Sairam INSTITUTIONS

Sri SAI RAM ENGINEERING COLLEGE
INSTITUTE OF TECHNOLOGY
West Tambaram, Chennai - 44

YEAR	SEM
2	3

COURSE CODE 20CSPC402

DATABASE MANAGEMENT SYSTEM

UNIT No. 1

DATABASE DESIGN

Version: 1.XX

4 QUALITY EDUCATION

UNIT 1

Database Design

1.1 Purpose of Database System, Views of Data

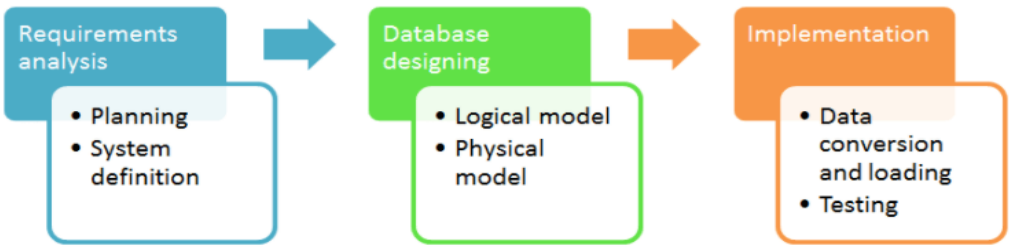
Database design can be generally defined as a collection of tasks or processes that enhance the designing, development, implementation, and maintenance of enterprise data management system. Designing a proper database reduces the maintenance cost thereby improving data consistency and the cost-effective measures are greatly influenced in terms of disk storage space. Therefore, there has to be a brilliant concept of designing a database. The designer should follow the constraints and decide how the elements correlate and what kind of data must be stored.

The main objectives behind database designing are to produce physical and logical design models of the proposed database system. To elaborate this, the logical model is primarily concentrated on the requirements of data and the considerations must be made in terms of monolithic considerations and hence the stored physical data must be stored independent of the physical conditions. On the other hand, the physical database design model includes a translation of the logical design model of the database by keep control of physical media using hardware resources and software systems such as Database Management System (DBMS).

IMPORTANCE OF DATABASE DESIGN The important consideration that can be taken into account while emphasizing the importance of database design can be explained in terms of the following points given below.

1. Database designs provide the blueprints of how the data is going to be stored in a system. A proper design of a database highly affects the overall performance of any application.
2. The designing principles defined for a database give a clear idea of the behavior of any application and how the requests are processed.
3. Another instance to emphasize the database design is that a proper database design meets all the requirements of users.
4. Lastly, the processing time of an application is greatly reduced if the constraints of designing a highly efficient database are properly implemented.

LIFE CYCLE



Although, the life cycle of a database is not an important discussion that has to be taken forward in this article because we are focused on the database design. But, before jumping directly on the designing models constituting database design it is important to understand the overall workflow and life-cycle of the database.

Requirement Analysis

First of all, the planning has to be done on what are the basic requirements of the project under which the design of the database has to be taken forward. Thus, they can be defined as:-

Planning - This stage is concerned with planning the entire DDLC (Database Development Life Cycle). The strategic considerations are taken into account before proceeding.

System definition - This stage covers the boundaries and scopes of the proper database after planning.

Database Designing

The next step involves designing the database considering the user-based requirements and splitting them out into various models so that load or heavy dependencies on a single aspect are not imposed. Therefore, there has been some model-centric approach and that's where logical and physical models play a crucial role.

Physical Model - The physical model is concerned with the practices and implementations of the logical model.

Logical Model - This stage is primarily concerned with developing a model based on the proposed requirements. The entire model is designed on paper without any implementation or adopting DBMS considerations.

Implementation

The last step covers the implementation methods and checking out the behavior that matches our requirements. It is ensured with continuous integration testing of the database with different data sets and conversion of data into machine understandable language. The manipulation of data is primarily focused on these steps where queries are made to run and check if the application is designed satisfactorily or not.

Data conversion and loading - This section is used to import and convert data from the old to the new system.

Testing - This stage is concerned with error identification in the newly implemented system. Testing is a crucial step because it checks the database directly and compares the requirement specifications.

DATABASE DESIGN PROCESS

The process of designing a database carries various conceptual approaches that are needed to be kept in mind. An ideal and well-structured database design must be able to:

1. Save disk space by eliminating redundant data.
2. Maintains data integrity and accuracy.
3. Provides data access in useful ways.
4. Comparing Logical and Physical data models.

Logical

A logical data model generally describes the data in as many details as possible, without having to be concerned about the physical implementations in the database. Features of logical data model might include:

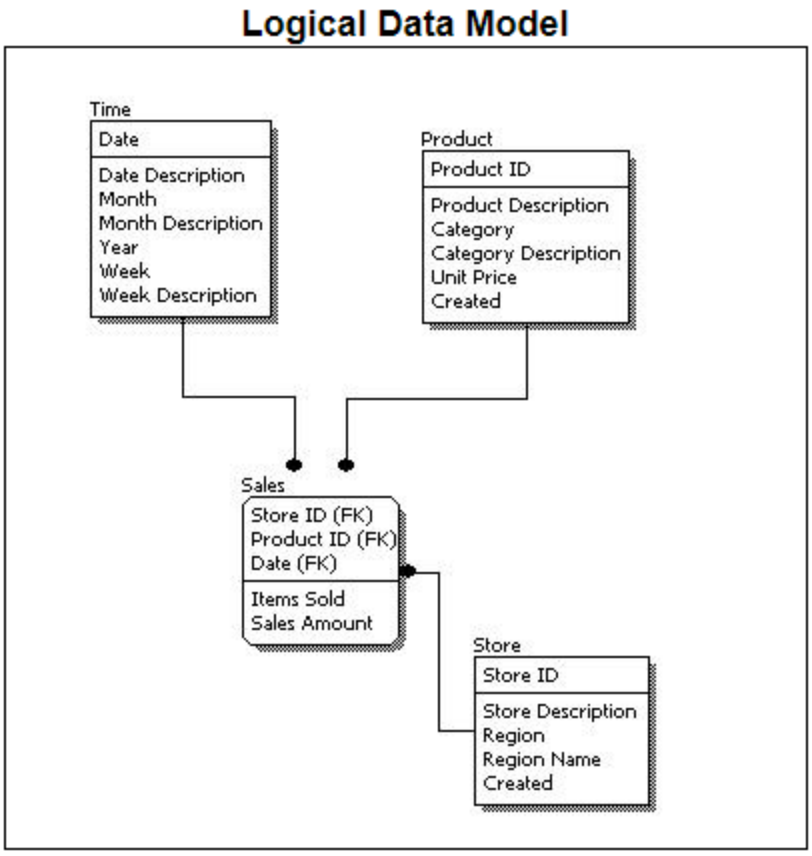
1. All the entities and relationships amongst them.
2. Each entity has well-specified attributes.
3. The primary key for each entity is specified.
4. Foreign keys which are used to identify a relationship between different entities are specified.
5. Normalization occurs at this level.

A logical model can be designed using the following approach:

1. Specify all the entities with primary keys.
2. Specify concurrent relationships between different entities.
3. Figure out each entity attributes
4. Resolve many-to-many relationships.
5. Carry out the process of normalization.

Also, one important factor after following the above approach is to critically examine the design based on requirement gathering. If the above steps are strictly followed, there are chances of creating a highly efficient database design that follows the native approach.

To understand these points, see the image below to get a clear picture.



If we compare the logical data model as shown in the figure above with some sample data in the diagram, we can come up with facts that in a conceptual data model there are no presence of a primary key whereas a logical data model has primary keys for all of its attributes. Also, logical data model the cover relationship between different entities and carries room for foreign keys to establish relationships among them.

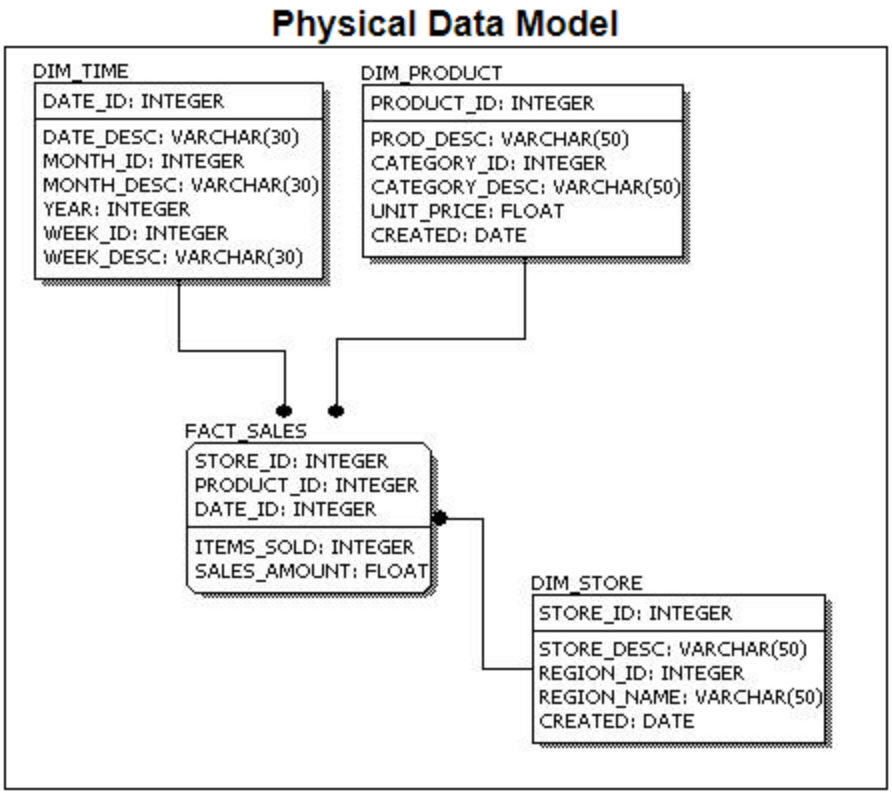
Physical

A Physical data mode generally represents how the approach or concept of designing the database. The main purpose of the physical data model is to show all the **structures** of the table including the **column name, column data type, constraints, keys(primary and foreign)**, and the relationship among tables. The following are the features of a physical data model:

1. Specifies all the columns and tables.
2. Specifies foreign keys that usually define the relationship between tables.
3. Based on user requirements, de-normalization might occur.
4. Since the physical consideration is taken into account so there will straightforward reasons for difference than a logical model.
5. Physical models might be different for different RDBMS. For example, the data type column may be different in MySQL and SQL Server.

While designing a physical data model, the following points should be taken into consideration:

1. Convert the entities into tables.
2. Convert the defined relationships into foreign keys.
3. Convert the data attributes into columns.
4. Modify the data model constraints based on physical requirements.



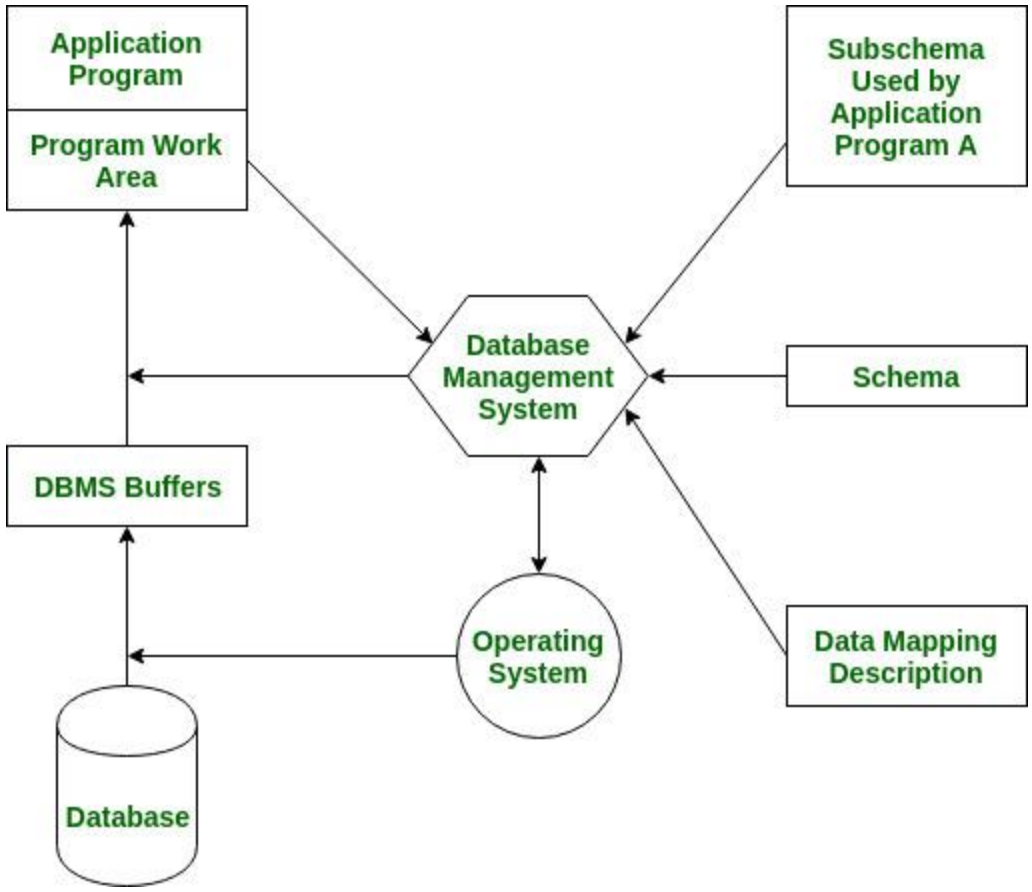
Comparing this physical data model with the logical with the previous logical model, we might conclude the differences that in a physical database entity names are considered table names and attributes are considered column names. Also, the data type of each column is defined in the physical model depending on the actual database used.

PURPOSE OF DATABASE SYSTEM

In DBMS, database systems provide a safe and effective platform to manage vast amounts of data. Their role is to provide services like data organization, storage, and manipulation, as well as to guarantee data integrity. A database system's primary goal is to facilitate data retrieval and provide a dependable storage platform for essential data.

- Efficient storage and retrieval are allowed by structured organization of data through database systems utilizing predefined schemas and data models.
- DBMS maintains the reliability and accuracy of the information and returns it through enforced constraints and rules defined in the database schema that eliminates data redundancy and anomalies, respectively.

- Protecting confidential data is crucial and database systems successfully achieve this with their safeguards against unauthorized access.
- Database systems prioritize the security of sensitive data with their solid mechanisms in place to preserve data confidentiality.
- The inclusion of strong security measures in database systems ensures the protection of sensitive data and upholds its confidentiality. Confidentiality and privacy of data are maintained by utilizing resilient security measures within database systems.
- Collaboration made easy with DBMS. With the provision of a platform to access and manipulate data, multiple users can now work together and ensure data consistency across various applications. Data sharing and collaboration are now synonymous with the help of DBMS.
- Data backups and transaction management are mechanisms provided by database systems to ensure data durability. Safeguarding data against system crashes and failures is their main priority.



1.2 Views of data

VIEW OF DATA IN DBMS

Abstraction is one of the main features of database systems. Hiding irrelevant details from user and providing abstract view of data to users, helps in easy and efficient **user-database** interaction. In the previous tutorial, we discussed the three level of DBMS architecture, The top level of that architecture is "view level". The view level provides the "**view of data**" to

the users and hides the irrelevant details such as data relationship, database schema, constraints, security etc from the user.

To fully understand the view of data, you must have a basic knowledge of data abstraction and instance & schema. Refer these two tutorials to learn them in detail.

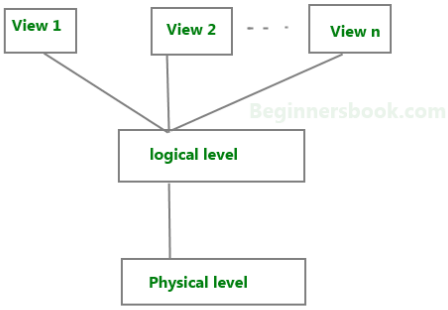
1. **Data abstraction**:Database systems are made-up of complex data structures. To ease the user interaction with database, the developers hide internal irrelevant details from users. This process of hiding irrelevant details from user is called data abstraction.
2. **Instance and schema**: Design of a database is called the schema. Schema is of three types: Physical schema, logical schema and view schema. The data stored in database at a particular moment of time is called instance of database. Database schema defines the variable declarations in tables that belong to a particular database; the value of these variables at a moment of time is called the instance of that database.

DATA ABSTRACTION IN DBMS

Database systems are made-up of complex data structures. To ease the user interaction with database, the developers hide internal irrelevant details from users. This process of hiding irrelevant details from user is called **data abstraction**. The term "irrelevant" used here with respect to the user, it doesn't mean that the hidden data is not relevant with regard to the whole database. It just means that the **user is not concerned about that data**.

For example: When you are booking a train ticket, you are not concerned how data is processing at the back end when you click "book ticket", what processes are happening when you are doing online payments. **You are just concerned about the message that pops up when your ticket is successfully booked**. This doesn't mean that the process happening at the back end is not relevant, it just means that you as a user are not concerned what is happening in the database.

THREE LEVELS OF ABSTRACTION



Three Levels of data abstraction

Physical level: This is the lowest level of data abstraction. It describes how data is actually stored in database. You can get the complex data structure details at this level.

Logical level: This is the middle level of 3-level data abstraction architecture. It describes what data is stored in database.

View level: Highest level of data abstraction. This level describes the user interaction with database system.

Example: Let's say we are storing customer information in a customer table. At **physical level** these records can be described as blocks of storage (bytes, gigabytes, terabytes etc.) in memory. These details are often hidden from the programmers.

At the **logical level** these records can be described as fields and attributes along with their data types, their relationship among each other can be logically implemented. The programmers generally work at this level because they are aware of such things about database systems.

At **view level**, user just interact with system with the help of GUI and enter the details at the screen, they are not aware of how the data is stored and what data is stored; such details are hidden from them.

Instance and schema in DBMS

DBMS Schema

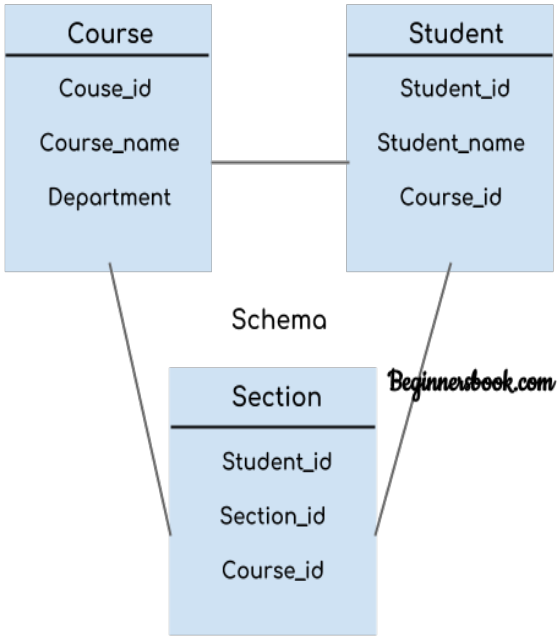
Definition of schema: Design of a database is called the schema. For example: An **employee** table in database exists with the following attributes:

```
EMP_NAME  EMP_ID   EMP_ADDRESS  EMP_CONTACT
--------  ------   -----------  -----------
```

This is the schema of the **employee** table. Schema defines the attributes of tables in the database. **Schema is of three types: Physical schema, logical schema and view schema.**

- Schema represents the **logical view** of the database. It helps you understand what data needs to go where.
- Schema can be represented by a diagram as shown below.
- Schema **helps the database users to understand the relationship between data**. This helps in efficiently performing operations on database such as insert, update, delete, search etc.

In the following diagram, we have a schema that shows the relationship between **three tables: Course, Student and Section**. The diagram only shows the design of the database, it doesn't show the data present in those tables. Schema is only a structural view(design) of a database as shown in the diagram below.



The design of a database at physical level is called **physical schema**, how the data stored in blocks of storage is described at this level.

Design of database at logical level is called **logical schema**, programmers and database administrators work at this level, at this level data can be described as certain types of data records gets stored in data structures, however the internal details such as implementation of data structure is hidden at this level (available at physical level).

Design of database at view level is called **view schema**. This generally describes end user interaction with database systems.

To learn more about these schemas, refer 3 level data abstraction architecture.

DBMS Instance

Definition of instance: The data stored in database at a particular moment of time is called instance of database. Database schema defines the attributes in tables that belong to a particular database. The value of these attributes at a moment of time is called the instance of that database.

For example, we have seen the schema of table "employee" above. Let's see the table with the data now. At this moment the table contains two rows (records). This is the the current instance of the table "employee" because this is the data that is stored in this table at this particular moment of time.

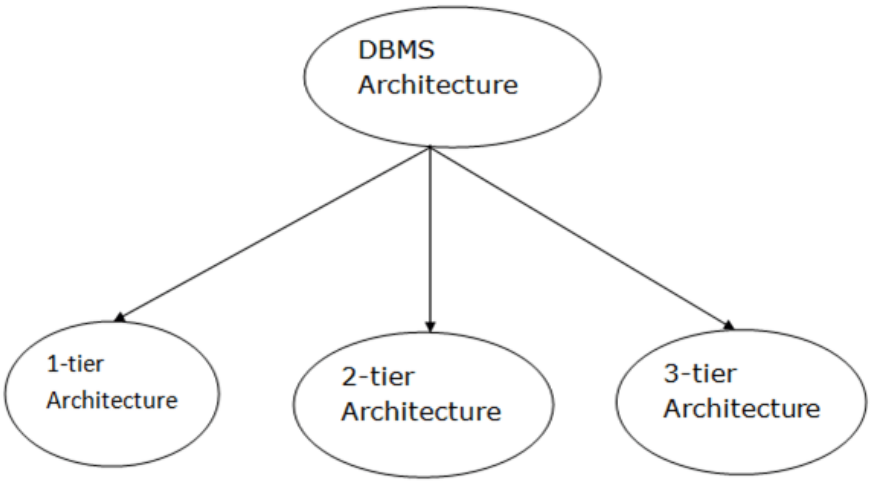
```
EMP_NAME   EMP_ID  EMP_ADDRESS  EMP_CONTACT
-------    ------  -----------  -----------
Chaitanya  101     Noida        95********
Ajeet      102     Delhi        99********
```

Let's take another example: Let's say we have a single table student in the database, today the table has 100 records, so today the instance of the database has 100 records. We are going to add another 100 records in this table by tomorrow so the instance of database tomorrow will have 200 records in table. In short, at a particular moment the data stored in database is called the instance, this changes over time as and when we add, delete or update data in the database.

1.3 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.

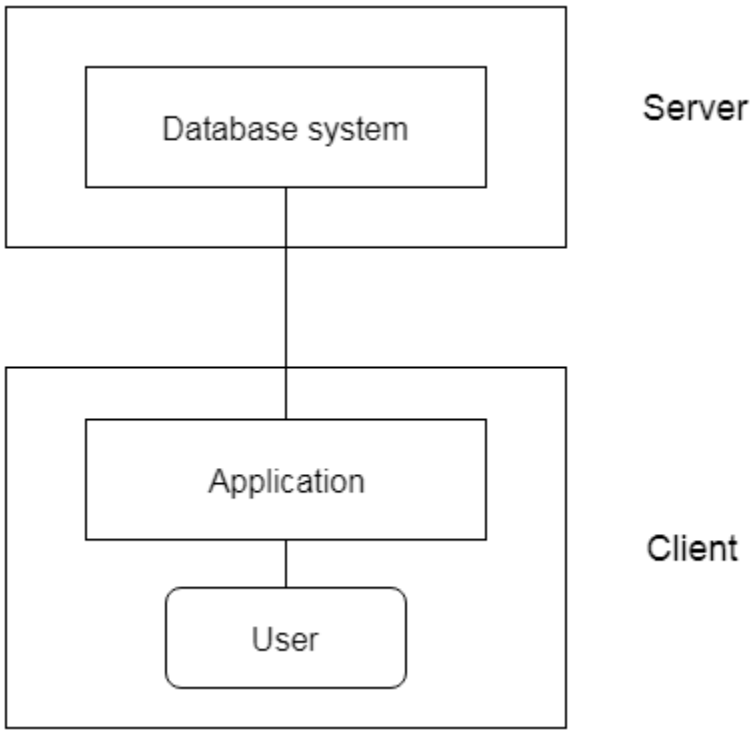


Fig: 2-tier Architecture

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.

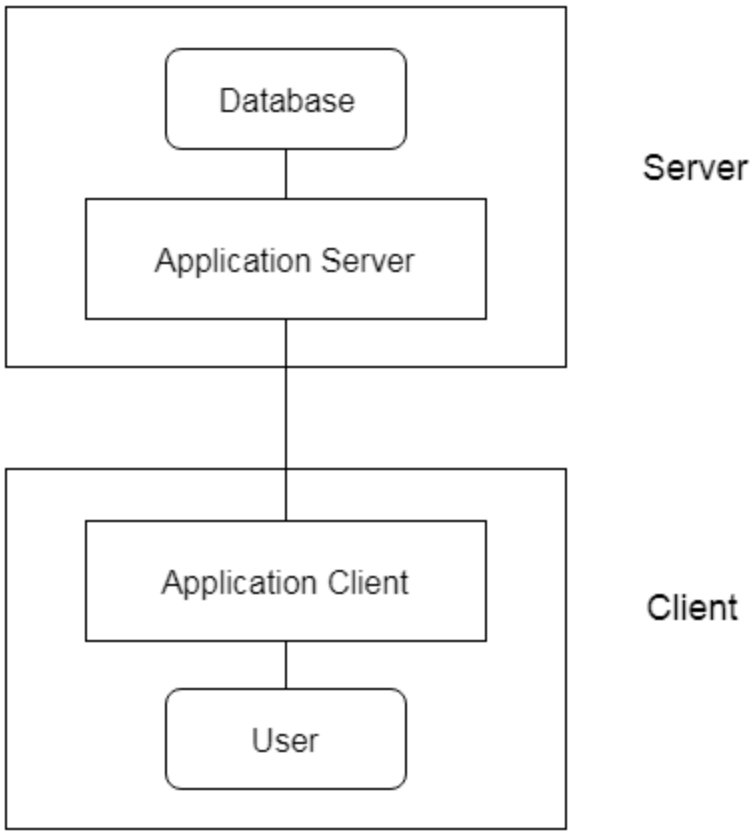
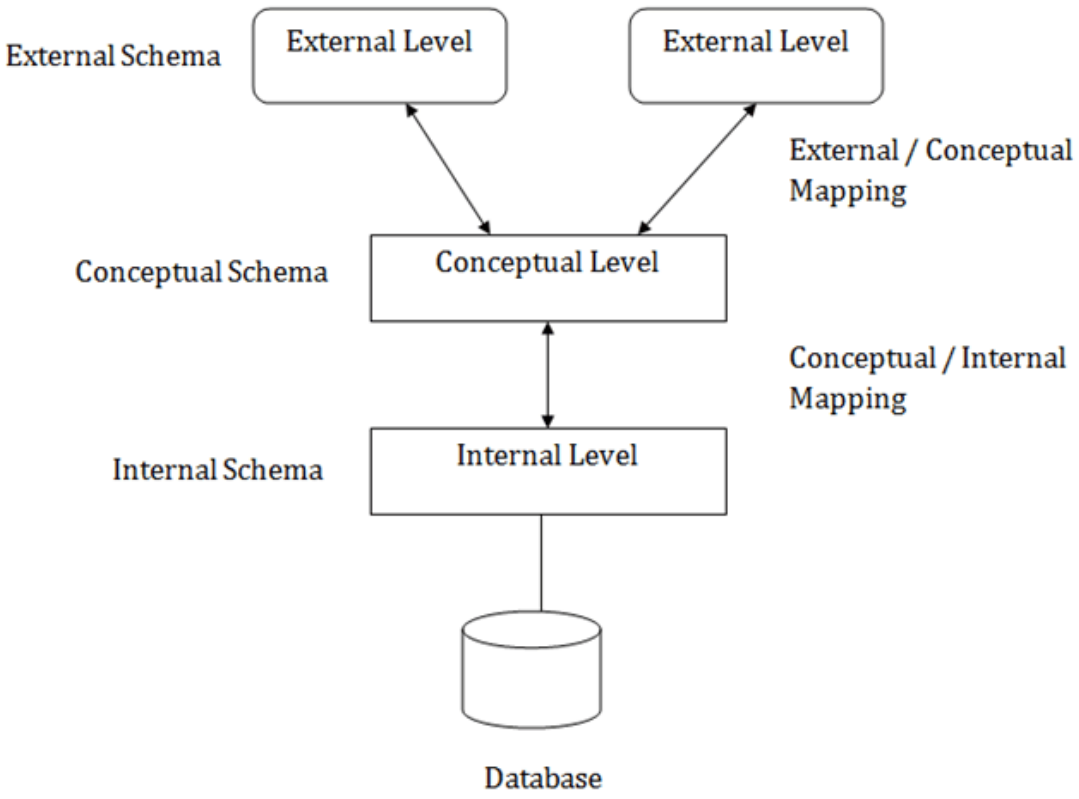


Fig: 3-tier Architecture

THREE SCHEMA ARCHITECTURE

- The three schema architecture is also called ANSI/SPARC architecture or three-level architecture.
- This framework is used to describe the structure of a specific database system.
- The three schema architecture is also used to separate the user applications and physical database.
- The three schema architecture contains three-levels. It breaks the database down into three different categories.

The three-schema architecture is as follows:



In the above diagram:

- It shows the DBMS architecture.
- Mapping is used to transform the request and response between various database levels of architecture.
- Mapping is not good for small DBMS because it takes more time.
- In External / Conceptual mapping, it is necessary to transform the request from external level to conceptual schema.
- In Conceptual / Internal mapping, DBMS transform the request from the conceptual to internal level.

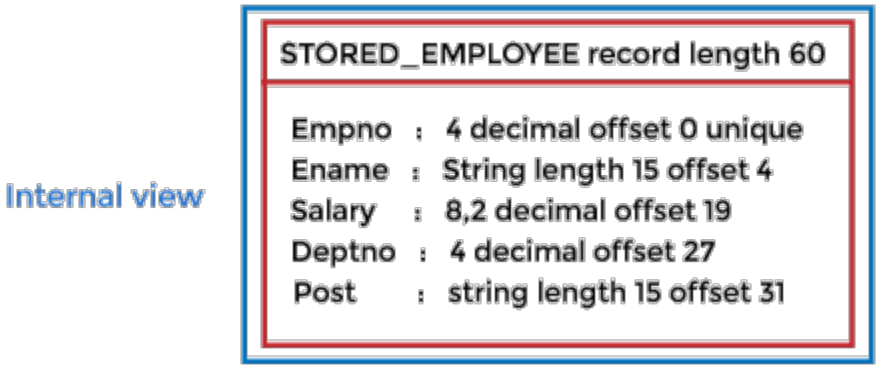
Objectives of Three schema Architecture

The main objective of three level architecture is to enable multiple users to access the same data with a personalized view while storing the underlying data only once. Thus it separates the user's view from the physical structure of the database. This separation is desirable for the following reasons:

- Different users need different views of the same data.
- The approach in which a particular user needs to see the data may change over time.

- The users of the database should not worry about the physical implementation and internal workings of the database such as data compression and encryption techniques, hashing, optimization of the internal structures etc.
- All users should be able to access the same data according to their requirements.
- DBA should be able to change the conceptual structure of the database without affecting the user's
- Internal structure of the database should be unaffected by changes to physical aspects of the storage.

1. Internal Level

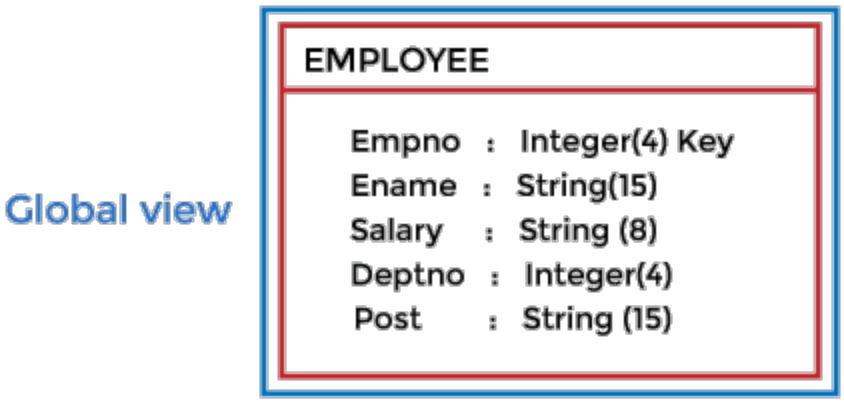


- The internal level has an internal schema which describes the physical storage structure of the database.
- The internal schema is also known as a physical schema.
- It uses the physical data model. It is used to define that how the data will be stored in a block.
- The physical level is used to describe complex low-level data structures in detail.

The internal level is generally is concerned with the following activities:

- Storage space allocations.
  For Example: B-Trees, Hashing etc.
- Access paths.
  For Example: Specification of primary and secondary keys, indexes, pointers and sequencing.
- Data compression and encryption techniques.
- Optimization of internal structures.
- Representation of stored fields.

2. Conceptual Level



- The conceptual schema describes the design of a database at the conceptual level. Conceptual level is also known as logical level.
- The conceptual schema describes the structure of the whole database.
- The conceptual level describes what data are to be stored in the database and also describes what relationship exists among those data.
- In the conceptual level, internal details such as an implementation of the data structure are hidden.
- Programmers and database administrators work at this level.

3. External Level

- At the external level, a database contains several schemas that sometimes called as subschema. The subschema is used to describe the different view of the database.
- An external schema is also known as view schema.
- Each view schema describes the database part that a particular user group is interested and hides the remaining database from that user group.
- The view schema describes the end user interaction with database systems.

Mapping between Views

The three levels of DBMS architecture don't exist independently of each other. There must be correspondence between the three levels i.e. how they actually correspond with each other. DBMS is responsible for correspondence between the three types of schema. This correspondence is called Mapping.

There are basically two types of mapping in the database architecture:

- Conceptual/ Internal Mapping
- External / Conceptual Mapping

Conceptual/ Internal Mapping

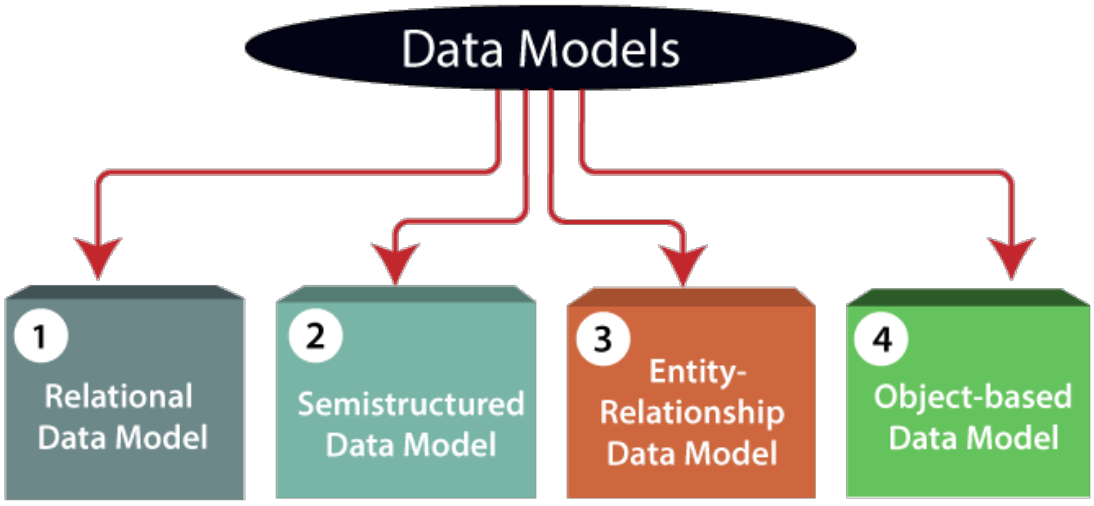
The Conceptual/ Internal Mapping lies between the conceptual level and the internal level. Its role is to define the correspondence between the records and fields of the conceptual level and files and data structures of the internal level.

External/ Conceptual Mapping

The external/Conceptual Mapping lies between the external level and the Conceptual level. Its role is to define the correspondence between a particular external and the conceptual view.

1.4 DATA MODELS

Data Model is the modeling of the data description, data semantics, and consistency constraints of the data. It provides the conceptual tools for describing the design of a database at each level of data abstraction. Therefore, there are following four data models used for understanding the structure of the database:



1) Relational Data Model: This type of model designs the data in the form of rows and columns within a table. Thus, a relational model uses tables for representing data and in-between relationships. Tables are also called relations. This model was initially described by Edgar F. Codd, in 1969. The relational data model is the widely used model which is primarily used by commercial data processing applications.

2) Entity-Relationship Data Model: An ER model is the logical representation of data as objects and relationships among them. These objects are known as entities, and relationship is an

association among these entities. This model was designed by Peter Chen and published in 1976 papers. It was widely used in database designing. A set of attributes describe the entities. For example, student_name, student_id describes the 'student' entity. A set of the same type of entities is known as an 'Entity set', and the set of the same type of relationships is known as 'relationship set'.

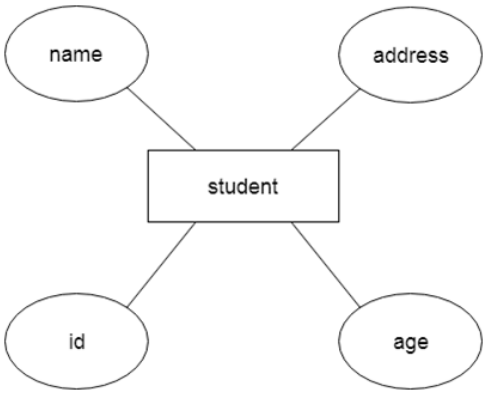
3) Object-based Data Model: An extension of the ER model with notions of functions, encapsulation, and object identity, as well. This model supports a rich type system that includes structured and collection types. Thus, in 1980s, various database systems following the object-oriented approach were developed. Here, the objects are nothing but the data carrying its properties.

4) Semistructured Data Model: This type of data model is different from the other three data models (explained above). The semistructured data model allows the data specifications at places where the individual data items of the same type may have different attributes sets. The Extensible Markup Language, also known as XML, is widely used for representing the semistructured data. Although XML was initially designed for including the markup information to the text document, it gains importance because of its application in the exchange of data.

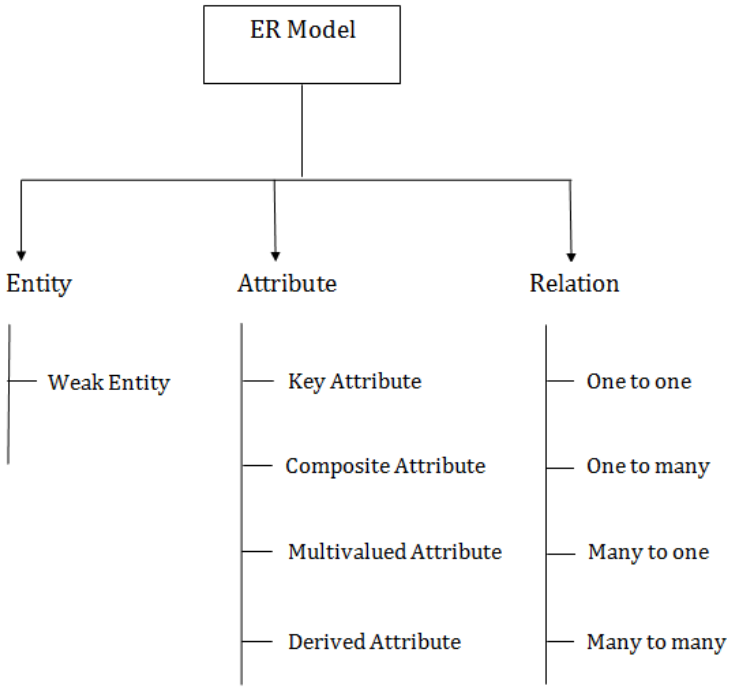
1.5 ER (ENTITY RELATIONSHIP) DIAGRAM IN DBMS

- 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.



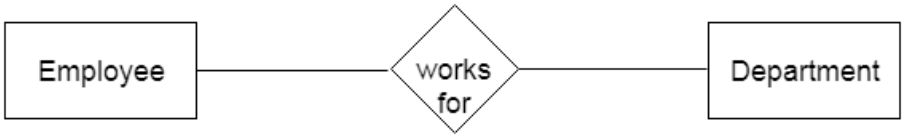
Component of ER Diagram



1. Entity:

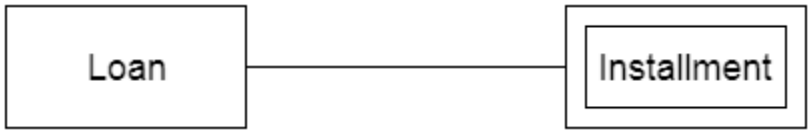
An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

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



a. Weak Entity

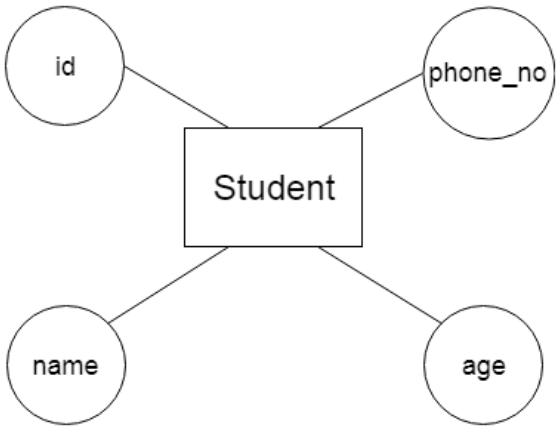
An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.



2. Attribute

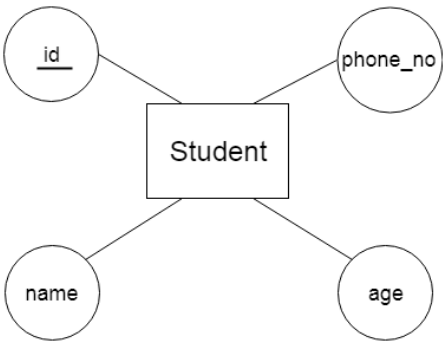
The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

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



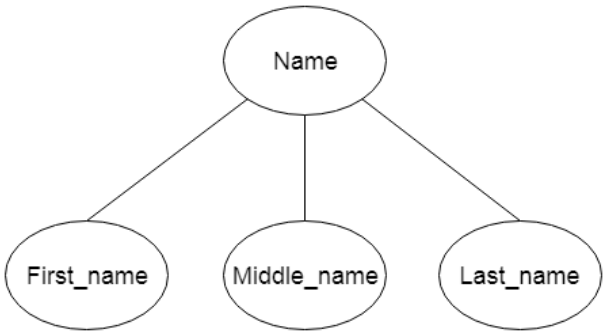
a. Key Attribute

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

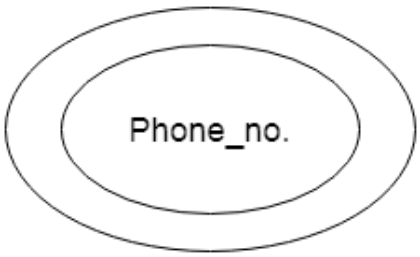
An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



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.

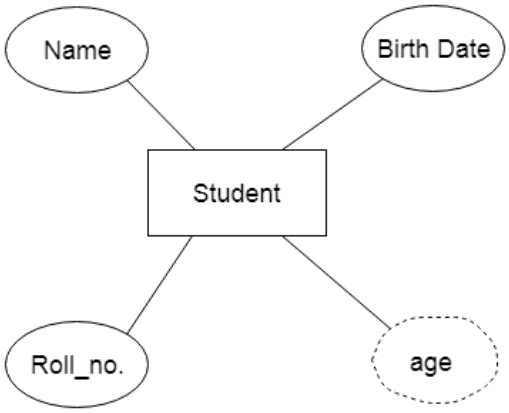
For example, a student can have more than one phone number.



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

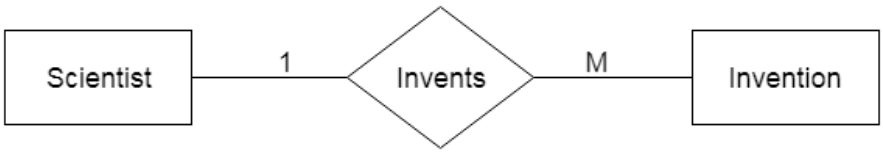
When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

For example, A surgeon will be headed by head of the department.

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.

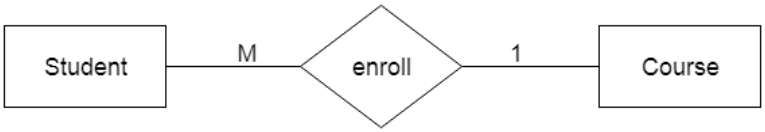
For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.



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.

NOTATION OF ER DIAGRAM

Database can be represented using the notations. In ER diagram, many notations are used to express the cardinality. These notations are as follows:

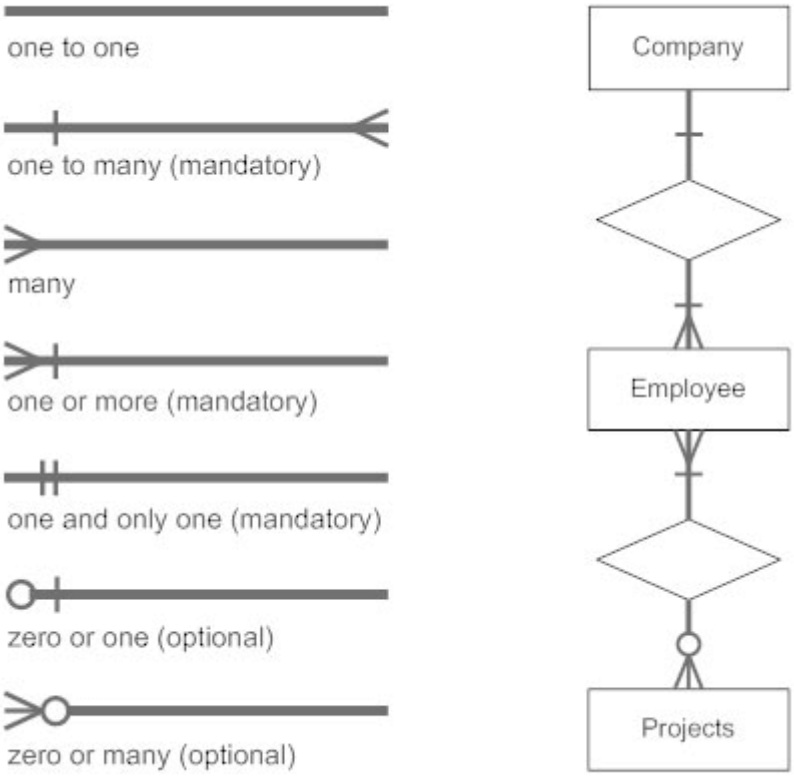
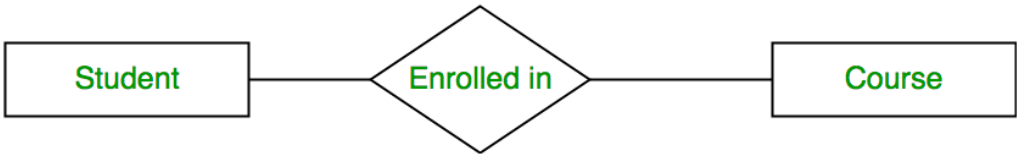


Fig: Notations of ER diagram

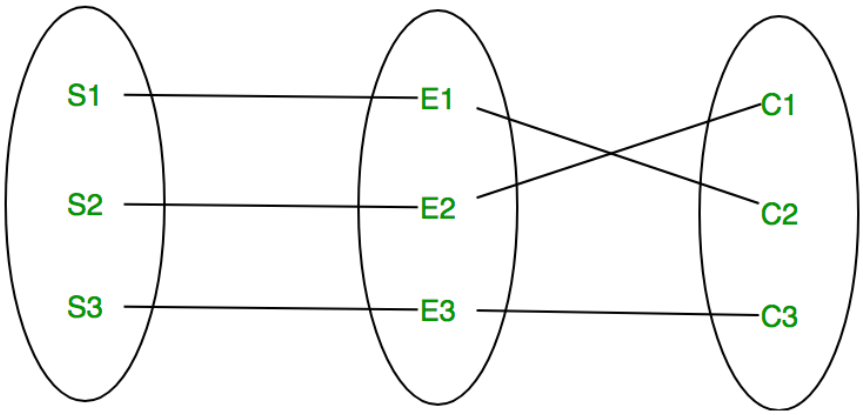
Relationship Type and Relationship Set

A Relationship Type represents the association between entity types. For example, 'Enrolled in' is a relationship type that exists between entity type Student and Course. In ER diagram, the relationship type is represented by a diamond and connecting the entities with lines.



Entity-Relationship Set

A set of relationships of the same type is known as a relationship set. The following relationship set depicts S1 as enrolled in C2, S2 as enrolled in C1, and S3 as registered in C3.

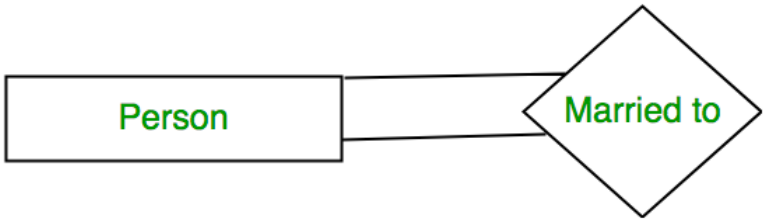


Relationship Set

Degree of a Relationship Set

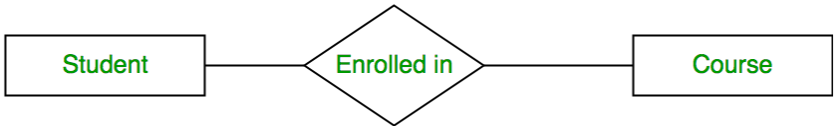
The number of different entity sets participating in a relationship set is called the degree of a relationship set.

1. Unary Relationship: When there is only ONE entity set participating in a relation, the relationship is called a unary relationship. For example, one person is married to only one person.



Unary Relationship

2. Binary Relationship: When there are TWO entities set participating in a relationship, the relationship is called a binary relationship. For example, a Student is enrolled in a Course.



Binary Relationship

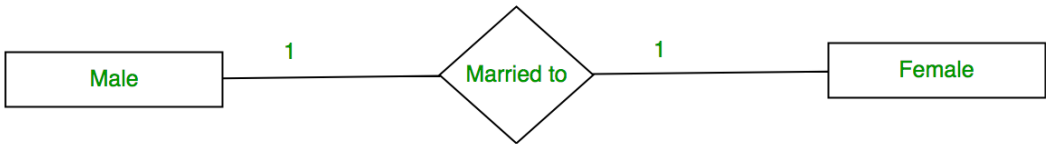
3. n-ary Relationship: When there are n entities set participating in a relation, the relationship is called an n-ary relationship.

Cardinality

The number of times an entity of an entity set participates in a relationship set is known as cardinality. Cardinality can be of different types:

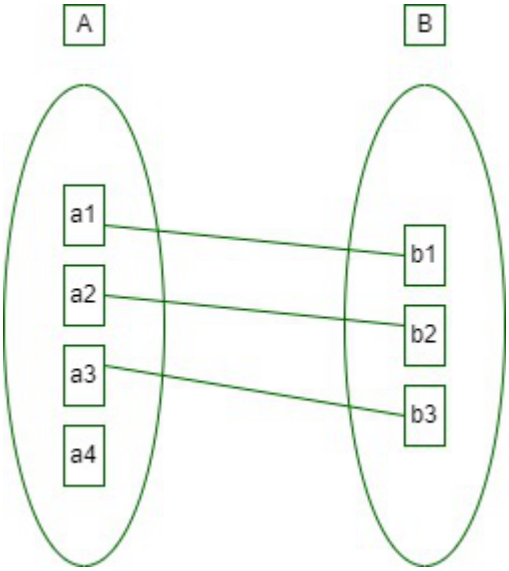
1. One-to-One: When each entity in each entity set can take part only once in the relationship, the cardinality is one-to-one. Let us assume that a male can marry one female and a female can marry one male. So the relationship will be one-to-one.

the total number of tables that can be used in this is 2.



One-to-One Cardinality

Using Sets, it can be represented as:

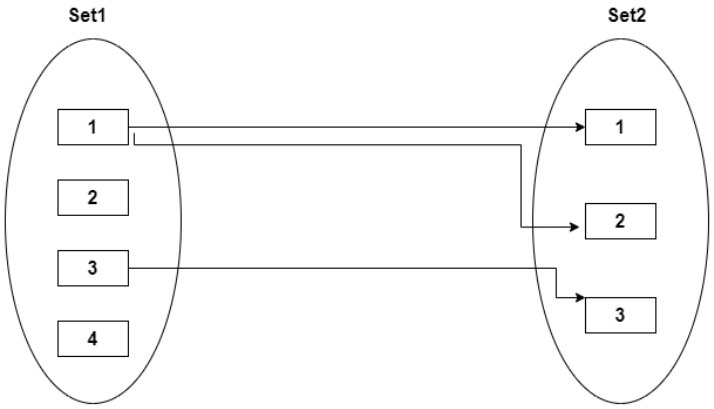


Set Representation of One-to-One

2. One-to-Many: In one-to-many mapping as well where each entity can be related to more than one relationship and the total number of tables that can be used in this is 2.

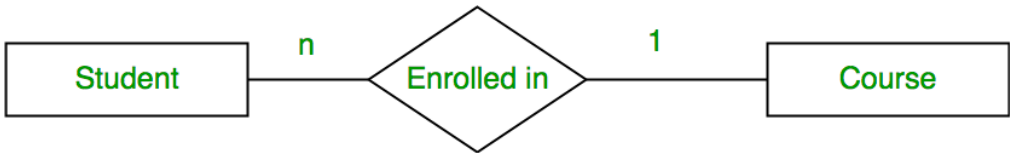
One to Many

Using sets, one-to-many cardinality can be represented as:



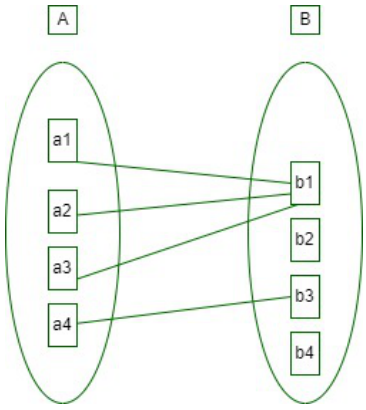
Set Representation of One-to-Many

3. Many-to-One: When entities in one entity set can take part only once in the relationship set and entities in other entity sets can take part more than once in the relationship set, cardinality is many to one. Let us assume that a student can take only one course but one course can be taken by many students. So the cardinality will be n to 1. It means that for one course there can be n students but for one student, there will be only one course.
The total number of tables that can be used in this is 3.



Many-to-One Relationship

Using Sets, it can be represented as:



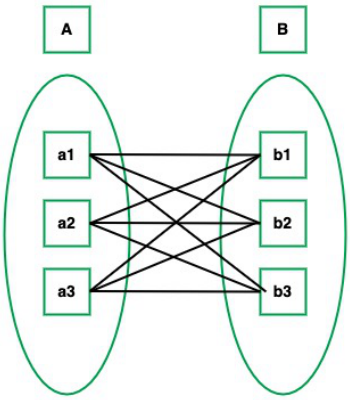
Set Representation of Many-to-One

In this case, each student is taking only 1 course but 1 course has been taken by many students.

4. Many-to-Many: When entities in all entity sets can take part more than once in the relationship cardinality is many to many. Let us assume that a student can take more than one course and one course can be taken by many students. So the relationship will be many to many.
the total number of tables that can be used in this is 3.

Many-to-Many

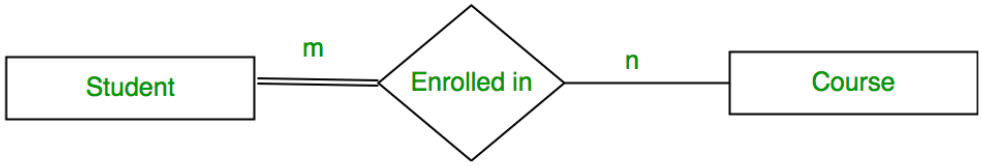
Using Sets, it can be represented as:



Many-to-Many Set Representation

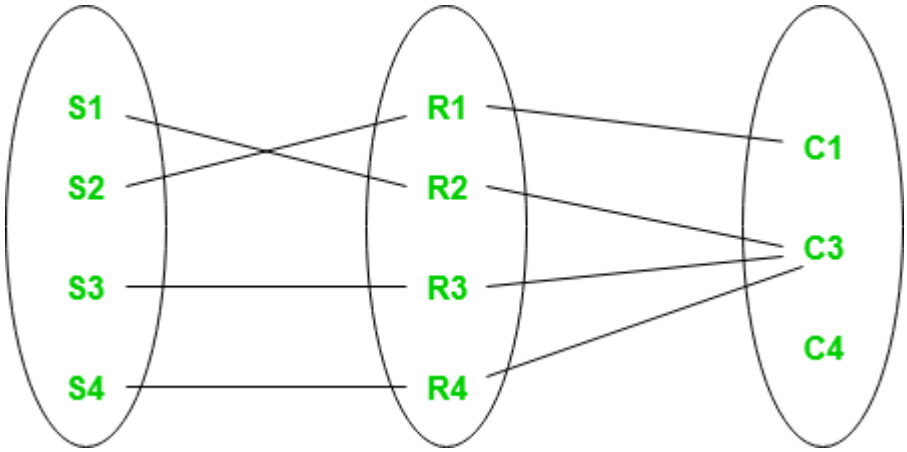
In this example, student S1 is enrolled in C1 and C3 and Course C3 is enrolled by S1, S3, and S4. So it is many-to-many relationships.

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

Using Set, it can be represented as,



Set representation of Total Participation and Partial Participation

Every student in the Student Entity set participates in a relationship but there exists a course C4 that is not taking part in the relationship.

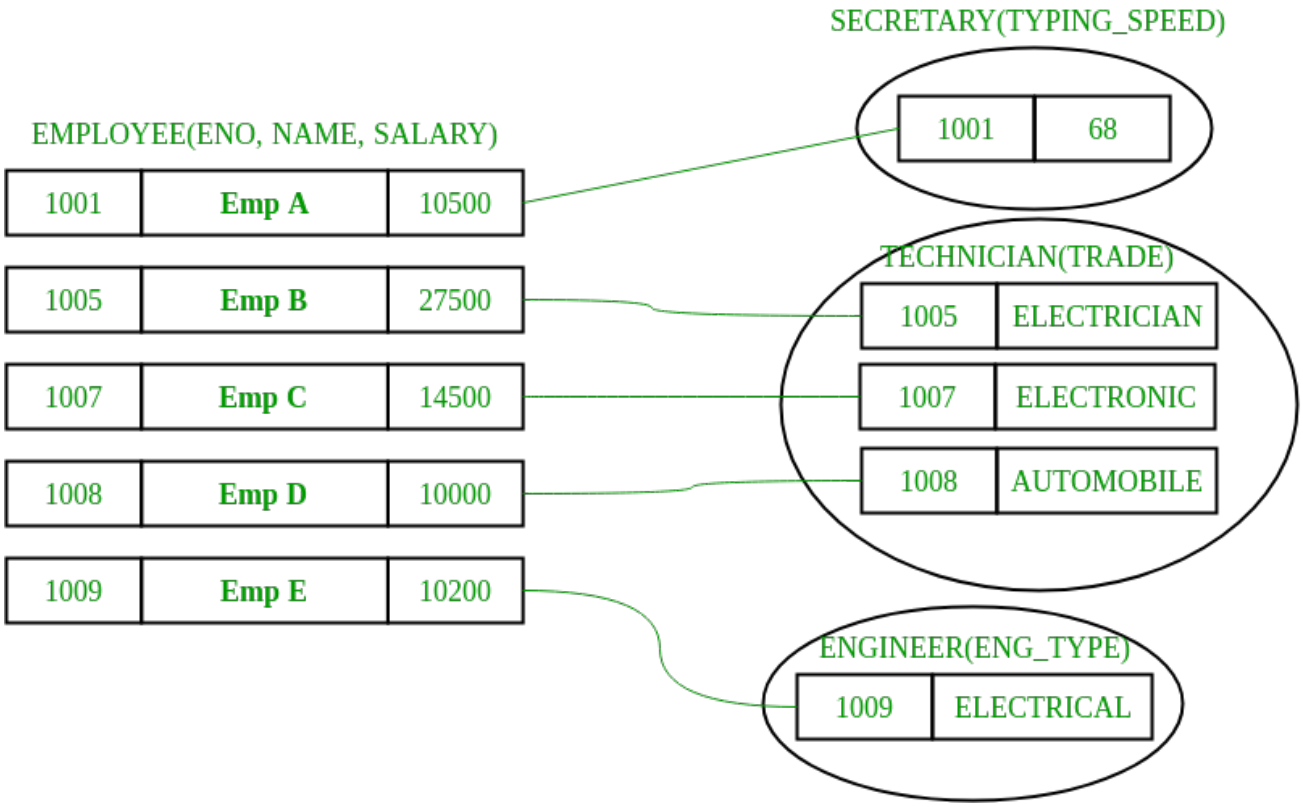
Today the complexity of the data is increasing so it becomes more and more difficult to use the traditional ER model for database modeling. To reduce this complexity of modeling we have to make improvements or enhancements to the existing ER model to make it able to handle the complex application in a better way.

Enhanced entity-relationship diagrams are advanced database diagrams very similar to regular ER diagrams which represent the requirements and complexities of complex databases.

It is a diagrammatic technique for displaying the Sub Class and Super Class; Specialization and Generalization; Union or Category; Aggregation etc.

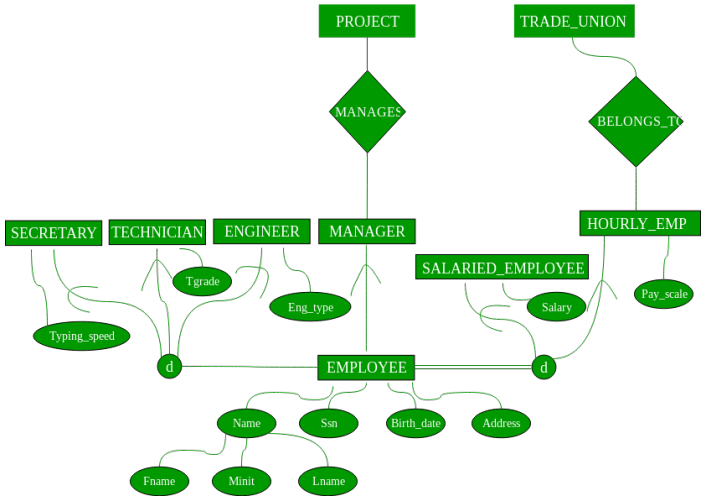
Generalization and Specialization: These are very common relationships found in real entities. However, this kind of relationship was added later as an enhanced extension to the classical ER model. **Specialized** classes are often called **subclass** while a **generalized class** is called a superclass, probably inspired by object-oriented programming. A sub-class is best understood by **"IS-A analysis"**. The following statements hopefully make some sense to your mind "Technician IS-A Employee", and "Laptop IS-A Computer".
An entity is a specialized type/class of another entity. For example, a Technician is a special Employee in a university system Faculty is a special class of Employees. We call this phenomenon generalization/specialization. In the example here Employee is a generalized entity class while the Technician and Faculty are specialized classes of Employee.

Example:
This example instance of **"sub-class"** relationships. Here we have four sets of employees: Secretary, Technician, and Engineer. The employee is a super-class of the rest three sets of individual sub-class is a subset of Employee set.



- An entity belonging to a sub-class is related to some super-class entity. For instance emp, no 1001 is a secretary, and his typing speed is 68. Emp no 1009 is an engineer (sub-class) and her trade is "Electrical", so forth.
- Sub-class entity "inherits" all attributes of super-class; for example, employee 1001 will have attributes eno, name, salary, and typing speed.

1.6 ENHANCED ER MODEL OF ABOVE EXAMPLE



Constraints – There are two types of constraints on the "Sub-class" relationship.

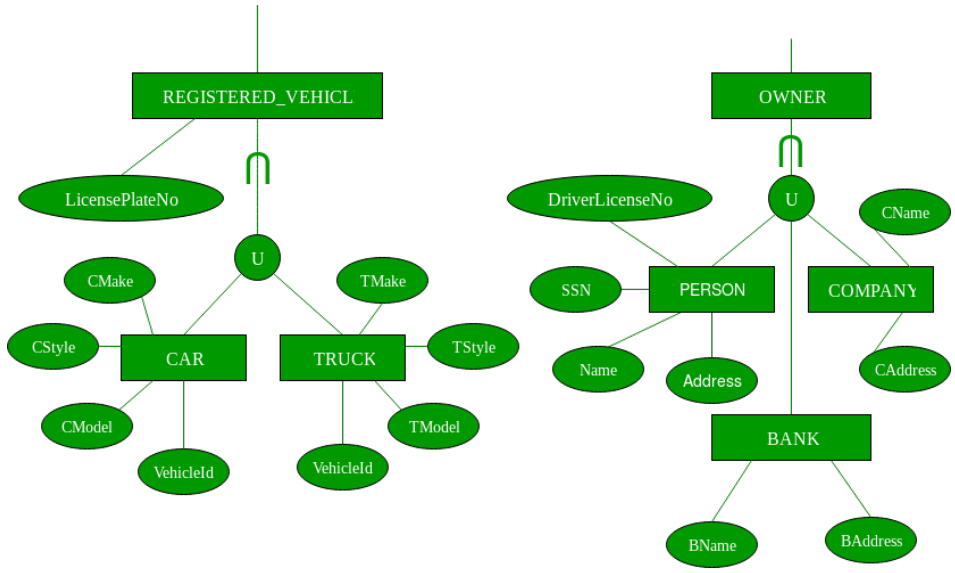
1. **Total or Partial –** A sub-classing relationship is total if every super-class entity is to be associated with some sub-class entity, otherwise partial. Sub-class "job type based employee category" is partial sub-classing – not necessary every employee is one of (secretary, engineer, and technician), i.e. union of these three types is a proper subset of all employees. Whereas other sub-classing "Salaried Employee AND Hourly Employee" is total; the union of entities from sub-classes is equal to the total employee set, i.e. every employee necessarily has to be one of them.
2. **Overlapped or Disjoint –** If an entity from a super-set can be related (can occur) in multiple sub-class sets, then it is overlapped sub-classing, otherwise disjoint. Both the examples: job-type based and salaries/hourly employee sub-classing are disjoint.

Note – *These constraints are independent of each other: can be "overlapped and total or partial" or "disjoint and total or partial". Also, sub-classing has transitive properties.*

Multiple Inheritance (sub-class of multiple superclasses**) –** An entity can be a sub-class of multiple entity types; such entities are sub-class of multiple entities and have multiple super-classes; Teaching Assistant can subclass of Employee and Student both. A faculty in a university system can be a subclass of Employee and Alumnus. In multiple inheritances, attributes of sub-class are the union of attributes of all super-classes.

Union –

- Set of Library Members is **UNION** of Faculty, Student, and Staff. A union relationship indicates either type; for example, a library member is either Faculty or Staff or Student.
- Below are two examples that show how **UNION** can be depicted in ERD – Vehicle Owner is UNION of PERSON and Company, and RTO Registered Vehicle is UNION of Car and Truck.



You might see some confusion in Sub-class and UNION; consider an example in above figure Vehicle is super-class of CAR and Truck; this is very much the correct example of the subclass as well but here use it differently we are saying RTO Registered vehicle is UNION of Car and Vehicle, they do not inherit any attribute of Vehicle, attributes of car and truck are altogether

independent set, where is in sub-classing situation car and truck would be inheriting the attribute of vehicle class.

1.7 ENHANCED ENTITY-RELATIONSHIP (EER) MODEL

It is an extension of the original Entity-Relationship (ER) model that includes additional concepts and features to support more complex data modeling requirements. The EER model includes all the elements of the ER model and adds new constructs, such as subtypes and supertypes, generalization and specialization, and inheritance.

Here are some of the key features of the EER model:

- **Subtypes and Supertypes:** The EER model allows for the creation of subtypes and supertypes. A supertype is a generalization of one or more subtypes, while a subtype is a specialization of a supertype. For example, a vehicle could be a supertype, while car, truck, and motorcycle could be subtypes.
- **Generalization and Specialization:** Generalization is the process of identifying common attributes and relationships between entities and creating a supertype based on these common features. Specialization is the process of identifying unique attributes and relationships between entities and creating subtypes based on these unique features.
- **Inheritance:** Inheritance is a mechanism that allows subtypes to inherit attributes and relationships from their supertype. This means that any attribute or relationship defined for a supertype is automatically inherited by all its subtypes.
- **Constraints:** The EER model allows for the specification of constraints that must be satisfied by entities and relationships. Examples of constraints include cardinality constraints, which specify the number of relationships that can exist between entities, and participation constraints, which specify whether an entity is required to participate in a relationship.
- Overall, the EER model provides a powerful and flexible way to model complex data relationships, making it a popular choice for database design. An Enhanced Entity-Relationship (EER) model is an extension of the traditional Entity-Relationship (ER) model that includes additional features to represent complex relationships between entities more accurately. Some of the main features of the EER model are:
- **Subclasses and Superclasses:** EER model allows for the creation of a hierarchical structure of entities where a superclass can have one or more subclasses. Each subclass inherits attributes and relationships from its superclass, and it can also have its unique attributes and relationships.
- **Specialization and Generalization:** EER model uses the concepts of specialization and generalization to create a hierarchy of entities. Specialization is the process of defining subclasses from a superclass, while generalization is the process of defining a superclass from two or more subclasses.
- **Attribute Inheritance:** EER model allows attributes to be inherited from a superclass to its subclasses. This means that attributes defined in the superclass are automatically inherited by all its subclasses.
- **Union Types: E**ER model allows for the creation of a union type, which is a combination of two or more entity types. The union type can have attributes and relationships that are common to all the entity types that make up the union.

- **Aggregation:** EER model allows for the creation of an aggregate entity that represents a group of entities as a single entity. The aggregate entity has its unique attributes and relationships.
- **Multi-valued Attributes:** EER model allows an attribute to have multiple values for a single entity instance. For example, an entity representing a person may have multiple phone numbers.
- **Relationships with Attributes:** EER model allows relationships between entities to have attributes. These attributes can describe the nature of the relationship or provide additional information about the relationship.