Proceedings of International Conference on Distributed
Systems, Software Engineering and Database Systems

A User Interface for an ODMG-93 Compliant Database

Liu Yi-Huang, Hong Ting-Yue, Hou Kai-Liang, Daniel J. Buehrer
Institute of Computer Science and Information Engr.
National Chung Cheng University

Abstract

This paper describes a User Interface that is based on a model for persistent collections and their operations as implemented in an ODMG-93 [1] compliant database. Our User Interface is implemented using a Java applet [2] on the client side. This client generates ODMG-93 based Object Definition Language and query code that is sent over the network to an ODMG-93 compliant database server. Besides, this user interface also provides multimedia functions for displaying image files and playing music files, and visiting WWW Sites via their URLs.*

* Java is a registered trademark of
Sun Microsystems, Inc.

Keywords: ODMG93, ODL, OQL,
CORBA, IDL

1. Introduction

In this paper we present a user interface that is being implemented using a Java applet. Java™ is a programming language for writing client and server applications. When comparing Java-based interfaces with others [3-7], one must remember that Java-based user interfaces can run on almost any machines, O.S, and terminal type.

One of the major advantages of Java is that these applets can be downloaded on demand, so that users do not have to be concerned with installing various applications such as database systems. This makes it easy to develop a sequence of database implementations, each adding more capabilities, such as security and backup/recovery. The most recent version of the database is then loaded automatically into the client's machine on demand, depending on what kind of database the user connects to. Another major advantage of Java is that it provides some powerful and useful libraries such as the Abstract Window Toolkit, Applets, etc. These libraries make it easy for the user to design and implement the user interface.

Java is a particularly useful language for writing database queries of arbitrary complexity, since the queries are guaranteed not to affect the client's machine. Java is sufficiently robust to permit the user to write arbitrarily-complex database applications, including windows' operations to display the data in a user-friendly format, but since the extended Java query language interpreter is an applet, it is guaranteed not to affect other users of the client machine. The database system on which our user interface is based is a prototype making use of a Java applet to generate Object Database Manangement Group's (ODMG-93) Object Definition Language (ODL). It retrieves class information and sends queries by using query functions, receiving objects, image files, audio files or URLs through the network from a compliant

ODMG-93 database server. The user interface consists of six important windows: main, image, music, WWW, ODL and query. The ODL window can define new classes for the database. The query window is capable of retrieving, storing and updating persistent data in an object-oriented format.

The remainder of this paper is organized as follows. In Section 2, we describe the advantages of the object-oriented query model over the relational query model. In Section 3 we describe further details of the user interface and give an example. In Section 4, a summary and conclusions are given.

2. Object-oriented queries

2.1 Object Definition Language

The Object Definition Language defines the characteristics of ODL types (which we call "classes"), including their properties and operations. The ODMG framework describes objects with ODL, an extension of the CORBA IDL object typing language. The syntax of ODL is as follows:

interface new_type_name (:
inheritance_list)
(**extent** extent_name
key(s) key_list)
{ **attribute** domain_type
attribute_name
relationship target_path **inverse**
traversal_path
};

The "extent", "key(s)", and "relationship" keywords are ODMG-93's ODL extensions of OMG's IDL. They have been added to IDL to permit the description of concepts which are often used in database type declarations.

2.2 Object-oriented queries vs. relational queries

As mentioned, ODL is an extension of IDL, the interface description language of CORBA. IDL is used to describe object-oriented data which is to be transmitted via remote procedure calls. The major problem of such CORBA-like systems is their inability to transmit pointers via such remote procedure calls. Such pointers are often only meaningful for the caller, and they are usually not useful to the callee unless he has some way to dereference them. One of the major problems is that object-oriented databases are usually full of such pointers. These are the relations of ODMG-93 databases, and these relations represent reference joins. In relational databases, such join conditions must be specified for each query, since relational databases contain no such pointers, and primary keys must be used to achieve the effect of pointers.

So, for instance, in an object-oriented database, one immediately knows which students are advised by which professors, which are teaching which courses, etc. simply by using the dot notation [8] to follow binary relationships. ODL (of ODMG-93) extends IDL to include such binary relationships, and it permits n-ary relationships to be viewed as either sets, bags, arrays, or lists. IDL only offers arrays and sequences, and it does not specify how collections of complex objects can represent binary relationships. Moreover, it does not offer ODL's inverse relations, which permit the binary relations to be viewed as collections from the viewpoint of either side of the relation. Insertions/deletions to an ODMG-93 relation also affect the corresponding

inverse relation. Likewise, modifications to an inverse relation also affect the original relation.

Asking an object-oriented query is equivalent to selecting elements from one or more of these binary relations by using reference joins along with SELECT conditions. In our model, each such object-oriented query is given a name, and it is assigned as a new binary relationship from the current object(s) to the objects which have been selected.

Since object-oriented queries use dot notation to express the reference joins, they are usually quite a bit simpler than the corresponding SQL query. Moreover, although the pointers in object-oriented graphs are object identifiers, when the selected subgraphs are passed to the client for display, the object identifiers can be converted into subscripts of the corresponding collections, so the object identifiers do not need to be displayed directly. The user-interface can use these subscripts to do the implicit joins on demand, selecting the subgraph which is connected to the items that have been chosen by the user, by either using a query or the mouse. If the client's memory is insufficient, then caching will have to be handled by the database server, with the client requesting subcollections of the given collections, as is currently done by ODMG for relational databases. Fortunately, unlike ODBC relational databases, the object-oriented database servers can simply write the collections to temporary files which can be read by the client side, and they do not have to be further bothered by the client.

3 The user interface

3.1 An overview

In our user interface, there are six important windows including the main, ODL, query, image, music, and WWW windows. A simple sketch of the user interface model is shown in Figure 3.1. In this section, we will describe the ODL window and the Query window in more detail, since they are the most important.

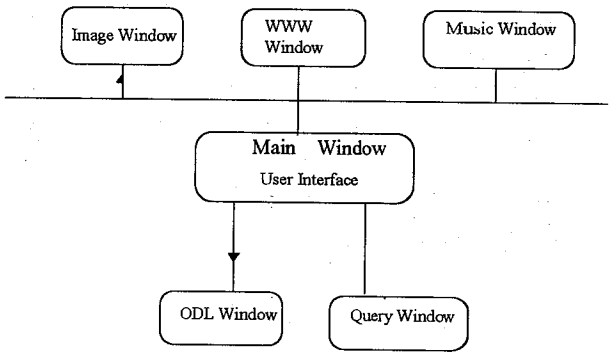

Figure 3.1 User Interface model design

The ODL window (see Figure 3.2) is used to permit the user to define a new class for the database server. The user can create a class either by keying in using a keyboard or by interactively using the mouse. The design model is shown in Figure 3.2. The ODL window displays the previous class definitions as well as the new class being defined.

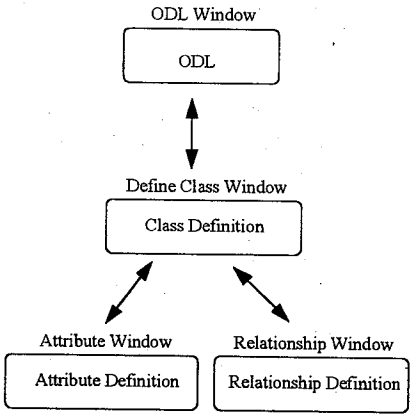

Figure 3.2 ODL window model design

The Query window (see Figure 3.3) permits the user to ask a query to the database server. If the query is a name query, then the user interface pops up the Class window to let the user choose the desired class or its members. If the query is a query constraint, then the Select Class window pops up to let the user select the classes to be used in the Query Constraints window. Whenever an error occurs, the message window pops up automatically. These windows are shown in Figure 3.3.

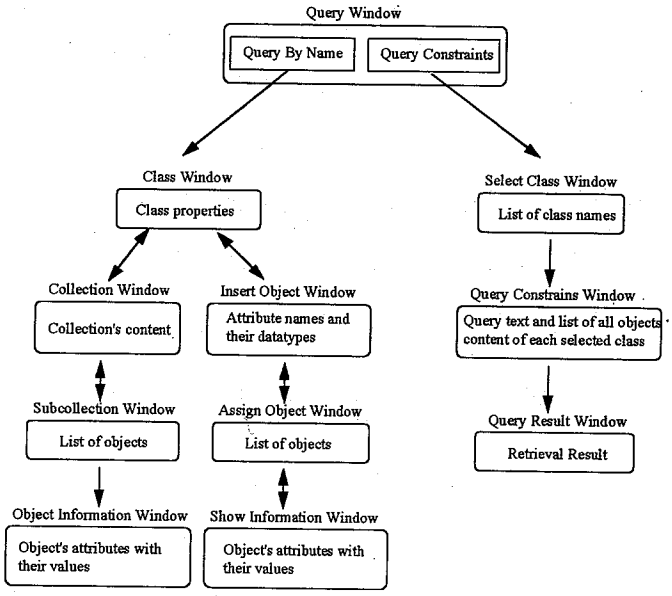

Figure 3.3 Query window model design

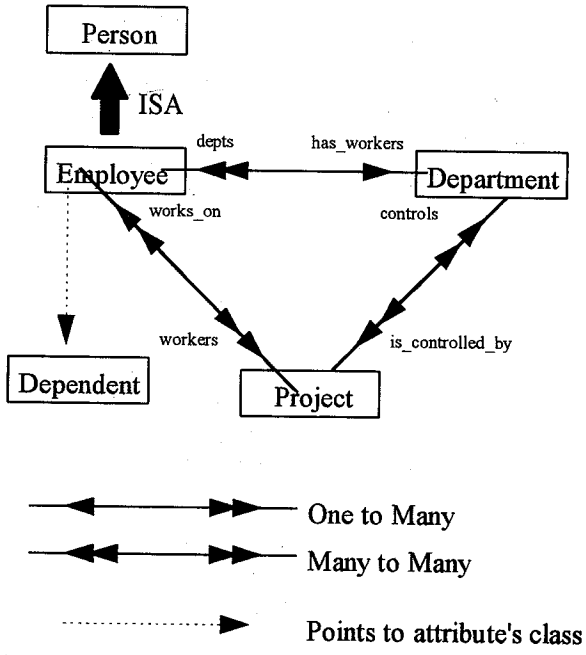

Figure 3.4 Schema Graph

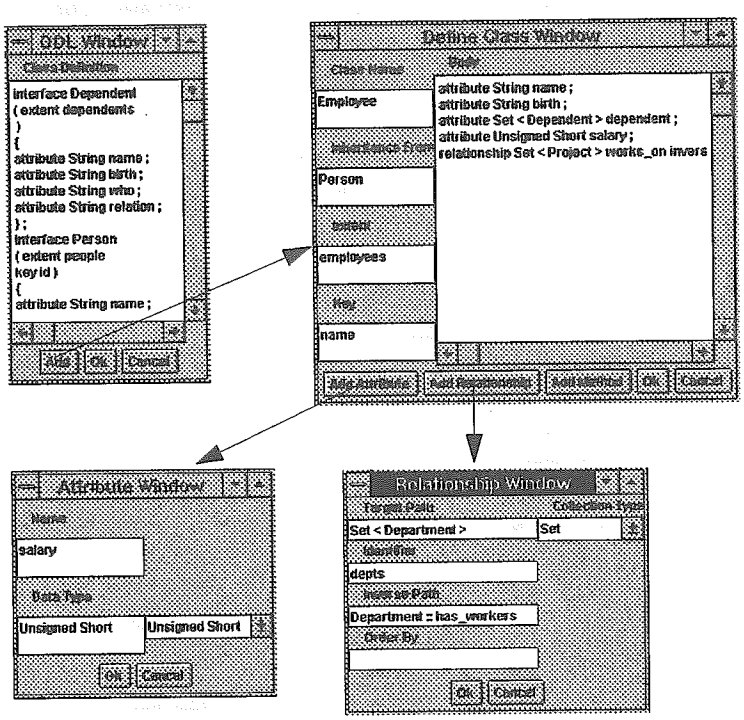

Figure 3.5 Define classes

3.2 Example

A. ODL

When the user wants to define classes, he can first click the "ODL" button from the Main window. The user interface will pop up an ODL window. Consider the schema graph shown in Figure 3.4. The sequence of steps to define classes is shown in Figure 3.5. The ODL window contains the class definitions that the user has already input but has not yet sent to the database.

B. Query

There are two kinds of query provided by the database, including query by name and by query constraints.

B.1 Query By Name

If the user wants to view the collection of a class, he should first click on the "Query By Name" button from the Query window. The user interface will pop up the Name Query window to permit the user to input either the class name or the class name followed by '::' and one or more attribute or relation names separated by periods. If the user inputs the class name "Employee", after committing the class name to the database, the user interface will receive this class's information. If the user inputs "Employee::dependent", then the Collection window will receive the information for the union of all dependents of any employees. There are also other functions provided in the Collection window for adding attributes, adding relationships, renaming the class, renaming the extent, renaming properties, and inserting objects.

The user can insert an object by

clicking the Insert Object button in the class window. The user interface will pop up the Insert Object window (Figure 3.6). Because the class "Employee" has a superclass "Person", the superclass attributes are also shown in the Insert Object window to permit the user assign values. The "dependent" attribute's datatype is a set of objects. The user can assign its value by clicking on its datatype and using the Assign Object window to choose dependents to be added. If he double clicks on the "*10" object, then this object's contents are shown in the Show Information window. If not all the objects are defined, the user can fill in the attributes values and click the "New" button to add a new object to class "Dependent".

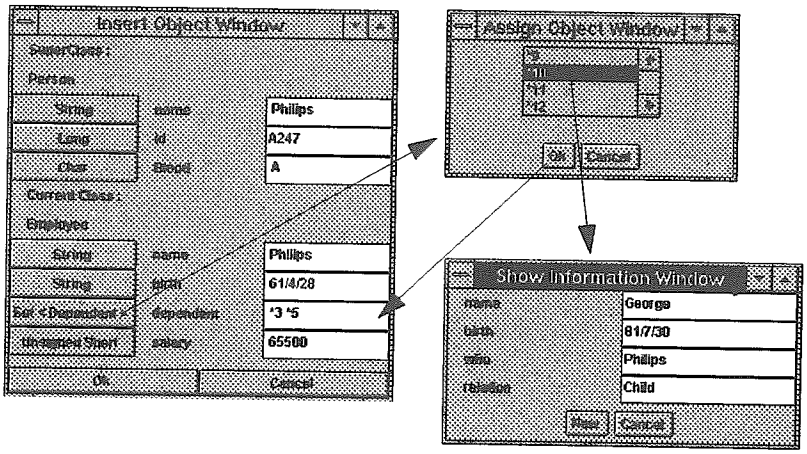

Figure 3.6 Inserting an Object

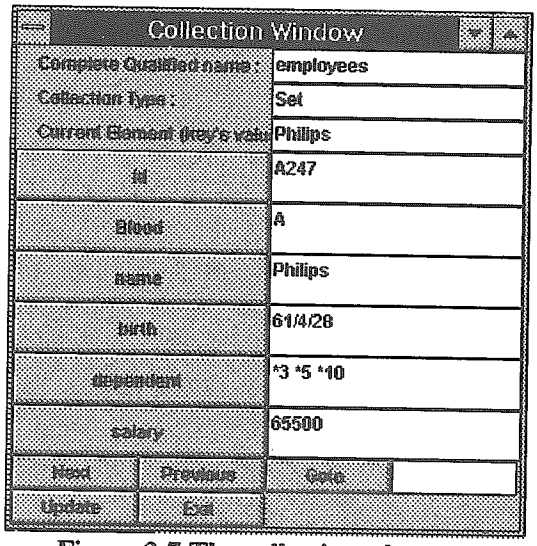

Figure 3.7 The collection of a class

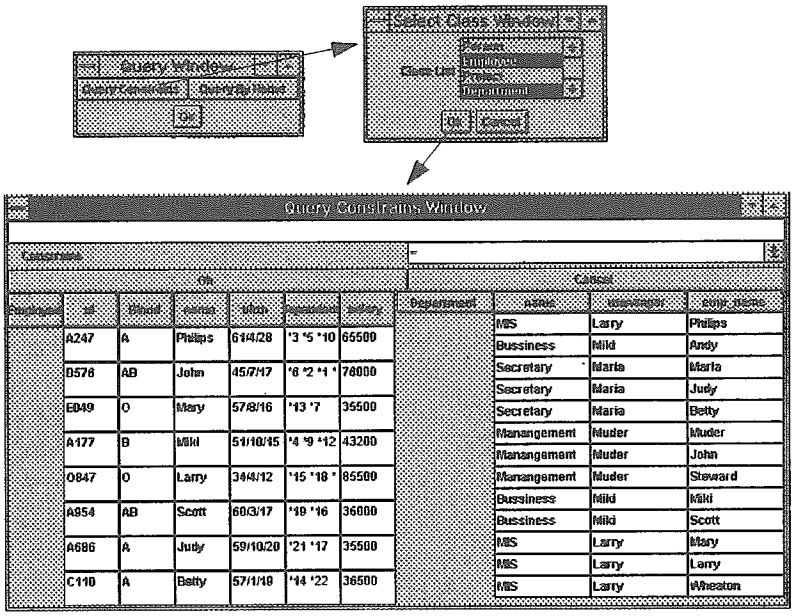

Figure 3.8 Pop up the Query Constraints window

When the user wants to view or update the class's collection, he can click on the "extent" button from the Class window. The user interface will pop up the Collection window which contains the collection of all members of the class. In the Figure 3.7, the Collection window shows the class's extent name, the collection type, the current element's key value, and the current element's content. Besides, it permits the user to view the next or previous object's contents. Also it permits the user to view the particular object by filling in a number in the goto field. The user can update this object's content after modifying the attribute's value.

B.2 Query constraints

If the user wants to ask a query, he should first click on the "Query Constraints" button from the Query window. The user interface will pop up the Select Class window to permit the user to select the classes that will be used in the Query Constraints window (Figure 3.8). If the user double clicks the class "Employee and the class "Department" in the Select Class window, then all the extents' contents of class "Employee" and class "Department" are shown in the form of a grid in the Query Constraints window.

When the user wants to make a query, he can input either by using a keyboard or interactively by using a mouse. For example, the user can first click the attribute "name" in the class "Employee", and then choose a constraint "=" from the choice of constraints (=, >, >=, <, <=), and then click the attribute "emp_name" in the class "Department". Another query constraint can be achieved by clicking the attribute "salary" in the class "Employee", and then choosing a constraint ">=" from the choice of constraints and then using a keyboard to input 50000. After pushing the "Ok" button to commit the query, the Query Result window is popped up. This contains the objects which satisfy the query.

4. Conclusions

In this paper, a user interface for an ODMG-93 Compliant Database was described. The user interface is a Java applet. This database system is a prototype system that demonstrates a simple, easy, and user-friendly user interface. The user can create/retrieve data to/from a database by means of the user interface, which then creates ODL/query commands that are sent through the network to the database server. In addition, the user interface also provides the multimedia and WWW facilities that enable the user to view images, listen to music, and visit WWW sites via their URLs.

One advantage of using a Java applet for the database front-end is that database applications can be inserted as links into other WWW hypermedia. Another major advantage is that the newest version of the database applet will automatically be downloaded to the user's machine, making it possible to create a sequence of database implementations with more and more features. The traditional burden of supporting multiple versions on multiple platforms is eliminated.

Our future work involves adding more capabilities such as security and performance improvements. The Java applet improves security since database impostors are theoretically unable to foul up the client's machine. However, a way must still be found to prevent database impostors from being able to act like a Trojan Horse, and request the user's password, thus gaining illegal access to the database server. Another future improvement involves using class algebra's sorted disjunctive normal forms[9] to assign heirarchy to each query result.

References

[1]Cattell, R.G.G., ed. *The Object Database Standard: ODMG-93*, Morgan Kaufmann Publ, 1996.

[2] Java Home Page. http://java.sun.com/

[3] Buschek, J.J., "COMPAS Windows: A Point-and-Click Interface to Oracle", *in Proc. of IEEE Conf. on Oceans Engineering for Today's Technology and Tomorrow's Preservation*, 1994, vol.1, pp.I/15-20.

[4] Kao, C.H.; Evens, M., "A Windows User Interface to Data Retrieval and Report Generation for a Diabetic Patient Database", *Proc. Seventh Symposium on Computer-Based Medical Systems*, IEEE Comput. Soc. Press, 1994, pp.98-103.

[5] Lam, H.; Chen, H.M.; Ty,F.S.; Qiu, J.; Su, S.Y.W., "A Graphical Interface for an Object-Oriented Query Language", *Proc. Fourteenth Annual International Computer Software and Applications Conference*, IEEE Comput. Soc. Press, 1990, pp.231-237.

[6] Powersoft Corp., *User's Guide*, 1991,1994.

[7] Zloof M., Query By Example, IBM Systems Journal, vol.16, no.4, 1977.

[8]Tsukamoto, M.; Nishio, S.; Fujio, M., "DOT: A Term Representation Using DOT Algebra for Knowledge-Bases", *Deductive and Object-Oriented Databases, Second International Conference, DOOD'91 Proceedings*, 1991, pp.391-410.

[9] Buehrer, D.J., "Class Algebra as a Description Logic", *Description Logic '96 Workshop Proceedings*, Boston, Nov. 2-4, 1996, Available at http://www.cs.ccu.edu.tw/~dan/buehrer.ps .