# Database Management Systems

**Database:**

The database is a collection of inter-related data which is used to retrieve, insert and delete the data efficiently. It is also used to organize the data in the form of a table, schema, views, and reports, etc.

**Database Management System**

● Database management system is a software which is used to manage the database. For example: MySQL, Oracle, etc are a very popular commercial database which is used in different applications.

● DBMS provides an interface to perform various operations like database creation, storing data in it, updating data, creating a table in the database and a lot more.

● It provides protection and security to the database. In the case of multiple users, it also maintains data consistency.

**Database Applications**

Applications where we use Database Management Systems are:

● Telecom: There is a database to keeps 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.

● Sales: To store customer information, production information and invoice details.

● Airlines: To travel though airlines, we make early reservations, this reservation information along with flight schedule is stored in database.

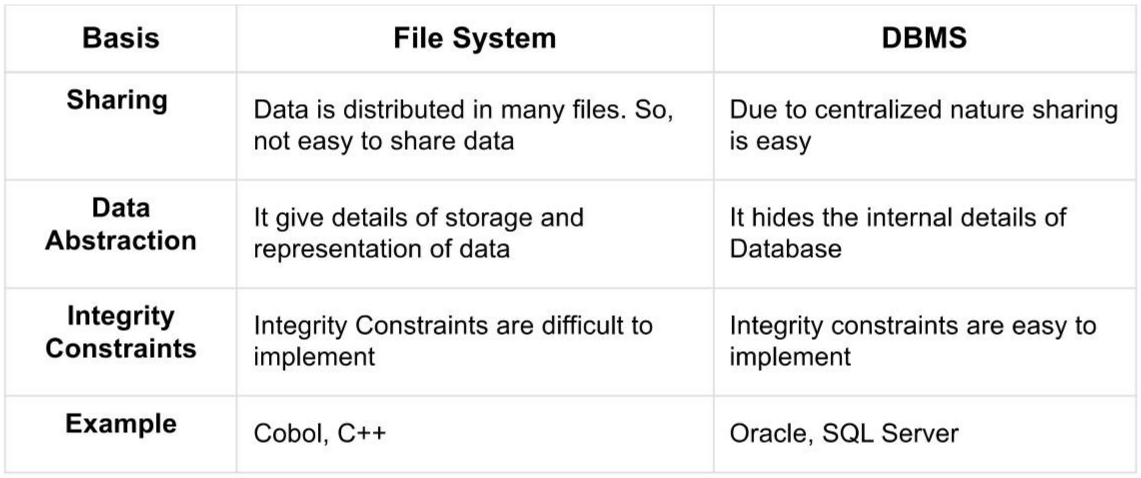
● Education sector: Database systems are frequently used in schools and colleges to store and retrieve the data regarding student details, staff details, course details, exam details, payroll data, attendance details, fees details etc. There is a hell lot amount of inter-related data that needs to be 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.

**Difference between File System and DBMS:**

## Difference between File System and DBMS:

| Basis | File System | DBMS |
|---|---|---|
| **Structure** | The file system is a way of arranging the files in a storage medium within a computer. | DBMS is software for managing the database. |
| **Data Redundancy** | Redundant data can be present in a file system. | In DBMS there is no redundant data. |
| **Backup and Recovery** | It doesn't provide backup and recovery of data if it is lost. | It provides backup and recovery of data even if it is lost. |
| **Query processing** | There is no efficient query processing in the file system. | Efficient query processing is there in DBMS. |
| **Consistency** | There is less data consistency in the file system. | There is more data consistency because of the process of normalization. |
| **Complexity** | It is less complex as compared to DBMS. | It has more complexity in handling as compared to the file system. |
| **Security Constraints** | File systems provide less security in comparison to DBMS. | DBMS has more security mechanisms as compared to file systems. |
| **Cost** | It is less expensive than DBMS. | It has a comparatively higher cost than a file system. |
| **Data Independence** | There is no data independence. | In DBMS data independence exists. |
| **User Access** | Only one user can access data at a time. | Multiple users can access data at a time. |
| **Meaning** | The user has to write procedures for managing databases | The user not required to write procedures. |



| Basis | File System | DBMS |
|---|---|---|
| Sharing | Data is distributed in many files. So, not easy to share data | Due to centralized nature sharing is easy |
| Data Abstraction | It give details of storage and representation of data | It hides the internal details of Database |
| Integrity Constraints | Integrity Constraints are difficult to implement | Integrity constraints are easy to implement |
| Example | Cobol, C++ | Oracle, SQL Server |

## Three schema architecture/Data Abstraction and Data Independence

There are mainly 3 levels of data abstraction:

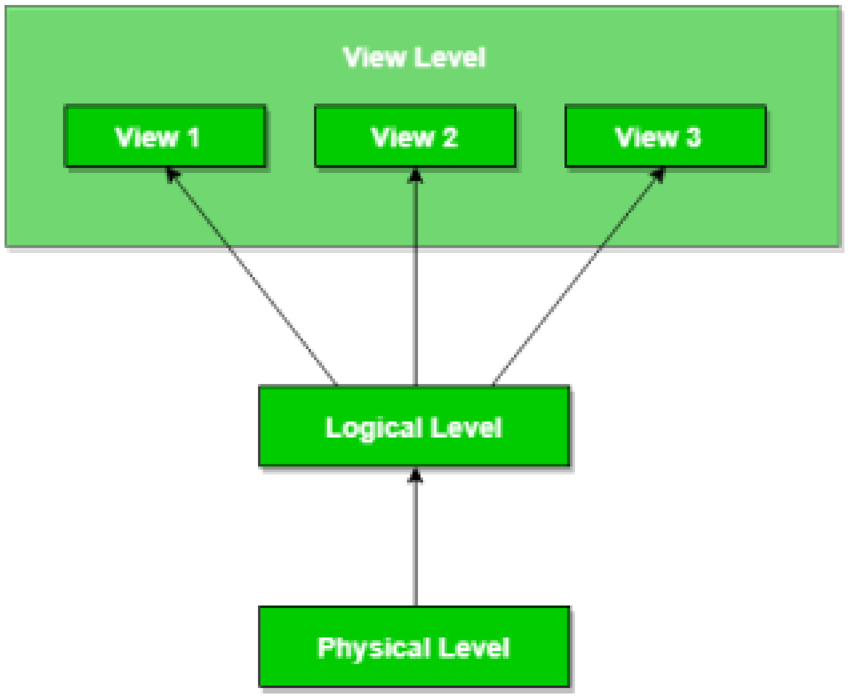
**Physical**: This is the lowest level of data abstraction. It tells us how the data is actually stored in memory. The access methods like sequential or random access and file organization methods like B+ trees, hashing used for the same. Usability, size of memory, and the number of times the records are factors that we need to know while designing the database.

Suppose we need to store the details of an employee. Blocks of storage and the amount of memory used for these purposes are kept hidden from the user.

**Logical**: This level comprises the information that is actually stored in the database in the form of tables. It also stores the relationship among the data entities in relatively simple structures. At this level, the information available to the user at the view level is unknown.

We can store the various attributes of an employee and relationships, e.g. with the manager can also be stored.

**View**: This is the highest level of abstraction. Only a part of the actual database is viewed by the users. This level exists to ease the accessibility of the database by an individual user. Users view data in the form of rows and columns. Tables and relations are used to store data. Multiple views of the same database may exist. Users can just view the data and interact with the database, storage and implementation details are hidden from them.



The main purpose of data abstraction is to achieve data independence in order to save time and cost required when the database is modified or altered.

We have namely two levels of data independence arising from these levels of abstraction :

**Physical level data independence**: It refers to the characteristic of being able to modify the physical schema without any alterations to the conceptual or logical schema, done for optimization purposes, e.g., Conceptual structure of the database would not be affected by any change in storage size of the database system server. Changing from sequential to random access files is one such example. These alterations or modifications to the physical structure may include:

● Utilizing new storage devices.

● Modifying data structures used for storage.

● Altering indexes or using alternative file organization techniques etc.

**Logical level data independence**: It refers characteristic of being able to modify the logical schema without affecting the external schema or application program. 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.

**DBMS Architecture**

● The DBMS design depends upon its architecture. The basic client/server architecture is used to deal with a large number of PCs, web servers, database servers and other components that are connected with networks.

● The client/server architecture consists of many PCs and a workstation which are connected via the network.

● DBMS architecture depends upon how users are connected to the database to get their request done.

**Types of DBMS Architecture**

Database architecture can be seen as a single tier or multi-tier. But logically, database architecture is of two types like: 2-tier architecture and 3-tier architecture.

**1-Tier Architecture**

● In this architecture, the database is directly available to the user. It means the user can directly sit on the DBMS and uses it.

● Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users.

The 1-Tier architecture is used for development of the local application, where programmers can directly communicate with the database for the quick response.

**2-Tier Architecture**

● The 2-Tier architecture is same as basic client-server. In the two-tier architecture, applications on the client end can directly communicate with the database at the server side. For this interaction, API's like: ODBC, JDBC are used.

● The user interfaces and application programs are run on the client-side.

● The server side is responsible to provide the functionalities like: query processing and transaction management.

● To communicate with the DBMS, client-side application establishes a connection with the server side.

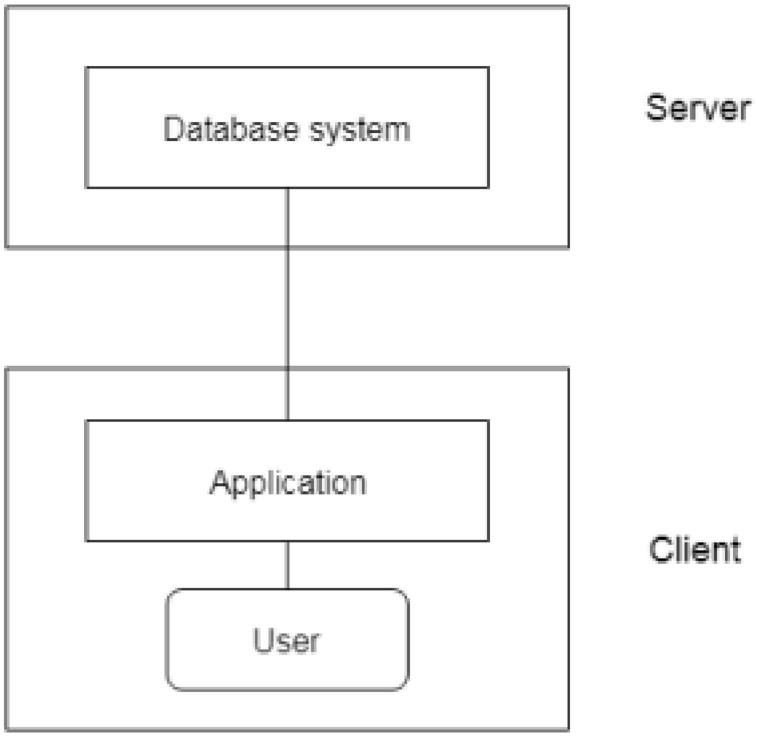


Fig: 2-tier Architecture

**3-Tier Architecture**

● The 3-Tier architecture contains another layer between the client and server. In this architecture, client can't directly communicate with the server.

● The application on the client-end interacts with an application server which further communicates with the database system.

● End user has no idea about the existence of the database beyond the application server. The database also has no idea about any other user beyond the application.

● The 3-Tier architecture is used in case of large web application.

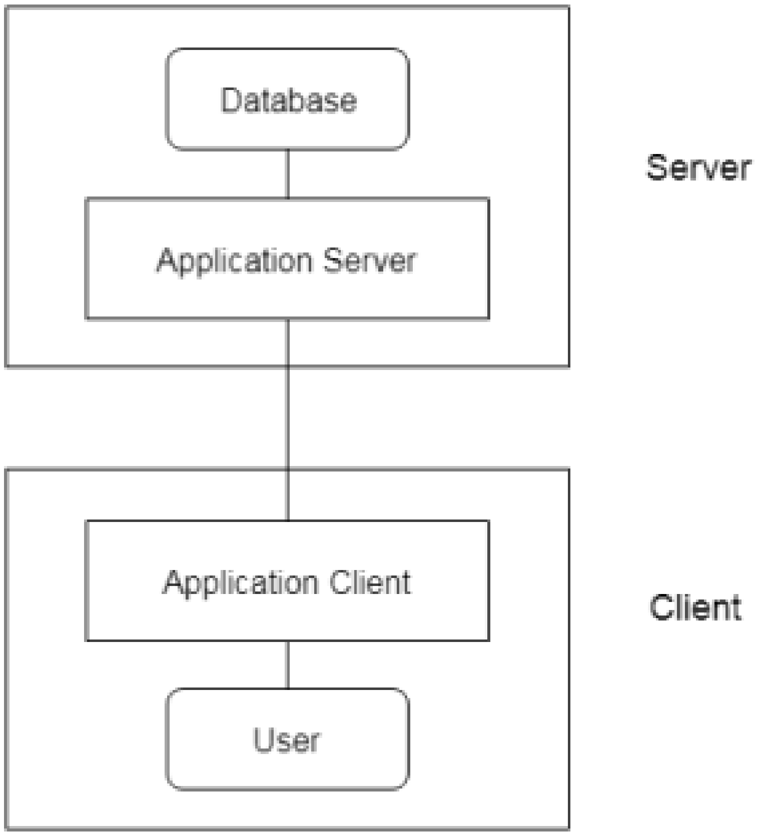
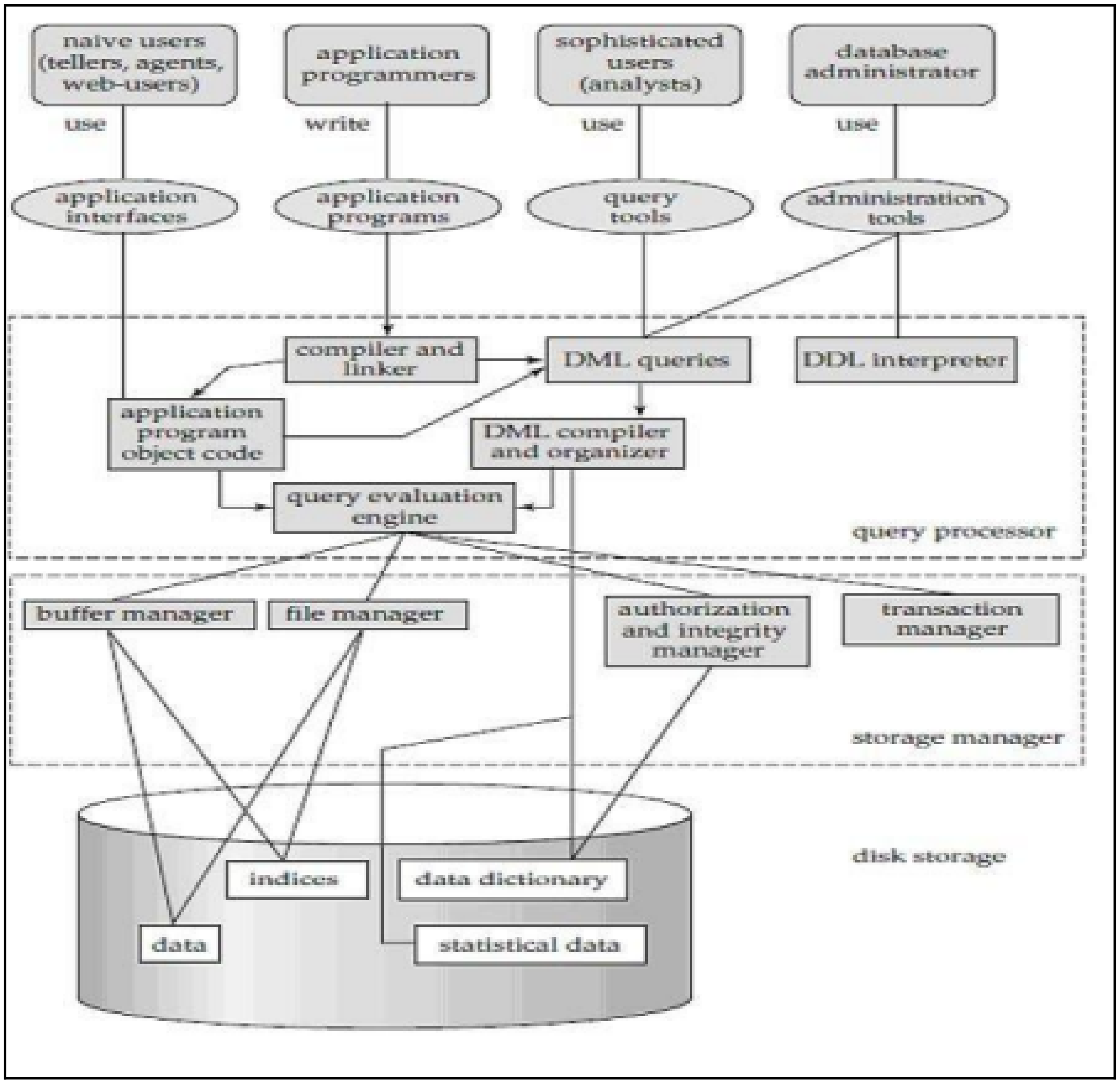


Fig: 3-tier Architecture

## Structure of a DBMS

A database system is partitioned into modules that deal with each of the responsibilities of the overall system. The functional components of a database system can be broadly divided into the storage manager and the query processor components. The storage manager is important because databases typically require a large amount of storage space. The query processor is important because it helps the database system simplify and facilitate access to data.



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.

**Query Processor**

The query processor components include

**DDL interpreter**, which interprets DDL statements and records the definitions in the data dictionary.

**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.

A query can usually be translated into any of a number of alternative evaluation plans that all give the same result. The DML compiler also performs query optimization, that is, it picks the lowest cost evaluation plan from among the alternatives.

Query evaluation engine, which executes low-level instructions generated by the DML compiler.

**Storage Manager**

A *storage manager* is a program module that provides the interface between the lowlevel data stored in the database and the application programs and queries submitted to the system. The storage manager is responsible for the interaction with the file manager. The raw data are stored on the disk using the file system, which is usually provided by a conventional operating system. 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:

Authorization and integrity manager, which tests for the satisfaction of integrity constraints and checks the authority of users to access data.
Transaction manager, which ensures that the database remains in a consistent (correct) state despite system failures, and that concurrent transaction executions proceed without conflicting.

File manager, which manages the allocation of space on disk storage and the data structures used to represent information stored on disk

**Buffer manager**, which is responsible for fetching data from disk storage into main memory, and deciding what data to cache in main memory. The 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.

**Transaction Manager**

A transaction is a collection of operations that performs a single logical function in a database application. Each transaction is a unit of both atomicity and consistency. Thus, we require that transactions do not violate any database-consistency constraints. That is, if the database was consistent when a transaction started, the database must be consistent when the transaction successfully terminates. Transaction - manager ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.

**Database Administrators**

The life cycle of database starts from designing, implementing to administration of it. A database for any kind of requirement needs to be designed perfectly so that it should work without any issues. Once all the design is complete, it needs to be installed. Once this step is complete, users start using the database. The database grows as the data grows in the database. When the database becomes huge, its performance comes down. Also accessing the data from the database becomes challenge. There will be unused memory in database, making the memory inevitably huge. These administration and maintenance of database is taken care by database Administrator – DBA.

A DBA has many responsibilities. A good performing database is in the hands of DBA. Database Administrators coordinate all the activities of the database system. They have all the permissions.
Tasks of DBA

- Creating the schema
- Specifying integrity constraints
- Storage structure and access method definition
- Granting permission to other users.
- Monitoring performance
- Routine Maintenance

## Data Models in DBMS

Data models in DBMS help to understand the design at the conceptual, physical, and logical levels as it provides a clear picture of the data making it easier for developers to create a physical database.

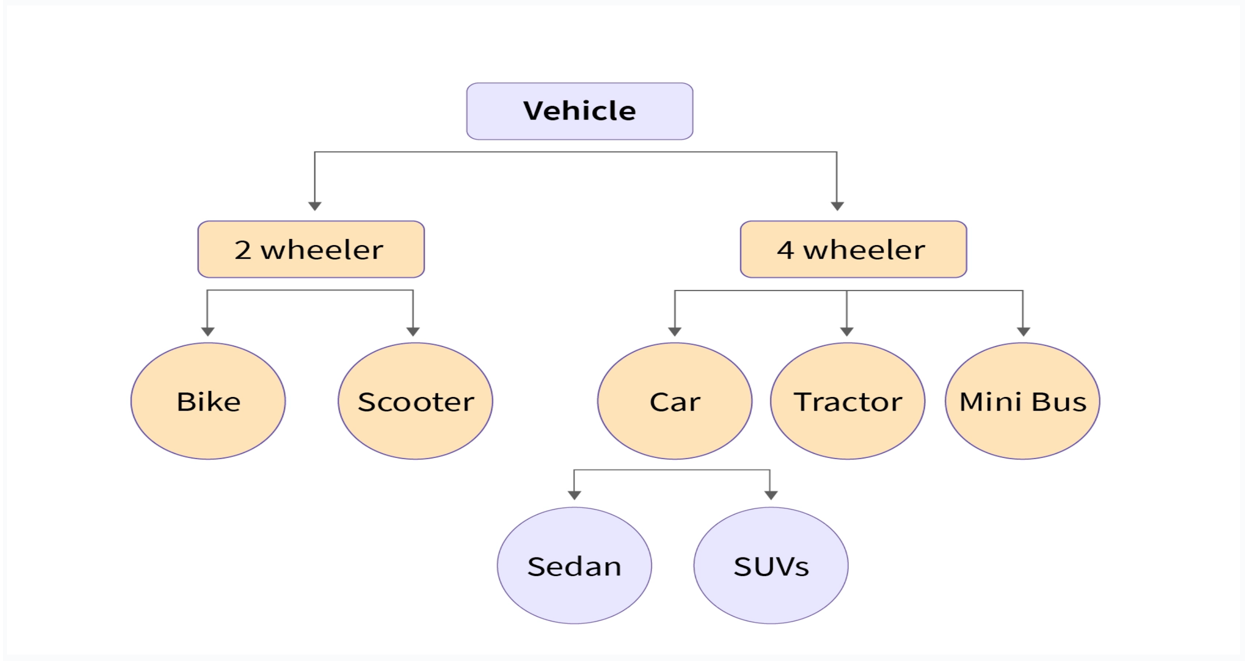
Data models are used to describe how the data is stored, accessed, and updated in a DBMS. A set of symbols and text is used to represent them so that all the members of an organization can understand how the data is organized. It provides a set of conceptual tools that are vastly used to represent the description of data.

There are many types of data models that are used in the industry.

## Types of Data Models in DBMS

### Hierarchical Model

The hierarchical data model is one of the oldest data models, developed in the 1950s by IBM. In this data model, the data is organized in a hierarchical tree-like structure. This data model can be easily visualized because each record in DBMS has one parent and many children (possibly 0) as shown in the image given below.



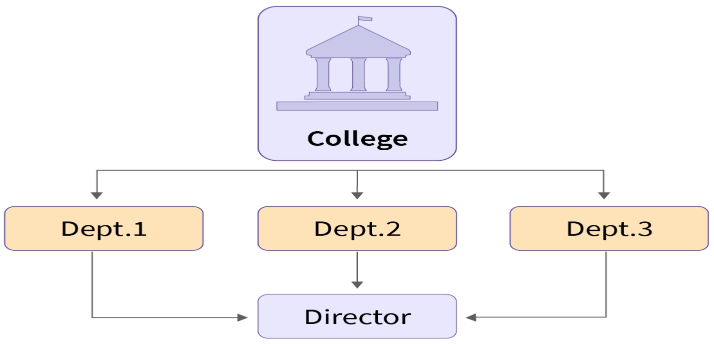
The above-given image represents the data model of the Vehicle database, vehicle are classified into two types Viz. two-wheelers and four-wheelers and then they are further classified.

The main drawback we can see here is we can only have one too many relationships under this model, hence the hierarchical data model is very rarely used nowadays.

### Network Model

A network model is nothing but a generalization of the hierarchical data model as this data model allows many to many relationships therefore in this model a record can also have more than one parent.

The network model in DBMS can be represented as a graph and hence it replaces the hierarchical tree with a graph in which object types are the nodes and relationships are the edges.



Here you can see all three departments are linked with the director which was not possible in the hierarchical data model.

In the network model, there can be many possible paths to reach a node from the root node (College is the root node in the above case), therefore the data can be accessed efficiently when compared to the hierarchical data model. But, on the other hand, the process of insertion and deletion of data is quite complex.

Relational Model

This is the most widely accepted data model. In this model, the database is represented as a collection of relations in the form of rows and columns of a two-dimensional table. Each row is known as a tuple (a tuple contains all the data for an individual record) while each column represents an attribute. For example -

| Stu. Id | Name | Branch |
|---|---|---|
| 101 | Naman | CSE |
| 102 | Saloni | ECE |
| 103 | Rishabh | IT |
| 104 | Pulkit | ME |

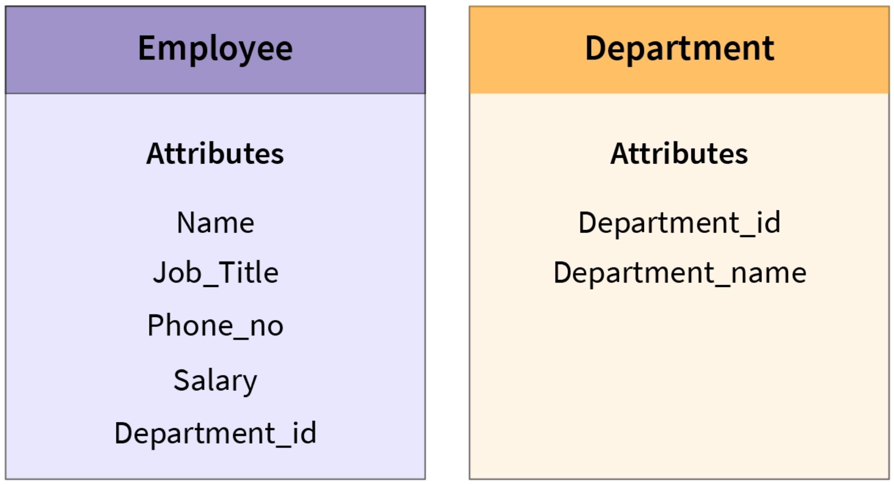
The above table shows a relation "STUDENT" with attributes such as Stu. Id, Name, and Branch which consists of 4 records or tuples.

Check out this article to learn more about the Relational model in DBMS.

Object-Oriented Data model

As suggested by its name, the object-oriented data model is a combination of object-oriented programming and relational data model. In this data model, the data and their relationship are represented in a single structure which is known as an object.

Since data is stored as objects we can easily store audio, video, images, etc in the database which was very difficult and inconvenient to do in the relational model. As shown in the image below two objects are connected with each other through links.

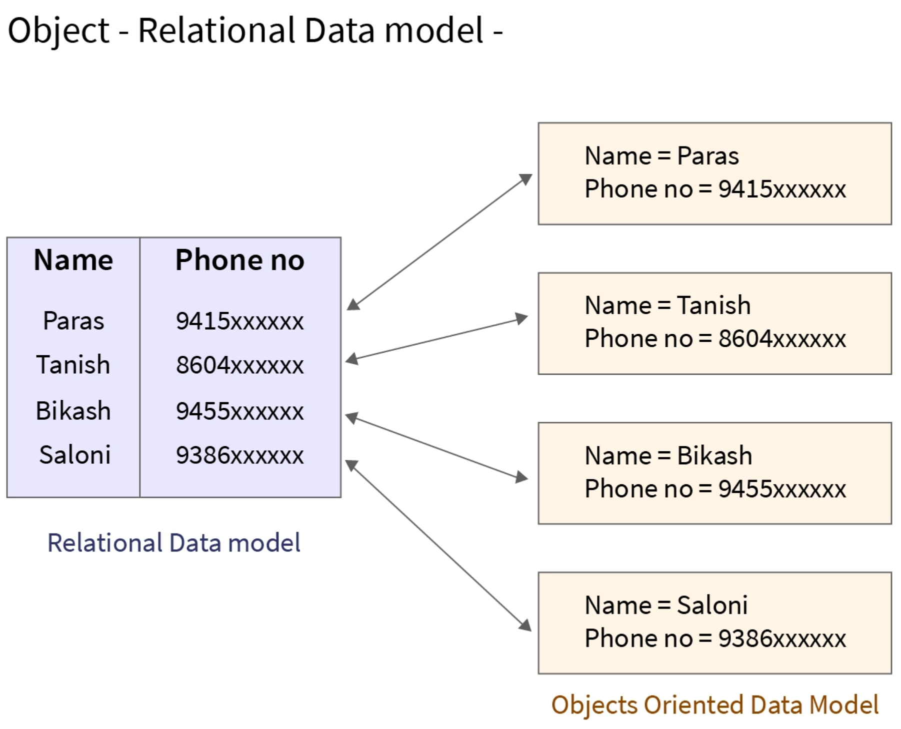


In the above image, we have two objects that are Employee and Department in which all the data is contained in a single unit (object). They are linked with each other as they share a common attribute

Object Relational Data Model

Again as suggested by its name, the object-relational data model is an integration of the object-oriented model and the relational model. Since it inherits properties from both of the models it supports objects, classes, etc like object-oriented models, and tabular structures like the relational model.

For example -



It provides data structures and operations used in the relational model and also provides features of object-oriented models like classes, inheritance, etc. The only drawback of this data model is that it is complex and quite difficult to handle.

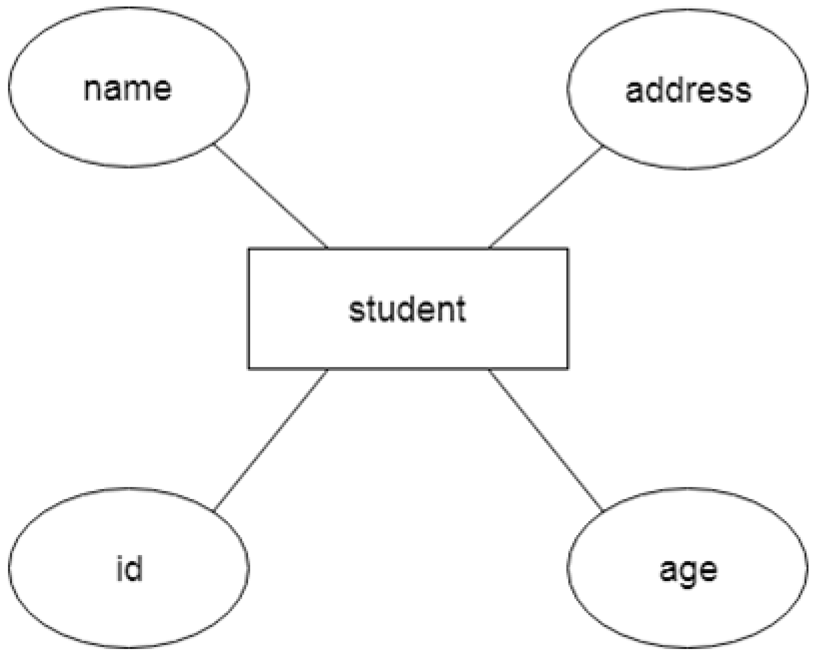
## ER (Entity Relationship) Diagram in DBMS

○ ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.

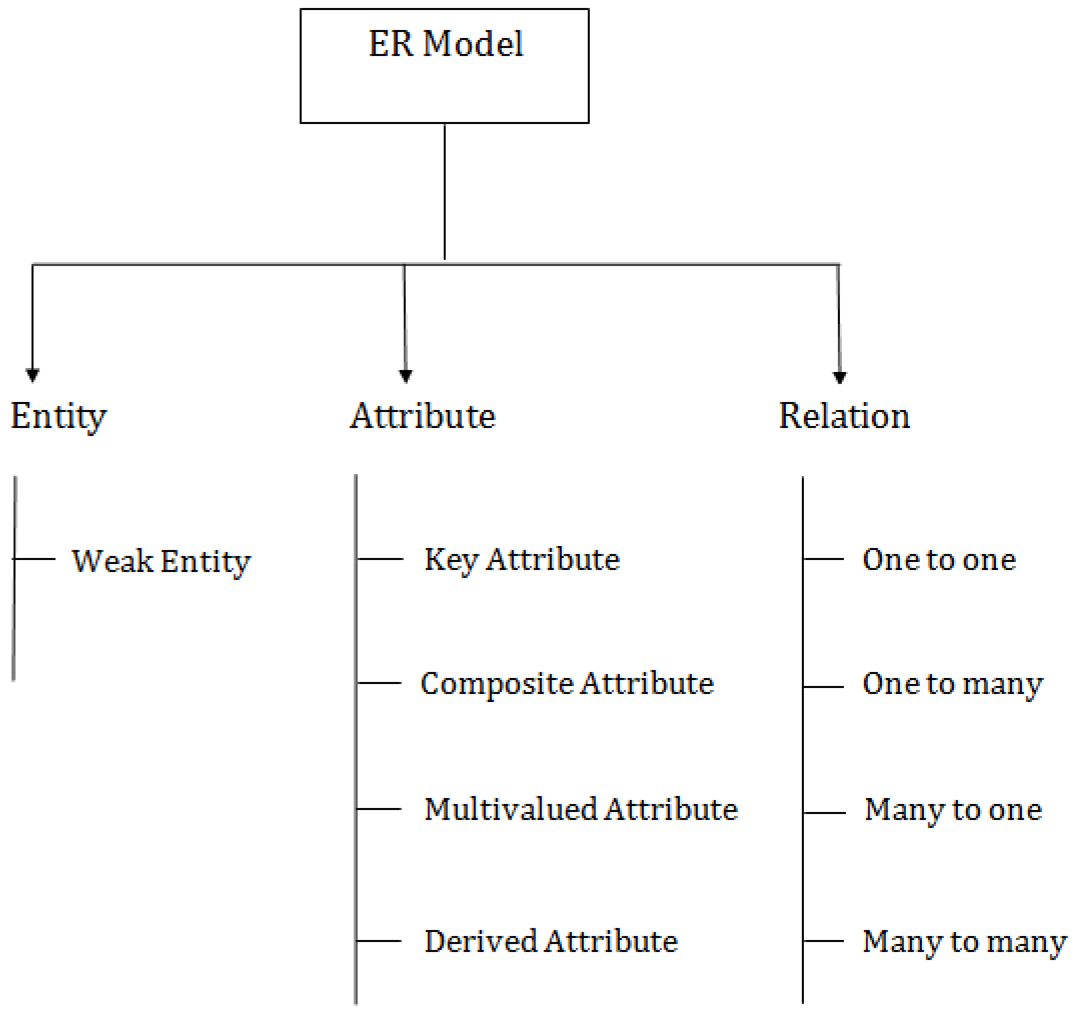
○ It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.

○ In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

For example, Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age, etc. The address can be another entity with attributes like city, street name, pin code, etc and there will be a relationship between them.



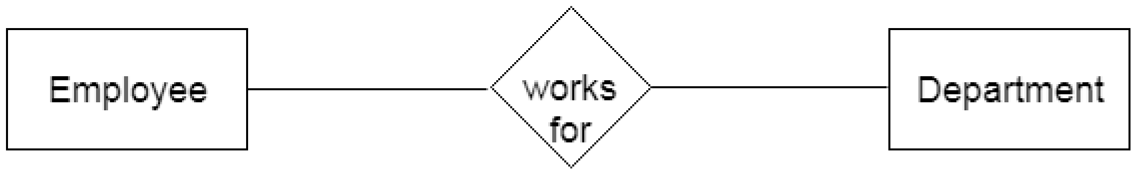
## Component of ER Diagram



## 1. Entity:

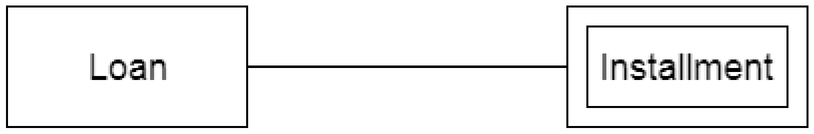
An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



a. Weak Entity

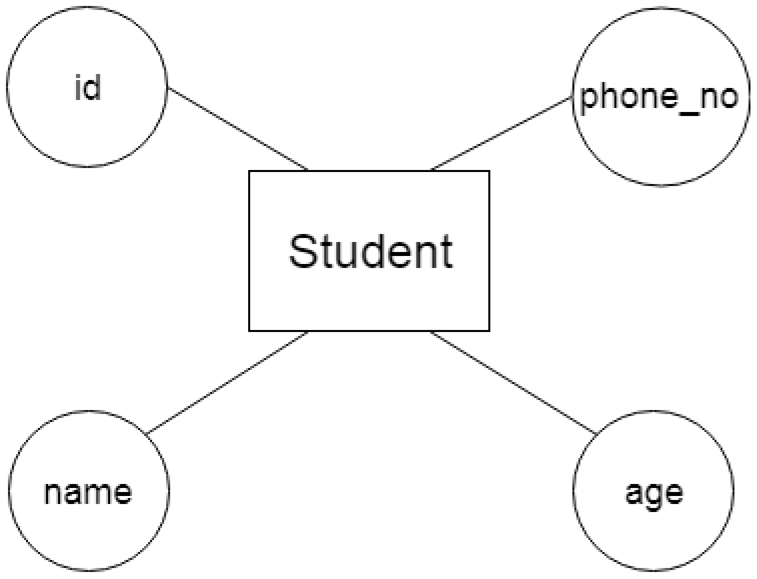
An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.



## 2. Attribute

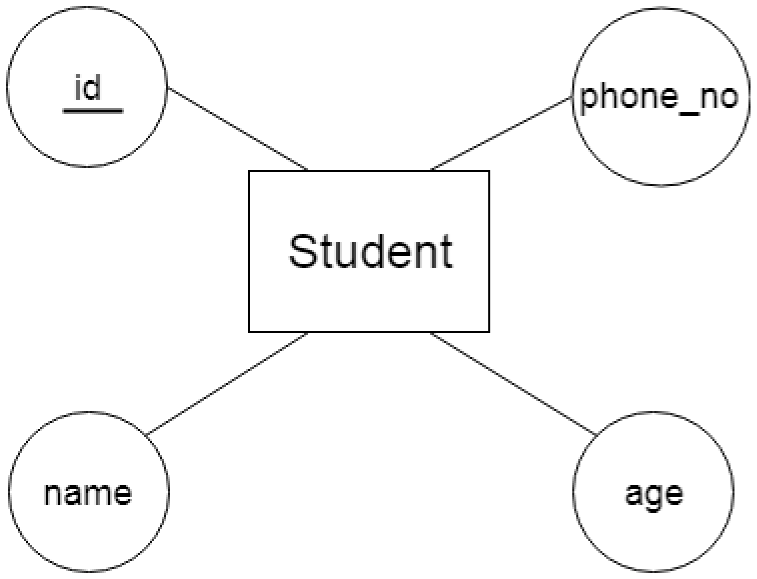
The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

For example, id, age, contact number, name, etc. can be attributes of a student.



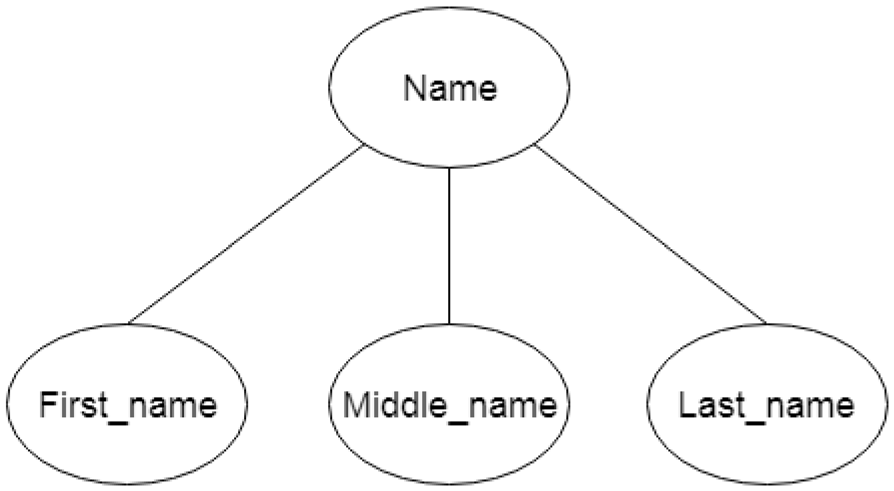
**a. Key Attribute**

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.



**b. Composite Attribute**

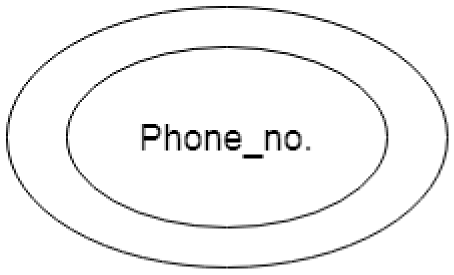
An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



**c. 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.

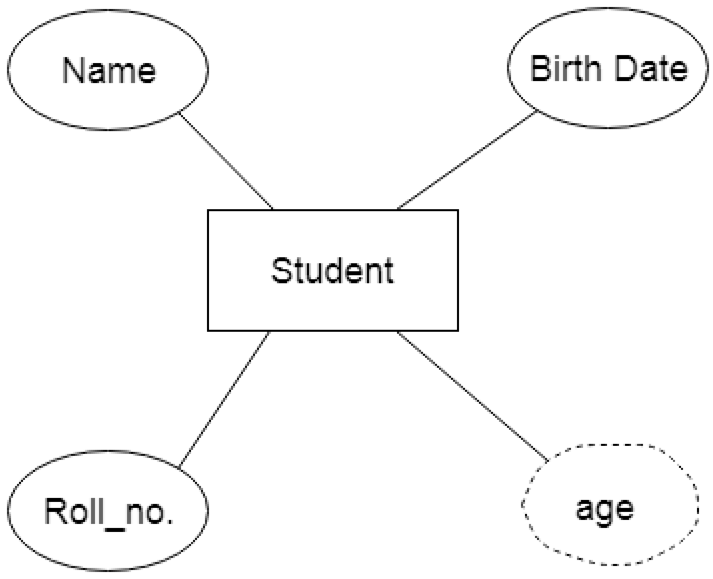
For example, a student can have more than one phone number.



**d. Derived Attribute**

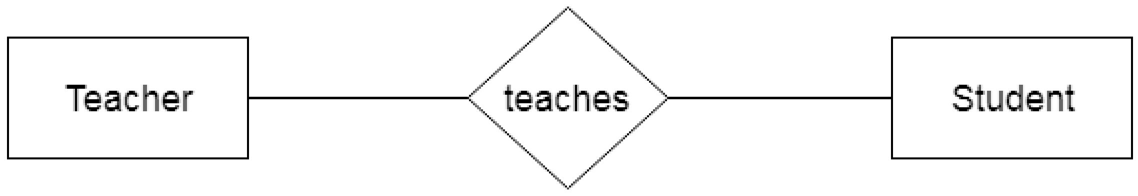
An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.

For example, A person's age changes over time and can be derived from another attribute like Date of birth.



3. Relationship

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

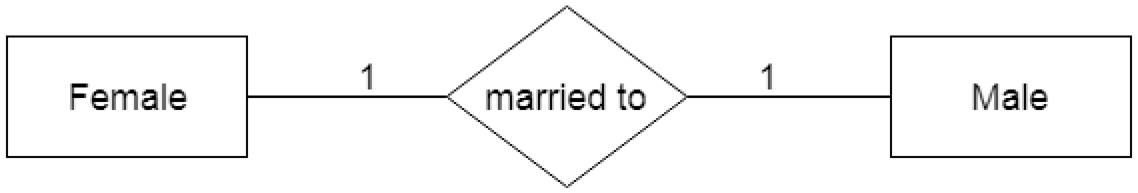


Types of relationship are as follows:

**a. One-to-One Relationship**

When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

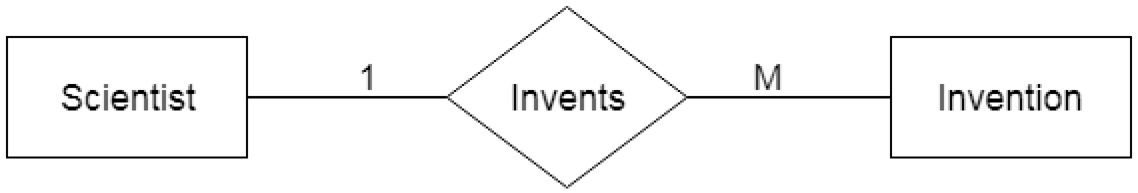
For example, A female can marry to one male, and a male can marry to one female.



**b. One-to-many relationship**

When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

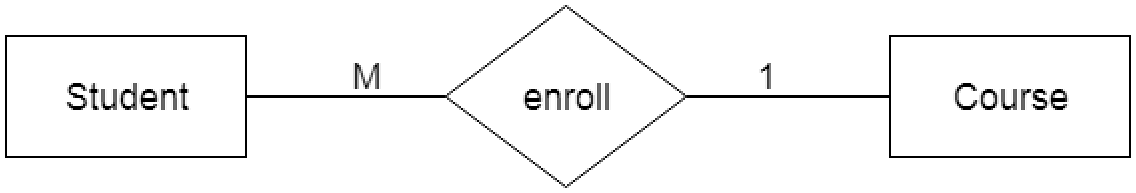
For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.



c. Many-to-one relationship

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

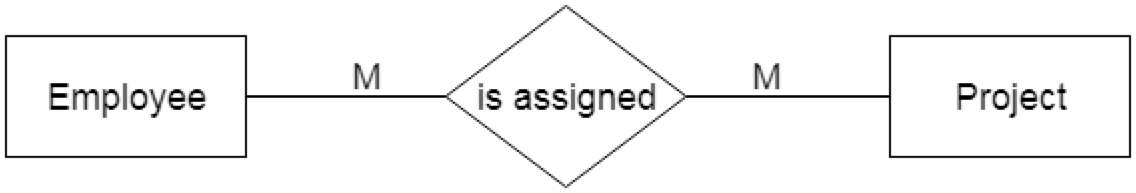
For example, Student enrolls for only one course, but a course can have many students.



**d. Many-to-many relationship**

When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

For example, Employee can assign by many projects and project can have many employees.

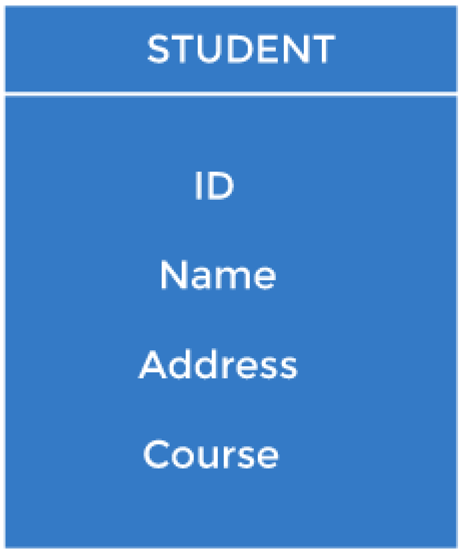


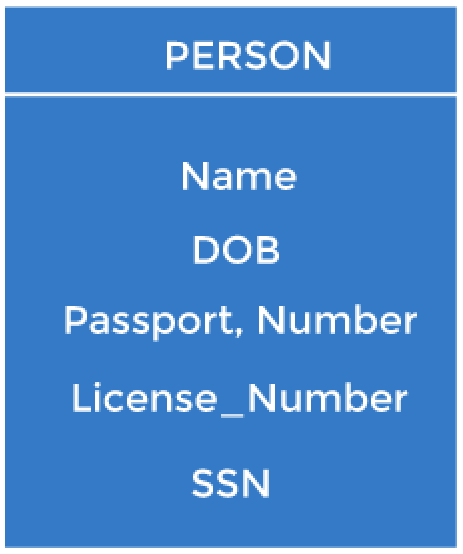
## Keys

- Keys play an important role in the relational database.

○ It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.

For example, ID is used as a key in the Student table because it is unique for each student. In the PERSON table, passport_number, license_number, SSN are keys since they are unique for each person.





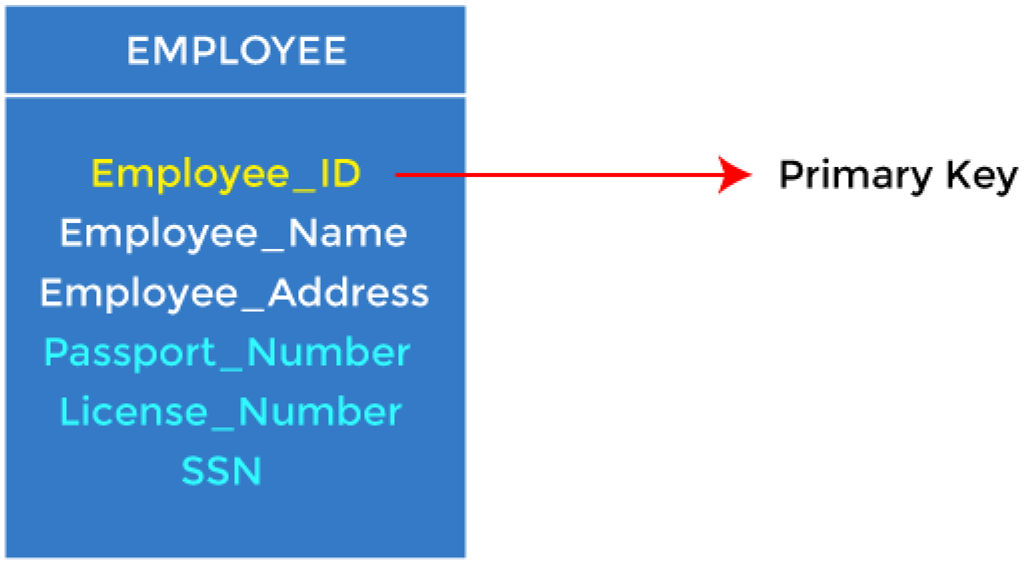
Types of keys:

1. Primary key

○ It is the first key used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys, as we saw in the PERSON table. The key which is most suitable from those lists becomes a primary key.

○ In the EMPLOYEE table, ID can be the primary key since it is unique for each employee. In the EMPLOYEE table, we can even select License_Number and Passport_Number as primary keys since they are also unique.

○ For each entity, the primary key selection is based on requirements and developers.

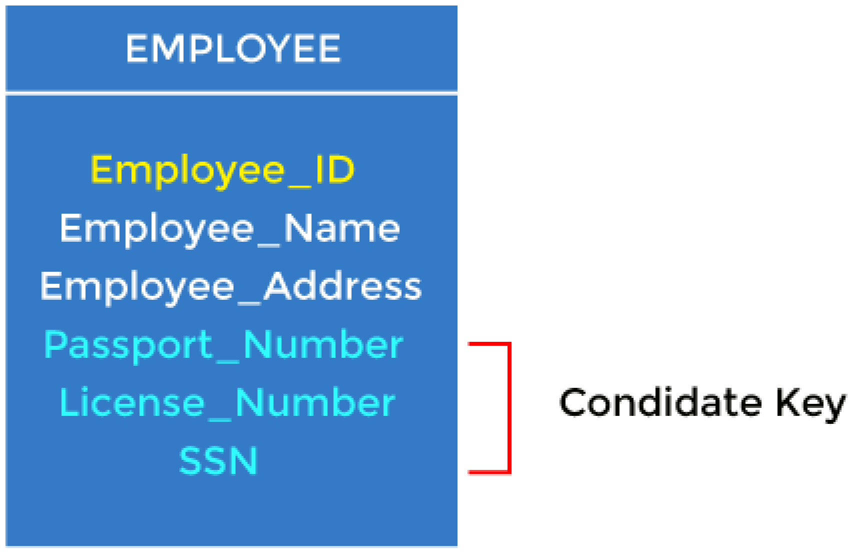


## 2. Candidate key

○ A candidate key is an attribute or set of attributes that can uniquely identify a tuple.

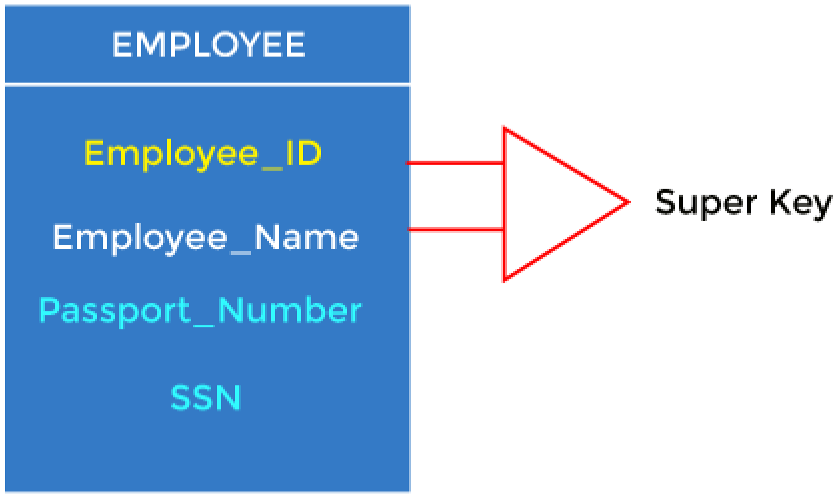
○ Except for the primary key, the remaining attributes are considered a candidate key. The candidate keys are as strong as the primary key.

For example: In the EMPLOYEE table, id is best suited for the primary key. The rest of the attributes, like SSN, Passport_Number, License_Number, etc., are considered a candidate key.



## 3. Super Key

Super key is an attribute set that can uniquely identify a tuple. A super key is a superset of a candidate key.



For example: In the above EMPLOYEE table, for(EMPLOEE_ID, EMPLOYEE_NAME), the name of two employees can be the same, but their EMPLYEE_ID can't be the same. Hence, this combination can also be a key.

The super key would be EMPLOYEE-ID (EMPLOYEE_ID, EMPLOYEE-NAME), etc.

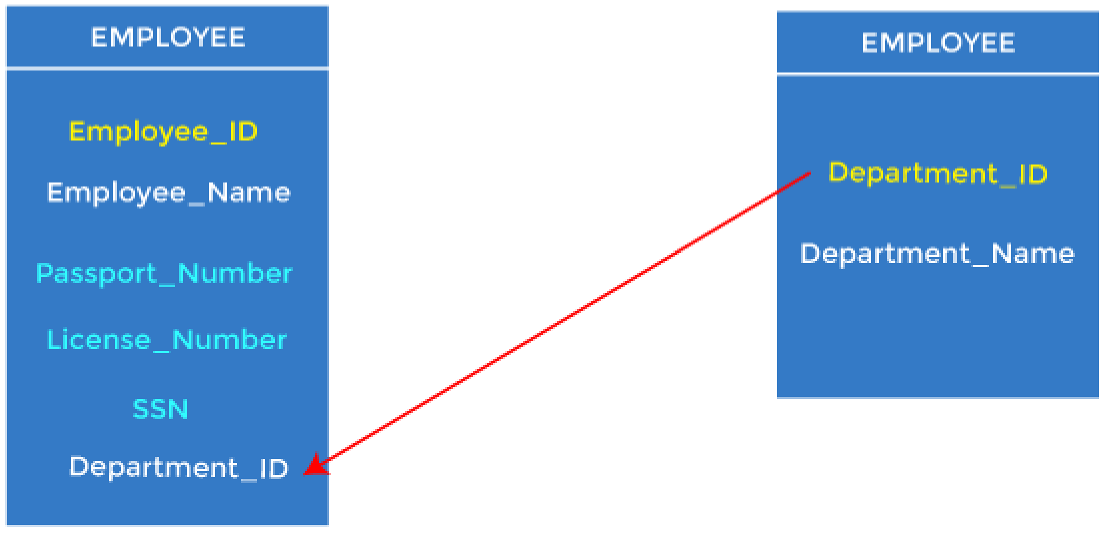
## 4. Foreign key

○ Foreign keys are the column of the table used to point to the primary key of another table.

○ Every employee works in a specific department in a company, and employee and department are two different entities. So we can't store the department's information in the employee table. That's why we link these two tables through the primary key of one table.

○ We add the primary key of the DEPARTMENT table, Department_Id, as a new attribute in the EMPLOYEE table.

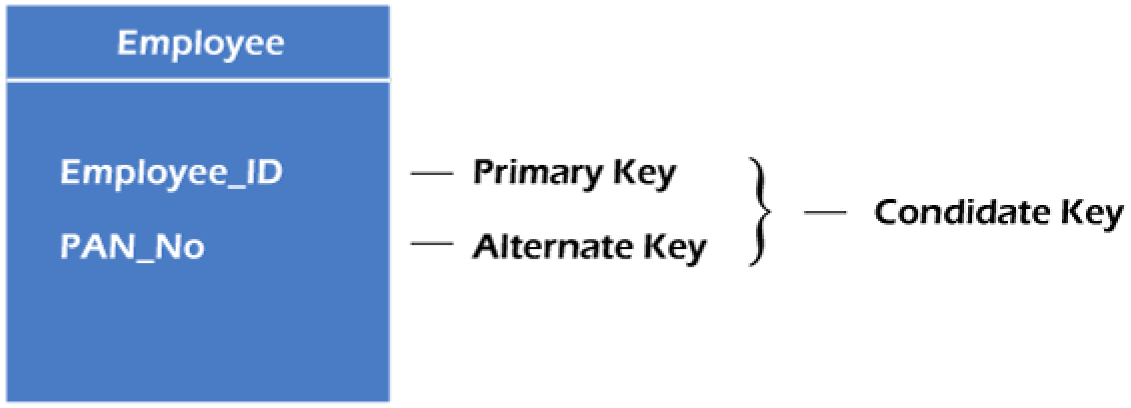
○ In the EMPLOYEE table, Department_Id is the foreign key, and both the tables are related.



## 5. Alternate key

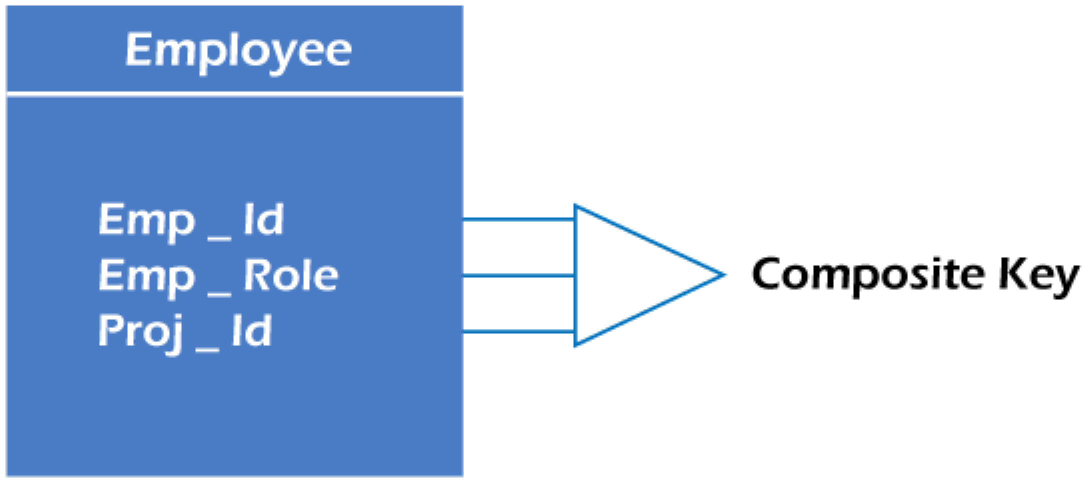
There may be one or more attributes or a combination of attributes that uniquely identify each tuple in a relation. These attributes or combinations of the attributes are called the candidate keys. One key is chosen as the primary key from these candidate keys, and the remaining candidate key, if it exists, is termed the alternate key. In other words, the total number of the alternate keys is the total number of candidate keys minus the primary key. The alternate key may or may not exist. If there is only one candidate key in a relation, it does not have an alternate key.

For example, employee relation has two attributes, Employee_Id and PAN_No, that act as candidate keys. In this relation, Employee_Id is chosen as the primary key, so the other candidate key, PAN_No, acts as the Alternate key.



## 6. Composite key

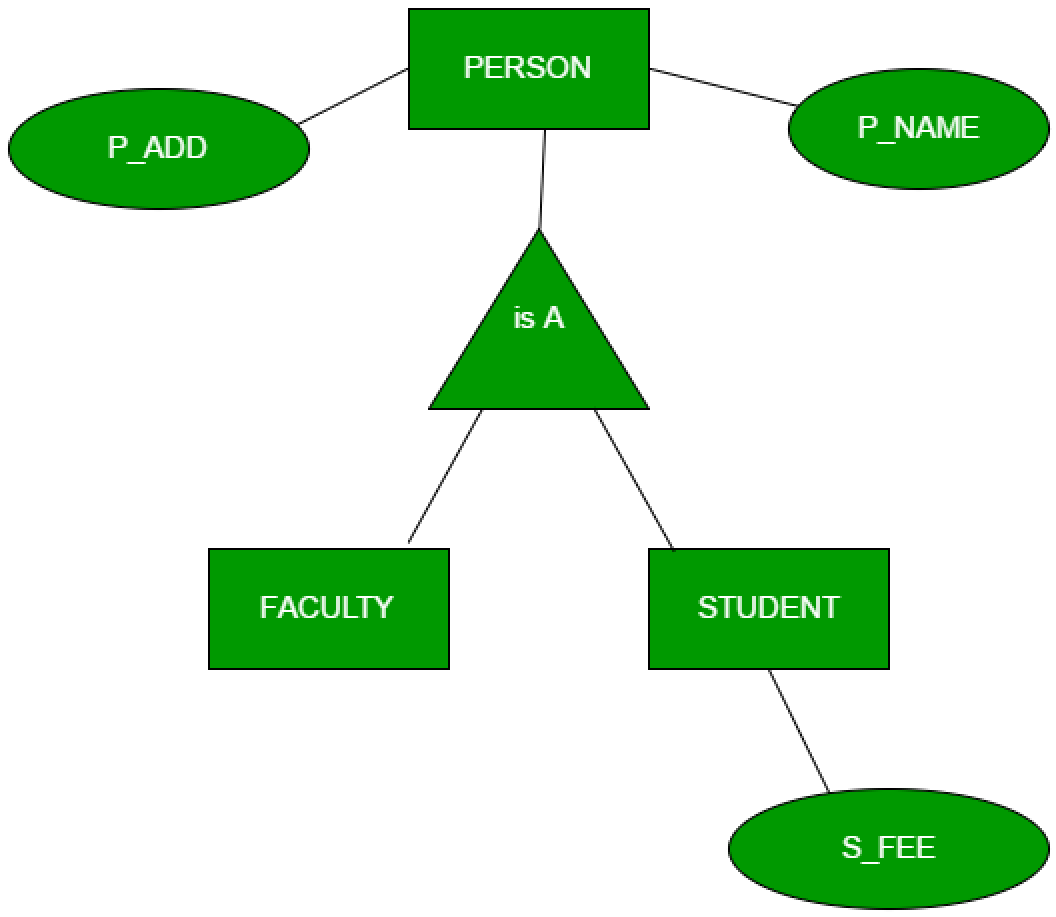
Whenever a primary key consists of more than one attribute, it is known as a composite key. This key is also known as Concatenated Key.



For example, in employee relations, we assume that an employee may be assigned multiple roles, and an employee may work on multiple projects simultaneously. So the primary key will be composed of all three attributes, namely Emp_ID, Emp_role, and Proj_ID in combination. So these attributes act as a composite key since the primary key comprises more than one attribute.

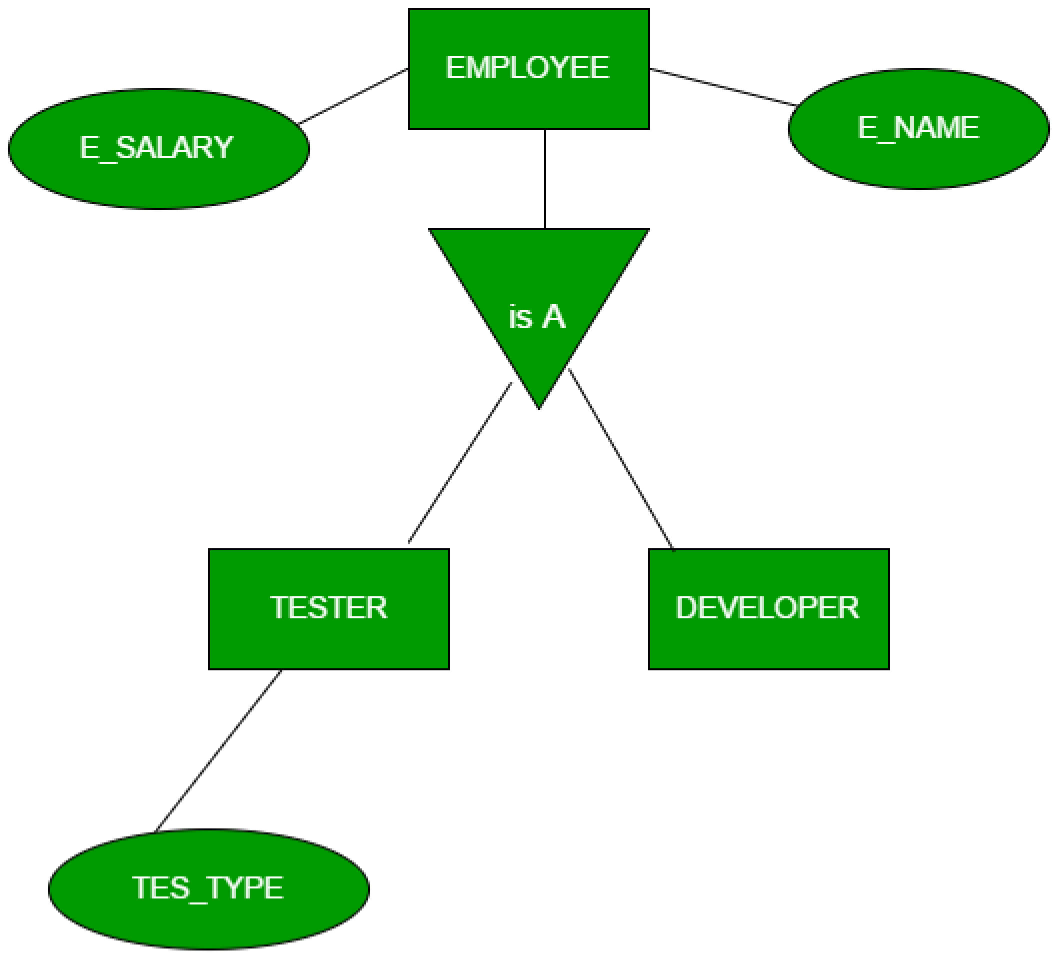
**Additional features of ER model**

**Generalization** –
Generalization is the process of extracting common properties from a set of entities and create 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. For Example, STUDENT and FACULTY can be generalized to a higher level entity called PERSON as shown in Figure 1. In this case, common attributes like P_NAME, P_ADD become part of higher entity (PERSON) and specialized attributes like S_FEE become part of specialized entity (STUDENT).



**Specialization** –

In specialization, an entity is divided into sub-entities based on their characteristics. It is a top-down approach where higher level entity is specialized into two or more lower level entities. For Example, EMPLOYEE entity in an Employee management system can be specialized into DEVELOPER, TESTER etc. as shown in Figure 2. In this case, common attributes like E_NAME, E_SAL etc. become part of higher entity (EMPLOYEE) and specialized attributes like TES_TYPE become part of specialized entity (TESTER).



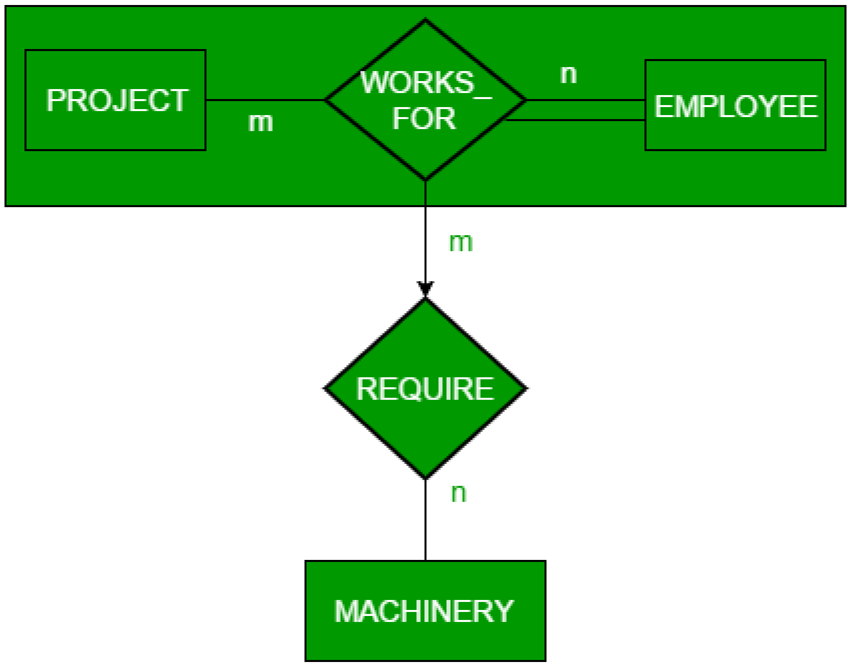
Specialization

**Aggregation** –

An ER diagram is not capable of representing relationship between an entity and a relationship which may be required in some scenarios. In those cases, a relationship with its corresponding entities is aggregated into a higher level entity. Aggregation is an abstraction through which we can represent relationships as higher level entity sets.

For Example, Employee working for a project may require some machinery. So, REQUIRE relationship is needed between relationship WORKS_FOR and entity

MACHINERY. Using aggregation, WORKS_FOR relationship with its entities EMPLOYEE and PROJECT is aggregated into single entity and relationship REQUIRE is created between aggregated entity and MACHINERY.



Aggregation

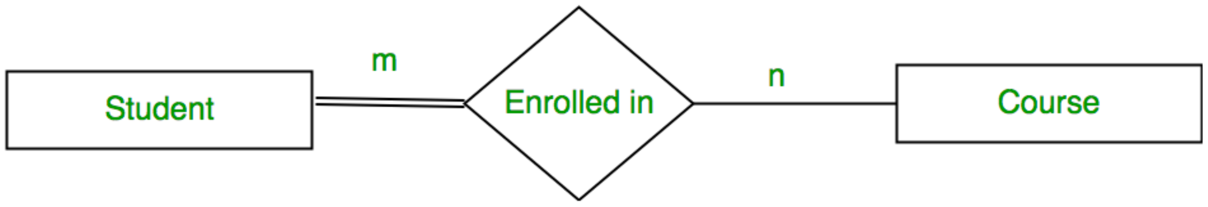
**Participation Constraint**

Participation Constraint is applied to the entity participating in the relationship set.

**1. Total Participation** – Each entity in the entity set must participate in the relationship. If each student must enroll in a course, the participation of students will be total. Total participation is shown by a double line in the ER diagram.

**2. Partial Participation** – The entity in the entity set may or may NOT participate in the relationship. If some courses are not enrolled by any of the students, the participation in the course will be partial.

The diagram depicts the ‘Enrolled in’ relationship set with Student Entity set having total participation and Course Entity set having partial participation.



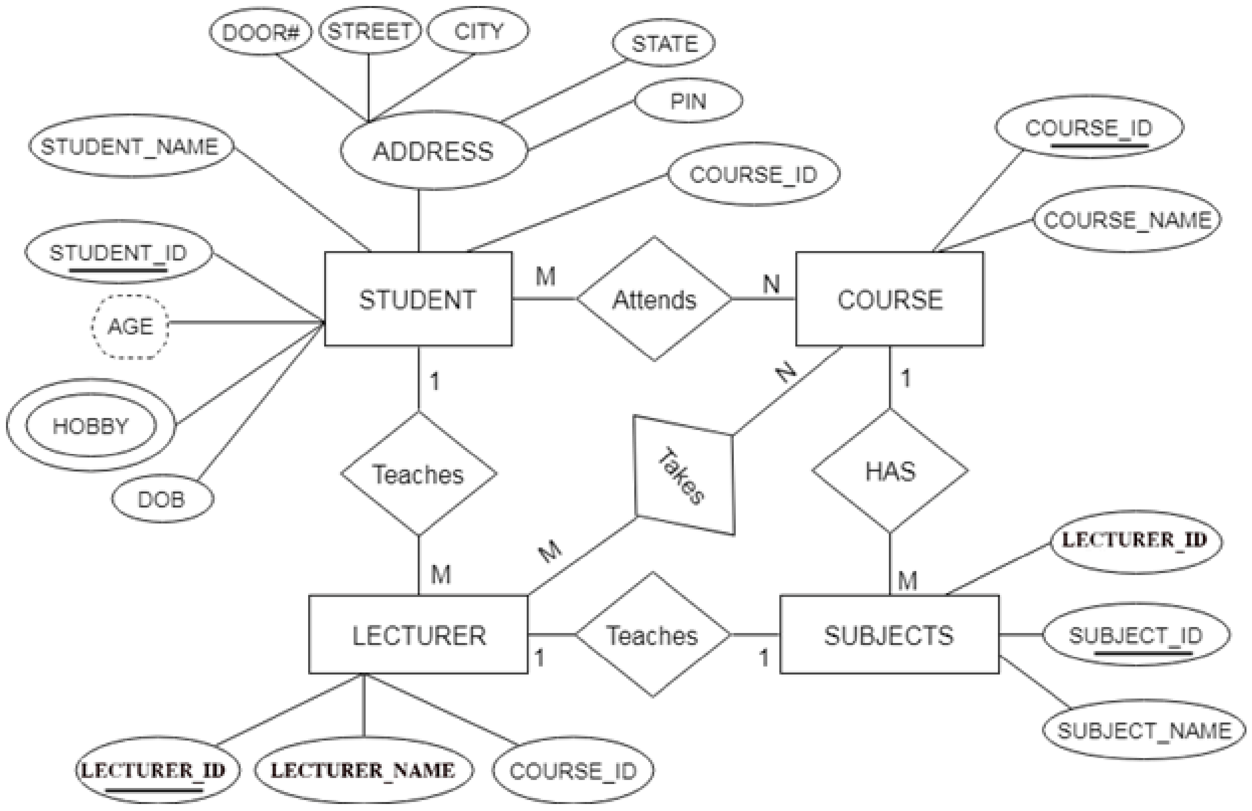
*Total Participation and Partial Participation*

*Reduction of ER diagram to Table*

*The database can be represented using the notations, and these notations can be reduced to a collection of tables.*

*In the database, every entity set or relationship set can be represented in tabular form.*

*The ER diagram is given below:*



There are some points for converting the ER diagram to the table:

○ **Entity type becomes a table.**

In the given ER diagram, LECTURE, STUDENT, SUBJECT and COURSE forms individual tables.

○ **All single-valued attribute becomes a column for the table.**

In the STUDENT entity, STUDENT_NAME and STUDENT_ID form the column of STUDENT table. Similarly, COURSE_NAME and COURSE_ID form the column of COURSE table and so on.

○ A key attribute of the entity type represented by the primary key.

In the given ER diagram, COURSE_ID, STUDENT_ID, SUBJECT_ID, and LECTURE_ID are the key attribute of the entity.

- **The multivalued attribute is represented by a separate table.**

In the student table, a hobby is a multivalued attribute. So it is not possible to represent multiple values in a single column of STUDENT table. Hence we create a table STUD_HOBBY with column name STUDENT_ID and HOBBY. Using both the column, we create a composite key.

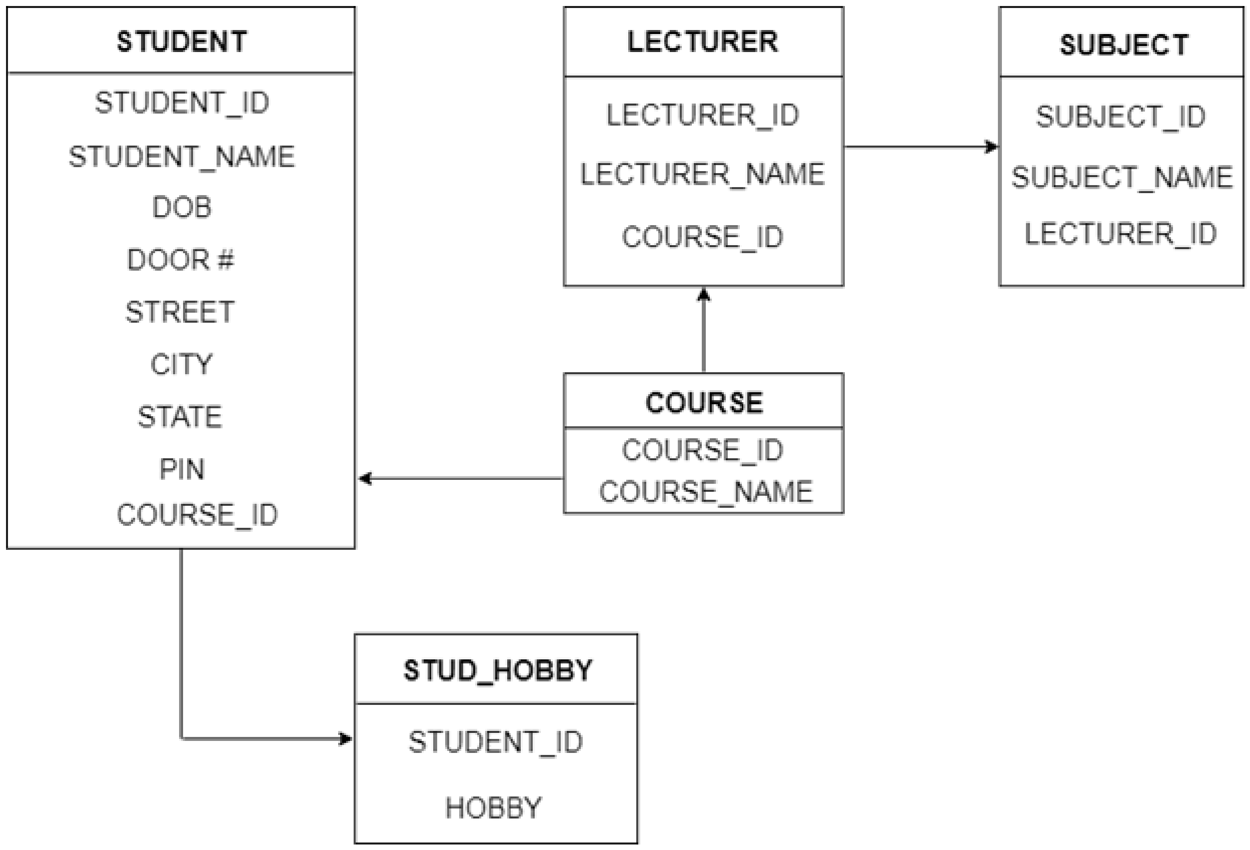
- **Composite attribute represented by components.**

In the given ER diagram, student address is a composite attribute. It contains CITY, PIN, DOOR#, STREET, and STATE. In the STUDENT table, these attributes can merge as an individual column.

- **Derived attributes are not considered in the table.**

In the STUDENT table, Age is the derived attribute. It can be calculated at any point of time by calculating the difference between current date and Date of Birth.

Using these rules, you can convert the ER diagram to tables and columns and assign the mapping between the tables. Table structure for the given ER diagram is as below:


STUDENT
STUDENT_ID
STUDENT_NAME
DOB
DOOR #
STREET
CITY
STATE
PIN
COURSE_ID
LECTURER
LECTURER_ID
LECTURER_NAME
COURSE_ID
SUBJECT
SUBJECT_ID
SUBJECT_NAME
LECTURER_ID
COURSE
COURSE_ID
COURSE_NAME
STUD_HOBBY
STUDENT_ID
HOBBY