# UNIT – I

**Introduction to Database:** File System Organization: Sequential, Pointer, Indexed, Direct. Purpose of Database System, Database Characteristics, Users of Database System, Advantages of DBMS Approach, Schemas and Instances, Three Schema Architecture and Data Independence, The Database System Environment

**The ER Model**: Overview of the Design issues, Entity-Relationship model Examples, Weak Entity Sets, Entity-Relationship Design issues, Extended ER features.

**COURSE OBJECTIVES:**

- To get familiar with fundamental concepts of database management such as database design, database languages, and database-system implementation

**COURSE OUTCOMES:**

✓ Develop the knowledge of fundamental concepts of database management systems.

## INTRODUCTION TO DATABASE

A **database** is a structured collection of data. The data are typically organized to model relevant aspects of reality (for example, the availability of rooms in hotels), in a way that supports processes requiring this information (for example, finding a hotel with vacancies).

The term *database* is correctly applied to the data and their supporting data structures, and not to the database management system (DBMS). The database data collection with DBMS is called a database system.

The term *database system* implies that the data are managed to some level of quality (measured in terms of accuracy, availability, usability, and resilience) and this in turn often implies the use of a general-purpose database management system (DBMS).

## FILE SYSTEM ORGANIZATION:

Various types of file organization include:

1. **Sequential Access:** A sequential file contains records organized by the order in which they were entered. The order of the records is fixed. Records in sequential files can be read or written only sequentially.
   - After you place a record into a sequential file, you cannot shorten, lengthen, or delete the record. However, you can update (REWRITE) a record if the length does not change. New records are added at the end of the file.
   - If the order in which you keep records in a file is not important, sequential organization is a good choice. Sequential output is also useful for printing reports.

2. **Direct Access:** Direct access file is also known as random access or relative file organization. In direct access file, all records are stored in direct access storage device, such as hard disk. The records are randomly placed throughout the file. The records are updated directly and rewritten back in the same location.
   - This file organization is useful for immediate access to large amount of information. It is used in accessing large databases. It is also called as hashing.
3. **Pointer based access:** Each file is a linked list of disk blocks. The disk blocks can be scattered anywhere on the disk. The directory entry contains a pointer to the starting and the ending file block. Each block contains a pointer to the next block occupied by the file.
   - This is very flexible in terms of file size. File size can be increased easily since the system does not have to look for a contiguous chunk of memory.
   - This method does not suffer from external fragmentation. This makes it relatively better in terms of memory utilization.
4. **Indexed file organization:** An indexed file contains records ordered by a *record key*. A record key uniquely identifies a record and determines the sequence in which it is accessed with respect to other records. Each record contains a field that contains the record key.
   - The possible record transmission (access) modes for indexed files are sequential, random, or dynamic. When indexed files are read or written sequentially, the sequence is that of the key values.
   - An indexed file can also use *alternate indexes*, for example, you could access a file through employee department rather than through employee number.

## PURPOSE OF DATABASE SYSTEM

The typical file processing system is supported by a conventional operating system. The system stores permanent records in various files, and it needs different application programs to extract records from, and add records to, the appropriate files.

A file processing system has a number of major disadvantages.

**1. Data Redundancy and Inconsistency:** In file processing, every user group maintains its own files for handling its data processing applications. For example, consider the UNIVERSITY database. Here, two groups of users might be the course registration personnel and the accounting office. The accounting office also keeps data on registration and related billing information, whereas the registration office keeps track of student courses and grades. Storing the same data multiple times is called data redundancy. This redundancy leads to several problems:

- ✓ Need to perform a single logical update multiple times.
- ✓ Storage space is wasted.
- ✓ Files that represent the same data may become inconsistent.
- ✓ Data inconsistency in the various copies of the same data may no longer agree.

Example: One user group may enter a student's birth date erroneously as JAN-24-1995, whereas the other user groups may enter the correct value of JAN-29-1995.

**2. Difficulty in Accessing Data:** File processing environments do not allow needed data to be retrieved in a convenient and efficient manner. Suppose that one of the bank officers needs to find out the names of all customers who live within a particular area. The bank officer has now two choices: either obtain the list of all customers and extract the needed information manually or ask a system programmer to write the necessary application program. Both alternatives are obviously unsatisfactory. Suppose that such a program is written, and that, several days later, the same officer needs to trim that list to include only those customers who have an account balance of $10,000 or more. A program to generate such a list does not exist. Again, the officer has the preceding two options, neither of which is satisfactory.

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

**4. Integrity Problems:** The data values stored in the database must satisfy certain types of consistency constraints. Example: The balance of certain types of bank accounts may never fall below a prescribed amount. Developers enforce these constraints in the system by addition appropriate code in the various application programs

**5. Atomicity Problems:** Atomic means the transaction must happen in its entirety or not at all. It is difficult to ensure atomicity in a conventional file processing system. Example: Consider a program to transfer $50 from account A to account B. If a system failure occurs during the execution of the program, it is possible that the $50 was removed from account A but was not credited to account B, resulting in an inconsistent database state.

**6. Concurrent Access Anomalies:** For the sake of overall performance of the system and faster response, many systems allow multiple users to update the data simultaneously. In such an environment, interaction of concurrent updates is possible and may result in inconsistent data. To guard against this possibility, the system must maintain some form of supervision. But supervision is difficult to provide because data may be accessed by many different application programs that have not been coordinated previously. Example: When several reservation clerks try to assign a seat on an airline flight, the system should ensure that each seat can be accessed by only one clerk at a time for assignment to a passenger.

**7. Security Problems:** Enforcing security constraints to the file processing system is difficult.

## DATABASE SYSTEM APPLICATIONS

- ✓ Banking: All Transactions
- ✓ Universities: Registration, Grades
- ✓ Sales: Customers, Products, Purchases
- ✓ Online Retailers: Order Tracking, Customized Recommendations
- ✓ Manufacturing: Production, Inventory, Orders, Supply Chain
- ✓ Airlines: Reservations, Schedules
- ✓ Human Resources: Employee Records, Salaries, Tax Deductions

## DATABASE CHARACTERISTICS

1. **Real-world entity:** Data management systems have been designed keeping in mind the needs of business organizations. They help businesses manage large amounts of data efficiently. These systems store huge volumes of data and provide easy ways to search through them. Some examples of such applications are Microsoft Access, Oracle, MySQL, etc.
2. **Relational databases:** Relational databases were first introduced in the 1970s. In this type of database, each record contains fields called attributes. Each attribute represents one piece of information about a particular object.
   - For example, if you want to keep track of your personal details then you will need three different attributes namely name, address, phone number. All of these attributes together form a single row in a table. This means that every time we add new information into our database, we must insert multiple rows into the same table.
   - If we do not follow this rule then we may end up having duplicate entries in our database. So relational databases allow us to organize data using relations between objects.
3. **Structured query language:** Structured Query Language was developed in the 1980s. It provides a way to write queries against a database. Queries written in SQL are known as structured queries because they use predefined structures to define relationships among entities.

   An example would be: SELECT * FROM employees WHERE department 'IT'.

   Here, the word '*' stands for all columns from the employee table.

   We can also specify only certain columns while selecting other ones.
4. **Isolation of data and application:** A database system is not the same thing as its data. A database works and organizes, whereas data is said to be passive. Metadata, which is data about data, is stored by the database management system.

   Also, in a traditional file management system, the structure of data files is defined in the application programs so the user had to change everything. But in DBMS, the structure of data files is not stored in the program but it is stored in the system catalog. Internal improvement of data efficiency or any changes in the data doesn't have an effect on application software with the help of this- that is, it offers insulation between Data and Program.
5. **ACID properties:** DBMS adheres to the concepts of Atomicity, Consistency, Isolation, and Durability, or ACID Properties. These concepts are applied to transactions, which operate and play around with data in a database. In multi-transactional environments, ACID properties help the database stay healthy in case of failure.
6. **Multiuser and concurrent access:** Data can be accessed and manipulated in parallel with the help of the DBMS. Users are unaware of the restrictions on transactions when handling the same data item. DBMS offers multiple views for various clients. A client who is in the Sales division will have an alternate perspective on the database than an individual working in the Production office. This component empowers the clients to have a concentrated perspective on the database as indicated by their prerequisites.

7. **Object oriented programming:** Object oriented programming is another technique used to design programs. Objects contain properties and methods. Properties represent variables or constants which hold values. Methods are procedures that operate on those properties. The main advantage of OOP over procedural programming is that it allows developers to create reusable components by encapsulating their functionality within classes. Classes are collections of related code and data elements. Developers can reuse existing classes instead of writing similar pieces of code again and again.
8. **Transactional processing:** Transactions are an essential part of any application. Transactions ensure consistency across multiple users accessing the system at the same time. When transactions fail due to errors, the entire transaction should roll back so that no changes made during the failed operation remain permanent. Transaction processing ensures that when there is a problem, everything remains consistent.
9. **Data mining:** Data mining refers to techniques used to analyze vast quantities of unstructured data. There are many types of data mining algorithms like classification, clustering, association rules, regression analysis, decision trees, neural networks, genetic algorithms, etc.
10. **Distributed database systems:** Distributed databases store information across several computers connected through a computer network. This type of architecture makes it easy to add more servers without having to rebuild the whole software infrastructure. A distributed database has three parts: client applications, server applications, and shared storage. Client applications access the shared storage using standard protocols such as TCP/IP. Server applications provide services to clients. Shared storage provides persistent storage space where all the data resides. In addition, each node stores metadata about other nodes in the cluster.
11. **Relational databases:** Relational databases organize data into tables consisting of rows and columns. Each row represents a single record while each column holds a value associated with a specific field. Tables may be linked together to form relationships between them.

    For example, if you have two people named John Smith and Jane Doe then they might both live in New York City. You could link these two tables together by adding a third table called People_in_NewYorkCity containing only the names of the people who live in NYC. Then you would use this new table to relate the first two tables together.
12. **SQL:** SQL stands for Structured Query Language. It was developed by IBM in 1969. The language allows programmers to create queries that can retrieve or modify data stored in relational databases. Queries consist of commands written in English followed by keywords. These keywords tell the database what kind of action to perform. Some examples include SELECT, INSERT, UPDATE and DELETE.
13. **Less redundancy and More consistency:** The DBMS follows the rules of normalization, which splits a relation when there is more than one attribute with the same value. Normalization is a method of reducing data redundancy. Every relation in a database is consistent, that's the state of consistency. Attempts to leave databases in different states can be detected with methods and techniques. A database management system can give greater consistency than earlier forms of data storage.

14. **Data security and integrity:** DBMS gives security to the information put away in this is on the grounds that all clients have different rights to access the database. A portion of the client can get to the entire information in the database while others can get to a small segment of the database.
    - For instance, a fluid mechanics instructor can only access those documents that are identified with his subject; however, the HOD of the division can get to records of all subjects that are identified with their department.
    - One of the main attributes of a database management system (DBMS)-integrity guarantees the quality and dependability of the database framework. It protects from unapproved access to the database and makes it safer. It allows only consistent and exact information into the database.

## USERS OF DATABASE SYSTEM

Primary goal of a database system is to retrieve information from and store new information into the database. People who work with a database can be categorized as database users or database administrators.

**Database Users:** There are four different types of database-system users, differentiated by the way they expect to interact with the system.

**1) Naïve Users:** Naïve users are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously. The typical user interface for naïve users is a forms interface, where the user can fill in appropriate fields of the form. Naïve users may also simply read reports generated from the database.

- As an example, consider a student, who during class registration period, wishes to register for a class by using a Web interface. Such a user connects to a Web application program that runs at a Web server. The application first verifies the identity of the user, and allows her to access a form where she enters the desired information. The form information is sent back to the Web application at the server, which then determines if there is room in the class and if so adds the student information to the class roster in the database.

**2) Application Programmers:** Application programmers are computer professionals who write application programs. Application programmers can choose from many tools to develop user interfaces. Rapid application development (RAD) tools are tools that enable an application programmer to construct forms and reports with minimal programming effort.

**3) Sophisticated Users:** Sophisticated users interact with the system without writing programs. Instead, they form their requests either using a database query language or by using tools such as data analysis software. Analysts who submit queries to explore data in the database fall in this category.

**4) Specialized Users:** Specialized users are sophisticated users who write specialized database applications that do not fit into the traditional data-processing framework. Among these applications are computer-aided design systems, systems that store data with complex data types (for example, graphics data and audio data), and environment-modeling systems.

## Database Administrator:

One of the main reasons for using DBMS is to have central control of both the data and the programs that access those data. ***A person who has such central control over the system is called a database administrator (DBA)***. The **functions/responsibilities** of a DBA include:

1. **Schema Definition:** The DBA creates the original database schema by executing a set of data definition statements in the DDL.
2. **Storage Structure and Access-Method Definition**
3. **Schema and Physical-Organization Modification:** The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization, or to alter the physical organization to improve performance.
4. **Granting of Authorization for Data Access:** By granting different types of authorization, the database administrator can regulate which parts of the database various users can access. The authorization information is kept in a special system structure that the database system consults whenever someone attempts to access the data in the system.
5. **Routine Maintenance:** Examples of the database administrator's routine maintenance activities are:
   - ✓ Periodically backing up the database, to prevent loss of data in case of disasters such as flooding.
   - ✓ Ensuring that enough free disk space is available for normal operations, and upgrading disk space as required.
   - ✓ Monitoring jobs running on the database and ensuring that performance is not degraded by very expensive tasks submitted by some users.

## ADVANTAGES OF DBMS APPROACH

Various advantages of DBMS approach are:

1. **Better Data Transferring:** Database management creates a place where users have an advantage of more and better managed data. Thus making it possible for end-users to have a quick look and to respond fast to any changes made in their environment.
2. **Better Data Security:** As number of users increases data transferring or data sharing rate also increases thus increasing the risk of data security. It is widely used in corporation world where companies invest money, time and effort in large amount to ensure data is secure and is used properly. DBMS provide a better platform for data privacy and security policies thus, helping companies to improve Data Security.
3. **Better data integration:** Due to Database Management System we have an access to well managed and synchronized form of data thus it makes data handling very easy and gives integrated view of how a particular organization is working and also helps to keep a track on how one segment of the company affects other segment.

4. **Minimized Data Inconsistency:** Data inconsistency occurs between files when different versions of the same data appear in different places. For Example, data inconsistency occurs when a student name is saved as “John Wayne” on a main computer of school but on teacher registered system same student name is “William J. Wayne”, or when the price of a product is $86.95 in local system of company and its National sales office system shows the same product price as $84.95. So if a database is properly designed then Data inconsistency can be greatly reduced hence minimizing data inconsistency.
5. **Faster data Access:** The Data base management system (DBMS) helps to produce quick answers to database queries thus making data accessing faster and more accurate. For example, to read or update the data. For example, end users, when dealing with large amounts of sale data, will have enhanced access to the data, enabling faster sales cycle.
   Some queries may be like:
   - What is the increase of the sale in last three months?
   - What is the bonus given to each of the salespeople in last five months?
6. **Better decision making:** Due to DBMS now we have better managed data and improved data accessing because of which we can generate better quality information hence on this basis better decisions can be made. Better Data quality improves accuracy, validity and time it takes to read data. DBMS does not guarantee data quality, it provides a framework to make it is easy to improve data quality.
7. **Increased end-user productivity:** The data which is available with the help of combination of tools which transform data into useful information helps end user to make quick, informative and better decisions that can make difference between success and failure in the global economy.
8. **Simple:** Data base management system (DBMS) gives simple and clear logical view of data

## SCHEMAS AND INSTANCES

**Schema:** The logical structure of the database (or) the overall design of the database is called the database schema. Schemas are changed infrequently, if at all. A database schema corresponds to the variable declarations (along with associated type definitions) in a program. Types of Schema include:

***Physical Schema:*** describes the database design at the physical level

***Logical Schema:*** describes the database design at the logical level.

A database may also have several schemas at the view level, sometimes called ***subschemas*** that describe different views of the database.

✓ The logical schema is the most important, in terms of its effect on application programs, since programmers construct applications by using the logical schema.

**Instance:** Collection of information stored in the database at a particular moment is called an instance of the database. Values of variables in a program at a point in time relate to an instance of database schema.

***Physical Data Independence:*** The ability to modify the physical schema without changing the logical schema is known as Physical data independence. Usually, applications depend on the logical schema.

***Logical Data Independence:*** Ability to modify logical schema without changing physical schema. It is harder to achieve as application programs are usually heavily dependent on logical structure of data.

## THREE SCHEMA ARCHITECTURE

**Data Abstraction:** A major purpose of a database system is to provide users with an abstract view of the data. That is, the system hides certain details of how the data are stored and maintained. For the system to be usable, it must retrieve data efficiently. There are several levels of abstraction:

1) **Physical Level:** The lowest level of abstraction describes ***how*** the data are actually stored. The physical level describes complex low-level data structures in detail.
2) **Logical Level:** The next-higher level of abstraction describes ***what*** data are stored in the database, and what relationships exist among those data. The logical level thus describes the entire database in terms of a small number of relatively simple structures. Database administrators, who must decide what information to keep in the database, use the logical level of abstraction.

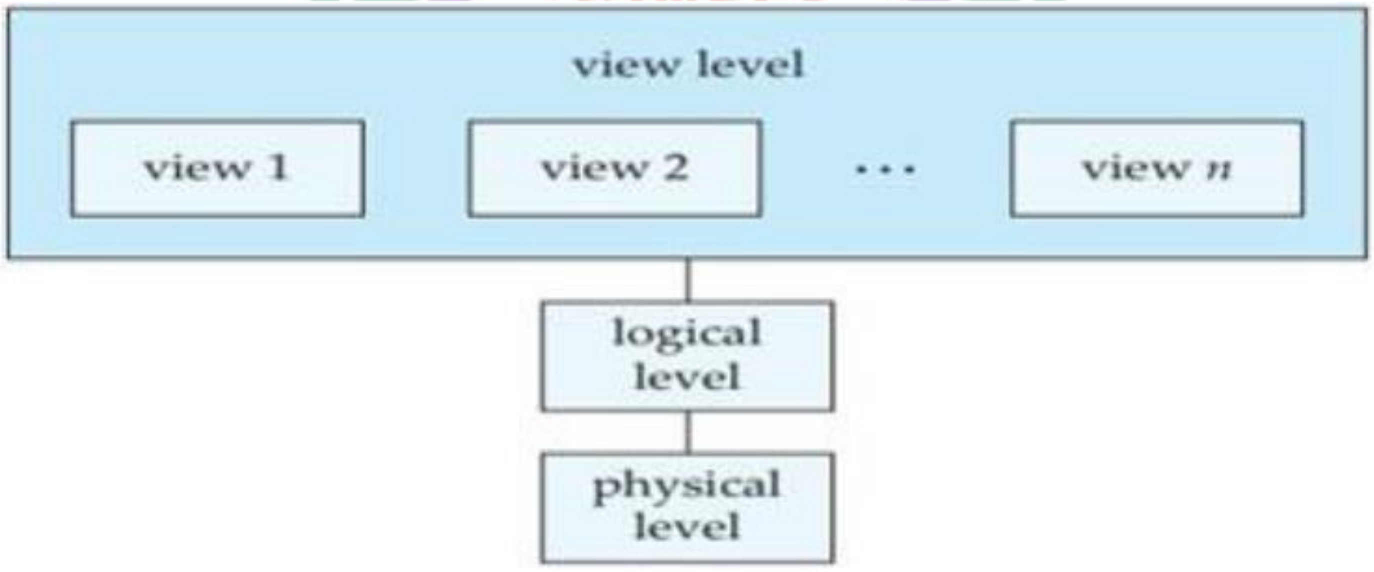


**Figure** The three levels of data abstraction.

*3)* **View Level:** Highest level of abstraction describes part of entire database for a particular group of users. Even though the logical level uses simpler structures, complexity remains because of the variety of information stored in a large database. Many users of the database system do not need all this information; instead, they need to access only a part of the database. View level of abstraction exists to simplify their interaction with the system. ***The system may provide many views for same database.***

## DATABASE SYSTEM ENVIRONMENT

One of the primary aims of a database is to supply users with an abstract view of data, hiding a certain element of how data is stored and manipulated. Therefore, the starting point for the design of a database should be an abstract and general description of the information needs of the organization that is to be represented in the database. And hence you will require an environment to store data and make it work as a database..

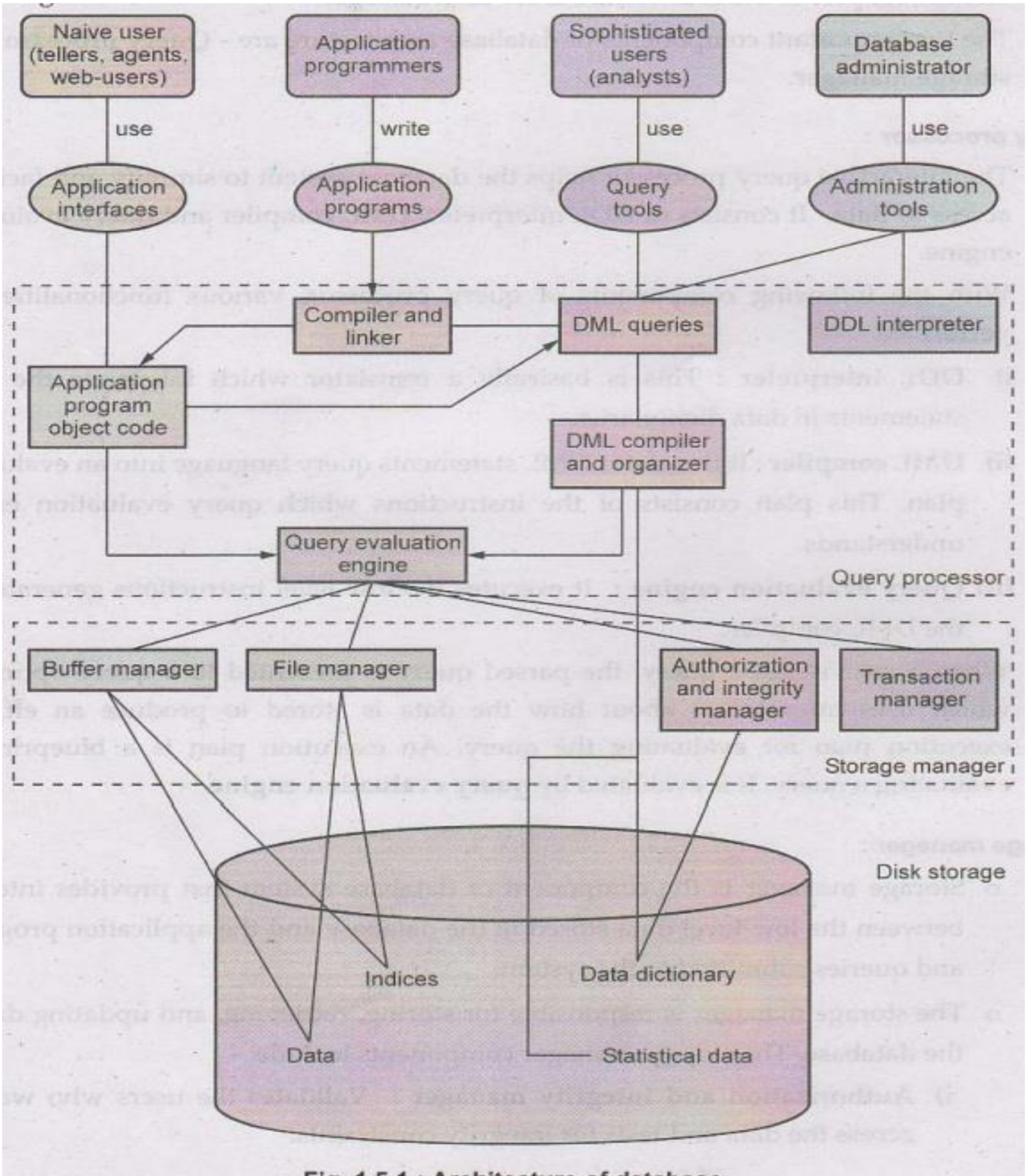


**Fig. 1.5.1 : Architecture of database**

**Storage Manager:** Storage manager is the component of a database system that provides interface between low-level data stored in database, application programs and queries submitted to the system.
The storage manager is important because databases typically require a large amount of storage space. Corporate databases range in size from hundreds of gigabytes to, for the largest databases, terabytes of data. Since the main memory of computers cannot store this much information, the information is stored on disks. Data are moved between disk storage and main memory as needed.

❖ *The storage manager is responsible for the interaction with the file manager*. The raw data are stored on the disk using the file system provided by the operating system. The storage manager is responsible for storing, retrieving, and updating data in the database.

The storage manager components include:

1) **Authorization and Integrity Manager**, which tests for the satisfaction of integrity constraints and checks the authority of users to access data.
2) **Transaction Manager**, which ensures that the database remains in a consistent (correct) state despite system failures, and that concurrent transaction executions proceed without conflicting.
3) **File Manager**, which manages the allocation of space on disk storage and the data structures used to represent information stored on disk.
4) **Buffer Manager**, which is responsible for fetching data from disk storage into main memory, and deciding what data to cache in main memory. Buffer manager is a critical part of the database system, since it enables the database to handle data sizes that are much larger than the size of main memory.

The storage manager implements ***several data structures*** as part of the physical system implementation:

- Data files, which store the database itself.
- Data dictionary, which stores metadata about structure of database, in particular schema of database.
- Indices, which can provide fast access to data items. Like the index in a textbook, a database index provides pointers to those data items that hold a particular value.

**The Query Processor:** The query processor is important because it helps the database system to simplify and facilitate access to data. It allows database users to obtain good performance while being able to work at the view level. It is the job of the database system to translate updates and queries written in a nonprocedural language, at the logical level, into an efficient sequence of operations at the physical level.
The query processor components include:

1) **DDL interpreter**, which interprets DDL statements and records the definitions in the data dictionary.
2) **DML compiler**, which translates 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*; that is, it picks lowest cost evaluation plan among the alternatives.
3) **Query evaluation engine**, which executes low-level instructions generated by the DML compiler.

**The ER Model**: Overview of the Design issues

1) Use of Entity Set vs Attributes
The use of an entity set or attribute depends on the structure of the real-world enterprise that is being modelled and the semantics associated with its attributes. It leads to a mistake when the user use the primary key of an entity set as an attribute of another entity set. Instead, he should use the relationship to do so. Also, the primary key attributes are implicit in the relationship set, but we designate it in the relationship sets.

2) Use of Entity Set vs. Relationship Sets
It is difficult to examine if an object can be best expressed by an entity set or relationship set. To understand and determine the right use, the user need to designate a relationship set for describing an action that occurs in-between the entities. If there is a requirement of representing the object as a relationship set, then its better not to mix it with the entity set.

# ENTITY-RELATIONSHIP MODEL

The entity-relationship (E-R) data model uses a collection of basic objects, called ***entities***, and ***relationships*** among these objects. It develops a conceptual design for the database.

**ENTITY:** An entity is a real time "thing" or "object". For example, each person is an entity, and bank accounts can be considered as entities. Entities are described in a database by a set of attributes. For example, the attributes ID, name, and salary may describe an EMPLOYEE entity. The attribute ID is used to identify EMPLOYEE uniquely.

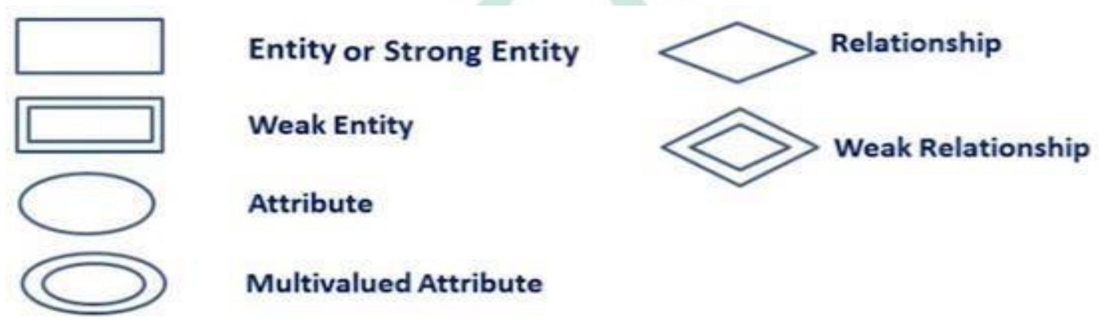
**RELATIONSHIP:** A relationship is an association among several entities. For example, a member relationship associates an instructor with his/her department.

The set of all entities of the same type and the set of all relationships of the same type are termed an ***entity se***t and ***relationship set****, respectively*.

The overall structure (schema) of a database can be expressed graphically by an entity- relationship (E-R) diagram. An E-R diagram is represented with:

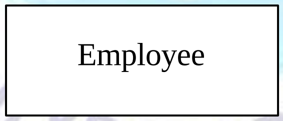
- ***Entity sets*** are represented by a partitioned rectangular box with the entity set name in the header and the attributes listed below it.
- ***Attributes*** are represented by Eclipse. These are used to describe the properties of an entity.
- ***Relationship sets*** are represented by a diamond connecting a pair of related entity sets. The name of the relationship is placed inside the diamond.

**Symbols used in E-R Diagrams:**

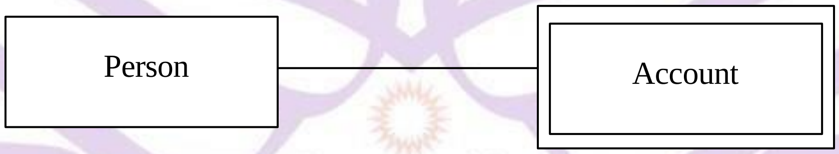


**ENTITY SETS:** An ***entity*** is a “thing” or “object” in the real world that is distinguishable from all other objects. For example, each person is an entity. An entity has a set of properties, called attributes.

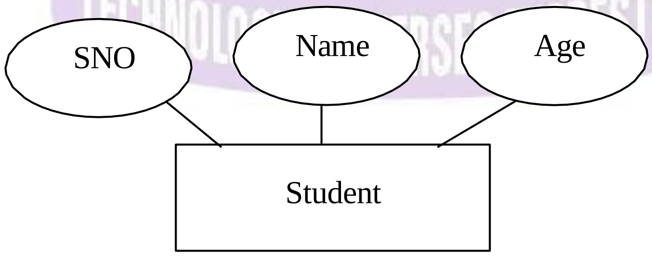
An ***entity set*** is a set of entities of the same type that share the same properties, or attributes. The set of all people who are employees at a company can be defined as the entity set Employee. It is represented as:



**Weak Entity:** An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.

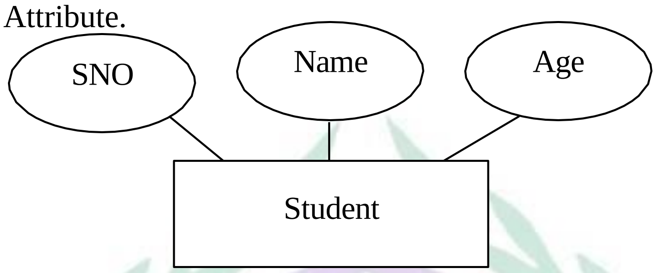


**ATTRIBUTES:** An attribute is used to describe the property of an entity. Eclipse is used to represent an attribute. For example, SNo, Name, Age, etc. can be attributes of a student.

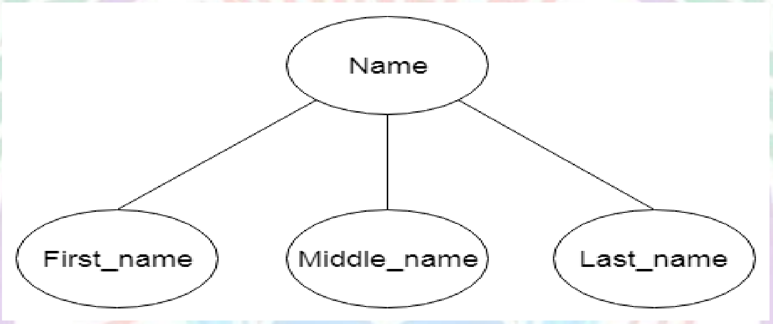


**Types of Attributes in E-R Model:**

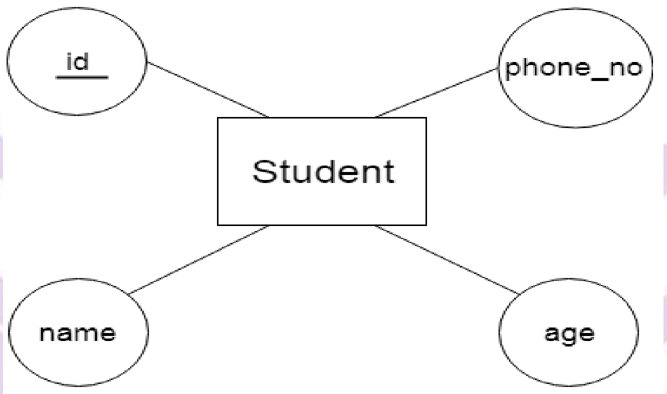
1) **Simple Attribute:** An attribute that contains atomic values, and which cannot be divided further is known as a Simple Attribute.



2) **Composite Attribute:** An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.

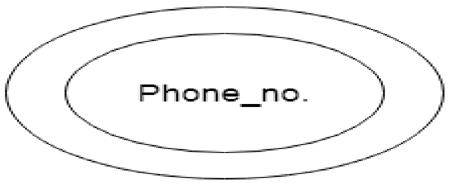


3) **Key Attribute:** The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.



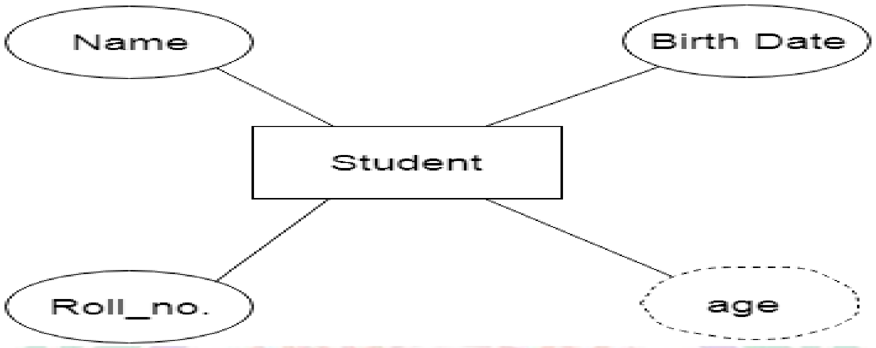
4) **Multivalued Attribute:** An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

For example, a student can have more than one phone number.

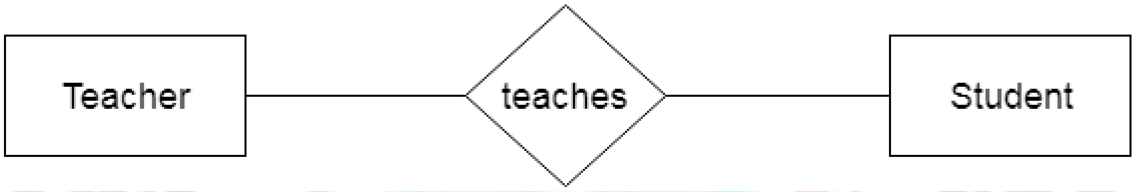


5) **Derived Attribute:** An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.

For example, A person's age can be derived from another attribute like Birth_Date.

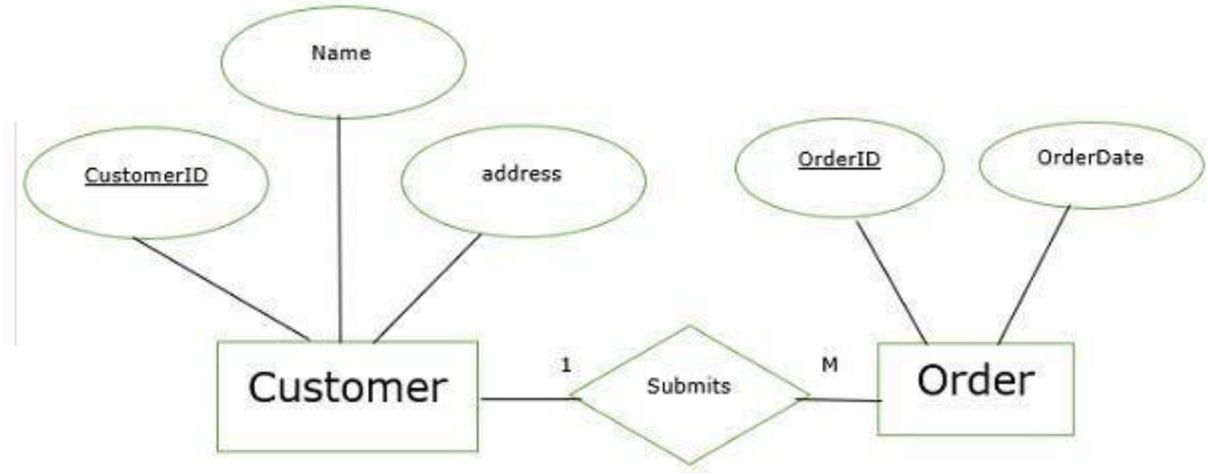


**RELATIONSHIP:** A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

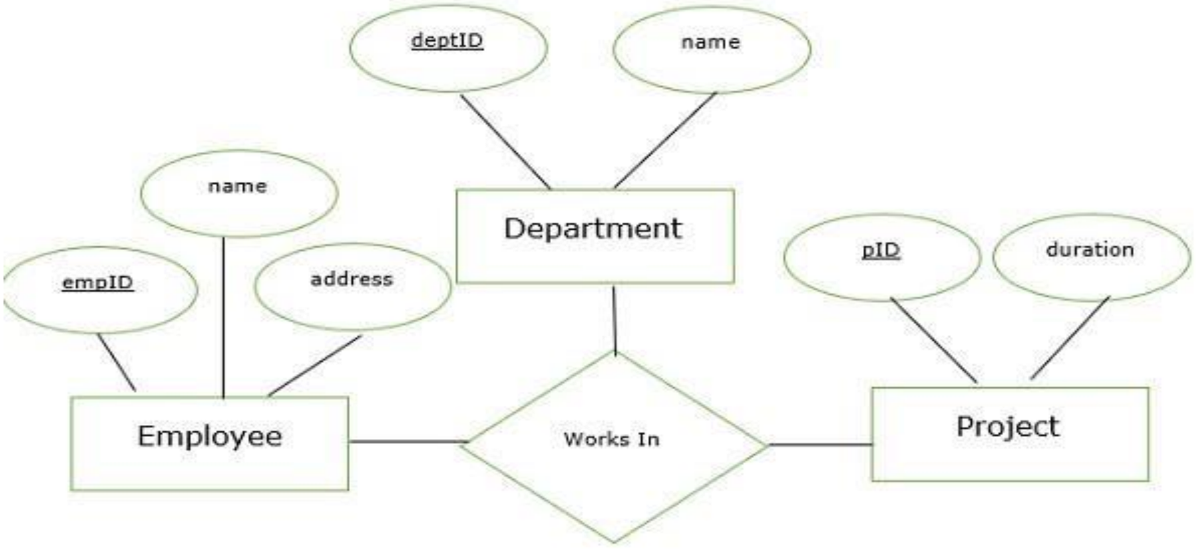


## Types of Relationships E-R Diagrams:

1) **Binary Relationship:** E-R diagrams that contain two entity sets depict binary relationship. The degree of binary relationship is 2.



2) **Ternary E-R Diagrams:** E-R diagrams that have three entity sets show ternary relationship. The degree of ternary relationship is 3.

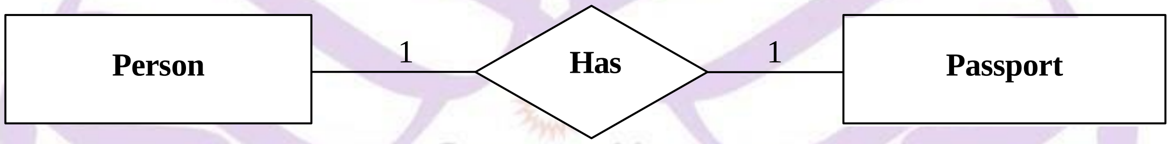


3) **N-ary Relationship:** n entities are involved in the relationship

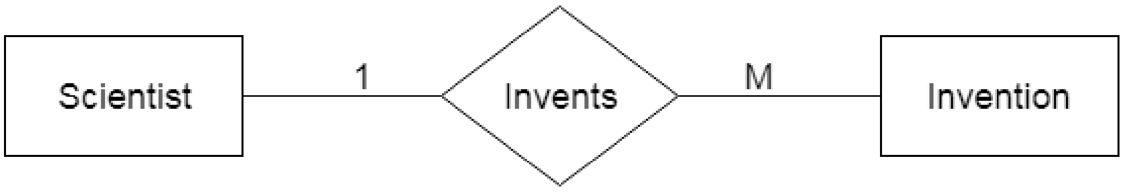
**CONSTRAINTS:** An E-R schema may define certain constraints to which the contents of a database must conform.

***Mapping Cardinalities:*** Mapping cardinalities express the number of entities to which another entity can be associated via a relationship set. Mapping cardinalities are most useful in describing binary relationship sets. For a binary relationship set R between entity sets A and B, the mapping cardinality must be one of the following:

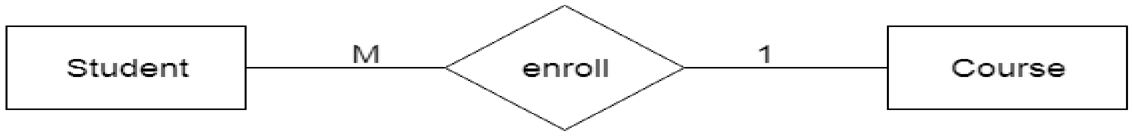
1) **One-to-One:** An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.
   For example, a Person can have only one Passport.



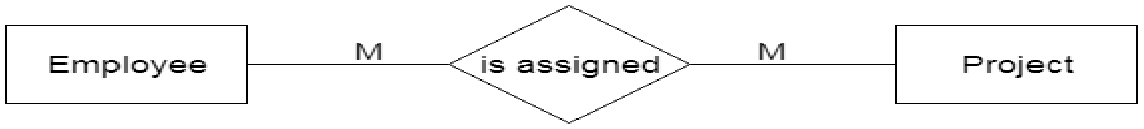
2) **One-to-Many:** An entity in A is associated with any number of entities in B. An entity in B, however, can be associated with at most one entity in A.
   For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.



3) **Many-to-One:** An entity in A is associated with at most one entity in B. An entity in B, however, can be associated with any number (zero or more) of entities in A.
For example, Student enrolls for only one course, but a course can have many students.

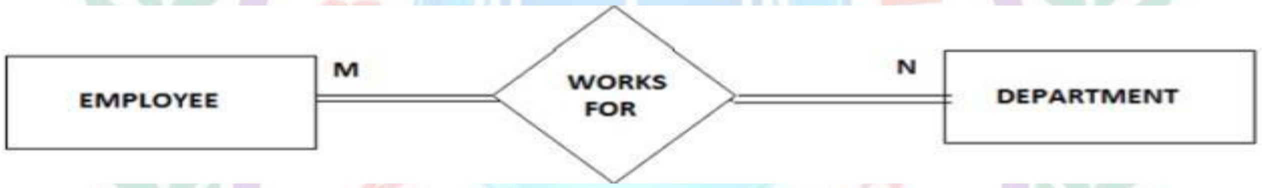


4) **Many-to-Many:** An entity in A is associated with any number (zero or more) of entities in B, and an entity in B is associated with any number (zero or more) of entities in A.
For example, Employee can assign by many projects and project can have many employees.

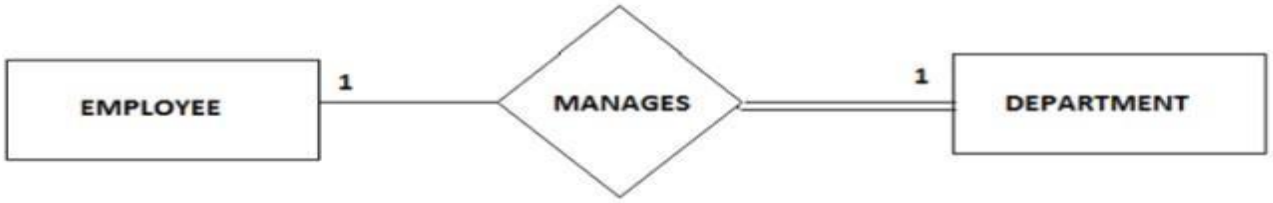


***Participation Constraints:*** There are two types of Participation constraints.

1) **Total Participation:** Each entity in the entity set is involved in at least one relationship in a relationship set. Total Participation is represented by double line in ER diagram.



2) **Partial Participation:** Each entity in entity set may or may not occur in at least one relationship in a relationship set.



**ER Design Issues:** Users often mislead the concept of the entities, relationships and the design process of the ER diagram. Thus, it leads to a complex structure of the ER diagram and certain issues that does not meet the characteristics of the real-world enterprise model.

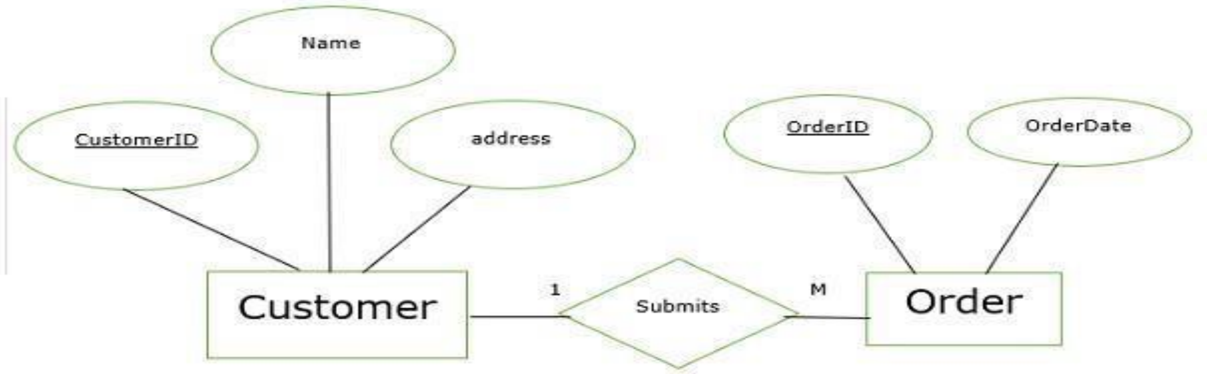
1) **Use of Entity Set vs Attributes:** Use of an entity set or attribute depends on the structure of the real-world enterprise that is being modelled. It leads to a mistake when the user use primary key of an entity set as an attribute of another entity set. Instead, he should use the relationship to do so.
2) **Use of Entity Set vs. Relationship Sets:** It is difficult to examine if an object can be best expressed by an entity set or relationship set. To understand and determine the right use, the user need to designate a relationship set for describing an action that occurs in-between the entities. If

there is a requirement of representing the object as a relationship set, then its better not to mix it with the entity set.

3) **Use of Binary vs n-ary Relationship Sets:** Generally, the relationships described in the databases are binary relationships. However, non-binary relationships can be represented by several binary relationships. For example, a ternary relationship can also be represented by two binary relationships.
4) **Placing Relationship Attributes:** The mapping cardinality can become an affective measure in the placement of the relationship attributes. So, it is better to associate the attributes of one-to-one or one-to-many relationship sets with any participating entity sets, instead of any relationship set.

**Conversion of E-R Diagram to Table:** The following rules are used to convert the ER diagram to tables and assign the mapping between the tables.

- Entity type becomes a table.
- All single-valued attribute becomes a column for the table.
- A key attribute of the entity type represented by the primary key.
- The multi valued attribute is represented by commas, or by a separate table.
- Composite attribute represented by components.
- Derived attributes are not considered in the table.



The above E-R diagram is converted as:

**Customer Relation/Table**

| <u>**CustomerID**</u> | **Name** | **Address** |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |

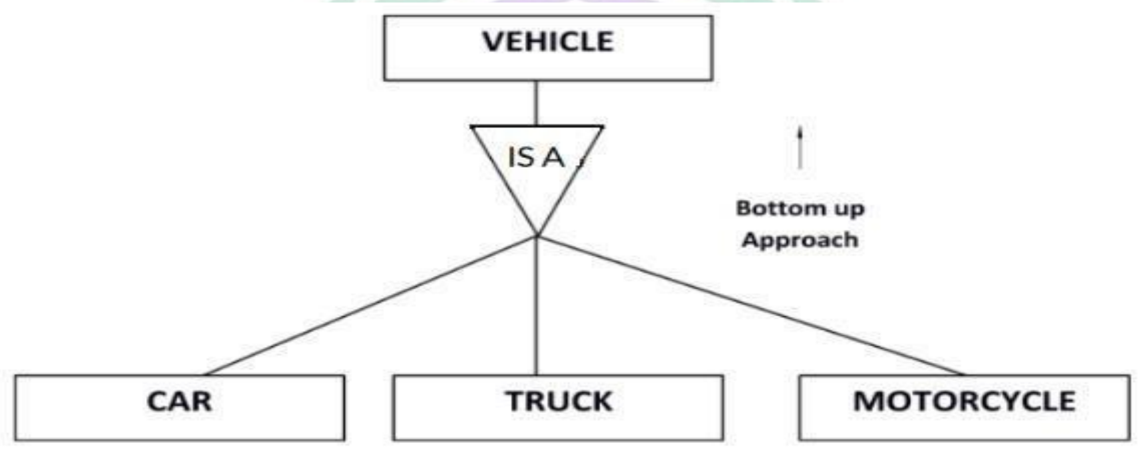
**Order Relation/Table**

| <u>**OrderID**</u> | **OrderDate** |
|---|---|
| | |
| | |
| | |
| | |
| | |

## EXTENDED ER:

**Generalization:** Generalization is like a bottom-up approach in which two or more entities of lower level combine to form a higher level entity if they have some attributes in common.

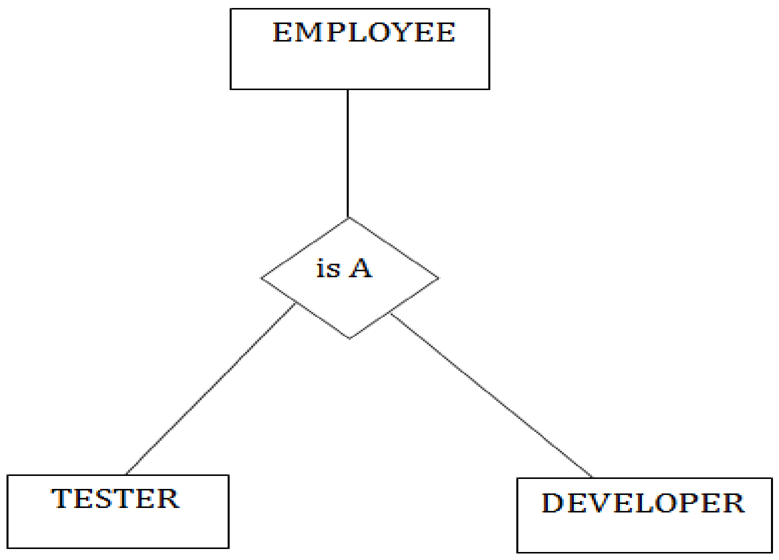
- Generalization is more like subclass and superclass system, but the only difference is the approach. Generalization uses the bottom-up approach.
- In generalization, entities are combined to form a more generalized entity, i.e., subclasses are combined to make a superclass.



**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.

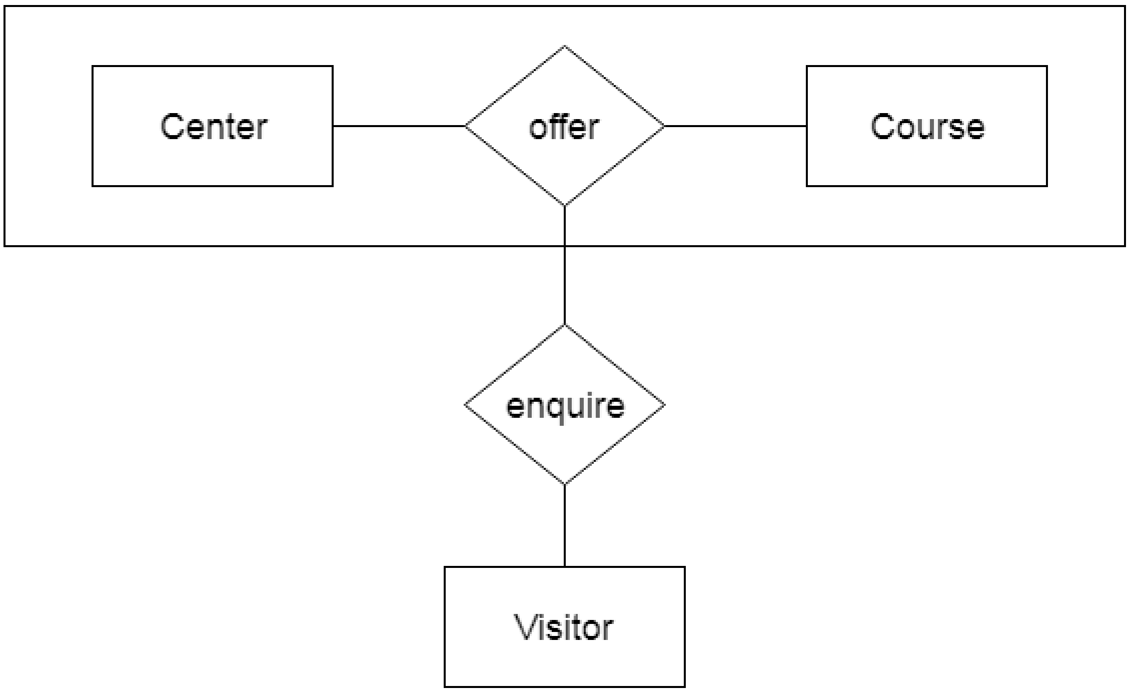
- Normally, the superclass is defined first, the subclass and its related attributes are defined next, and relationship set are then added.

For example, an EMPLOYEE entity can be specialized as TESTER or DEVELOPER based on what role they play in the company.



**Aggregation:** In aggregation, the relation between two entities is treated as a single entity. In aggregation, relationship with its corresponding entities is aggregated into a higher level entity.

For example, Center entity offers the Course entity act as a single entity in the relationship which is in a relationship with another entity visitor. In the real world, if a visitor visits a coaching center then he will never enquiry about the Course only or just about the Center instead he will ask the enquiry about both.


Center
offer
Course
enquire
Visitor