E1-20: An object oriended layer on top of Oracle*

L D Alahakoon**
(*Dept. of Computer Science and Statistics, Univ. of Colombo, Colombo 3,*
***Present address: Data Processing Department, Colombo Stock Exchange, Colombo 1*)

The design of an object oriented interface to the Oracle RDBMS is described. The interface consists of a command language with a few essential and basic commands. It can access Oracle only through SQL and does not access Oracle Forms or Report writer utilities. This can be considered as a base for further work on developing an object oriented layer on top of a relational database.

The interface that was designed consists of a small command language, called OBJECT*SQL. Classes, instances and methods (on Classes), can be defined and deleted (dropped), and methods can be executed on existing classes, by using this command language.

This command language, the data structure and the algorithms for the manipulation of the data structures have been programmed using Microsoft C. This set of programs is considered as Layer 1 of the Interface, which actually implements and controls the object oriented concepts. The set of Embedded SQL and Dynamic SQL statements which connect Layer 1 to The ORACLE DBMS is considered as Layer 2.

In developing this interface the following limitations were assumed due to the time constraint:

(a) There are no complex objects;
(b) Only single inheritance is possible;
(c) Methods will only consist of valid SQL statements.

ORACLE being a Relational Data Base Management System, the actual data has to be stored in relational tables. Therefore a mapping scheme was designed from the object oriented schema to the relational schema.

The implementation of this mapping scheme was done by using 4 data structures. All these data structures used the linked method of implementation as they were required to be dynamic so that classes, methods and instances could be defined or dropped by the user at any time.

The most important of these data structures was the binary tree structure, which was used to implement the concept of "Class hierarchy". At the beginning, this tree, has only the root node. Other child nodes are added when new classes are defined.

The structure number 2 is to store the method names that are defined on each class. This is implemented with a Linked List structure. Whenever a method is defined its name will be linked to the corresponding class name in this structure, where the class names are attached automatically at the definition of the classes.

There is also a structure which will store the attributes-names and key-name of each class. This structure becomes useful when the key-name of a class is required for it to be attached to its superclass.

There is another Linked List structure which will store each method defined (SQL statement) as a character string. When a method has to be run (or executed) on a class, this method will be extracted from this method store structure, and passed on to Oracle to be executed as a Dynamic SQL statement.

When a work session is over all the data in these data structures would be saved by converting into character streams and stored, and at the start all the data thus stored would be re-converted to the respective structures to be available for use.

The user has to access the database through a command language (OBJECT*SQL) that has been developed. Whenever a command was used the entered data would be manipulated through the 4 structures by a set of algorithms, so that they would be arranged according to the object oriented concepts programmed into these algorithms and structures. Error messages would also be sent during this phase.

All these data structures and algorithms used to manipulate data in the structures were programmed in Microsoft C. The final output of these C programs are taken as a character string which is converted to a VARCHAR variable in Embedded SQL, and executed using the Dynamic SQL facility in ORACLE.

Although some limitations exist (no multiple inheritance and complex objects) the model supports class hierarchy, single inheritance, and simple objects. Therefore this model supports the basic object oriented concepts. The limitations were imposed with the idea of keeping to a time plan of one year. But the programs can very easily be extended to include these facilities.

The program language consists of only 10 commands, and therefore several important tasks (such as edit class) cannot be done using this language. The commands programmed are the essential ones required for the user, but other commands could be added to the system. *Presented at the SEARC '95 Conference. This work won the SLAAS Computer Award, 1995.