# Unit – I

**DATA:**

Data is a collection of facts, figures and statistics related to an object. For example: Students fill an admission form when they get admission in college. The form consists of raw facts about the students. These raw facts are student's name, father name, address etc. The purpose of collecting this data is to maintain the records of the students during their study period in the college.

**INFORMATION:**

Processed data is called information. OR The manipulated and processed form of data is called information. For example: Data collected from census is used to generate different type of information. The government can use it to determine the literacy rate in the country. Government can use the information in important decision to improve literacy rate.

**ADVANTAGES OF THE DBMS:**

1. **Improved data sharing:**

The DBMS helps create an environment in which end users have better access to more and better-managed data. Such access makes it possible for end users to respond quickly to changes in their environment.

**2. Improved data security:**

The more users access the data, the greater the risks of data security breaches. Corporations invest considerable amounts of time, effort, and money to ensure that corporate data are used properly. A DBMS provides a framework for better enforcement of data privacy and security policies.

**3. Data integration:**

Wider access to well-managed data promotes an integrated view of the organization's operations and a clearer view of the big picture. It becomes much easier to see how actions in one segment of the company affect other segments.

**4. Minimized data inconsistency:**

Data inconsistency exists when different versions of the same data appear in different places. For example, data inconsistency exists when a company's sales department stores a sales representative's name as "Bill Brown" and the company's personnel department stores that same person's name as "William G. Brown," or when the company's regional sales office shows the price of a product as $45.95 and its national sales office shows the same product's price as $43.95. The probability of data inconsistency is greatly reduced in a properly designed database.

**5. Improved data access:**

The DBMS makes it possible to produce quick answers to ad hoc queries. From a database perspective, a query is a specific request issued to the DBMS for data manipulation—for example, to read or update the data. Simply put, a query is a question, and an ad hoc query is a spur-of-the-moment question. The DBMS sends back an answer (called the query result set) to the application. For example, end users, when dealing with large amounts of sales data, might want quick answers to questions (ad hoc queries) such as:

- What was the dollar volume of sales by product during the past six months?
- What is the sales bonus figure for each of our salespeople during the past three months?
- How many of our customers have credit balances of $3,000 or more?

**6. Improved decision making:**

Better-managed data and improved data access make it possible to generate better-quality information, on which better decisions are based. The quality of the information generated depends on the quality of the underlying data. Data quality is a comprehensive approach to promoting the accuracy, validity, and timeliness of the data. While the DBMS does not guarantee data quality, it provides a framework to facilitate data quality

initiatives.

**7. Increased end-user productivity:**

The availability of data, combined with the tools that transform data into usable information, empowers end users to make quick, informed decisions that can make the difference between success and failure in the global economy.

**8. Controlling Data Redundancy:**

In non-database systems (traditional computer file processing), each application program has its own files. In this case, the duplicated copies of the same data are created at many places. In DBMS, all the data of an organization is integrated into a single database. The data is recorded at only one place in the database and it is not duplicated. For example, the dean's faculty file and the faculty payroll file contain several items that are identical. When they are converted into database, the data is integrated into a single database so that multiple copies of the same data are reduced to-single copy.

In DBMS, the data redundancy can be controlled or reduced but is not removed completely. Sometimes, it is necessary to create duplicate copies of the same data items in order to relate tables with each other.

By controlling the data redundancy, you can save storage space. Similarly, it is useful for retrieving data from database using queries.

**8. Backup and Recovery Procedures:**

In a computer file-based system, the user creates the backup of data regularly to protect the valuable data from damaging due to failures to the computer system or application program. It is a time consuming method, if volume of data is large. Most of the DBMSs provide the 'backup and recovery' sub-systems that automatically create the backup of data and restore data if required. For example, if the computer system fails in the middle (or end) of an update operation of the program, the recovery sub-system is responsible for making sure that the database is restored to the state it was in before the program started executing.

**DISADVANTAGES OF DATABASE**

**1. Increased costs:**

Database systems require sophisticated hardware and software and highly skilled personnel. The cost of maintaining the hardware, software, and personnel required to operate and manage a database system can be substantial. Training, licensing, and regulation compliance costs are often overlooked when database systems are implemented.

**2. Management complexity:**

Database systems interface with many different technologies and have a significant impact on a company's resources and culture. The changes introduced by the adoption of a database system must be properly managed to ensure that they help advance the company's objectives. Given the fact that database systems hold crucial company data that are accessed from multiple sources, security issues must be assessed constantly.

**3. Maintaining currency:**

To maximize the efficiency of the database system, you must keep your system current. Therefore, you must perform frequent updates and apply the latest patches and security measures to all components. Because database technology advances rapidly, personnel training costs tend to be significant. Vendor dependence. Given the heavy investment in technology and personnel training, companies might be reluctant to change database vendors. As a consequence, vendors are less likely to offer pricing point advantages to existing customers, and those customers might be limited in their choice of database system components.

**4. Frequent upgrade/replacement cycles:**

DBMS vendors frequently upgrade their products by adding new functionality. Such new features often come bundled in new upgrade versions of the software. Some of these versions require hardware upgrades. Not only do the upgrades themselves cost money, but it also costs money to train database users and administrators to properly use and manage the new features.

**5. Appointing Technical Staff:**
The trained technical persons such as database administrator and application programmers etc are required to handle the DBMS. You have to pay handsome salaries to these persons. Therefore, the system cost increases.

**SCHEMA**
A schema is a collection of named objects. Schemas are generally stored in a data dictionary. Although a schema is defined in text database language, the term is often used to refer to a graphical depiction of the database structure. In relational database technology, schemas provide a logical classification of objects in the database. Some of the objects that a schema may contain include tables, views, aliases, indexes, triggers, and structured types.

**INSTANCES**
The data in the database at a particular moment of time is called an instance or a database state. In a given instance, each schema construct has its own current set of instances. Many instances or database states can be constructed to correspond to a particular database schema. Every time we update (i.e., insert, delete or modify) the value of a data item in a record, one state of the database changes into another state. The following figure shows an instance of the ITEM relation in a database schema.

| ITEM | | |
|---|---|---|
| **ITEM-ID** | **ITEM_DESC** | **ITEM_COST** |
| 1111A | Nutt | 3 |
| 1112A | Bolt | 5 |

**DATABASE ADMINISTRATOR**
the people responsible for managing databases are called database administrators. Each database administrator, dubbed DBA for the sake of brevity may be engaged in performing various database manipulation tasks such as archiving, testing, running, security control etc. all related to the environmental side of the databases.

**The responsibilities of a DBA**

1. Designing the logical and physical schemas, as well as widely-used portions of the external schema.
2. Security and authorization.
3. Data availability and recovery from failures.
4. Database tuning: The DBA is responsible for evolving the database, in particular the conceptual and physical schemas to ensure adequate performance as user requirements change.

**USERS IN DBMS**
Users are of 4 types:
1. Application programmers or Ordinary users
2. End users
3. Database Administrator (DBA)
4. System Analyst

**1. Application programmers or Ordinary users**: These users write application programs to interact with the database. Application programs can be written in some programming language such a COBOL, PL/I, C++, JAVA or some higher level fourth generation language. Such programs access the database by issuing the appropriate request, typically a SQL statement to DBMS.

**2. End Users:** End users are the users, who use the applications developed. End users need not know about the working, database design, the access mechanism etc. They just use the system to get their task done. End users are of two types:

a) Direct users b) Indirect users

**a) Direct users:** Direct users are the users who se the computer, database system directly, by following instructions provided in the user interface. They interact using the application programs already developed, for getting the desired result. E.g. People at railway reservation counters, who directly interact with database.

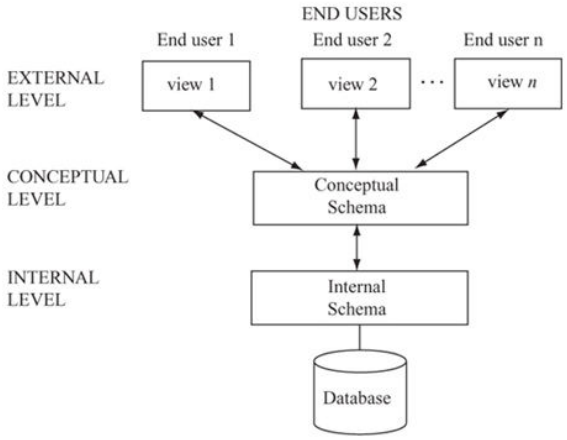
**b) Indirect users:** Indirect users are those users, who desire benefit form the work of DBMS indirectly. They use the outputs generated by the programs, for decision making or any other purpose. They are just concerned with the output and are not bothered about the programming part.

**3. Database Administrator (DBA):** Database Administrator (DBA) is the person which makes the strategic and policy decisions regarding the data of the enterprise, and who provide the necessary technical support for implementing these decisions. Therefore, DBA is responsible for overall control of the system at a technical level. In database environment, the primary resource is the database itself and the secondary resource is the DBMS and related software administering these resources is the responsibility of the Database Administrator (DBA).

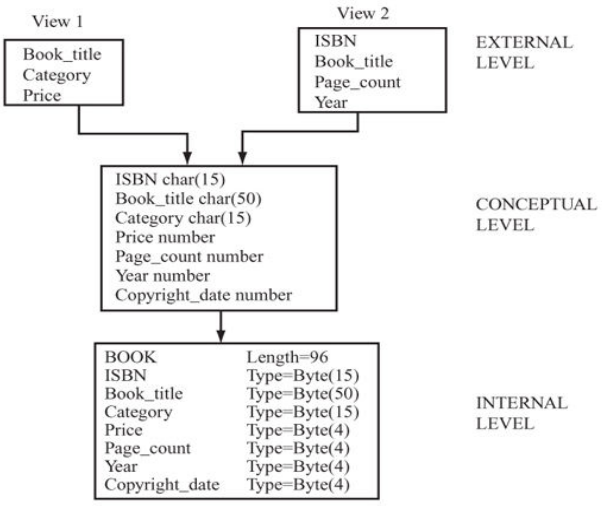
**4. System Analyst:** System Analyst determines the requirement of end users, especially naive and parametric end users and develops specifications for transactions that meet these requirements. System Analyst plays a major role in database design, its properties; the structure prepares the system requirement statement, which involves the feasibility aspect, economic aspect, technical aspect etc. of the system.

**3 LEVEL ARCHITECTURE / THREE-SCHEMA ARCHITECTURE:**

In this architecture, the overall database description can be defined at three levels namely internal, conceptual, and external levels. This is shown below:




Three levels of Online Book database (BOOK file)




- **Internal level**: It is the lowest level of data abstraction that deals with the physical representation of the database on the computer and thus, is also known as physical level. It describes how the data is physically stored and organized on the storage medium. At this level, various aspects are considered to achieve optimal runtime performance and storage space utilization. These aspects include storage space allocation techniques for data and indexes, access paths such as indexes, data compression and encryption techniques, and record placement.
- **Conceptual level**: This level of abstraction deals with the logical structure of the entire database and it is also known as logical level. It describes what data is stored in the database, the relationships among the data and complete view of the user's requirements without any concern for the physical implementation. It hides the complexity of physical storage structures. The conceptual view is the overall view of the database and it includes all the information that is going to be represented in the database.
- **External level**: It is the highest level of abstraction that deals with the user's view of the database and it is also known as view level. Most of the users and application programs do not require the entire data stored in the database. The external level describes a part of the database for a particular group of users. It permits users to access data in a way that is customized according to their needs, so that the same data can be seen by different users in different ways at the same time. It provides a powerful and flexible security mechanism by hiding the parts of the database from certain users, as the user is not aware of existence of any attributes that are missing from the view.

**DATA MODEL:**

A data model is a collection of concepts that can be used to describe the structure of a database. Data models can be broadly distinguished into 3 main categories-

1. Hierarchical Model
2. Network Model
3. Relational Model
4. Object-Oriented Model
5. Semi structured Model
6. Entity-relationship model

**1. Relational Model:**

Principle in a relational database is the **table**, a tabular arrangement of data values:

- A table is called a relation.

- The row (or record) in the table is called a tuple
- The column (or field) is called an attribute.
- The number of tuples is called the cardinality
- The number of attributes is called the degree of the table

**2.Network Model:**

Network model is a collection data in which records are physically linked through linked lists .A DBMS is said to be a Network DBMS if the relationships among data in the database are of type many-to-many. The relationships among many-to-many appear in the form of a network. Thus the structure of a network database is extremely complicated because of these many-to-many relationships in which one record can be used as a key of the entire database. A network database is structured in the form of a graph that is also a data structure.

**3. Hierarchical Model:**

The Hierarchical Data Model is a way of organizing a database with multiple one to many relationships. The structure is based on the rule that one parent can have many children but children are allowed only one parent. This structure allows information to be repeated through the parent child relationships. This database structure was one of the first used because it lends itself very well to linear type storage mediums, such as the data tapes that were used when database were first created.

**4. Object-Oriented Model:**

Object DBMSs add database functionality to object programming languages. They bring much more than persistent storage of programming language objects. A major benefit of this approach is the unification of the application and database development into a seamless data model and language environment. As a result, applications require less code, use more natural data modeling, and code bases are easier to maintain.

**5. Entity Relationship (ER) Model:**

The entity-relationship (E-R) model is the most popular conceptual model used for designing a database. It was originally proposed by Dr. Peter Chen in 1976 as a way to unify the network and relational database views. The E-R model views the real world as a set of basic objects (known as entities), their characteristics (known as attributes), and associations among these objects (known as relationships). The entities, attributes, and relationships are the basic constructs of an E-R model.

**FILE SYSTEM VS. DATA BASE MANAGEMENT SYSTEM**

1. Files act locally where as DBMS saves directly in a database
2. Saves in temporary locations where as DBMS in well arranged and permanent data base locations.
3. in File System Transactions are not possible where as various transactions like insert, delete, view, updating etc r possible in DBMS
4. Data will be accessed through single or various files where as in DBMS, tables (schema) is used to access data
5. A "File manager" is used to store all relationships in directories in File Systems where as a data base manager (administrator) stores the relationship in form of structural tables
6. Last.... but not the least.... Data in data bases are more secure compared to data in files!!

**DATABASE LANGUAGES:**

Database language a generic term referring to a class of languages used for defining and accessing databases. A particular database language will be associated with a particular database management system.

There are four types of database languages:

1. **Data Definition Language (DDL):** Statements are used to define the database structure or schema.
   Some examples:

- CREATE - to create objects in the database

- ALTER - alters the structure of the database
- DROP - delete objects from the database

2. **Data Manipulation Language (DML):** Statements are used for managing data within schema objects. Some examples:

- SELECT - Retrieve data from the a database
- INSERT - Insert data into a table
- UPDATE - Updates existing data within a table
- DELETE - deletes all records from a table, the space for the records remain

3. **Data Control Language (DCL)**: DCL statements control access to data and the database using statements such as GRANT and REVOKE. A privilege can either be granted to a User with the help of GRANT statement. The privileges assigned can be SELECT, ALTER, DELETE, EXECUTE, INSERT, INDEX etc. In addition to granting of privileges, you can also revoke (taken back) it by using REVOKE command. Some examples:

- GRANT - gives user's access privileges to database
- REVOKE - withdraw access privileges given with the GRANT command

4. **Transaction Control (TCL)**: Statements are used to manage the changes made by DML statements. It allows statements to be grouped together into logical transactions. Some examples:

- COMMIT - save work done
- SAVEPOINT - identify a point in a transaction to which you can later roll back
- ROLLBACK - restore database to original since the last COMMIT

**OVERALL SYSTEM ARCHITECTURE:**

DBMS (Database Management System) acts as an interface between the user and the database. The user requests the DBMS to perform various operations (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.

A database System is divided into modules based on their function. The functional components of a database system can be broadly divided into the storage manager and the query processor components.

1. Storage Manager
2. Query Processor

1. **Storage Manager:** The storage manager is important because database typically require a large amount of storage space. So it is very important efficient use of storage, and to minimize the movement of data to and from disk. A storage manager is a program module that provides the interface between the low-level data stored in the database and the application programs and the queries submitted to the system. The Storage manager is responsible for the interaction with the file manager. The Storage manager translates the various DML statements into low level file system commands. Thus the storage manager is responsible for storing, retrieving, and updating data in the database. The storage manager components include the following.
   - Authorization and Integrity Manager
   - Transaction Manger
   - File Manager
   - Buffer Manger

   Authorization and Integrity Manger tests for the satisfaction of integrity constraints and checks the authority of users to access data. Transaction manager ensures that the database remains in a consistent state and allowing concurrent transactions to proceed without conflicting. The file manager manages the allocation of

space on disk storage and the data structures used to represent information stored on disk. The Buffer manager is responsible for fetching the data from disk storage into main memory and deciding what data to cache in main memory.

The storage manager implements the following data structures as part of the physical system implementation. Data File, Data Dictionary, Indices. Data files stores the database itself. The Data dictionary stores Meta data about the structure of database, in particular the schema of the database. Indices provide fast access to data items.

2. **Query Processor:** The Query Processor simplifies and facilitates access to data. The Query processor includes the following component.
   - DDL Interpreter
   - DML Compiler
   - Query Evaluation Engine

   The DDL interpreter interprets DDL statements and records the definition in the data dictionary. The DML compilers translate DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands. The DML compiler also performs query optimization, which is it picks the lowest cost evaluation plan from among the alternatives. Query evaluation engine executes low level instructions generated by the DML compiler.
3. **User:**

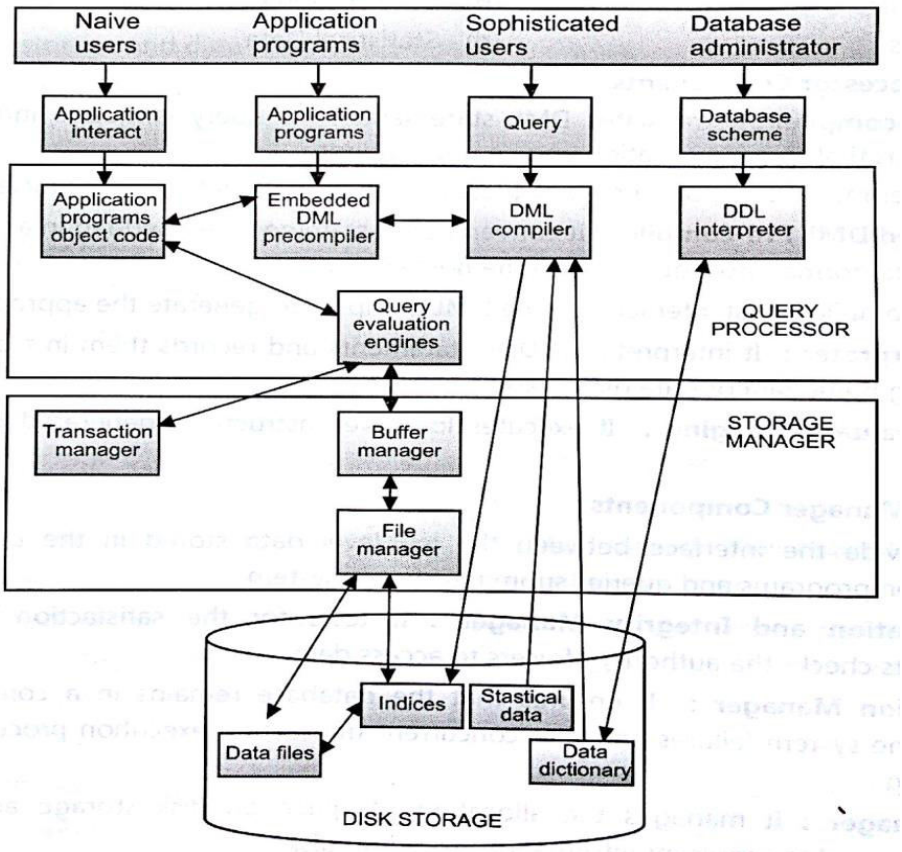
   **A) Database administrator:** DBA is responsible for authorizing access to the database, for coordinating and

   Monitoring its use and acquiring software and hardware resources as needed.

   **Sophisticated end users** – engineers, scientists, analysts who implement applications to meet their requirements.

   **B) Application programmer:** These users write application programs to interact with the database. Application programs can be written in some programming language such a COBOL, PL/I, C++, JAVA or some higher level fourth generation language. Such programs access the database by issuing the appropriate request, typically a SQL statement to DBMS.

   **C) Naive User:** Naive users access data through application programs that have been written for them. They do not need to know any details of the structure of language of the database system. They are totally Unsophisticated users who never know writing programs or even small queries.




**System structure**

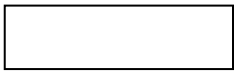
# Unit – II

## ENTITY RELATIONSHIP MODEL & BASIC CONCEPTS:

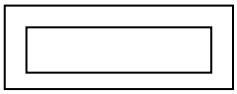
It is a semantic data model that is used for the graphical representation of the conceptual database design.
Entity relationship model defines the conceptual view of database. It works around real world entity and association among them. At view level, ER model is considered well for designing databases.
The Entity Relationship (ER) model consists of different types of entities. The existence of an entity may depends on the existence of one or more other entities, such an entity is said to be existence dependent. Entities whose existence not depending on any other entities is termed as not existence dependent. Entities based on their characteristics are classified as follows.

**ENTITY:**
A real-world thing either animate or inanimate that can be easily identifiable and distinguishable, called entity. Entities are represented by means of rectangles. Rectangles are named with the entity set they represent.


**TYPES OF ENTITY:**
**Strong entity:** An entity set that has a primary key is called as Strong entity set. Strong entity represented by rectangle which is shown below.

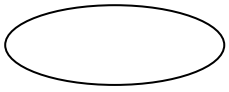


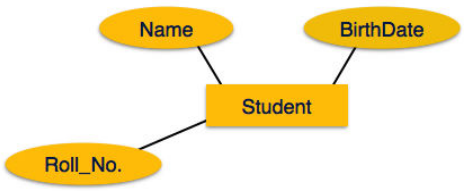
**Weak entity:** An entity set that does not have sufficient attributes to form a primary key is termed as a weak entity set. Weak entity represented by double rectangle which is shown below.



**ATRIBUTE:**
Attributes are properties of entities. Attributes are represented by means of ellipse. Every ellipse represents one attribute and is directly connected to its entity (rectangle). For example the employee is the entity and employee's name, age, address, salary and job etc are the attribute. Attribute is represented by ellipse.

Example:






**TYPES OF ATTRIBUTES:**
**Simple attribute:**

Simple attribute consists of a single atomic value. A simple attribute cannot be subdivided. For example the attributes age, sex etc is simple attributes.

**Composite attribute:**

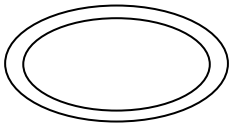
A composite attribute is an attribute that can be further subdivided. For example the attribute ADDRESS can be subdivided into street, city, state, and zip code.

**Single-valued attribute:**

A single valued attribute can have only a single value. For example a person can have only one 'date of birth', 'age' etc. That is a single valued attributes can have only single value. But it can be simple or composite attribute. That is 'date of birth' is a composite attribute; 'age' is a simple attribute. But both are single valued attributes.

**Multi-value attributes:**

Multivalve attributes can have multiple values. For instance a person may have multiple phone numbers, multiple degrees etc. Multivalve attributes are shown by a double line connecting to the entity in the ER diagram.

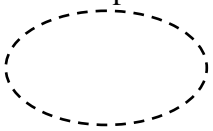


**Stored attribute:**

The value for the derived attribute is derived from the stored attribute. For example 'Date of birth' of a person is a stored attribute. The value for the attribute 'AGE' can be derived by subtracting the 'Date of Birth'(DOB) from the current date. Stored attribute supplies a value to the related attribute.

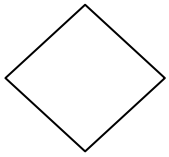
**Derived attribute:**

An attribute that's value is derived from a stored attribute. Example: age, and it's value is derived from the stored attribute Date of Birth. It is represented by dotted ellipse.

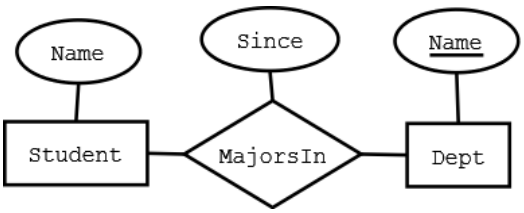


**RELATIONSHIP:**

The association among entities is called relationship. For example, employee entity has relation works_at with department. Another example is for student who enrolls in some course. Works_at and Enrolls are called relationship. Relationship is represented by diamond box.



Example: Student (entity type) is related to Department (entity type) by MajorsIn (relationship type).



**MAPPING CONSTRAINTS / CARDINALITY:**

While creating relationship between two entities. We may often need to face the cardinality problem. This simply means that how many entities of the first set are related to how many entities of the second set. Cardinality can be of the following four types.

**One-to-One:**

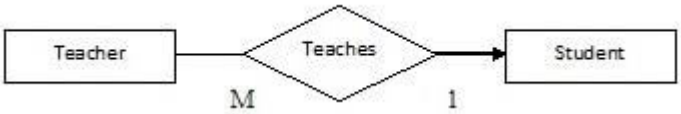
Only one entity of the first set is related to only one entity of the second set. Example A teacher teaches a student. Only one teacher is teaching only one student.


**One-to-Many:**

Only one entity of the first set is related to multiple entities of the second set. Example A teacher teaches students. Only one teacher is teaching many students.

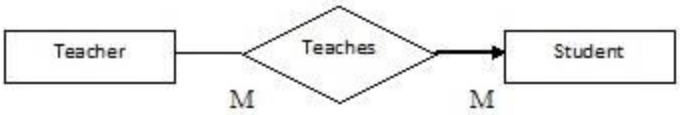

**Many-to-One:**

Multiple entities of the first set are related to multiple entities of the second set. Example Teachers teach a student. Many teachers are teaching only one student.




**Many-to-Many:**

Multiple entities of the first set is related to multiple entities of the second set. Example Teachers teach students. In any school or college many teachers are teaching many students. This can be considered as a two way one-to-many relationship.




**KEYS:**

A key is an attribute of a table which helps to identify a row. There can be many different types of keys:

**Super Key or Candidate Key**: It is such an attribute of a table that can uniquely identify a row in a table. Generally they contain unique values and can never contain NULL values. There can be more than one super key or candidate key in a table Example within a STUDENT table Roll and Mobile No. can both serve to uniquely identify a student.

**Primary Key:** It is one of the candidate keys that are chosen to be the identifying key for the entire table. Example although there are two candidate keys in the STUDENT table, the college would obviously use Roll as the primary key of the table.

**Alternate Key:** This is the candidate key which is not chosen as the primary key of the table. They are named so because although not the primary key, they can still identify a row.
**Composite Key:** Sometimes one key is not enough to uniquely identify a row. Example in a single class Roll is enough to find a student but in the entire school merely searching by the Roll is not enough because there could be 10 classes in the school and each one of them may contain a certain roll no 5. To uniquely identify the student we have to say something like "class VII, roll no 5". So a combination of two or more attributes is combined to create a unique combination of values such as Class + Roll.
**Foreign Key:** Sometimes we may have to work with an attribute that does not have a primary key of its own. To identify its rows, we have to use the primary attribute of a related table. Such a copy of another related table's primary key is called foreign key.

**E-R DIAGRAM (ERD):**
An entity-relationship diagram (ERD) is a data modeling technique that graphically illustrates an information system's entities and the relationships between those entities. An ERD is a conceptual and representational model of data used to represent the entity framework infrastructure.
The elements of an ERD are:

- **Entities:** A real-world thing either animate or inanimate that can be easily identifiable and distinguishable.
- **Attributes:** Entities are represented by means of their properties, called attributes.
- **Relationships:** The association among entities is called relationship.

Creating an E-R DIAGRAM includes:

1. Identifying and defining the entities
2. Determining all interactions between the entities
3. Analyzing the nature of interactions/determining the cardinality of the relationships
4. Creating the ERD

**ENTITY FEATURES:**

- Specialization
- Generalization
- Attribute Inheritance
- Aggregation

**Specialization:**
The process of designating sub groupings with in an entity set is called specialization. We use IS A relationship to represent specialization. IS A relationship may also be referred as super class-subclass relationship
Example: Person IS A Employee
Person IS A Customer
Employee IS A Manager
**Generalization:**
The design process takes place in bottom up manner. Multiple entity sets are synthesized into a higher level entity set on the basis of common features.
Example: Employee and Customer entities can be synthesized into a higher level entity Person.
**Attribute inheritance:**
The attributes of higher level entity set are inherited by lower level entity set.
**Aggregation:**
Aggregation is an abstraction in which relationship sets are treated as higher level entity sets. Here a relationship set is embedded inside an entity set, and these entity sets can participate in relationships.

**DESIGN OF AN E-R DATABASE SCHEMA:**

Database design is the process of producing a detailed data model of a database. This logical data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a data definition language, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity.

The term database design can be used to describe many different parts of the design of an overall database system. Principally, and most correctly, it can be thought of as the logical design of the base data structures used to store the data. In the relational model these are the tables and view. In an object database the entities and relationships map directly to object classes and named relationships. However, the term database design could also be used to apply to the overall process of designing, not just the base data structures, but also the forms and queries used as part of the overall database application within the database management system

1. Determine the purpose of the database - This helps prepare for the remaining steps.
2. Find and organize the information required - Gather all of the types of information to record in the database, such as product name and order number.
3. Divide the information into tables - Divide information items into major entities or subjects, such as Products or Orders. Each subject then becomes a table.
4. Turn information items into columns - Decide what information needs to be stored in each table. Each item becomes a field, and is displayed as a column in the table. For example, an Employees table might include fields such as Last Name and Hire Date.
5. Specify primary keys - Choose each table's primary key. The primary key is a column, or a set of columns, that is used to uniquely identify each row. An example might be Product ID or Order ID.
6. Set up the table relationships - Look at each table and decide how the data in one table is related to the data in other tables. Add fields to tables or create new tables to clarify the relationships, as necessary.
7. Refine the design - Analyze the design for errors. Create tables and add a few records of sample data. Check if results come from the tables as expected. Make adjustments to the design, as needed.
8. Apply the normalization rules - Apply the data normalization rules to see if tables are structured correctly. Make adjustments to the tables

**REDUCTION OF E-R SCHEMA TO TABLE:**

A database which conforms to an E R diagram can be represented by collection of table's .For each entity set and for each relationship set in the database, we will create unique tables, which is assigned the name of the corresponding entity set or relationship sets. Each table has a no. of columns which have unique names. Each row in the table corresponds to an entity or a relationship.

A database that conforms to an E-R database schema can be represented by a collection of tables. For each entity set and for each relationship set, there is a unique table. A table is a chart with rows and columns. The set of all possible rows is the Cartesian product of all columns. A row is also known as a tuple or a record. A table has an unlimited number of rows. Each column is also known as a field. There are few points to reduction of E-R schema.

**1. Strong Entity Sets:**

It is common practice for the table to have the same name as the entity set. There is one column for each attribute.

**2. Weak Entity Sets:**

There is one column for each attribute, plus the attribute(s) the form the primary key of the strong entity set that the weak entity set depends upon.

**3. Relationship Sets:**

We represent a relationship with a table that includes the attributes of each of the primary keys plus any descriptive attributes (if any).

There is a problem that if one of the entities in the relationship is a weak entity set. There would be no unique information in the relationship table and therefore may be omitted. Another problem can occur if there is an existence dependency. In that case you can combine the two tables.

**4. Multivalve Attributes:**

When an attribute is multivalve, remove the attribute from the table and create a new table with the primary key and the attribute, but each value will be a separate row.

**5. Generalization:**

Create a table for the higher-level entity set. For each lower-level entity set, create a table with the attributes for that specialization and include the primary key from the higher-level entity set.