Database Design & Applications

Introduction

Objectives

At the end of this module, you will be able to understand:

- DBMS Fundamentals
- DBMS Concepts
- DBMS Architecture
- Abstraction & Three-schema architecture
- Basic modules of DBMS
- DBMS Users
- Data Modeling

What is Data?

- Data represents recordable facts.
- Data aids in producing information, which is based on facts.
- For example, if we have data about marks obtained by all students, we can then conclude about toppers and average marks.

What is a Database?

- **Database** is a collection of related data and data is a collection of facts and figures that can be processed to produce information.
- It is an organized collection of structured information, or data, typically stored electronically.
- A Database is usually controlled and managed by a Database Management System

Applications of Databases

Typical examples of us having and using information from Databases are:

- **Banking:** all transactions
- **Airlines:** reservations, schedules
- **Universities:** registration, grades
- **Sales:** customers, products, purchases, Manufacturing: production, inventory, orders, supply chain
- **Human resources:** employee records, salaries, tax deductions

Databases touch all aspects of our daily lives

Single and Multi-User Systems

1. Single User Database Systems: In this DBMS, at one time, only a single user can access the database. Hence, the user can use all the resources at all times. All these systems are used for personal usage, such as personal computer experience. In this type of DBMS, both the physical and application layer can be used by the user.

Example: Personal Computer DBMS

2. Multi-User Database Systems: These DBMSs support two or more two users accessing the database simultaneously. Multi-user systems contain all the mini-computers and mainframe computers. In a mainframe computer, the database may exist on a single computer, and in other computers, the database may be distributed on multiple computers. Multiple users can update data while working together simultaneously.

Database Management System (DBMS)

- A Database Management System (DBMS) is used to Manage and Control a Database or set of Databases
- A Database Management System stores data in such a way that it becomes easier to retrieve, manipulate, and produce information.
- It usually consists of set of programs to manage the data
- DBMS provides an environment that is both convenient and efficient to use.

DBMS - Characteristics

Traditionally, data was organized in file formats. DBMS was a new concept then, and all the research was done to make it overcome the deficiencies in traditional style of data management. A modern DBMS has the following characteristics:

- **Real-world entity:** A modern DBMS is more realistic and uses real-world entities to design its architecture. It uses the behavior and attributes too. For example, a school database may use students as an entity and their age as an attribute.

- **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.

DBMS - Characteristics

- **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. A DBMS can provide greater consistency as compared to earlier forms of data storing applications like file-processing systems.

- **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. Traditionally it was not possible where file-processing system was used.

DBMS - Characteristics

- **ACID Properties:** DBMS follows the concepts of ACID, which stands for:
  Atomicity, Consistency, Isolation, and Durability
  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.

- **Multi-user and Concurrent Access:** DBMS supports multi-user environment and allows them to access and manipulate data in parallel. Though there are restrictions on transactions when users attempt to handle the same data item, but users are always unaware of them.

- **Multiple views:** DBMS offers multiple views for different users. A user who is in the Sales department will have a different view of database than a person working in the Production department. This feature enables the users to have a concentrate view of the database according to their requirements.

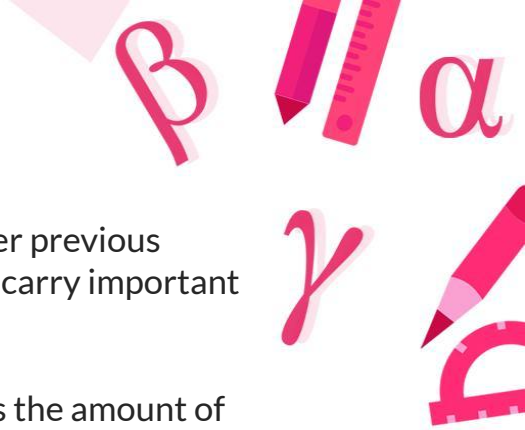
DBMS - Characteristics

- **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. For example, a user in the Sales department cannot see the data that belongs to the Purchase department. Additionally, it can also be managed how much data of the Sales department should be displayed to the user. Since a DBMS is not saved on the disk as traditional file systems, it is very hard for miscreants to break the code.

DBMS Advantages

- The DBMS provides as the broker between the user and the data source.
- The DBMS gets all program demands and converts them into the complex functions required to meet up with those demands.

DBMS Disadvantages



- Although the database program results in considerable benefits over previous information control approaches, information source techniques do carry important disadvantages.
- Databases Management system can sometimes become complex as the amount of data grows, and the relations between pieces of data become more complicated.
- Complex database systems may lead to isolated databases where the information cannot be shared from one system to another

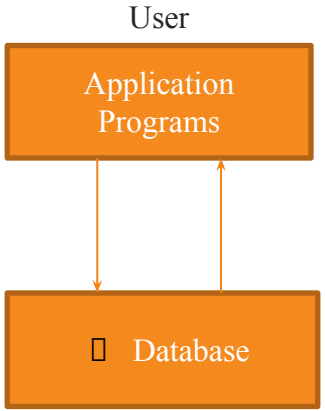
DBMS Architecture

- The design of a DBMS depends on its architecture. It can be centralized or decentralized or hierarchical. The architecture of a DBMS can be seen as either single tier or multi-tier. An n-tier architecture divides the whole system into related but independent n modules, which can be independently modified, altered, changed, or replaced.

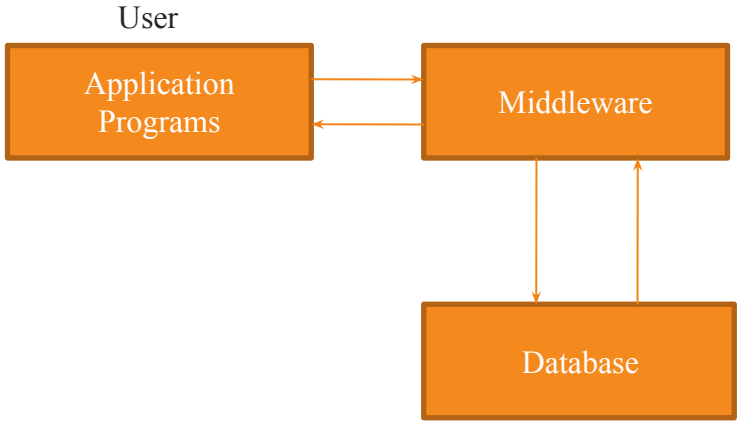
- In **1-tier architecture**, the DBMS is the only entity where the user directly sits on the DBMS and uses it. Any changes done here will directly be done on the DBMS itself. It does not provide handy tools for end-users. Database designers and programmers normally prefer to use single-tier architecture.

- If the architecture of DBMS **is 2-tier**, then it must have an application through which the DBMS can be accessed. Programmers use 2-tier architecture where they access the DBMS by means of an application. Here the application tier is entirely independent of the database in terms of operation, design, and programming.

DBMS Architecture



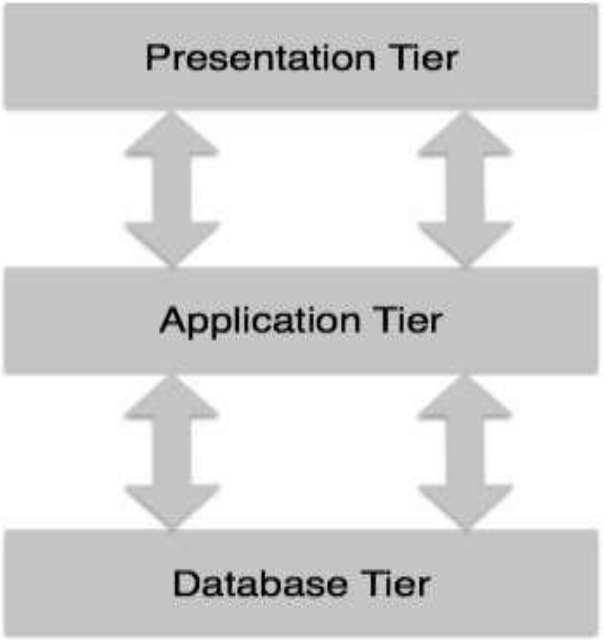
Two-tier Architecture



Three-tier Architecture

3-tier Architecture

A 3-tier architecture separates its tiers from each other based on the complexity of the users and how they use the data present in the database. It is the most widely used architecture to design a DBMS.



3-tier Architecture

- **Database (Data) Tier:** At this tier, the database resides along with its query processing languages. We also have the relations that define the data and their constraints at this level.

- **Application (Middle) Tier:** At this tier reside the application server and the programs that access the database. For a user, this application tier presents an abstracted view of the database. End-users are unaware of any existence of the database beyond the application. At the other end, the database tier is not aware of any other user beyond the application tier. Hence, the application layer sits in the middle and acts as a mediator between the end-user and the database.

- **User (Presentation) Tier:** End-users operate on this tier and they know nothing about any existence of the database beyond this layer. At this layer, multiple views of the database can be provided by the application. All views are generated by applications that reside in the application tier.

Multiple-tier database architecture is highly modifiable, as almost all its components are independent and can be changed independently.

Types of DBMS

- Hierarchical databases.
- Network databases.
- Relational databases.
- Object-oriented databases.
- NoSQL databases.

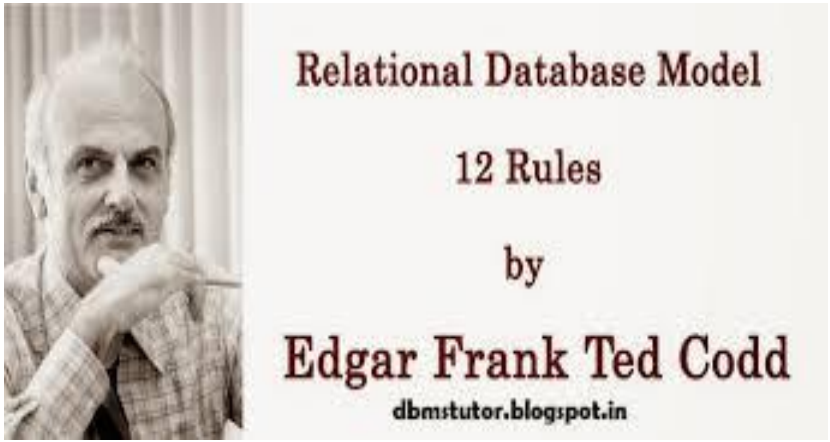
Relational DBMS (DBMS)

A relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model as invented by E. F. Codd

Basic Functions of RDBMS

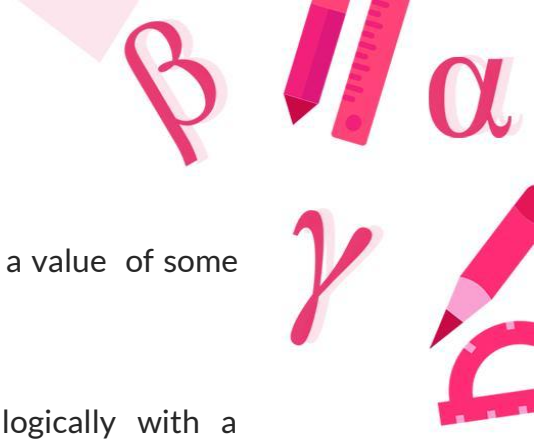
1. Stores the data
2. Processes the data
3. Secure the data
4. Validates the Data

CODD'S 12 RULES



- **Dr Edgar F. Codd**, after his extensive research on the Relational Model of database systems, came up with twelve rules of his own, which according to him, a database must obey in order to be regarded as a true relational database.
- These rules can be applied on any database system that manages stored data using only its relational capabilities.
- This is a foundation rule, which acts as a base for all the other rules.

CODD'S 12 RULES



Rule 1 - Information Rule

The data stored in a database, may it be user data or metadata, must be a value of some table cell. Everything in a database must be stored in a table format.

Rule 2 - Guaranteed Access Rule

Every single data element (value) is guaranteed to be accessible logically with a combination of table-name, primary-key (row value), and attribute-name (column value). No other means, such as pointers, can be used to access data.

Rule 3 - Systematic Treatment of NULL Values

The NULL values in a database must be given a systematic and uniform treatment. This is a very important rule because a NULL can be interpreted as one the following: data is missing, data is not known, or data is not applicable.

Rule 4 - Active Online Catalog

The structure description of the entire database must be stored in an online catalog, known as **data dictionary**, which can be accessed by authorized users. Users can use the same query language to access the catalog which they use to access the database itself.

CODD'S 12 RULES

Rule 5 - Comprehensive Data Sub- Language Rule

A database can only be accessed using a language having linear syntax that supports data definition, data manipulation, and transaction management operations. This language can be used directly or by means of some application. If the database allows access to data without any help of this language, then it is considered as a violation.

Rule 6 - View Updating Rule

All views of a database, which can theoretically be updated, must also be updatable by the system.

Rule 7 – High -Level Insert, Update & Delete Rule

A database must support high-level insertion, updation, and deletion. This must not be limited to a single row, that is, it must also support union, intersection and minus operations to yield sets of data records.

Rule 8 - Physical Data Independence

The data stored in a database must be independent of the applications that access the database.

Any change in the physical structure of a database must not have any impact on how the data is being accessed by external applications.

CODD'S 12 RULES

Rule 9 – Logical Data Independence

The logical data in a database must be independent of its user's view (application). Any change in logical data must not affect the applications using it. For example, if two tables are merged or one is split into two different tables, there should be no impact or change on the user application. This is one of the most difficult rule to apply.

Rule 10 - Integrity Independence Rule

A database must be independent of the application that uses it. All its integrity constraints can be independently modified without the need of any change in the application. This rule makes a database independent of the front-end application and its interface.

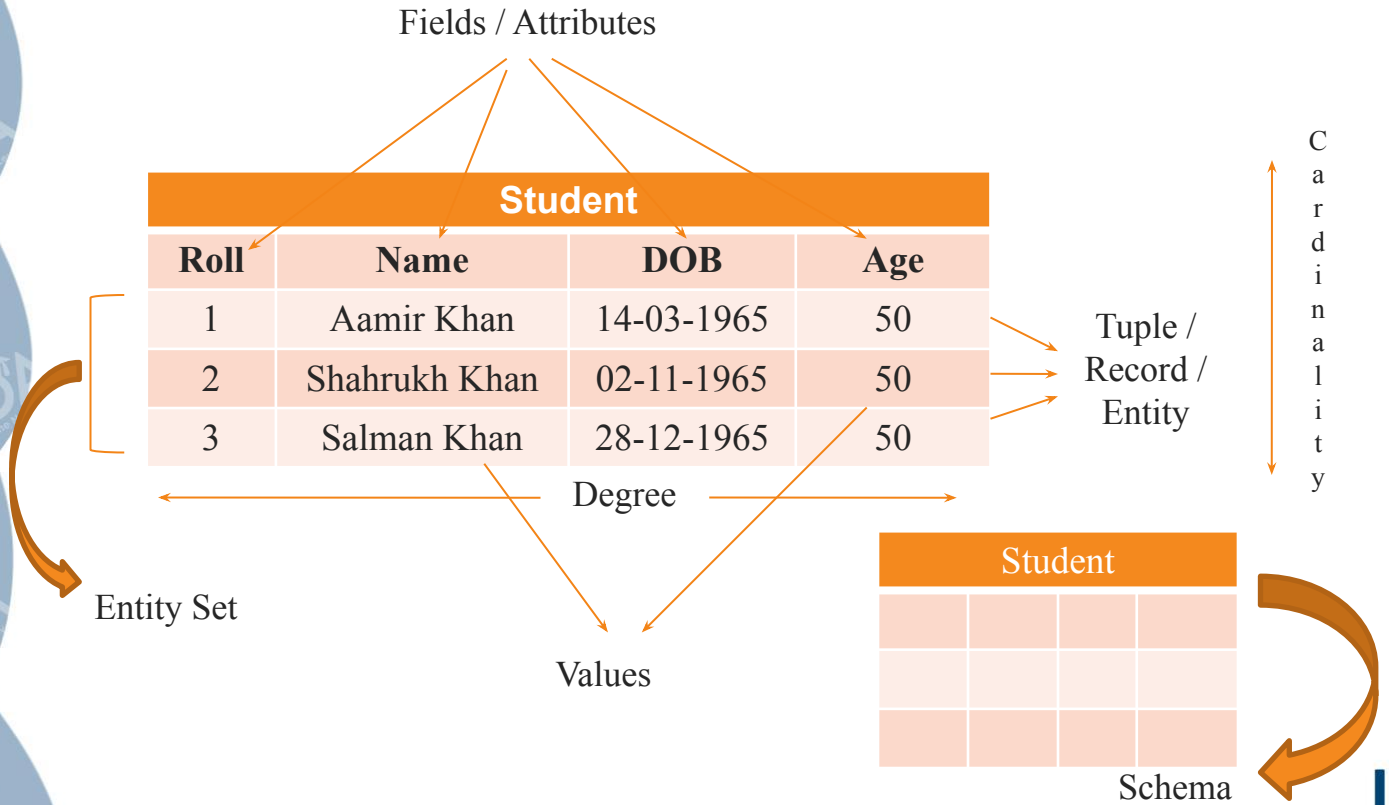
Rule 11 - Distribution Independence Rule

The end-user must not be able to see that the data is distributed over various locations. Users should always get the impression that the data is located at one site only. This rule has been regarded as the foundation of distributed database systems.

Rule 12 - Non-Subversion Rule

If a system has an interface that provides access to low-level records, then the interface must not be able to subvert the system and bypass security and integrity constraints.

Relational Schema



Few Terminologies

Schema – the logical structure of the database (e.g., set of customers and accounts and the relationship between them).

Instance – the actual content of the database at a particular point in time.

Degree – the number of columns associated with a relation or table.

Cardinality – the number of rows in a table.

Domain – the range of possible values for an attribute.
e.g. – the domain for the attribute gender may be {'M', 'F'}
the domain for the attribute title may be { 'Mr.', "Ms.', 'Mrs.', 'Dr.'}

Relational Model - Example

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

Figure 1.3 A sample relational database.

Relational Model

Presents a sample relational database comprising three tables

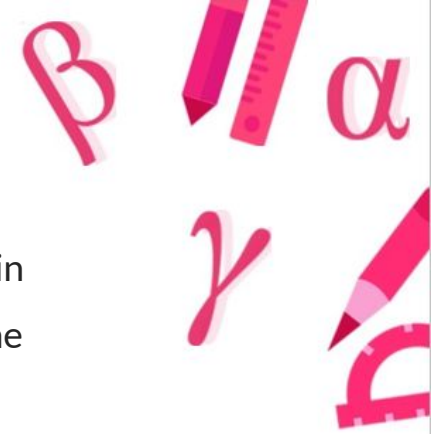
- One shows details of bank **customers**, the second shows **accounts**, and the third shows which **accounts belong to which customers**.
- Each table contains records of a particular type.
- Each record type defines a fixed number of fields, or attributes.
- The columns of the table correspond to the attributes of the record type
- A special character (such as a comma) may be used to delimit the different attributes of a record, and another special character (such as a newline character) may be used to delimit records.
- The relational model hides such low-level implementation details from database developers and users.

Difficulty in accessing data

- Suppose that one of the bank officers needs to find out the names of all customers who live within a particular postal-code area.
- The officer asks the data-processing department to generate such a list.
- Conventional file-processing environments do not allow needed data to be retrieved in a convenient and efficient manner.
- More responsive data-retrieval systems are required for general use.

Data isolation

- Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.



Integrity problems

- The data values stored in the database must satisfy certain types of consistency constraints.
- For example, the balance of a bank account may never fall below a prescribed amount (say, $100).
- Developers enforce these constraints in the system by adding appropriate code in the various application programs.
- When new constraints are added, it is difficult to change the programs to enforce them.
- The problem is compounded when constraints involve several data items from different files.

Atomicity problems

- A computer system, like any other mechanical or electrical device, is subject to failure.
- In many applications, it is crucial that, if a failure occurs, the data be restored to the consistent state that existed prior to the failure.
- Consider a program to transfer $50 from account A to account B.
- If a system failure occurs during the execution of the program, it is possible that the $50 was removed from account A but was not credited to account B, resulting in an inconsistent database state.
- Clearly, it is essential to database consistency that either both the credit and debit occur, or that neither occur.

Atomicity problems

- That is, the funds transfer must be atomic—it must happen in its entirety or not at all.
- It is difficult to ensure atomicity in a conventional file-processing system.

Concurrent-access anomalies

- For the sake of overall performance of the system and faster response, many systems allow multiple users to update the data simultaneously.
- In this environment, interaction of concurrent updates may result in inconsistent data.
- Consider bank account A, containing $500.
- If two customers withdraw funds (say $50 and $100 respectively) from account A at about the same time, the result of the concurrent executions may leave the account in an incorrect (or inconsistent) state.

Concurrent-access anomalies

- Suppose that the programs executing on behalf of each withdrawal read the old balance, reduce that value by the amount being withdrawn, and write the result back.
- If the two programs run concurrently, they may both read the value $500, and write back $450 and $400, respectively.
- Depending on which one writes the value last, the account may contain either $450 or $400, rather than the correct value of $350.
- Therefore, the system must maintain some form of supervision.
- In file systems supervision is difficult to provide because data may be accessed by many different application programs that have not been coordinated previously.

ACID properties of Transaction

1. Atomicity

2. Consistency

3. Isolation

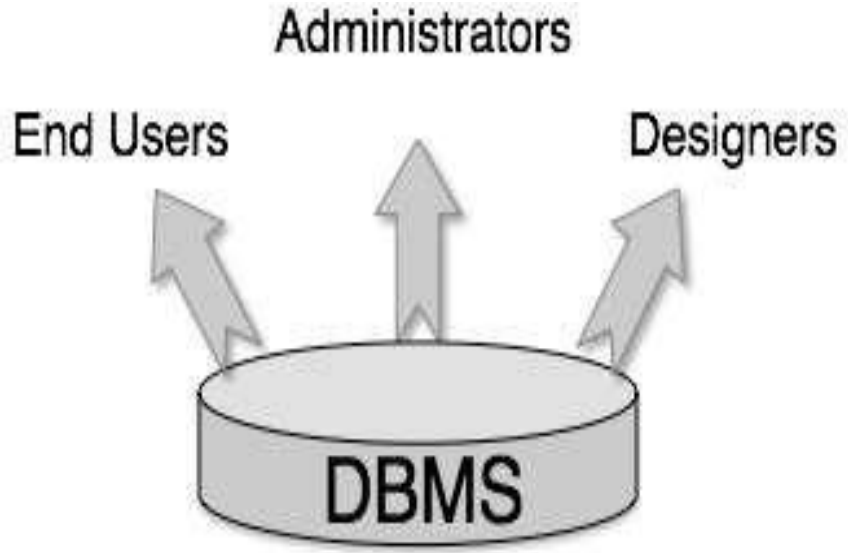
4. Durability

Security problems

- Not every user of the database system should be able to access all the data.
- For example, in a banking system, payroll personnel need to see only that part of the database that has information about the various bank employees.
- They do not need access to information about customer accounts.
- But, since application programs are added to the system in an ad hoc manner, enforcing such security constraints is difficult.

DBMS Users

A typical DBMS has users with different rights and permissions who use it for different purposes. **S**ome users retrieve data and some back it up.
The users of a DBMS can be broadly categorized as follows:



DBMS Users

- **Administrators:** Administrators maintain the DBMS and are responsible for administering the database. They are responsible to look after its usage and by whom it should be used. They create access profiles for users and apply limitations to maintain isolation and force security. Administrators also look after DBMS resources like system license, required tools, and other software and hardware related maintenance.

- **Designers:** Designers are the group of people who actually work on the designing part of the database. They keep a close watch on what data should be kept and in what format. They identify and design the whole set of entities, relations, constraints, and views.

- **End Users:** End users are those who actually reap the benefits of having a DBMS. End users can range from simple viewers who pay attention to the logs or market rates to sophisticated users such as business analysts.

Database Administrator (DBA)

- One of the main reasons for using DBMSs is to have central control of both the data and the programs that access those data.
- A person who has such central control over the system is called a **database administrator** (**DBA**).

DBA Responsibilities

- Granting of authorization for data access. By granting different types of authorization, the database administrator can regulate which parts of the database various users can access.
- The authorization information is kept in a special system structure that the database system consults whenever someone attempts to access the data in the system

DBA Responsibilities

- Routine maintenance. Examples of the database administrator's routine maintenance activities are:
- Periodically backing up the database, either onto tapes or onto remote servers, to prevent loss of data in case of disasters such as flooding.
- Ensuring that enough free disk space is available for normal operations, and upgrading disk space as required.
- Monitoring jobs running on the database and ensuring that performance is not degraded by very expensive tasks submitted by some users.

Thank You!

LearnOA®
To The Next Level...