UNIT - I
INTRODUCTION to DBMS (Part -I)

1.1 What is a Database Management System?

Data

- Data is raw fact or figures or entity.
- When activities in the organization takes place, the effect of these activities need to be recorded which is known as Data.

Information

- Processed data is called information
- The purpose of data processing is to generate the information required for carrying out the business activities.

Database

- Database may be defined in simple terms as a collection of data
- A database is a collection of related data.

Database Management System

- A Database Management System (DBMS) is a collection of program that enables user to create and maintain a database.
- The DBMS is hence a general purpose software system that facilitates the process of defining
  Constructing and manipulating database for various applications.

- ***History***
  - 1950s-60s: magnetic tape and punched cards
  - 1960s-70s: hard disks, random access, file systems
  - 1970s-80s: relational model becoming competitive
  - 1980s-90s: relational model dominant, object-oriented databases
  - 1990s-00s: web databases and XML

Why Study Databases?

- They touch every aspect of our lives
- ***Applications:***
  - Banking: all transactions
  - Airlines: reservations, schedules
  - Universities: registration, course enrolment, grades
  - Sales: customers, products, purchases
  - Manufacturing: production, inventory, orders, supply chain
  - Human resources: employee records, salaries, tax deductions
  - Telecommunications: subscribers, usage, routing
  - Computer accounts: privileges, quotas, usage
  - Records: climate, stock market, library holdings
- ***Explosion of unstructured data on the web:***
  - Large document collections
  - Image databases, streaming media

1.2 Why not use file systems?

- ***Data redundancy and inconsistency***
  - Multiple file formats
  - Duplication of information in different files

- ***Difficulty in accessing data***
  - Need to write a new program to carry out each new task
- ***Data isolation***
  - Multiple files and formats
- ***Integrity problems***
  - Integrity constraints (e.g. account balance > 0) become part of program code
  - Hard to add new constraints or change existing ones
- ***Maintenance problems***
  - When we add a new field, all existing applications must be modified to ignore it
- ***Atomicity of updates***
  - Failures may leave database in an inconsistent state with partial updates carried out
  - E.g. transfer of funds from one account to another should either complete or not happen at all
- ***Concurrent access by multiple users***
  - Concurrent accessed needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
    - E.g. two people reading a balance and updating it at the same time
  - Security problems

Database systems offer solutions to all the above problems

Advantages of DBMS.

- Due to its centralized nature, the database system can overcome the disadvantages of the file system-based system

1. **Data independency**: Application program should not be exposed to details of data representation and storage DBMS provides the abstract view that hides these details.
2. **Efficient data access**: DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently.
3. **Data integrity and security**: Data is accessed through DBMS, it can enforce integrity constraints.

   E.g.: Inserting salary information for an employee.
4. **Data Administration**: When users share data, centralizing the data is an important task, Experience professionals can minimize data redundancy and perform fine tuning which reduces retrieval time.
5. **Concurrent access and Crash recovery:** DBMS schedules concurrent access to the data. DBMS protects user from the effects of system failure.

1.3 The Levels of Abstraction

View of Data:

- A major purpose of a database system is to provide users with an abstract view of the data. That is, the system hides certain details of how the data are stored and maintained.

1.3.1 Data Abstraction

- For the system to be usable, it must retrieve data efficiently. The need for efficiency has led designers to use complex data structures to represent data in the database. Since many

database-system users are not computer trained, developers hide the complexity from users through several levels of abstraction, to simplify users' interactions with the system:

Physical level:

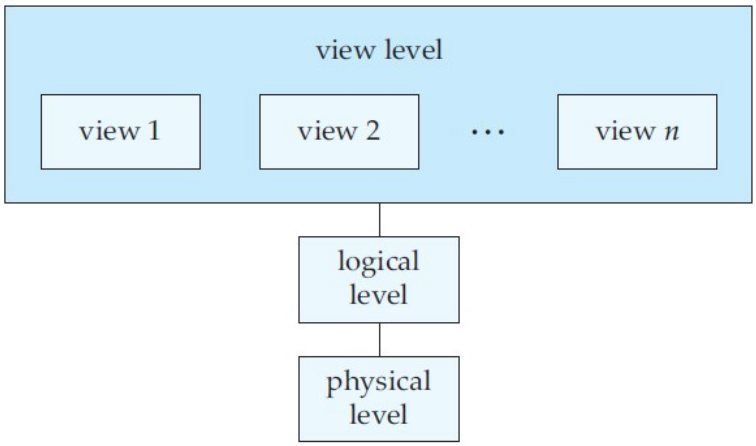
- The lowest level of abstraction describes how the data are actually stored. The physical level describes complex low-level data structures in detail.

Logical level:

The next-higher level of abstraction describes what data are stored in the database, and what relationships exist among those data. The logical level thus describes the entire database in terms of a small number of relatively simple structures. Although implementation of the simple structures at the logical level may involve complex physical-level structures, the user of the logical level does not need to be aware of this complexity. This is referred to as physical data independence. Database administrators, who must decide what information to keep in the database, use the logical level of abstraction.

View level:

The highest level of abstraction describes only part of the entire database. Even though the logical level uses simpler structures, complexity remains because of the variety of information stored in a large database. Many users of the database system do not need all this information; instead, they need to access only a part of the database. The view level of abstraction exists to simplify their interaction with the system. The system may provide many views for the same database.



1.4 Schemas *vs.* Instances

- ***Schema***
  - The logical structure of the database.
  - e.g., the database consists of information about a set of customers and accounts and the relationship between them.
  - Analogous to type information of a variable in a program.
- ***Instance***
  - the actual content of the database at a particular point in time.
  - Analogous to the value of a variable.
- ***Data Independence***
  - the ability to modify a schema in one-level (i.e. Internal Schema or Conceptual Schema) without affecting a schema in the next-higher-level (i.e. Conceptual or External schema)
  - Applications depend on the logical schema
  - Database engines take care of efficient storage and query processing

Data independence are of two types:

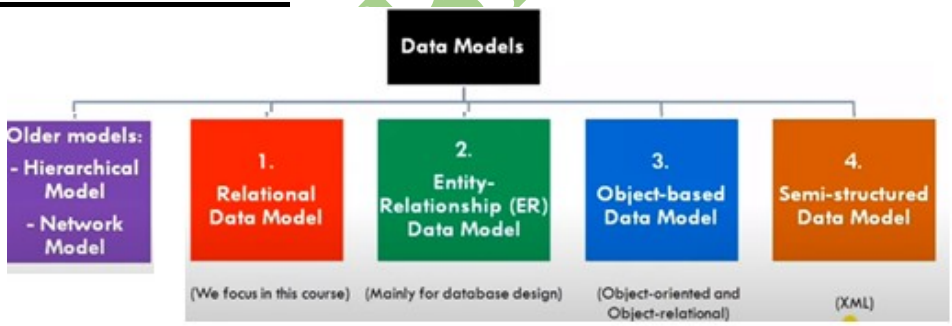
- Physical Data Independence: Physical data independence is the ability to modify the physical schema – i.e. internal schema which describes the physical storage devices or structure to store the data – without affecting the conceptual schema – application programs.
- Logical Data Independence: Logical data is the ability to modify the logical schema – i.e. Conceptual Schema, which decides what information is to be kept in the database – without affecting the next higher level schema – i.e., External Schema – application program.

1.5 Data Models in DBMS

Data Model gives us an idea that how the final system will look like after its complete implementation.

- A Data Model in DBMS is the concept of tools that are developed to summarize the description of the database.
- It defines how the logical structure of a database is modelled A Data Model is collection of conceptual tools for describing:
  - Data
  - Data relationships
  - Data semantics and
  - Consistency constraints
- It describes the design of a database at each level of data abstraction.
- It defines how data is connected to each other and how they are processed & stored inside the system

Types of Data Models

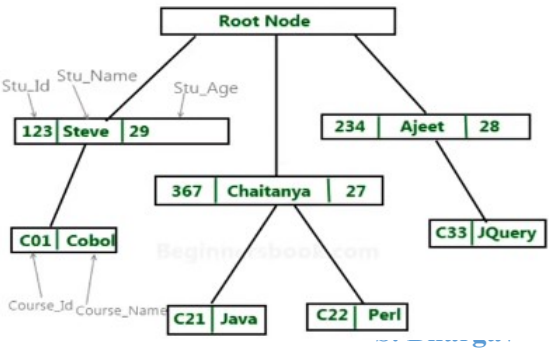


Older Data Models

- Hierarchical Data Model
- Network Data Model
  - Hierarchical Data Model and Network Data Model preceded the Relational data model
  - But today they are replaced by Relational data model

Hierarchical Data Model

- It was the first DBMS model.
- In hierarchical model, data is organized into a tree like structure with each record is having one parent record and many children.
- The main drawback of this model is that, it can have only one to many relationships between nodes.
- Hierarchical models are rarely used now.



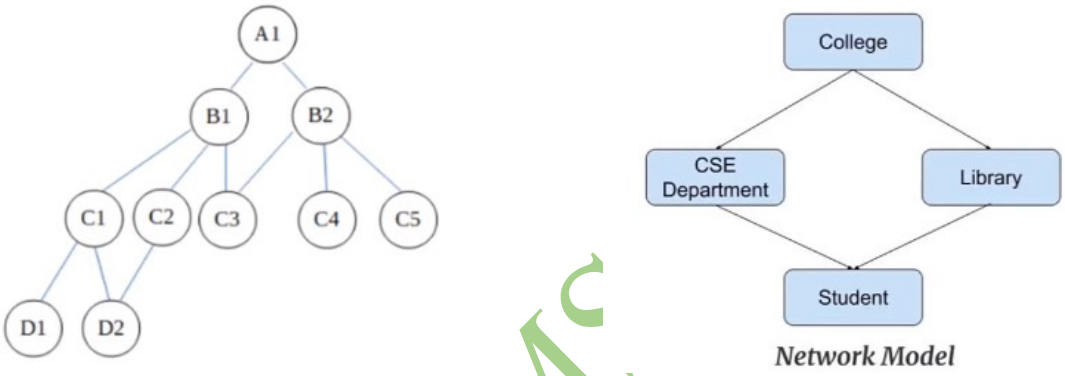
Example

- Let's we have few students and few courses:
  - a course can be assigned to a single student only,
  - however a student take any number of courses

  So this relationship becomes one to many.

Network Model

- This model is an extension of the hierarchical model. It was the most popular model before the relational model.
- Network Model is same as hierarchical model except that it has graph-like structure rather than a tree-based structure and are allowed to have more than one parent node.
- It supports many-to-many data relationships.
- This was the most widely used database model, before Relation model was introduced.

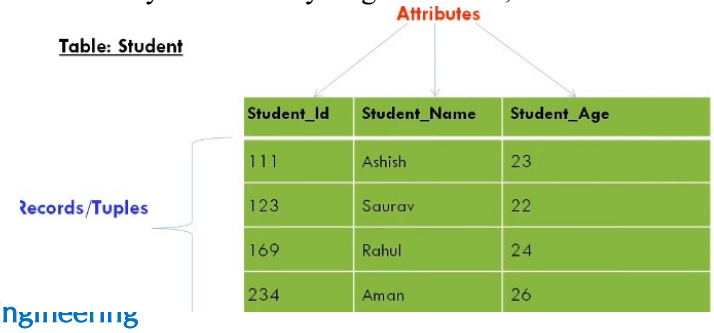


Network Model

Example: the node has two parent node i.e. CSE Department and Library.
This was earlier not possible in the hierarchical model.

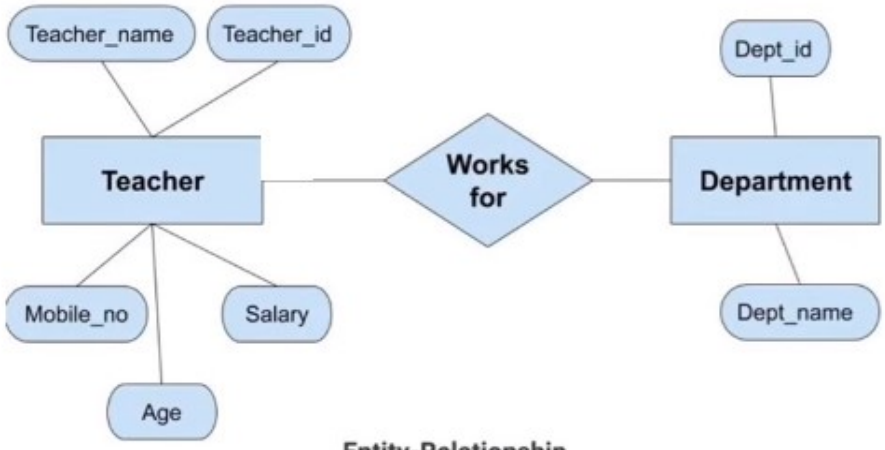
1. Relational Model

- Most widely used model by commercial data processing applications
- It uses collection of tables for representing data and the **relation**ships among those data
- Data is stored in tables called **Relation**s
- Each table is a group of column and rows, where column represents attribute of an entity and rows represents records (or tuples)
- Attribute or field: Each column in a **relation** is called an attribute. The values of the attribute should be from the same domain.
  - Example: we have different attributes of the student like Student_Id, Student _ Name, Student_Age, etc.
- Tuple or Record: Each row in the **relation** called tuple. A tuple defines a collection of attribute values. So each row in a **relation** contains unique values.
  - Example: each row has all the information about any specific individual like the first row has information about student Ashish.
- This model was initially described by Edgar F. Codd, in 1969.



2. Entity-Relationship (ER) Model

- ER Model is a high-level data model diagram
- ER model describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram).
- An ER model is a design or blueprint of a database that can later be implemented as a database.
- It is based on the notion of real-world entities and relationships among them.
- ER diagram has the following three components:
    - **Entities:** Entity is a real-world thing or object. It can be a person, place, or even a concept.
        - Example: Teachers, Students, Course, Building, Department, etc. are some of the entities of a School Management System.
    - **Attributes:** An entity contains a real-world property called attribute. This is the characteristics of that attribute.
        - Example: The entity teacher has the property like teacher id name, salary, age, etc.
    - **Relationship:** Relationship tells how two attributes are related.
        - Example: Teacher works for a department.

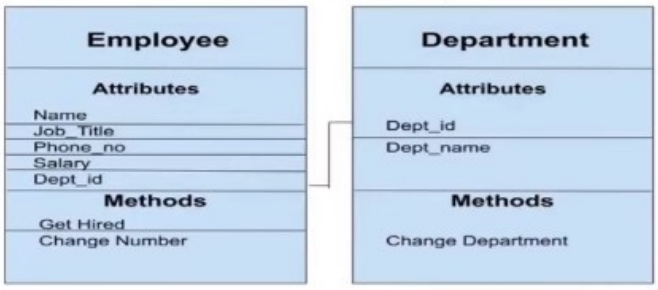


Entity-Relationship Model

    - The entities are Teacher and Department.
    - The attributes of Teacher entity are Teacher_Name, Teacher_id, Age, Salary, mobile_number.
    - The attributes of entity are Dept_id, Dept_name.
    - The two entities are connected using the relationship. Here, each teacher works for a department.

3. Object-based Data Model

- An extension of the ER model with notions of functions, encapsulation, and object identity, as well.
    - In this model, both the data and relationship are present in a single structure known as an object.
    - Two are more objects are connected through links. We use this link to relate one object to other objects.



Object_Oriented_Model

- Here, two objects Employee and Department.
- All the data and relationships of each object are contained as a single unit.
- The attributes like Name, Job_title Of the employee and the methods which will be performed by that Object are stored as a single Object
- The two objects are connected through a common attribute i.e. the Department_id and the communication between these two will be done with the help of this common id i.e. department_id.

4. Semi-structured Data model

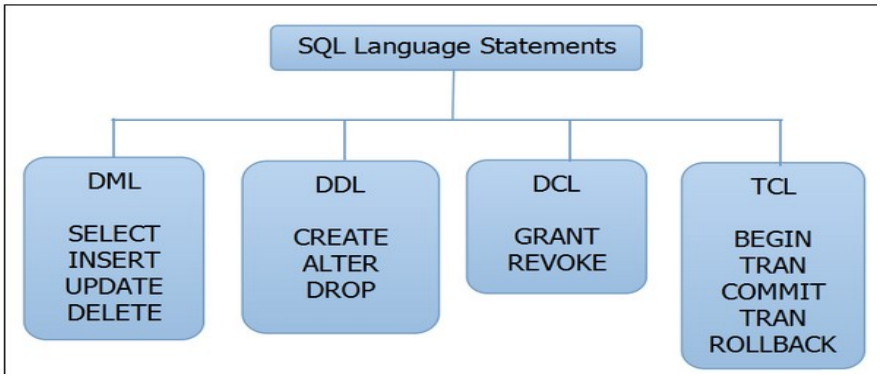
- Semi-structured model is an evolved form of the relational model.
- The semi-structured data model allows the data specifications at places where the individual data items of the same type may have different sets of attributes.
- In this model, some entities may have missing attributes while others may have an extra attribute.
- This model gives flexibility in storing the data. It also gives flexibility to the attributes.
    - Example: If we are storing any value in any attribute then that value can be either atomic value or a collection of values.
- The Extensible Markup Language (XML) is widely used for representing the semi-structured data.
    - In XML we can create use tags and use different mark ups to describe the data.

Example XML or JSON data

<student 1>
<Roll. No.>..... </Roll. No.>
<Name.>..... < /Name>
<Class.>..... < /Class>
<Age.>..... < / Age>
</student 1 >
<student 2 >
<Name.>..... < /Name>
<Class.>..... < /Class>
<Age.>..... < / Age>
</student 2 >

1.6 Database Languages:

- SQL language is divided into four types of primary language statements: DML, DDL, DCL and TCL. Using these statements, we can define the structure of a database by creating and altering database objects, and we can manipulate data in a table through updates or deletions. We can also control which user can read/write data or manage transactions to create a single unit of work.



Mainly two types of Database Languages:

- **Data Definition Language (DDL):** to specify database schema
- **Data Manipulation Language (DML):** to express database queries and updates

In practical, DDL and DML are not separate languages, instead they are the parts of a single database language such as widely use (Structured Query Language)

Data Definition Language (DDL)

DDL (stands for Data Definition Language) is used for specifying the database schema.

- used by the DBA and database designers to specify the conceptual schema of a database
- DDL is used for creating tables, schema, indexes, constraints etc. in database.
- Eg: create table account (
  Account_number char(10),
  balance integer)
- DDL is used to store the information of metadata like the number of tables and schemas, their names, indexes, columns in each table, constraints, etc.
  - DDL compiler generates a set of table templates stored in a data dictionary.
  - Data dictionary contains Metadata (i.e. data about data)
    - Database Schema
      - Table name
      - Column names & types
    - Integrity Constraints
      - Primary Key
    - Authorization
      - Who can access

DDL Commands in SQL:

- **CREATE:** Used to create the database instance
- **ALTER:** used to alter the structure of the database
- **DROP:** used to delete the database instance
- **TRUNCATE:** Used to remove all records from a table
- **RENAME:** used to rename database instances
  - **Example:** create department(
    dept _ name char(20),
    building char(5),
    budget numeric(12.2));

Note: These commands either defines or update the database schema that's why they come under Data Definition language.

Data Manipulation Language (DML):

- DML (stands or Data Manipulation Language) is used or accessing and manipulating data in CI database.
- It allows users to insert, update, delete and retrieve data from the database.
- DML is also known as Query Language
    - A query is a Statement requesting the retrieval Of information
- Two classes of DML languages:
    - Procedural DMLs/High-level DMLs:
        - User specifies what data is required and how to get those data
        - Procedural DML is embedded into a high-level programming language like javat etc.
        - Eg: PL/SQL
    - Declarative (Non-Procedural) DMLs/Low-level DMLs:
        - User specifies what data is require only; without specifying how to get those data
        - Declarative DMLs are usually easier to learn and use than procedural DMLs. However, since a user does not have to specify how to get the data, the database system has to figure out an efficient means of accessing data.
        - E :SQL

DML Commands in SQL:

- SELECT: used to retrieve data from a database.
- INSERT: used te insert data into a table.
- UPDATE: used to update existing data within a table.
- DELETE: used to delete all records from a table.
    - **Example: select student_name, Student_age from student;**

Data Control Language (DCL)

- **Data Control Language (DCL)** is used to control privilege in Databases i.e. DCL is used for granting and revoking user access on a database (Authorization)
- To perform any operation in the database, such as for creating tables or views, need privileges.
- In particular, it is a component of Structured Query Language (SQL)

DCL Commands:

- GRANT —To give user access privileges to a database
- REVOKE — To take back permissions from the user

- The operations for which privileges may be granted to or revoked from a user or role apply to both the Data definition language (DDL) and the Data manipulation language (DML).

TCL (Transaction Control Language)

- **TCL (Transaction Control Language)** commands are used to manage transactions in the database.
- TCL is used to run the changes made by the DML statement.
- The changes in the database that we made using DML commands are either performed or rollbacked using TCL.

  TCL Commands:
    - COMMIT: to save the transaction on the database
    - ROLLBACK to restore the database to original since the last Commit.
    - SAVEPOINT Save point command is Used to temporarily save a transaction so that YOU can rollback to that point whenever necessary.

Database Users

- Database Users are the persons who interact with the database and take the benefits of database.
- Users are differentiated by the way they expect to interact with the system.
- Four types of DB users:
  1. Naive users/Native users/End users
  2. Application programmers
  3. Sophisticated Users
  4. specialized users

Naive Users/Native Users/End Users: They are the unsophisticated users who use the existing application to interact with the database.

- Example: People accessing database over the web, bank tellers, clerical a staffs, etc.

Application Programmers: They are the computer professionals who write the application programs. They interact with system through DML queries.

- For example: Writing a C program to generate the report of employees who are working in particular department, will involve a query to fetch the data from database. It will include an embedded SQL query in the C Program.

Sophisticated Users: They interact with the system by writing SQL queries directly through the query processor (like SQL) without writing application programs.

- Example: Analysts, who submits SOL queries to explore data in the DBMS.

Specialized Users: They are also sophisticated users who write specialized database applications that do not fit into the traditional data processing framework. They are the developers who develop the complex programs to the requirement.

- Example: Computer-Aided Design (CAD) Systems, Expert Systems, Knowledge Based System, etc. that store complex data types (graphics and audio data) & environment modelling systems.

Database Administrators DBAs

- ✓ DBA is a person or group that is responsible for supervising both the database and the use of the DBMS
- ✓ Database Administrators (DBAs) coordinate all the activities of the database system.
- ✓ They use specialized software to store and organize data
- ✓ They have all the permissions

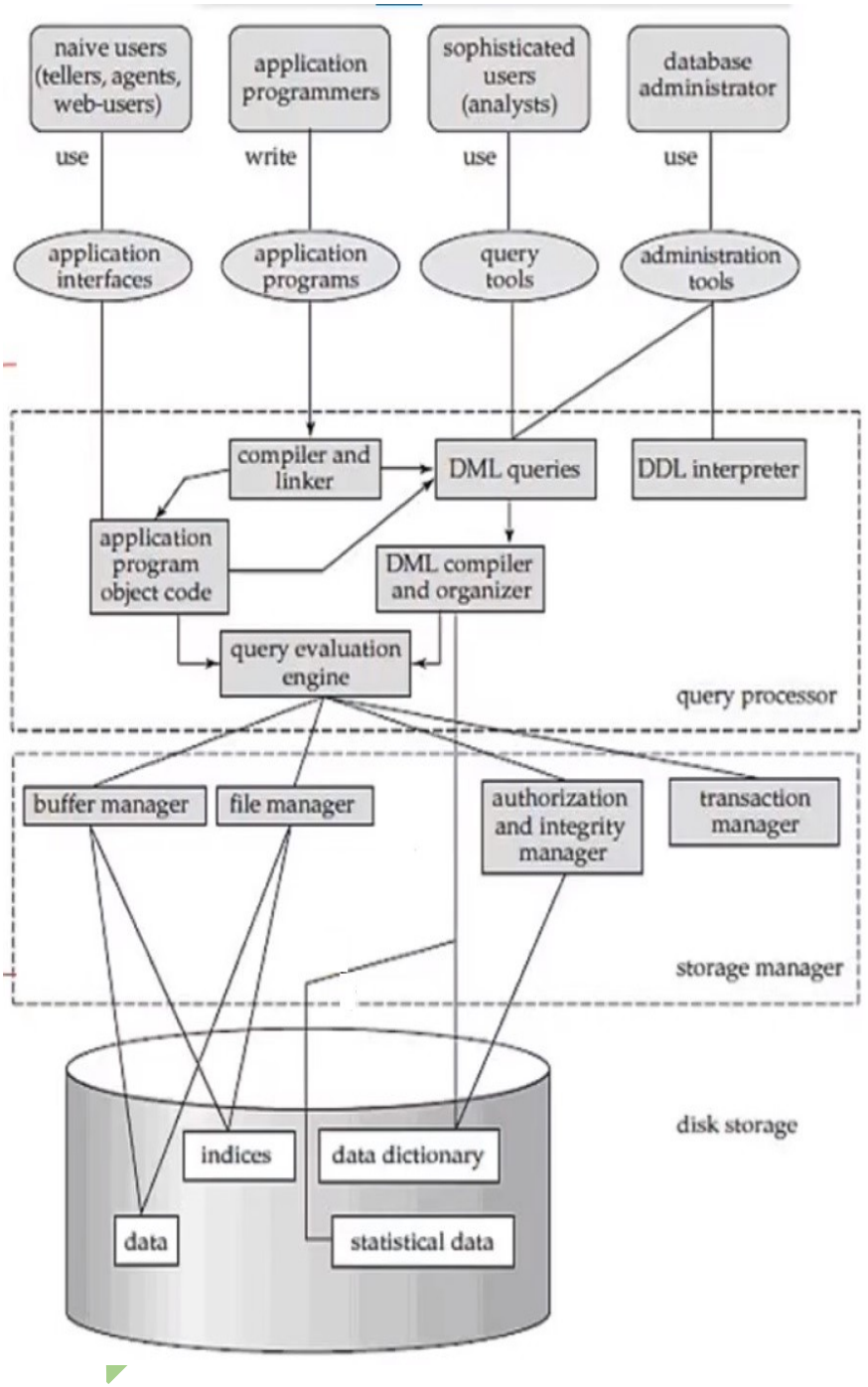
DBAs Tasks

- ✓ Schema definition
- ✓ Storage structure and access method definition
- ✓ Schema and physical organization modification
- ✓ Specifying integrity constraints
- ✓ Granting user authority to access database
- ✓ Monitoring performance & responding to changes in requirements
- ✓ Routine Maintenance
- ✓ Acting as liaison with users
- ✓ Backing up and restoring databases

Database System Structure

- Database System Structure is partitioned into modules for different functions.
- Some functions (e.g. file systems) may be provided by the operating system.

- The database system is divided into two functional components:
  1. Query Processor
  2. Storage Manager



1. The **Users** issues a queries using a particular database language like SQL

2. The **Query Processor** processes the queries or programs with the help of its components

3. The **Storage Manager** access the stored data from the **disk**

4. **DBMS** returns the results to the user

1. Query Processor

Query processor helps the database system, simplify and facilitate access to data.
The work of query processor is to execute the query successfully

- It interprets the requests (queries) received from end user via an application program into instructions
- It also executes the user request which is received from the DML compiler.

Query Processor components:

DDL Interpreter

- It interprets the DDL statements (like schema definition) into a set of table containing metadata (data about data) stored in a data dictionary

DML Compiler

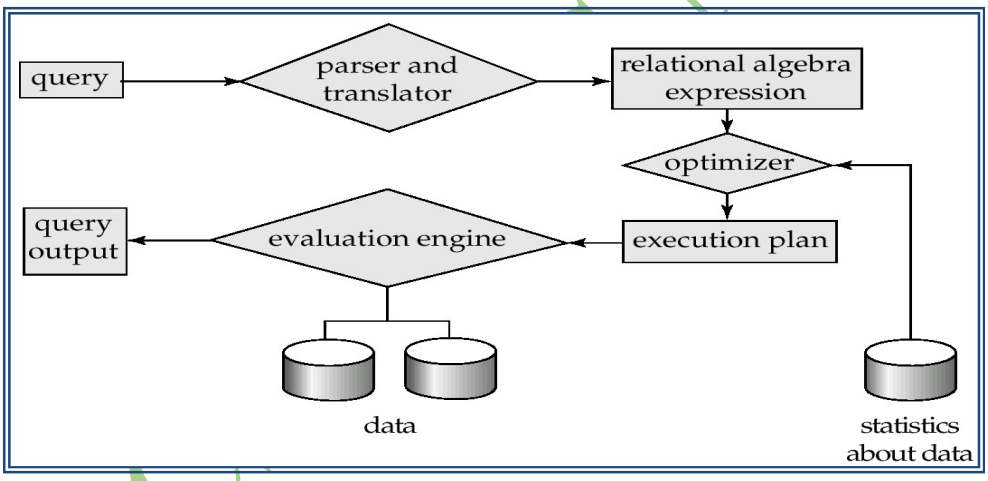
- It translates the DML statements (like select, insert, update, delete) into low level instruction (Object code), so that they can be executed.
    - A query can USUally be translated into any of a number of alternative evaluation plans that all give the same result. The DML compiler also performs query optimization, that is, it picks the lowest cost evaluation plan from among the alternatives.
    - Query optimization determines the efficient way to execute a query with different possible query plans.

Embedded DML Pre-compiler

- It processes DML statements embedded in an application program into procedural calls.

Query Evaluation Engine

- It executes the low level instruction (object code) generated by DML Compiler.



2. Storage Manager

- Storage Manager is a program that provides an interface between the data stored in the database and the queries received.
- The storage manager is responsible for the interaction with the file manager.
- The raw data are stored on the disk using the file system, which is usually provided by a conventional operating system.
- The storage manager translates the various DML statements into low-level file-system commands
- Thus, It is responsible for updating, storing, deleting, and retrieving data in the database
- It is also known as Database Control System. It maintains the consistency and integrity of the database by applying the constraints and executes the DCL statements.

Storage Manager Components:

Authorization Manager and Integrity Manager

- It checks the authority of users to access data and checks the integrity constraints when the database is modified.

Transaction Manager

- It ensures that the database remains in a consistent (correct) state despite system failures, and that concurrent transaction executions proceed without conflicting

File Manager

- It manages the allocation of space on disk storage and the data structures Used to represent information stored on disk

Buffer Manager

- It is responsible for fetching data from disk storage into main memory, and deciding what data to cache in main memory.

Disk Storage

- The Storage Manager implements the following data structures as part of the physical system implementation:

Data Files: It stores the database itself

Data Dictionary: It contains the information about the structure of any database. or, it stores metadata about the database, in particular the schema of the database.

Indices: It provides faster retrieval of data item.

Statistical Data: It stores statistical information about the data in the database. This information is used by the query processor to select efficient ways to execute a query.

➢ **Transaction Management**

- A *transaction* is a collection of operations that performs a single logical function in a database application
  - **E.g. transfer funds from one account to another**
- Transaction-management component ensures that the database remains in a consistent state despite system failures
- Concurrency-control manager controls the interaction among the concurrent transactions, to ensure the consistency of the database.
  - **E.g. simultaneous withdrawals**

- ACID Properties:

A - Atomicity / Accessing the Data
C - Concurrency Access
I - Integrity Problems / Inconsistency
D - Data Redundancy