Object-Relational Database Management System (ORDBMS) Using Frame Model Approach

H K Wong and Anthony S. Fong
Department of Electronic Engineering
City University of Hong Kong
Tat Chee Ave, Kowloon, Hong Kong
E-mail: eefong@cityu.edu.hk

ABSTRACT

This paper describes a methodology and development approach to implement Object Relational Database Management System (ORDBMS) with Frame Model (Fong's 1997). The Frame Model method is a solution to extend Relational Database Management System (RDBMS) to handle complex data such as objects. Objects can be represented in Frame Model schema as classes, constraints and methods. Schema translation is the first step to implement the Object-Relational Database Management System (ORDBMS). This paper investigates a stepwise methodology for schema translation from a Relation Model to a Frame Model. The second step is the implementation of ORDBMS to incorporate RDBMS as kernel, Application Program Interface (API) as interactive SQL and a method command interpreter. Methods are pre-compiled and stored in RDBMS as stored procedures. Methods can be invoked directly through API or a constraint class that is defined in the Frame Model schema when the associated interactive SQL is issued from the API.

1. INTRODUCTION

Relational Database (RDB) has been a legacy system even though it has limitations on multimedia system, foreign key data redundancy and intensive engineering applications. Object-oriented Database Management System (OODBMS) is capable to handle complex data such as object. As a result, it is abstractive, flexible and productive in database system development and enhancement. The success of relational database system does not come simply from a higher level of data independence and simplified data model than the preceding systems such as network model and hierarchical model, but from standardization that it offers. The acceptance of the Structured Query Language (SQL) as a de facto standard allows a high degree of portability and interoperability among systems, simplifies the learning of new RDBMS, and represents a wide endorsement of the relation approach. Unfortunately, OODBMS has been slow to add SQL support to the systems, thereby making legacy systems very difficult to migrate or convert. There is a great gap between RDBMS and OODBMS. Consequently a new Object Relational Database Management System (ORDBMS) is designed to bridge the gap between RDBMS and OODBMS, and to provide object significant features and maintain the advantages of RDBMS.

2. FRAME MODEL

The Frame Model is similar to an object-oriented database that structures an application domain into classes and its data into relational tables. These classes are organized via generalization, aggregation and user-defined relationships. The Frame Model is significant as it consists of a set of data, as well as active and dynamic data structure of the legacy data models, with the constraint structures to resolve their synonyms and homonyms conflicts. These constraints include integrity constraint enforcement, derived data maintenance, triggers, protection, and version control, etc. The Frame Model unifies data and rules to enable these advanced features to be implemented effectively.

The Frame Model follows the object-oriented paradigm. All the conceptual entities are modeled as objects. Similar attributes and behavior objects are grouped into a class. Both facts and rules are viewed as objects in the Frame Model design. The Frame Model logical schema can be arranged in a class format and a relation format. The Frame Model logical schema in a class format is shown in Table 1 (Fong and Huang, 1997).

Header_Class {	
Class_Name	/* a unique name in all systems */
Parents	/* a list of class names */
Operation	/* program call for operations */
Class_Type	/* type of a class, e.g. active or static */

0-7695-0540-6/00 $10.00 © 2000 IEEE

```
}

Attribute_Class {
  Attribute_Name/* a unique name in this class */
  Class_Name    /* reference to the header class */
  Method_Name   /* a unique name in this class */
  Attribute_Type /* the data type for the attribute */
  Default_Value /* a predefined value for the attribute */
  Cardinality   /* single or multi-valued */
  Description   /* description of the attribute */
}

Method_Class {
  Method_Name   /* a unique name in this class */
  Class_Name    /* reference to the header class */
  Parameters    /* a list of arguments for the method */
  Method_Type   /* the output data type */
  Condition     /* the rule conditions */
  Action        /* the rule actions */
}

Constraint_Class {
  Constraint_Name /* a unique name for each constraint */
  Class_Name    /* reference to the header class */
  Method_Name   /* constraint method name */
  Parameters    /* a list of arguments for the method */
  Ownership     /* the class name of the method owner */
  Event         /* triggered event */
  Sequence      /* method action time */
  Timing        /* the method action timer */
}
```

Table 1: The Logical Schema of the Frame Model in Class Format.

The Frame Model consists of two classes: static and active (dynamic). Static classes represent factual data entities and active classes represent rule entities. An active class is event driven, obtaining data from the database when invoked by a certain event. The static class stores data in its own database. The two classes use the same structure. Combining these two types of objects within the inheritance hierarchy structure enables the Frame Model to represent heterogeneous knowledge.

3. OVERVIEW ARCHITECTURE

The architecture of Frame Model ORDBMS is composed of Schema Translation and Object Frame Model Agent (OFMA).

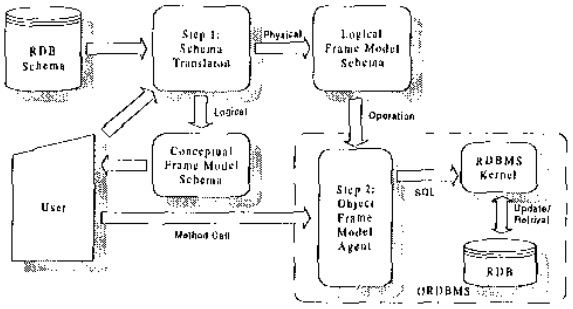

Figure 1: The ORDBMS Architecture.

As shown in Figure 1, the ORDBMS architecture consists of Schema Translation, OFMA and RDBMS. The Schema Translation is to translate a relational schema to a Frame Model schema interactively with confirmation. The OFMA is to translate the object-oriented query (methods) to SQL syntax and methods syntax. The OFMA is divided into two parts: the SQL interpreter and the method interpreter that are described in more details in later part of this paper. The Relational Database Management System (RDBMS) and Relation Database (RDB) are to store the logical schema of the Frame Model in class format and data respectively.

4. METHODOLOGY OF OBJECT RELATIONAL DATABASE MANAGEMENT SYSTEM

Schema Translation is a process to convert a schema expressed in one data model into an equivalent schema expressed in a different data model. During the process of schema translation, all the functional and inclusion dependencies in the schema must be identified and preserved. The translated schema must be logically equivalent to the original schema. Schema Translation is significant and commonly used in developing interoperable systems for various database models. It enables database interoperability in a multi-database environment. The ideal situation is to provide a multi-database model, e.g. a universal database model, which allows users to access various databases without concerning the underlying data models.

Object Frame Model Agent (OFMA) is an application program interface (API) that manipulates both standard SQL and method operations. Frame Model methodology employs a set of definition tables in RDBMS to represent the object, object instance, object inheritance and object relationship. The object definitions are stored in four systems tables: Header Class, Attributes Class, Method Class and Constraint Class. Activities that related to object behavior will be queried from the four classes

definition tables. Such activities will be re-modeled and executed in RDBMS as defined by Frame Model Classes.

4.1 Methodology for Schema Translation

In this paper, a stepwise methodology is investigated for Schema Translation from a Relational Model to a Frame Model. The following is a stepwise methodology to help users to capture a Relational Model schema to a Frame Model schema.

Discover Database Keys - By examining the relational schema, the database keys can be easily located, such as primary keys, foreign keys, composite keys, and the components of the composite keys, with user assistance.

Discover ISA Relationship - By examining the occurrences of the primary keys of the same name with the same domain, we can locate their ISA relationships. A user needs to input the type of class, including superclass-subclass type, subclass-superclass type and same class level, for each set of relations found.

Discover Cardinality - By examining the occurrences between a relation's primary key and another relation's foreign key, the cardinality between the two relations can be located. A user needs to input the type of cardinality, including one-to-one (1:1) relationship, one-to-many (1:n) relationship and many-to-many (m:n) relationship, for each set of relations found.

Discover Participation - By examining a parent relation and its child relation, one can determine its participation semantic by checking the null value of the foreign key of the child relation. If the foreign key of the child relation has a null value or it is allowed to have a null value, there is a partial participation between the parent and the child relations. If the foreign key of the child relation must be a non-null value, there is a total participation between the parent and the child relations.

Discover Generalization - By examining the primary keys among subclass relations under the same superclass relation, the generalization semantic can be easily determined. User needs to input the type of generalization, including disjoint generalization and overlap generalization, for each set of relations found.

Discover Categorization - By examining the primary key among superclass relations above the same subclass relation, the chance of having categorization semantic can be easily located. User needs to input whether each set of relations contains the categorization semantic.

Discover Aggregation - By examining the occurrences of the relations that consist of primary keys composed of all composite keys, they can be considered as relationship relations. If these relationship relations are also in referential integrity with other relations, then one maps these relationship relations to aggregation.

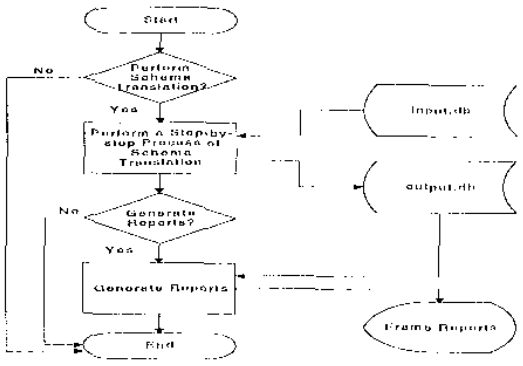

Figure 2: The Systems Flow Chart of the Schema Translation Mechanism.

4.2 Methodology for Object Frame Model Agent (OFMA)

The prototyping of Object Frame Model Agent (OFMA) is to construct an application program interface (API) that can manipulate both standard SQL and method operations. The API is implemented with Visual Basic 5.0, and incorporated with an RDBMS database and Sybase SQL Anywhere 5.0, to store the Frame Model Class definitions, stored procedures for method call and data tables for the existing application database.

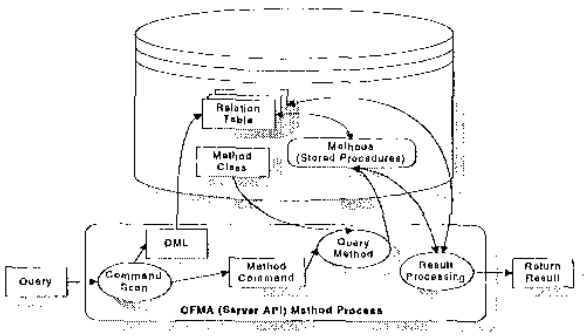

Figure 3: Process Diagram for OFMA Server API Process.

The OFMA Process

The API process handles user input command as shown in Figure 3. The command can be DML or Method. The API process first scans the input command, when the

command is DML command. OFMA then parses the DML directly to RDBMS. All syntax checking and execution are carried out by RDBMS. When the command is a method, OFMA decodes the input token to extract the method name, class name, and input parameters (if any). OFMA also performs syntax checking on the input token for error. OFMA is then to verify the existence of class name and method name from the Frame Model Class that was defined. When all syntax and verification have been validated, OFMA will invoke the store procedure that was pre-compiled for the corresponding method and parse the parameter that was previous stored (if any). After execution of the store procedure, OFMA will return result whatever returns by RDBMS.

Object Frame Model Agent (OFMA) Logic Flow

The implementation of OFMA is divided into three parts: the command scanner, the method interpreter and DML interpreter.

i. Command Scanner

The command scanner is to identify the input command type: DML or Method. The first word in the command line is compared with identifier that classifies the DML command or Method command. If the input command is DML, the DML Interpreter routine will be invoked, and all DML syntax and runtime error checking will be performed by RDBMS. When the first word in command line is not DML command, the Command Scanner will perform syntax checking and extract the input parameters for the method command.

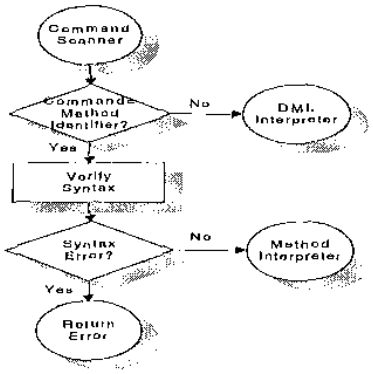

Figure 4: Logic Flow for Command Scanner.

ii. The Method Interpreter

When the command is looking for a method that is defined in the Method Class, the interpreter will decode the class name, the method name and the argument(s). This decoded information will verify by OFMA from the Frame Model schema. OFMA will then execute the command by invoking the associated stored procedure.

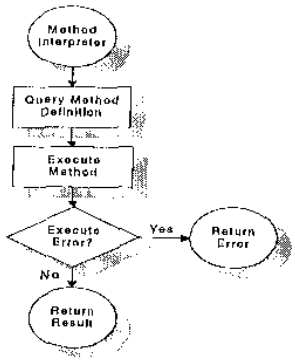

Figure 5: Logic Flow for Method Interpreter.

iii. The DML Interpreter

There are four kinds of SQL commands: select, insert, update and delete. For a SQL command that performs modification of data, OFMA will check for any constraint that was defined in Constraint Class. Execute the constraint method according to the triggered event and the sequence of firing, and then execute the SQL command.

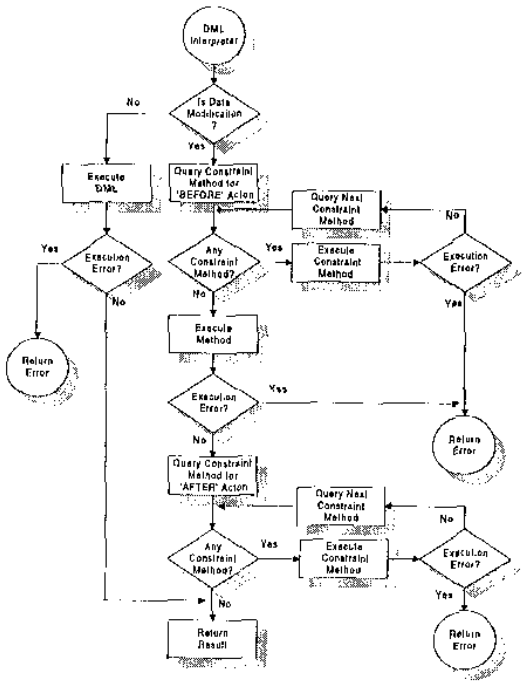

Figure 6: Logic Flow for DML Command Interpreter.

4.3 Object Oriented Behavior of OFMA

Encapsulation - Attributes and methods in an object are bounded within an object and some of them are not accessible outside the object. As a result the only way can an object be accessed and operated is through system function calls, which are the methods of the object to communicate with the external.

Inheritance - Attributes and methods in an object can be owned from its superclass(es) above on its branch of the class hierarchy. A class may inherit from a single class (single inheritance) or multiple superclasses (multiple inheritance).

Polymorphism - A message sent to different objects may have different corresponding behaviors. For example, the command new() creates an instance for patient and department will have different results for patient and department.

Event Driven - Some operations should be triggered for certain events happening to an object. For example, create a new record of a particular object, it will invoke / trigger another method call "check_duplicate" to check whether the record already exists or not.

Class Constraint - Additional constraint to enhance the referential integrity which does not provided by relational database, e.g. prototype class ward-record (Ward_rec) and outpatient-record (Outpatient_rec). In the case of creation a ward_rec record, a constraint method will check whether the record exists in Outpatient_rec or not. If the record has already existed in the Outpatient_rec, then the creation is not allowed.

5. CONCLUSION

Object-relational database adds new functions and flexibility for data storage and business modeling. As compared to relational database system, object-relational technology greatly expands the space of alternatives available to the schema design, such as using inheritance hierarchy, making inter-object references or key/foreign-key pairs, and using set-valued attributes to represent multi-valued attributes. Object-relational technology offers new options to overcoming weaknesses in pure objects or pure relation solutions.

Frame Model methodology employs a set of definition tables in RDBMS to represent the object, object instance, object inheritance, multi-valued attributes and objet relationship. Object definitions are stored in four systems tables: Header Class, Attributes Class, Method Class and Constraint Class. Activities that related to object behavior will query from the four classes definition tables. Such activities will be re-modeled and executed in RDBMS as defined by Frame Model Classes.

In this paper, ORDBMS, schema translation is to translate a Relational Model into a Frame Model schema, and several OO features are implemented such as encapsulation, polymorphism, inheritance, event driven and class constraints.

The possibility to use Frame Model Class definition is demonstrated in this paper by building a prototype interface allow user to input standard SQL command and method command in syntax "call method_name (parameter 1, parameter 2 ...) on class class_name".

6. REFERENCES

Fong, J. and S.M. Huang. 1996. Architecture of a Universal Database: A Frame Model Approach. Internal Paper. City University of Hong Kong. Hong Kong.

Fong, J. and S.M. Huang. 1997. Information Systems Reengineering. Springer-Verlag Singapore Pte. Ltd. Singapore.

Fong, J. 1992. Methodology for Schema Translation from Hierarchical or Network into Relational. Information and Software Technology. Hong Kong.

Johannesson, P. and K. Kalman. 1990. A Method for Translating Relational Schemas into Conceptual Schemas. Entity-Relationship Approach to Database Design and Querying. North-Holland.

Korth, H.F. and A. Silberschatz. 1991. Database System Concepts. McGraw-Hill, Inc. Singapore.

Loo, K.L. 1995. Interoperability of Relational Database and Object-Oriented Database. CS Project. City University of Hong Kong. Hong Kong.

Webb, J., McKelvy M., Martinsen, R., et la. 1995. Using Visual Basic 4. Que Corporation. IN.

Betting on ORDBMS on April 1998 Byte.

Sybase's SQL Anywhere 5.0 User Manual.