Implementation Aspects of an Object-Oriented DBMS

Asuman Dogac Mehmet Altinel Cetin Ozkan
Ilker Durusoy
Software Research and Development Center
Scientific and Technical Research Council of Turkiye
Middle East Technical University (METU)
06531 Ankara Turkiye
email: asuman@srdc.metu.edu.tr

Extended Abstract

1 Introduction

This paper describes the design and implementation of an OODBMS, namely the METU Object-Oriented DBMS (MOOD). MOOD [Dog 94b] is developed on the Exodus Storage Manager (ESM) [ESM 92] and therefore some of the kernel functions like storage management, concurrency control, backup and recovery of data were readily available through ESM. In addition ESM has a client-server architecture and each MOOD process is a client application in ESM. The kernel functions provided by MOOD are the optimization and interpretation of SQL statements, dynamic linking of functions, and catalog management. SQL statements are interpreted whereas functions (which have been previously compiled with C++) within SQL statements are dynamically linked and executed. A query optimizer is implemented by using the Volcano Query Optimizer Generator. A graphical user interface, namely MoodView [Arp 93a, Arp 93b], is developed using Motif. MoodView displays both the schema information and the query results graphically. Additionally it is possible to update the database schema and to traverse the references in query results graphically.

The system is coded in GNU C++ on Sun Sparc 2 workstations. MOOD has a SQL-like object-oriented query language, namely MOODSQL [Ozk 93b, Dog 94c]. MOOD type system is derived from C++, thus eliminating the impedance mismatch between MOOD and C++. The users can also access the MOOD Kernel from their application programs written in C++. For this purpose MOOD Kernel defines a class named UserRequest that contains a method for the execution of MOODSQL statements. The MOOD source code is available both for anonymous ftp users from ftp.cs.wisc.edu and for the WWW users from the site http://www.srdc.metu.edu.tr along with its related documents.

In MOOD, each object is given a unique Object Identifier (OID) at object creation time by the ESM which is the disk start address of the object returned by the ESM. Object encapsulation is considered in two parts, method encapsulation and attribute encapsulation. These encapsulation properties are similar to the public and private declarations of C++.

Methods can be defined in C++ by users to manipulate user defined classes and after compilation, they are dynamically linked and executed during the interpretation of SQL statements. This late binding facility is essential since database environments enforce run-time modification of schema and objects. With our approach, the interpretation of functions are avoided thus increasing the efficiency of the system. Dynamic linking primitives are implemented by the use of the shared object facility of SunOS [Sun 90]. Overloading is realized by making use of the signature concept of C++.

2 MOOD Kernel Design Considerations and an Overview

Objects are grouped in the abstraction level of a class, in other words, classes have extensions. Class extensions are implemented as ESM files. A class in the system has an unique type identifier which is inherited from a meta class named MoodsRoot. This type identifier is used in accessing the catalog to obtain the type information to be used in interpreting the ESM storage objects which are untyped arrays of bytes. The relation between classes and instances is a 1:n relation, i.e., under a class there could be any number of instances associated with it, but an instance can not be associated with more than one class. Class inheritance mechanism of the MOOD is multiple inheritance. The name resolution is handled as in standard C++. Additionally in our system in case of name conflicts, if the scope resolution operator is not