OODBMS

Unit-3

Sub-Unit-3.2

OODBMS Architecture Approach

- According to existing application requirements, OODB architecture can be divided into following categories:

1. Extended Relational Model Approach
2. Semantic Database Approach
3. Object Oriented Programming Language extension approach
4. DBMS Generator approach

Extended Relational Model Approach

- Standalone OODB where all of the data available is stored in object data model. Thus there is no overhead in mapping objects to application objects. This is mostly useful for complex data.
- The database supports object inheritance similar to object oriented programming. This architecture is rare as the underlying design of the database is inefficient.
- Example: if class ‘A’ extends class ‘B’, they would have to be placed in different tables to allow for further inheritance of class ‘B’. This design then requires locks on all the classes involved and slow down transactions.

Semantic Database Approach

- Object Relational DBMS: OODB acts as a staging layer for existing data in relational database. The data in relational database are mapped to object models and stored in object data database. Thus allowing application which require object models to tap into the object database and reduce overhead of mapping relational data to objects.
- The second approach towards an OODB architecture is by enabling standard SQL database to support hierarchical data encapsulation.

OOPL Extension Approach

- Object-oriented concepts can be used as a design tool, and be encoded into, for example, a relational database (analogous to modeling data with E-R diagram and then converting to a set of relations).
- The concepts of object orientation can be incorporated into a programming language that is used to manipulate the database.

  a) Object-relational systems – add complex types and object-orientation to relational language.

  b) Persistent programming languages – extend object-oriented programming language to deal with databases by adding concepts such as persistence and collections.

Persistent Programming Language

- Persistent programming languages:

–**allow objects to be created and stored in a database** without any explicit format changes (format changes are carried out transparently).

–**allow objects to be manipulated in-memory i.e.** do not need to explicitly loaded from or store to the database.

– **allow data to be manipulated directly from the programming** language without having to go through a data manipulation language like SQL.

DBMS Generator Language Approach

- Since no DBMS can provide all the specialized functions and operations needed by a specific application area (e.g. text processing, GIS), these "toolkits" enable customizable DBMSs to be built. Examples include GENESIS and EXODUS .
- Further discussion of the data management requirements for environments such as project management, office documents, geographical and spatial information can be found in [Oxbor91].

Object Definition Language

- ODL is the standardized language for defining the structure of database with respect to the object data model.
- Example: Class declarations

  Interface < name > { elements = attributes, relationships, methods }

  Type Date Tuple {year, day, month}
- ODL creates a layer of abstraction making data language and database independent (Standalone OODB or Object Relational DB) to allow applications to move between compliant databases or different language implementations.

Object Definition Language

- The ODL (Object Definition Language) is an extension of CORBA's IDL and as such has a C++ flavor.
- ODL defines three components of the object oriented data model:
  a) Abstraction
  b) Inheritance
  c) Encapsulation

a) Abstraction

- In OODB, abstract data model is implemented as a graph, with vertices representing the objects and edges representing relations.
- The idea of persistent database objects and the general properties governing their creation, manipulation and deletion, is in fact captured as an abstract class called 'Database Object'.

Object Definition Language

a) Abstraction(contd.)

- **Object instance:** An entity in an object model is called an object instance.
- **Object identification:** Every object instance has a unique identity. This id used to reference object instances.
- **Object Class**: Similar object instances are grouped together into a class. The class defines the structure and the attributes are used to instantiate objects which conform to the classes specification.
- **Object reference or relationships:** The object model directly supports references. Object instances ”reference” each other using object identities.

Object Definition Language

b) Inheritance

- Derived directly from object oriented programming languages, object data models also allow for inheritance, polymorphism and overriding.
- The database schema defines data classes and their associations with one another. Associations may be expressed through variable-domain bindings and through inheritance (both static and dynamic).

Object Definition Language

c) Encapsulation:

- Encapsulation in the object model concept allows for including processing or behavior with the object instances defined by the class. This allows code and data to be packaged together.
- This includes an interface and an implementation for each object. The interface to an object is visible to an application. The implementation consists of attributes which represent its state and methods, which represent the operations which can be performed on the object.
- This provides "logical data independence" by Encapsulation, makes boundaries among objects clear, communication among objects explicit, and hides implementation details.

Object Query Language

- Developed by ODMG(Object Data Management Group), Object Query Language allows SQL-like queries to be performed on a OODB.
- Like SQL, it is a declarative language. Based loosely on SQL, OQL includes additional language constructs which allow for object oriented design such as operation invocation and inheritance.
- Query Structures look very similar in SQL and OQL but the results returned are different .

Object Query Language

- OQL is a Declarative query language
  - Not computationally complete
- Syntax based on SQL (select, from, where)
- Additional flexibility (queries with user defined operators and types)

1) That which is interpreted as a table name in SQL, is interpreted as a collection-object name in OQL;

2) That which is interpreted as the name of a column of a table in SQL is interpreted as an object characteristic name (an attribute, relationship, or operation name) in OQL.

Object Query Language

- Example: OQL query to obtain Voter names who are from the state of Colorado

Select distinct v.name from voters v

Where v.district = "Kathmandu"

Voter Id	Name	District
V01	Ram Kumar	Kathmandu
V02	Shyam	Bhaktapur
VO3	Hari	Lalitpur
V04	Sita	Kathmandu

Results from SQL

Table with rows

Name
Ram Kumar
Sita

Results from OQL

Collection of objects

String	String
Ram Kumar	Sita

OQL(contd.)

- Characterizing the objects to be retrieved involves defining a *selection, or filter predicate* (i.e. Boolean expression) that each object must satisfy. The standard operations that object-oriented systems provide for composing a predicate include
  - constants (for direct compa*rison),*
  - comparison operators (such as "less than"),
  - set operations (minimally union, intersection, and set difference),
  - combination (which acts like Cartesian product), and
- partitioning (which acts like projection).

select E

for each Emp!cyee E

where E.Division.C*ompany.President.Age > 50*