VIT-AP UNIVERSITY

Course code : **CSE2007**
Course title : **Database Management System**
Module : **1**
Topic : **2**

Introduction to Database Management System

Objectives

This session will give the knowledge about

- Data models
- DBMS Architecture
- DDL Queries
- DML Queries

Data Models

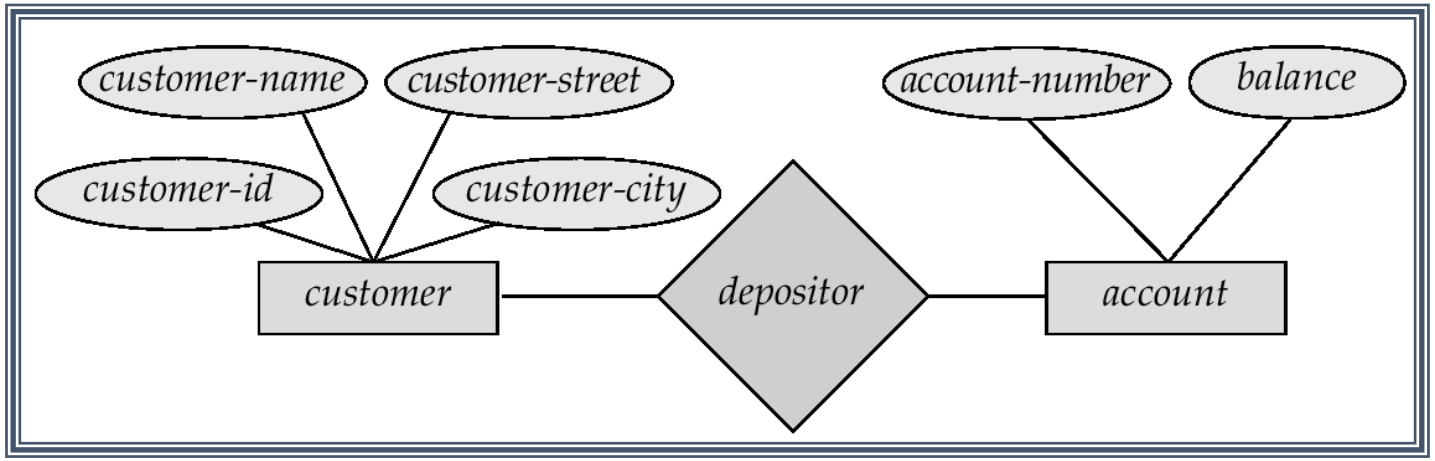
The underlying structure of the database is called as data model. The different types of data models are:

- Entity relationship model
- Relational model
- Hierarchical model
- Network model
- Object oriented model
- Object relational model

Entity relationship model

The entity relationship model consists of a collection of basic objects, called entities and its relationship with other entities.

- Entity: an entity is a ‘thing’ or ‘object’ in the real world
- Relationship: a relationship is an association among several entities.



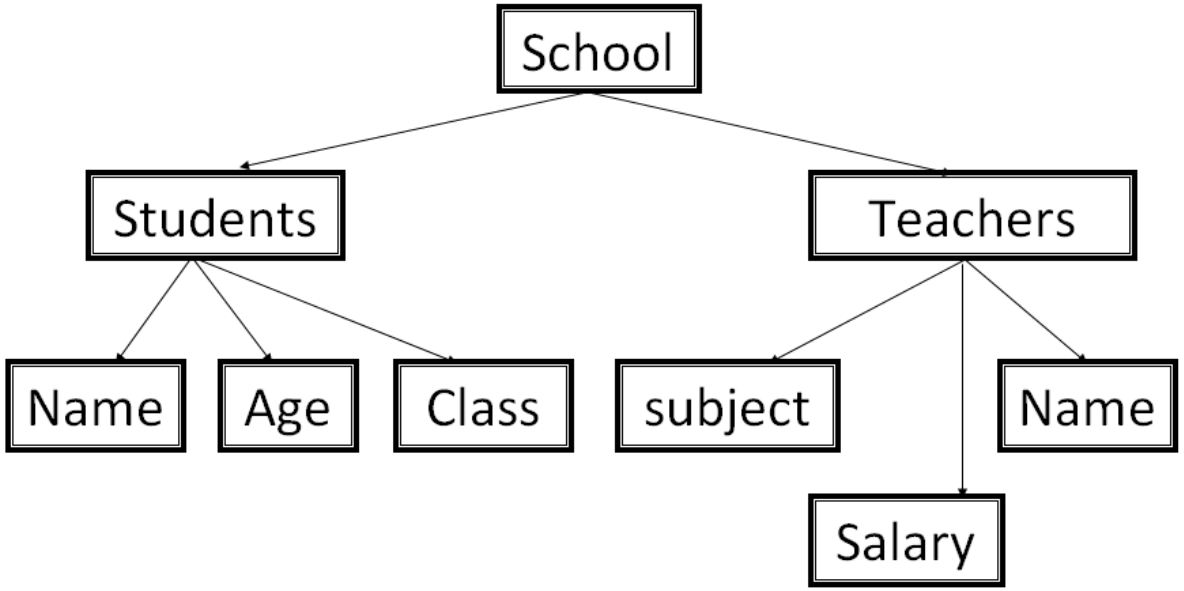
Relational model

The relational model represents data the relationships among data by a collection of tables, which has a number of columns with unique names.

PROPERTYID	RENTALAMOUNT	NOOFBEDROOMS	LOCATION	CITY
CHE1101	20000	2	AnnaNagar	Chennai
BEN1102	20000	2	HSR	Bengaluru
CHE1103	8000	1	Tambaram	Chennai
BEN1104	25000	2	HSR	Bengaluru
CHE1105	12000	2	Annanagar	Chennai

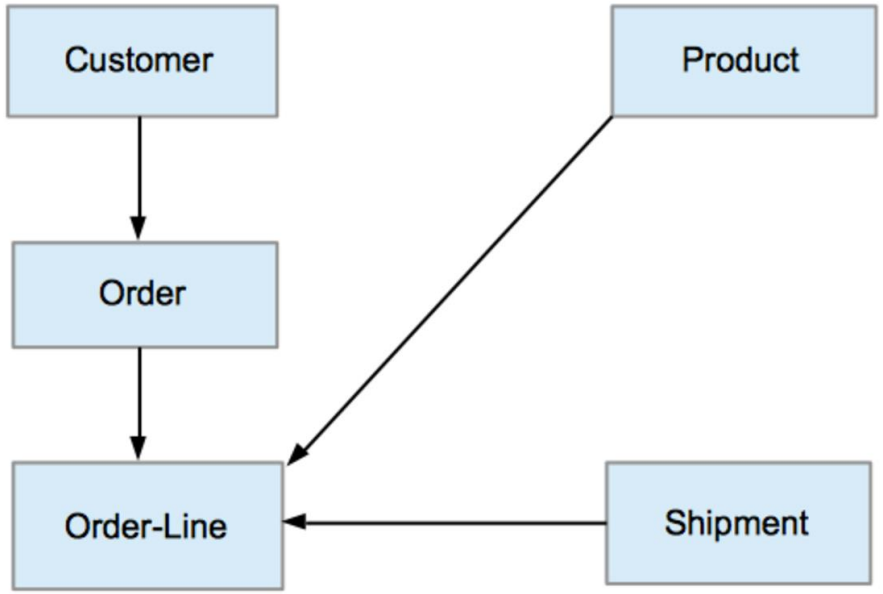
Hierarchical Model

Hierarchical database links records together in a tree data structure such that each record type has only one owner.



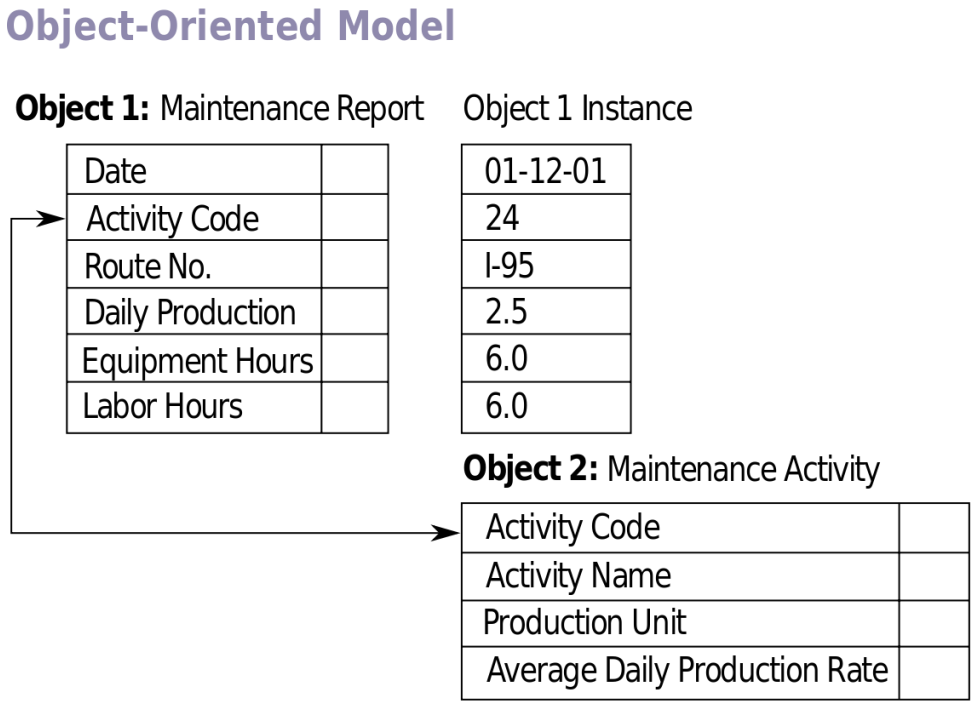
Network Model

Network database is based on directed graph theory. It replaces the hierarchical tree with a graph thus allowing more general collection among the data.



Object Oriented Model

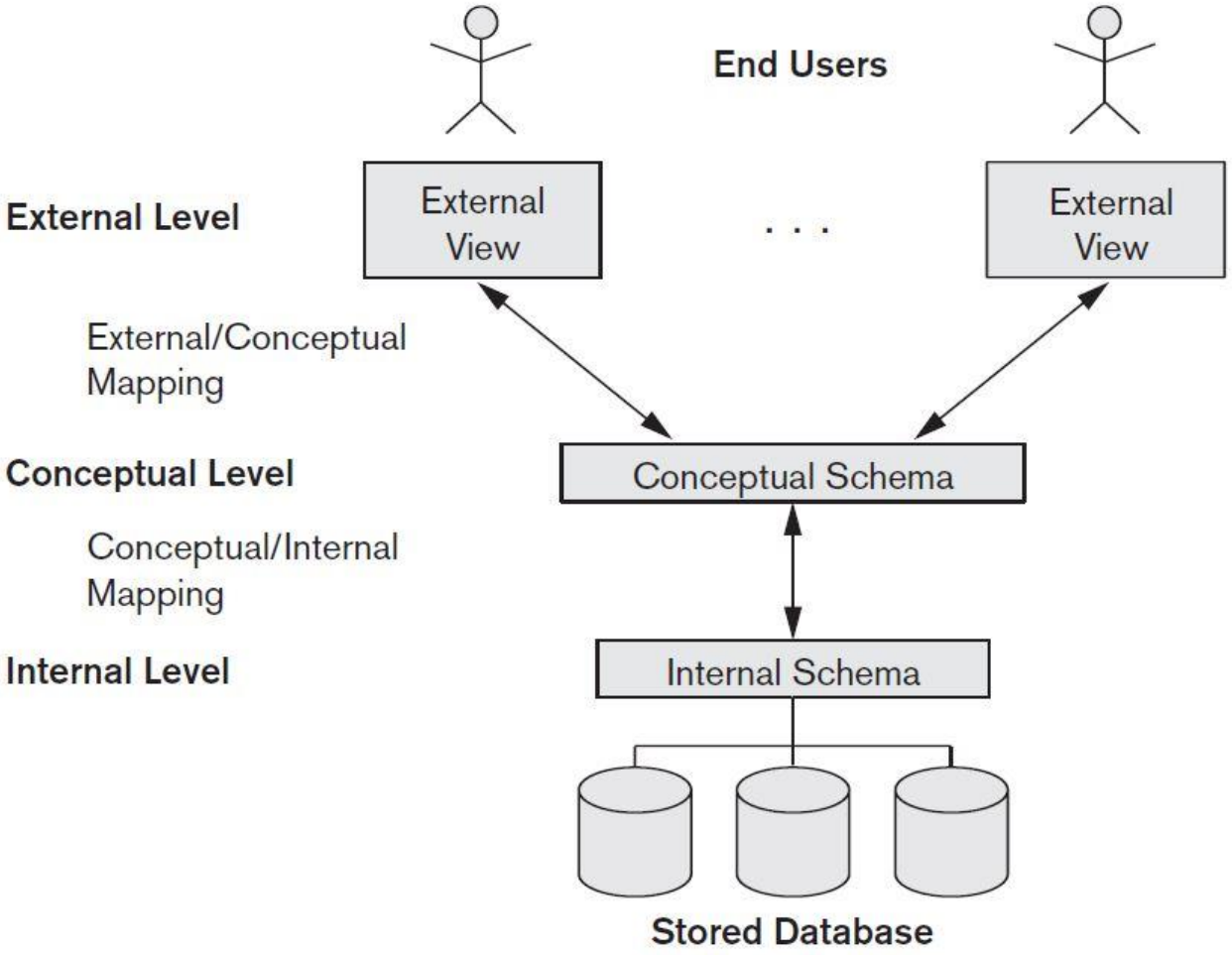
The object oriented model is based on a collection of objects. An object contains values stored in instance variables within the object.



Object Relational Model

A system that includes both object infrastructure and a set of relational extenders is called an object relational model. Object relational systems combines the advantages of modern object oriented programming languages with relational database feature.

Three-Schema Architecture



Three-Schema Architecture

1. The **internal level** has an **internal schema**, which describes the physical storage structure of the database. It uses a physical data model and describes the complete details of data storage and access paths for the database.

2. The **conceptual level** has a **conceptual schema**, which describes the structure of the whole database for a community of users. It hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints.

3. The **external** or **view level** includes a number of **external schemas** or **user views**. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group.

Data Independence

Data independence can be defined as the capacity to change the schema at one level of a database system without changing the schema at the next higher level. We can define two types of data independence:

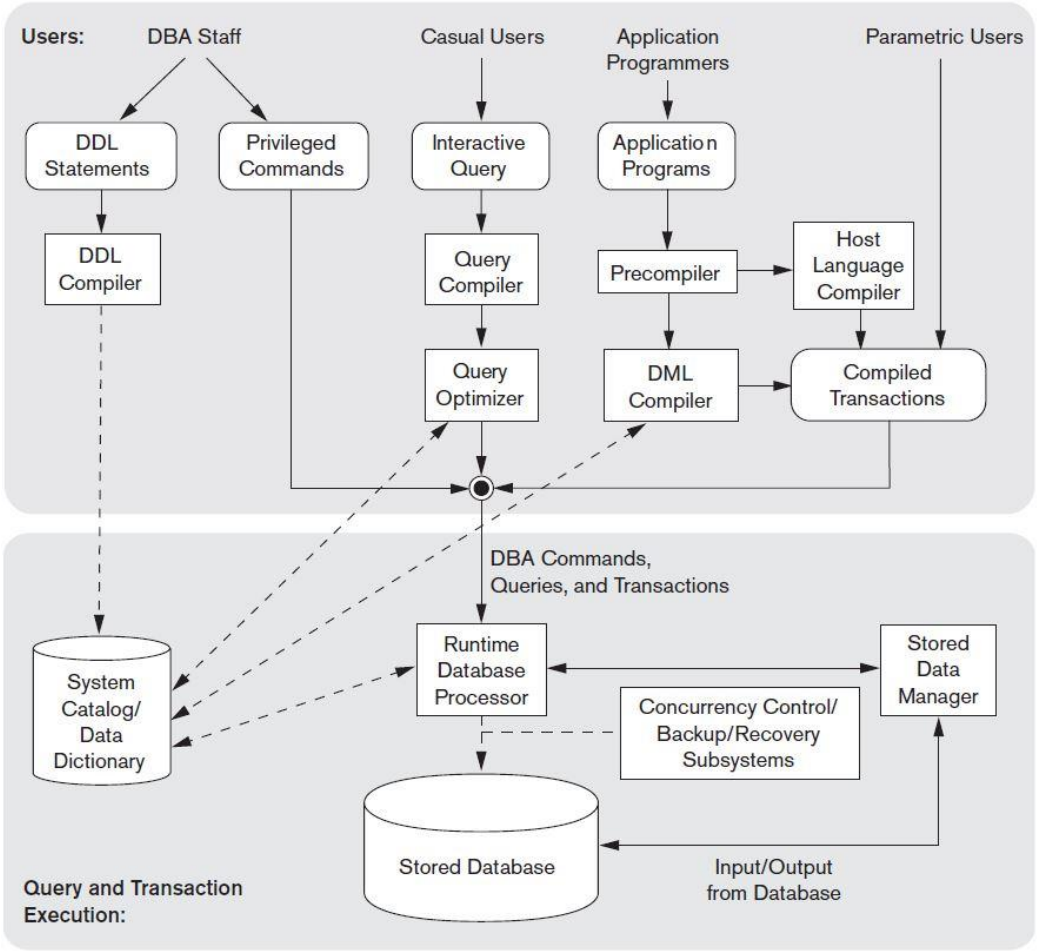
- **Logical data independence** is the capacity to change the conceptual schema without having to change external schemas or application programs.
- **Physical data independence** is the capacity to change the internal schema without having to change the conceptual schema.

Data Independence

- **physical data independence** exists in most databases and file environments where physical details such as the exact location of data on disk, and hardware details of storage encoding, placement, compression, splitting, merging of records, and so on are hidden from the user.

- **logical data independence** is harder to achieve because it allows structural and constraint changes without affecting application programs – a much stricter requirement.

Component modules of a DBMS



Component modules of a DBMS

- **Buffer management** module to schedule disk read/write, because this has a considerable effect on performance. Reducing disk read/write improves performance

- A **higher-level stored data manager** module of the DBMS controls access to DBMS information that is stored on disk.

- **The DDL compiler** processes schema definitions, specified in the DDL, and stores descriptions of the schemas (meta-data) in the DBMS catalog.

Component modules of a DBMS

- **Interactive query interface** is to provide the interface between the user and DBMS for basic information

- **Query compiler** used to parse and validate the correctness of the query syntax, file names and data elements, and so on. It also compiles the verified queries them into an internal form.

- **Query optimizer** is concerned with the rearrangement and possible reordering of operations, elimination of redundancies, and use of correct algorithms and indexes during execution.

Component modules of a DBMS

- **The precompiler** extracts DML commands from an application program written in a host programming language.

- **DML compiler** is to compile DML commands into object code for database access.

- The **runtime database processor** executes

  (1) the privileged commands,

  (2) the executable query plans, and

  (3) the canned transactions

  with runtime parameters.

Component modules of a DBMS

- It works with the **system catalog** and may update it with statistics.
- It also works with the **stored data manager**, which in turn uses basic operating system services for carrying out low-level input/output (read/write) operations between the disk and main memory
- **Concurrency control and Backup and recovery** are integrated into the working of the runtime database processor for purposes of transaction management.

Structured Query Languages

SQL is an interface between the user and the data base.

SQL enables

1. Table creation in the data base.
2. Querying the data base for information.
3. Modifying the data in the data base.
4. Deleting the data in the data base, etc.,

Database Languages

- If DBMSs has no separation of levels, the **data definition language (DDL)**, is used by the DBA and by database designers to define both (conceptual and internal) schemas.

- If DBMSs has clear separation of levels, the DDL is used to specify the conceptual schema only. Another language, the **storage definition language (SDL)**, is used to specify the internal schema.

- For a true three-schema architecture, the **view definition language (VDL)**, to specify user views and their mappings to the conceptual schema.

- **Most DBMSs uses the DDL** is to define both conceptual and external schemas

Database Languages

- DBMS provides **data manipulation language (DML)** for manipulations include retrieval, insertion, deletion, and modification of the data.
- There are two main types of DMLs.
  - **A high-level or nonprocedural DML** statements either to be entered interactively from a display monitor or terminal or to be embedded in a any programming language.
  - **A low-level or procedural DML** must be embedded in a general-purpose programming language. It retrieves individual records from the database and processes each separately. Low-level DMLs are also called record-at-a-time DMLs.

Data Definition Languages

DDL specifies a set of definitions of data base.

DDL consists of

1. Create : Create a table
2. Alter : Alter the table
3. Truncate : Cut the table
4. Drop : Delete the table

Data Manipulation Languages

A data manipulation language enables the users to access and manipulate the data.

DML consists of

1. Select : Select the values from the table.
2. Insert : Insert the values into the table.
3. Delete : Delete the values from the table.
4. Update : Update the values in the table.

SQL Examples

1) **Create** : Creating an employee table.

Statement for creating a table:

SQL> **create table** employee (name **varchar(10)**, id **number(10)**, address **varchar(20)**, salary **number(5)**);

SQL> table created

SQL Examples

2) **Insert**: Inserting values in the employee table.

Statement for inserting values:

SQL>**insert into** employee **values (**‘divya’, 01,’cbe’, 30000**);**

SQL> 1 row updated

SQL>**insert into** employee **values (**‘vani’, 02,’cbe’, 45000**);**

SQL> 1 row updated

SQL>**insert into** employee **values (**‘kalai’, 03,’cbe’, 50000**);**

SQL> 1 row updated

SQL Examples

3) **Display**: used to display the table

SQL> **select * from** employee;

Employee Table

name	id	address	salary
divya	01	cbe	30000
vani	02	cbe	450000
kalai	03	cbe	50000

Summary

This session will give the knowledge about

- Data models
- DBMS Architecture
- DDL Queries
- DML Queries