Object Oriented Analysis and Design

UNIT-IV: Object-oriented Design Axioms- Design Patterns- Designing Classes- Class Visibility- Refining Attributes-designing Methods And Protocols- Access Layer: DBMS-OODBMS.

TEXT BOOK : Ali Bahrami, “Object Oriented Systems Development”, TATA Mcgraw-hill Edition, 2008.

PREPARED BY

DR. D. DEVAKUMARI

Content

- Object-Oriented Design Axioms
- Design Patterns
- Designing classes
- Class visibility
- Refining attributes
- designing methods and protocols
- Access layer: DBMS
- OODBMS.

Object-Oriented Design Axioms

- An AXIOM is a fundamental truth that always is observed to be valid and for which there is no counter example or exception.
- A theorem is a proposition that may not be self evident but can be proven from accepted axioms.
- A corollary is a proposition that follows from an axiom or another proposition that has been proven.
- Axiom l. The independence axiom. Maintain the independence of components.
- Axiom 2. The independence axiom. Minimize the information content of the design.
- Axiom I States that, during the design process, as we go from requirement and use case to a system component each component must satisfy that requirement without affecting other requirements.
- Axiom 2 is concerned with simplicity. "The best theory explains the known facts with a minimum amount of complexity and maximum simplicity and straight forwardness.

Design Patterns

- Design patterns are devices that allow systems to share knowledge about their design, by describing commonly recurring structures of communicating components that solve a general design problem within a particular context.
- In programming, we have encountered many problems that occurred before and will occur again. the question we must ask ourselves is how we are going to solve it this time.

Class visibility: Designing well-defined public, private, and protected protocols.

- Two problems. One is the protocol, or interface to the class operations and its visibility; and the other is how it is implemented.
- Public protocols define the functionality and external messages of an object; private protocols define the implementation of an object.
- The private protocol (visibility) of the class, includes messages that normally should not be sent from other objects; it is accessible only to operations of that class.
- The public protocol [visibility) defines the stated behaviour of the class as a citizen in population and is important information for users as well as future descendants. So it is accessible to all classes.
- In a protected protocol (visibility), subclasses the can use the method in addition to the class itself.

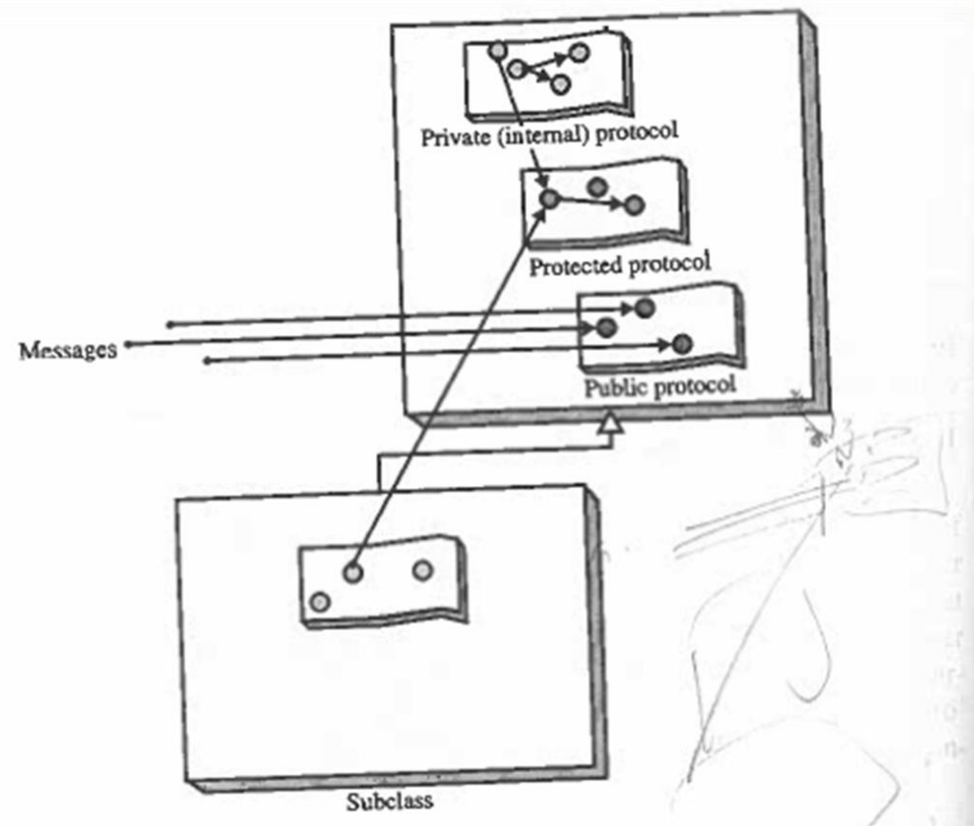


FIGURE 10–1
Public protocols define the functionality and external messages of an object, while private protocols define the implementation of an object.

private and protected protocol Layers: Internal

- Items in these layers define the implementations of the object.

public protocol Layer: External

- Items in this layer define the functionality of the object

Designing classes

- The three basic types of attributes are

1. single-value attributes.

2. Multiplicity or multivalue attributes.

3. Reference to another object, or instance connection.

Designing Methods and Protocols:

- Once you have designed your methods in some formal structure such as UML activity diagrams with an OCL - object constraint language description, they can be converted to programming language manually or in automated fashion (ie., using CASE tools).

- A class can provide several types of Methods:-

1. **Constructor**: Method that creates instances (objects) of the class.

2. **Destructor**: The Method that destroys instances.

3. **Conversion method**: The method that converts a value from one unit of measure to another.

4. **Copy method**: The method that copies the contents of one instance to another instance.

5. **Attribute set**: The method that sets the values of one or more attributes.

6. **Attribute get**: The method that returns the values of one or more attributes.

7. **I/O methods**: The methods that provide or receive data to or from a device.

8. **Domain specific**: The method specific to the application.

- Your goal should be to maximize cohesiveness among objects and software components to improve coupling because only a minimal amount of essential information should be passed between components.

Design Issues: Avoiding Design pitfalls

- It is much better to have a large set of simple classes than a few large, complex classes.
- Your goal should be maximum reuse of what you have to avoid creating new classes as much as possible.
- Each class must have a single, clearly defined purpose.
- Move some functions into new classes that the object would use. Apply corollary 1 (uncoupled design with less information content).
- Break up the class into two or more classes. Apply corollary 3 (large number of simple classes).

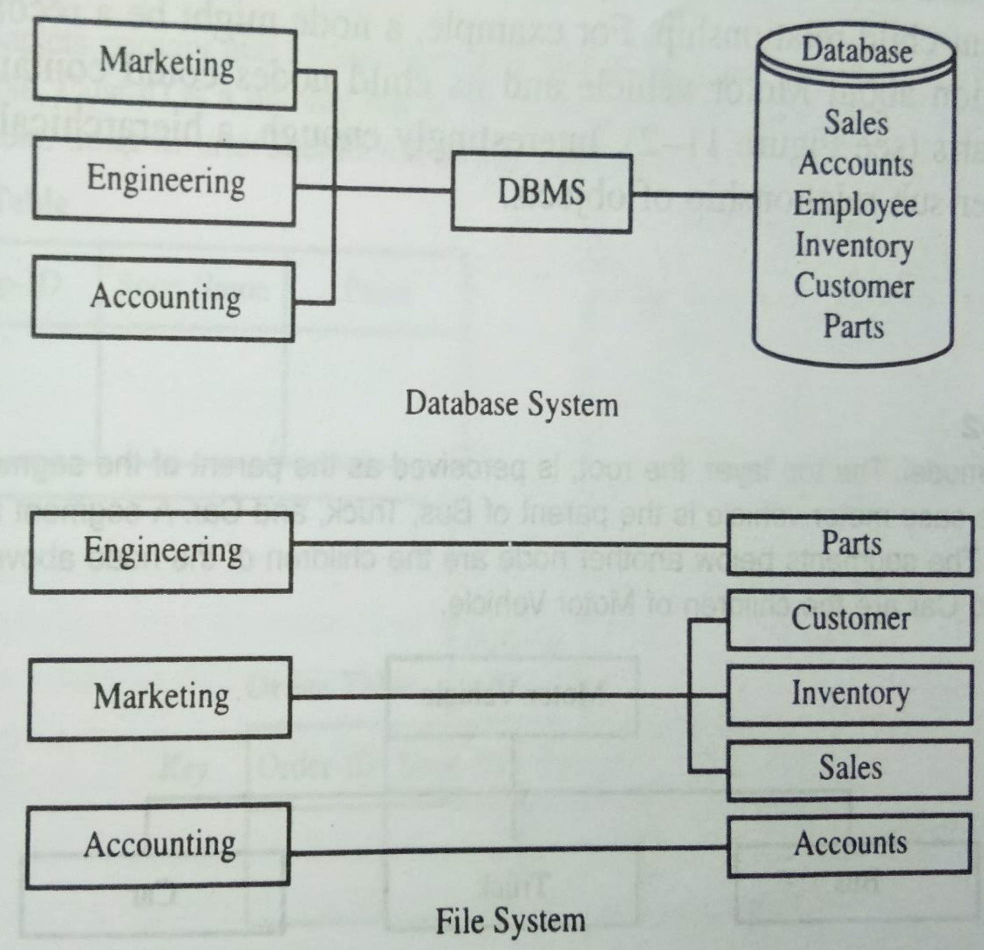
UML operation presentation

- The following operation presentation has been suggested by the UML.
- The operation syntax is this:-
- visibility name:(parameter-list): return-type-
- expression
- where Visibility is one of:
- + public visibility (accessibility to all classes).
- # protected Visibility (accessibility to subclasses and operations of the class).
- -private visibility (accessibility only to operations of the class).

Database Management systems

- A DBMS is a set of programs that enable the creation and maintenance of a collection of related data.
- DBMS contains not only the data but a complete definition of the data formats it manages.
- This description is known as the schema, or meta-data, and contains a completed definition of the data formats, such as the data Structures types, and constraints.

FIGURE 11–1
Database system vs. file system.



Database Views

- The DBMS provides the database users with a conceptual representation that is independent of the low-level details (physical view) of how the data are stored.
- The database can provide an abstract data model that uses logical concepts such as field, records, and tables and their interrelationships.

Database Models

- Database Models may be grouped into two categories:
- Conceptual models and implementation models.
- The conceptual model focuses on the logical nature of that data presentation.
- Therefore, the conceptual model is concerned with what is represented in the database and the implementation model is concerned with how it is represented.

Hierarchical Model: -

- The hierarchical model represents data as a single- rooted tree.
- Each node in the tree represents a data object and the connections represent a parent-child relationship.

Network Model: -

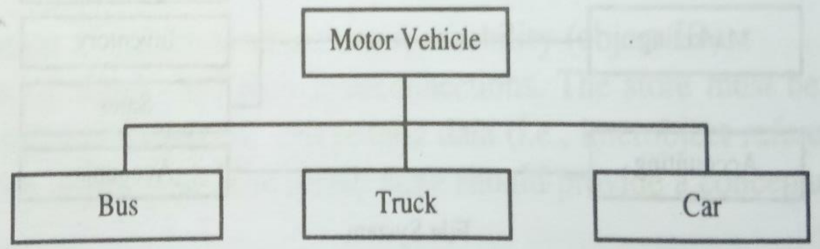
- Unlike the hierarchical model, a network model's record can have more than one parent.

Relational Model:-

- The columns of each table are attributes that define the data or value domain for entries in that column.
- The rows of each table are tuples representing individual data objects being stored.
- A relational table should have only one primary key. A primary key is a combination of one or more attributes whose value unambiguously locates each row in the table.
- In soup-ID, Lust-ID, and order-ID are primary keys in soup, customer, and order tables.
- A foreign key is a primary key of one table that is embedded in another table to link the tables.

FIGURE 11-2

A hierarchical model. The top layer, the root, is perceived as the parent of the segment directly below it. In this case motor vehicle is the parent of Bus, Truck, and Car. A segment also is called a node. The segments below another node are the children of the node above them. Bus, Truck, and Car are the children of Motor Vehicle.



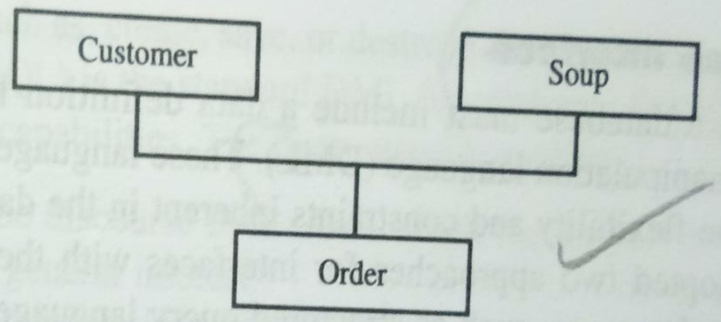
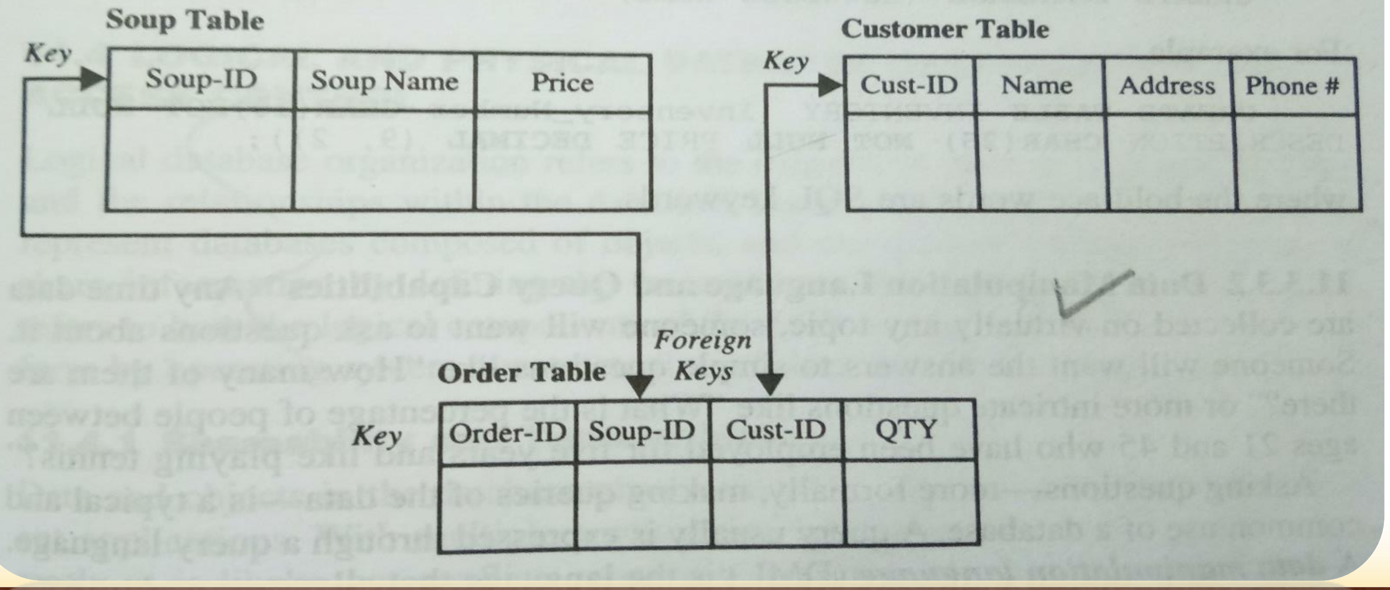


FIGURE 11-3

Network model. An Order contains data from both Customer and Soup.

FIGURE 11-4

The figure depicts primary and foreign keys in a relation database. Soup-ID is a primary key of the Soup table, Cust-ID is a primary key of the Customer table, and Order-ID is a primary key of the Order table. Soup-ID and Cust-ID are foreign keys in the Order table.



Database Interface:-

- The interface on a database must include a data definition language (DDL), a query, and data manipulation language (DML). Database systems have adopted two approaches for interfaces with the system.

- One is to embed a database language, such as structured query language (SQL), in the host programming language. Another approach is to extend the host programming language with database related constructs.

Database schema and Data Definition language

- A data definition language (DDL) is the language used to describe the structure of and relationship between objects stored in a database. This structure of information is termed the database Schema.

CREATE SCHEMA AUTHORIZATION [creator)

CREATE DATABASE (Database name)

For example,

CREATE TABLE INVENTORY (Inventory_Number CHAR (10) NOT NULL DESCRIPTION CHAR (25) NOT NULL PRICE DECIMAL (9,2)):

Data Manipulation language and Query capabilities :_

- A data manipulation language (DML) is the language that allows users to access and manipulate (such as, create, save, or destroy) data organization. The Structured query language (SQL) is the standard DML for relational DBMSs.

object-oriented Database Management systems:

- The object-oriented language management system is a marriage of object oriented programming and database technology to provide what we now call object-oriented databases.

- Many advantages accrue from including the definition of operations with the definition of data.

- First, the defined operations apply universally and are not dependent on the particular database application running at the moment.

- Second, the data types can be extended to support complex data Such as multimedia by defining new object classes that have operations to support the new Kinds of information.

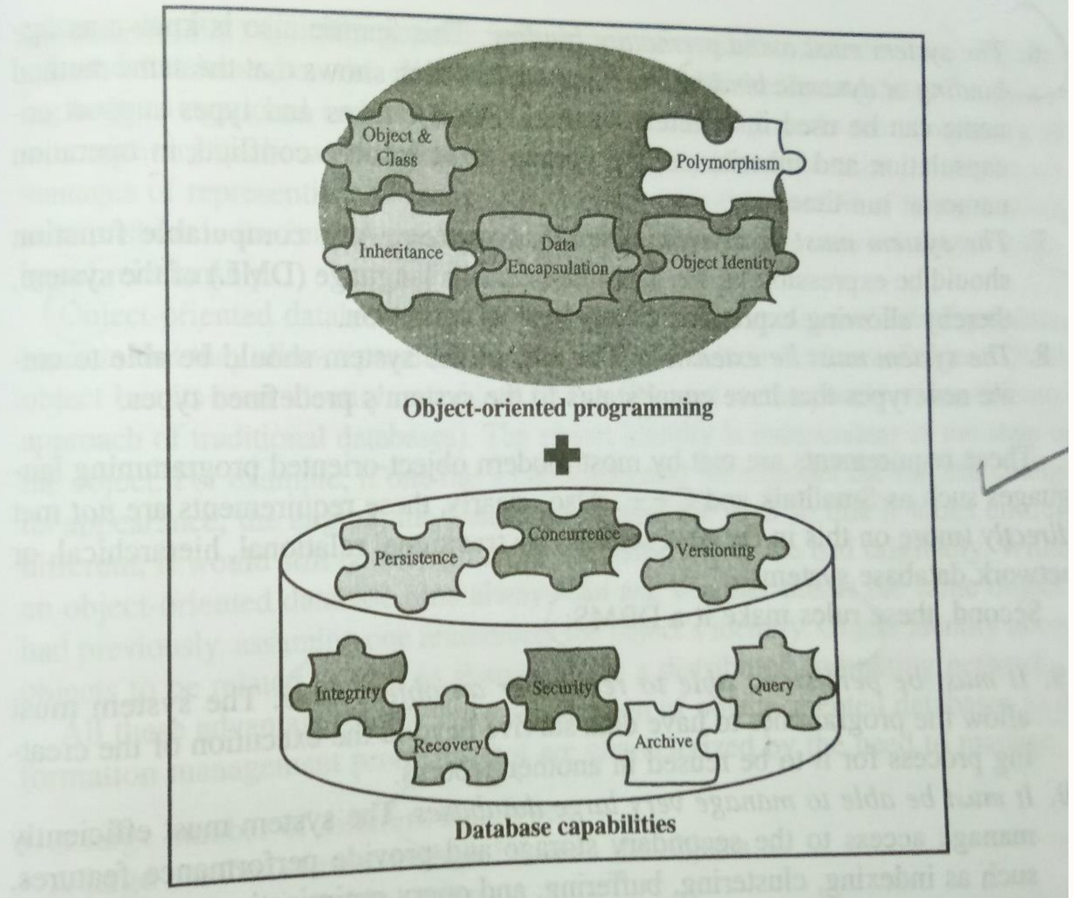


FIGURE 11–10
The object-oriented database management system is a marriage of object-oriented programming and database technology.

The rules that make it an object-oriented system are as follows-

1. The system must support complex objects. A system must provide simple atomic types of objects (integers, characters, etc.) from which complex objects can be built by applying constructors to atomic objects or other complex objects or both.
2. Object identity must be supported. A data object must have an identity and existence independent of its values
3. Objects must be encapsulated. An object must encapsulate both a program and its data.
4. The system must support types or classes. The system must support either the type concept embodied by CTH) or the class concept (embodied by Smalltalk).
5. The system must support inheritance. classes and types can participate in a class hierarchy. The primary advantage of inheritance is that it factors out shared code and interfaces.
6. The system must avoid premature binding. This feature also is known as late binding or dynamic binding.
7. The system must be computationally complete.
8. The system must be extensible.

Rules Make it a DBMS:

1. It must be persistent, able to remember an object state.
2. It must be able to manage very large databases.
3. It must accept concurrent users.
4. It must be able to recover from hardware and software failures.
5. Data query must be simple.

Object-oriented Databases versus Traditional Databases: -

- One obvious difference between the traditional and object_oriented databases is derived from the object's ability to interact with other objects and with itself.
- In contrast to conventional database systems, where records play a passive role. yet another distinguishing feature of object-oriented database is inheritance.
- Relational database systems do not explicitly provide in heritance of attributes and methods.
- Object-oriented database abo differ from the more traditional relational databases in that they allow representation and storage of data in the from of objects.
- The application of object-oriented databases to information management problems that are characterized by the need to manage

1. A large number of different datatypes.

2. A large number of relationships between the objects.

3. Objects with complex behaviours.

- Application areas where this kind of complexity exists include engineering, manufacturing,
- Simulations, office automation, and large information systems ("NO more fishing for Data" is a real-world example of this).