of identifying different types of attribute relationships.

The work reported in this paper discusses several fundamental issues which have to be addressed during schema analysis in the context of object-oriented multibases and concentrates predominantly on semantic incompleteness issues. Towards this end, we use classification scheme in conjunction with determining inter-schema linkages according to the semantics and structure of analyzed CDBS schema items and a formal methodology for describing inter-schema relationships and their semantics. Research activities described in this paper build upon and partially extend previous research work that we conducted on schema analysis [20], [21], [22]. In [20, 21] we proposed a knowledge-based framework and discussed the basic features of a semi-automated tool for representing the semantics of inter-schema components and for reasoning about them, while in [22] we argued in favor of introducing case-based reasoning techniques for semi-automating the process of discovering structural and semantic relatedness between component schemas in CDBSs.

The remainder of this paper is organized as follows: Section-2 introduces the salient features of an object-oriented model used for discussing our approach to schema analysis. Section-3 outlines our basic approach to schema analysis in object-oriented multibases systems and proposes a formal methodology and an algorithm for type comparison in inheritance lattices, i.e., object-oriented schemas. Finally, Section-4 presents our conclusions and discusses future research directions.

2 DATA MODELING

In this paper we assume that the MDBS uses a common object-oriented database model which adapts knowledge representa-
Figure 1: Examples of Component Schemas

3 A METHODOLOGY FOR OBJECT-ORIENTED SCHEMA ANALYSIS

As a basis for our analysis of and approach to the various types of discrepancies that exist in object-oriented MDBSs, we consider developing an application for a collaborative long-term project conducted between three distinct institutions: a university’s computer science department, an engineering department and a computer company. Thus the need to integrate at least three distinct but interconnected component databases. Parts of the individual CDBS schemas are shown in Figure 1, where each database carries a tag name of the form $DB_i (1 \leq i \leq 3)$. Figure 3 shows possible properties for the two types Person and Staff. This figure also indicates which properties these two types share in common and the properties which are private to each of them - referred to as additional properties in the figure.

As with most methodologies, we determine potential relationships between pairs of similarly structured CDBS schema types on the basis of comparison. Currently, a semi-automated knowledge-driven schema analysis tool (SAKSAT) is under development whereby a human-operator is in a position to interact with this relatively sophisticated schema analyzer - which also stores derivation histories and cases of previous findings in a case base - in order to identify potentially related schema types and/or attributes [22]. This approach presents many similarities with BERDI [14]. SAKSAT acts as an "intelligent" assistant and is geared only towards determining the semantic relatedness of schema types, discovering implicit facts about them, reasoning about its findings and classifying the related types with the aid of a human operator whenever it does not have sufficient information. BERDI, on the other hand, addresses the entire schema integration cycle and focuses on the use of graphical user interface facilities to expedite integration. In this sense it uses graphical display facilities for representing extended ER schemas (called source schemas) and graphical query facilities for retrieving model level information (e.g., attributes and relationships with same names) and dictionary information (e.g., aliases, creators, businesses purposes, schemas of origin, entity descriptions and so on).

Following the relevant literature [11] we refer to that part of schema analysis used for determining relationships among different schema types as type assertions. Inter-type assertions represent serious sources of problems because they do not exist in the separate schemas, as such and have to be identified during the analysis process. Previous research work for detecting inter-schema relationships either totally relied on attribute equivalence or used heuristics to order pairs of objects for review by the database administrator (DBA) [21]. In fact, two complementary methods were proposed for determining type assertions: attribute subsumption [10, 12] and instance subsumption [10, 11]. Although these types of approaches present certain similarities and may in fact influence the treatment of
type assertions they are rather restrictive in their present form.

Furthermore, the proposed methodology poses a tremendous mental burden on the DBA. The entire process is ‘manual’ and the DBA acts only as a ‘technician’ who mechanically encodes the findings regarding type relatedness but cannot interact with the system to help it further refine type/attribute assertions based on previous findings. In order to overcome these limitations, it is necessary to examine a variety of existing or implied semantic links between types (and their respective attributes) in the individual CDBSs. Moreover, in order to identify potential type assertions, it is imperative that designers rely on the assistance of semi-automated intelligent schema analysis tools whereby the human operator may be asked to apply his knowledge if additional information is needed to further refine the nature of type assertions.

### 3.1 A Theoretical Basis for Determining Commonality and Inferring Similarity

We base our discoveries regarding existing commonalities and inferred similarities in CDBS schemas on a sound theoretical foundation. In the following, we provide formal definitions of inter-schema assertions based both on type definitions as well as existing type instances. For this purpose, we use two fundamental concepts from the AI areas of Analogy and Induction, namely existing commonality and inferred similarity [28]. Subsequently, we classify the various kinds of inter-type assertions and base our findings on these two concepts. In the following, we give a list of helpful definitions for this purpose:

**Figure 3: Common and Additional Attributes**

<table>
<thead>
<tr>
<th>$T_1/T_2$</th>
<th>Person</th>
<th>Staff</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Common</strong></td>
<td>Name</td>
<td>Name</td>
</tr>
<tr>
<td>Attributes</td>
<td>Id-No</td>
<td>Id-No</td>
</tr>
<tr>
<td></td>
<td>Address</td>
<td>Address</td>
</tr>
<tr>
<td></td>
<td>Tel-No</td>
<td>Tel-No</td>
</tr>
<tr>
<td><strong>Additional</strong></td>
<td>Age</td>
<td>Rank</td>
</tr>
<tr>
<td>Attributes</td>
<td>Sex</td>
<td>Experience</td>
</tr>
</tbody>
</table>

Let $P(T)$ be the set of properties which are defined in a given type $T$ and $P_e(T)$ be the set of all permitted, i.e., conformant, properties of type $T$ which are currently not included in its signature. We refer to such properties as shareable properties of the type $T$.

Since $P_e(T)$ includes the properties which are defined in type $T$ as well as those which are not included in its type's definition but are allowed to be part of it (e.g., attributes like age and sex in Figure 3 are shareable properties of type $T_2$) we then have $P(T) \subseteq P_e(T)$.

Now for any object $x$, let $P_e(x)$ be the set of all properties which are shareable by that object $x$. Further, let $P_e(x)$ be the set of all those properties not shareable by the object $x$. Hence:

$$P_e(T) = \bigcap_{x \in R(T)} P_e(x),$$

where $P_e(T)$ denotes the set of properties which are shareable by all objects in $R(T)$; $P_e(T)$ denotes the set of properties which are not shareable by all objects in $R(T)$; $P_e(T)$ denotes the set of properties which are shareable by some objects in $R(T)$.

If $C(T_1, T_2)$ is the set of common properties between types $T_1$ and $T_2$, then we have:

$$C(T_1, T_2) = C(T_2, T_1) = P(T_1) \cap P(T_2).$$

For example from Figure 4, one may observe that $C(T_1, T_2) = C(T_2, T_1) = \text{name, id-no, address, tel-no}$.

If $Q(T_1, T_2)$ is the set of properties that a type $T_1$ includes in addition to its common properties with $T_2$, then we have:

$$Q(T_1, T_2) = P(T_1) - C(T_1, T_2).$$

For example, from Figure 4, one may observe that $Q(T_1, T_2) = \text{age, sex}$.

Using the above set definitions, we can derive the inferred similarities formula as follows:

$$(C(T_1, T_2) \neq \emptyset) \land (q_i \in Q(T_1, T_2)) \Rightarrow (q_i \in P_e(T_2)).$$

The above formula is used to prescribe the interaction between the schema analysis tool and the DBA. Loosely speaking, this formula corresponds to a form of a question that should be answered by the DBA during his interaction with SAKSAT. The interpretation for this formula is as follows: consider the additional properties of type $T_1$ referred to as $(Q(T_1, T_2))$ and ask the DBA if they can also be shared by type $T_2$.

Obviously, there are three possible answers to the above question: (1) all the objects of $R(T_2)$ can share the additional properties of $T_1$; (2) not all the objects of $R(T_2)$ can share these properties; (3) all the objects of $R(T_2)$ cannot share these properties. The following formulae correspond to these three types of answers:

1. $C(T_1, T_2) \neq \emptyset \land (q_i \in Q(T_1, T_2)) \rightarrow (q_i \in P_e(T_2)).$
2. $C(T_1, T_2) \neq \emptyset \land (q_i \in Q(T_1, T_2)) \rightarrow (q_i \in P_e(T_2)).$
3. $C(T_1, T_2) \neq \emptyset \land (q_i \in Q(T_1, T_2)) \rightarrow (q_i \in P_e(T_2)).$

We exemplify this type of system/DBA interaction in the context of Figure 4. When SAKSAT compares the type Person in $DB_1$ with the type Staff in $DB_3$, it finds that the former type has two additional attributes: age and sex. Subsequently, it considers these two attributes and asks whether type Staff can...
also share them. The obvious answer is ‘Yes, all staff objects
can have these two attributes’. Similarly, type Staff includes
three attributes not contained in type Person, namely rank,
salary and experience. The same type of question is now
asked for Person. The obvious answer here is ‘Some can, some
cannot’. This answer in fact implies that Staff is a subtype of
Person. In this way, the system derives the inferred similarities
between related types.

In the following section we use these formal definitions in
order to classify type assertions.

3.2 Classifying Inter-Type Assertions

In what follows we give definitions and examples of the most
representative classes of type assertions in an object-oriented
multidatabase environment. For the classification scheme that
follows we assume that the types T1 and T2 belong to two
different schemas in different CDBSs.

Definition 1: T1 is Equivalent to (Eq) T2 iff \( \mathcal{R}(T_1) = \mathcal{R}(T_2) \).

Finding 1: If T1 Eq T2, then either
(1) \( \mathcal{C}(T_1,T_2) \neq \emptyset \) \& \( \mathcal{Q}(T_1,T_2) = \emptyset \) \& \( \mathcal{Q}(T_2,T_1) = \emptyset \); or
(2) \( \mathcal{C}(T_1,T_2) \neq \emptyset \) \& \( \mathcal{Q}(T_1,T_2) \in \mathcal{Q}(T_2,T_1) \) \& \( \mathcal{Q}(T_2,T_1) \in \mathcal{Q}(T_1,T_2) \).

Definition 1.1: For case 1.(1) above, we say that T1 and T2 are Identical (Id);

Definition 1.2: This case is a subcase of definition 1.1 and corresponds to case 1.(2) above. Here we say that T1 and T2 are Isomorphic (Iso).

Description 1: If two types are equivalent, then their real
world semantic classes are equal. If two types are equivalent
then their definitions are either the same - in which case they are
equal or if one type has additional properties, these
properties can be shared, i.e., they are permissible, by the other type
- in which case the types are isomorphic.

Definition 2: T1 is a Generalization (Ge) of T2 iff \( \mathcal{R}(T_1) \supset \mathcal{R}(T_2) \) and T2 is a Specialization (Sp) of T1.

Finding 2: If T1 Ge T2, then either
(1) \( \mathcal{C}(T_1,T_2) \neq \emptyset \) \& \( \mathcal{Q}(T_1,T_2) = \emptyset \) \& \( \mathcal{Q}(T_2,T_1) = \emptyset \); or
(2) \( \mathcal{C}(T_1,T_2) \neq \emptyset \) \& \( \mathcal{Q}(T_1,T_2) \in \mathcal{Q}(T_2,T_1) \) \& \( \mathcal{Q}(T_2,T_1) \in \mathcal{Q}(T_1,T_2) \).

Definition 2.1: For case 2.(1) above, we say that T1 is a
Supertype (Sup) of T2.

Definition 2.2: This case is a subcase of definition 2.1 and corresponds to case 2.(2). Here we say that T1 is a Conceptual
Supertype (Csup) of T2.

Description 2: If T1 is a generalization of T2 then the real
world semantic class of T2 is a subset of that of T1. And all the
definitions of T2 are logically subsumed by T2.

Definition 3: T1 and T2 are Category (Ca) related if \( \mathcal{R}(T_1) \cap \mathcal{R}(T_2) = \emptyset \), and C(T1,T2) \neq \emptyset holds.

Finding 3: If T1 Ca T2, then
\( \mathcal{C}(T_1,T_2) \neq \emptyset \) \& \( \mathcal{Q}(T_1,T_2) \in \mathcal{Q}(T_2,T_1) \) \& \( \mathcal{Q}(T_2,T_1) \in \mathcal{Q}(T_1,T_2) \).

Description 3: The fact that two types are category related
The following rules are used to accelerate the process of type comparison:

- Rule 1: If \(T_2\) is a subtype of \(T_1\), then \(T_2\) is disjoint with all the other siblings of \(T_1\) and their descendants.

- Rule 2: If \(T_1\) and \(T_2\) are equivalent, then all the siblings of \(T_1\) and their descendants may be overlapping with all the siblings of \(T_2\) and their descendants.

- Rule 3: If \(T_1\) and \(T_2\) are category related, then all the descendants of \(T_1\) and \(T_2\) are category related to each other.

Using the above rules during the process of type comparison in object-oriented schemas, the search times will be significantly reduced.

In the following we present an algorithm for inter-type comparison in inheritance lattices. We assume that \(IR\) is an inter-schema assertions matrix, where an element \(IR[T_i, T_j]\) is used to denote the sort of inter-type assertion holding between types \(T_i\) and \(T_j\) in different CDBs. According to the inter-type assertion definitions in Section 3.2, \(IR[T_i, T_j]\) is allowed to assume only one of the following values: \{Eq, Ge, Sp, Ro, Ca, Un, Dsj\}. These symbols correspond to assertions of the type Equivalent, Generalization, Specialization, Role, Category, Unknown, Disjoint, respectively.

The \(IR\) matrix is initialized with \(Un\) entries in all its positions. The algorithm which is shown in Figure 5 presents a systematic method for comparing pairs of types in diverse CDBs. This algorithm can be easily generalized and be used for comparison among \(n\)-types rather than pairs of types only. In fact in a companion paper we have argued in favor of the use of case-base reasoning techniques for this purpose [22].

**Figure 5**: An Algorithm for Lattice-based Type Comparison.
4 CONCLUSIONS

Determining the structural and semantic relatedness of types is paramount in our effort to analyze and integrate diverse component database schemas in a multidatabase environment. Unless one understands what the individual object types mean, how they are organized, how they are semantically related to each other, and if there are any conflicts between them, one cannot integrate them effectively. The most critical resource in our effort to resolve these problems is knowledge of the internal structures and semantics of conceptual schemas.

In this paper we identified several fundamental issues which have to be addressed during schema analysis in the context of object-oriented multidatabases. We introduced a formal framework for determining commonalities and inferring similarities between inter-schema types, classified inter-type assertions according to the semantics and structure of analyzed types and proposed an algorithm for type comparison in inheritance lattices. Currently, a case based methodology is being explored to semi-automate the process of discovering structural and semantic conflicts and relatedness of type components from disparate component data sources. As this methodology leans heavily on reasoning, deductive and object-oriented facilities, our implementation vehicle relies on a novel data model based on a merger of the object-oriented database system ONTOS and the expert system shell CLIPS.

Acknowledgments

We gratefully acknowledge the contribution of Brendan McKay for his suggestions and helpful comments regarding an earlier version of this article.

References


