DATABASE MANAGEMENT SYSTEMS
UNIT 1 – TOPIC 1
INTRODUCTION TO DBMS

Database

A database is a collection of data, typically describing the activities of one or more related organizations.

Example: a university database might contain information about the following:
• *Entities* such as students, faculty, courses, and classrooms.
• *Relationships* between entities, such as students' enrollment in courses, faculty teaching courses, and the use of rooms for courses.

Database Management System (DBMS)

- A database management system, or DBMS, is software designed to assist in maintaining and utilizing large collections of data.
- The alternative to using a DBMS is to store the data in files and write application-specific code to manage it.

A Historical Perspective or History of Data base Systems

- Finding information from a huge volume of papers or deleting/modifying an entry is a difficult task in pen and paper based approach.
- To overcome the hassles faced in manual record keeping, it is desirable to computerize storage of data.
- From the earliest days of computers, storing and manipulating data have been a major application focus.
- In the late 1960s, IBM developed the Information Management System (IMS) DBMS, used even today in many major installations.
- IMS formed the basis for an alternative data representation framework called the *hierarchical data model.*
- In 1970, Edgar Codd, at IBM's San Jose Research Laboratory, proposed a new data representation framework called the *relational data model.*
- In the 1980s, the relational model consolidated its position as the dominant DBMS paradigm, and database systems continued to gain widespread use.
- SQL was standardized in the late 1980s, and the current standard, SQL: 1999, was adopted by the American National Standards Institute (ANSI) and International Organization for Standardization (ISO).
- Specialized systems have been developed by numerous vendors for creating *data warehouses,* consolidating data from several databases, and for carrying out specialized analysis.
- Commercially, database management systems represent one of the largest and most vigorous market segments.

FILE SYSTEMS VERSUS DBMS

File System	Database Management System (DBMS)
1. It is a software system that manages and controls the data files in a computer system.	1. It is a software system used for creating and managing the databases. DBMS provides a systematic way to access, update, and delete data.
2. File system does not support multi-user access.	2. Database Management System supports multi-user access.
3. Data consistency is less in the file system.	3. Data consistency is more due to the use of normalization.
4. File system is not secured.	4. Database Management System is highly secured.
5. File system is used for storing the unstructured data.	5. Database management system is used for storing the structured data.
6. In the file system, data redundancy is high.	6. In DBMS, Data redundancy is low.
7. No data backup and recovery process is present in a file system.	7. There is a backup recovery for data in DBMS.
8. Handling of a file system is easy.	8. Handling a DBMS is complex.
9. Cost of a file system is less than the DBMS.	9. Cost of database management system is more than the file system.
10. If one application fails, it does not affect other application in a system.	10. If the database fails, it affects all application which depends on it.
11. In the file system, data cannot be shared because it is distributed in different files.	11. In DBMS, data can be shared as it is stored at one place in a database.
12. These system does not provide concurrency facility.	12. This system provides concurrency facility.
13. **Example:** NTFS (New technology file system), EXT (Extended file system), etc.	13. **Example:** Oracle, MySQL, MS SQL Server, DB2, Microsoft Access, etc.

ADVANTAGES OF A DBMS

- **Data Independence**: The DBMS provides an abstract view of the data that hides data representation and storage details.
- **Efficient Data Access**: A DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently.
- **Data Integrity and Security**: If data is always accessed through the DBMS, the DBMS can enforce integrity constraints. For example, before inserting salary information for an employee, the DBMS can check that the department budget is not exceeded. Also, it can enforce *access controls* that govern what data is visible to different classes of users.
- **Data Administration**: When several users share the data, centralizing the administration of data can offer significant improvements.
- **Concurrent Access and Crash Recovery**: A DBMS schedules concurrent accesses to the data in such a manner that users can think of the data as being accessed by only one user at a time. Further, the DBMS protects users from the effects of system failures.
- **Reduced Application Development Time:** The high-level interface to the data, facilitates quick application development.

Database System Applications

Applications where we use Database Management Systems are:

- **Telecom**: There is a database to keep track of the information regarding calls made, network usage, customer details etc. Without the database systems it is hard to maintain that huge amount of data that keeps updating every millisecond.
- **Industry**: Where it is a manufacturing unit, warehouse or distribution centre, each one needs a database to keep the records of ins and outs. For example distribution centre should keep a track of the product units that supplied into the centre as well as the products that got delivered out from the distribution centre on each day; this is where DBMS comes into picture.
- **Banking System**: For storing customer info, tracking day to day credit and debit transactions, generating bank statements etc. All this work has been done with the help of Database management systems.
- **Education sector**: Database systems are frequently used in schools and colleges to store and retrieve the data regarding student details, staff details, course details, exam details, payroll data, attendance details, fees details etc. There is a hell lot amount of inter-related data that needs to be stored and retrieved in an efficient manner.
- **Online shopping**: You must be aware of the online shopping websites such as Amazon, Flipkart etc. These sites store the product information, your addresses and preferences, credit details and provide you the relevant list of products based on your query. All this involves a Database management system.

DATABASE MANAGEMENT SYSTEMS

UNIT 1 – TOPIC 2

DATA MODELS

DATA MODEL

- A **data model** is the underlying structure of DBMS. It is a collection of high-level data description constructs that hide many low-level storage details.
- A data model describes how a database's logical structure is represented.
- It is a conceptual tool used to describe data, relationship, semantics and constraints.
- Most database management systems today are based on the **relational data model.**
- A **semantic data model** is a more abstract, high-level data model that makes it easier for a user to come up with a good initial description of the data in an enterprise.
- A widely used semantic data model called the entity-relationship (ER) model allows us to pictorially denote entities and the relationships among them.

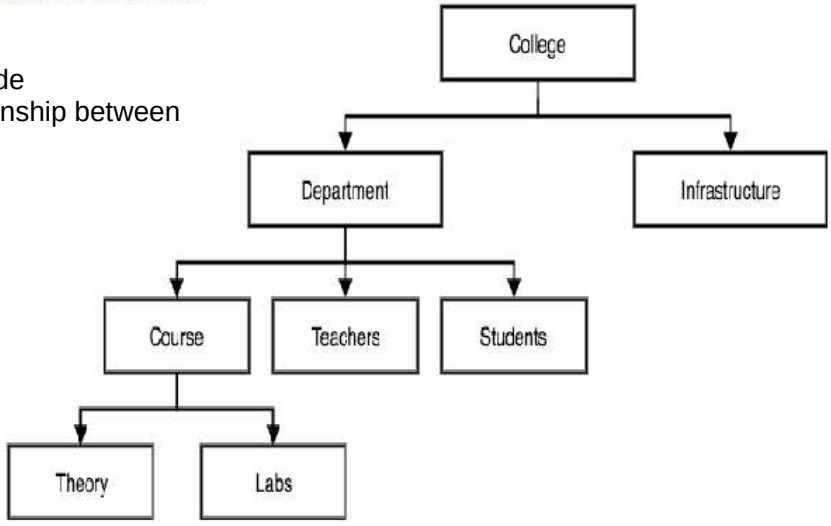
Data model types

- Older models:
  - Network model
  - Hierarchical model
- Relational model
- Entity-Relationship data model (mainly for database design)
- Object-Oriented data models

Hierarchical Model

- Hierarchical Model was the first DBMS model.
- This model organizes the data in the hierarchical tree structure.
- The hierarchy starts from the root which has root data and then it expands in the form of a tree adding child node to the parent node.
- This model easily represents some of the real-world relationships like food recipes, sitemap of a website etc.

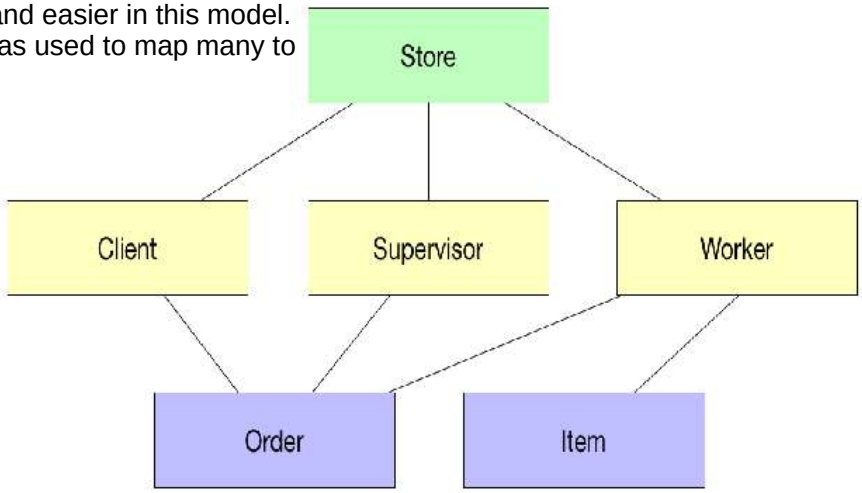
1.It has only one parent node
2.It has one to many relationship between two different types of data.



Network Model

- This model is an extension of the hierarchical model.
- It was the most popular model before the relational model.
- This model is the same as the hierarchical model; the only difference is that a record can have more than one parent.
- It replaces the hierarchical tree with a graph.

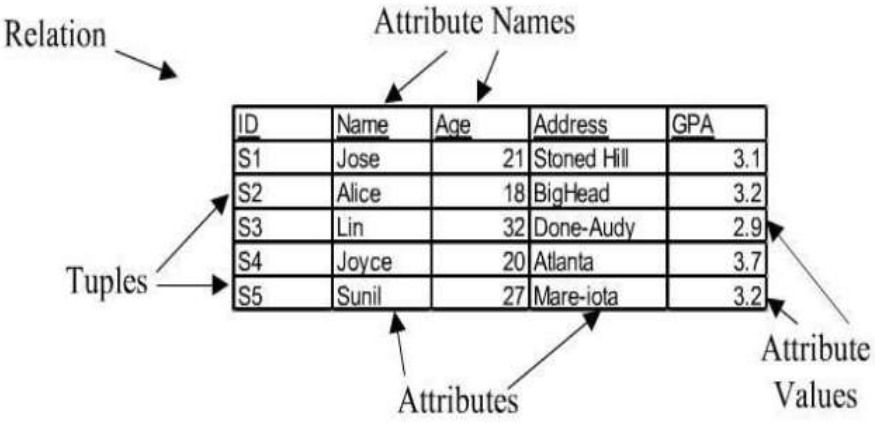
Accessing data is fast and easier in this model. This database model was used to map many to many relationships



Relational model

- The central data description construct in this model is a relation, which can be thought of as a set of records.
- A description of data in terms of a data model is called a schema.
- In the relational model, the schema for a relation specifies its name, the name of each field (or attribute or column), and the type of each field.
- Students (*sid:* string, *name:* string, *login:* string, *age:* integer, *gpa:* real)

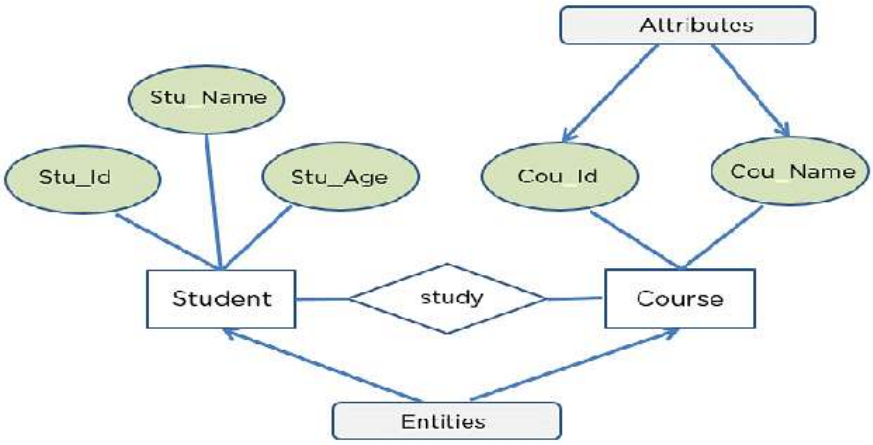
Edgar codd in 1970.most widely used data model



ID	Name	Age	Address	GPA
S1	Jose	21	Stoned Hill	3.1
S2	Alice	18	BigHead	3.2
S3	Lin	32	Done-Audy	2.9
S4	Joyce	20	Atlanta	3.7
S5	Sunil	27	Mare-iota	3.2

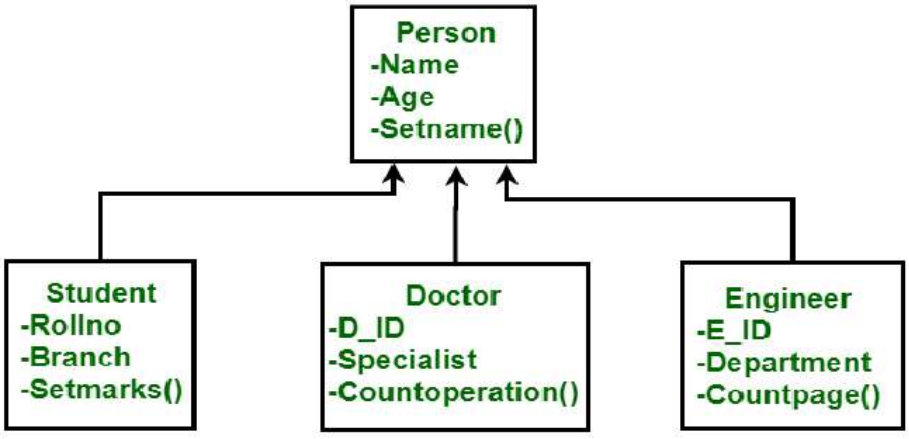
Entity-Relationship Model

- Entity-Relationship Model or simply ER Model is a high-level data model diagram.
- In this model, we represent the real-world problem in the pictorial form to make it easy for the stakeholders to understand.
- It is also very easy for the developers to understand the system by just looking at the ER diagram.
- We use the ER diagram as a visual tool to represent an ER Model.



Object-Oriented Data Model

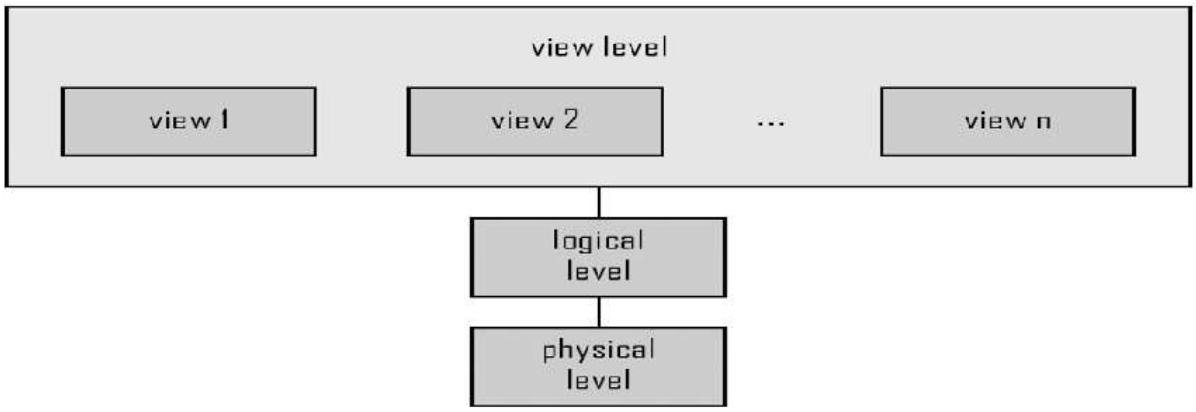
- The real-world problems are more closely represented through the object-oriented data model.
- In this model, both the data and relationship are present in a single structure known as an object.
- We can store audio, video, images, etc in the database which was not possible in the relational model (although you can store audio and video in relational database, it is advised not to store in the relational database).
- In this model, two are more objects are connected through links. We use this link to relate one object to other object.



DATABASE MANAGEMENT SYSTEMS
UNIT 1 – TOPIC 3
LEVELS OF ABSTRACTION & DATA INDEPENDENCE

LEVELS OF DATA ABSTRACTION

In a Database Management System (DBMS), abstraction refers to the process of hiding the complexity of the underlying data structure and operations from the users. This abstraction is essential for managing large volumes of data efficiently and for providing a simplified interface for users to interact with the database.



There are 3 level architecture of database design.

Physical level / Internal View: (How) it is the lowest level of data abstraction. It tells us how the data is actually stored in memory. While designing the database few basic information needs to be considered like, storage allocation, indexing methods, data retrieval mechanisms, usability, size of the memory etc.

Example : record : customer

Logical level / Conceptual View: (What) The conceptual level, also known as the logical level, is the level of abstraction that defines the overall structure (schema) of the database. It hides the details of the physical storage from the database users and applications. It defines the relationships and constraints among the data elements without specifying how the data is physically stored.

The conceptual level is designed to provide a high-level view of the database that is independent of any specific implementation. It allows users to define the structure of the database in a way that is meaningful to them without worrying about the details of how the data is stored or accessed.

Example :

type *customer* = record
customer_id : string;
customer_name : string;
customer_street : string;
customer_city : string;
end;

View level / External View: (Who)

The external level, also known as the View level, is the highest level of abstraction and is closest to the end users. It defines how the data is viewed by the users or applications that access the database. It provides a high-level view of the database that is tailored to the specific needs of each user or application.

Ex:website of clg showing different to students and teachers

The external level is designed to provide a level of abstraction that shields the users or applications from the complexities of the underlying database. It allows users to work with the data in a way that is meaningful to them without worrying about the details of how the data is stored or organized.

Benefits of Levels of Abstraction in DBMS

The levels of abstraction in DBMS provide several benefits to users and applications, including:

- The levels of abstraction in DBMS provide a separation between the way data is viewed by users or applications and how it is stored and accessed by the database management system. This allows changes to be made to the physical storage and access methods without affecting the external or conceptual levels.
- The levels of abstraction in DBMS make it easier for database administrators to manage the database. They can make changes to the physical storage and access methods without affecting the users or applications that interact with the database.
- The levels of abstraction in DBMS allow the database management system to optimize the physical storage and access methods for performance without affecting the way data is viewed by users or applications.
- The levels of abstraction in DBMS allow users or applications to view the data in a way that is meaningful to them without worrying about the underlying implementation details. This makes it easier to adapt to changing business requirements and user needs.

Data Independence

The ability to modify the schema in one level without affecting the schema in next higher level is called data independence. It helps to keep the data separated from all programs that makes use of it.

- **Physical Data Independence** – The ability to modify the physical schema without changing the logical schema. It helps to keep the data separated from all program that makes use of it. **Example**: The Conceptual structure of the database would not be affected by any changes like utilizing a new storage device, change in storage size of the database system server, modifying the data structures used for storage, using an alternate file organization technique, changing from sequential to random access files etc.
- **Logical data independence**: The ability to modify the logical schema without affecting the schema in next higher level (external schema). The user view of the data would not be affected by any changes to the conceptual view of the data. These changes may include insertion or deletion of attributes, altering table structures entities or relationships to the logical schema, etc.

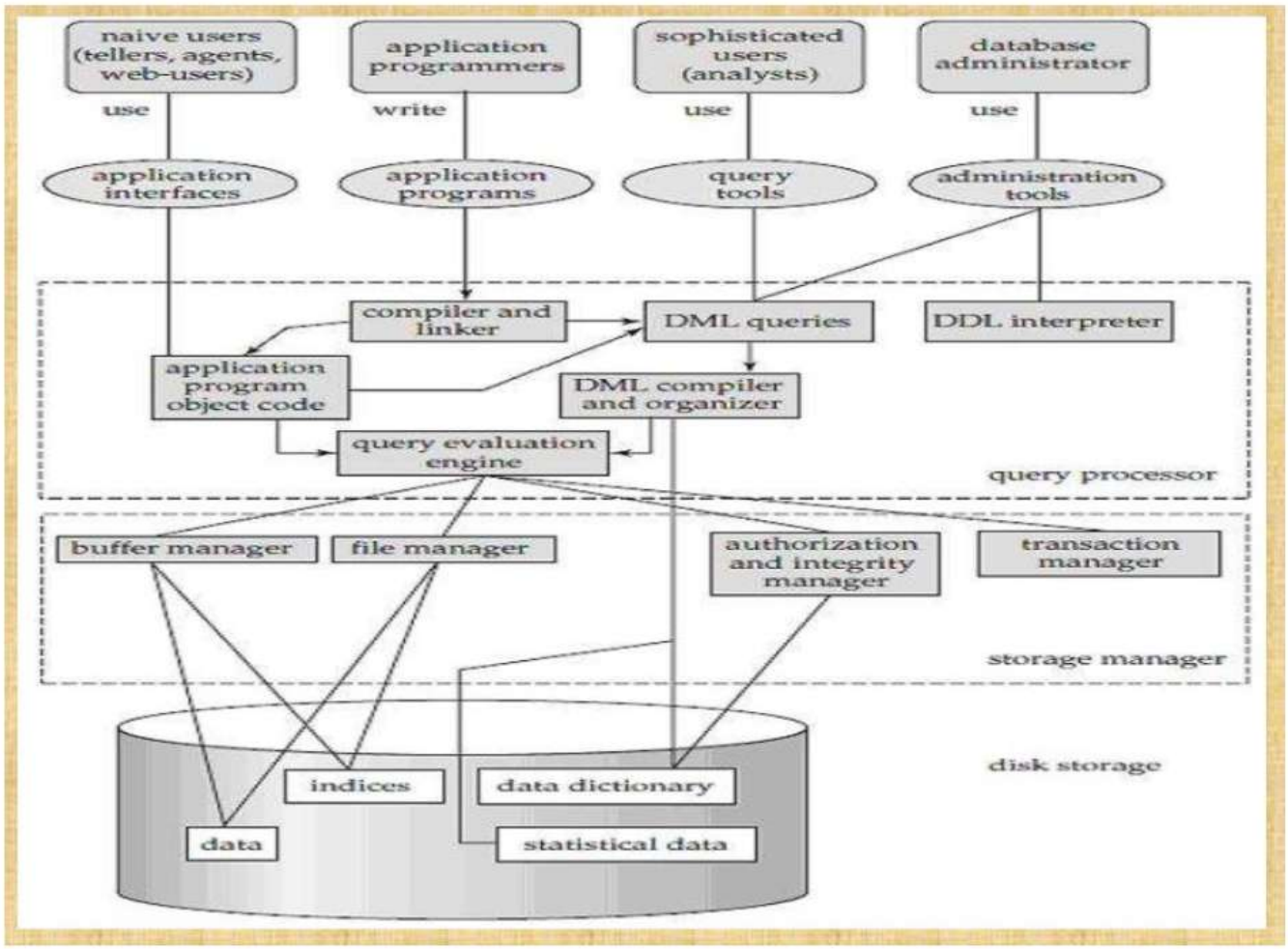
DATABASE MANAGEMENT SYSTEMS
UNIT 1 – TOPIC 4
STRUCTURE OF DBMS

STRUCTURE OF DBMS

Database Management System is the combination of a Database and all the functionalities to organize and manage the data. It is software that allows us to efficiently interact with data at various levels of abstraction.

As Database Management System is a complex set of programs, it is necessary to understand all the components of DBMS so that we can easily manage the database

- A database system is partitioned into modules, each of that deal with responsibilities of the overall system.
- **The structure of DBMS refers to the logical representation of all of its modules.**
- The functional components of a database system can be broadly divided into the
  - **Query processor**
  - **Storage manager and**
  - **Disk Storage**



1. **Query Processor:**
   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 contains the following components

   - **DDL Interpreter** –
     DDL Interpreter interprets DDL statements like those used in schema definitions (such as create, remove, etc.). This interpretation yields a set of tables that include the meta-data (data of data) that is kept in the data dictionary.
   - **DML Compiler** –
     DML Compiler converts DML statements like select, update, and delete into low-level instructions or simply machine-readable object code, to enable execution. It processes the DML statements into low level instruction (machine language), so that they can be executed.
   - **Query Evaluation Engine** –
     It evaluates the SQL commands used to access the database's contents before returning the result of the query. A single query can be translated into a number of evaluation plans.
   - **Query Optimizer** –
     query optimization determines the most effective technique to carry out a query.

2. **Storage Manager :**
   - Storage Manager is a program that provides an interface between the data stored in the database and the queries received.
   - 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.
   - It is responsible for updating, storing, deleting, and retrieving data in the database.

It contains the following components –

- **Authorization Manager** –
  It ensures role-based access control, i.e. it checks whether the particular person is privileged to perform the requested operation or not.

- **Integrity Manager** –
  It checks the integrity constraints when the database is modified.

- **Transaction Manager** –
  It controls concurrent access by performing the operations in a scheduled way that it receives the transaction. Thus, it ensures that the database remains in the consistent state before and after the execution of a transaction.

- **File Manager** –
  It manages the file space and the data structure used to represent information in the database.

- **Buffer Manager** –
  It is responsible for cache memory and the transfer of data between the secondary storage and main memory.

3. **Disk Storage:**
   A DBMS can use various kinds of Data Structures as a part of physical system implementation in the form of disk storage.

   It contains the following components – Data Files, Data Dictionary, and Indices

   Data Files – It stores the data in the files supported by the native Operating System.

   Data Dictionary – It contains the information about the structure of any database object. It is the repository of information that governs the metadata.

   Indices – These indices are used to access and retrieve the data in a very fast and efficient way. It provides faster retrieval of data item.

Types of Database Users

1. Database Administrator (DBA):

- It is a person or a team, who is responsible for managing the overall database management system.
- It is the leader of the database. It is like a super-user of the system.
- It is responsible for the administration of all the three levels of the database.

DBA is responsible for:

- Deciding the instances for the database.
- Defining the Schema
- Liaising with Users
- Define Security
- Back-up and Recovery
- Monitoring the performance

2. Database Designers:

- Database designers design the appropriate structure for the database, where we share data.

3. System Analyst:

- System analyst analyses the requirements of end users, especially naïve and parametric end users.

4. Application Programmers:

- Application programmers are computer professionals, who write application programs.

5. Naïve Users / Parametric Users:

- Naïve Users are Un-sophisticated users, which has no knowledge of the database. These users are like a layman, which has a little bit of knowledge of the database.
- Naive Users are just to work on developed applications and get the desired result.
- For Example: Railway's ticket booking users are naive users. Or Clerical staff in any bank is a naïve user because they don't have any DBMS knowledge but they still use the database and perform their given task.

6. Sophisticated Users:

- Sophisticated users can be engineers, scientists, business analyst, who are familiar with the database. These users interact with the database but they do not write programs

7. Casual Users / Temporary Users:

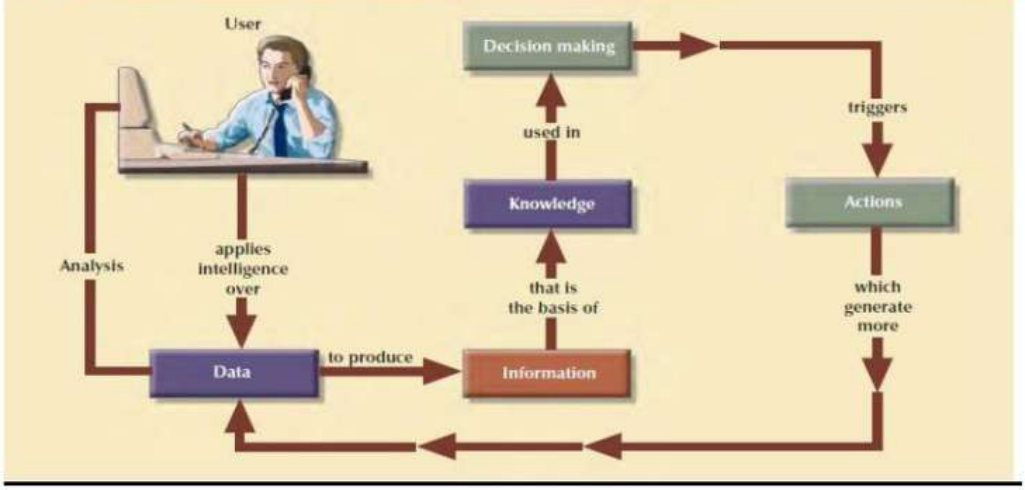
- These types of users communicate with the database for a little period of time.

DATABASE MANAGEMENT SYSTEMS
Data Base Administrator

Data as a Corporate Asset:

Data are a valuable resource that can translate into information. If the information is accurate and timely, it is likely to trigger actions that enhance the company's competitive position and generate wealth. In effect, an organization is subject to a data-information-decision cycle; that is, the data user applies intelligence to data to produce information that is the basis of knowledge used in decision making by the user. This cycle is illustrated in Figure below.

FIGURE 15.1 The data-information-decision-making cycle



The decisions made by high-level managers trigger actions within the organization's lower levels. Such actions produce additional data to be used for monitoring company performance. Thus, data form the basis for decision making, strategic planning, control, and operations monitoring. To manage data as a corporate asset, managers must understand the value of information—that is, processed data.

Role of a Database in an Organization:

Data are used by different people in different departments for different reasons. Therefore, data management must address the concept of shared data. The DBMS facilitates:

- Interpretation and presentation of data in useful formats
- Distribution of data and information to the right people at the right time.
- Data preservation and monitoring the data usage for adequate periods of time.
- Control over data duplication and use, both internally and externally.
- Data Security and Concurrent access.

The database's predominant role is to support managerial decision making at all levels in the organization while preserving data privacy and security.

DataBase Administration Function:

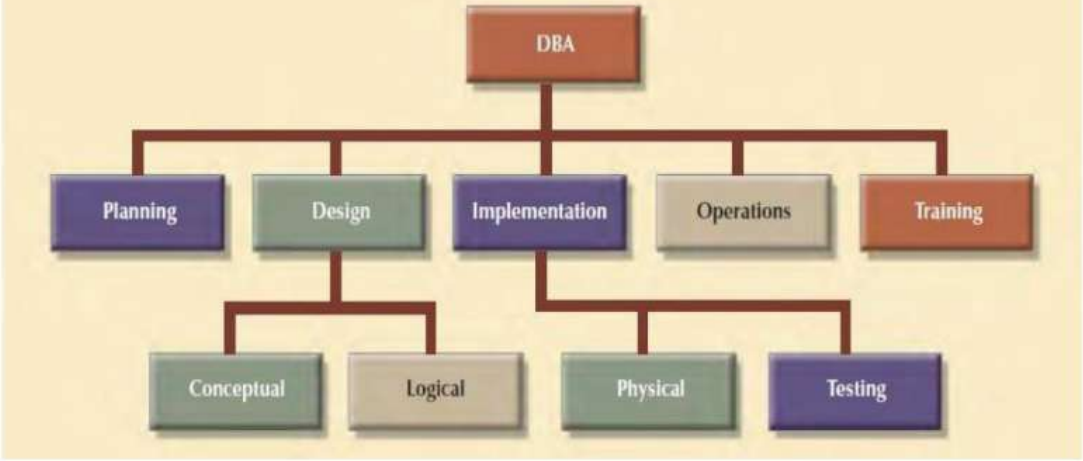
The person responsible for the control of the centralized and shared database became known as the **DataBase Administrator (DBA).**

It is common practice to define the DBA function by dividing the DBA operations according to the Database Life Cycle (DBLC) phases. If that approach is used, the DBA function requires personnel to cover the following activities:

- Database planning, including the definition of standards, procedures, and enforcement.
- Database requirements gathering and conceptual design.
- Database logical and transaction design.
- Database physical design and implementation.
- Database testing and debugging.
- Database operations and maintenance, including installation, conversion, and migration.
- Database training and support.
- Data quality monitoring and management.

Figure below represents an appropriate DBA functional organization according to that model.

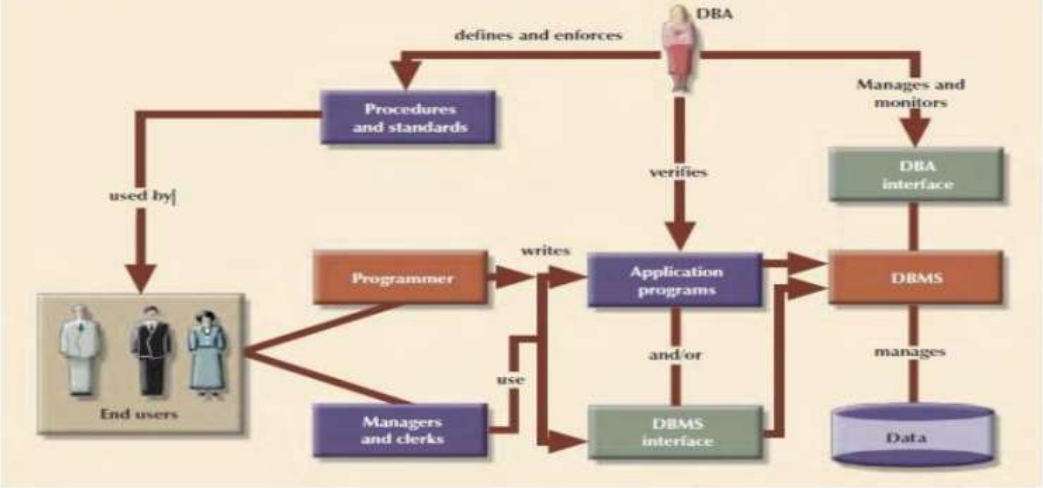
FIGURE 15.4 A DBA functional organization



The interactions between the people and data in an organization, places the DBA in the dynamic environment as explored in the Figure below. The DBA is the focal point for data/user interaction. The DBA defines and enforces the procedures and standards to be used by programmers and end users during their work with the DBMS.

The DBA also verifies that programmer and end-user access meets the required quality and security standards.

FIGURE 15.6 A summary of DBA activities



DBA skills are be divided into two categories **Managerial Role**
Technical Role

The DBA's Managerial Role

As a manager, the DBA must concentrate on the control and planning dimensions of database administration. Therefore, the DBA is responsible for:

- Coordinating, monitoring, and allocating database administration resources: i.e. people and data.
- Defining goals and formulating strategic plans for the database administration function.

1. End-User Support

The DBA interacts with the end user by providing data and information support services to the organization's departments.

- Gathering user requirements.
- Building end-user confidence.
- Resolving conflicts and problems.
- Finding solutions to information needs.
- Ensuring quality and integrity of data and applications.
- Managing the training and support of DBMS users.

2. Policies, Procedures and Standards

A prime component of a successful data administration strategy is the continuous enforcement of the policies,procedures, and standards for correct data creation, usage, distribution, and deletion within the database.

- **Policies** are general statements of direction or action that communicate and support DBA goals.
- **Standards** describe the minimum requirements of a given DBA activity; they are more detailed and specificthan policies.
- **Procedures** are written instructions that describe a series of steps to be followed duringthe performance ofa given activity.

3. Data Security, Privacy and Integrity

The security, privacy, and integrity of the data in the database are of great concern to DBAs who manage current DBMS installations. The distribution of data across multiple sites, has made data maintenance, security, and integrity very critical. The DBAs must team up with Internet security experts to build security mechanisms to safeguard data from possible attacks or unauthorized access.

4. Data Backup and Recovery

The DBA must also ensure that the data in the database can be fully recovered in case of physical data loss or loss of database integrity. Data loss can be partial or total. A partial loss is caused by a physical loss of part of the database or when part of the database has lost integrity. A total loss might mean that the database continues to exist but its integrity is entirely lost or that the entire database is physically lost.

Disaster management includes all of the DBA activities designed to secure data availability following a physical disaster or a database integrity failure.

The DBA's Technical Role

The DBA's technical activities include the selection, installation, operation, maintenance, and upgrading of the DBMS and utility software, as well as the design, development, implementation, and maintenance of the application programs that interact with the database.

The technical aspects of the DBA's job are rooted in the following areas of operation:

1. **Evaluating, selecting, and installing the DBMS and related utilities**.
   Selecting the database management system, utility software, and supporting hardware to be used in the organization. The selection plan is based on organization's needs and the features like DBMS model, storage capacity, backup-recovery, concurrency control, performance, portability, cost etc.

2. **Designing and implementing databases and applications.**

   The DBA function usually requires that several people be dedicated to database modeling and design activities. The DBA also works with applications programmers to ensure the quality and integrity of

database design and transactions. Such support services include reviewing the database application design to ensure that transactions are correct, efficient and compliant.

3. Testing and evaluating databases and applications.

The DBA must also provide testing and evaluation services for all of the database and enduser applications. Testing starts with the loading of the tested database. That database contains test data for the applications, and its purpose is to check the data definition and integrity rules of the database and application programs. The testing and evaluation of a database application cover all aspects of the system technical, evaluation of written documentation, observance of standards for naming, documenting, and coding, Data duplication conflicts with existing data, the enforcement of all data validation rules.

4. Operating the DBMS, utilities, and applications.

DBMS operations can be divided into four main areas:

- System support.
- Performance monitoring and tuning.
- Backup and recovery.
- Security auditing and monitoring.

5. Training and supporting users.

Training people to use the DBMS and its tools is included in the DBA's technical activities. The DBA provides technical training in the use of the DBMS and its utilities for the applications programmers.

6. Maintaining the DBMS, utilities, and applications.

Maintenance activities are dedicated to the preservation of the DBMS environment. Periodic DBMS maintenance includes management of the physical or secondary storage devices, upgrading the DBMS and utility software, migration and conversion services for data in incompatible formats or for different DBMS software.

DATABASE MANAGEMENT SYSTEMS
Database Design and E R Model

Database design is the set of procedures or collection of tasks used for organizing the data to implement a database.
It involves several key steps to ensure efficient storage, retrieval, and manipulation of data.

The database design process can be divided into six steps. The ER model is most relevant to the first three steps.

1. **Requirements Analysis:**
    - what data is to be stored in the database,
    - what applications must be built on top of it, and
    - what operations to be performed to meet performance requirements.

2. **Conceptual Database Design**: Based on the information gathered, develop a high-level description of the data to be stored in the database, along with the constraints.

3. **Logical Database Design**: We must choose a DBMS to implement our database design, and convert the conceptual database design into a database schema in the data model of the chosen DBMS.

4. **Schema Refinement**: The fourth step in database design is to analyze the collection of relations in our relational database schema to identify potential problems, and to refine it.

5. **Physical Database Design**: In this step, we consider typical expected workloads that our database must support and further refine the database design to ensure that it meets desired performance criteria.

6. **Application and Security Design**: Any software project that involves a DBMS must consider aspects of the application that go beyond the database itself. Implement security measures such as authorization, authentication, and encryption to protect the database. Enforce data integrity etc.

Entity-Relationship Model:

- An Entity-Relationship Model specifies the enterprise schema that represents the logical structure of the database graphically.

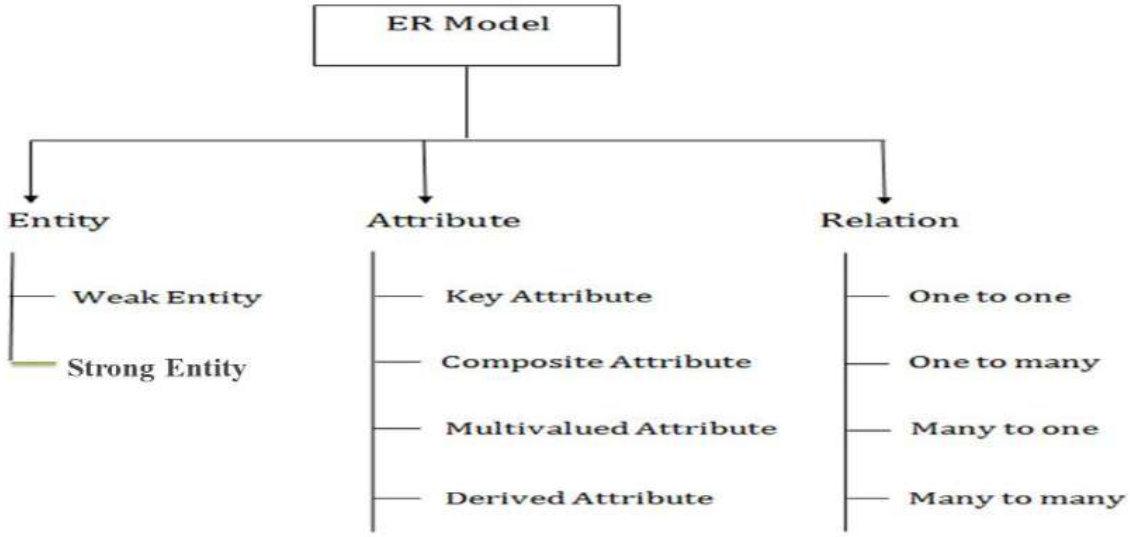
- The **entity-relationship (ER) data model** allows us to describe the data involved in a real-world enterprise in terms of objects and their relationships and is widely used to develop an initial database design.

- It provides useful concepts that allow us to move front an informal description of what users want from their database to a more detailed, precise description that can be implemented in a DBMS.

History of ER models

- Peter Chen proposed ER Diagrams in 1971 to create a uniform convention that can be used as a conceptual modeling tool.
- ER diagrams are used to model and design relational databases.

Components of ER Diagram:



Entity and Entity Set:

- An **entity** is an object that exists and is distinguishable from other objects. Entities are objects of physical(Person, place, thing) or conceptual existence (sales, concert etc.)
  - Examples:
    - Person: PROFESSOR, STUDENT
    - Place: STORE, UNIVERSITY
    - Object: MACHINE, BUILDING
    - Event: SALE, REGISTRATION
- An **entity** is represented with a RECTANGLE

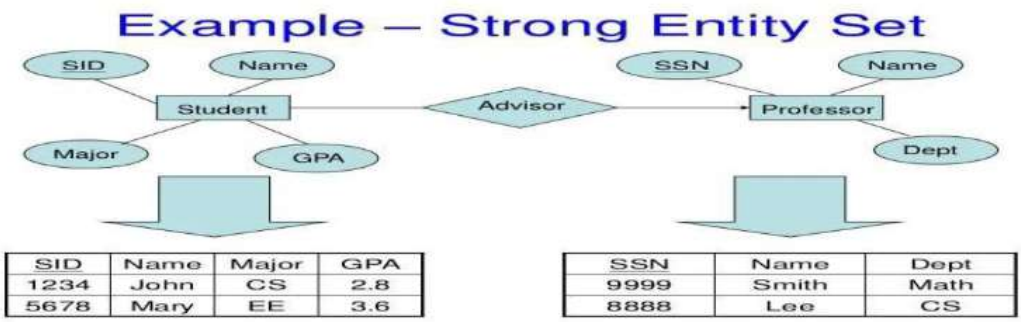
An **entity set** is a set of entities of the same type that share the same properties.
Example: set of all persons, companies, trees, holidays

Strong Entity

The Strong Entity is the one whose existence does not depend on the existence of any other entity in a schema. It is denoted by a single rectangle. A strong entity always has the **primary key** in the set of attributes that describes the strong entity. It indicates that each entity in a strong entity set can be uniquely identified.

Set of similar types of strong entities together forms the Strong Entity Set. A strong entity holds the relationship with the weak entity via an **Identifying Relationship**, which is denoted by **double diamond in the ER diagram**. On the other hands, the relationship

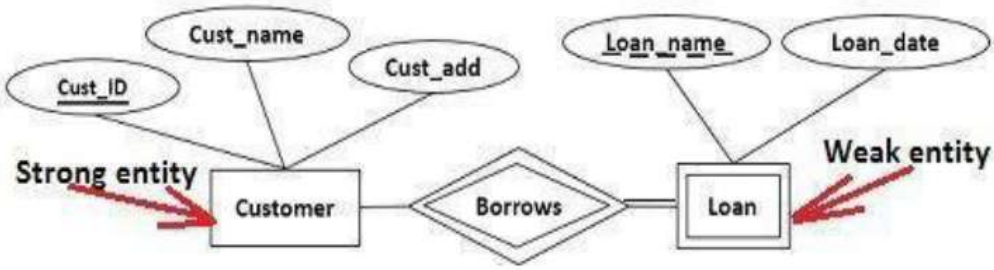
between two strong entities is denoted by a single diamond and it is simply called as a relationship.



Weak Entity

A Weak entity is the one that depends on its owner entity i.e. a strong entity for its existence. A weak entity is denoted by **the double rectangle.** Weak entities do not have the primary key instead it has a partial key that uniquely discriminates the weak entities. The primary key of a weak entity is a composite key formed from the primary key of the strong entity and partial key of the weak entity.

The collection of similar weak entities is called Weak Entity Set. The relationship between a weak entity and a strong entity is always denoted with an Identifying Relationship i.e. double diamond.



ATTRIBUTES

- An entity is represented by a set of attributes that is descriptive properties possessed by all members of an entity set.

  Example:

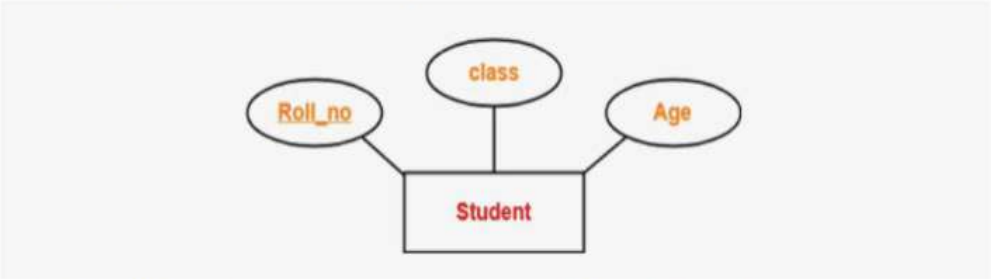
  - *customer* = (*Customer_id, name, street, city, salary*)
  - *movie*= (*title, director, written by, duration, release date*)

Attributes are represented with **ELLIPSE**

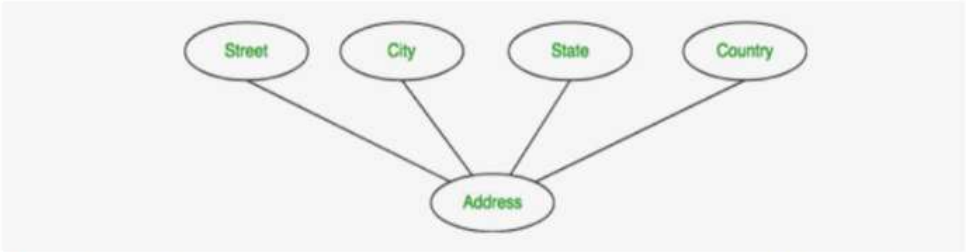
Use **LINES** to link attributes to entities

An attribute can be of many types, here are different types of attributes defined in ER database model:

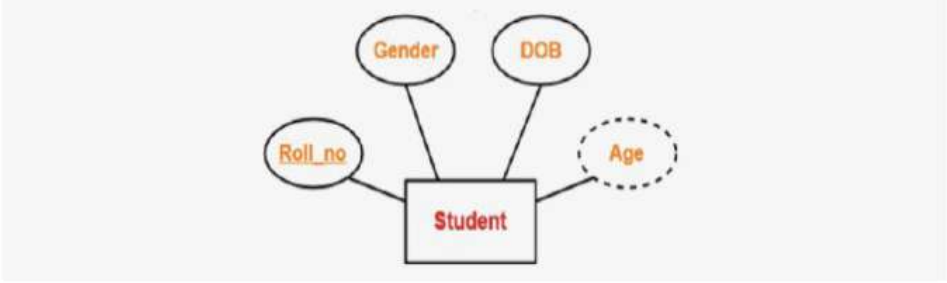
1. **Simple attribute**: The attributes with values that are atomic and cannot be broken down further are simple attributes. For example, student's age.



2. **Composite attribute**: A composite attribute is made up of more than one simple attribute. For example, student's address will contain, house no., street name, pin code etc.

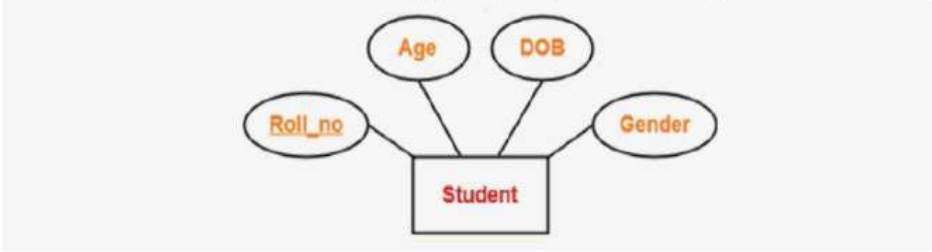


3. **Derived attribute**: These are the attributes which are not present in the whole database management system, but are derived using other attributes. For example, *average age of students in a class*.

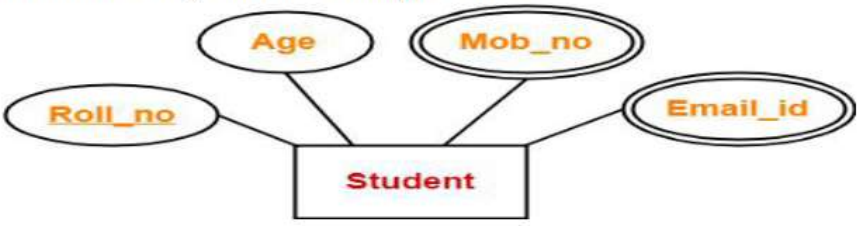


Example: age, and its value is derived from the stored attribute Date of Birth.

4. **Single-valued attribute**: As the name suggests, they have a single value.

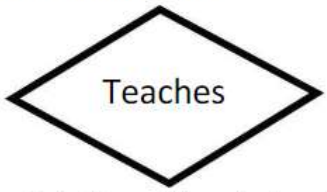


5. **Multi-valued attribute**: They can have multiple values.

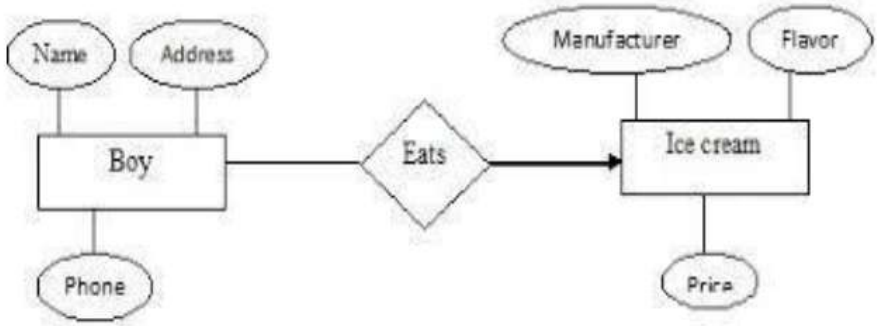


Relationship

- It is an association between two entities. It shows how the two entities are connected.
- Named set of all similar relationships with the same attributes and relating to the same entity types
- Relationship is represented with DIAMOND



- Relationships relate entities within the entity sets involved in the relationship type to each other.



ER – Relationships

Relationship Set: Collection of similar relationships.

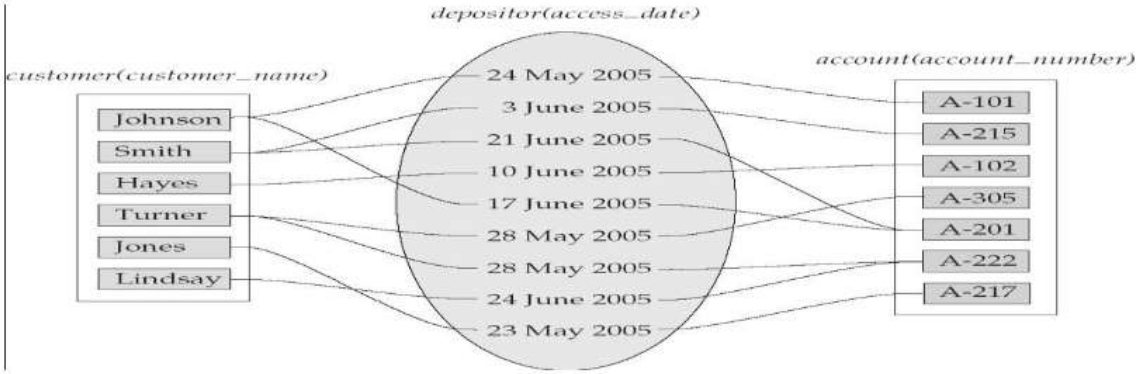
- An attribute can also be property of a relationship set. For instance, the depositor relationship set between entity sets customer and account may have the attribute access-date

321-12-3123	Jones	Main	Harrison
019-28-3746	Smith	North	Rye
677-89-9011	Hayes	Main	Harrison
555-55-5555	Jackson	Dupont	Woodside
244-66-8800	Curry	North	Rye
963-96-3963	Williams	Nassau	Princeton
335-57-7991	Adams	Spring	Pittsfield

customer

L-17	1000
L-23	2000
L-15	1500
L-14	1500
L-19	500
L-11	900
L-16	1300

loan



DATABASE MANAGEMENT SYSTEMS
Database Design and E R Model

Degree of Relationship

A Relationship describes relation between entities. Relationship is represented using diamonds or rhombus.

Degree of relationship represents the number of entity types that associate in a relationship.

Types:

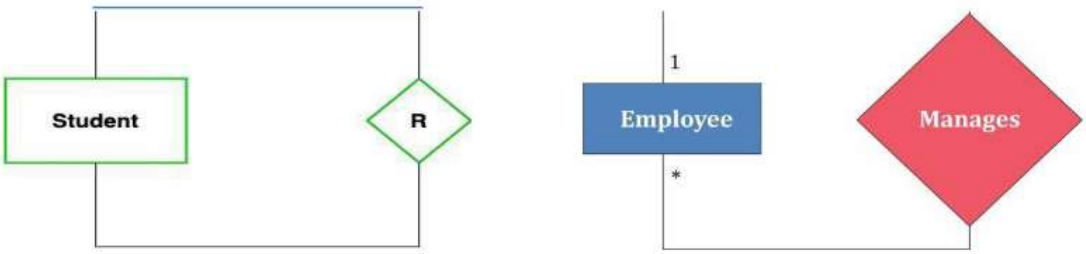
Now, based on the number of linked entity types, we have 4 types of degrees of relationships.

1. Unary Relationship
2. Binary Relationship
3. Ternary Relationship
4. N-ary Relationship

1. Unary Relationship

In this type of relationship, both the associating entity type are the same. in other words, in a relation only one entity set is participating then such type of relationship is known as a unary relationship or Recursive Relationship

Example: In a particular class, we have many students, there are monitors too. So, here class monitors are also students.



2. Binary Relationship

In a Binary relationship, there are two types of entity associates.

In other words, in a relation when two entity sets are participating then such type of relationship is known as a binary relationship. This is the most used relationship and one can easily be converted into a relational table.

This is further divided into three types.

One to One Relationship

This type of relationship is rarely seen in real world

The above example describes that one student can enroll only for one course and a course will also have only one Student. This is not what you will usually see in real-world relationships.

One to Many Relationships

The below example showcases this relationship, which means that 1 student can opt for many courses, but a course can only have 1 student. Sounds weird! This is how it is.

Many to One Relationship

It reflects business rule that many entities can be associated with just one entity. For example, Student enrolls for only one Course but a Course can have many Students.

Many to Many Relationships

A many-to-many relationship occurs when multiple records in a table are associated with multiple records in another table.

3. Ternary Relationship

Relationship of degree three is called Ternary relationship.

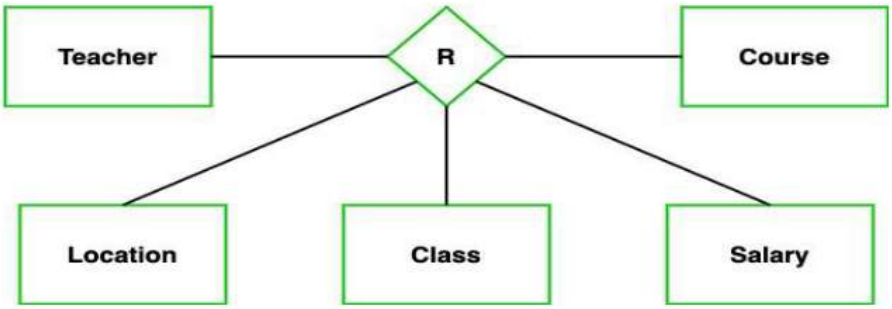
A Ternary relationship involves three entities. In such relationships we always consider two entities together and then look upon the third.

- The above relationship involves 3 entities.
- Company operates in Sector, producing some Products.

4.. n-ary Relationship

In the N-ary relationship, there are n types of entity that associates. There is one limitation of the N-ary relationship, as there are many entities so it is very hard to convert into an entity, rational table.

Example: We have 5 entities Teacher, Class, Location, Salary, Course. So, here five entity types are associating we can say an n-ary relationship is 5.



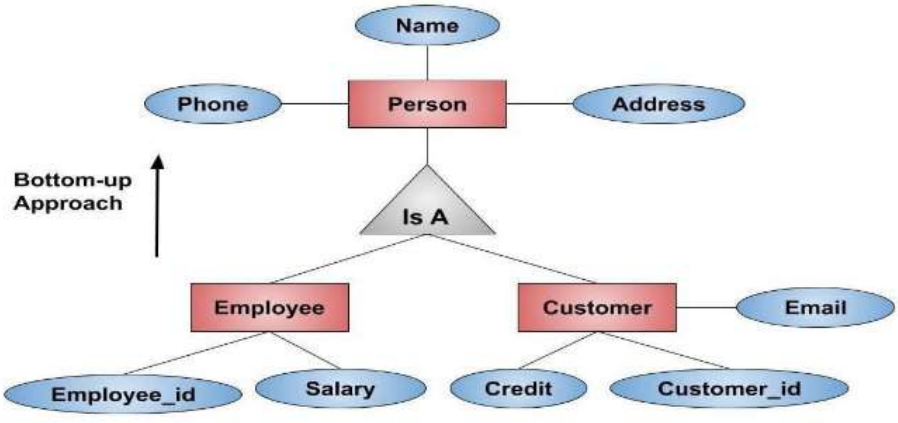
ADDITIONAL FEATURES OF THE ER MODEL

Class Hierarchies:

Using the ER model for bigger data creates a lot of complexity while designing a database model, So in order to minimize the complexity Generalization, Specialization, and Aggregation were introduced in the ER model.

1. **Generalization** –
   Generalization is the process of extracting common properties from a set of entities and creating a generalized entity from it. It is a **bottom-up approach** in which two or more entities can be generalized to a higher-level entity if they have some attributes in common.

ISA (`is a') Hierarchies is used to represent Generalizatio - referred as a superclass-subclass relationship.

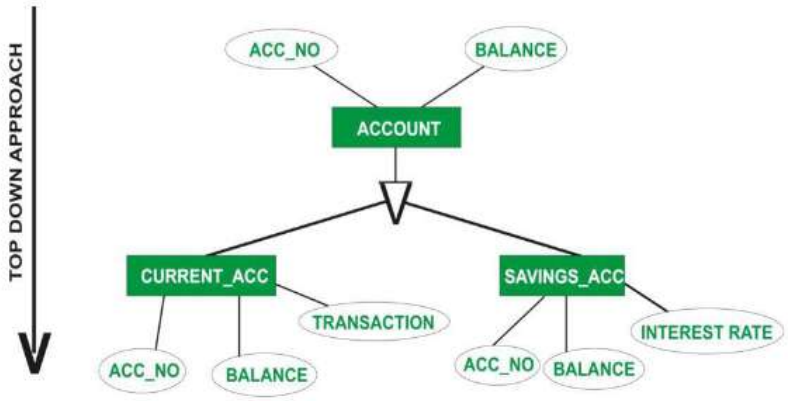


2. **Specialization** –

Specialization is a **top-down approach**, and it is opposite to Generalization.
In specialization, one higher level entity can be broken down into two lower level entities.
Specialization is used to identify the subset of an entity set that shares some distinguishing characteristics.

Normally, the superclass is defined first, the subclass and its related attributes are defined next, and relationship set are then added.

ISA (`is a') Hierarchies is used to represent Specialization - referred as a superclass-subclass relationship.



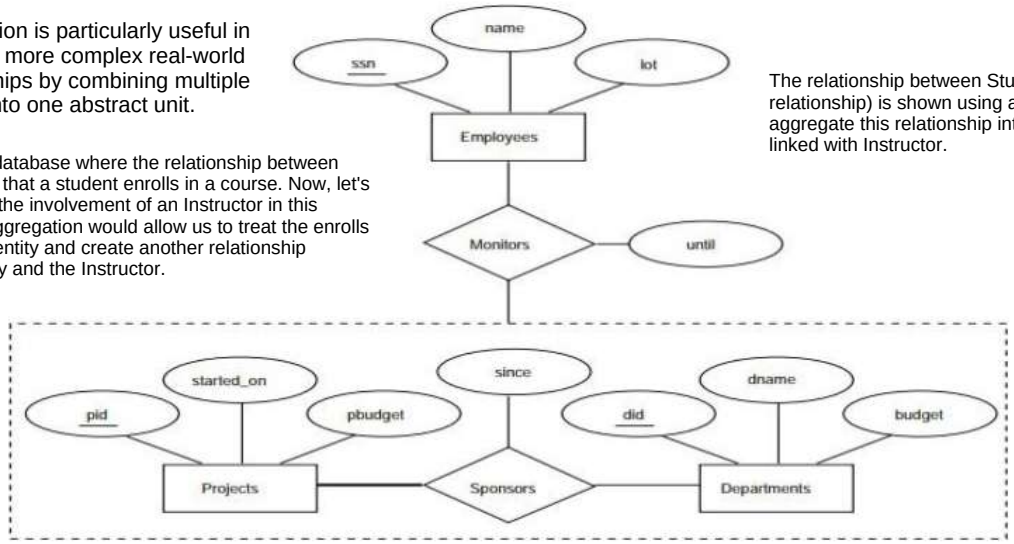
3. Aggregation:

Sometimes we have to model a relationship between a collection of entities and relationships. There is a limitation with E-R model that it cannot express relationships among relationships. Aggregation allows us to indicate that a relationship set (identified through a dashed box) participates in another relationship set.

So aggregation is an abstraction through which relationship is treated as higher level entities.

Aggregation is particularly useful in modeling more complex real-world relationships by combining multiple entities into one abstract unit.

Consider a university database where the relationship between Student and Course is that a student enrolls in a course. Now, let's say we want to model the involvement of an Instructor in this relationship as well. Aggregation would allow us to treat the enrolls relationship as a new entity and create another relationship between this new entity and the Instructor.



The relationship between Student and Course (the enrolls relationship) is shown using a diamond shape, and we then aggregate this relationship into a new entity which can be linked with Instructor.

Participation Constraints:

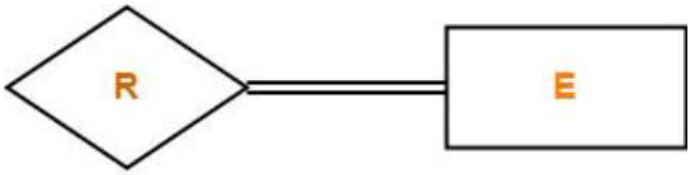
Participation constraints define the least number of relationship instances in which an entity must compulsorily participate.

Types of Participation Constraints:

There are two types of participation constraints- Total Particiapation
Partial Participation

1. Total Participation

- It specifies that each entity in the entity set must compulsorily participate in at least one relationship instance in that relationship set.
- That is why, it is also called **as mandatory participation**.
- Total participation is represented using a double line between the entity set and relationship set.

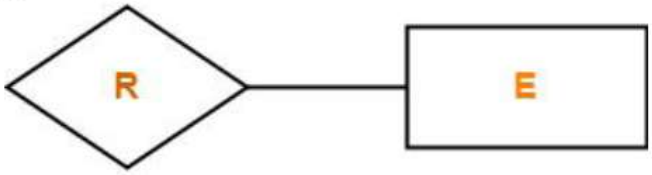


Total Participation

- Double line between the entity set "Student" and relationship set "Enrolled in" signifies total participation.
- It specifies that each student must be enrolled in at least one course.

2. Partial Participation

- It specifies that each entity in the entity set may or may not participate in the relationship instance in that relationship set.
- That is why, it is also called as **optional participation.**
- Partial participation is represented using a single line between the entity set and relationship set.



Partial Participation

- Single line between the entity set "Course" and relationship set "Enrolled in" signifies partial participation.

- It specifies that there might exist some courses for which no enrollments are made.

DATABASE MANAGEMENT SYSTEMS
Conceptual Design Using the ER Model

Developing an ER diagram presents several choices, including the following:

- Should a concept be modeled as an entity or an attribute?
- Should a concept be modeled as an entity or a relationship?
- What are the relationship sets and their participating entity sets?
- Should we use binary or ternary relationships?
- Should we use aggregation?

1. Entity vs. Attribute

While identifying the attributes of an entity set, it is sometimes not clear whether a property should be modeled as an attribute or as an entity set.

Example : Consider the relationship "Employee WorksIn Department", if we want to add address information to the Employee entity set.

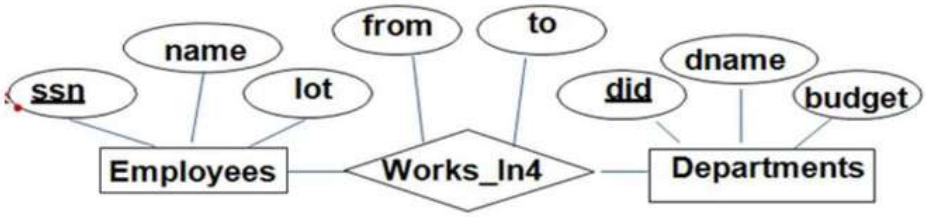
Should *address* be an attribute of Employees or an entity (connected to Employees by a relationship)?

Now this depends upon the use we want to make of address information, and the semantics of the data:

- If we have several addresses per employee, *address* must be an entity (since attributes cannot be set-valued) and an association between employees and addresses using a relationship say Has_address.
- If the structure (city, state, zipcode etc.) is important, e.g., we want to retrieve employees in a given city, *address* must be modeled as an entity with these attributes (since attribute values are atomic).

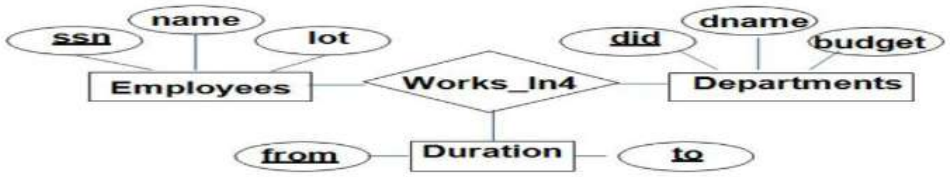
Consider another example below:

In the "Employee WorksIn Department" Relationship, it is possible for an employee to work in a given department over more than one period, which is not possible to accomodate as an attribute.



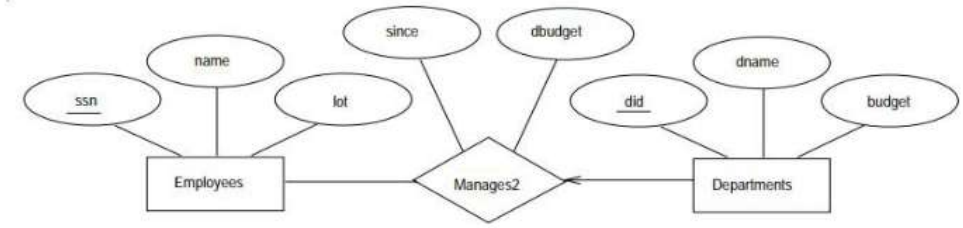
- Works_In4 does not allow an employee to work in a department for two or more periods.

- Similar to the problem of wanting to record several addresses for an employee, We want to record *several values of the descriptive attributes for each instance of this relationship.* Accomplished by introducing new entity set, **Duration.**



2. Entity vs. Relationship

Look at the example below : Consider the relationship set called Manages.



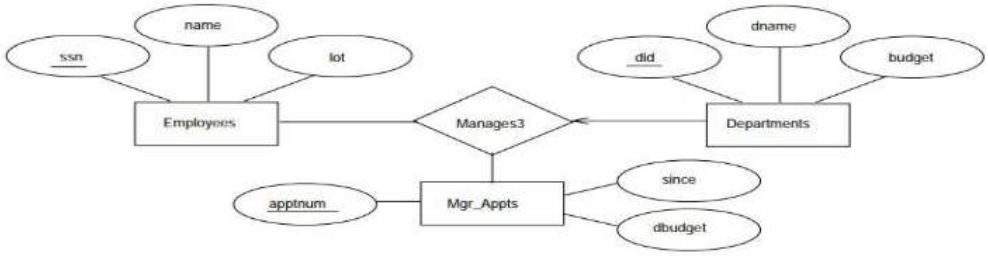
Suppose that each department manager is given a discretionary budget. There is at most one employee managing a department, but a given employee could manage several departments; we store the starting date and discretionary budget for each manager-department pair.

What if a manager manages more than one department and the discretionary budget is a sum of all the departments he manages.

Redundancy: given employee will have the same value in the dbudget field

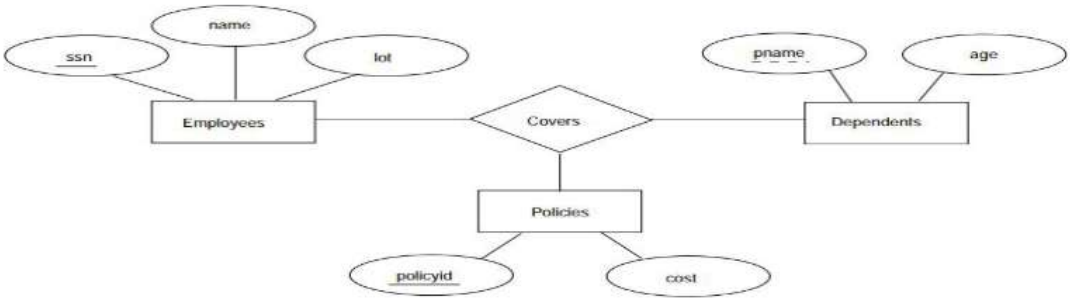
Misleading: Suggests dbudget associated with department - mgr combination.

We can address these problems by associating dbudget with the appointment of the employee as manager of a group of departments. In this approach, we model the appointment as an entity set, say Mgr_Appt, and use a ternary relationship, say Man ages3, to relate a manager, an appointment, and a department.



3. Binary vs. Ternary Relationships

Consider an ER Diagram that models a situation in which an employee can own several policies, each policy can be owned by several employees, and each dependent can be covered by several policies.



If each policy is owned by just 1 employee, and each dependent is tied to the covering policy, the ternary relationship as above is inaccurate.

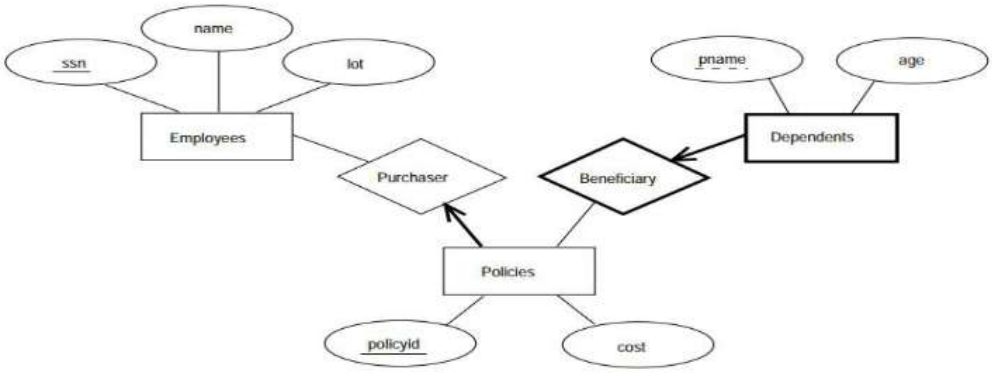
Suppose that we have the following additional requirements:

- A policy cannot be owned jointly by two or more employees.
- Every policy must be owned by some employee.
- each dependent entity is uniquely identified by taking pname in conjunction with the policyid of a policy entity

Solution can be:

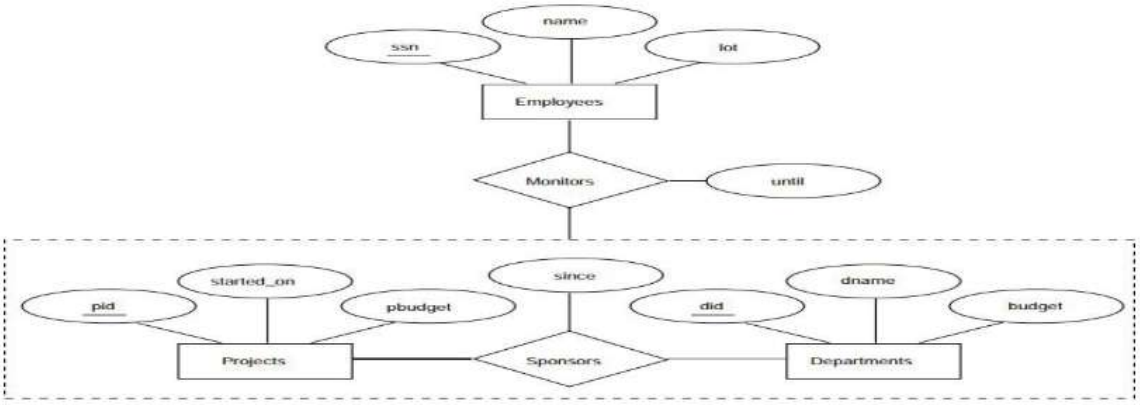
- The first requirement suggests that we impose a key constraint on Policies with respect to Covers, but this constraint has the unintended side effect that a policy can cover only one dependent.
- The second requirement suggests that we impose a total participation constraint on Policies. This solution is acceptable if each policy covers at least one dependent.
- The third requirement forces us to introduce an identifying relationship that is binary

The best way to model this situation is to use two binary relationships, as shown in Figure

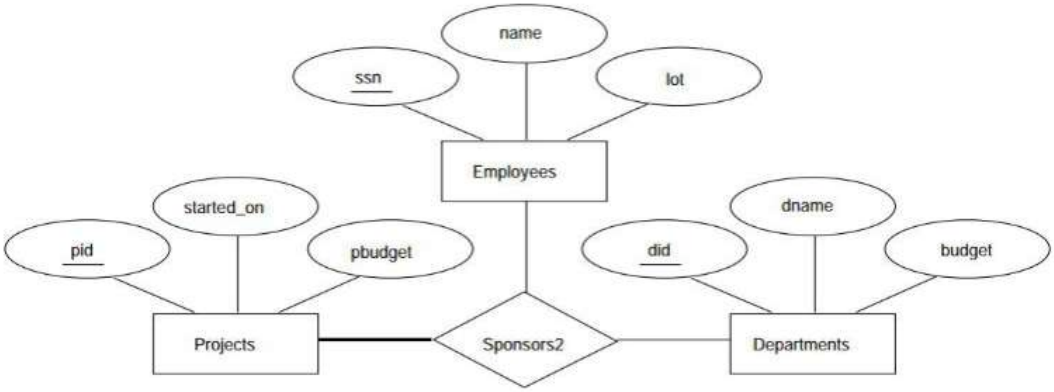


4. Aggregation v/s Ternary relationship

- The choice between using aggregation or ternary relationship is mainly determined by existence of a relationship that relates relationship set to entity set.
- The choice may also be guided by certain integrity constraints that we want to express.



According to the ER diagram shown in Figure above, a project can be sponsored by any number of departments, a department can sponsor one or more projects, and each sponsorship is monitored by one or more employees. If we don"t need to record the „**until'** attribute of Monitors, then we might reasonably use a ternary relationship as shown below without the need for aggregation.



Consider the constraint that each sponsorship (of a project by a department) be mon itored by at most one employee. We cannot express this constraint in terms of the Sponsors2 relationship set. On the other hand, we can easily express the constraint by drawing an arrow from the aggregated relationship Sponsors to the relationship Monitors in the aggregation ER diagram.

DATABASE MANAGEMENT SYSTEMS
SQL – DATA DEFINITION LANGUAGE

Structured query language (SQL) is a programming language for storing and processing information in a relational database.

We can use SQL statements to store, update, remove, search, and retrieve information from the database and also to maintain and optimize database performance.

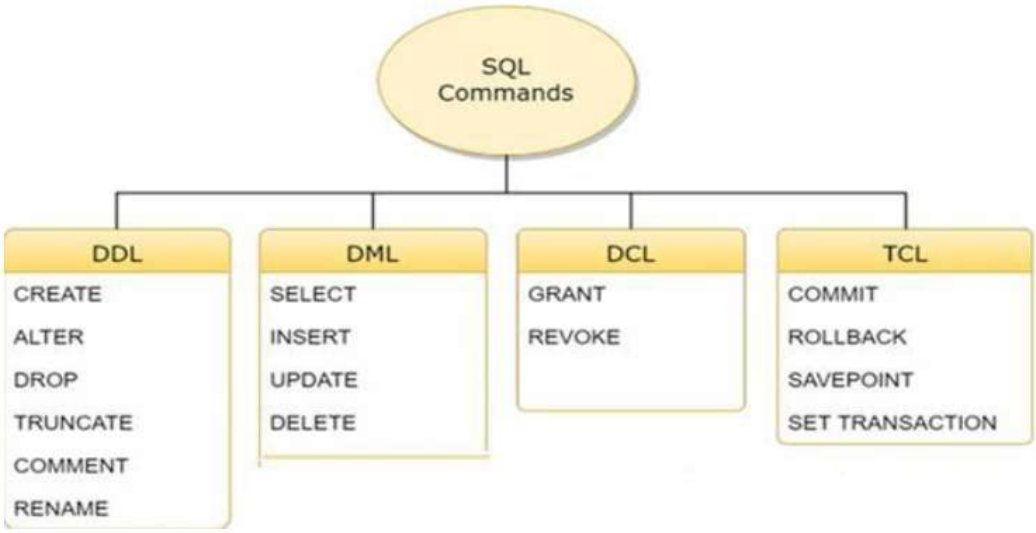
SQL was invented in the 1970s based on the relational data model. It was initially known as the Structured English Query Language (SEQUEL).

This query language became the standard of ANSI in the year of 1986 and ISO in the year of 1987.

Types of SQL Commands:

Here are five types of widely used SQL queries.

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- Data Query Language
- Data Control Language (DCL)
- Transaction Control Language (TCL)



What is DDL?

Data definition language (DDL) refers to SQL commands that design the database structure. i.e. it deals with **database schemas and descriptions**, of how the data should reside in the database.Database engineers use DDL to create and modify database objects based on the business requirements. For example, the database engineer uses the CREATE command to create database objects such as tables, views, and indexes.

CREATE:

To create a database and its objects like (table, index, views, store procedure, function, and triggers)

Syntax: CREATE Objtype Objname;

CREATE DATABASE

Syntax : create database database_name;

Ex: create database testdb;

Use the SHOW statement to find out what databases currently exist on the server:

Syntax: SHOW DATABASES;

Ex: show databases;

We have to select the database on which we want to run the database queries.

syntax : USE DATABASE_NAME;

Ex: use testdb;

Creating a Table:

The CREATE TABLE statement is used to create a new table in a database.

Syntax:

```
CREATE TABLE table_name
( column1 datatype(size),
column2 datatype(size),
column3 datatype(size),
....
);
```

Example: create table emp (eid int(5), ename varchar(15));

The column parameters specify the names of the columns of the table.

The datatype parameter specifies the type of data the column can hold

(e.g. varchar, integer, date, etc.)

Create Table using Another Table :

A copy of an existing table can also be created using CREATE TABLE.

- The new table gets the same column definitions. All columns or specific columns can be selected.
- If you create a new table using an existing table, the new table will be filled with the existing values from the old table.

Syntax:

```
CREATE TABLE new_table_name AS
   SELECT column1, column2,...
   FROM existing_table_name
   WHERE.....;
```

Example: The following SQL creates a new table called "DupCust" (which is a copy of the "Customers" table):

```
CREATE TABLE DupCust AS
SELECT customername, contactno
FROM customers;
```

Data Types in MySQL:

In MySQL there are three main data types: **string, numeric, and date and time**

String Data Type:

The attribute domain includes characters – aphabets, numerals and special characters.

Data type	Description
CHAR(size)	A FIXED length string (can contain letters, numbers, and special characters). The *size* parameter specifies the column length in characters - can be from 0 to 255. Default is 1
VARCHAR(size)	A VARIABLE length string (can contain letters, numbers, and special characters). The *size* parameter specifies the maximum column length in characters - can be from 0 to 65535
BINARY(size)	Equal to CHAR(), but stores binary byte strings. The *size* parameter specifies the column length in bytes. Default is 1

Numeric Data Type:

Data type	Description
BIT(size)	A bit-value type. The number of bits per value is specified in size. The size parameter can hold a value from 1 to 64. The default value for size is 1.
BOOL / BOOLEAN	Zero is considered as false, nonzero values are considered as true.
INT(size) / INTEGER(size)	A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The size parameter specifies the maximum display width (which is 255)
FLOAT(size, d)	A floating point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter. This syntax is deprecated in MySQL 8.0.17, and it will be removed in future MySQL versions
DOUBLE(size, d)	A normal-size floating point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter
DECIMAL(size, d) / DEC(size, d)	An exact fixed-point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter. The maximum number for size is 65. The maximum number for d is 30. The default value for size is 10. The default value for d is 0.

Date and Time Datatype:

Data type	Description
DATE	A date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31'
DATETIME(fsp)	A date and time combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. Adding DEFAULT and ON UPDATE in the column definition to get automatic initialization and updating to the current date and time
TIMESTAMP(fsp)	A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC.
TIME(fsp)	A time. Format: hh:mm:ss. The supported range is from '-838:59:59' to '838:59:59'
YEAR	A year in four-digit format. Values allowed in four-digit format: 1901 to 2155, and 0000. MySQL 8.0 does not support year in two-digit format.

DROP

Drops commands remove tables and databases from RDBMS.

To drop a Table

Syntax: DROP TABLE tablename;

Ex: DROP TABLE emp ;

To drop a database

syntax: DROP DATABASE databasename;

Ex: Drop database kmitcsma;

TRUNCATE:

This command is used to delete all the rows from the table and free the space containing the table.

Syntax:

TRUNCATE TABLE table_name;

Example:

TRUNCATE table students

ALTER:

1. The ALTER TABLE statement is used to modify the structure of table, i.e. **add, delete, or modify columns** in an existing table.
2. The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

   ALTER- ADD
   ALTER- MODIFY
   ALTER-DROP

To add a new column to the Table :

Syntax:

ALTER TABLE table_name ADD (columnname-1 datatype(size), columnname-2 datatype(size)….,columnname-n datatype(size));

Ex: Alter Table emp add (dob date, mobile int(10));

To Drop a column from an existing table:

Syntax:

ALTER TABLE tablename DROP column columnname;

Ex: alter table emp drop column mobile;

To modify a column in the Table :

Syntax:

ALTER TABLE table_name MODIFY columnname-1 datatype(size);

Ex: Alter Table emp modify ename varchar(25);

To add a constraint:

Syntax:

ALTER TABLE tablename ADD constrainttype (columnname);

Ex: alter table emp add primary key(eid);

RENAME:

Renaming the existing table:

RENAME TABLE table_name to new table_name;

Ex:

mysql> rename table persons to emp;
Query OK, 0 rows affected (3.17 sec)

Renaming the column name in an existing table:

ALTER TABLE table_name
RENAME COLUMN old_name TO new_name;

DATABASE MANAGEMENT SYSTEMS
DATA MANIPULATION LANGUAGE

SQL stands for Structured Query Language. It is used for storing and managing data in Relational Database Management System (RDBMS).

It is a standard language for Relational Database System. It enables a user to create, read, update and delete relational databases and tables.

All the RDBMS like MySQL, Informix, Oracle, MSAccess and SQL Server use SQL as their standard database language.

SQL allows users to query the database in a number of ways, using English-like statements.

NOTE:
Structure query language is not case sensitive. Generally, keywords of SQL are written in uppercase.
Statements of SQL are dependent on text lines. We can use a single SQL statement on one or multiple text line.

What is SQL Process?

When an SQL command is executing for any RDBMS, then the system figure out the best way to carry out the request and the SQL engine determines that howto interpret the task.

In the process, various components are included. These components can be optimization Engine, Query-engine, Query dispatcher, classic query-engine, etc.

What is Advantages of SQL?

- High speed
- No coding needed
- Well defined standards
- Portability
- Interactive language
- Multiple data view

DATA MANIPULATION LANGUAGE

A DML (data manipulation language) refers to a computer programming language that allows you to add (insert), delete (delete), and alter (update) data in a database. A DML is typically a sublanguage of a larger database language like SQL, A DML (data manipulation language) is a group of computer languages that provide commands for manipulating data in databases.

DDL vs DML Commands

DDL	DML
Used to define database objects like tables, indexes, views, etc.	Used to manipulate data within the database.
Examples of DDL statements include CREATE, ALTER, and DROP.	Examples of DML statements include SELECT, INSERT, UPDATE, and DELETE.
Changes made using DDL affect the structure of the database.	Changes made using DML affect the data stored in the database.
DDL statements are not transactional, meaning they cannot be rolled back.	DML statements are transactional, meaning they can be rolled back if necessary.
DDL statements are usually executed by a database administrator.	DML statements are executed by application developers or end-users.
DDL statements are typically used during the design and setup phase of a database.	DML statements are used during normal operation of a database.
Examples of DDL statements: CREATE TABLE, DROP TABLE, ALTER TABLE, CREATE INDEX, etc.	Examples of DML statements: SELECT, INSERT, UPDATE, DELETE, etc.

DML Commands

Command	Description
INSERT	Used to insert new data records or rows in the database table
UPDATE	Used to set the value of a field or column for a particular record to a new value
DELETE	Used to remove one or more rows from the database table
SELECT	Used to retrieve data from one or more tables

1. INSERT

INSERT commands in SQL are used to insert data records or rows in a database table.
In an INSERT statement, we specify both the column_names for which the entry has to be made along with the data value that has to be inserted.

Insert can be used in various forms.

1. **Values only** - When the data is inserted into all fields of the table, then use values only and maintain one-to-one correspondence (the no. of values and the order should match with the fields defined in the table).

   Syntaxa:

   INSERT INTO table_name VALUES (value1, value2, value3, ...)
   By VALUES, we mean the value of the corresponding columns.

Example:

```
INSERT INTO customers
        VALUES ('1006','2020-03-04',3200,'Sukesh Kumar', 'DL', '1008');
```

2. **Column names and Values** - When the data is inserted into selected fields of the table or in user defined order, then the column names would be specified before the values.

Syntax:

INSERT INTO table_name (column_name_1, column_name_2, column_name_3, ...)
VALUES (value1, value2, value3, ...)

By VALUES, we mean the value of the corresponding columns.

Example:

```
INSERT INTO customers( customer_id, sale_date, sale_amount, salesperson, order_id)
        VALUES (1005,'2019-12-12',4200,'R K Rakesh','1007');
```

3. **Multiple rows** – multiple rows (records) can be inserted using a single insert statement by including the input values as sets separated by comma.

Syntax:

INSERT INTO table_name (column_name_1, column_name_2, ...)
VALUES (value1, value2, ...) , (value1, value2, ...), …

Example:

```
INSERT INTO customers( customer_id, sale_amount, salesperson, order_id)
        VALUES (1006,8200,'R Mukesh','1008'), (1007, 1000, 'R Nagesh', '1009');
```

4. **Data from other Tables** - The command copies data from one table and inserts it into another table. It requires that data types in source and target tables match.

Syntax

Copy all columns from one table to another table:

INSERT INTO ***table2***
SELECT * FROM ***table1***
[WHERE ***condition]*;**

Copy only some columns from one table into another table:

INSERT INTO ***table2*** **(*column1*, *column2*, *column3*, ...)**
SELECT ***column1*, *column2*, *column3*, ...* **FROM** ***table1***
[WHERE ***condition]*** **;**

Example:

```
INSERT INTO Customers (CustomerName, City, Country)
        SELECT SupplierName, City, Country FROM Suppliers;
```

2. UPDATE

UPDATE command or statement is used to modify the value of an existing column in a database table. The UPDATE statement can be used to update single or multiple columns on the basis of our specific needs.

Syntax:;

```
UPDATE table_name
SET column_name_1 = value1, column_name_2 = value2, ...
[ WHERE condition ];
```

Example:

1. UPDATE customers SET store_state = 'DL' WHERE store_state = 'NY';
   In this example, we have modified the value of store_state for a record where store_state was 'NY' and set it to a new value 'DL'.

2. UPDATE employees SET bonus = 5000;
   In this example, all the rows of the employees table will set the bonus value to 5000.

3. DELETE

DELETE statement in SQL is used to remove one or more rows from the database table. It does not delete the data records permanently. We can always perform a rollback operation to undo a DELETE command.
With DELETE statements we can use the WHERE clause for filtering specific rows.

Syntax :

```
DELETE FROM table_name [ WHERE condition ];
```

Example :

1. DELETE FROM customers WHERE store_state = 'MH' AND customer_id = '1001';
   This will delete the details of customer with id=1001 whose store_state is 'MH'

2. DELETE FROM customers;
   This will delete all the records from the customer table. This command is also equivalent to the TRUNCATE command in DDL.

4. SELECT

A Query (SELECT) in SQL is a statement to retrieve data from one or more tables.
The data returned is stored in a result table, called the result-set.

It can have six clauses, but only the first two (select and from) are mandatory. The clauses are specified in the following order.

Syntax:

Select ***<List of Columns and expressions (usually involving columns)>***

From ***<List of Tables & Join Operators>***

[Where ***<List of Row conditions joined together by And, Or, Not>]***

[SGroup By ***<list of grouping columns>]***

[Having ***<list of group conditions connected by And, Or, Not >]***

[Order By ***<list of sorting specifications>]***

Examples:

Syntax to select all rows and columns:

Syntax: SELECT * FROM tablename;

Example: select * from employee;

Syntax to select selected columns and all rows:

Syntax: SELECT column1, column2... from tablename;
Here, column1, column2, ... are the field names of the table you want to select data from.

Example: select eid, ename afrom employee;

Syntax to select selected rows and selected columns:

Syntax: **SELECT column1, column2.. from tablename where condition;**

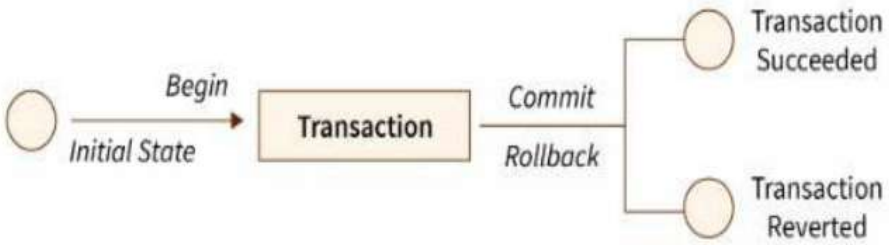
Example: select eid, ename from employee where eid=101;

DATABASE MANAGEMENT SYSTEMS
Transaction Control Language (TCL) Commands

TCL stands for Transaction Control Language in SQL. Transaction Control Language (TCL) is a set of special commands that deal with the transactions within the database. Basically, they are used to manage transactions within the database. TCL commands ensure the integrity and consistency of data by allowing to control the behavior of transactions.

A transaction is a unit of work that is performed against a database in SQL. In other words, a transaction is a single, indivisible database action.

In SQL, each transaction begins with a particular set of task and ends only when all the tasks in the set is completed successfully. However, if any (or a single) task fails, the transaction is said to fail.



TCL Commands – Save point Commit and Roll back:

COMMIT command:

COMMIT command is used to permanently save any transaction into the database.
When we use any DML command like INSERT, UPDATE or DELETE, the changes made by these commands are not permanent, until the current session is closed, the changes made by these commands can be rolled back.
To avoid that, we use the COMMIT command to mark the changes as permanent.

SYNTAX : COMMIT;

ROLLBACK command :

This command restores the database to last commited state.
If we have used the UPDATE command to make some changes into the database or inserted a record, and realise that those changes were not required, then we can use the ROLLBACK command to rollback those changes, if they were not commited using the COMMIT command.

SYNTAX : ROLLBACK;

It is also used with SAVEPOINT command to jump to a savepoint in an ongoing transaction. The savepoints are like checkpoints, they temporarily save a transaction up to where the transaction can be rolled back

SYNTAX: ROLLBACK TO savepoint_name;

SAVEPOINT command:

SAVEPOINT command is used to temporarily save a transaction so that you can rollback to that point whenever required.

SYNTAX: SAVEPOINT savepoint_name;

In short, using this command we can name the different states of our data in any table and then rollback to that state using the ROLLBACK command whenever required

Example : Using Savepoint and Rollback:

Following is the table class,

id	name
1	Abhi
2	Adam
4	Alex

MySQL, set the autocommit option as shown below

SET AUTOCOMMIT=0;

Let us use some SQL queries on the above table and see the results.

- INSERT INTO class VALUES(5, 'Rahul');
- COMMIT;
- UPDATE class SET name = 'Abhijit' WHERE id = '5';
- SAVEPOINT A;
- INSERT INTO class VALUES(6, 'Chris');
- SAVEPOINT B;
- INSERT INTO class VALUES(7, 'Bravo');
- SAVEPOINT C
- SELECT * FROM class;

The resultant table will look like,

id	Name
1	Abhi
2	Adam

4	Alex
5	Abhijit
6	Chris
7	Bravo

Now let's use the ROLLBACK command to roll back the state of data to the savepoint B.

- ROLLBACK TO B;
- SELECT * FROM class;

The output now – current table would look like

id	Name
1	Abhi
2	Adam
4	Alex
5	Abhijit
6	Chris

Now let's again use the ROLLBACK the state of data to the savepoint A

- ROLLBACK TO A;
- SELECT * FROM class;

Now the table will look like,

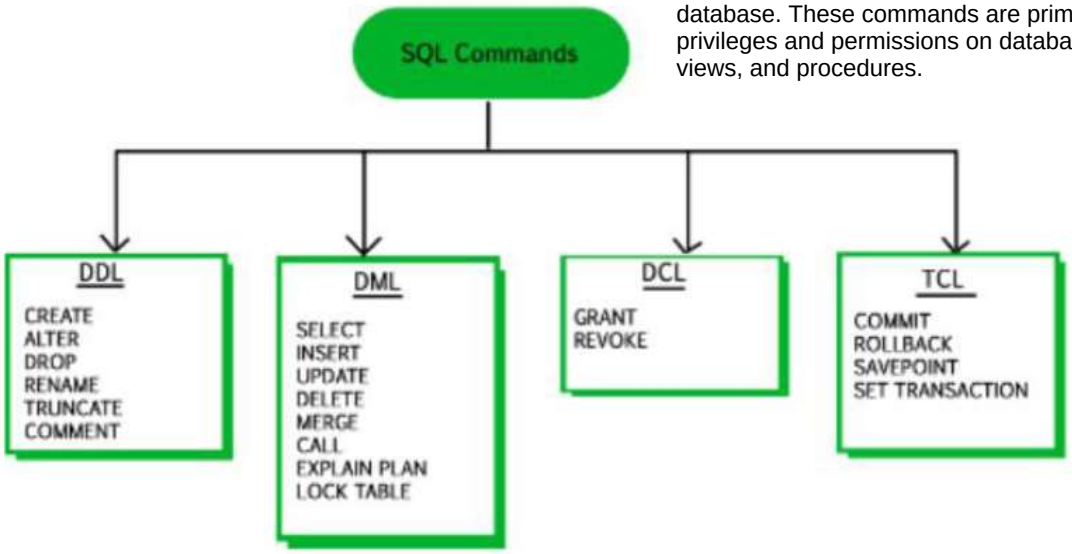
id	name
1	Abhi
2	Adam
4	Alex
5	Abhijit

DATABASE MANAGEMENT SYSTEMS
Data Control Language

Data Controlling Language:

Data Controlling Language (DCL) helps users to retrieve and modify the data stored in the database with some specified queries. Grant and Revoke belong to these types of commands of the Data controlling Language. DCL is a component of SQL commands.

In the context of Database Management Systems (DBMS), DCL stands for Data Control Language. It consists of commands used to control access to data stored in a database. These commands are primarily used for setting privileges and permissions on database objects like tables, views, and procedures.



Two types of DCL commands can be used by the user in SQL. These commands are useful, especially when several users access the database. It enables the administrator to manage access control. The two types of DCL commands are as follows:

- GRANT
- REVOKE

Grant Privileges on Table

This command allows the administrator to assign particular privileges or permissions over a database object, such as a table, view, or procedure.

It enables system administrators to assign privileges and roles to the MySQL user accounts so that they can use the assigned permission on the database whenever required.

Syntax

```
GRANT privilege_name(s)
ON object
TO user_account_name;
```

Parameter	Description
privilege_name(s)	**It specifies the access rights or grant privilege to user accounts. If we want to give multiple privileges, then use a comma operator to separate them.**
object	**It determines the privilege level on which the access rights are being granted. It means granting privilege to the table; then the object should be the name of the table.**
user_account_name	**It determines the account name of the user to whom the access rights would be granted.**

The privileges to assign are listed below. It can be any of the following values:

Privilege	Description
SELECT	Ability to perform SELECT statements on the table.
INSERT	Ability to perform INSERT statements on the table.
UPDATE	Ability to perform UPDATE statements on the table.
DELETE	Ability to perform DELETE statements on the table.
REFERENCES	Ability to create a constraint that refers to the table.
ALTER	Ability to perform ALTER TABLE statements to change the table definition.
ALL	ALL does not revoke all permissions for the table. Rather, it revokes the ANSI-92 permissions which are SELECT, INSERT, UPDATE, DELETE, and REFERENCES.

Object

The name of the database objects that you are granting permissions for. In the case of granting privileges on a table, this would be the table name.

User_account_name

The name of the user who will be granted these privileges.

Example

Let's look at some examples of how to grant privileges on tables in SQL Server.

For example, if you wanted to grant SELECT, INSERT, UPDATE, and DELETE privileges on a table called employees to a user name smithj, you would run the following GRANT statement:

GRANT SELECT,INSERT,UPDATE, DELETE ON employees TO smithj;

You can also use the ALL keyword to indicate that you wish to grant the ANSI-92 permissions

(ie: SELECT, INSERT, UPDATE, DELETE, and REFERENCES) to a user named smithj.

GRANT ALL ON employees TO smithj;

If you wanted to grant only SELECT access on the employees table to all users, you could grant the privileges to the public role.

GRANT SELECT ON employees TO public;

Revoke Privileges on Table:

As the name suggests, revoke is to take away.

The REVOKE command enables the database administrator to remove the previously provided privileges or permissions from a user over a database or database object, such as a table, view, or procedure.

The REVOKE commands prevent the user from accessing or performing a specific operation on an element in the database.

In simple language, the REVOKE command terminates the ability of the user to perform the mentioned SQL command in the REVOKE query on the database or its component.

The primary reason for implementing the REVOKE query in the database is to ensure the data's security and integrity.

Syntax

The syntax for revoking privileges on a table in SQL Server is:

REVOKE privileges
ON object
FROM user_account_name;

Example

Let's look at some examples of how to revoke privileges on tables in SQL Server.

For example, if you wanted to revoke DELETE privileges on a table called employees from a user named anderson, you would run the following REVOKE statement:

```
REVOKE DELETE ON employees FROM anderson;
```

If you wanted to revoke ALL ANSI-92 permissions

(ie: SELECT, INSERT, UPDATE, DELETE, and REFERENCES) on a table for a user named anderson, you could use the ALL keyword as follows:

```
REVOKE ALL ON employees FROM anderson;
```

If you had granted SELECT privileges to the public role (ie: all users) on the employees table and you wanted to revoke these privileges, you could run the following REVOKE statement:

```
REVOKE SELECT ON employees FROM public;
```

Differences between Grant and Revoke commands:

S.NO	Grant	Revoke
1	This DCL command grants permissions to the user on the database objects.	This DCL command removes permissions if any granted to the users on database objects.
2	It assigns access rights to users.	It revokes the user access rights of users.
3	For each user you need to specify the permissions.	If access for one user is removed; all the particular permissions provided by that users to others will be removed.
4	When the access is decentralized granting permissions will be easy.	If decentralized access removing the granted permissions is difficult.

Advantages of DCL commands:

- **Security:** the primary reason to implement DCL commands in the database is to manage the access to the database and its object between different users. It ensures the security and integrity of the data stored in the database.
- **Granular control:** DCL commands provide granular control to the data administrator over the database. Thus, it enables the admin to create different levels of access to the database.
- **Flexibility:** The data administrator can implement DCL commands on specific commands and queries in the database. It allows the administrator to grant or revoke user permissions and privileges as per their needs. It provides flexibility to the administrator that allows them to manage access to the database.

Disadvantages of DCL commands:

- **Complexity:** It increases the complexity of database management. If many users are accessing the database, keeping track of permission and privileges provided to every user in the database becomes very complex.
- **Time-Consuming:** It is time-consuming to assign the permissions and privileges to each user separately.
- **Risk of human error:** Human administrators execute DCL commands and can make mistakes in granting or revoking privileges. Thus, giving unauthorized access to data or imposing unintended restrictions on access.
- **Lack of audit trail:** There may be no built-in mechanism to track changes to privileges and permissions over time. Thus, it is extremely difficult to determine who has access to the data and when that access was granted or revoked.