DATABASE MANAGEMENT

Database is a collection of data and

- **Management System** is a set of programs to store and retrieve those data.
- **DBMS** is a collection of inter-related data and set of programs to store & access those data in an easy and effective manner.
- **Purpose of DBMS**
- Database systems are basically developed for large amount of data. When dealing with huge amount of data, there are two things that require optimization:
- **Storage of data and retrieval of data.**
- **Storage**: The data is stored in such a way that it use less space as the duplicate data has been removed before storage.
- **Fast Retrieval of data:** Along with storing the data in an optimized and systematic manner, it is also important that we retrieve the data quickly when needed. Database systems ensure that the data is retrieved as quickly as possible.

Data

Data means known facts, which can be recorded and implicit meaning. data is also a collection of facts and figures.

Information

Information means processed or organized data.which can be drived from data and facts.

Data-Item (field)

It is a character or group of characters that has a specific meaning.

e.g Rollno ,Name .

Record

It is a collection of logically related fields.Record consists of values for each field.

File

It is a collection related records.which arranged in a specific sequence.

Metadata

Set of data that describes and gives information about another data.

In other words, data about data is called metadata.

System Catalog

The system catalog is a collection of tables and views that contain important information about a database. A system catalog is available for each database.

Data warehouse

A data warehouse is a decision support database that is maintained separately from the organization's operational database.

Data dictionary

Data dictionary is a file that contains metadata that is usually a part of the system catalog.

- It have following for components:
  - Entities
  - Attributes
  - Relationships
  - Key

Characteristics of Database Management System

Stores any kind of data

Adatabase management system should be able to store any kind of data. Any kind of data that exists in the real world can be stored in DBMS because we need to work with all kinds of data that is present around us.

Support ACID Properties

Atomicity

Atomicity means that all transactions must follow "all or nothing" rule. Each transaction is said to be atomic. If one part of the transaction fails, the entire transaction fails.e.g consider an ATMtransaction where you are moving money from one account to another. There are two parts in this transaction, first you remove money from one account, then you add money to another account. If one of these two parts fail, The entire transaction is considered invalid, and the transaction must be rolled back to the state before the transaction started.

Consistency

This means that, the database will always be in a consistent state. Only valid data will be written to the database. E.g if a column is constrained to be NOT NULLand an application attempts to add a row with a NULLvalue in that column, the entire transaction must fail, and no part of the row may be added to the database.

Isolation

Isolation keep transaction separated from each other until they are finished.

Durability

This ensures that the transaction committed to the database will not be lost. Durability is ensured through the use of database backups and transaction logs that facilitate the restoration of committed transactions in spite of any subsequent software or hardware failures.

Represents complex relationship between data

Data stored in a database is connected with each other and a relationship is made in between data. DBMS should be able to represent the complex relationship between data to make the efficient and accurate use of data.

Backup and recovery

There are many chances of failure of whole database. At that time no one will be able to get the database back and for sure company will be in a big loss. The only solution is to take backup of database and whenever it is needed, it can be stored back.

Structures and described data

A database should not contains only the data but also all the structures and definitions of the data. This data represent itself that what actions should be taken on it. These descriptions include the structure, types and format of data and relationship between them.

Data integrity

This is one of the most important characteristics of database management system. Integrity ensures the quality and reliability of database system. It protects the unauthorized access of database and makes it more secure. It brings only the consistence and accurate data into the database.

Concurrent use of database

There are many chances that many users will be accessing the data at the same time. They may require altering the database system concurrently. At that time, DBMS supports them to concurrently use the database without any problem.

Advantage of DBMS

1. Improved data sharing:
2. Improved data security:
3. Better data integration:
4. Minimized data inconsistency:
5. Improved data access:
6. Improved decision making:

Disadvantage of DBMS

1 Complexity.

2.Size

3. Maintaining currency
4. Frequent upgrade/replacement *cycles:*

Functions and responsibilities of DBAs

DBA: person in the organization who controls the design and the use of the database refers as DBA.

1. Schema Definition:

The DBA definition the logical Schema of the database.A Schema refers to the overall logical structure of the database.

According to this schema, database will be developed to store required data for an organization.

2. Storage Structure and Access Method Definition:

The DBA decides how the data is to be represented in the stored database.

3. Assisting Application Programmers:

The DBA provides assistance to application programmers to develop application programs.

4. Physical Organization Modification:

The DBA modifies the physical organization of the database to reflect the changing needs of the organization or to improve performance.

5. Approving Data Access:

The DBA determines which user needs access to which part of the database.

According to this, various types of authorizations are granted to different users.

6. Monitoring Performance:

The DBA monitors performance of the system. The DBA ensures that better performance is maintained by making changes in physical or logical schema if required.

7. Backup and Recovery:

Database should not be lost or damaged.

The DBA ensures this periodically backing up the database on magnetic tapes or remote servers.

Incase of failure, such as virus attack database is recovered from this backup.

DA and DBA

- DA(Data Administrator) and DBA (Database Administrator) both are responsible for managing database for an organization.
  They differ from each other in their required skills and responsibilities.
- Data Administrator (DA):

"Person in the organization who controls the data of the database refers data administrator."

- DAdetermines what data to be stored in database based on requirement of the organization.
- DAworks on such as requirements gathering, analysis, and design phases.
- DAdoes not to be a technical person, any kind of knowledge about database technology can be more beneficiary
- DAis some senior level person in the organization. in short, DAis a business focused person but should understand about the database technology.

Different Types of Database Users in DBMS

1. Application Programmers

Application programmers are the one who writes application programs that uses the database. These application programs are written in programming languages like COBOL or PL (Programming Language 1), Java and fourth generation language. These programs meet the user requirement and made according to user requirements. Retrieving information, creating new information and changing existing information is done by these application programs.

They interact with DBMS through DML (Data manipulation language) calls. And all these functions are performed by generating a request to the DBMS. If application programmers are not there then there will be no creativity in the whole team of Database.

2.End Users

End users are those who access the database from the terminal end. They use the developed applications and they don't have any knowledge about the design and working of database. These are the second class of users and their main motto is just to get their task done.

3.**Casual User**

These users have great knowledge of query language. Casual users access data by entering different queries from the terminal end. They do not write programs but they can interact with the system by writing queries.

4.**Naive**

Any user who does not have any knowledge about database can be in this category. There task is to just use the developed application and get the desired results. For example: Clerical staff in any bank is a naïve user. They don't have any dbms knowledge but they still use the database and perform their given task.

5. DBA (Database Administrator)

DBA can be a single person or it can be a group of person. Database Administrator is responsible for everything that is related to database. He makes the policies, strategies and provides technical supports.

6. **System Analyst**

System analyst is responsible for the design, structure and properties of database. All the requirements of the end users are handled by system analyst. Feasibility, eco

WORKERS BEHIND THE SCENE

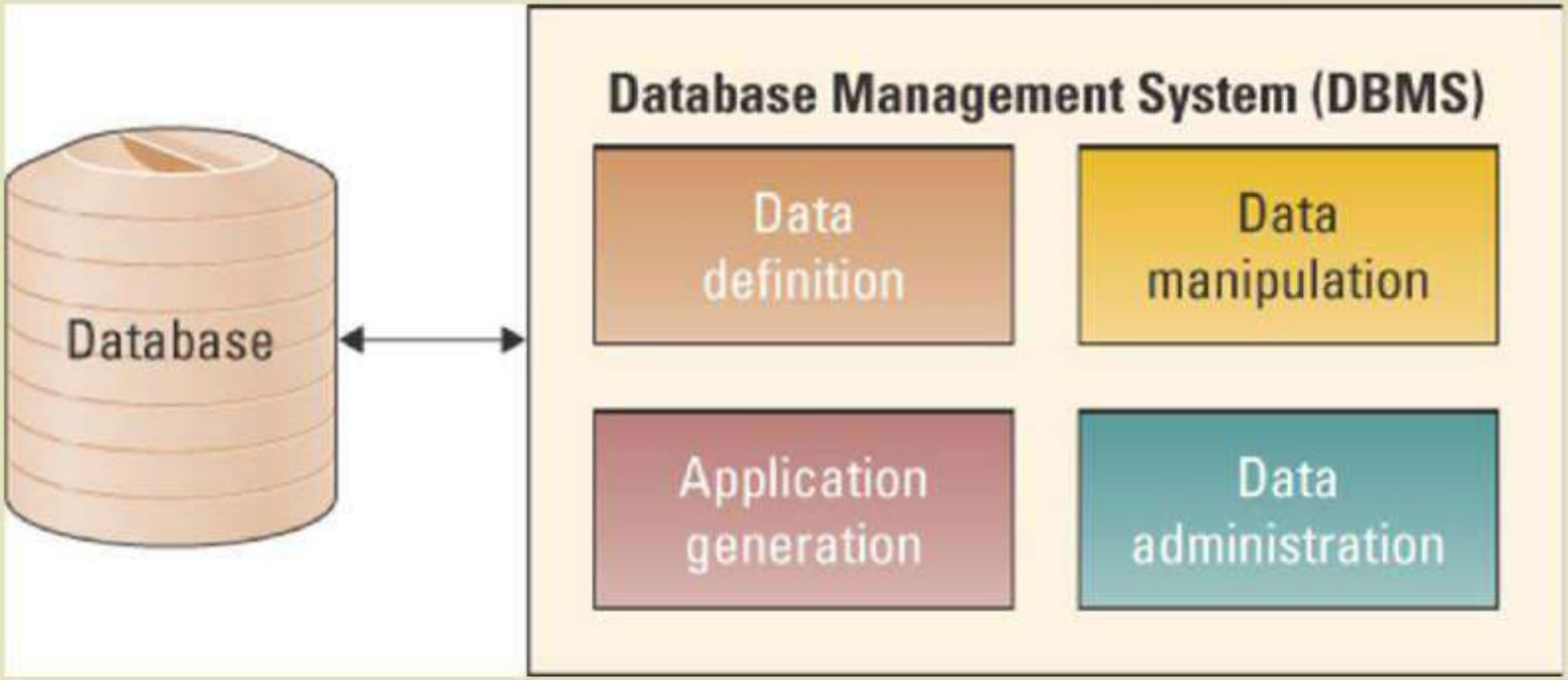
DBMS system designers and implementers • Design and implement the DBMS modules and interfaces as a software package

Tool developers • Design and implement tools

Operators and maintenance personnel • Responsible for running and maintenance of hardware and software environment for database system

DATABASE MANAGEMENT SYSTEMS

- Four components of a DBMS



- Structure Components, and Functions of DBMS
- Structure of DBMS:
- DBMS (Database Management System) acts as an interface between the user and the database. The user requests the DBMS to perform various operations such as insert, delete, update and retrieval on the database.
- The components of DBMS perform these requested operations on the database and provide necessary data to the users.

- The various components of DBMS are described below:
- Components of a DBMS
- The components of DBMS can be divided into two parts:
- *Function and Services of DBMS*
- DDL Compiler:
  - Data Description Language compiler processes schema definitions specified in the DDL.
  - It includes metadata information such as the name of the files, data items, storage details of each file, mapping information and constraints etc.

DML Compiler and Query optimizer:

The DML commands such as insert, update, delete, retrieve from the application program are sent to the DML compiler for compilation into object code for database access.

The object code is then optimized in the best way to execute a query by the query optimizer and then send to the data manager.

Data Manager:

The Data Manager is the central software component of the DBMS also knows as Database Control System.

The Main Functions Of Data Manager Are:

Convert operations in user's Queries coming from the application programs or combination of DML Compiler and Query optimizer which is known as Query Processor from user's logical view to physical file system.Controls DBMS information access that is stored on disk.It also controls handling buffers in main memory.It also enforces constraints to maintain consistency and integrity of the data.It also synchronizes the simultaneous operations performed by the concurrent users.It also controls the backup and recovery operations.

Data Dictionary: Data Dictionary, which stores metadata about the database, particular the schema of the database.names of the tables, names of attributes of each table, length of attributes, and number of rows in each table.

Data Files:

Which store the database itself.

Compiled DML:

The DML complier converts the high level Queries into low level file access commands known as compiled DML.

End Users:

The second class of users then is end user, who interacts with system from online workstation or terminals.

Query Processor Units:

Interprets DDL statements into a set of tables containing metadata.
Translates DML statements into low level instructions that the query evaluation engine understands.

- Functions of DBMS:
  - DBMS free the programmers from the need to worry about the organization and location of the data i.e. it shields the users from complex hardware level details.
  - DBMS can organize process and present data elements from the database. This capability enables decision makers to search and query database contents in order to extract answers that are not available in regular Reports.
  - Programming is speeded up because programmer can concentrate on logic of the application.
  - It includes special user friendly query languages which are easy to understand by non programming users of the system.
- The service provided by the DBMS includes :-
  - Authorization services like log on to the DBMS start the database stop the Database etc.
  - Transaction supports like Recovery, Rollback etc,
  - Import and Export of Data.
  - Maintaining data dictionary
  - User's Monitoring

2.1 Data Model

- Models define how the logical structure of a database is modeled. Data Models are fundamental entities to introduce abstraction in a DBMS. Data models define how data is connected to each other and how they are processed and stored inside the system.
- The very first data model could be flat data-models, where all the data used are to be kept in the same plane.
- Entity-Relationship Model
- Entity-Relationship (ER) Model is based on the notion of real-world entities and relationships among them. The ER Model creates entity set, relationship set, general attributes and constraints.
- ER Model is best used for the conceptual design of a database.

Relational model

The most common model, the relational model sorts data into tables, also known as relations, each of which consists of columns and rows. Each column lists an attribute of the entity in question, such as price, zip code, or birth date. Together, the attributes in a relation are called a domain. A particular attribute or combination of attributes is chosen as a primary key that can be referred to in other tables, when it's called a foreign key.

Hierarchical model

The hierarchical model organizes data into a tree-like structure, where each record has a single parent or root. Sibling records are sorted in a particular order. That order is used as the physical order for storing the database. This model is good for describing many real-world relationships.

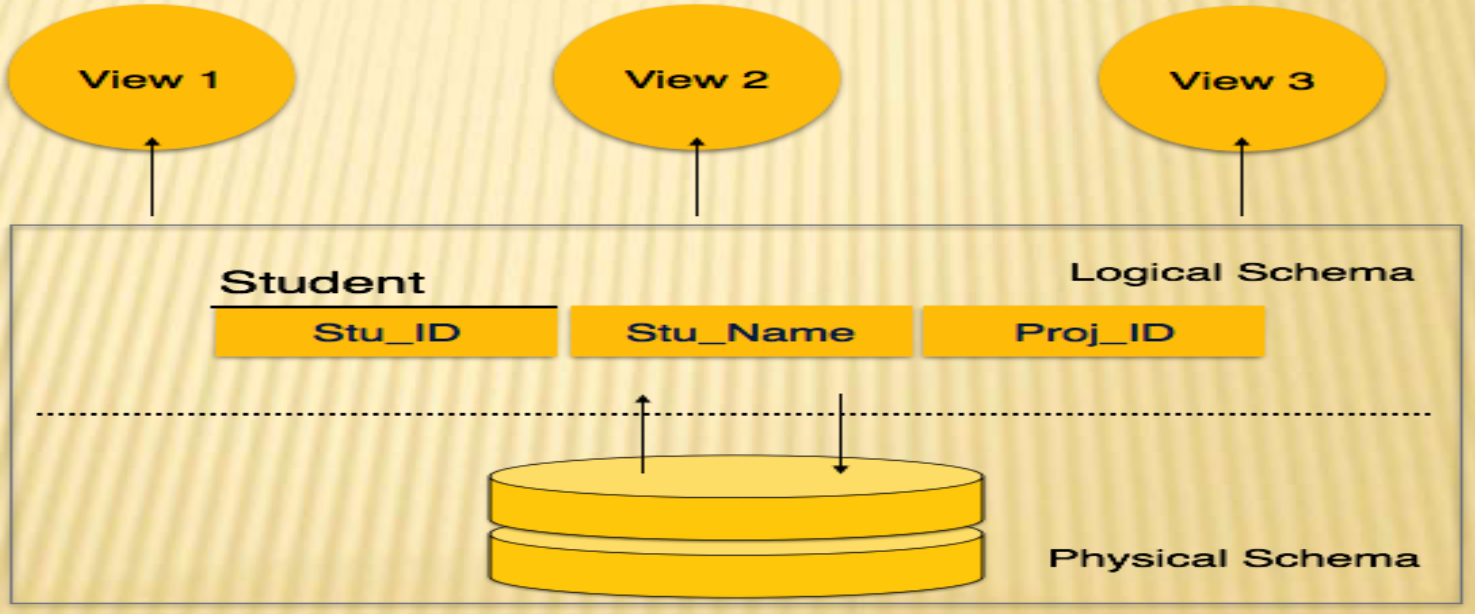
Network model

The network model builds on the hierarchical model by allowing many-to-many relationships between linked records, implying multiple parent records. Based on mathematical set theory, the model is constructed with sets of related records. Each set consists of one owner or parent record and one or more member or child records. A record can be a member or child in multiple sets, allowing this model to convey complex relationships.

Database Schema

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.

A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful.



A database schema can be divided broadly into two categories −

Physical Database Schema − This schema pertains to the actual storage of data and its form of storage like files, indices, etc. It defines how the data will be stored in a secondary storage.

Logical Database Schema − This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.

Database Instance

It is important that we distinguish these two terms individually. Database schema is the skeleton of database. It is designed when the database doesn't exist at all. Once the database is operational, it is very difficult to make any changes to it. A database schema does not contain any data or information.

A database instance is a state of operational database with data at any given time. It contains a snapshot of the database. Database instances tend to change with time. A DBMS ensures that its every instance (state) is in a valid state, by diligently following all the validations, constraints, and conditions that the database designers have imposed.

Data Independence

A database system normally contains a lot of data in addition to users' data. For example, it stores data about data, known as metadata, to locate and retrieve data easily. It is rather difficult to modify or update a set of metadata once it is stored in the database. But as a DBMS expands, it needs to change over time to satisfy the requirements of the users. If the entire data is dependent, it would become a tedious and highly complex job.

Logical Data Independence

Logical data is data about database, that is, it stores information about how data is managed inside. For example, a table (relation) stored in the database and all its constraints, applied on that relation.

Logical data independence is a kind of mechanism, which liberalizes itself from actual data stored on the disk. If we do some changes on table format, it should not change the data residing on the disk.

Physical Data Independence

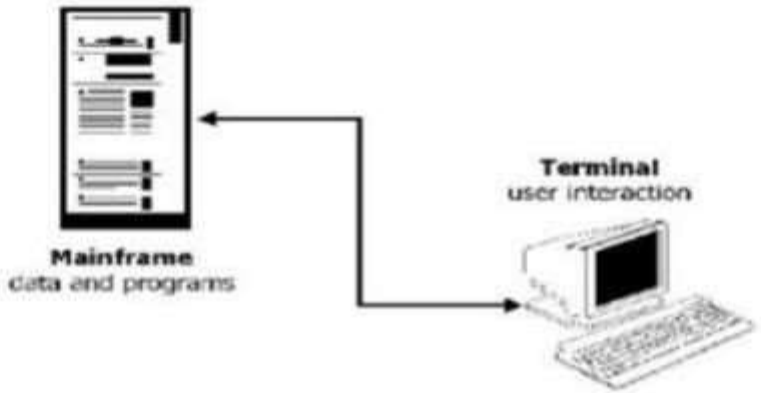
All the schemas are logical, and the actual data is stored in bit format on the disk. Physical data independence is the power to change the physical data without impacting the schema or logical data.

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.

Single Tier Architecture

- Time of Huge Main-Frame computers
- Single Computer manage all processes
- All Resources Attached to the same computer
- Access via dumb Terminals



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.

3-tier Architecture

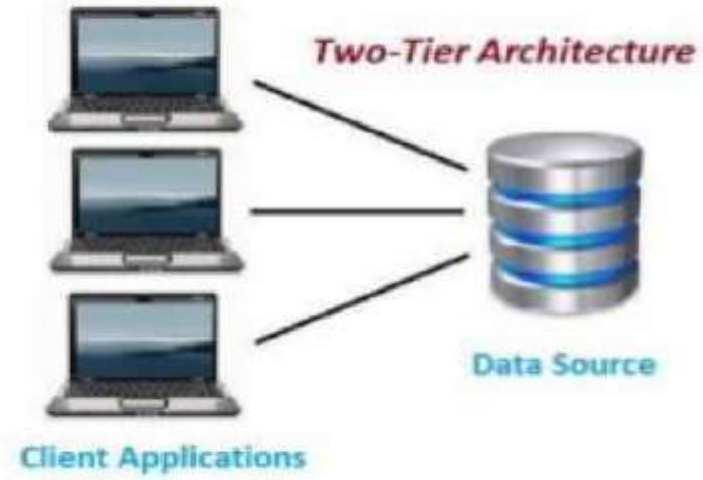
A 3-tier architecture separates its tiers from each other based on the complexity of the users and how they use the data present in the database. It is the most widely used architecture to design a DBMS.

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.

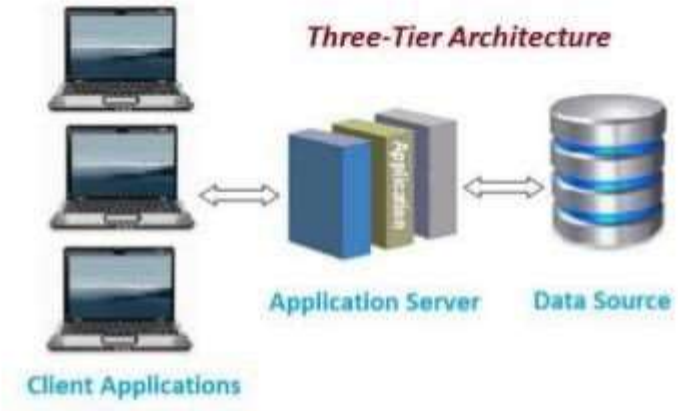
2-TIER ARCHITECTURE

- It is client-server architecture
- Direct communication
- Run faster(tight coupled)



3-TIER ARCHITECTURE

- Web based application
- Three layers:
  1) Client layer
  2) Business layer
  3) Data layer



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.

DBMS Interface ::A database management system (DBMS) interface is a user interface which allows for the ability to input queries to a database without using the query language itself. A DBMS interface could be a web client, a local client that runs on a desktop computer, or even a mobile app.

Entity

An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity.

An entity set is a collection of similar types of entities. An entity set may contain entities with attribute sharing similar values. For example, a Students set may contain all the students of a school; likewise a Teachers set may contain all the teachers of a school from all faculties. Entity sets need not be disjoint.

Attributes

Entities are represented by means of their properties, called **attributes**. All attributes have values. For example, a student entity may have name, class, and age as attributes.

There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.

Types of Attributes

Simple attribute − Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits.

Composite attribute − Composite attributes are made of more than one simple attribute. For example, a student's complete name may have first_name and last_name.

Derived attribute − Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, average_salary in a department should not be saved directly in the database, instead it can be derived. For another example, age can be derived from data_of_birth.

Single-value attribute − Single-value attributes contain single value. For example − Social_Security_Number.

Multi-value attribute − Multi-value attributes may contain more than one values. For example, a person can have more than one phone number, email_address, etc.

Entity-Set and Keys

Key is an attribute or collection of attributes that uniquely identifies an entity among entity set.

For example, the roll_number of a student makes him/her identifiable among students.

Super Key − A set of attributes (one or more) that collectively identifies an entity in an entity set.

Candidate Key − A minimal super key is called a candidate key. An entity set may have more than one candidate key.

Primary Key − A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

Relationship

The association among entities is called a relationship. For example, an employee **works_at** a department, a student **enrolls** in a course. Here, Works_at and Enrolls are called relationships.

Relationship Set

A set of relationships of similar type is called a relationship set. Like entities, a relationship too can have attributes. These attributes are called **descriptive attributes**.

Degree of Relationship

The number of participating entities in a relationship defines the degree of the relationship.

Binary = degree 2

Ternary = degree 3

n-ary = degree

Relational Model

The most popular data model in DBMS is the Relational Model. It is more scientific a model than others. This model is based on first-order predicate logic and defines a table as an n-ary relation.

The main highlights of this model are – Data is stored in tables called relations. Relations can be normalized.

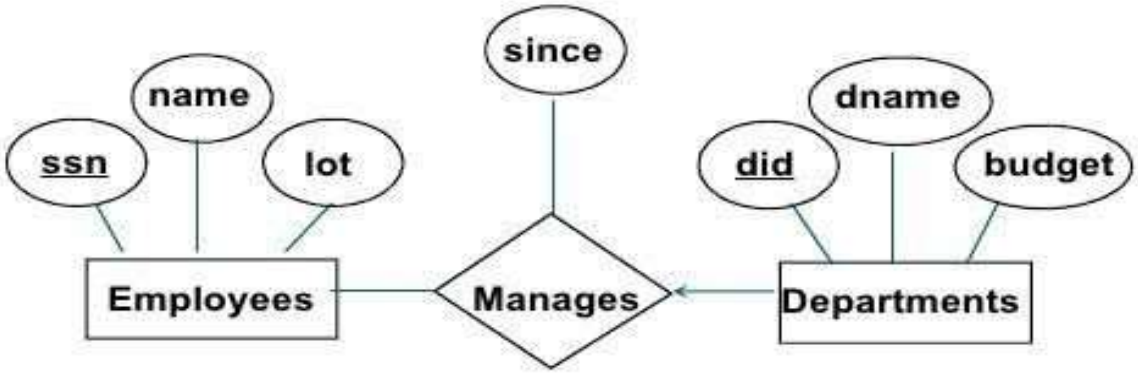
In normalized relations, values saved are atomic values. Each row in a relation contains a unique value.

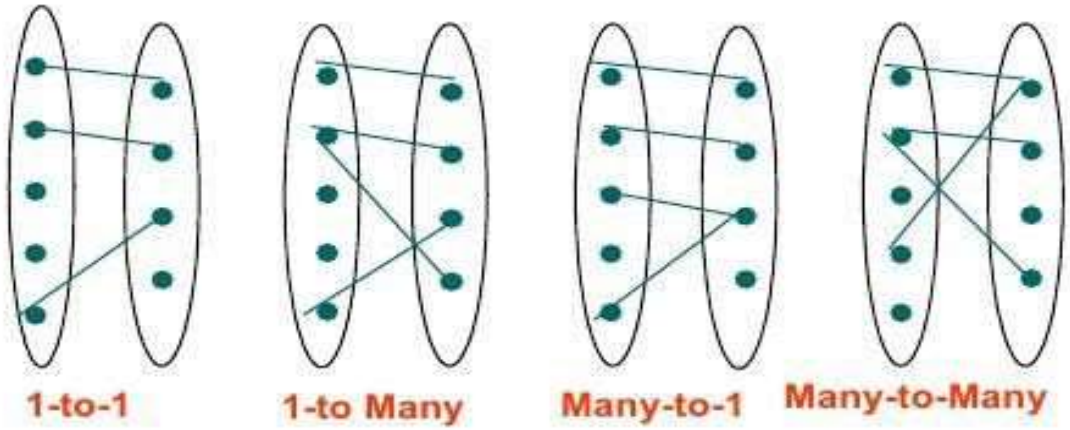
Each column in a relation contains values from a same domain.

Additional features of the ER model

Key Constraints

- Consider Works_In: An employee can work in many departments; a dept can have many employees.
- In contrast, each dept has at most one manager, according to the *key constraint* on Manages.





Relational data model is the primary data model, which is used widely around the world for data storage and processing. This model is simple and it has all the properties and capabilities required to process data with storage efficiency.

Concepts

Tables − In relational data model, relations are saved in the format of Tables. This format stores the relation among entities. A table has rows and columns, where rows represents records and columns represent the attributes.

Tuple − A single row of a table, which contains a single record for that relation is called a tuple.

Relation instance − A finite set of tuples in the relational database system represents relation instance. Relation instances do not have duplicate tuples.

Relation schema − A relation schema describes the relation name (table name), attributes, and their names.

Relation key − Each row has one or more attributes, known as relation key, which can identify the row in the relation (table) uniquely.

Attribute domain − Every attribute has some pre-defined value scope, known as attribute domain.

Constraints

Every relation has some conditions that must hold for it to be a valid relation. These conditions are called **Relational Integrity Constraints**. There are three main integrity constraints −

Key constraints

Domain constraints

Referential integrity constraints

Key Constraints

There must be at least one minimal subset of attributes in the relation, which can identify a tuple uniquely. This minimal subset of attributes is called **key**for that relation. If there are more than one such minimal subsets, these are called ***candidate keys***.

Key constraints force that −

in a relation with a key attribute, no two tuples can have identical values for key attributes.

a key attribute can not have NULL values.

Key constraints are also referred to as Entity Constraints.

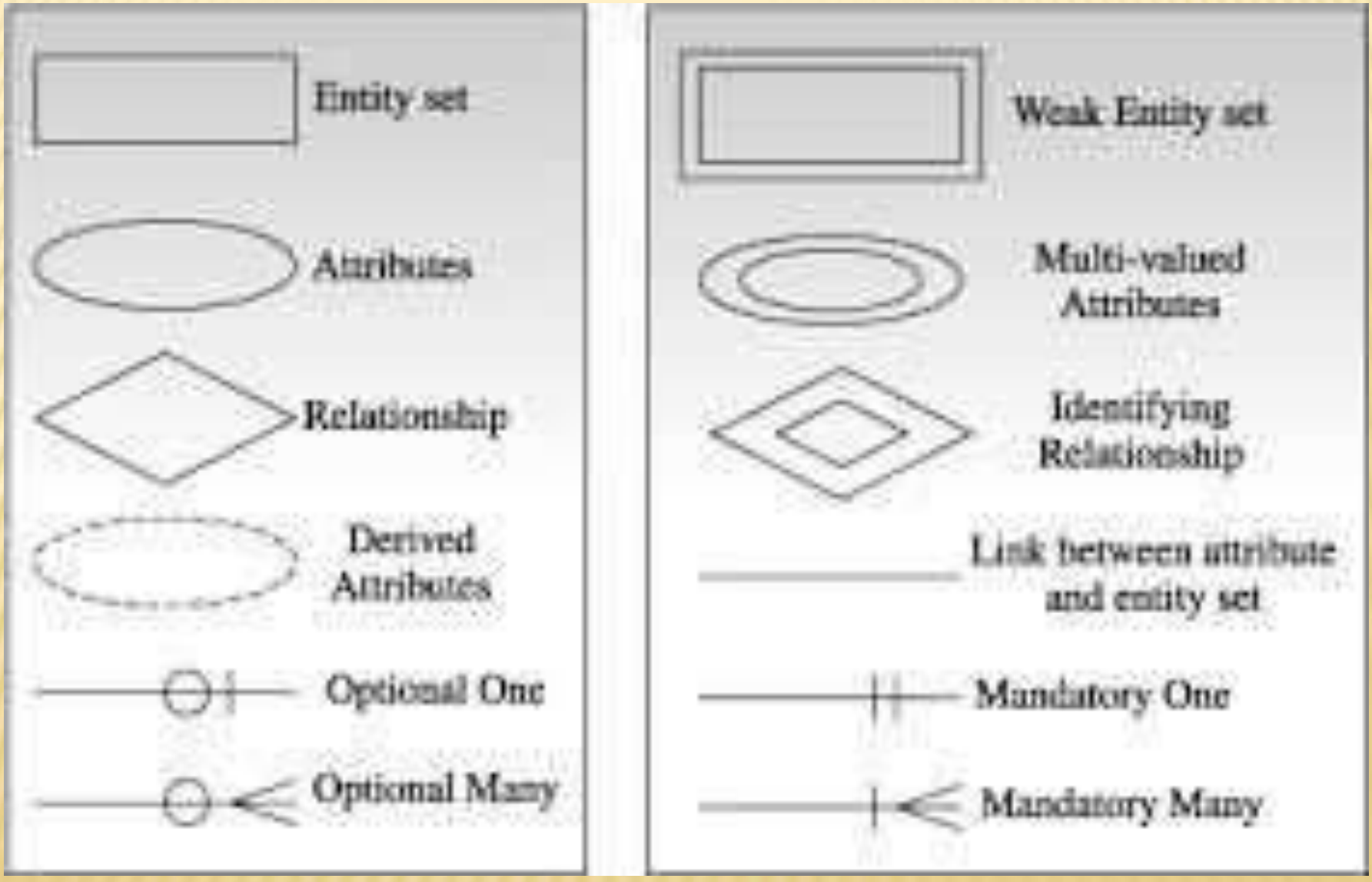
Domain Constraints

Attributes have specific values in real-world scenario. For example, age can only be a positive integer. The same constraints have been tried to employ on the attributes of a relation. Every attribute is bound to have a specific range of values. For example, age cannot be less than zero and telephone numbers cannot contain a digit outside 0-9.

Referential integrity Constraints

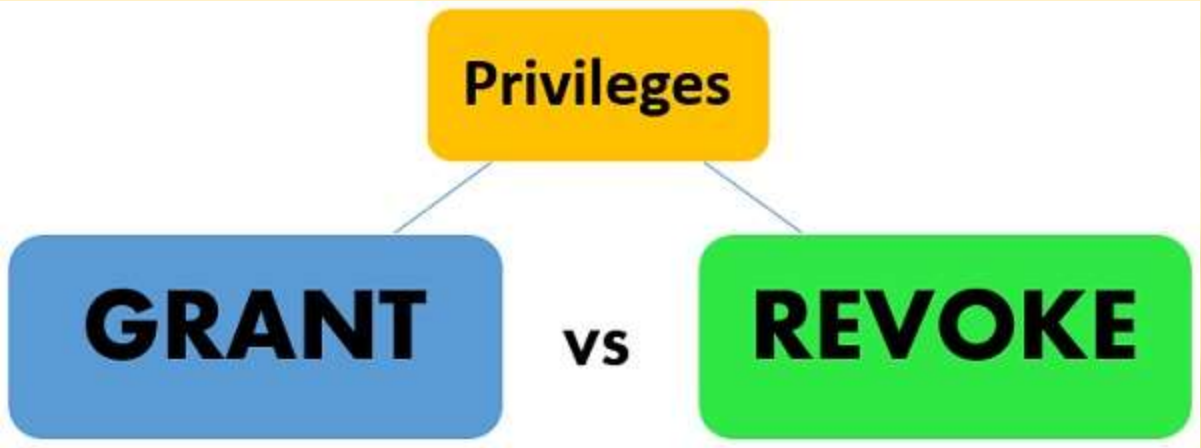
Referential integrity constraints work on the concept of Foreign Keys. A foreign key is a key attribute of a relation that can be referred in other relation.

Referential integrity constraint states that if a relation refers to a key attribute of a different or same relation, then that key element must exist.


Entity set
Attributes
Relationship
Derived Attributes
Optional One
Optional Many
Weak Entity set
Multi-valued Attributes
Identifying Relationship
Link between attribute and entity set
Mandatory One
Mandatory Many

Indexing ::

We know that data is stored in the form of records. Every record has a key field, which helps it to be recognized uniquely. **Indexing** is a data structure technique to efficiently retrieve records from the database files based on some attributes on which the **indexing** has been done.



Definition of Grant

The database administrator defines the **GRANT** command in SQL for giving the access or privileges to the users of the database. Three major components which are involved in the authorization are the users, privilege/s (operations) and a database object. The **user** is the one who triggers the execution of the application program. **Operations** are the component which is embedded in an application program. The **operations** are performed on database objects such as relation or view name.

SYNTAX of GRANT Command:

grant <privilege record>
on <relation title or view title>
to <user/role record>;

Here the privilege list could involve select, insert, update and delete operations or combination of them. These three aspects of the command are checked by authorization control before proceeding.

When an owner account A1 of the relation (table) R grants privilege to another account A2 on R then the account A2 can access the relation R and is authorised to give the privileges to another account on R. If the A1 revokes the privileges from A2 on R1 then, all the privileges that A2 propagated will be revoked automatically by the system. So, this is how the privileges on tables can propagate. Thus, a DBMS permitting propagation should follow the privileges that are granted so that the privileges can be revoked easily.

Let's take an example to illustrate the Granting of privileges. We have two schemas for the tables Faculty and Department and accounts A1 and A2.

GRANT SELECT, INSERT, UPDATE **ON** FACULTY, DEPARTMENT **TO** A1, A2;

In the above given example, the account A1 and A2 are allowed to perform the select, insert and update operations on the employee and department table.

Definition of Revoke

The **REVOKE** command in SQL is defined to take away the granted privileges (authorizations) from the user of the database. The one who has the authority to withdraw the privileges is the database administrator.

SYNTAX of REVOKE Command:

revoke <privilege list>

on <relation name or view name>

from <user/role list>;

DDL COMMANDS

1.Create Statement ::

(i)Create table <tablename>

(Fieldname datatype(size) Key constraint));

(ii) Create table <tablename>as select * from <oldtablename>

2.Alter Statement::

(i)Alter table <tablename>

Add(Fieldname datatype(size) Key constraint));

(ii)Alter table <tablename>

Modify(Fieldname datatype(size));

3.Describe Statement::

Desc <tablename>

4.Drop Statement:: Drop

table <tablename>

DML Command::

1.insert Statement::

Insert into <tablename>

Values(&fieldname1,……..);

2.Update Statement::

Update tablename

Set fieldname=expression;

3.Delete statement

Delete from tablename;

4.Select statement::

Select * from tablename;