Concept of Object DataBases

Animesh Bhadra

Semester - 1

M.Tech - Adv Information Technology

- Disadvantages of Traditional Data Models.
  - Complex Structures for objects.
  - Longer Duration Transactions
  - New Data types for storing images / large textual items
  - Need to define non standard application specific operations.

- Advantages of Object Oriented Approach.
  - Not limited by data types and query languages.
  - Combines both *structure* and *operations* of Complex Objects.
  - Increasing use of Object-Oriented Programming languages.
  - Newer SQL standard are incorporating the changes related to OODB

OverView of Object-Oriented Concepts

- Object Oriented Approach are applied to
  - Databases.
  - Software Engineering
  - Knowledge Bases.
  - Artificial Intelligence.
  - Computer Systems.

- Property of Object Orientation.
  - *Class* : Blue Print of a Object.
  - *Abstract Data Type*:
    - Hides Internal data structures
    - *Encapsulation* : Specifies all possible external operations.
  - *Message Passing*:
  - *Inheritance* :

- **Pure OO Programming Language**
  - SmallTalk : A language explicitly designed to be object oriented.

- **Hybrid OO Programming Language.**
  - C++ : A language which incorporates OO Concepts into and Already existing language.

- **Object**
  - **State**: Value
  - **Behavior** : Operations.

- *Transient Object*
  - Object which exits only during program execution.
- *Persistent Object*
  - OO DataBase store persistent object permanently on secondary storage.
  - Allows the sharing of these objects among multiple program.
    - This requires incorporation of
      - Indexing Mechanisms.
      - Concurrency control
      - Recovery.

- Goal of OO DataBase
  - Maintain Direct correspondence between real world and database objects.
  - Provides **Object Identifier (OID)** a unique system generated identifier.
    - In relational model, if a primary key is changed, the tuple have new identity.
  - **Object Structure of arbitrary complexity**
    - In relational model, information about a object is often *scattered* over many relations.

- **Instance Variable**
  - Holds value which defines internal state.
  - Similar to an *attribute* in the relational model.
    - except, Instance variable can have encapsulation within the object, and are not visible from outside.
  - *Complete encapsulation* is relaxed in most OO Data Bases, as all the operations on an object be predefined.

- **Encapsulation**: Operation is defined in two parts.
  - *Signature or interface*: Specifies the operation name and arguments (parameters)
  - *Method or Body*: Specifies implementation of the operations.
- Operations can be invoked by passing a *message* to an object.
  - This encapsulation permits modification of the internal structure of an object.

- **Inheritance**
  - Easier to develop the data type of a system incrementally.
  - Reuse existing type definitions when creating new type of object.

Object Identity, Object Structure, and Type Constructors

Object Identity

- An OO Database system provided a **unique identity** to each independent object stored in database.
  - Implemented Via system generated, *Object Identifier (OID)*
    - Value of OID is not visible to the external user.
    - Internally managed by system to identify each object uniquely.

- Property of OIDs
  - OIDs are **immutable**,
  - OID value of a particular object should not change.
    - Each OID should be used only once.

- Some system required to have OIDs for Data Type.
  - OO Database allows, for representation of both object and values.
    - Object have OIDs
    - Value does not have OIDs.

- v = object state.
  - Object State is based on the Constructors.
    - atomic Values
    - the state V is in the form of
      - an = attribute name
      - in = OID
    - Is a set of Object Identifiers.

Encapsulation of Operations, Methods, and Persistence

- Encapsulation is related to *abstract data type* and *information hiding*
- Traditional database models, did not implement these.

- **Interfaces** | **Signature**: external users are aware of
  - Defines name and arguments of each operations.
  - Invoked by the use of **Message Passing**

- Instance Variable
  - **Visible** : available for reading by external sources
  - **Hidden** : always encapsulated
    - Update Operations.

- **Naming**
  - Give the Object a Unique Name.
  - Given by means of
    - Specific Statement
    - Operation in programs.
  - Provides as an **entry points** to the database.

- **Reachability**
  - Make object reachable from some persistent object.
  - An Object B is said to be **reachable** from an object A if a sequence of reference in the object graph lead from Object A to Object B.

Type and Class Hierarchies and Inheritance

One main characteristic of OO Data Base is that they allow type hierarchies and inheritance.

Example:

```
PERSON : Name,Address,Birth_Date,Age,SSN

EMPLOYEE subtype-of PERSON : Salary,Hire_Date,Senirority
```

Complex Object

There are two types of complex objects.
Structured : Made of Components and defined by applying the available type constructor recursively
UnStructured. : Most large data like image or textual object

Structured Complex Objects

- A **Structured complex object** differs from an unstructured complex object in that the Object's structure is defined by repeated application of the type constructors.
- **Ownership Semantics** : Applies when the sub-objects of a complex object are encapsulated within the complex object.
- **Reference Semantics**: Applies when the components of the complex objects are themselves independent objects but may be referenced from complex objects.

Other Object Oriented Concepts

1. Polymorphism (Operator Overloading)
2. Multiple Inheritance and Selective Inheritance
3. Versions and Configurations

Polymorphism (Operator Overloading)

- This concepts allows that same *operator name* or *symbols* to be bound to two or more different *implementations.*

```
GEOMETRY_OBJECT : Shape, Area, ReferencePoint

RECTANGLE subtype-of GEOMETRY_OBJECT(Shape = 'rectangle') : Width, Height
TRIANGLE subtype-of GEOMETRY_OBJECT(Shape = 'triangle') : Side1,Side2,Angle
```

Versions and Configurations

- **Concurrent Engineering**
- **Version Graph**
- **Configurations**
