**INTRODUCTION:**

* A Database Management System (DBMS) is a software system that is designed to manage and organize data in a structured manner.
* It allows users to create, modify, and query a database, as well as manage the security and access controls for that database.

**Key features of DBMS :**

**Data modeling**:

It provides tools for creating and modifying data models, which define the structure and relationships of the data in a database.

**Data storage and retrieval**:

It is responsible for storing and retrieving data from the database, and can provide various methods for searching and querying the data.

**Concurrency control**:

It provides mechanisms for controlling concurrent access to the database, to ensure that multiple users can access the data without conflicting with each other.

**Data integrity and security**:

It provides tools for enforcing data integrity and security constraints, such as constraints on the values of data and access controls that restrict who can access the data.

**Backup and recovery**: It provides mechanisms for backing up and recovering the data in the event of a system failure.

Types DBMS:

* Relational Database Management System (RDBMS).
* Non-Relational Database Management System (NoSQL or Non-SQL).

**RDBMS**:

* Data is organized in the form of tables and each table has a set of rows and columns.
* The data is related to each other through primary and foreign keys.

**NoSQL**:

* Data is organized in the form of key-value pairs, document, graph, or column-based.
* These are designed to handle large-scale, high-performance scenarios.

Database is a collection of interrelated data which helps in the efficient retrieval, insertion, and deletion of data from the database and organizes the data in the form of tables, views, schemas, reports, etc.

**Types of Data Languages**

* Data Definition Language (**DDL**)
* Data Manipulation Language(**DML**)
* Data Control Language(**DCL**)
* Transactional Control Language(**TCL**)

**DDL**

* It is the short name for Data Definition Language, which deals with database schemas and descriptions, of how the data should reside in the database.
* CREATE: to create a database and its objects like (table, index, views, store procedure, function, and triggers).
* ALTER: alters the structure of the existing database.
* DROP: delete objects from the database.
* TRUNCATE: remove all records from a table, including all spaces allocated for the records are removed
* COMMENT: add comments to the data dictionary
* RENAME: rename an object

**DML**

* It is the short name for Data Manipulation Language which deals with data manipulation and includes most common SQL statements such SELECT, INSERT, UPDATE, DELETE, etc.
* It is used to store, modify, retrieve, delete and update data in a database.
* SELECT: retrieve data from a database
* INSERT: insert data into a table
* UPDATE: updates existing data within a table
* DELETE: Delete all records from a database table
* MERGE: UPSERT operation (insert or update)
* CALL: call a PL/SQL or Java subprogram
* EXPLAIN PLAN: interpretation of the data access path
* LOCK TABLE: concurrency Control

**DCL**

* **It**is short for Data Control Language which acts as an access specifier to the database.(basically to grant and revoke permissions to users in the database)
* **GRANT:** grant permissions to the user for running DML(SELECT, INSERT, DELETE,…) commands on the table
* **REVOKE**: revoke permissions to the user for running DML(SELECT, INSERT, DELETE,…) command on the specified table.

**TCL**

* It is short for Transactional Control Language which acts as an manager for all types of transactional data and all transactions.
* Some of the command of TCL are:
* **Roll Back**: Used to cancel  or Undo changes made in the database
* **Commit**: It is used to apply or save changes in the database
* **Save Point**: It is used to save the data on the temporary basis in the database

**Database Management System:**The software which is used to manage databases is called Database Management System (DBMS). For Example, MySQL, Oracle, etc. are popular commercial DBMS used in different applications. DBMS allows users the following tasks:

* **Data Definition:** It helps in the creation, modification, and removal of definitions that define the organization of data in the database.
* **Data Updation:** It helps in the insertion, modification, and deletion of the actual data in the database.
* **Data Retrieval:** It helps in the retrieval of data from the database which can be used by applications for various purposes.
* **User Administration:** It helps in registering and monitoring users, enforcing data security, monitoring performance, maintaining data integrity, dealing with concurrency control, and recovering information corrupted by unexpected failure.

**Some of the key features of a DBMS include:**

1. Data Definition: A DBMS allows users to define the structure of the database, including the tables, fields, and relationships between tables.
2. Data Manipulation: A DBMS allows users to insert, update, and delete data in the database, as well as retrieve data using queries.
3. Data Security: A DBMS provides security features to prevent unauthorized access to the database and to protect the data from theft, loss, or corruption.
4. Data Integrity: A DBMS provides mechanisms to maintain the accuracy and consistency of the data in the database, including enforcing constraints, such as unique keys, foreign keys, and check constraints.
5. Data Backup and Recovery: A DBMS provides mechanisms to back up the data in the database and to recover data in case of data loss or corruption.

**There are several types of DBMS, including:**

1. Relational DBMS (RDBMS): An RDBMS stores data in tables with rows and columns, and uses SQL (Structured Query Language) to manipulate the data.
2. Object-Oriented DBMS (OODBMS): An OODBMS stores data as objects, which can be manipulated using object-oriented programming languages.
3. NoSQL DBMS: A NoSQL DBMS stores data in non-relational data structures, such as key-value pairs, document-based models, or graph models.

**DBMS has many advantages, including:**

1. Improved data sharing: A DBMS allows multiple users to access and manipulate data in the database simultaneously.
2. Reduced data redundancy: A DBMS eliminates data redundancy by storing data in a centralized database.
3. Improved data consistency: A DBMS enforces constraints to ensure data consistency and accuracy.
4. Improved data security: A DBMS provides security features to protect the data from unauthorized access and to prevent data loss or corruption.
5. Improved data integrity: A DBMS enforces data integrity constraints to ensure that the data in the database is accurate and consistent.

# Application of DBMS

There are different fields where a database management system is utilized. Following are a few applications which utilize the information base administration framework –

1. **Railway Reservation System –**  
   In the rail route reservation framework, the information base is needed to store the record or information of ticket appointments, status about train’s appearance, and flight. Additionally, if trains get late, individuals become acquainted with it through the information base update.
2. **Library Management System –**  
   There are lots of books in the library so; it is difficult to store the record of the relative multitude of books in a register or duplicate. Along these lines, the data set administration framework (DBMS) is utilized to keep up all the data identified with the name of the book, issue date, accessibility of the book, and its writer.
3. **Banking –**  
   Database the executive’s framework is utilized to store the exchange data of the client in the information base.
4. **Education Sector –**  
   Presently, assessments are led online by numerous schools and colleges. They deal with all assessment information through the data set administration framework (DBMS). In spite of that understudy’s enlistments subtleties, grades, courses, expense, participation, results, and so forth all the data is put away in the information base.
5. **Credit card exchanges –**  
   The database Management framework is utilized for buying on charge cards and age of month to month proclamations.
6. **Social Media Sites –**  
   We all utilization of online media sites to associate with companions and to impart our perspectives to the world. Every day, many people group pursue these online media accounts like Pinterest, Facebook, Twitter, and Google in addition to. By the utilization of the data set administration framework, all the data of clients are put away in the information base and, we become ready to interface with others.
7. **Broadcast communications –**  
   Without DBMS any media transmission organization can’t think. The Database the executive’s framework is fundamental for these organizations to store the call subtleties and month to month postpaid bills in the information base.
8. **Account –**  
   The information base administration framework is utilized for putting away data about deals, holding and acquisition of monetary instruments, for example, stocks and bonds in a data set.
9. **Online Shopping –**   
   These days, web-based shopping has become a major pattern. Nobody needs to visit the shop and burn through their time. Everybody needs to shop through web based shopping sites, (for example, Amazon, Flipkart, Snapdeal) from home. So all the items are sold and added uniquely with the assistance of the information base administration framework (DBMS). Receipt charges, installments, buy data these are finished with the assistance of DBMS.
10. **Human Resource Management –**   
    Big firms or organizations have numerous specialists or representatives working under them. They store data about worker’s compensation, assessment, and work with the assistance of an information base administration framework (DBMS).
11. **Manufacturing –**   
    Manufacturing organizations make various kinds of items and deal them consistently. To keep the data about their items like bills, acquisition of the item, amount, inventory network the executives, information base administration framework (DBMS) is utilized.
12. **Airline Reservation System –**   
    This framework is equivalent to the railroad reservation framework. This framework additionally utilizes an information base administration framework to store the records of flight takeoff, appearance, and defer status.
13. **Healthcare:**DBMS is used in healthcare to manage patient data, medical records, and billing information.
14. **Data retrieval:**DBMS provides a way to retrieve data quickly and easily using search queries.
15. **Data manipulation:**DBMS provides tools to manipulate data, such as sorting, filtering, and aggregating data.
16. **Security:**DBMS provides security features to ensure that only authorized users have access to the data.
17. **Data backup and recovery:**DBMS provides tools to back up data and recover it in case of system failures or data loss.
18. **Multi-user access:**DBMS allows multiple users to access and modify data simultaneously.
19. **Reporting and analysis:** DBMS provides tools to generate reports and analyze data to gain insights and make informed decisions.

# DBMS Architecture 1-level, 2-Level, 3-Level

A Database store a lot of critical information to access data quickly and securely. Hence it is important to select the correct architecture for efficient data management. DBMS Architecture helps users to get their requests done while connecting to the database. We choose database architecture depending on several factors like the size of the database, number of users, and relationships between the users. There are two types of database models that we generally use, are logical model and physical model. Several types of architecture are there in the database which we will deal with in the next section.

## Types of DBMS Architecture

There are several types of DBMS Architecture that we use according to the usage requirements. Types of DBMS Architecture are discussed here.

* 1-Tier Architecture
* 2-Tier Architecture
* [3-Tier Architecture](https://www.geeksforgeeks.org/introduction-of-3-tier-architecture-in-dbms-set-2/)

### ****1-Tier Architecture****

In 1-Tier Architecture the database is directly available to the user, the user can directly sit on the DBMS and use it that is, the client, server, and Database are all present on the same machine. For Example: to learn SQL we set up an SQL server and the database on the local system. This enables us to directly interact with the relational database and execute operations. The industry won’t use this architecture they logically go for 2-Tier and 3-Tier Architecture.

**Advantages of 1-Tier Architecture**

Below mentioned are the advantages of 1-Tier Architecture.

* **Simple Architecture:** 1-Tier Architecture is the most simple architecture to set up, as only a single machine is required to maintain it.
* **Cost-Effective:** No additional hardware is required for implementing 1-Tier Architecture, which makes it cost-effective.
* **Easy to Implement:**1-Tier Architecture can be easily deployed, and hence it is mostly used in small projects.

### ****2-Tier Architecture****

The 2-tier architecture is similar to a basic[client-server model](https://www.geeksforgeeks.org/client-server-model/). The application at the client end directly communicates with the database on the server side. APIs like ODBC and JDBC are used for this interaction. The server side is responsible for providing query processing and transaction management functionalities. On the client side, the user interfaces and application programs are run. The application on the client side establishes a connection with the server side in order to communicate with the DBMS.   
An advantage of this type is that maintenance and understanding are easier, and compatible with existing systems. However, this model gives poor performance when there are a large number of users.



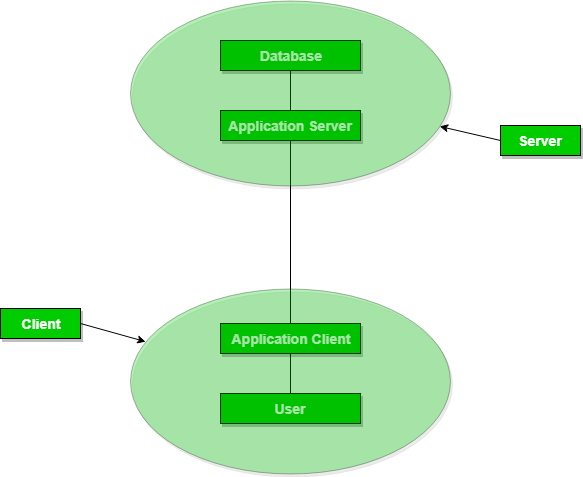
*DBMS 2-Tier Architecture*

**Advantages of 2-Tier Architecture**

* **Easy to Access:** 2-Tier Architecture makes easy access to the database, which makes fast retrieval.
* **Scalable:** We can scale the database easily, by adding clients or by upgrading hardware.
* **Low Cost:** 2-Tier Architecture is cheaper than 3-Tier Architecture and [Multi-Tier Architecture](https://www.geeksforgeeks.org/multi-tier-architecture-of-data-warehouse/).
* **Easy Deployment:** 2-Tier Architecture is easy to deploy than 3-Tier Architecture.
* **Simple:** 2-Tier Architecture is easily understandable as well as simple because of only two components.

### ****3-Tier Architecture****

In [3-Tier Architecture](https://www.geeksforgeeks.org/introduction-of-3-tier-architecture-in-dbms-set-2/), there is another layer between the client and the server. The client does not directly communicate with the server. Instead, it interacts with an application server which further communicates with the database system and then the query processing and transaction management takes place. This intermediate layer acts as a medium for the exchange of partially processed data between the server and the client. This type of architecture is used in the case of large web applications.



*DBMS 3-Tier Architecture*

**Advantages of 3-Tier Architecture**

* **Enhanced scalability:** Scalability is enhanced due to distributed deployment of application servers. Now, individual connections need not be made between the client and server.
* **Data Integrity:** 3-Tier Architecture maintains Data Integrity. Since there is a middle layer between the client and the server, data corruption can be avoided/removed.
* **Security:**3-Tier Architecture Improves Security. This type of model prevents direct interaction of the client with the server thereby reducing access to unauthorized data.

**Disadvantages of 3-Tier Architecture**

* **More Complex:**3-Tier Architecture is more complex in comparison to 2-Tier Architecture. Communication Points are also doubled in 3-Tier Architecture.
* **Difficult to Interact:** It becomes difficult for this sort of interaction to take place due to the presence of middle layers.

# Introduction of ER Model

The Entity Relational Model is a model for identifying entities to be represented in the [database](https://www.geeksforgeeks.org/what-is-database/) and representation of how those entities are related. The ER data model specifies enterprise schema that represents the overall logical structure of a database graphically.

The Entity Relationship Diagram explains the relationship among the entities present in the database. ER models are used to model real-world objects like a person, a car, or a company and the relation between these real-world objects. In short, ER Diagram is the structural format of the database.

## Why Use ER Diagrams In DBMS?

* ER diagrams are used to represent the E-R model in a database, which makes them easy to be converted into relations (tables).
* ER diagrams provide the purpose of real-world modeling of objects which makes them intently useful.
* ER diagrams require no technical knowledge and no hardware support.
* These diagrams are very easy to understand and easy to create even for a naive user.
* It gives a standard solution for visualizing the data logically.

## Symbols Used in ER Model

ER Model is used to model the logical view of the system from a data perspective which consists of these symbols:

* **Rectangles:**Rectangles represent Entities in ER Model.
* **Ellipses:**Ellipses represent Attributes in ER Model.
* **Diamond:**Diamonds represent Relationships among Entities.
* **Lines:**Lines represent attributes to entities and entity sets with other relationship types.
* **Double Ellipse:**Double Ellipses represent [Multi-Valued Attributes](https://iotap.geeksforgeeks.org/problems/what-is-the-difference-between-single-valued-and-multi-valued-attributes).
* **Double Rectangle:**Double Rectangle represents a Weak Entity.

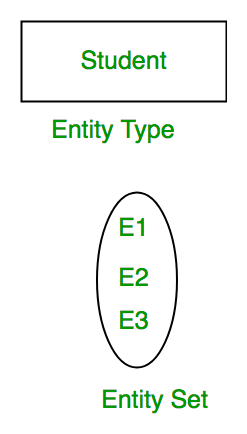
## Components of ER Diagram

ER Model consists of Entities, Attributes, and Relationships among Entities in a Database System.

### Entity

An Entity may be an object with a physical existence – a particular person, car, house, or employee – or it may be an object with a conceptual existence – a company, a job, or a university course.

**Entity Set:** An Entity is an object of Entity Type and a set of all entities is called an entity set. For Example, E1 is an entity having Entity Type Student and the set of all students is called Entity Set. In ER diagram, Entity Type is represented as:



*Entity Set*

#### 1. Strong Entity

A [Strong Entity](https://www.geeksforgeeks.org/difference-between-strong-and-weak-entity/) is a type of entity that has a key Attribute. Strong Entity does not depend on other Entity in the Schema. It has a primary key, that helps in identifying it uniquely, and it is represented by a rectangle. These are called Strong Entity Types.

#### 2. Weak Entity

An Entity type has a key attribute that uniquely identifies each entity in the entity set. But some entity type exists for which key attributes can’t be defined. These are called [Weak Entity types](https://www.geeksforgeeks.org/weak-entity-set-in-er-diagrams/).

**For Example,** A company may store the information of dependents (Parents, Children, Spouse) of an Employee. But the dependents don’t have existed without the employee. So Dependent will be a **Weak Entity Type** and Employee will be Identifying Entity type for Dependent, which means it is **Strong Entity Type**.

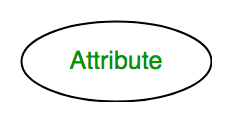
A weak entity type is represented by a Double Rectangle. The participation of weak entity types is always total. The relationship between the weak entity type and its identifying strong entity type is called identifying relationship and it is represented by a double diamond.



*Strong Entity and Weak Entity*

### ****Attributes****

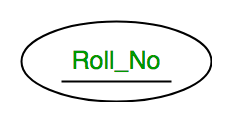
[Attributes](https://www.geeksforgeeks.org/types-of-attributes-in-er-model/) are the properties that define the entity type. For example, Roll\_No, Name, DOB, Age, Address, and Mobile\_No are the attributes that define entity type Student. In ER diagram, the attribute is represented by an oval.



*Attribute*

#### **1. Key Attribute**

The attribute which **uniquely identifies each entity** in the entity set is called the key attribute. For example, Roll\_No will be unique for each student. In ER diagram, the key attribute is represented by an oval with underlying lines.



*Key Attribute*

#### **2. Composite Attribute**

An attribute **composed of many other attributes** is called a composite attribute. For example, the Address attribute of the student Entity type consists of Street, City, State, and Country. In ER diagram, the composite attribute is represented by an oval comprising of ovals.



*Composite Attribute*

#### **3. Multivalued Attribute**

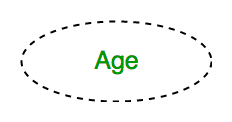
An attribute consisting of more than one value for a given entity. For example, Phone\_No (can be more than one for a given student). In ER diagram, a multivalued attribute is represented by a double oval.



*Multivalued Attribute*

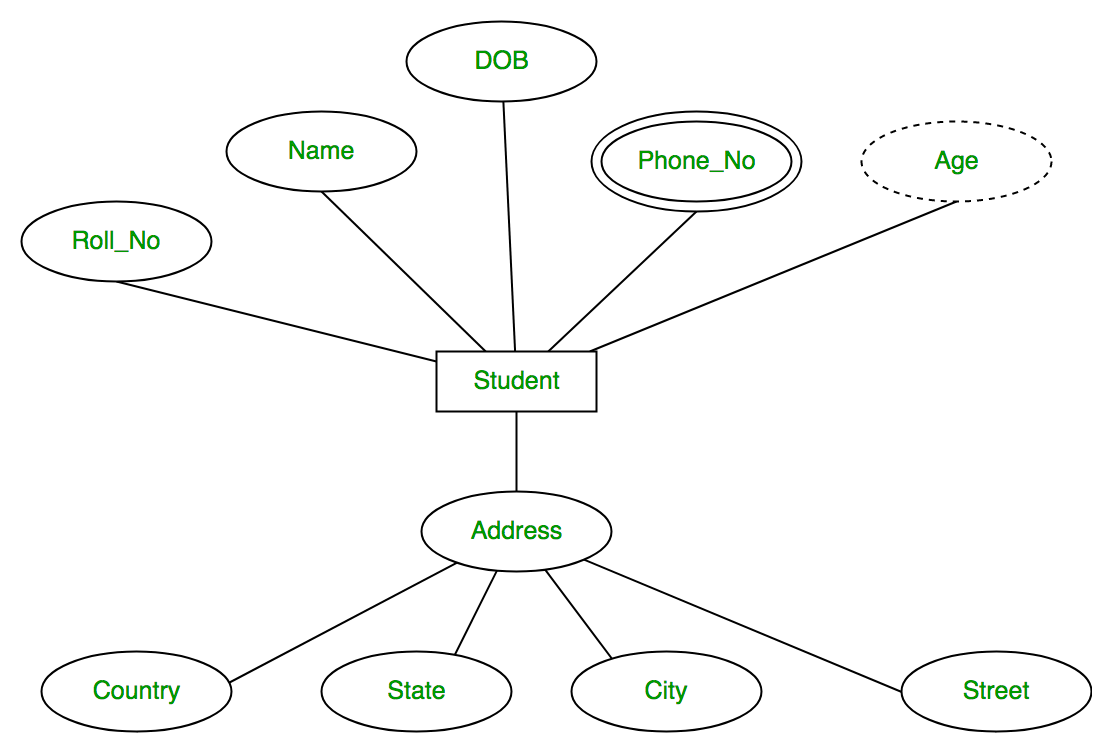
#### **4. Derived Attribute**

An attribute that can be derived from other attributes of the entity type is known as a derived attribute. e.g.; Age (can be derived from DOB). In ER diagram, the derived attribute is represented by a dashed oval.



*Derived Attribute*

The Complete Entity Type Student with its Attributes can be represented as:



*Entity and Attributes*

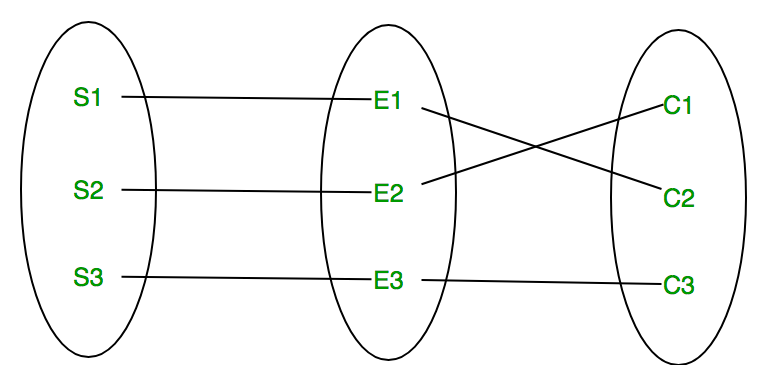
### ****Relationship Type and Relationship Set****

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.

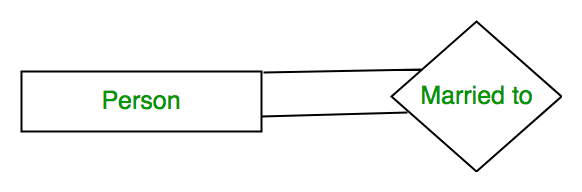


*Relationship Set*

#### **Degree of a Relationship Set**

The number of different entity sets participating in a relationship set is called the [degree of a relationship set.](https://www.geeksforgeeks.org/degree-of-relations-in-dbms/)

**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*

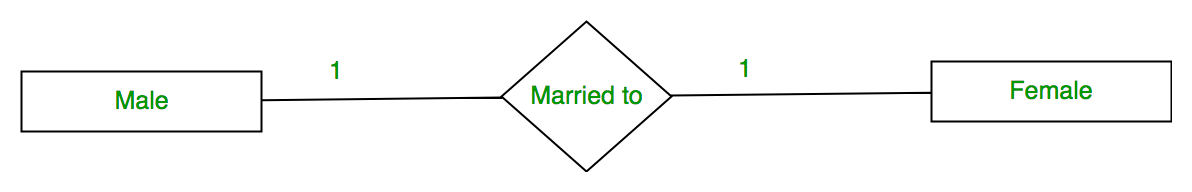
**3. n-ary Relationship:**When there are n entities set participating in a relation, the relationship is called an n-ary relationship.

#### **Cardinality**

The number of times an entity of an entity set participates in a relationship set is known as [cardinality](https://www.geeksforgeeks.org/cardinality-in-dbms/). Cardinality can be of different types:

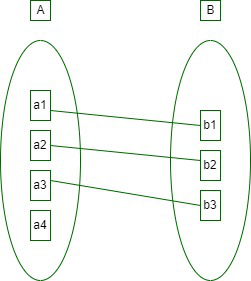
**1. One-to-One:** When each entity in each entity set can take part only once in the relationship, the cardinality is one-to-one. Let us assume that a male can marry one female and a female can marry one male. So the relationship will be one-to-one.

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



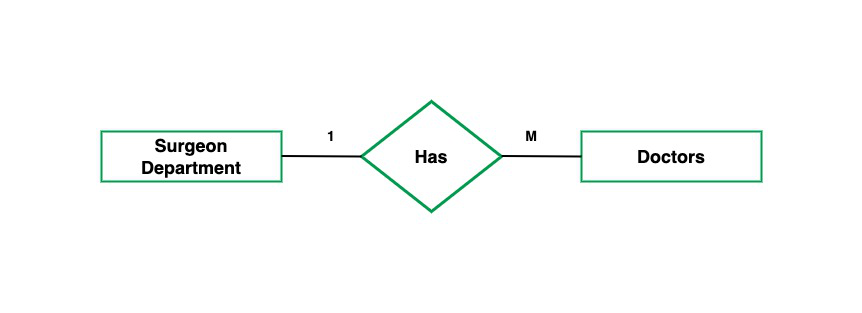
*One-to-One Cardinality*

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.



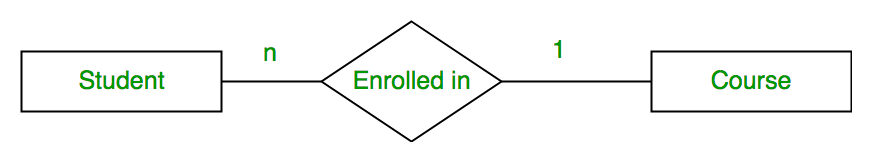
*One to Many*

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

*Set Representation of One-to-Many*

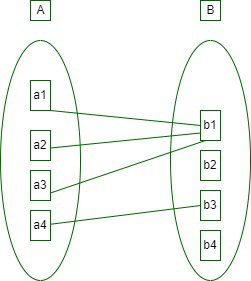
**3. Many-to-One:** When entities in one entity set can take part only once in the relationship set and entities in other entity sets can take part more than once in the relationship set, cardinality is many to one. Let us assume that a student can take only one course but one course can be taken by 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:

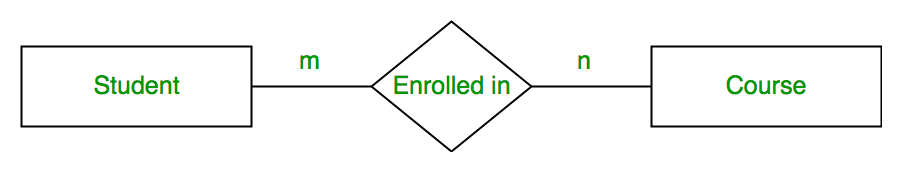


*Set Representation of Many-to-One*

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

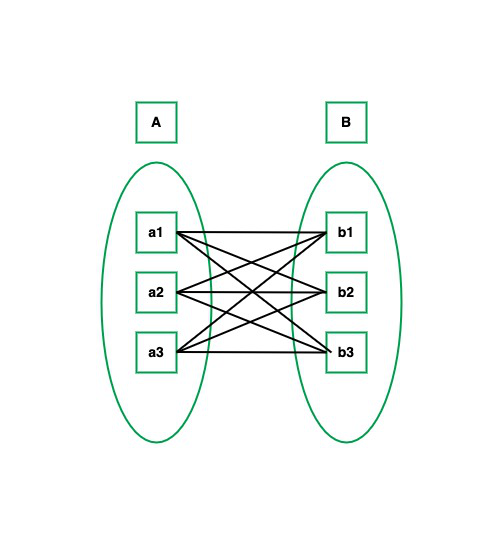
**4. Many-to-Many:**When entities in all entity sets can take part more than once in the relationship cardinality is many to many. Let us assume that a student can take more than one course and one course can be taken by many students. So the relationship will be many to many.

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.

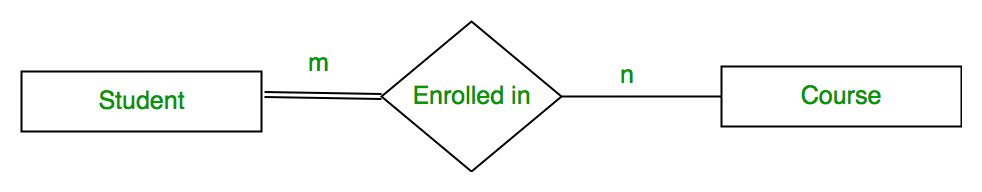
### ****Participation Constraint****

[Participation Constraint](https://www.geeksforgeeks.org/structural-constraints-of-relationships-in-er-model/) is applied to the entity participating in the relationship set.

**1. Total Participation –** Each entity in the entity setmust 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.

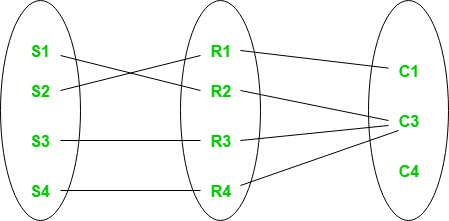
**2. Partial Participation –** The entity in the entity set may or may NOT participate in the relationship. If some courses are not enrolled by any of the students, the participation in the course will be partial.

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,

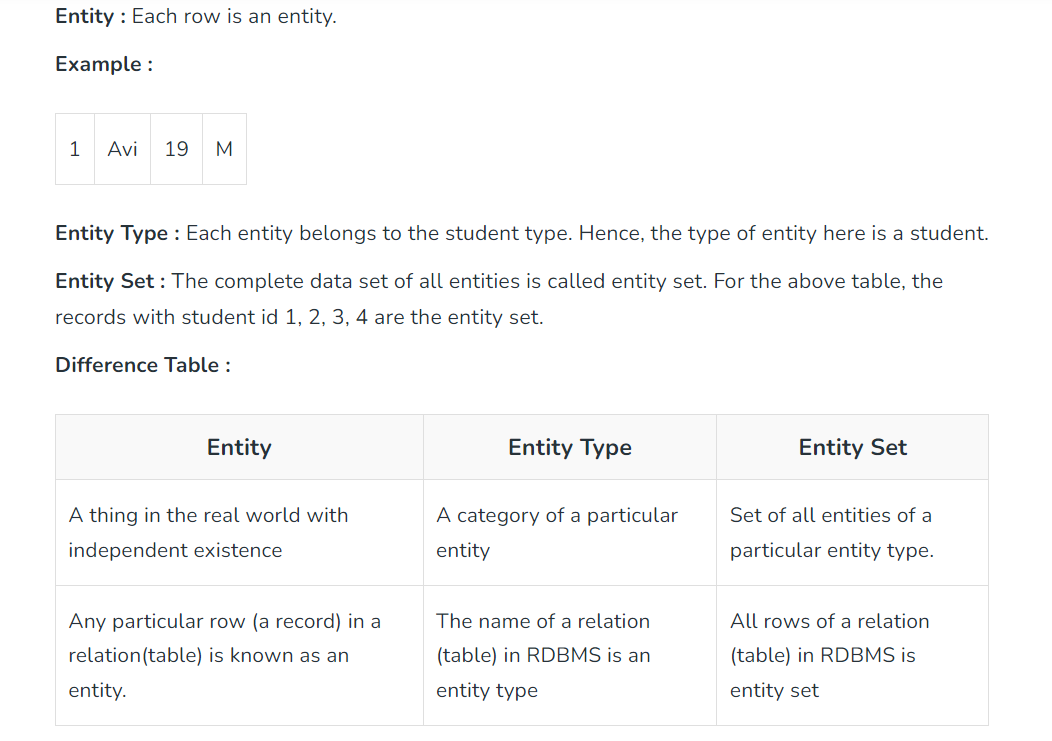


*Set representation of Total Participation and Partial Participation*

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.

## How to Draw ER Diagram?

* The very first step is Identifying all the Entities, and place them in a Rectangle, and labeling them accordingly.
* The next step is to identify the relationship between them and pace them accordingly using the Diamond, and make sure that, Relationships are not connected to each other.
* Attach attributes to the entities properly.
* Remove redundant entities and relationships.
* Add proper colors to highlight the data present in the database.



A Relational Database Model consists of relations to connect them by key fields. A relation has some attributes. The relation is represented in rows and columns. Each column of the relation is called an attribute. Each row in the relation is called a tuple. Each relation can have one unique column i.e. primary key. Each relation can have n-columns and n-tuple. Each relation is preceded by the name of that relation. The fields of the relations are separated by commas and placed within the parentheses of the relation. The relational model represents data in the form of relations or tables.

**Example**: Relational Model can be represented as shown below

STUDENT (StudNo, Sname, Special)  
ENROLLMENT (StudNo, Subcode, marks)  
SUBJECT (Subcode, Subname, Maxmarks, Faccode)  
FACULTY (Faccode, Fname, Dept)

## **Terminologies**

[**Relational Schema:**](https://www.geeksforgeeks.org/relation-schema-in-dbms/) Schema represents the structure of a relation.

**Example:** Relational Schema of STUDENT relation can be represented as STUDENT(STUD\_NO, STUD\_NAME,  STUD\_PHONE, STUD\_STATE, STUD\_COUNTRY, STUD\_AGE).

[**Relational Instance:**](https://www.geeksforgeeks.org/difference-between-schema-and-instance-in-dbms/) The set of values present in a relationship at a particular instance of time is known as a relational instance as shown in Table 1 and Table 2.

**Attribute:** Each relation is defined in terms of some properties, each of which is known as an attribute.

**Example:** STUD\_NO, STUD\_NAME, etc. are attributes of relation STUDENT.

**The domain of an attribute:**The possible values an attribute can take in a relation is called its domain.

**Example:** domain of STUD\_AGE can be from 18 to 40.

**Tuple:** Each row of a relation is known as a tuple.

**Example:** STUDENT relation given below has 4 tuples.

**NULL values:** Values of some attribute for some tuples may be unknown, missing, or undefined which are represented by NULL. Two NULL values in a relationship are considered different from each other. Table 1 and Table 2 represent the relational model having two relations STUDENT and STUDENT\_COURSE.

**STUDENT TABLE**

| **STUD\_NO** | **STUD\_NAME** | **STUD\_PHONE** | **STUD\_STATE** | **STUD\_COUNTRY** | **STUD\_AGE** |
| --- | --- | --- | --- | --- | --- |
| 1 | RAM | 9716271721 | Haryana | India | 20 |
| 2 | RAM | 9898291281 | Punjab | India | 19 |
| 3 | SUJIT | 7898291981 | Rajasthan | India | 18 |
| 4 | SURESH | 9985286317 | Punjab | India | 21 |

**STUDENT COURSE TABLE**

| **STUD\_NO** | **COURSE\_NO** | **COURSE\_NAME** |
| --- | --- | --- |
| 1 | C1 | DBMS |
| 2 | C2 | Computer Networks |
| 1 | C2 | Computer Networks |

## **RDBMS Vendors**

There are several vendors that offer Relational Database Management Systems (RDBMS). Here are some of the most popular ones:

**Oracle:**Oracle Database is one of the most widely used RDBMS products in the market. It is known for its robustness, scalability, and reliability. It is used by many large enterprises and is particularly well-suited for data warehousing and transaction processing.

**Microsoft:**Microsoft SQL Server is a popular RDBMS used in Windows environments. It offers a range of features, including data mining, business intelligence, and reporting services.

**IBM:** IBM DB2 is a popular RDBMS used in enterprise environments. It offers high availability, disaster recovery, and scalability features.

**MySQL:** MySQL is an open-source RDBMS used by many small to medium-sized businesses. It is known for its ease of use, flexibility, and low cost.

**PostgreSQL:** PostgreSQL is another popular open-source RDBMS. It is known for its scalability, reliability, and support for complex transactions.

**SAP:** SAP HANA is an in-memory RDBMS that is designed for high-performance analytics and data processing. It is often used in enterprise environments for real-time reporting and business intelligence.

## **Relational Algebra**

It is a procedural Language. It consists of a set of operators that can be performed on relations. [Relational Algebra](https://www.geeksforgeeks.org/introduction-of-relational-algebra-in-dbms/) forms the basis for many other high-level data sub-languages like [SQL](https://www.geeksforgeeks.org/sql-tutorial/) and [QBE](https://www.geeksforgeeks.org/query-by-example-qbe/).   
 Relational algebra has mainly 9 types of operators.

* [UNION](https://www.geeksforgeeks.org/union-c/)
* [INTERSECTION](https://www.geeksforgeeks.org/sql-intersect-clause/)
* [MINUS](https://www.geeksforgeeks.org/sql-minus-operator/)
* [TIMES](https://www.geeksforgeeks.org/concept-of-time-in-database/)
* [SELECTION](https://www.geeksforgeeks.org/difference-between-selection-and-projection-in-dbms/)
* [PROJECTION](https://www.geeksforgeeks.org/difference-between-selection-and-projection-in-dbms/)
* [JOIN](https://www.geeksforgeeks.org/sql-join-set-1-inner-left-right-and-full-joins/)
* [DIVISION](https://www.geeksforgeeks.org/sql-division/)
* [RENAME](https://www.geeksforgeeks.org/sql-alter-rename/)

**1. UNION (U):**A and B are two relations. It displays total values (Attributes) in both relations. It avoids duplicate values in both relations. U symbol can be used.

**Syntax:**

*A UNION B (or) A U B*

**Example:**

A = { clerk, manager, salesman}  
 B = { president, clerk, manager}  
 A UNION B = {clerk, manager, salesman, president}

**2. INTERSECTION (∩):**A and B are two relations. It displays common elements in both relations. “∩” symbol can be used.

**Syntax:**

*A INTERSECT B (or) A ∩ B*

**Example:**

A = { clerk, manager, salesman}  
B = { president, clerk, manager}  
A INTERSECT B = { clerk, manager}

**3. DIFFERENCE (─):**A and B are two relations. It displays elements in relation A not in relation B. 

**Syntax:**

*A MINUS B (OR) A ─ B*

**Example:**

A = { clerk, manager, salesman}  
B = { president, clerk, manager}  
A MINUS B = {salesman}

**4. CARTESIAN PRODUCT(X):** A and B are two relations. It has a new relation consisting of all pair wises combinations of all elements in A and B. The relation A has “m” elements and relation B has “n” elements, then the resultant relation will be “ m \* n “. 

**Syntax:**

*A TIMES B (OR) A X B*

**Example:**

A = { clerk, manager, salesman}  
B = { president, clerk, manager}  
A TIMES B = { (clerk, president),   
(clerk, clerk),(clerk, manager),   
(manager, president), (manager, clerk),  
(manager, manager),(salesman, president),   
(salesman, clerk), (salesman, manager) }

**5. SELECTION (σ):** It displays all the attributes or columns of all the rows in a relation. “σ” operator can be used to select the attributes in a relation.

The  select operation

In general SELECT operation is denoted by

(σ)(R)

(σ)(Sigma): SELECT operator

 θ: Selection condition

R: Relation or relational algebra expression.

In general the select condition is a Boolean condition (i.e. an expression using logical connective) of terms that have the form attribute OP constant 1 OP attribute2 where OP is the comparison operators <,>,=,>= etc.  
**Syntax:**

*σ condition (relation name)*

**6. PROJECTION (π):** It displays some specified columns in a relation. “π” operator can be used to select some specified columns in a relation. It selects tuples that satisfy the given predicate from a relation. It displays some specified columns by using some conditions.

**Syntax:**

***Syntax:****π(col1,col2…) Relation Name*

**Example:**

π(sno, sname, total) MARKS

**7. JOIN( ):**It combines two or more relations. Symbols can be used. It can be mainly divided into mainly 4 types. These are mainly

* [Inner Join](https://www.geeksforgeeks.org/sql-join-set-1-inner-left-right-and-full-joins/)
* [Outer Join](https://www.geeksforgeeks.org/sql-join-set-1-inner-left-right-and-full-joins/)
* [Left Outer Join](https://www.geeksforgeeks.org/sql-join-set-1-inner-left-right-and-full-joins/)
* [Right Outer Join](https://www.geeksforgeeks.org/sql-join-set-1-inner-left-right-and-full-joins/)

**8. DIVIDE (÷):** It divides the tuple from one relation to another relation

**Syntax:**

*A DIVIDE B (OR) A ÷ B*

**Example:**

A = { clerk, manager, salesman}   
B = { clerk, manager}  
A DIVIDE B = {salesman}

**9. RENAME(ρ):** It gives another name to the relation.

**Syntax:**

*ρ(OLD RELATION, NEW RELATION)*

**Example:**

ρ(STUDENT, MARKS)  
It changes the “student” relation to “Marks” relation.  
It also renames the specified column.  
It changes the old-column name to new-column name.

### Features of the relational model and Codd’s Rules :

**Tables/Relations:**The basic building block of the relational model is the table or relation, which represents a collection of related data. Each table consists of columns, also known as attributes or fields, and rows, also known as tuples or records.

**Primary Keys:**In the relational model, each row in a table must have a unique identifier, which is known as the primary key. This ensures that each row is unique and can be accessed and manipulated easily.

**Foreign Keys:**Foreign keys are used to link tables together and enforce referential integrity. They ensure that data in one table is consistent with data in another table.

**Normalization:**The process of organizing data into tables and eliminating redundancy is known as normalization. Normalization is important in the relational model because it helps to ensure that data is consistent and easy to maintain.

**Codd’s Rules:**Codd’s Rules are a set of 12 rules that define the characteristics of a true relational DBMS. These rules ensure that the DBMS is consistent, reliable, and easy to use.

Atomicity, Consistency, Isolation, Durability (ACID): The ACID properties are a set of properties that ensure that transactions are processed reliably in the relational model. Transactions are sets of operations that are executed as a single unit, ensuring that data is consistent and accurate.

## **Advantages of Relational Algebra**

Relational Algebra is a formal language used to specify queries to retrieve data from a relational database. It has several advantages that make it a popular choice for managing and manipulating data. Here are some of the advantages of Relational Algebra:

**Simplicity:** Relational Algebra provides a simple and easy-to-understand set of operators that can be used to manipulate data. It is based on a set of mathematical concepts and principles, which makes it easy to learn and use.

**Formality:**Relational Algebra is a formal language that provides a standardized and rigorous way of expressing queries. This makes it easier to write and debug queries, and also ensures that queries are correct and consistent.

**Abstraction:** Relational Algebra provides a high-level abstraction of the underlying database structure, which makes it easier to work with large and complex databases. It allows users to focus on the logical structure of the data, rather than the physical storage details.

**Portability:**Relational Algebra is independent of any specific database management system, which means that queries can be easily ported to other systems. This makes it easy to switch between different databases or vendors without having to rewrite queries.

**Efficiency:** Relational Algebra is optimized for efficiency and performance, which means that queries can be executed quickly and with minimal resources. This is particularly important for large and complex databases, where performance is critical.

**Extensibility:** Relational Algebra provides a flexible and extensible framework that can be extended with new operators and functions. This allows developers to customize and extend the language to meet their specific needs.

## **Disadvantages of Relational Algebra**

**While Relational Algebra has many advantages, it also has some limitations and disadvantages that should be considered when using it. Here are some of the disadvantages of Relational Algebra**:

**Complexity:** Although Relational Algebra is based on mathematical principles, it can be complex and difficult to understand for non-experts. The syntax and semantics of the language can be challenging, and it may require significant training and experience to use it effectively.

**Limited Expressiveness:**Relational Algebra has a limited set of operators, which can make it difficult to express certain types of queries. It may be necessary to use more advanced techniques, such as subqueries or joins, to express complex queries.

**Lack of Flexibility:**Relational Algebra is designed for use with relational databases, which means that it may not be well-suited for other types of data storage or management systems. This can limit its flexibility and applicability in certain contexts.

**Performance Limitations:**While Relational Algebra is optimized for efficiency and performance, it may not be able to handle large or complex datasets. Queries can become slow and resource-intensive when dealing with large amounts of data or complex queries.

**Limited Data Types:** Relational Algebra is designed for use with simple data types, such as integers, strings, and dates. It may not be well-suited for more complex data types, such as multimedia files or spatial data.

**Lack of Integration:**Relational Algebra is often used in conjunction with other programming languages and tools, which can create integration challenges. It may require additional programming effort to integrate Relational Algebra with other systems and tools.

Relational Algebra is a powerful and useful tool for managing and manipulating data in relational databases, it has some limitations and disadvantages that should be carefully considered when using it.

## **Codd’s Twelve Rules of Relational Database**

Codd rules were proposed by E.F. Codd which should be satisfied by the [relational model](https://www.geeksforgeeks.org/relational-model-in-dbms/). Codd’s Rules are basically used to check whether DBMS has the quality to become [Relational Database Management System (RDBMS)](https://www.geeksforgeeks.org/rdbms-full-form/). But, it is rare to find that any product has fulfilled all the rules of Codd. They generally follow the 8-9 rules of Codd. E.F. Codd has proposed 13 rules which are popularly known as Codd’s 12 rules. These rules are stated as follows:

* **Rule 0: Foundation Rule**– For any system that is advertised as, or claimed to be, a relational database management system, that system must be able to manage databases entirely through its relational capabilities.
* **Rule 1: Information Rule**– Data stored in the Relational model must be a value of some cell of a table.
* **Rule 2: Guaranteed Access Rule**– Every data element must be accessible by the table name, its primary key, and the name of the attribute whose value is to be determined.
* **Rule 3: Systematic Treatment of NULL values**– NULL value in the database must only correspond to missing, unknown, or not applicable values.
* **Rule 4: Active Online Catalog**– The structure of the database must be stored in an online catalog that can be queried by authorized users.
* **Rule 5: Comprehensive Data Sub-language Rule-** A database should be accessible by a language supported for definition, manipulation, and transaction management operation.
* **Rule 6: View Updating Rule-** Different views created for various purposes should be automatically updatable by the system.
* **Rule 7: High-level insert, update and delete rule-** Relational Model should support insert, delete, update, etc. operations at each level of relations. Also, set operations like Union, Intersection, and minus should be supported.
* **Rule 8: Physical data independence-** Any modification in the physical location of a table should not enforce modification at the application level.
* **Rule 9: Logical data independence-** Any modification in the logical or conceptual schema of a table should not enforce modification at the application level. For example, merging two tables into one should not affect the application accessing it which is difficult to achieve.
* **Rule 10: Integrity Independence-** Integrity constraints modified at the database level should not enforce modification at the application level.
* **Rule 11: Distribution Independence-** Distribution of data over various locations should not be visible to end-users.
* **Rule 12: Non-Subversion Rule- Low-level** access to data should not be able to bypass the integrity rule to change data.

Keys are one of the basic requirements of a [relational database model](https://www.geeksforgeeks.org/relational-model-in-dbms/). It is widely used to identify the [tuples(rows)](https://www.geeksforgeeks.org/tuple-in-dbms/) uniquely in the table. We also use keys to set up relations amongst various columns and tables of a relational database.

## **Different Types of Keys in the Relational Model**

1. [Candidate Key](https://www.geeksforgeeks.org/difference-between-primary-and-candidate-key/)
2. [Primary Key](https://www.geeksforgeeks.org/difference-between-primary-and-candidate-key/)
3. [Super Key](https://www.geeksforgeeks.org/difference-between-super-key-and-candidate-key/)
4. [Alternate Key](https://www.geeksforgeeks.org/sql-alternate-key/)
5. [Foreign Key](https://www.geeksforgeeks.org/foreign-key-constraint-in-sql/)
6. [Composite Key](https://www.geeksforgeeks.org/composite-key-in-sql/)

**1. Candidate Key:** The minimal set of attributes that can uniquely identify a tuple is known as a candidate key. For Example, STUD\_NO in STUDENT relation.

* It is a minimal super key.
* It is a super key with no repeated data is called a candidate key.
* The minimal set of attributes that can uniquely identify a record.
* It must contain unique values.
* It can contain NULL values.
* Every table must have at least a single candidate key.
* A table can have multiple candidate keys but only one primary key (the primary key cannot have a NULL value, so the candidate key with a NULL value can’t be the primary key).
* The value of the Candidate Key is unique and may be null for a tuple.
* There can be more than one candidate key in a relationship.

**Example:**

STUD\_NO is the candidate key for relation STUDENT.

**Table STUDENT**

| **STUD\_NO** | **SNAME** | **ADDRESS** | **PHONE** |
| --- | --- | --- | --- |
| 1 | Shyam | Delhi | 123456789 |
| 2 | Rakesh | Kolkata | 223365796 |
| 3 | Suraj | Delhi | 175468965 |

* The candidate key can be simple (having only one attribute) or composite as well.

**Example:**

{STUD\_NO, COURSE\_NO} is a composite

candidate key for relation STUDENT\_COURSE.

**Table STUDENT\_COURSE**

| **STUD\_NO** | **TEACHER\_NO** | **COURSE\_NO** |
| --- | --- | --- |
| 1 | 001 | C001 |
| 2 | 056 | C005 |

**Note:**In[SQL](https://www.geeksforgeeks.org/sql-tutorial/) Server a unique constraint that has a nullable column, **allows** the value ‘**null**‘ in that column **only once**. That’s why the STUD\_PHONE attribute is a candidate here, but can not be a ‘null’ value in the primary key attribute.

**2. Primary Key:** There can be more than one candidate key in relation out of which one can be chosen as the primary key. For Example, STUD\_NO, as well as STUD\_PHONE, are candidate keys for relation STUDENT but STUD\_NO can be chosen as the primary key (only one out of many candidate keys).

* It is a unique key.
* It can identify only one tuple (a record) at a time.
* It has no duplicate values, it has unique values.
* It cannot be NULL.
* Primary keys are not necessarily to be a single column; more than one column can also be a primary key for a table.

**Example:**

STUDENT table -> Student(STUD\_NO, SNAME,

ADDRESS, PHONE) , STUD\_NO is a primary key

**Table STUDENT**

| **STUD\_NO** | **SNAME** | **ADDRESS** | **PHONE** |
| --- | --- | --- | --- |
| 1 | Shyam | Delhi | 123456789 |
| 2 | Rakesh | Kolkata | 223365796 |
| 3 | Suraj | Delhi | 175468965 |

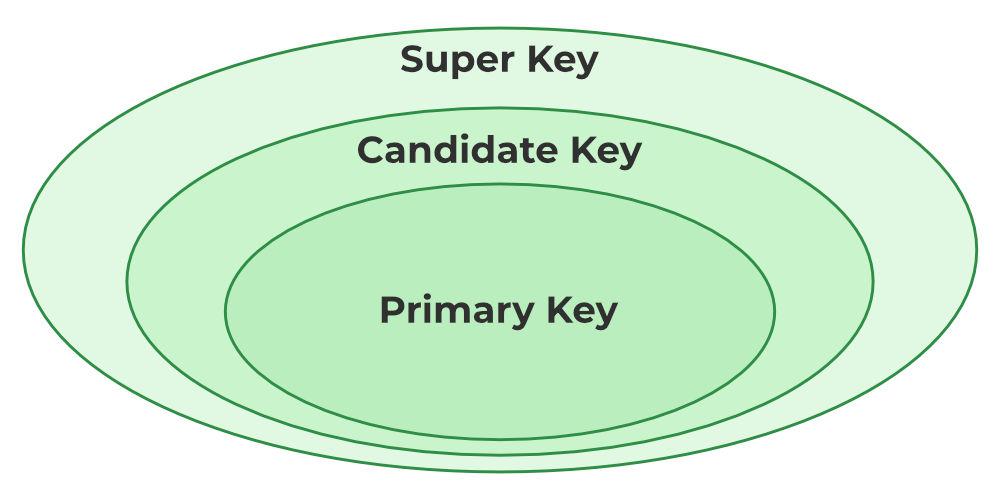
**3. Super Key:** The set of attributes that can uniquely identify a tuple is known as Super Key. For Example, STUD\_NO, (STUD\_NO, STUD\_NAME), etc. A super key is a group of single or multiple keys that identifies rows in a table. It supports NULL values.

* Adding zero or more attributes to the candidate key generates the super key.
* A candidate key is a super key but vice versa is not true.

**Example:**

Consider the table shown above.

STUD\_NO+PHONE is a super key**.**



*Relation between Primary Key, Candidate Key, and Super Key*

**4. Alternate Key:** The candidate key other than the primary key is called an alternate key.

* All the keys which are not primary keys are called alternate keys.
* It is a secondary key.
* It contains two or more fields to identify two or more records.
* These values are repeated.
* Eg:- SNAME, and ADDRESS is Alternate keys

**Example:**

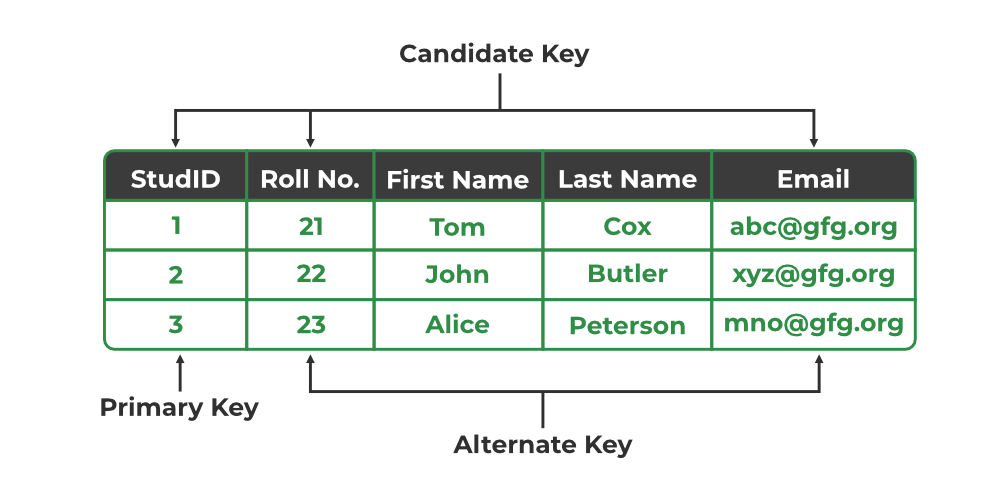
Consider the table shown above.

STUD\_NO, as well as PHONE both,

are candidate keys for relation STUDENT but

PHONE will be an alternate key

(only one out of many candidate keys).



*Primary Key, Candidate Key, and Alternate Key*

**5. Foreign Key:** If an attribute can only take the values which are present as values of some other attribute, it will be a foreign key to the attribute to which it refers. The relation which is being referenced is called referenced relation and the corresponding attribute is called referenced attribute the relation which refers to the referenced relation is called referencing relation and the corresponding attribute is called referencing attribute. The referenced attribute of the referenced relation should be the primary key to it.

* It is a key it acts as a primary key in one table and it acts as   
  secondary key in another table.
* It combines two or more relations (tables) at a time.
* They act as a cross-reference between the tables.
* For example, DNO is a primary key in the DEPT table and a non-key in EMP

**Example:**

Refer Table STUDENT shown above.

STUD\_NO in STUDENT\_COURSE is a

foreign key to STUD\_NO in STUDENT relation.

**Table STUDENT\_COURSE**

**6. Composite Key:**Sometimes, a table might not have a single column/attribute that uniquely identifies all the records of a table. To uniquely identify rows of a table, a combination of two or more columns/attributes can be used.  It still can give duplicate values in rare cases. So, we need to find the optimal set of attributes that can uniquely identify rows in a table.

* It acts as a primary key if there is no primary key in a table
* Two or more attributes are used together to make a composite key.
* Different combinations of attributes may give different accuracy in terms of identifying the rows uniquely.

**Example:**

FULLNAME + DOB can be combined

together to access the details of a student.