Transform More Semantics from Relational Databases into Object-Oriented Databases

Ge Yu, Guoren Wang, Huaiyuan Zheng
Department of Computer Science and Engineering,
Northeastern University, P.R. China
yuge@csce.kyushu-u.ac.jp

Akifumi Makinouchi
Department of Computer Science and Communication Engineering,
Kyushu University, Japan
akifumi@csce.kyushu-u.ac.jp

Abstract

This paper presents a 2-phase schema transformation method for schema transformation from relational databases into object-oriented databases. The method is based on the identification of the explicit semantics and implicit semantics in schemas. In this paper, the mapping rules and transformation operations are defined for the transformation mechanism, and the transformation statements and query translation algorithms based on the G-Relational algebra are provided for global processing. The study is meaningful for integrating heterogeneous databases by object-oriented data models.

1 Introduction

1.1 Motivation

OSAM* is an object-oriented semantic data model for advanced applications such as CIMS, knowledge processing systems[12]. For its strongly modeling capability of both object-oriented data models and semantic data models, OSAM*-based heterogeneous distributed database systems have been developed such as IMDAS[8] and CIMBASE[14]. In the two systems, OSAM* is used as the common data model, and two OSAM*-based global languages OQL[1] and OSDL[15] are provided for global manipulations, respectively. Using OQL or OSDL, the data stored in the local databases such as relational databases and network databases are manipulated transparently in a top-down manner. However, the data residing in pre-existing databases cannot be manipulated because the database integration mechanisms are not provided.

Traditional DBMSs (Relational, Network and Hierarchical DBMSs) have been used for a long time before new DBMSs such as Object-oriented DBMSs become available, and many important applications are established by them. Recently, applications tend to use new DBMSs, however, DBMSs in use cannot be replaced in a short period because of restrictions on software techniques and implementation costs. Even if an old DBMS is replaced, the data managed by it need to be reserved and converted into the new database. In current situations, old DBMSs and new DBMSs have to co-operate in the same application system. The facilities, which can transform schemas and data between old DBMSs and new DBMSs dynamically, are required to support such operations. In this paper, the transformation problems between relational databases and object-oriented databases are addressed, and a facility for (semi-)automated schema transformation is designed. This mechanism can be embedded in a heterogeneous database system to support database integration in a bottom-up manner.

1.2 Design Principle

To implement database integration, many issues need to be studied such as system architecture, integration mechanism, query processing, transaction management, concurrency control, integrity and security checking[10], among which schema transformation is essential[2]. As relational databases and object-oriented databases are two types of popular databases, many papers have been published on their integration methods. A main approach is the view integration technique[3][5][7][9], which builds a global integration view on the top of local schemas. In such a system, a global database is a mapping of local databases and an operation on the global database is translated into the operations on the local database.

A view integration is implemented by a schema transformation mechanism. Because a schema con-
tains both the structural information and semantic information about data, a schema transformation should include the structural transformation and semantic transformation. Because the structure of a data model is well defined, the structural transformation can be done easily by direct mapping. On the other hand, the semantic transformation is a difficult job because it is very hard to describe the semantics clearly. In this paper, the semantics in a schema is divided into the explicit semantics which is defined explicitly such as referential integrity, and the implicit semantics which is hidden in the schemas or included in the application programs. To make a correct schema transformation, both types of semantics must be identified and captured into the global schemas. This paper aims at solving the semantic transformation problems. In the course of schema transformation, both translation and integration problems are involved[2], but this paper concentrates on the solution of translation and does not discuss the integration issues in detail although our transformation method can also be employed to solve integration issues such as name conflicts, domain conflicts and structural conflicts.

Our transformation mechanism is intended to support both view query and view update, therefore, the following transformation reversibility principles are obeyed:

- At schema level, a global schema represents the same database represented by the local schema. It requires that the local constraints also be enforced in the global schema.
- At instance level, a manipulation on the global view can be translated into the manipulations on the corresponding local databases.

1.3 Related Work

Our research is mainly related to following two papers in respect of the transformation concept and implementation techniques.

In paper [9], a framework for the generation of virtual view is proposed, a set of schema restructuring operators is defined and a compact integration tool is developed. However, since the integration model and translation algorithm are in functional approach, the result is not adequate to object-oriented databases. Besides, the complex constraints as described later in section 2.3 are not considered, so the semantics enriching function is not powerful.

In paper [6], an algorithm for automatically translating an external relational schema into an object-oriented system is presented. The inclusion dependency graph is exploited to capture the semantics of local schemas. However, as to database integration, two issues are not solved: (1) the implicit semantics is not included in the global schemas, thus the semantics of the global schemas can not cover local schemas. (2) the query translation from the global schema to the local schema is not presented so that the manipulations on the integrated database are not supported.

In this paper, we propose a semantics-based 2-phase schema transformation according to the semantics identification. In phase 1, the explicit semantics are automatically translated and captured. In phase 2, the implicit semantics are added and enriched by supplement operations. The G-Relational Algebra[11] is employed for describing and automating the transformation.

The paper is organized as follows. Section 2 describes the schemas and the underlying semantics to be transformed. Section 3 presents the transformation method including mapping rules and transformation operations. Section 4 describes the transformation statements for global schema definition. Section 5 describes the query translation algorithm for global manipulation. Finally, section 6 gives the conclusions and future works.

2 Problem Definition

In this section, a brief overview of the relational data model and the OSAM* data model is given, and then the underlying semantics to be transformed are discussed.

2.1 Relational Schema

The relational data model uses relations to express the information of both entities and relationships among entities[4]. Formally, a relational schema is expressed as a 4-tuple \((R, A, P, C)\), where

- \(R\) is a relation set in which each relation has a unique name.
- \(A\) is an attribute set in which each attribute has a unique name, a data type, the range of value, and the relation that it belongs to.
- \(P\) is a set of the primary-key of each relation.
- \(C\) is a constraint set which must be obeyed by relations in \(R\). The constraints fall into explicit constraints and implicit constraints, corresponding to explicit semantics and implicit semantics, respectively. An example of explicit constraints is the foreign key dependency.
2.2 OSAM* schema

The OSAM* data model expresses objects with associations and methods[12]. The structural properties of an object are defined by its various associations with other objects; the behavioral properties of an object are defined by user-defined procedures and knowledge rules. Formally, an OSAM* schema is defined as a 6-tuple \((E, I, D, A, C, M)\), where

- \(E\) is a set of entity object classes (named \(E\)-classes). Each \(E\)-class has a unique name.
- \(I\) is a set of the identifier attributes of each \(E\)-class.
- \(D\) is a set of domain object classes (named \(D\)-classes). The function of a \(D\)-class is to specify an atomic or composite domain of valid values for the objects of \(E\)-classes.
- \(A\) is a set of associations between one class with other class(es). An association has a unique name, a type (Aggregation, Generalization, Interaction, Composition or Cross-Product), and some links specifying the associated object class(es).
- \(C\) is a set of constraints, which also fall into explicit and implicit types.
- \(M\) is a set of methods and rules, by which integrities and other semantic constraints can be declaratively enforced.

2.3 Semantics in Local Schema

From the viewpoint of Entity and Relationship, we classify the underlying explicit and implicit semantics of a local relational schema into three types: Generalization, Interaction and Aggregation, in which constraints are encompassed, respectively. The semantics and main constraints to be considered are presented as follows.

Generalization Entities which have common attributes and constraints can be generalized into a generic entity. For example, a undergraduate student and a graduate student have the same attributes such as number, name, major, minor etc., thus a generic entity student can be generated from them to describe their common properties. Let \(R\) be a relation representing the generic entity, and \(R_1, R_2, ..., R_n\) be relations representing sub-entities. An generalization hierarchy encompasses constraints:

1. For each \(i(1 \leq i \leq n)\), \(R\) and \(R_i\) has a specialization constraint that each instance in \(R\) must belong to \(R_i\) or not.

Interaction For a relationship among interacting entities, a relation can be defined by the primary key of the interacting entities. Let \(R\) be a relation representing the relationship, and \(R_1, R_2, ..., R_n\) be relations representing interacting entities. An interaction hierarchy encompasses constraints:

1. For each \(i(1 \leq i \leq n)\), \(R_i\) has a participation constraint that its instance participates in at least one relationship or not.

2. \(R_i\) and \(R_j\) \((i \neq j)\) have a cardinality constraint of 1:1, 1:n or n:m.

Aggregation An entity is consisted of attributes and member(consituent) entities. Let \(R\) be a relation representing the entity, \(A\) be an attribute in \(R\), and \(R'\) be a relation representing a member entity. An aggregation hierarchy encompasses constraints:

1. \(A\) has a primary key constraint if \(A\) is a primary key attribute of \(R\).

2. \(A\) has a foreign key constraint if \(A\) is a primary key attribute of another relation.

3. \(R\) and \(R'\) have a cardinality constraint of 1:1 or 1:n.

4. \(R\) has a participation constraint that it has at least one instance of \(R'\) or not.

3 Transformation Method

To perform effective and correct schema transformation, a semantics-based 2-phase schema transformation method is designed as follows:

Phase 1: Semantics Capturing

The explicit semantics of local schemas is captured and automatically translated into the global schema. The global schema created in Phase 1 is called Base schema, which is created in terms of the local schema. Correspondently, an object class in it is called Base class. The Base Schema does not contain the entire semantics of the local schema.

Phase 2: Semantics Enriching

The implicit semantics of local schemas are identified and appended into the global schemas. The global schema created in Phase 2 is called Derived global schema, which is created in terms of a Base schema or another Derived schema. Correspondently, an object class in it is called Derived class. Finally, the Base schema and the Derived
schema are combined into a global concept schema for the global database. Let \( S \) be a semantics function, then following condition should be satisfied: 
\[
S(\text{Local-Schema}) = S(\text{Base-Schema}) \cup S(\text{Derived-Schema})
\]

To implement 2-phase transformation, following techniques are adopted:

1. In Phase 1, translation rules are defined to translate automatically the explicit semantics of the local schema into the global schema.
2. In Phase 2, transformation operations are designed to enrich the semantics of the global schema by appending as many as implicit semantics. If some implicit semantics are still missed, the methods and rules for the object classes have to be defined to include them.

### 3.1 Translation Rule

The semantic translation rules are defined as follows:

1. A relation \( R \) is mapped to a \( Base \) \( E \)-class \( E \), and \( R \)'s name is used as \( E \)'s name.
2. An attribute \( A \) of \( R \) is mapped to an \( A \)-association attribute \( A \)-attr of \( E \), and \( A \)'s name is used as \( A \)-attr's name.
3. A primary key attribute \( Key(R) \) of \( R \) is mapped to an identifier attribute \( Id(E) \) of \( E \).
4. The data type and range of \( A \) is mapped to a \( Base \) \( D \)-class \( D \), and the name of the data type is used as \( D \)'s name. The same data types are mapped to one \( D \)-class.
5. A foreign key constraint is mapped to an \( A \)-association from the referring \( E \)-class to the referred \( E \)-class identified by the foreign key attribute.

### 3.2 Transformation Operation

To enrich the semantics of a global schema with those implicit semantics as stated in subsection 2.3, necessary semantic transformation operations are defined for the three types of semantics. At first, some notations are given:

- \( E \) denotes an \( E \)-class and \( Id(E) \) denotes the identifier attribute(s) of \( E \).
- \( G \text{-subc}(E) \) denotes the set of the \( G \)-association sub-classes of \( E \).
- \( l \text{-role}(E) \) denotes the set of the \( l \)-association role-classes of \( E \).
- \( A \text{-attr}(E) \) denotes the set of the \( A \)-association attributes of \( E \).
- An equation \( A := B \) means that the elements of the set \( B \) are assigned to the set \( A \).

Next, five transformation operations are defined and shown in Figure 1 by Semantic Diagram.

**Definition 1. GENERALIZE** Given \( n(n \geq 1) \) \( E \)-classes \( E_1, E_2, ..., E_n \), if following conditions are held: (1) there is no \( G \)-association between any two of them, (2) they have the same identifier attribute: \( Id(E_1) = Id(E_2) = ... = Id(E_n) \), (3) they have the same \( A \)-association sub-set: \( \{ A_1, ..., A_q \} \), then a GENERALIZE operation on them creates a generalization hierarchy in which a new \( E \) is the super-class and \( E_1, E_2, ..., E_n \) are sub-classes, and \( G \text{-subc}(E) := \{ E_1, E_2, ..., E_n \} \), \( A \text{-attr}(E) := \{ A_1, A_2, ..., A_q \} \), \( A \text{-attr}(E_i) := A \text{-attr}(E_i) - \{ A_1, A_2, ..., A_q \} \) for each \( i(1 \leq i \leq n) \).

**Definition 2. MERGE** Given \( n(n \geq 2) \) \( E \)-classes \( E_1, E_2, ..., E_n \), if they have the same structural properties, that is, for each \( i,j(1 < i, j \leq n \) and \( i \neq j \)), following conditions are held: (1) \( Id(E_i) = Id(E_j) \), (2) \( G \text{-subc}(E_i) = G \text{-subc}(E_j) \), (3) \( l \text{-role}(E_i) = l \text{-role}(E_j) \), (4) \( A \text{-attr}(E_i) = A \text{-attr}(E_j) \), then a MERGE operation on them generates a new \( E \)-class \( E \), and \( Id(E) := Id(E_1) \), \( l \text{-role}(E) := l \text{-role}(E_1) \), \( A \text{-attr}(E) := A \text{-attr}(E_1) \).

**Definition 3. GLINK** Given two \( E \)-classes \( E_1 \) and \( E_2 \), if following conditions are held: (1) there is no any direct or indirect \( G \)-associations between them, (2) \( G \text{-subc}(E_1) = \phi \), (3) \( Id(E_1) = Id(E_2) \), (4) \( A \text{-attr}(E_1) \subseteq A \text{-attr}(E_2) \), then, a GLINK operation on them creates a generalization hierarchy in which \( E_1 \) is the super-class and \( E_2 \) is the sub-class, and \( A \text{-attr}(E_1') := A \text{-attr}(E_1) \), \( l \text{-role}(E_1') := l \text{-role}(E_1) \), \( G \text{-subc}(E_1') := \{ E_2 \} \), \( G \text{-attr}(E_1') := G \text{-attr}(E_2) \), \( l \text{-role}(E_1') := l \text{-role}(E_2) \), \( A \text{-attr}(E_2') := A \text{-attr}(E_2) - A \text{-attr}(E_1) \).

**Definition 4. INTERACT** Given three \( E \)-classes \( E_1, E_2, E_3 \), if following conditions are held: (1) there is no any associations between any two of them, (2) \( Id(E_1) = Id(E_2) \cup Id(E_3) \), (3) \( Id(E_2) \cap Id(E_3) = \phi \), then an INTERACT operation on them creates an interaction hierarchy in which \( E_2 \) and \( E_3 \) are two role classes of \( E_1 \), and \( l \text{-role}(E_1) := \{ E_2, E_3 \} \).
Definition 5. **AGGR.EGATE** Given an E-class \( E_1 \) and \( A-attr(E_1) = \{A_1, A_2, ..., A_m, ..., A_p\} \), then an **AGGREGATE** operation on it creates an aggregation hierarchy in which new E-class \( E \) is a member class and \( E'_1 \) is an owner class, and \( A-attr(E) := \{A_k+1, ..., A_q\}, G-subc(E'_1) := G-subc(E_1), I-role(E'_1) := I-role(E_1), A-attr(E'_1) := \{A_1, ..., A_k, E\}. \)

**Figure 1:** Transformation Operation

4 Transformation Statement

According to two transformation phases, two sets of transformation statements are provided respectively for the global schema designer.

First, a set of definition statements which are compatible with OSDL[15] is provided for the schema translation in Phase 1. The brief formats of three basic statements are given as follows:

1. **DECLARE** statement
   
   DECLARE <D-class name> =  
   
   - <Sys-defined D-class>;  
   - <User-defined D-class>;  
   - <data-constructor>  
   of <atomic data type>;  

   The statement defines a D-class for a data type of a local schema.

2. **CREATE** statement
   
   CREATE <Base E-class name> A <A-attr-list>  
   from <local relation name>  
   composite-key <A-attr-list>;  

   The statement defines a Base E-class for a local relation. The attributes from the local schema can be renamed or filtered out during the definition.

3. **FOREIGN-KEY** statement
   
   FOREIGN-KEY of <referring E-class name>  
   from <referred E-class name>  
   ( <foreign-key desc-list> );  

   The statement maps a foreign key constraint among local relational schemas by an A-association.

Next, a set of transformation statements for semantics enriching in Phase 2 are defined. The main statements are defined in terms of transformation operations in section 3.2. The brief formats of five basic statements are given as follows:

1. **GENERALIZE** statement
   
   GENERALIZE <sub-E-class desc-list>  
   into <super-E-class name>  
   [ constraint (<constraint-desc>) ];  

   The statement captures the generic semantics of one or more specific E-classes.

2. **MERGE** statement
   
   MERGE <E-class name-list>  
   into <E-class name>;  

   The statement captures the shared semantics of several E-classes.

3. **GLINK** statement
   
   GLINK <sub-E-class name>  
   to <super-E-class name>  
   [ constraint (<constraint-desc>) ];  

   The statement expresses the generalization semantics between two E-classes.

4. **INTERACT** statement
   
   INTERACT (<role E-class desc-list>)  
   into <E-class name> (<role-desc-list>)  
   [ constraint (<constraint-desc>) ];  

304
The statement expresses the interaction relationship semantics among different E-classes.

(5) **AGGREGATE** statement

AGGREGATE (<attr-name-list>)
  of <owner-E-class name>
  into <member-E-class name>
  [ constraint (<constraint-desc> ) ]

The statement captures the specific semantics for a general E-class.

In addition, other statements for schema reconstructing such as rename and drop are also provided. Since no new concepts in them, their descriptions are neglected.

The relational schema has foreign key constraints such as:

- `FACULTY.fac dept C DEPT.dept no`
- `DEPT.chairman C FACULTY.fac no`
- `TA.major C DEPT.dept no`

In Phase 1, the relational schema is translated into a **Base OSAM* schema** as shown in Figure 2, in which some attributes are renamed by **DEFINE** statement such as `TA.ta no` is renamed as `ss#`. In Phase 2, following transformation statements are applied upon the **Base schema**, and then a final global schema is obtained as shown in Figure 3.

Finally, an example is given to illustrate how to transform a relational schema into an **OSAM* schema**. The relational schema is given as follows, where * denotes the primary key of a relation.

```sql
FACULTY(fac no*: INT, fac.name: VARCHAR(20),
  degree: INT, specialty: VARCHAR(8), fac dept: INT);
DEPT(dept no*: INT, name: VARCHAR(20),
  chairman: INT, college: VARCHAR(30));
TA(ta no*: INT, ta name: VARCHAR(20),
  major: INT,
  gpa: CHAR(2), mode: CHAR(1), tac: INT, sex: CHAR);
RA(ra no*: INT, ra name: VARCHAR(20),
  major: INT,
  gpa: CHAR(2), mode: CHAR(1), rac: INT, age: INT);
UNDERGRAD(ta no*: INT, ta name: VARCHAR(20),
  major: INT, gpa: CHAR(2), minor: INT;
ADVISING1(fac no*: INT, ta no*: INT, start date: DATE);
ADVISING2(fac no*: INT, ra no*: INT, start date: DATE);
```

**Figure 2**: Basic **OSAM* schema after Phase 1

**Figure 3**: Final **OSAM* schema after Phase 2
5 Query Translation

In an OSAM* based database system, the objects are represented by G-relations \([ll][13]\). A G-relation is an extended relation supporting OID and nested structures. For OSDL query processing, the G-relations of each Base E-class and each Derived E-class need be generated. The generation rules are:

1. The G-relation of each Base E-class is generated from the corresponding Local relation.
2. The G-relation of each Derived E-class is represented by the mapping from the Base E-class.

Therefore, all manipulations on a Base E-class or a Derived E-class are actually performed on the G-relations of Base E-classes. Because the mapping between each Base E-class and each local relation is one-to-one, and constraints of the local schema are also enforced by the global schema, the global manipulation can be correctly executed on the local relations.

Next, the generation algorithm and mapping rules for Base E-class and Derived E-class are presented, in which following denotations are used:

1. \(E, E_1, E_2, \ldots, E_n\) denote the E-classes of relation \(R, R_1, R_2, \ldots, R_n\), respectively.
2. \(GRR\) denotes the G-relation of a relation \(R\);
3. \(GRE_1, GRE_2, \ldots, GRE_n\) denote the G-relation of E-classes: \(E, E_1, E_2, \ldots, E_n\), respectively.
4. \(Attr(R)\) and \(Key(R)\) denote the attribute set and primary key of a relation \(R\), respectively.
5. \(Sub-attr(R)\) and \(Sub-attr(GRE)\) denote the attribute subset of a relation \(R\) and a G-relation \(GRE\), respectively.

5.1 G-Relation of Base E-class

The G-Relation of a Base E-class is directly generated from a local relation.

**Input:** Local relation \(R_1, R_2, \ldots, R_n\), Base E-class \(E_1, E_2, \ldots, E_n\).

**Output:** G-Relation \(GRE_1, GRE_2, \ldots, GRE_n\).

**Algorithm:**

1. For each local relation \(R_i(1 \leq i \leq n)\), create an OID-allocation table (OAT) by assigning the primary key of each tuple with a unique OID:
   \[
   OAT(R_i) := \{(OID, Key(R_i))\}
   \]
2. For each local relation \(R_i(1 \leq i \leq n)\), create an initial G-relation \(GRR_i\) by a natural join operation:
   \[
   GRR_i := R_i \bowtie OAT(R_i)
   \]
3. For each Derived E-class \(E_i(1 \leq i \leq n)\), create a G-relation from the G-relation of the corresponding local relation \(R_i\):
   \[
   GRE_i := \Pi_{Sub-attr(GRE)}(\pi_p(GRR_i))
   \]
   where \(p\) is a predicate in the upper-level query expression and is used for reducing the size of the relation.

5.2 G-Relation of Derived E-class

A Derived E-class is generated from a Base E-class and/or other existing Derived E-classes. After all Base E-classes are generated, the G-relations of a Derived E-classes can be expressed by the relations of related E-classes. The conversion expressions are defined in terms of the transformation operations which create the Derived E-class.

**Expression 1. GENERALIZE** Suppose \(E\) is an E-class generated from the classes: \(E_1, E_2, \ldots, E_n\) by a GENERALIZE operation, then
\[
GRE = \bigcup_{i=1}^{n} \left( \Pi_{A-attr(E_i)}(GRE_i) \right)
\]

**Expression 2. MERGE** Suppose \(E\) is an E-class generated from \(E_1, E_2, \ldots, E_n\) by a MERGE operation, then
\[
GRE = \bigcup_{i=1}^{n} \left( \Pi_{A-attr(E_i)}(GRE_i) \right)
\]

**Expression 3. GLINK** Suppose \(E_1\) is a super-class in a generalization hierarchy generated from two E-class \(E_1\) and \(E_2\) by a GLINK operation, then
\[
GRE_1 = \bigcup_{i=1}^{n} \left( \Pi_{A-attr(E_1)}(GRE_2) \right)
\]

**Expression 4. AGGREGATE** Suppose \(E\) is a specific E-class generated from the the attribute subset of a general E-class \(E_1\) by an AGGREGATE operation, then
\[
GRE := \Pi_{Sub-attr(GRE_1)}(GRE_1)
\]

As to other operations such as INTERACT and FOREIGN-KEY, because they only affect the structures of the global schema and do not compose or decompose object instances, their conversions can be done simply by their definitions.
6 Conclusions

In this paper, a 2-phase transformation method is presented for integrating a relation database into an object-oriented database. Moreover, a schema transformation system has been implemented for CIMBASE. By this system the pre-existing Ingres and ORACLE databases can be integrated into a CIMBASE database. The method has features as follows:

- Supporting global data transparency. The application on global schemas need not know the local information since all conversions are automatically performed by the transformation mechanism.

- Supporting local data autonomy. The applications on local schemas are not affected after the transformation because the transformation mechanism does not make any changes on local schemas and a global database is only a virtual view of a local database.

- Supporting dynamic manipulations on relational databases from the global level because the data consistency of global view and local database are automatically maintained by the conversion algorithm and expressions and all constraints on local databases are also enforced by the global level.

- Supporting smooth evolution from a relational database to an object-oriented database. The data in a local database can be converted into the global database by the transformation mechanism when it is required to replace an old DBMS by a new DBMS.

The future works include: (1) to develop a graphical transformation tool and improve the efficiency of the automatic transformation; (2) to keep the dynamic consistency between the local schema and the global schema, that is, to make the changes in the local schema be reflected in the global schema synchronously. and (3) to extend the 2-phase transformation method to handle Network and Hierarchical databases.

References


