Malaysian Journal of Computer Science, Vol. 15 No. 1, June 2002, pp. 77-83

A PROPOSED OBJECT SCHEMA MANAGEMENT TOOL

Tong Ming Lim
School of Business and Information Technology
Monash University Sunway Malaysia
No 2 Jalan Kolej, Bandar Sunway
46150 Petaling Jaya
Selangor Darul Ehsan
Malaysia

Sai Peck Lee
Faculty of Computer Science & Information Technology
University of Malaya
50603 Kuala Lumpur
Malaysia
email: saipeck@fsktm.um.edu.my

ABSTRACT

Object Schema Management Tool (OSMT) is the primary graphical component in the Object Schema Management Facility (OSMF) that provides services such as remove a class, add a class, delete a class and manage a class for a business class domain [5]. In addition, the Object Schema Management Facility (OSMF) also contains a set of language constructs for Object Schema Language (OSL), which manages Business Class Schemas (BCS) for business domains in a persistence framework [1, 2, 3, 4]. This paper discusses the enhancements and functionalities completed for the latest version of OSMT. A discussion on various approaches that a class schema is managed and maintained in DBMSs currently is presented in the subsequent section. It gives an overview of a GUI-based business class schema tool, the generation of business classes in a separate text file manually through the Interface Definition Language (IDL) and the creation of business class schemas using OSL as proposed in this research. Next, the OSMF component is examined in greater detail. The paper also examines pros and cons of the OSMT component and discusses possible OSMT enhancements for the current OSMF component in the framework.

Keywords: ***Object Schema Management Tool, Object Schema Management Facility, Business Class Schema, Object Schema Language***

1.0 INTRODUCTION

The Object-oriented database management system (OODBMS) is the database system of the next generation due to its capability to handle complex information in CAD/CAM systems, multimedia systems and complex geographical management systems [7, 8, 9]. Legacy database management systems such as relational database management system (RDBMS) and multidimensional database management system (MDBMS) are lack of these features, hence, in order to allow existing applications to slowly migrate to the object technology and take advantage of the technology, emerging intermediate solutions such as object-relational database systems and object wrappers are rushing into the market in order to fill up the migration gap. This research paper provides a closer view on the improved version of the Object Schema Management Tool (OSMT) component in the Object Schema Management Facility (OSMF) in an object wrapper [1, 2, 3, 4]. These enhanced features are integrating class interfaces and methods' code using OSMT, defining and managing relationships among classes, loading and saving of classes and managing persistence objects of existing classes with OSMT.

2.0 BUSINESS CLASS SCHEMA MANAGEMENT

Business classes defined by developers must be easy to maintain and manage regardless of the type of database management system and the type of object-oriented programming language in use. This section looks at three different ways of managing business class schemas in the OSMF component. These three approaches are GUI-based OSMT, interface definition language (IDL) proposed by Object Data Management Group (ODMG) [9] and object schema language (OSL) proposed in this research. The class schema management facility proposed in this paper has a few objectives. First of all, a developer must have the ease of generating and maintaining business class schemas using a GUI tool. A GUI-based OSMT allows ease of class schema maintenance and productivity. Secondly, in order to allow more flexibility for business application developers, an object schema language is provided for manual changes. In addition, as the standard proposed by ODMG is revised, an object wrapper must have the flexibility to adapt to the latest standard in its own pace as well as maintaining forward and backward compatibility. Finally, the OSMT must not have limitations with respect to different database management systems and object-oriented programming languages.

2.1 Interface Definition Language

The IDL proposed by ODMG is the standard to which most commercial OODBMS products comply [9]. IDL is a complete business class schema definition language that allows a developer to capture and define the semantics of a business domain [9]. The developers have to remember the IDL syntax in order to define the business class schema correctly. Due to the fact that many commercial products do not come with a GUI tool that allows developers to quickly and easily maintain and manage existing business class schema, managing business class schema could be tedious. Hence, it is difficult to achieve high productivity and effectiveness.

As IDL standard is revised, improved and enhanced over the years, many new constructs and features are added into the standard. One of the major concerns for developers is, as existing technology and standard are revised, it could become difficult to maintain and could be incompatible with the current OSL in the wrapper. In addition, the object wrapper framework needs to examine the existing handling and wrapping capabilities before accepting new features and constructs proposed by the revised version of the IDL standard. Hence, in this research, the framework lays an IDL-OSL version control component in order to validate the version and compatibility of the existing IDL-based business class schemas. The object wrapper supports an IDL-based business class schema to be converted to the proposed OSL class schema file through an IDL-OSL conversion component. This feature allows the object wrapper framework to comply with the standard proposed by ODMG. As a result, the object wrapper framework accepts IDL-based business class schemas. The IDL-OSL conversion component also allows the existing OSL-based business class schemas to have backward compatibility with the older versions of IDL standard by validating with the IDL-OSL version control component. Developers who are not interested to change their existing business class schema do not have to modify their current class schema file. The version number control component keeps track of and maintains a history of the older IDL and OSL standard. Hence, the version number is used to identify different release of IDL and OSL. The IDL-OSL version control and conversion components also allow the object wrapper framework to have its own growth pace. Fig. 1 presents various approaches managed by the proposed object wrapper framework. It shows the process and flow for the generation of the IDL-based class schema file. As IDL-based business class schema is sent to the class schema management facility, the IDL-based class schema is validated by the IDL-OSL version number control component, converted by the conversion component, checked by the OSL syntax checker and finally saved as an OSL-based business class schema file.

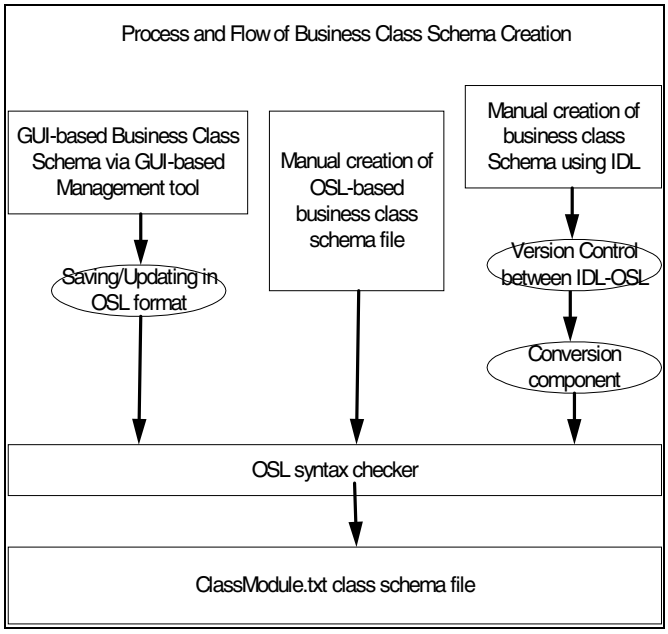

Fig. 1: Various ways for the Process and Flow of Business Class Schema Creation

2.2 GUI-Based Object Schema Management Tool

A graphical business Object Schema Management Tool (OSMT) has the advantage of ease of use and ease of maintenance. In the business application development environment, productivity and effectiveness are the most critical aspects of software development. In this research, a GUI-based tool is considered as the most important factor, which allows a developer to achieve high productivity and effectiveness in business class schema

management. Fig. 2 is the look of the GUI-based OSMT. It allows a developer to load and save a module of the business class schemas. Each module could hold virtually unlimited number of business class schemas. As class schemas are changed, changes could be made through the GUI-based tool and the modified version of the business class schemas will be saved into a designated file. The business class schema file holds class schemas compiled in OSL format. The OSL could be used to regenerate and reconstruct depending on the host OOPL.

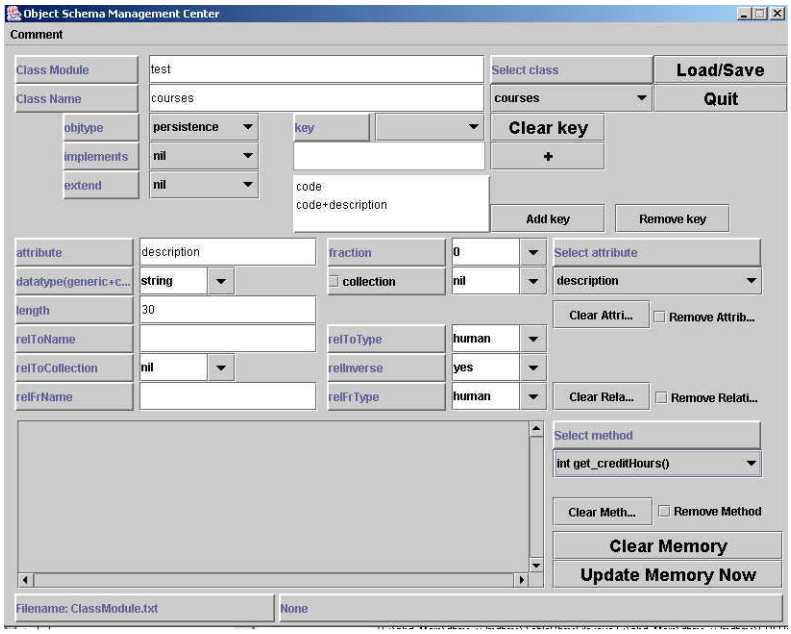

Fig. 2: A GUI-based Object Schema Management Tool

Methods for each class schema are typed and maintained through the GUI tool. The tool does not compile and validate the validity of these methods. As the class schemas are saved, information is saved into ClassModule.txt file in OSL format. Fig. 3 shows an example of class schemas saved in OSL format.

2.3 Object Schema Language

The object schema language (OSL) that is proposed in this research has a simple construct and syntax. The following is an example of a business domain captured by OSL. Fig. 4 shows a transition diagram for the OSL language set that is currently proposed and supported by the object wrapper framework.

An OSL class schema file consists of two parts. The first part consists of a module header. The module header has a 'module:' tag, followed by the module name.

Each class schema starts with a 'classname:' tag. The 'classname:' tag defines a new class in the current module. It is followed by a class name. Four class tags define the class's attributes. The 'objtype:' tag defines either the class is a 'persistence' or 'transient' object. The 'extends:' tag defines the class's superclass. In this design, OSL only supports single inheritance. If the current class does not inherit from any other classes, a 'nil' is used. Any class inherited by this class must be defined first. The 'implements:' tag defines as an interface class that contains only methods stub to be implemented in the current class. This design is not implemented currently. Finally, 'keys:' tag allows developers to define both compound key and simple key. A compound key is a key concatenated with multiple attributes.

The class body is separated into two types. If an attribute is a simple class attribute, complex class attribute or collection class attribute, it starts with an 'attribute:' tag. If it is a relationship attribute, it starts with 'relToName:' tag.

An attribute always starts with an 'attribute:' tag. It is followed by a 'datatype:' tag which defines either string, float or int or a valid class name. If the data type is a class name, such class name must be defined first; otherwise, the OSL syntax checker will return a class missing error. The next tag is 'length:'. A 'length:' tag is always followed by a numerical value that defines the size of this attribute. It is only used if it is a string attribute.

If the current attribute is a collection attribute, a ‘collection:’ tag is always followed after ‘attribute:’ tag.

```
//This is a testing class file
module:test
//
classname:human,objtype:persistence,extends:nil,implements:nil,keys:nil
attribute:ic,datatype:string,length:30
attribute:placesborn,datatype:string,length:20
begin_method:
String get_ic()
{
   return ic;
}
end_method
//
classname:address,objtype:persistence,extends:nil,implements:nil,keys:nil
attribute:num,datatype:int,length:0
attribute:street,datatype:string,length:30
attribute:postcode,datatype:int,length:0
attribute:state,datatype:string,length:10
attribute:country,datatype:string,length:20
begin_method:
String get_num()
{ return num; }
end_method
begin_method:
String get_country()
{ return country; }
end_method
//
classname:student,objtype:persistence,extends:human,implements:nil,keys:nil
attribute:name,datatype:string,length:30
attribute:age,datatype:int,length:0
attribute:address,datatype:address,length:50
relToName:takes,relToType:courses,relToCollection:set,relInverse:yes,relFrName:is_taken_by,relFrType:courses
begin_method:
address get_address() { return address; }
end_method
begin_method:
int get_age() { return age; }
end_method
//
classname:tutor,objtype:persistence,extends:student,implements:nil,keys:nil
attribute:salary,datatype:float,length:10
attribute:subject_code,datatype:String,length:30
relToName:assists,relToType:courses,relToCollection:nil,relInverse:yes,relFrName:has_tutor,relFrType:courses
begin_method:
float get_EPF()
{
   return salary * 0.15;
}
end_method
//
classname:books,objtype:persistence,extends:nil,implements:nil,keys:nil
attribute:title,datatype:string,length:30
attribute:publish_date,datatype:string,length:10
begin_method:
String get_title()
{ return title; }
end_method
//
classname:lecturer,objtype:persistence,extends:human,implements:nil,keys:name
attribute:name,datatype:string,length:30
attribute:qualification,datatype:string,length:30
attribute:specialized_area,datatype:string,length:50
attribute:has,collection:set,datatype:books,length:0
attribute:degrees,collection:set,datatype:string,length:30
begin_method:
String get_name()
{ return name; }
end_method
//
classname:courses,objtype:persistence,extends:nil,implements:nil,keys:code,keys:code+description
attribute:code,datatype:string,length:30
attribute:description,datatype:string,length:30
attribute:credits,datatype:int,length:0
relToName:is_taken_by,relToType:student,relToCollection:set,relInverse:yes,relFrName:takes,relFrType:student
relToName:has_tutor,relToType:tutor,relToCollection:nil,relInverse:yes,relFrName:assists,relFrType:tutor
begin_method:
int get_creditHours()
{ return credits; }
end_method
```

Fig. 3: A sample OSL class schema

The value that a 'collection:' tag takes on is either bag, set or list. In this case, 'datatype:' tag can take on either generic data type such as string, int or float, or a valid class defined earlier. A collection attribute does not use 'length:' tag at all. 'length:' tag will be ignored if it is present.

If the attribute starts with 'relToName:' tag, it is a relationship attribute. It is followed by the name of the relationship. The class that this link points to is defined using 'relToType:' tag. Immediately after 'relToType:' tag and its valid class name, the developer needs to set either a 'nil' or a valid collection type such as set, list and bag at the 'relToCollection:' tag. OSL also allows a relationship to have a two-way traversal capability by specifying 'yes' for the value of the 'relInverse:' tag; otherwise, just specify 'no'. If 'relInverse:' tag has a value 'yes', 'relFrName:' tag must specify the reverse relationship name, and this is followed by the 'relFrType:' tag which specifies the reverse class name. Otherwise, 'relFrName:' and 'relFrType' are ignored if they are present.

As soon as a module is defined, the OSL proposed in this research is saved into a text file. It is designed to be simple and easy to use. The GUI tool described in Section 2.2 allows a developer to manage and maintain business class schemas without learning OSL for productivity and effectiveness. Fig. 4 shows a complete transition diagram for the proposed OSL language implemented in this research.

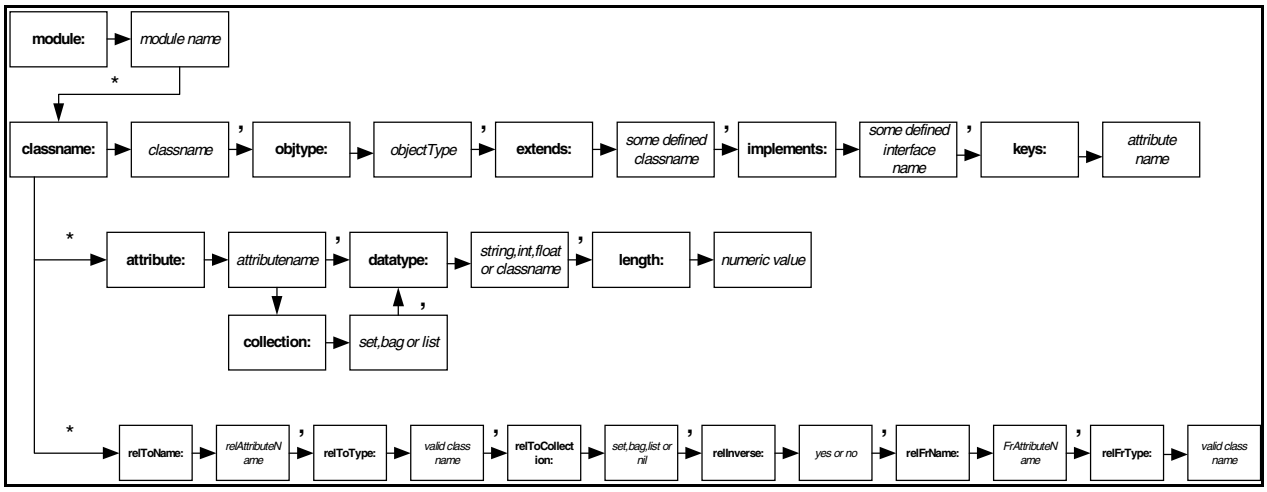

Fig. 4: OSL transition diagram

3.0 ENHANCED FEATURES IN OSMT COMPONENT

The entire OSMF component proposed in this project is shown in Fig. 1. Developers have the flexibility to define and manage business class schemas through the GUI tool, IDL or OSL. In Fig. 5, the full life cycle of the business class schemas is shown.

The class schema stored in ClassModule.txt file is generated through a class schema generation engine (indicated as a Generate Class box in Fig. 5). The OSMT component in the OSMF allows the developer to set the type of the host OOPL in which these business classes will be generated. The OSMF component design has been proposed to support three OOPLs currently. They are Java, C++ and Smalltalk. OSMF generates the business classes into a Java file if it is set to generate into Java OOPL; namely ClassSchema.Java. During the generation process, each class schema in the ClassSchema.Java file will automatically be appended with set_ and get_ methods for each attribute in the class.

Methods defined through the GUI tool will be generated into the ClassSchema.Java file, also during the Generate Class process. The OSMT component does not check for the host OOPL's syntax. The content of the ClassSchema.Java is compiled through the host OOPL compiler to determine whether it is syntactically and semantically correct.

As objects are read from the persistence stores to the memory, these objects are bound to objects of their respective classes through the predefined set_ methods in each class schema.

Whenever a business application is compiled and executed, ClassSchema.Java will be compiled and active in the main memory. The content of the ClassModule.txt file will also be loaded into the main memory. They will be active in the main memory throughout the entire life span of the application. A classTree instance is generated to work with ClassModule, ClassSchema, ClassSchemaDetail, TableDbms, TableSchema and TableSchemaDetail objects. These objects will work hand-in-hand with the class schemas in ClassSchema.Java in order to load and unload persistence objects from and to the underlying DBMS.

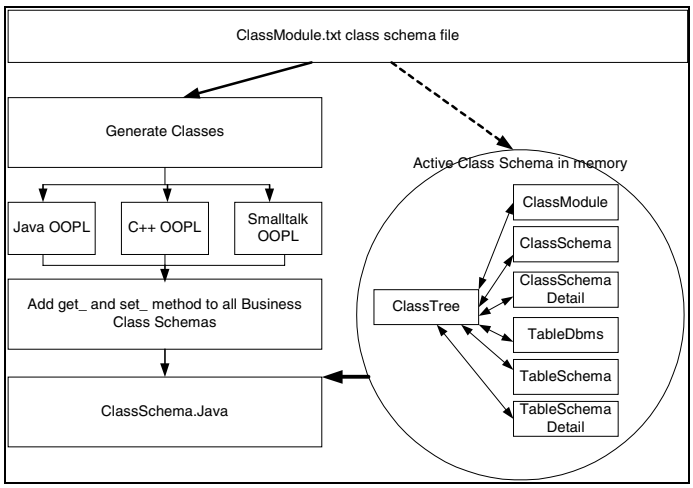

Fig. 5: Flow of class schemas' life cycle for a business domain in the proposed object wrapper framework

4.0 PROS AND CONS OF THE OSMT COMPONENT AND POSSIBLE ENHANCEMENTS

The proposed OSMF component in the object wrapper intends to handle both the IDL standard proposed by ODMG and to provide high productivity through the GUI tool with OSL in order to maintain its framework development pace for backward compatibility. Hence, the resulting component in the framework is tedious to construct and develop. Nevertheless, it has the following advantages:

a. Comply and understand class schemas created using IDL
b. Ease of use through GUI-based class schema management tool
c. Ease of regenerating class schema to multiple OOPLs
d. Allow backward compatibility with the older IDL standard through OSL using version control
e. Allow the existing class schema to work through the OSL layer even though IDL standard is revised and enhanced.

A few shortcomings are discovered during the implementation of the proposed OSMF component described above. It is difficult to construct and develop for multiple languages platform. As IDL is revised, OSL has to be revised in order to comply with the latest revised standard. In order to maintain multiple standard releases, large specification files are required to keep with the older standard specification in order for the existing class schemas that have limited features to continue functioning.

The OSMT component in OSMF has a few pros and cons as well. The pros of OSMT is that GUI user interface is more intuitive to work with and easy to use with very little learning curve. The disadvantage of OSMT is the complexity of development and coding.

OSMT is a GUI-based tool that helps users to maintain business classes easily with very little learning curve. Nevertheless, in order to allow dynamic business requirements change after a business class domain has been defined and implemented in the real world, process of migrating the existing persistence objects to the newly defined business classes are no easy task at all. The tool is now going through another round of designing and coding in order to cater for object migrating functionality as business classes are modified. Better OSMT will be produced in the coming version.

REFERENCES

[1] Sai Peck, Lee and Tong Ming, Lim, "Classes and Objects Management Multidimensional DBMS Data Model", in *IASTED International Conference Software Engineering (SE '97)*, San Francisco, California, 2-6 Nov. 1997.

[2] Sai Peck, Lee and Tong Ming, Lim, "Objects Collection Management in Multidimensional DBMS Data Model", in *ISORC Kyoto International Conference Hall*, Kyoto, Japan, April 20-22, 1998.

[3] Sai Peck, Lee and Tong Ming, Lim, "Objects to Multidimensional Database Wrapping Mechanism", in *Annual AoM/IaoM International Conference on Computer Science*, Westgate Hotel, San Diego, California, August 6-8, 1999.

[4] Sai Peck, Lee and Tong Ming, Lim, "An Object to R/MDBMS Persistence Framework", in *REDECS2001 conferences*, UNITEN-UPM, Kuala Lumpur, Oct. 22-25, 2001.

[5] Sai Peck, Lee and Tong Ming, Lim, "Object Schema Management Facilities in an Object Wrapper", in *SERP02 Las Vegas Multi Conferences*, 23 – 27 June 2002.

[6] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994.

[7] Elisa Bertino and Lorenzo Martino. *Object-Oriented Database Systems*. Addison-Wesley, 1998.

[8] Georg Lausen and Gottfried Vossen. *Models and Languages of Object-Oriented databases*, Addison-Wesley, 1998.

[9] R. G. G. Cattell and Douglas Barry. *The Object Data Standard: 3.0*, Morgan Kaufmann Publishers, 1999.

BIOGRAPHY

Tong Ming Lim is a PhD candidate at the Faculty of Computer Science and Information Technology, University of Malaya. He is working on the object-oriented wrapper in his final year. He worked in TAR College, as a GM and IT Director in two commercial software houses for about 10 years. He is currently lecturing in Monash University, Malaysia, on computer science courses such as database management system, artificial intelligence and computer graphic programming. He is an ACM member and IEEE member since 1993. He could be reached by lim.tong.ming@busit.monash.edu.my.

Sai Peck Lee is currently an associate professor at Faculty of Computer Science & Information Technology, University of Malaya. She obtained her Master of Computer Science from University of Malaya in August 1990, her Diplôme d'Études Approfondies (D. E. A.) in Computer Science from University of Pierre et Marie Curie (Paris VI) in July 1991 and her Ph.D. degree in Computer Science from University of Panthéon-Sorbonne (Paris I) in July 1994. Her current research interests include Software Engineering, Object-Oriented (OO) Methodology, Software Reuse and Framework-based Development, Information Systems and Database Engineering, OO Analysis and Design for E-Commerce Applications and Auction Protocols. She has published a number of research papers in several computer science journals as well as in local and international conferences. She is a member of IEEE Computer Society.