Converting Relational to Object-Oriented Databases

Joseph Fong
Computer Science Department, City University of Hong Kong, Kowloon, Hong Kong,
email: csjfong@cityu.edu.hk

Abstract

As object-oriented model becomes the trend of database technology, there is a need to convert relational to object-oriented database system to improve productivity and flexibility. The changeover includes schema translation, data conversion and program conversion. This paper describes a methodology for integrating schema translation and data conversion. Schema translation involves semantic reconstruction and the mapping of relational schema into object-oriented schema. Data conversion involves unloading tuples of relations into sequential files and reloading them into object-oriented classes files. The methodology preserves the constraints of the relational database by mapping the equivalent data dependencies.

Keywords: data conversion, relational database, object-oriented database, data dependencies

1 Introduction

Relational databases have over the last decade become an accepted solution to the issue of storing and retrieving data. Based upon the mathematical concept of a relation, these systems use tables (relations) and fixed sized field (domains) to represent the information and its inter-relationships. The mathematical rigour and simplicity of these system has been their major attraction. However, there are many drawbacks to such database systems. For one thing, the semantics of relational databases are often hidden within the many relationships which cannot be extracted without users' help.

Object-Oriented databases (OODB) offer solutions to many of these problems. Based on the notions of abstraction and generalization object oriented models capture the semantics and complexity of the data. Fundamental to the object-oriented approach are the concepts of class, instance and inheritance. An instance is an occurrence of a class, where a class is a description of an entity. Classes may inherit the attributes of one or more superclass(es) and thus capture some of the semantics of an object. Also OODB supports arbitrary data types such as dynamic addition of new data types from existing data types. An object-oriented model is thus more reusable and flexible in schema evolution and data storage.

Data conversion and schema translation have been studied by many authors (Marionos and Smit,1991; Fong and Ho,1993; Fong and Bloor,1994; Getta,1993; Liu et al,1993; Hainaut et al,1993). Without a well defined notion of equivalence between schema expressed in possibly different data models, initial efforts were a little ad hoc. As the understanding of equivalence developed (Lien,1982;Markowitz,1992) these efforts became more refined. Recent definitions of information capacity (Hull,1986) offer more flexibility than the earlier model of equivalence by introducing the notion of dominance.

To convert data from relational model to an object-oriented model, the prerequisite step is to translate a relational schema to an object-oriented (OO) schema. The methodology provides a systematic approach to solve different problems in schema and data conversion and is theoretical sound in preserving the semantics and constraints of relational database.

2 Methodology for data conversion from relational to object-oriented

In our method of data conversion, we have three steps which include schema translation, unload and reload data from relational to OODB as follows:

Step 1: Translate relational schema to OO schema

A database is said to be schema-equivalent to another database if there exists a mapping that maps the schema S_1 of the first database to the schema S_2 of the second database, such that all constraints(properties) in S_1, that are essential in the context of the first database, can be preserved in S_2. Two basic constraints in relational and object-oriented schemas are functional dependency(FD) and inclusion dependency(ID), and these can be further described as follows:

Definition of FD
Attribute Y is functionally dependent on attribute X, i.e., FD: $X \rightarrow Y$, iff each X-value has associated with it precisely one Y-value(at any one time). Attributes X and Y may be composite.

Definition of ID

ID: $Y \subseteq Z$ states that the set of values appearing in attribute Y must be a subset of the set of values appearing in attribute Z.

The mapping from a relational to an object-oriented schema is said to be based on a one-to-one correspondence between relations and class objects. The features of an class object can be contrasted with those of a relation as follows:

Rule 1: Relation $\Rightarrow$ class object

This first rule is a mandatory mapping which turns relations into class objects, allowing all future mappings to be carried out in the semantically richer object-oriented data model. The resulting classes contain all the attributes of the source relations. This can be illustrated by mapping relation Department into class object Department as follows:

Relational schema
Relation Department(Dept#, Dept-name)
FD: Dept# $\rightarrow$ Dept-name

Mapped OO schema
Class Department
attr Dept#: integer
attr Dept-name: string
end
FD:Department.OID $\rightarrow$ Dept#, Dept-name

Note: OID is object identifier. Each object must have an OID, an unique number created by OODB(Hughes, 1991).

Rule 2: Foreign key $\Rightarrow$ Association attribute

This mapping takes the value determined relationships of the relational model and maps them into association attributes in the OO model. The foreign key attributes are then dropped from the class, leaving the class with semantically meaningful attributes and association attributes with other classes.

Such mapping can be further described as follows:

Case 1: One-to-many or one-to-one relationship are represented by a containment hierarchy and "inverse variable".

For example, for a parent relation Department and a child relation Instructor, there is a foreign key of Dept# in the child relation Instructor which establishes the linkage between these two relations. When we map the relation Department in relational schema, we can have a class object Department which has a complex attribute Instructor such that the attributes and the methods of an independent class object Instructor is contained in the same class object Department as shown in Figure 1.

Relational schema
Relations Department(Dept#, Dept-name)
Relation Instructor(Inst-name, Inst-addr, *Dept#)
ID:Instructor.Dept# $\subseteq$ Department.Dept#

Mapped OO schema
Class Department
attr Dept#: integer
attr Dept-name: string
association attr hire ref set(Instructor)
end
Class Instructor
attr Inst-name: string
attr Inst-addr: string
association attr hired-by ref Department
end
ID:hired-by $\subseteq$ Department.OID

Similarly, when we map the relation instructor in relational schema to OO schema, we can use inverse variable such that in the two separate class object: Department and Instructor, there are (a) a set of instructors in the department, and (b) an department instance variable in every instructor. Thus, association attribute hire and hired-by are said to be inverses of each other, and they need to be kept mutually consistent as shown in Figure 2.(Date, 1995)

Case 2: Many-to-many relationship is represented by a containment hierarchy and "inverse variable".

For example, for relation Instructor and relation Course in m:n relationship, we have a relationship relation Section in relational schema. In OO schema, we can have two class object Instructor and Course, and a class object Section linked to each other through containment hierarchy and "inverse variable" as shown below:

<u>Relational schema</u>
Relations Course(<u>Course#</u>, Course-title)
Relation Student(<u>Student#</u>, Student_name)
Enroll(<u>Course#</u>, <u>Student#</u>, Grade)
ID:Enroll.Course# ⊆ Course.Course#
ID:Enroll.Student# ⊆ Student.Student#

<u>Mapped OO schema</u>
Class Course
 attr Course#: integer
 attr Course-title: string
end
Class Student
 attr Student#: integer
 attr Student_name: string
end
Class Enroll
 attr Grade: string
 association attr provided-by ref Course
 association attr registered-by ref Student
 end
ID:provided-by ⊆ Course.OID
ID:registered-by ⊆ Student.OID

Rule 3: isa relationship ⇒ inheritance

The subclass-to-superclass(i.e. isa) relationships in relational schema is represented by the class hierarchy. For example, relations Student and Graduate_student are in isa relationship such that the primary key of subclass relation Graduate_student is a subset of the primary key of relation Student. When we map these relations to OO schema, we have class object Student, class object Graduate_student such that subclass object Graduate_student inherits the attributes and methods of superclass object Student as shown in Figure 3.

<u>Relational schema</u>
Relation Student(<u>Student#,</u> Student_name)
Relation Graduate_student(<u>Student#</u>, Degree)
ID:Graduate_student.student# ⊆ Student.Student#

<u>Mapped OO schema</u>
Class Student
 attr Student#: integer
 attr Student_name: string
end
Class Graduate_student
 inherit Student
 attr Degree: string
end
ID:Graduate_student.OID ⊆ All Student.OID

Note: All class object OID means set of all OID of the deep extent of a class object that includes the OIDs of the instances of its subclass object and the OIDs of the instances which does not have a subclass object.

Step 2 Unload relations' tuples into sequential files

According to the translated OO schema, the tuples of each relation will be unloaded into a sequential file. The unload process will be divided into three substeps.

The first substep is to unload each relation's tuple into a file with all insert statements. (note: These statements will later be reloaded back to a target OODB such that each class will be initially loaded from the tuples of a corresponding relation.)

In the second substep, for each foreign key, its referred parent relation's tuple will be unloaded into another file with update statements. The idea is to make use of the stored OID when reloading the insert statement in the first substep. The update statement is to replace the association attribute when they are reloaded back to a target OODB.

In the third substep, for each subclass relation, its referred superclass relation's tuple will be loaded into a third file with update statements. (note: The idea is also to make use of the stored OID when reloading the insert statement in the first substep. The update statement is to implement the inheritance statements when they are reloaded back to a target OODB.)

The pseudo code for this process can be described as follows:

```
Begin
 Get all relation R1, R2...Rn within relational
 schema;

 For i = 1 to n do /*1st substep: load each class
             with corresponding relation tuple data */
 begin
  while Ri tuple found do
  Output non-foreign key attribute value to a
  sequential file Fi with insert statement;
 end;

 For j = 1 to n do /*2nd substep:update each loaded
             class with its association attribute value */
 begin
  while Rj tuple with a non-null foreign key value
   found do
  begin
   Get the referred parent relation's tuple from Rp
   where Rp is a parent relation to Rj;
  Output the referred parent relation's tuple to a
   sequential file Fj with update statement;
  end;

 For k = 1 to n do /*3rd substep:update each
  subclass to inherit its superclass attribute value */
 begin
  while a subclass relation Rk tuple found do
  begin
   Get the referred superclass relation's tuple from
   Rs where Rs is a superclass relation to Rk;
   Output the referred superclass relation's tuple to
   a sequential file Fk with update statement;
  end;
end;
```

Step 3 Reload sequential files to object-oriented database

As a prerequisite of the data conversion, a schema translation from relational to object-oriented model must have done beforehand. The translated object-oriented schema will then be created in the OODB. The sequential file F_i will first be reloaded into OODB to fill in the class attributes' values. The sequential file F_j will then be reloaded into the OODB to fill in each class association attribute's values. Lastly, the sequential file F_k will be reloaded to fill in each subclass's inherited attributes values.

3 Case study of data conversion from relational to object-oriented

To illustrate the application of the above methodology, we can use a modified University Enrollment system as an example.
We can apply the above algorithm to convert the following relations into OODB:

Relation Graduate Student

Student#	Degree
012888	M.Sc.
120008	M.B.A.

Relation Student

Student#	Student_name	Sex
012888	Paul Chitson	M
120008	Irene Kwan	F
117402	John Lee	M

Relation Course

Course	Course_title
CS101	Intro to Computer Science
IS201	System Analysis
IS301	Decision Support Systems

Relation Enroll

*Student	*Course	Grade
012888	CS101	A
120008	CS101	B

Its translated object-oriented schema is as follows:

```
Class Student
 attr student#: integer
 attr student_name: string
 attr sex: string
end

Class Graduate student
inherit Student
attr degree: string
end
```

```
Class Course
 attr course: string
 attr course_title: string
end

Class Enrol
 attr grade: string
 association attr registered-by ref student
 association attr provided-by ref course
end
```

By applying the algorithm specified, this step unloads data from each relation into a sequential file along with its association data from other relations.

The idea is to load the attribute data from the input relation, and the association attribute data from the loaded object-oriented database. The Select statement is to retrieve class occurrence that have been loaded into the object-oriented database with the stored OID. The association attributes are in italic.

For implementation, foreign key will be loaded with referred data using the stored OID. The insert and select statements in the followings are from a prototype written using UniSQL(UniSQL, 1993).

The content of file F_i after the first substep in unload process are the following insert statements:

insert into student (student#, student_name, sex) values (‘012888’, ‘Paul Chitson’, ‘M’)

insert into student (student#, student_name,
 sex) values (‘120008’, ‘Irene Kwan’, ‘F’)

insert into student (student#, student_name,
 sex) values (‘117402’, ‘John Lee’, ‘M’)

insert into graduate_student (student#,
 degree, *isa*) values (‘012888’,‘M.Sc.’,null)

insert into graduate_student (student#,
 degree,*isa*) values (‘120008’,‘M.B.A.’null)

insert into Enroll (grade, *registered-by*,
 provided-by) values (‘A’, *null, null*)

insert into Enroll (grade, *registered-by*,
 provided-by) values (‘B’, *null, null*)

insert into course (course, course_title),
values (‘IS101’, ‘Introduction to
 Computer Science’)

insert into course (course, course_title)
 values (‘IS201’, ‘System Analysis’)

insert into course(course, course_title),
 values (‘IS301’, ‘Decision Support
Systems’)

The content of file F_j after second substep are:

update Enroll
set *registered-by* = (select * from student
 where student# = ‘012888’)
set *provided-by* = (select * from course
where course = ‘IS101’)
where grade = ‘A’

update Enroll
set *registered-by* = (select * from
 graduate_student where student# =
 ‘120008’)
set *provided-by* = (select * from course
 where cours# = ‘IS101’)
where grade = ‘B’

The content of file F_k after third substep are:

update graduate_student
set student_name = (select * from student
 where student# = ‘012888’)
set sex = (select * from student where
 student# = ‘012888’)
where student# = ‘012888’

update graduate_student
set student_name = (select * from student
 where student# = ‘120008’)
set sex = (select * from student where
 student# = ‘1200008’)
where student# = ‘1200008’

4 Conclusion

A solution for data conversion from relational database into OODB has been proposed. The suggested methodology can unload data from a relational database into data for each class in an OODB. This data can then be reloaded into the OODB using the translated Object-Oriented schema. Such a conversion can map the data structure and semantics from relation tables in relational database to classes in OODB without loss of information. The primitive constraints of FDs and IDs can be preserved after schema translation and data conversion. These can be used to construct semantics such as cardinality, generalization, categorization and aggregation in both relational database (Elmasri &

Navathe,1989) and OODB (Fong,1995). The continuation research in the same area will permit program conversion between relational database into OODB.

Reference

Date, C.J. (1995) An Introduction to Database Systems (6th edition) Addison-Wesley Systems Programming Series, pp669-685.

Elmasri, R.A., & Navathe, S., (1989). Fundamentals of Database Systems. The Bensamin/Cummings Publishing Company, Inc.

Fong, J., & Ho, M., (1993). Knowledge-Based Approach for abstracting Hierarchical and Network Semantics, Lecture notes in Computer Science 823, Springer-Verlag, pp 509-519.

Fong, J., & Bloor, C., (1994). Data Conversion Rules from Network to Relational Databases, Information and Software Technology, vol 36, 1994, no 3, pp141-153.

Fong, J., (1995). Mapping Extended Entity Relationship Model to Object Modeling Technique, September 1995 ACM SIGMOD RECORD, pp76-82.

Getta, J.R., (1993). Translation of Extended Entity-Relationship Database Model into Object-Oriented Database Model, in (IFIP A-25) Interoperable Database Systems (DS-5), (edited by Hsiao et al) Elsevier Science Publishers B.V. (North-Holland), pp 87-100.

Hainant, J-L, et al (1993) Transformation-Based Database Reverse Engineering, Lecture notes in Computer Science 823, Springer-Verlag, pp. 364-375.

Hughes, J (1991) Object-Oriented Databases, Prentice Hall International Series in Computer Science

Hull, R., (1986). Relative Information Capacity of Simple Relational Schemata, SIAM Journal of Computing, vol 15, no 3, pp 856-886.

Hull, R., & Su, S., (1989). On Accessing Object-Oriented Databases: Expressive Power, Complexity, and Restrictions, Proceedings of SIGMOD International Conference, pp 147-158.

Lien, E., (1982) On the Equivalence of Database Models, JACM, v29, n2, pp 333-362.

Liu, C.T., et al (1993). An Entity-Relationship Approach to Schema Evolution, IEEE ICCI'93, pp 575-578.

Marinos, L., & Smit, R.A., (1991). From Relations to Objects: A Translation Methodology for an Object Oriented Front-End to RDBMSs, in (Papazoglou & Zeleznikow,1991), pp 148-167.

Markowitz, V.M., & Shoshani, A., (1992). Representing Extended Entity-Relationship Structures in Relational Databases: A Modular Approach, ACM TODS, v17, n3, pp 423-464.

UniSQL (1993). UniSQL Database Management System User Manual, UniSQL inc.