II B.Sc Computer Science

18BCS42C – DATABASE MANAGEMENT SYSTEM

UNIT-I

BASIC CONCEPTS: Introduction to databases – Conventional file processing – Purpose of database system – Characteristics of database approach – Advantages of using DBMS – Database concept and architecture – Data Abstraction – Data Models – Instances and Schema – Data Independence – Schema Architecture – Components of a DBMS – Database Languages – Database Manager – Database Administrator – Database Users.

Introduction to databases

- A database management system (DBMS) refers to the technology for creating and managing databases.
- DBMS is a software tool to organize (create, retrieve, update, and manage) data in a database.
- The main aim of a DBMS is to supply a way to store up and retrieve database information that is both convenient and efficient.
- A datum is a unit of data.
- The name indicates what the database is.
- A database is one of the essential components for many applications and is used for storing a series of data in a single set.
- In other words, it is a group/package of information that is put in order so that it can be easily accessed, manage, and update.
- In a database, even the smallest portion of information becomes the data.
- There are different types of databases. They are:
    - Bibliographic
    - full-text
    - numeric
    - images

Conventional file processing

- File processing systems was an early attempt to computerize the manual filing system that we are all familiar with.
- A file system is a method for storing and organizing computer files and the data they contain to make it easy to find and access them.
- File systems may use a storage device such as a hard disk or CD-ROM and involve maintaining the physical location of the files.
- The manual filing system works well when the number of items to be stored is small.

- It even works quite adequately when there are large numbers of items and we have only to store and retrieve them.
- However, the manual filing system breaks down when we have to cross-reference or process the information in the files.

Characteristics of File Processing System

- Here is the list of some important characteristics of file processing system:
- It is a group of files storing data of an organization.
- Each file is independent from one another.
- Each file is called a flat file.
- Each file contained and processed information for one specific function, such as accounting or inventory.
- Files are designed by using programs written in programming languages such as COBOL, C, C++.The physical implementation and access procedures are written into database application; therefore, physical changes resulted in intensive rework on the part of the programmer.
- As systems became more complex, file processing systems offered little flexibility, presented many limitations, and were difficult to maintain.

Limitations of the File Processing System / File-Based Approach

1. Separated and Isolated Data: To make a decision, a user might need data from two separate files. First, the files were evaluated by analysts and programmers to determine the specific data required from each file and the relationships between the data and then applications could be written in a programming language to process and extract the needed data.

2. Duplication of data: Often the same information is stored in more than one file. Uncontrolled duplication of data is not required for several reasons. Duplication is wasteful. It costs time and money to enter the data more than once. It takes up additional storage space, again with associated costs. Duplication can lead to loss of data integrity; in other words the data is no longer consistent.

3. Data Dependence: In file processing systems, files and records were described by specific physical formats that were coded into the application program by programmers. If the format of a certain record was changed, the code in each file containing that format must be updated. Furthermore, instructions for data storage and access were written into the application's

code. Therefore, .changes in storage structure or access methods could greatly affect the processing or results of an application.

4. Difficulty in representing data from the user's view: To create useful applications for the user, often data from various files must be combined. In file processing it was difficult to determine relationships between isolated data in order to meet user requirements.

5. Data Inflexibility: Program-data inter dependency and data isolation, limited the flexibility of file processing systems in providing users with ad-hoc information requests

6. Incompatible file formats: As the structure of files is embedded in the application programs, the structures are dependent on the application programming language. For example, the structure of a file generated by a COBOL program may be different from the structure of a file generated by a 'C' program. The direct incompatibility of such files makes them difficult to process jointly.

7. Data Security. The security of data is low in file based system because, the data is maintained in the flat file(s) is easily accessible. For Example: Consider the Banking System. The Customer Transaction file has details about the total available balance of all customers. A Customer wants information about his account balance. In a file system it is difficult to give the Customer access to only his data in the· file.

8. Transactional Problems. The File based system approach does not satisfy transaction properties like Atomicity, Consistency, Isolation and Durability properties commonly known as ACID properties.

9. Concurrency problems. When multiple users access the same piece of data at same interval of time then it is called as concurrency of the system. When two or more users read the data simultaneously there is ll(problem, but when they like to update a file simultaneously, it may result in a problem.

10. Poor data modelling of real world. The file based system is not able to represent the complex data and interfile relationships, which results poor data modelling properties.

Purpose of database system

- In the early days, database applications were built on topof file systems.
- Drawbacks of using file systems to store data:
- 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
- Atomicity of updates
- Failures may leave database in an inconsistent state with partial updates carried out.
- Concurrent access by multipleusers
- Concurrent accessed needed for performance
- Uncontrolled concurrent accesses can lead to inconsistencies
- Security problems
- Database systems offer solutions to all the above problems

Characteristics of database approach

- **Real-world entity** − A modern DBMS is more realistic and uses real-world entities to design its architecture. It uses the behaviour and attributes too.
- **Relation-based tables** − DBMS allows entities and relations among them to form tables. A user can understand the architecture of a database just by looking at the table names.
- **Isolation of data and application** − A database system is entirely different than its data. A database is an active entity, whereas data is said to be passive, on which the database works and organizes. DBMS also stores metadata, which is data about data, to ease its own process.
- **Less redundancy** − DBMS follows the rules of normalization, which splits a relation when any of its attributes is having redundancy in values. Normalization is a mathematically rich and scientific process that reduces data redundancy.
- **Consistency** − Consistency is a state where every relation in a database remains consistent. There exist methods and techniques, which can detect attempt of leaving database in inconsistent state.
- **Query Language** − DBMS is equipped with query language, which makes it more efficient to retrieve and manipulate data. A user can apply as many and as different filtering options as required to retrieve a set of data.

- **ACID Properties** − DBMS follows the concepts of **A**tomicity, **C**onsistency, **I**solation and **D**urability (normally shortened as ACID). These concepts are applied on transactions, which manipulate data in a database. ACID properties help the database stay healthy in multi-transactional environments and in case of failure.
- **Multiuser and Concurrent Access** − DBMS supports multi-user environment and allows them to access and manipulate data in parallel.
- **Multiple views** – Multiple view features enables the users to have a concentrate view of the database according to their requirements.
- **Security** − Features like multiple views offer security to some extent where users are unable to access data of other users and departments. DBMS offers methods to impose constraints while entering data into the database and retrieving the same at a later stage. DBMS offers many different levels of security features, which enables multiple users to have different views with different features.

Advantages of using DBMS

- **Data independence:** Application programs should be as free or independent as possible from details of data representation and storage. DBMS can supply an abstract view of the data for insulating application code from such facts.
- **Efficient data access:** DBMS utilizes a mixture of sophisticated concepts and techniques for storing and retrieving data competently. This feature becomes important in cases where the data is stored on external storage devices.
- **Data integrity and security:** If data is accessed through the DBMS, the DBMS can enforce integrity constraints on the data.
- **Data administration:** When several users share the data, integrating the administration of data can offer significant improvements.

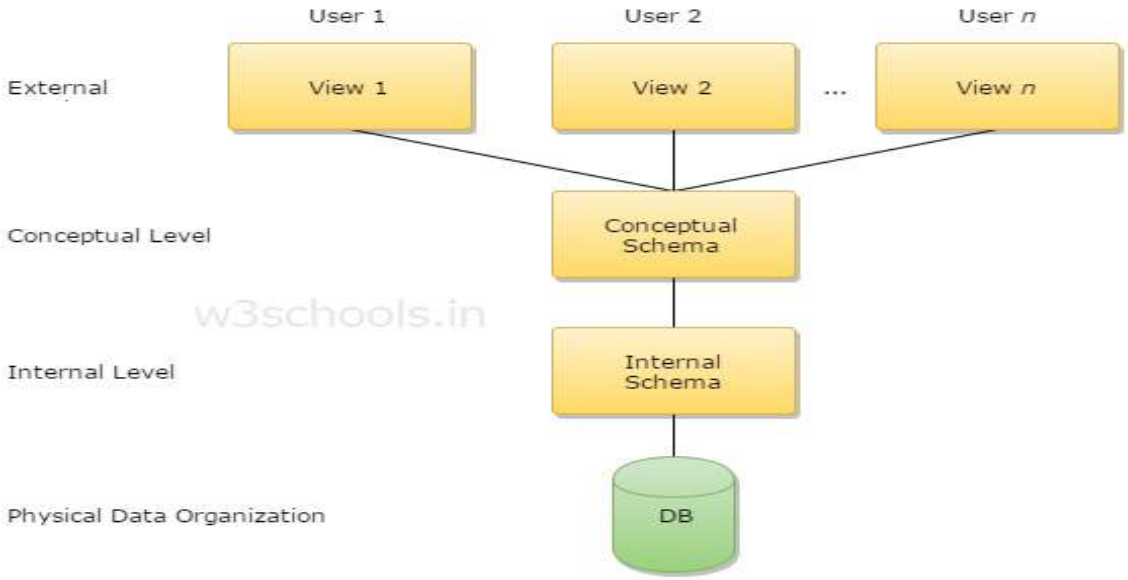
Database concept and architecture

- An early proposal for a standard terminology and general architecture for database systems was produced in 1971 by the DBTG (Data Base Task Group) appointed by the Conference on Data Systems and Languages (CODASYL, 1971).
- The DBTG recognized the need for a two level approach with a system view called the schema and user views called sub schema.

- The American National Standards Institute (ANSI) Standards Planning and Requirements Committee (SPARC) produced a similar terminology mid architecture in 1975 (ANSI 1975). ANSI-SPARC recognized the need for a three level approach with a system catalog.

There are following three levels or layers of DBMS architecture:

- External Level
- Conceptual Level
- Internal Level



ANSI_SPARC Architecture of the database system

What is Database Architecture?

- DBMS architecture helps in design, development, implementation, and maintenance of a database.
- A database stores critical information for a business. Selecting the correct Database Architecture helps in quick and secure access to this data.

1 tier Architecture

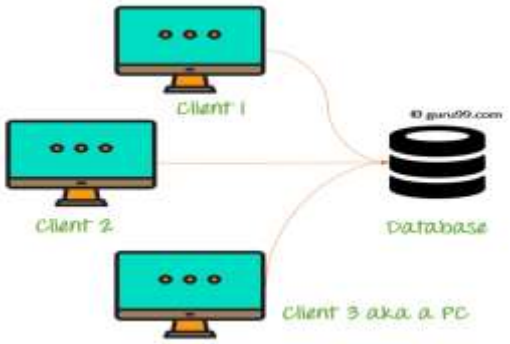
- The simplest of Database Architecture are **1 tier** where the Client, Server, and Database all reside on the same machine.
- Anytime you install a DB in your system and access it to practise SQL queries it is 1 tier architecture.
- But such architecture is rarely used in production.

2-tier Architecture

A two-tier architecture is a database architecture where

1. Presentation layer runs on a client (PC, Mobile, Tablet, etc)
2. Data is stored on a Server.

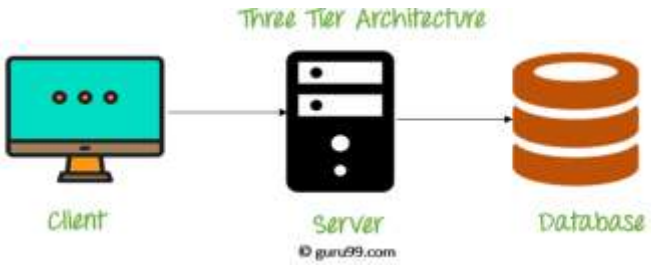
    - An application interface which is called ODBC (Open Database Connectivity) an API which allows the client-side program to call the DBMS.
    - Today most of the DBMS offers ODBC drivers for their DBMS. 2 tier architecture provides added security to the DBMS as it is not exposed to the end user directly.



3-tier Architecture

3-tier schema is an extension of the 2-tier architecture. 3-tier architecture has following layers

1. Presentation layer (your PC, Tablet, Mobile, etc.)
2. Application layer (server)
3. Database Server



This DBMS architecture contains an Application layer between the user and the DBMS, which is responsible for communicating the user's request to the DBMS system and send the response from the DBMS to the user.

The application layer (business logic layer) also processes functional logic, constraint, and rules before passing data to the user or down to the DBMS

3 tier architecture is the most popular DBMS architecture.

The goal of Three-tier architecture is:

- To separate the user applications and physical database
- Proposed to support DBMS characteristics
- Program-data independence
- Support of multiple views of the data

Data Abstraction

- Physical level describes how a record (e.g., customer) is stored.
- Logical level: describes data stored in database, and the relationships among thedata.
    - **type** customer =**record**
    - *name* : string; *street* : string; *city* : integer;
    - **end**;
- View level: application programs hide details of data types. Views can also hide information (e.g., salary) for security purposes.

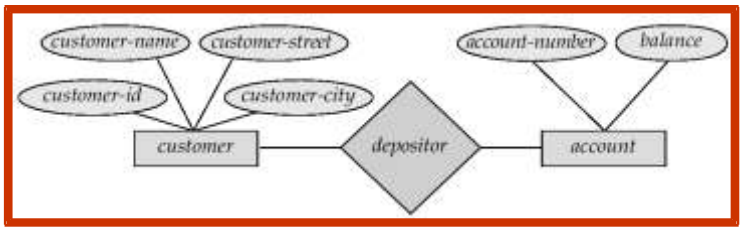
Data Models

- A collection of tools for describing
    - Data
    - data relationships
    - data semantics
    - data constraints

- Entity-Relationship model
- Relational model
- Other models:
    - object-oriented model
    - semi-structured data models
    - Older models: network model and hierarchical model

Entity-Relationship model

Example of schema in the entity-relationshipmodel



- E-R model of real world
    - Entities (objects)
        - E.g. customers, accounts, bank branch
- Relationships between entities
    - E.g. Account A-101 is held by customer Johnson
    - Relationship set *depositor* associates customers with accounts
- Widely used for database design
    - Database design in E-R model usually converted to design in the relational model (coming up next) which is used for storage and processing

Relational Model

- Example of tabular data in the relationalmodel

Attributes

Customer-Id	Customer-Name	Customer-Street	Customer-City	Account-Number
192-83-7465	Johnson	Alma	Palo Alto	A-101
019-28-3746	Smith	North	Rye	A-215
192-83-7465	Johnson	Alma	Palo Alto	A-201
321-12-3123	Jones	Main	Harrison	A-217
019-28-3746	Smith	North	Rye	A-201

customer-id	*customer-name*	*customer-street*	*customer-city*
192-83-7465	Johnson	12 Alma St.	Palo Alto
019-28-3746	Smith	4 North St.	Rye
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsay	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

(a) The *customer* table

account-number	*balance*
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

(b) The *account* table

customer-id	*account-number*
192-83-7465	A-101
192-83-7465	A-201
019-28-3746	A-215
677-89-9011	A-102
182-73-6091	A-305
321-12-3123	A-217
336-66-9999	A-222
019-28-3746	A-201

(c) The *depositor* table

Instances and Schemas

- Similar to types and variables in programming languages
- **Schema** – the logical structure of thedatabase
  - 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
  - **Physical schema**: database design at the physical level
  - **Logical schema**: database design at the logical level
- **Instance** – the actual content of the database at a particular point intime
  - Analogous to the value of a variable
- **Physical Data Independence** – the ability to modify the physical schema without changing the logicalschema
  - Applications depend on the logical schema
  - In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.

Data Independence

- Data Independence is defined as a property of DBMS that helps you to change the Database schema at one level of a database system without requiring to change the schema at the next higher level.

- Data independence helps you to keep data separated from all programs that make use of it.
- Data independence is an essential function for components of the system.

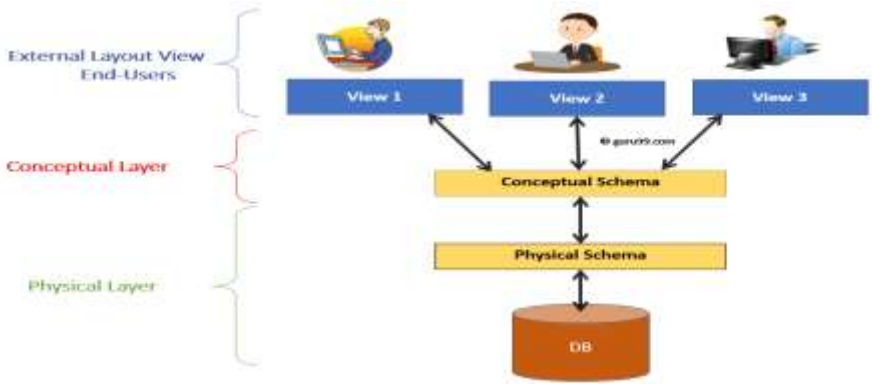
Types of Data Independence

In DBMS there are two types of data independence

1. Physical data independence
2. Logical data independence.

Levels of Database

1. Physical/Internal
2. Conceptual
3. External



Levels of DBMS Architecture Diagram

Physical Data Independence

- Physical data independence helps you to separate conceptual levels from the internal/physical levels.
- It allows you to provide a logical description of the database without the need to specify physical structures.
- Compared to Logical Independence, it is easy to achieve physical data independence.
- With Physical independence, you can easily change the physical storage structures or devices with an effect on the conceptual schema.
- Any change done would be absorbed by the mapping between the conceptual and internal levels.

- Physical data independence is achieved by the presence of the internal level of the database and then the transformation from the conceptual level of the database to the internal level.

Logical Data Independence

- Logical Data Independence is the ability to change the conceptual scheme without changing
    1. External views
    2. External API or programs
- Any change made will be absorbed by the mapping between external and conceptual levels.
- When compared to Physical Data independence, it is challenging to achieve logical data independence.

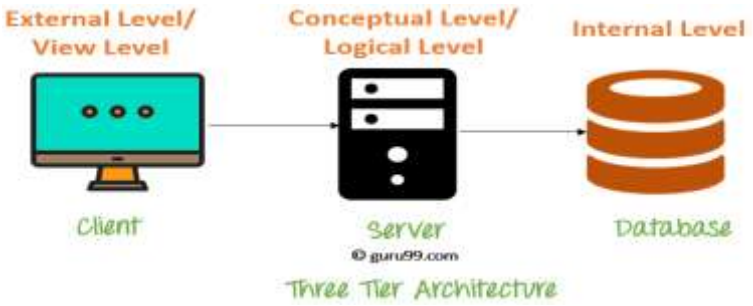
Importance of Data Independence

- Helps you to improve the quality of the data
- Database system maintenance becomes affordable
- Enforcement of standards and improvement in database security
- You don't need to alter data structure in application programs
- Permit developers to focus on the general structure of the Database rather than worrying about the internal implementation
- It allows you to improve state which is undamaged or undivided
- Database incongruity is vastly reduced.
- Easily make modifications in the physical level is needed to improve the performance of the system.

Schema Architecture

- Database systems comprise of complex data structures.
- Thus, to make the system efficient for retrieval of data and reduce the complexity of the users, developers use the method of Data Abstraction.
- There are mainly three levels of data abstraction:

    1. Internal Level: Actual PHYSICAL storage structure and access paths.
    2. Conceptual or Logical Level: Structure and constraints for the entire database
    3. External or View level: Describes various user views



Internal Level/Schema

- The internal schema defines the physical storage structure of the database.
- The internal schema is a very low-level representation of the entire database.
- It contains multiple occurrences of multiple types of internal record. In the ANSI term, it is also called "stored record'.

Facts about Internal schema:

- The internal schema is the lowest level of data abstraction
- It helps you to keeps information about the actual representation of the entire database. Like the actual storage of the data on the disk in the form of records
- The internal view tells us what data is stored in the database and how
- It never deals with the physical devices. Instead, internal schema views a physical device as a collection of physical pages

Conceptual Schema/Level

- The conceptual schema describes the Database structure of the whole database for the community of users.
- This schema hides information about the physical storage structures and focuses on describing data types, entities, relationships, etc.
- This logical level comes between the user level and physical storage view.
- However, there is only single conceptual view of a single database.

Facts about Conceptual schema:

- Defines all database entities, their attributes, and their relationships
- Security and integrity information

- In the conceptual level, the data available to a user must be contained in or derivable from the physical level

External Schema/Level

- An external schema describes the part of the database which specific user is interested in.
- It hides the unrelated details of the database from the user.
- There may be "n" number of external views for each database.
- Each external view is defined using an external schema, which consists of definitions of various types of external record of that specific view.
- An external view is just the content of the database as it is seen by some specific particular user.
- For example, a user from the sales department will see only sales related data.

Facts about external schema:

- An external level is only related to the data which is viewed by specific end users.
- This level includes some external schemas.
- External schema level is nearest to the user
- The external schema describes the segment of the database which is needed for a certain user group and hides the remaining details from the database from the specific user group

Goal of 3 level/schema of Database

Here, are some Objectives of using Three schema Architecture:

- Every user should be able to access the same data but able to see a customized view of the data.
- The users need not to deal directly with physical database storage detail.
- The DBA should be able to change the database storage structure without disturbing the user's views
- The internal structure of the database should remain unaffected when changes made to the physical aspects of storage.

Advantages Database Schema

- You can manage data independent of the physical storage

- Faster Migration to new graphical environments
- DBMS Architecture allows you to make changes on the presentation level without affecting the other two layers
- As each tier is separate, it is possible to use different sets of developers
- It is more secure as the client doesn't have direct access to the database business logic
- In case of the failure of the one-tier no data loss as you are always secure by accessing the other tier

Disadvantages Database Schema

- Complete DB Schema is a complex structure which is difficult to understand for every one
- Difficult to set up and maintain
- The physical separation of the tiers can affect the performance of the Database

Database Languages

- Database languages are used to read, update and store data in a database.
- There are several such languages that can be used for this purpose; one of them is SQL (Structured Query Language).
    - **Data Definition Language (DDL)**
    - **Data Manipulation Language (DML)**
    - **Structured Query Language (SQL)**
        - **Data Control language (DCL)**
        - **Transaction Control Language(TCL)**

Data Definition Language (DDL)

- DDL is used for specifying the database schema.
- It is used for creating tables, schema, indexes, constraints etc. in database.
    - To create the database instance – CREATE
    - To alter the structure of database – **ALTER**
    - To drop database instances – DROP
    - To delete tables in a database instance – **TRUNCATE**
    - To rename database instances – **RENAME**
    - To drop objects from database such as tables – **DROP**
    - To Comment – **Comment**

- Specification notation for defining the database schema
    - E.g.
      create table *account* (
      account-number **char**(10),
      balance **integer**)
- DDL compiler generates a set of tables stored in a *data dictionary*
- Data dictionary contains metadata (i.e., data about data)
    - database schema
- Data *storage and definition* language
    - Language in which the storage structure and access methods used by the database system are specified
    - Usually an extension of the data definition language

Data Manipulation Language (DML)

- DML is used for accessing and manipulating data in a database.
- The following operations on database comes under DML:
    - To read records from table(s) – SELECT
    - To insert record(s) into the table(s) – **INSERT**
    - Update the data in table(s) – UPDATE
    - Delete all the records from the table – DELETE
- Language for accessing and manipulating the data organized by the appropriate data model
- DML also known as query language
- Two classes of languages
    - Procedural – user specifies what data is required and how to get those data
    - Nonprocedural – user specifies what data is required without specifying how to get those data

Structured Query Language (SQL)

- SQL: widely used non-procedural language
- E.g. find the name of the customer with customer-id 192-83-7465
  select *customer.customer-name*
  from *customer*
  where *customer.customer-id* = '192-83-7465'
- E.g. find the balances of all accounts held by the customer with customer-id 192-83-7465
  select *account.balance*
  from *depositor*, *account*
  where *depositor.customer-id* = '192-83-7465' **and**
  depositor.account-number =account.account-number

- Application programs generally access databases through one of
    - Language extensions to allow embedded SQL
    - Application program interface (e.g. ODBC/JDBC) which allow SQL queries to be sent to a database

Data Control language (DCL)

- DCL is used for granting and revoking user access on a database –
    - To grant access to user – GRANT
    - To revoke access from user – REVOKE

Transaction Control Language(TCL)

- The changes in the database that we made using DML commands are either performed or rollback using TCL.
    - To persist the changes made by DML commands in database – COMMIT
    - To rollback the changes made to the database – ROLLBACK

Database Manager

- Transaction Management
- Storage Management

Transaction Management

- A *transaction* is a collection of operations that performs a single logical function in a database application.
- Transaction-management component ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.
- Concurrency-control manager controls the interaction among the concurrent transactions, to ensure the consistency of the database.

Storage Management

- Storage manager is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.
- The storage manager is responsible to the following tasks:
    - Interaction with the file manager
    - Efficient storing, retrieving and updating of data

Database Administrator

- Coordinates all the activities of the database system; the database administrator has a good understanding of the enterprise's information resources and needs.

- Database administrator's duties include:
    - Schema definition
    - Storage structure and access method definition
    - Schema and physical organization modification
    - Granting user authority to access the database
    - Specifying integrity constraints
    - Acting as liaison with users
    - Monitoring performance and responding to changes in requirements

Database User

- Users are differentiated by the way they expect to interact with the system
- Application programmers – interact with system through DML calls
- Sophisticated users – form requests in a database query language
- Specialized users – write specialized database applications that do not fit into the traditional data processing framework
- Naïve users – invoke one of the permanent application programs that have been written previously
    - E.g. people accessing database over the web, bank tellers,clerical staff

REFERENCES

1. Silberschatz A., Korth H. and Sudarshan S., "Database System Concepts", Tata McGraw Hill, 2011.
2. Elmasri R. and Navathe S.B., "Fundamentals of Database Systems", Pearson Education, 2016.
3. www.w3schools.com
4. www.guru99.com
5. www.tutorialspoint.com
6. www.beginnersbook.com
7. www.ecomputernotes.com

----------**********----------