OBJECT ORIENTED DATABASE
MANAGEMENT SYSTEMS

DEPARTMENT OF INFORMATION TECHNOLOGY
SIR C. R. REDDY COLLEGE OF ENGINEERING
ELURU-534007

BY
M. MALLIKARJUNA RAO
III/IV B.Tech (I.T)
651255027

K. VIJAY KUMAR
III/IV B.Tech (I.T)
651255044

Email address:

1. muttineni.mallikharjun@gmail.com
2. ksbs_itjbs@yahoo.com
INDEX

1. INTRODUCTION

  1.1 DEFINITION

2. OODBMS PERSPECTIVES

3. POINTER SWIZZLING TECHNIQUES

  3.1 CLASSIFICATION OF POINTER SWIZZLING

    A. COPY VERSES IN-PLACE SWIZZLING
    B. EAGER VERSES LAZY SWIZZLING
    C. DIRECT VERSES INDIRECT SWIZZLING

4. OODBMS Architecture

  4.1 client server

5. OODBMS Manifesto

6. BENCHMARKING

  6.1 OO1 benchmark
  6.2 OO7 benchmark

7. ADVANTAGES AND DISADVANTAGES

8. CONCLUSION
ABSTRACT

OODBMS has emerged as an important computational field, which is relatively to relational DBMS. OODBMS is generally used in a Multi–user client/server environment. It controls the concurrent access to objects. In this technical paper we presented the OODBMS. First review the problem which we define as it is a software design to assist maintaining and utilizing large collections of data. In this next we present OODBMS perspectives, which are the features Supported by OODBMS. Next we present pointer swizzling techniques and their classification. Next we present OODBMS architecture In that we describe client-server architecture, Modern DBMSs are Follow this type of architecture. Next we discuss the OODBMS manifesto it is mandatory features. Finally we conclude by giving some Advantages and disadvantages of OODBMSs.
1. INTRODUCTION:

Zdonik and Maier (1990) present a threshold model that an OODBMS must,

At a minimum satisfy,

a) It must provide database functionality;

b) It must support object identity;

c) It must provide encapsulation;

d) It must support objects with complex state.

An OODBMS as:

Object-orientation = abstract data types + inheritance + object identity;

OODBMS = object-orientation + database capabilities

OODBMS is used in traditional business applications require data to be stored in the form of rows and columns. Instead of data, objects must be stored as an object, and not as a group of rows and columns.

The OODBMS is started in the engineering and design domains, and also financial and telecommunications applications. Although the OODBMS market is still small, with a 3% share of the overall database market in 1997, the OODBMS continues to find new application areas, such as The World Wide Web. DBMS are primarily concerned with the creation and maintenance of large, long-lived collections of data.
1.1 DEFINITION:

Data base management system (DBMS) is software designed to assist in maintaining and utilizing large collection of data and their relationships.

An object oriented data base management system (OODBMS) is a manager of an object oriented database (OODB). An OODB is a persistent and sharable repository of objects defined in an object-oriented data model (OODM). An OODM is a data model that captures the semantics of objects supported in object oriented programming. There is no universally agreed OODM.

2. OODBMS PERSPECTIVES:

DBMSs are primarily concerned with the creation and maintenance of large, long-lived collection of data. Modern DBMSs are characterized by their support of the following features.

- **Data model:** A particular way of describing data, relationship between data, and constraints on the data.

- **Data persistence:** The ability for data to outlive the execution of a program and possibility the lifetime of the program itself.

- **Data sharing:** The ability for multiple applications (or instances of the
-e same one) to access common data, possibly at the same time.

- **Reliability:** The assurance that the data in the database is Protected from hardware and software failures.

- **Scalability:** The ability to operate on large amounts of data in simple ways.

- **Security and integrity:** The protection of the data against Unauthorized access, and the assurance that the data conforms to specified correctness and consistency rules.

- **Distribution:** The ability to physically distribute a logically Interrelated collection of shared data over a computer network, Preferably making the distribution transparent to the user.

### 3. POINTER SWIZZLING TECHNIQUES:

- **Pointer swizzling:** The action of converting object identifiers to main Memory pointers, and back again.

- **No swizzling:** In this technique objects are faulted into memory by The underlying object manager, and a handle is passed back to Application containing the object’s OID. The object identifier (OID) is Used every time the object is accessed.

- **Object referencing:** In this technique object references are
Immediately converted to virtual memory pointers when the object is Faulted into memory.

- **Hardware-based schemes:** Hardware-based swilling uses virtual Memory access protection violations to detect accesses of nonresident Objects. These schemes use the standard virtual memory hardware to Trigger the transfer of persistent data from disk to main memory. Once a page has been faulted in, objects are accessed on that page via normal virtual memory pointers and no further object residency checking is required. The hardware approach has been used in several commercial and research systems including Object store and Texas.

### 3.1 Classification of Pointer Swizzling:

Pointer swizzling techniques can be classified according to the following three

A) **Copy versus in-place swizzling:**

Copy swizzling may be more efficient as, in the worst case, only modified objects have to be swizzled back to their OIDs, whereas in-place technique may have to unswizzle an entire page of object if one object on the page is modified. On the other hand, copy approach, every object must be explicitly copied into the local object cache

B) **Eager versus lazy swizzling:**
Eager swizzling as the swizzling of all OIDs for persistent objects on all data pages used by the application before any object can be accessed.

Lazy swizzling swizzles pointers only as they are accessed or discovered. Lazy swizzling involves less overhead when an object is faulted into memory, but it does mean that two different types of pointer must be handled for every object access: a swizzled pointer and unswizzled pointer.

C) Direct versus indirect swizzling:

With direct swizzling, the virtual memory pointer of the referenced object is placed directly in the swizzled pointer; With indirect swizzling, the virtual memory pointer is placed in an intermediate object, which acts as a placeholder for the actual object. Thus, with the indirect scheme objects can be uncached without requiring the swizzled pointers that reference the object to be unswizzled also.

4. OODBMS Architecture:

4.1 Client-server:

Many commercial OODBMSs are based on client-server architecture to provide data to users, applications, and tools in distributed environment. There are types of client-server DBMS that vary in functionality assigned to each component.
A.) **Object server:**

This approach attempts to distribute the processing between the two Components. Typically, the server process is responsible for managing Storage, locks, commits to Secondary storage, logging and recovery. Enforcing security and integrity, query Optimization, and executing Stored procedures. The client is responsible for transaction managem -ent and interfacing to the programming language. This is the best arc -hitecture for cooperative, object-to-object processing in an open, distributed environment.

B.) **Page server:**

In this approach, most of the database processing is performed by the client. The server is responsible for secondary storage and providing pages at the client’s request.

C) **Database server:**

In this approach, most of the database processing is performed by the server. The client simply passes requests to the server, receives results, and passes them on to the application. This is the approach taken by many relational data base management systems (RDBMSs).
5. OODBMS Manifesto:

The object-oriented Database system Manifesto proposes thirteen mandatory features for an OODBMS, based on two criteria: it should be an object-oriented system and it should be a DBMS. In this first eight rules apply to the object Oriented characteristics.

1.) Complex objects must be supported.

2.) Object identity must be supported.

3.) Encapsulation must be supported.

4.) Type’s classes must be supported.
5.) Types or classes must be able to Inherit from their ancestors.

6.) Dynamic binding must be supported.

7.) The data manipulation language (DML) must computationally Complete.

8.) The set of data types must be extensible.

9.) Data persistence must be supported.

10.) The DBMS must be capable of managing very large databases.

11.) The DBMS must concurrent users.

12.) The DBMS must be capable of recovery from hardware and software failures.

13.) The DBMS must be providing a simple way of querying data.

6. Benchmarking:

Benchmarking is a tool for comparing the performance of DBMS.

There are two types of benchmarks in OODBMS, These are

**OO1 benchmark:** The object operations version 1 (OO1) benchmark is intended as a generic measure of OODBMS performance. It was designed to reproduce operations that are common in the advanced engineering applications. Such as finding all parts connected to a random part, all parts connected to one of those parts, and so on, to a depth of seven levels. This bench mark involves:
• Random retrieval of 1000 parts based on the primary key (the part number):

• Random insertion of 100 new parts and 300 randomly selected connections to these new parts, committed as one transaction;

• Random parts explosion up to seven levels deep, retrieving up to 3280 parts.

**OO7 benchmark:** The OO7 database schema is based on a complex parts hierarchy, where each part has associated documentation, and modules (objects at the top level of the hierarchy) have a manual. The tests are split into two groups. The first group is designed to test:

• Traversal speed (simple test of navigational performance similar to that measured in OO1)

• Traversal with updates (similar to the first test, but with updates covering every atomic part visited, a part in every composite part four times);

• Operations on the manual.

The second group contains declarative queries exact match, range searches, path Lookup, scan, a simulation of the make utility, and join.

7. **Advantages and Disadvantages:**
• Advantages:

1. Enriched modeling capabilities
2. Extensibility
3. Removal of impedance mismatch
4. More expressive query language
5. Support for schema evolution
6. Support for long-duration transactions
7. Applicability to advanced database applications
8. Improved performance.

• Disadvantages:

1. Lack of universal data model
2. Lack of experience
3. Lack of standards
4. Competition
5. Query optimization compromises encapsulation
6. Locking at object level may impact performance
7. Complexity
8. Lack of support for views
9. Lack of support for security.

8. CONCLUSION:

The OODBMS technology is boon to the today’s growing
Computational needs. The OODBMS proponents, which resembles the network debate of the 1970’s. This technology proponents claim that relational DBMSs are satisfactory for standard business applications but lack of the capability to support more complex applications. The relational supporters claim that relational technology is a necessary part of any real DBMS and that complex applications can be handled by extensions to the relational model. If OODBMSs are to become dominant they must change their image from being systems solely for complex applications to being systems that can also accommodate standard business applications with the same tools and the same ease of use as their relational counterparts.