XXII. Object-Oriented Database Design

Object-Oriented Database Management Systems (OODBMS)
Distributed Information Systems and CORBA
Designing Data Management Classes
The Persistent Object Approach
The Database Broker Approach

OODBMS

- Object-Oriented DBMS (OODBMS) are DBMS which are based on an Object-Oriented Data Model.
- Such data models are often inspited by OO programming languages, such as SmallTalk or C++.
- OODBMS are capable of storing complex objects, I.e., objects that are composed of other objects, and/or multi-valued attributes.
- The great advantage of OODBMS is that it is not necessary to transform the UML classes into a logical schema (e.g., relational).
- Their main disadvantage is that their technology is immature and they are only used in niche applications, such as CAD.

OODBMS vs RDBMS

- RDBMS have been around for more than 20 years, OODBMS are relatively new;
- RDBMS can handle >10^{10} records, OODBMS up to 10^{7}.
- OODBM good for storing complex descriptions (e.g., a plant schematic), RDMSs appropriate for simple, "flat" data.
- RDBMS control the DB market (>90%), OODBMS own <5% of the market.
- Most commercial RDBMS come with an "Object-Relational" extension which implements an object database on top of a RDBMS.

Object Database Standard

- Object Data Management Group has set a standard for Object Databases (version 3.0).
  - ODL - Object Definition Language
  - OML - Object Manipulation Language
- However, individual ODBMS do not necessarily conform to the standard (…usual story…)

ObjectStore PSE: An OODBMS

- ObjectStore PSE (Persistent Storage Engine):
  - provides persistence for Java programs;
  - builds navigational structure into the database;
  - requires all persistent objects to be instances of sub-classes of `COM.odi.Persistent`;
- ObjectStore provides full OODBMS functionality.

Distributed Information Systems

- Most information systems are distributed.
- This means that the objects that participate in a particular use case need not be on the same machine with other objects and users they are supposed to interact.
- One can use Remote Procedure Calls-RPC (C/C++) and Remote Method Invocation-RMI (Java).
- The object-oriented industrial standard for distributed objects is CORBA (***Common Object Request Broker Architecture***)

CORBA

- CORBA separates the interface of a class from its implementation. The implementation runs on one machine, the interface can be compiled on several other machines.
- When accessed by a client program, an object is treated as though it is in memory on the client machine; however, the object may actually be located on another machine.
- When the client program sends an object a message to invoke one of its operations, the message and parameters are converted into a format that can be sent over the network (***marshalling***)
- At the other end, the server unmarshals the data back into a message with arguments, and passes these on to the implementation of the target object.
- CORBA achieves this by means of programs known as ORBs (Object Request Brokers) that run on each machine.
- The ORBs communicate with each other by means of an Inter-ORB Protocol (IOP).
- Over the Internet, the protocol used is IIOP (Internet IOP).

Designing Data Management Classes

- Idea is to not use a DBMS (Relational or Object-Oriented.)
- Instead,design data management classes which handle persistence, caching, etc.
- These classes decouple applications from their persistent storage.
- Use data management classes whenever you need to:
  - ✓ Store an application object;
  - ✓ Search for or retrieve stored objects;
  - ✓ Interface with an external database.
- This solution won't work for large data sets!

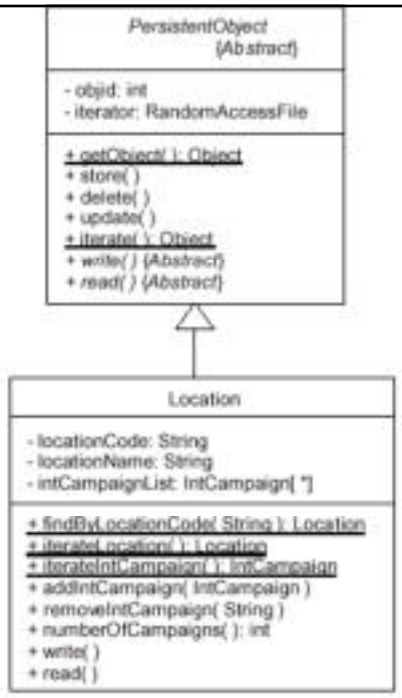
Data Storage Layer

- Options for locating the operations that handle the tasks of storing and retrieving objects:
  - ✓ All persistent objects in the system could inherit methods for storage from an abstract superclass - `PersistentObject`
  - ✓ Introduce separate classes into the system whose role is to deal with the storage and retrieval of other classes (Database broker approach)

`PersistentObject` Superclass Approach

- A superclass `PersistentObject` encapsulates the mechanisms for an object of any class to store itself in, or retrieve itself from a database.
- This superclass implements operations to get an object by object identifier, store, delete and update objects and to iterate through a set of objects (write and read operations).

PersistentObject

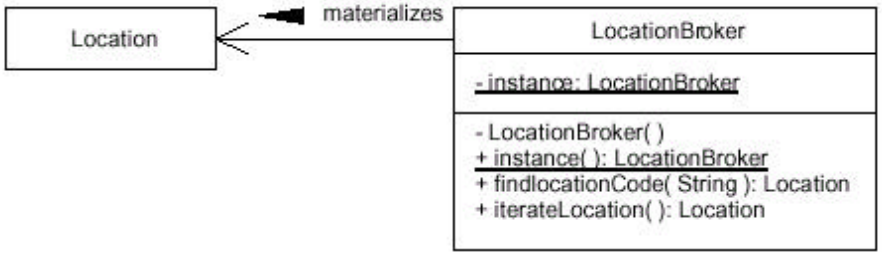


Database Broker Approach

- Each persistent class could be responsible for its own storage...
- ...but...
  - highly coupled (to storage mechanism);
  - lessens class cohesion;
  - class must now have expert knowledge of storage tasks;
  - these are unrelated to application tasks.
- Solution: indirection (add a go-between).
- Separates the business objects from their data storage implementation.
- The classes that provide the data storage services will be held in a separate package.
- For each business class that needs to be persistent, there will be an associated database broker class.

The Broker Class

- The broker class provides the mechanisms to materialize objects from the database and dematerialize them back to the database



The Database Broker

- The database broker object is responsible for:
  - ✓ “materialising” objects,
  - ✓ “dematerialising” objects,
  - ✓ caching objects.
- Application classes are insulated from storage.
- Allows migration of storage sub-systems, e.g., implement on existing relational system.
- Replace this with OODBMS.
- Application programs unaffected by change.

Caching Objects

- Objects can be cached for efficiency.
- The cache is a collection maintained by the database broker.
- When an object is requested, the cache is searched first.
- If the object sought is not in the cache it is materialised by the database broker from the database.

Transaction Management

- To manage transactions, we need to keep track of all changes made by a transaction, in case the transaction is aborted before it completes execution (and commits all its changes.)
- Multiple caches can be used for transaction management:
  - ***new clean cache:*** newly created objects
  - ***new dirty cache:*** newly created objects that have been amended
  - ***new delete objects:*** newly created objects that have been deleted
  - ***old clean cache:*** objects retrieved from the database
  - ***old dirty cache****:* retrieved objects that have been ammended
  - ***old delete objects****:* retrieved objects that have been deleted

Collections

- In systems where collection classes are used in design, these may be replaced by database broker objects.
- Database objects provide collection-like services for large volumes of data (more than you would maintain in a collection class in memory).

Additional Reading

- Rumbaugh et al. *Object-Oriented Modeling and Design*. Prentice-Hall, 1991; Chapter 17 - Relational Databases
- Larman, *Applying UML and Patterns*. Prentice-Hall, 1998. Chapter 38 - Frameworks, Patterns and Persistence
- Coad, *Object Models - Strategies, Patterns and Applications*. Prentice-Hall, 1997; Appendix C - Data Management

