Module 2

Entity – Relationship Data Model

Topics:

- Introduction to Data Models
- The Entity- Relationship (ER) Model
- Entity
- Entity Set
- Strong and Weak Entity
- Types of Attributes
- Keys
- Relationship Constraints: Cardinality and Participation
- Extended Entity- Relationship (EER) Model: Generalization
- Specialization and Aggregation

- # Introduction to Data Models

A Database model defines the logical design and structure of a database and defines how data will be stored, accessed and updated in a database management system. While the **Relational Model** is the most widely used database model, there are other models too:

- Hierarchical Model
- Network Model
- Entity-relationship Model
- Relational Model

---

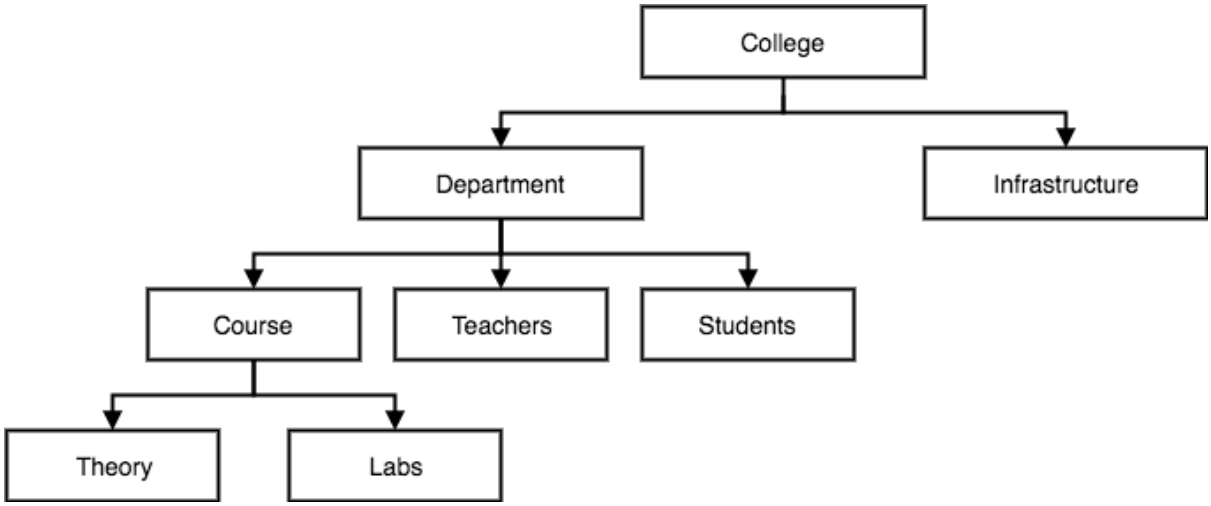
1. **Hierarchical Model**

This database model organises data into a tree-like-structure, with a single root, to which all the other data is linked. The heirarchy starts from the **Root** data, and expands like a tree, adding child nodes to the parent nodes.

In this model, a child node will only have a single parent node.

This model efficiently describes many real-world relationships like index of a book, recipes etc.

In hierarchical model, data is organised into tree-like structure with one one-to-many relationship between two different types of data, for example, one department can have many courses, many professors and of-course many students.

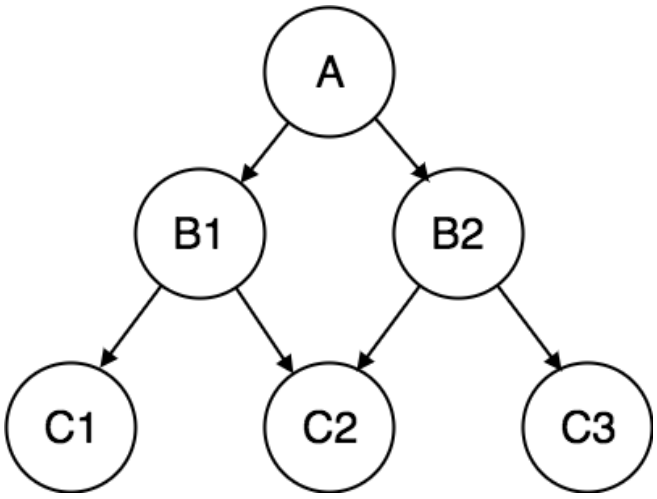


2. Network Model

This is an extension of the Hierarchical model. In this model data is organised more like a graph, and are allowed to have more than one parent node.

In this database model data is more related as more relationships are established in this database model. Also, as the data is more related, hence accessing the data is also easier and fast. This database model was used to map many-to-many data relationships.

This was the most widely used database model, before Relational Model was introduced.



3. Entity-relationship Model

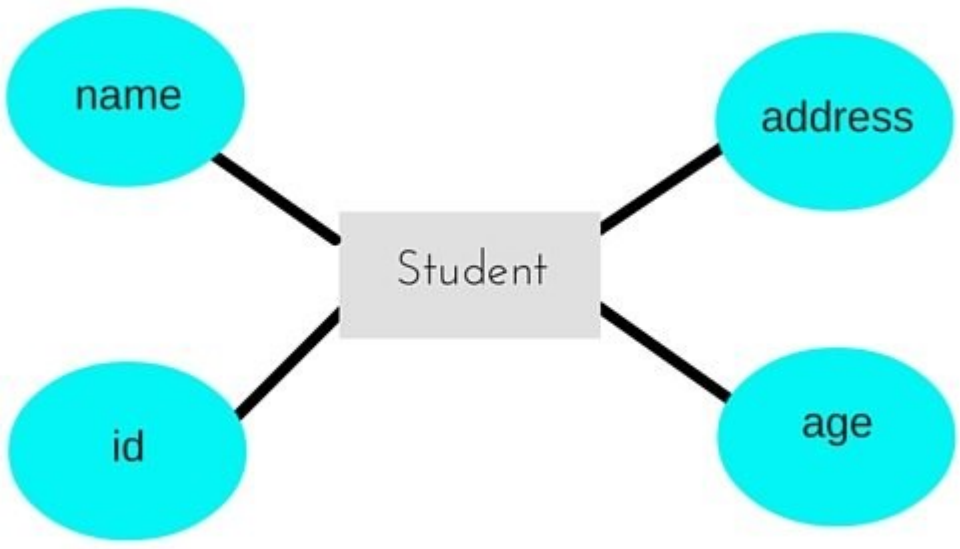
In this database model, relationships are created by dividing object of interest into entity and its characteristics into attributes.

Different entities are related using relationships.

E-R Models are defined to represent the relationships into pictorial form to make it easier for different stakeholders to understand.

This model is good to design a database, which can then be turned into tables in relational model.

Let's take an example, If we have to design a School Database, then **Student** will be an **entity** with **attributes** name, age, address etc. As **Address** is generally complex, it can be another **entity** with **attributes** street name, pincode, city etc, and there will be a relationship between them.



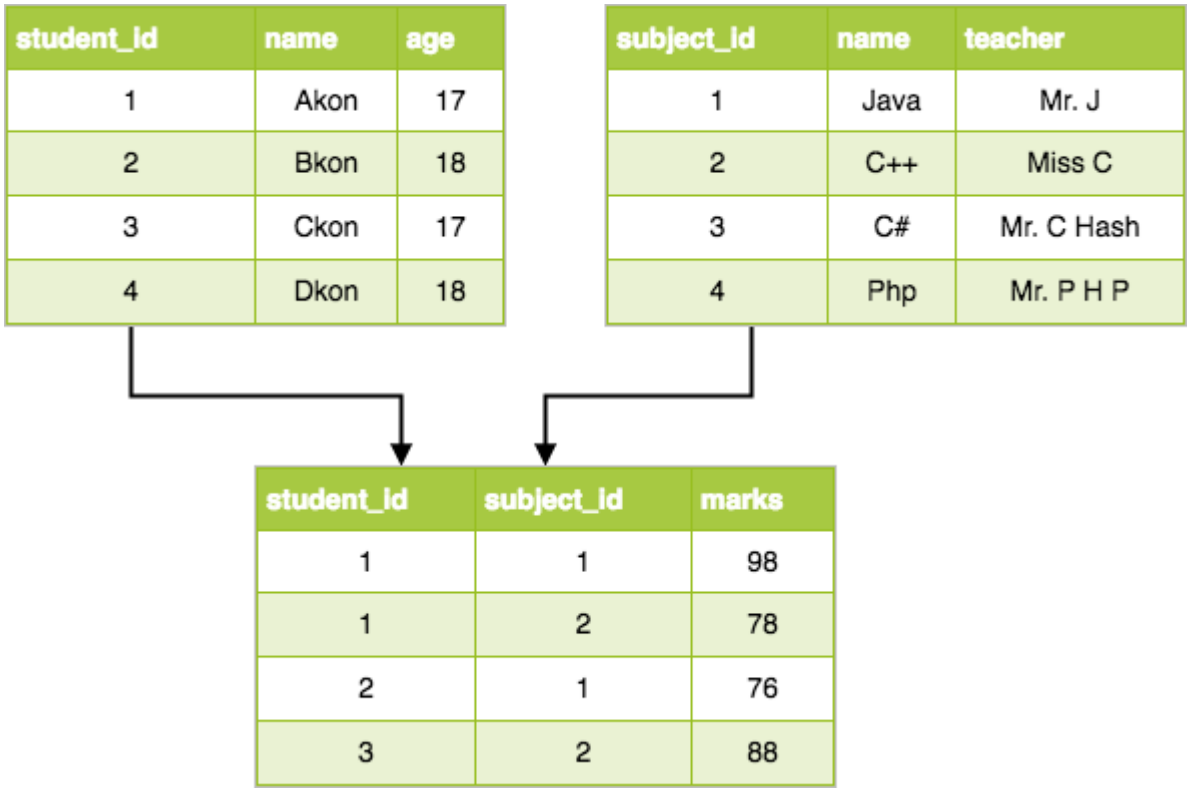
4. Relational Model

In this model, data is organised in two-dimensional **tables** and the relationship is maintained by storing a common field.

This model was introduced by E.F Codd in 1970, and since then it has been the most widely used database model, infact, we can say the only database model used around the world.

The basic structure of data in the relational model is tables. All the information related to a particular type is stored in rows of that table.

Hence, tables are also known as **relations** in relational model.



student_id	name	age
1	Akon	17
2	Bkon	18
3	Ckon	17
4	Dkon	18

subject_id	name	teacher
1	Java	Mr. J
2	C++	Miss C
3	C#	Mr. C Hash
4	Php	Mr. P H P

student_id	subject_id	marks
1	1	98
1	2	78
2	1	76
3	2	88

Difference between Hierarchical, Network and Relational Data Model:

S. No.	Hierarchical Data Model	Network Data Model	Relational Data Model
1.	In this model, to store data hierarchy method is used. It is the oldest method and not in use today.	It organizes records to one another through links or pointers.	It organizes records in the form of table and relationship between tables are set using common fields.

S. No.	Hierarchical Data Model	Network Data Model	Relational Data Model
2.	To organize records, it uses tree structure.	It organizes records in the form of directed graphs.	It organizes records in the form of tables.
3.	It implements 1:1 and 1:n relations.	In addition to 1:1 and 1:n it also implements many to many relationships.	In addition to 1:1 and 1:n it also implements many to many relationships.
4.	Pointers are used to establish relationships among records physically.	A linked list is used to establish a relationship among records physically.	The logical representation is used with rows and columns to depict relationship among records.
5.	Insertion anomaly exits in this model i.e. child node cannot be inserted without the parent node.	There is no insertion anomaly.	There is no insertion anomaly.
6.	Deletion anomaly exists in this model i.e. it is difficult to delete the parent node.	There is no deletion anomaly.	There is no deletion anomaly.
7.	Update leads to inconsistency problems because of the existence of multiple instances of a child record.	No such problem as only one instance of records exist.	Updating a record is easy and simple with the process of normalization, the redundant data gets removed.
8.	This model lacks data independence.	There is partial data independence in this model.	This model provides data independence.
9.	No such facility for querying database is supported.	No such facility for querying database is supported.	SQL-based declarative querying is supported.

S. No.	Hierarchical Data Model	Network Data Model	Relational Data Model
10.	It is used to access the data which is complex and asymmetric.	It is used to access the data which is complex and symmetric.	It is used to access the data which is complex and symmetric.
11.	Difficult to design a database because of its complexity.	Difficult to design a database and manipulate a database because of its complexity. Hence, it imposes a burden on the programmer.	It is easy to comprehend due to concealed physical level details from end-users.
12.	It is less flexible.	It is flexible as compared to the hierarchical model.	It is flexible as compared to the hierarchical model.
13.	&XML and XAML use this model.	VAX-DBMS, DMS-1100 of UNIVAC and SUPRADBMS's use this model.	It is mostly used in real world applications. Oracle, SQL.

• The Entity- Relationship (ER) Model

- **ER Model** stands for Entity Relationship Model is a high-level conceptual data model diagram. ER model helps to systematically analyze data requirements to produce a well-designed database. The ER Model represents real-world entities and the relationships between them. Creating an ER Model in DBMS is considered as a best practice before implementing your database.

- ER Modeling helps to analyze data requirements systematically to produce a well-designed database. So, it is considered a best practice to complete ER modeling before implementing your database.

Why use ER Diagrams?

Here, are prime reasons for using the ER Diagram

- Helps you to define terms related to entity relationship modeling

- Provide a preview of how all your tables should connect, what fields are going to be on each table
- Helps to describe entities, attributes, relationships
- ER diagrams are translatable into relational tables which allows you to build databases quickly
- ER diagrams can be used by database designers as a blueprint for implementing data in specific software applications
- The database designer gains a better understanding of the information to be contained in the database with the help of ERP diagram
- ERD Diagram allows you to communicate with the logical structure of the database to users.

ER Diagrams Symbols & Notations

Entity Relationship Diagram Symbols & Notations mainly contains three basic symbols which are rectangle, oval and diamond to represent relationships between elements, entities and attributes. There are some sub-elements which are based on main elements in ERD Diagram. ER Diagram is a visual representation of data that describes how data is related to each other using different ERD Symbols and Notations.

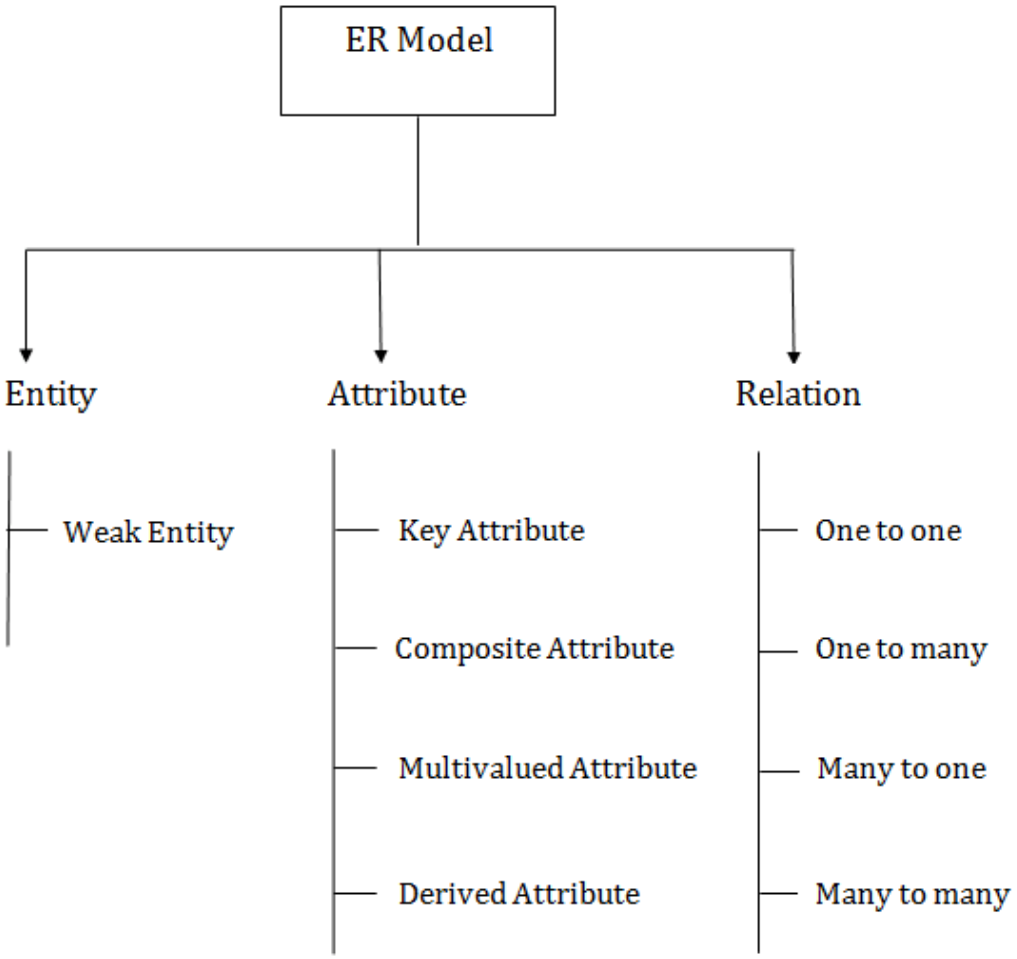
Following are the main components and its symbols in ER Diagrams:

- **Rectangles:** This Entity Relationship Diagram symbol represents entity types
- **Ellipses :** Symbol represent attributes
- **Diamonds:** This symbol represents relationship types
- **Lines:** It links attributes to entity types and entity types with other relationship types
- **Primary key:** attributes are underlined
- **Double Ellipses:** Represent multi-valued attributes

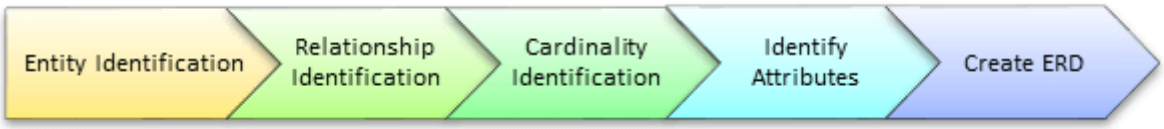
Components of the ER Diagram

This model is based on three basic concepts:

- Entities
- Attributes
- Relationships



How to Create an Entity Relationship Diagram (ERD)

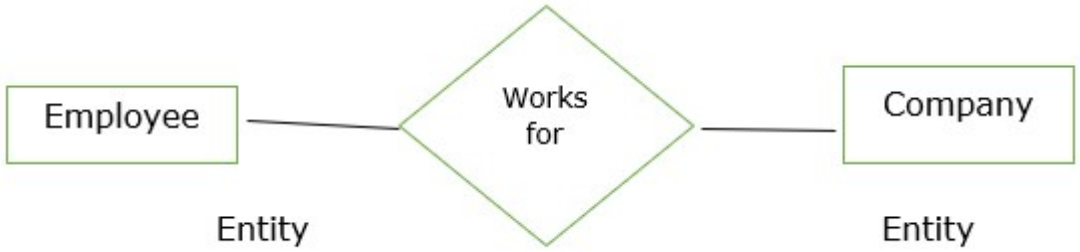


Steps to Create an ER Diagram

Entity

It may be an object, person, place or event that stores data in a database. In a relationship diagram an entity is represented in rectangle form. For example, students, employees, managers, etc.

The entity is pictorially depicted as follows –



Entity set

It is a collection of entities of the same type which share similar properties. For example, a group of students in a college and students are an entity set.

Entity is characterised into two types as follows –

- Strong entity set
- Weak entity set

Strong entity set

The entity types which consist of key attributes or if there are enough attributes for forming a primary key attribute are called a strong entity set. It is represented by a single rectangle.

For Example,

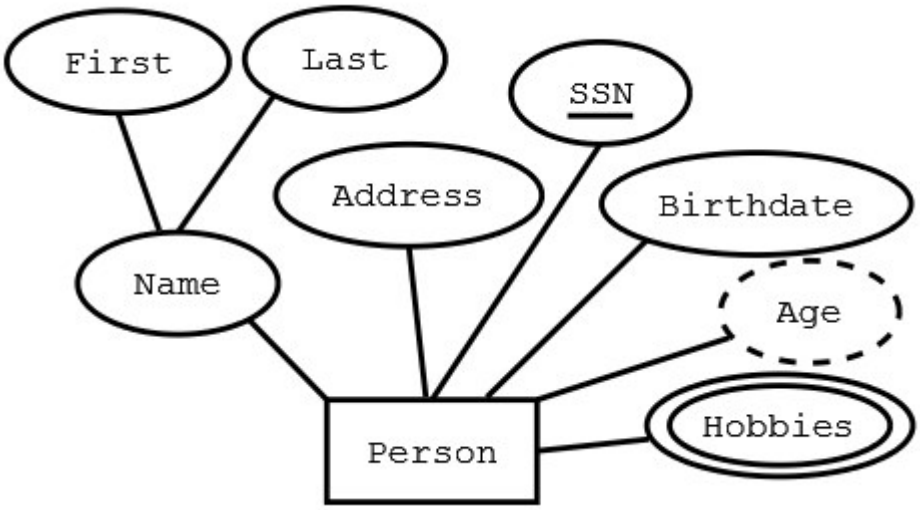
```
Roll no of student
EmpID of employee
```

Weak entity set

An entity does not have a primary key attribute and depends on another strong entity via foreign key attribute. It is represented by a double rectangle.

Attributes

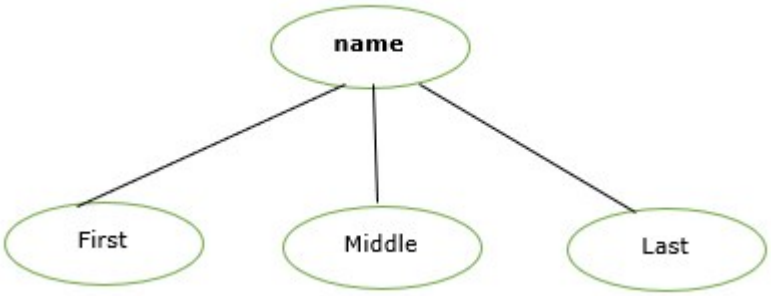
It is the name, thing etc. These are the data characteristics of entities or data elements and data fields.



Types of attributes

The types of attributes in the Entity Relationship (ER) model are as follows –

- **Single value attribute** – These attributes contain a single value. For example, age, salary etc.
- **Multivalued attribute** – They contain more than one value of a single entity. For example, phone numbers.
- **Composite attribute** – The attributes which can be further divided. For example, Name-> First name, Middle name, last name
- **Derived attribute** – The attribute that can be derived from others. For example, Date of Birth.



Relationship

It is used to describe the relation between two or more entities. It is represented by a diamond shape.

For Example, students study in college and employees work in a department.

The relationship is pictorially represented as follows –

Here works for is a relation between two entities.

Degree of Relationship

A relationship where a number of different entities set participate is called a degree of a relationship.

It is categorised into the following –

- Unary Relationship
- Binary Relationship
- Ternary Relationship
- n-ary Relationship

Types of relationship are as follows:

a. One-to-One Relationship

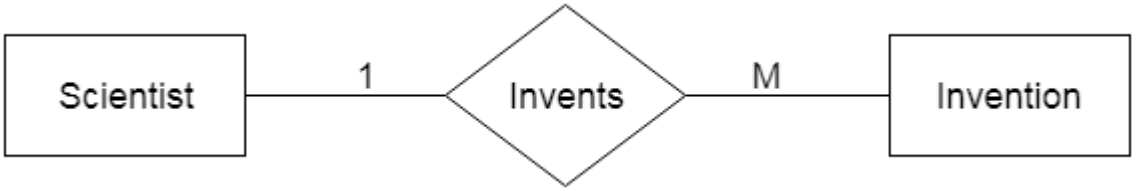
When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

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

b. One-to-many relationship

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

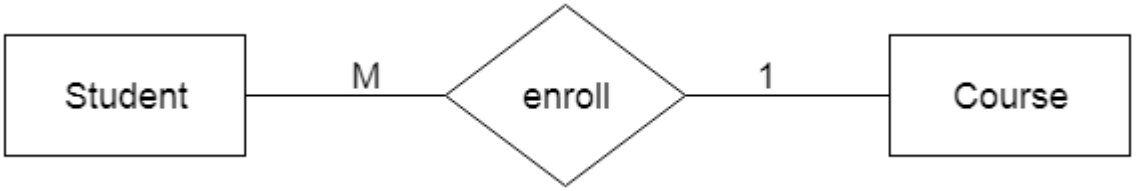
For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.



c. Many-to-one relationship

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

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



d. Many-to-many relationship

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

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

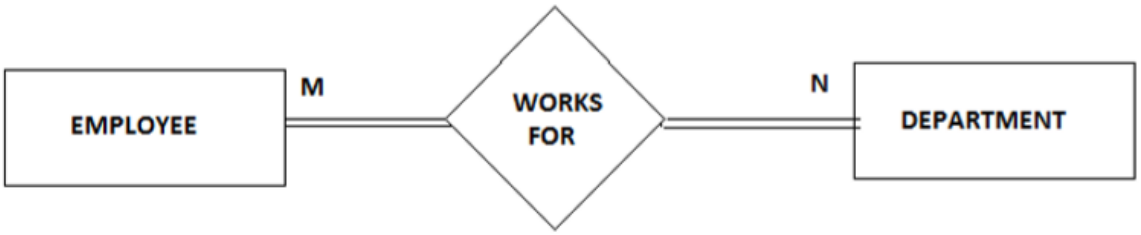
Entity Relationship Participation in Database

In a Relationship, Participation constraint specifies the existence of an entity when it is related to another entity in a relationship type. It is also called minimum cardinality constraint.

This constraint specifies the number of instances of an entity that can participate in a relationship type.

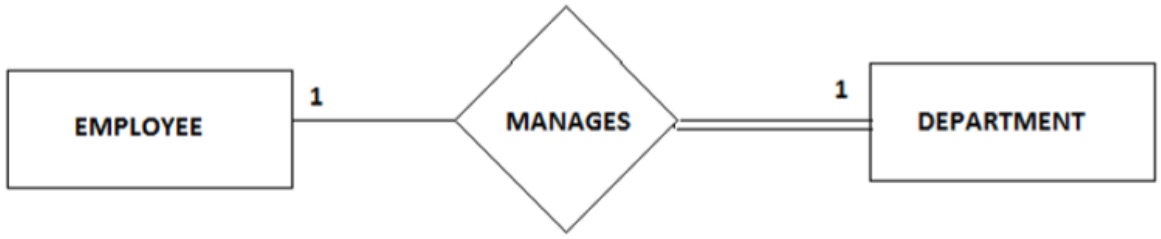
There are two types of Participation constraint –

- ## Total Participation
- Each entity in the entity set is involved in at least one relationship in a relationship set i.e. the number of relationship in every entity is involved is greater than 0.



- Consider two entities Employee and Department related via Works_For relationship. Now, every Employee works in at least one department therefore an Employee entity exist if it has at least one Works_For relationship with Department entity. Thus the participation of Employee in Works_For is total relationship.Total Participation is represented by double line in ER diagram.

- ## Partial Participation
- Each entity in entity set may or may not occur in at least one relationship in a relationship set.
- For example: Consider two entities Employee and Department and they are related to each other via Manages relationship. An Employee must manage a Department, he or she could be the head of the department. But not every Employee in the company manages the department. So, participation of employee in the Manages relationship type is partial i.e. only a particular set of Employees will manage the Department but not all.



- ## Cardinality in DBMS

In database management, cardinality plays an important role. Here cardinality represents the number of times an entity of an entity set participates in a relationship set. Or we can say that the cardinality of a relationship is the number of tuples (rows) in a relationship. Types of cardinality in between tables are:

- one-to-one
- one-to-many
- many-to-one
- many-to-many

Mapping Cardinalities

In a database, the mapping cardinality or cardinality ratio means to denote the number of entities to which another entity can be linked through a certain relation set. Mapping cardinality is most useful in describing binary relation sets, although they can contribute to the description of relation sets containing more than two entity sets. Here, we will focus only on binary relation sets means we will find the relation between entity sets A and B for the set R. So we can map any one of following the cardinality:

1. One-to-one: In this type of cardinality mapping, an entity in A is connected to at most one entity in B. Or we can say that a unit or item in B is connected to at most one unit or item in A.

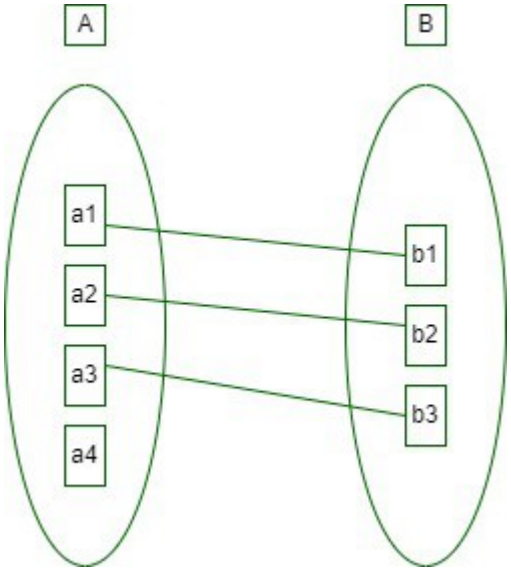


Figure 1

Example:
In a particular hospital, the surgeon department has one head of department. They both serve one-to-one relationships.

2. One-to-many: In this type of cardinality mapping, an entity in A is associated with any number of entities in B. Or we can say that one unit or item in B can be connected to at most one unit or item in A.

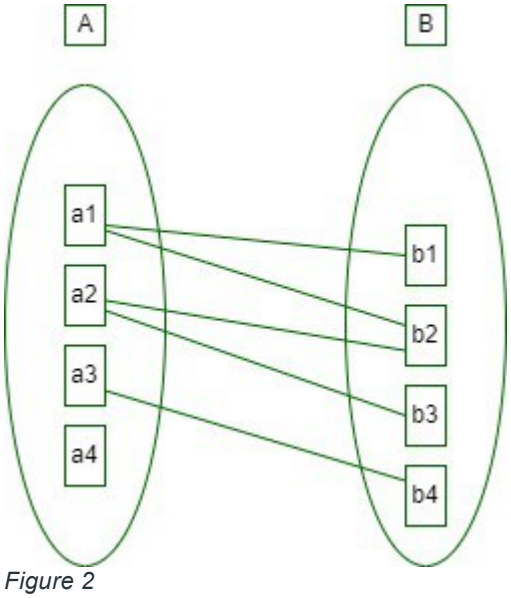


Figure 2

Example:
In a particular hospital, the surgeon department has multiple doctors. They serve one-to-many relationships.

3. Many-to-one: In this type of cardinality mapping, an entity in A is connected to at most one entity in B. Or we can say a unit or item in B can be associated with any number (zero or more) of entities or items in A.

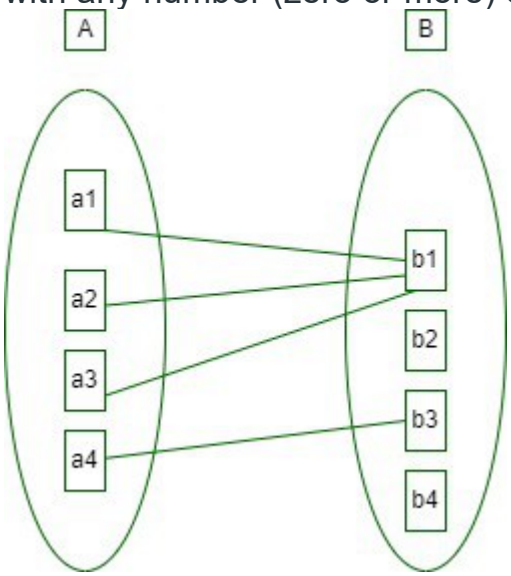
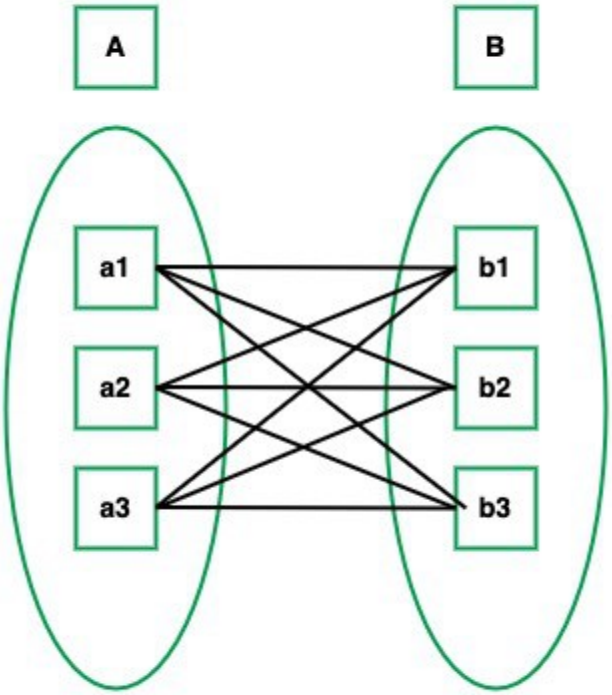


Figure 3

Example:
In a particular hospital, multiple surgeries are done by a single surgeon. Such a type of relationship is known as a many-to-one relationship.

4. Many-to-many: In this type of cardinality mapping, an entity in A is associated with any number of entities in B, and an entity in B is associated with any number of entities in A.



Example:
In a particular company, multiple people work on multiple projects. They serve many-to-many relationships.

The appropriate mapping cardinality for a particular relation set obviously depends on the real-world situation in which the relation set is modeled.

- If we have cardinality one-to-many or many to one then, we can mix relational tables with many involved tables.
- If the cardinality is many-to-many we cant mix any two tables.
- If we have a one-to-one relation and we have total participation of one entity then we can mix that entity with a relation table and if we have total participation of both entities then we can make one table by mixing two entities and their relation.

• Extended Entity- Relationship (EER) Model: Generalization, Specialization and Aggregation

As the complexity of data increased in the late 1980s, it became more and more difficult to use the traditional ER Model for database modelling. Hence some improvements or enhancements were made to the existing ER Model to make it able to handle the complex applications better.

Hence, as part of the **Enhanced ER Model**, along with other improvements, three new concepts were added to the existing ER Model, they were:

1. Generalization
2. Specialization
3. Aggregation

Extended ER is a high-level data model that incorporates the extensions to the original ER model. Enhanced ER models are high level models that represent the requirements and complexities of complex databases.

The extended Entity Relationship (ER) models are three types as given below –

- Aggregation
- Specialization
- Generalization

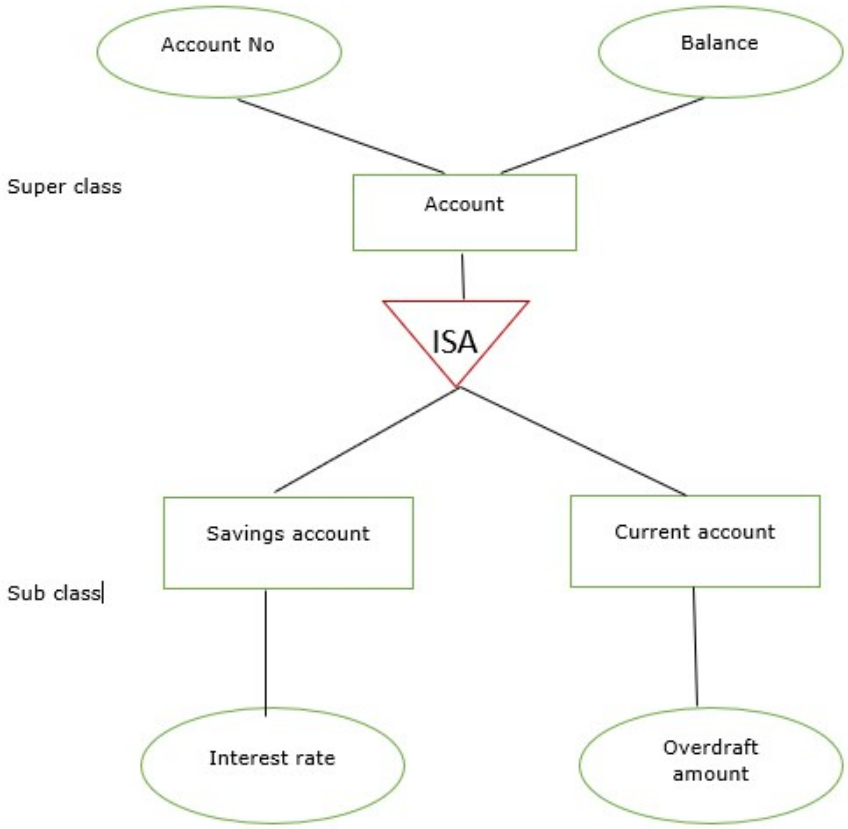
Specialization

The process of designing sub groupings within an entity set is called specialization. It is a top-down process. If an entity set is given with all the attributes in which the instances of the entity set are differentiated according to the given attribute value, then that sub-classes or the sub-entity sets can be formed from the given attribute.

Example

Specialization of a person allows us to distinguish a person according to whether they are employees or customers. Specialization of account creates two entity sets: savings account and current account.

In the E-R diagram specialization is represented by triangle components labeled ISA. The ISA relationship is referred as superclass- subclass relationship as shown below –



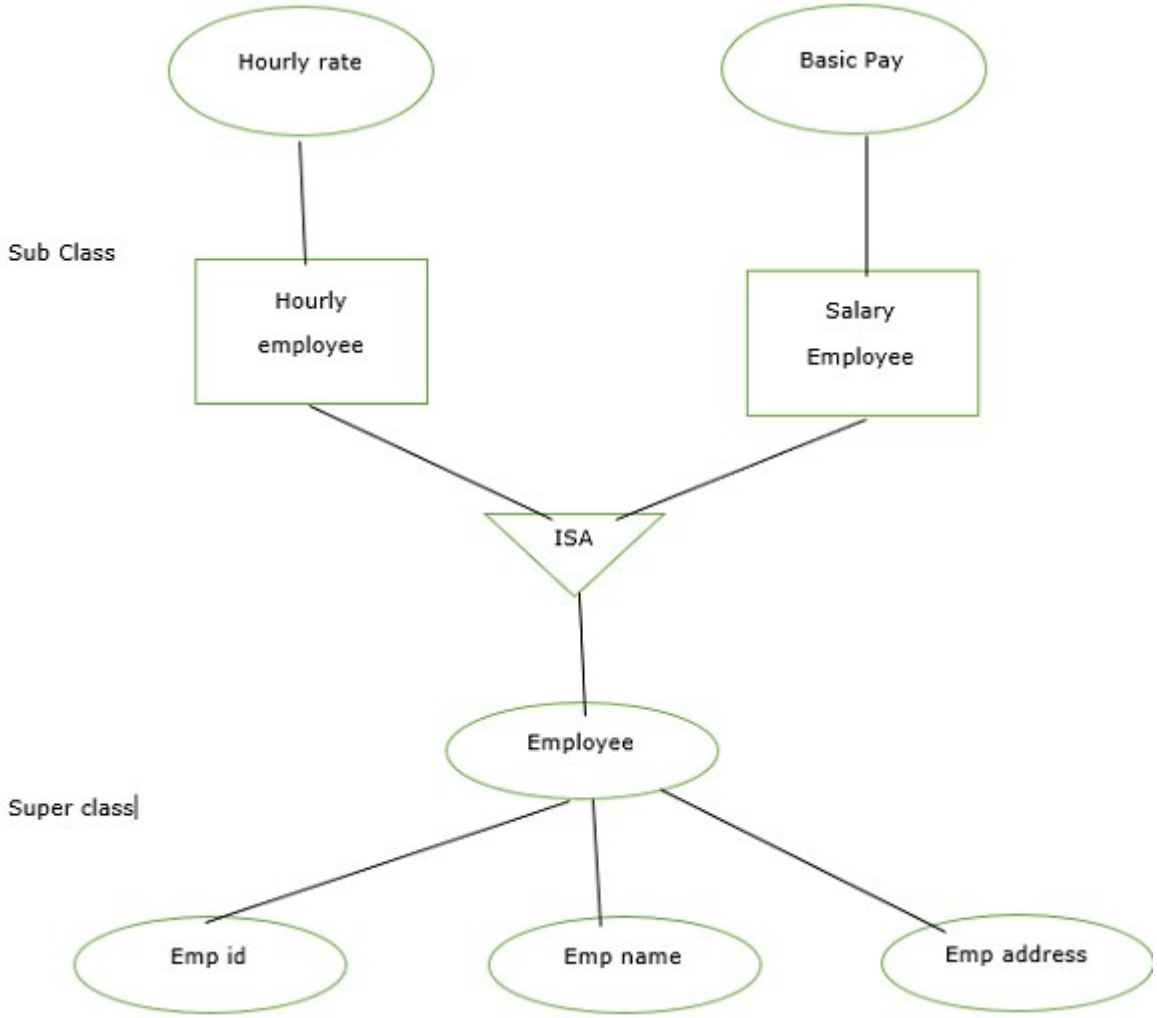
Generalization

It is the reverse process of specialization. It is a bottom-up approach.

It converts subclasses to superclasses. This process combines a number of entity sets that share the same features into higher-level entity sets.

If the sub-class information is given for the given entity set then, ISA relationship type will be used to represent the connectivity between the subclass and superclass as shown below –

Example

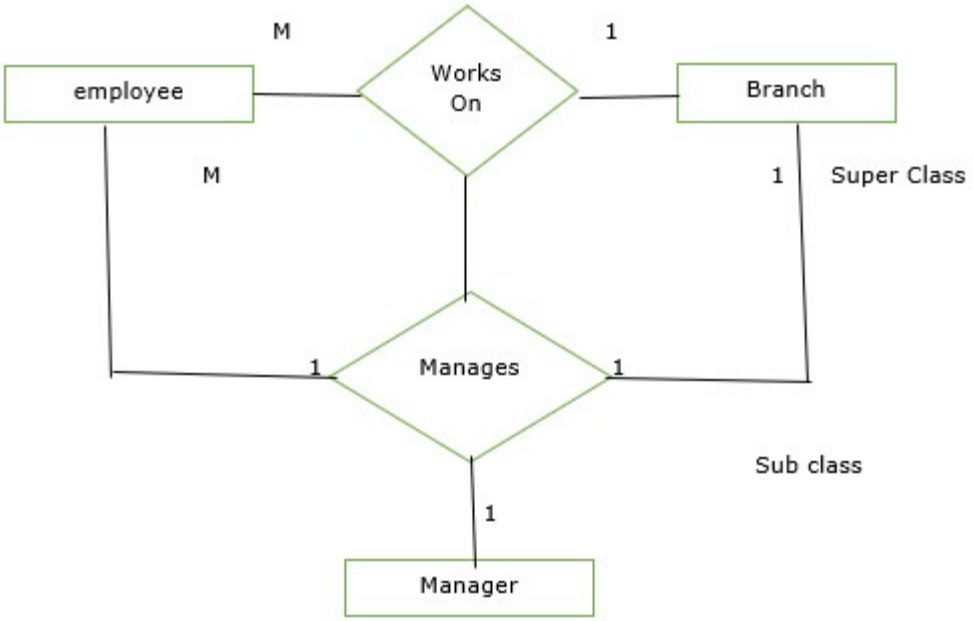


Aggregation

It is an abstraction in which relationship sets are treated as higher level entity sets and can participate in relationships. Aggregation allows us to indicate that a relationship set participates in another relationship set.

Aggregation is used to simplify the details of a given database where ternary relationships will be changed into binary relationships. Ternary relation is only one type of relationship which is working between three entities.

Aggregation is shown in the image below –



What are Keys in DBMS?

KEYS in DBMS is an attribute or set of attributes which helps you to identify a row(tuple) in a relation(table). They allow you to find the relation between two tables. Keys help you uniquely identify a row in a table by a combination of one or more columns in that table. Key is also helpful for finding unique record or row from the table. Database key is also helpful for finding unique record or row from the table.

Example:

Employee ID	FirstName	LastName
11	Andrew	Johnson
22	Tom	Wood
33	Alex	Hale

In the above-given example, employee ID is a primary key because it uniquely identifies an employee record. In this table, no other employee can have the same employee ID.

Why we need a Key?

Here are some reasons for using sql key in the DBMS system.

- Keys help you to identify any row of data in a table. In a real-world application, a table could contain thousands of records. Moreover, the

records could be duplicated. Keys in RDBMS ensure that you can uniquely identify a table record despite these challenges.
- Allows you to establish a relationship between and identify the relation between tables
- Help you to enforce identity and integrity in the relationship.

Types of Keys in DBMS (Database Management System)

There are mainly Eight different types of Keys in DBMS and each key has it's different functionality:

1. Super Key
2. Primary Key
3. Candidate Key
4. Alternate Key
5. Foreign Key
6. Compound Key
7. Composite Key
8. Surrogate Key

Let's look at each of the keys in DBMS with example:

- **Super Key –** A super key is a group of single or multiple keys which identifies rows in a table.
- **Primary Key –** is a column or group of columns in a table that uniquely identify every row in that table.
- **Candidate Key –** is a set of attributes that uniquely identify tuples in a table. Candidate Key is a super key with no repeated attributes.
- **Alternate Key –** is a column or group of columns in a table that uniquely identify every row in that table.
- **Foreign Key –** is a column that creates a relationship between two tables. The purpose of Foreign keys is to maintain data integrity and allow navigation between two different instances of an entity.
- **Compound Key –** has two or more attributes that allow you to uniquely recognize a specific record. It is possible that each column may not be unique by itself within the database.
- **Composite Key –** is a combination of two or more columns that uniquely identify rows in a table. The combination of columns guarantees uniqueness, though individual uniqueness is not guaranteed.

- **Surrogate Key –** An artificial key which aims to uniquely identify each record is called a surrogate key. These kind of key are unique because they are created when you don't have any natural primary key.

What is the Super key?

A superkey is a group of single or multiple keys which identifies rows in a table. A Super key may have additional attributes that are not needed for unique identification.

Example:

EmpSSN	EmpNum	Empname
9812345098	AB05	Shown
9876512345	AB06	Roslyn
199937890	AB07	James

In the above-given example, EmpSSN and EmpNum name are superkeys.

What is a Primary Key?

PRIMARY KEY in DBMS is a column or group of columns in a table that uniquely identify every row in that table. The Primary Key can't be a duplicate meaning the same value can't appear more than once in the table. A table cannot have more than one primary key.

Rules for defining Primary key:

- Two rows can't have the same primary key value
- It must for every row to have a primary key value.
- The primary key field cannot be null.
- The value in a primary key column can never be modified or updated if any foreign key refers to that primary key.

Example:

In the following example, <code>StudID</code> is a Primary Key.

StudID	Roll No	First Name	LastName	Email
1	11	Tom	Price	abc@gmail.com

2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com

What is the Alternate key?

ALTERNATE KEYS is a column or group of columns in a table that uniquely identify every row in that table. A table can have multiple choices for a primary key but only one can be set as the primary key. All the keys which are not primary key are called an Alternate Key.
Example:

In this table, StudID, Roll No, Email are qualified to become a primary key. But since StudID is the primary key, Roll No, Email becomes the alternative key.

StudID	Roll No	First Name	LastName	Email
1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com

What is a Candidate Key?

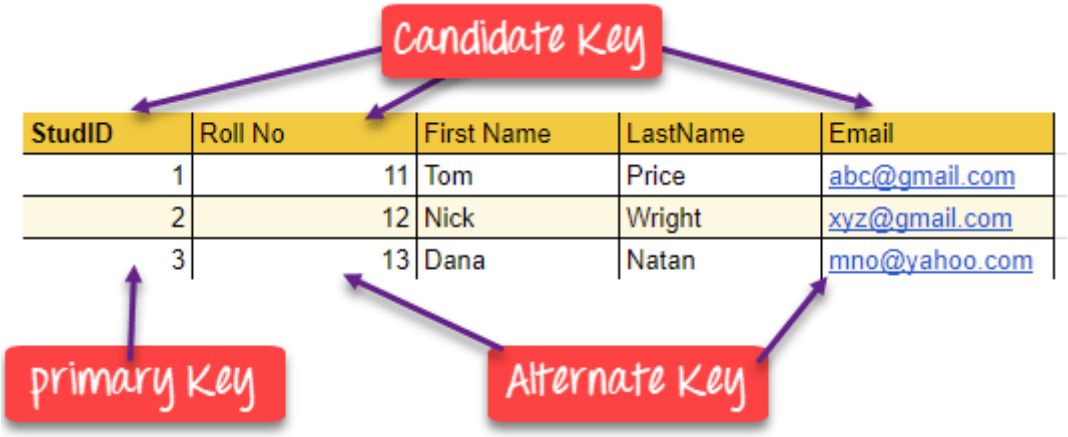
CANDIDATE KEY in SQL is a set of attributes that uniquely identify tuples in a table. Candidate Key is a super key with no repeated attributes. The Primary key should be selected from the candidate keys. Every table must have at least a single candidate key. A table can have multiple candidate keys but only a single primary key.
Properties of Candidate key:

- It must contain unique values
- Candidate key in SQL may have multiple attributes
- Must not contain null values
- It should contain minimum fields to ensure uniqueness
- Uniquely identify each record in a table

Candidate key Example: In the given table Stud ID, Roll No, and email are candidate keys which help us to uniquely identify the student record in the table.

StudID	Roll No	First Name	LastName	Email

1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com



Candidate Key in DBMS

What is the Foreign key?

FOREIGN KEY is a column that creates a relationship between two tables. The purpose of Foreign keys is to maintain data integrity and allow navigation between two different instances of an entity. It acts as a cross-reference between two tables as it references the primary key of another table.
Example:

DeptCode	DeptName
001	Science
002	English
005	Computer

Teacher ID	Fname	Lname
B002	David	Warner
B017	Sara	Joseph
B009	Mike	Brunton

In this key in dbms example, we have two table, teach and department in a school. However, there is no way to see which search work in which department.

In this table, adding the foreign key in Deptcode to the Teacher name, we can create a relationship between the two tables.

Teacher ID	DeptCode	Fname	Lname
B002	002	David	Warner
B017	002	Sara	Joseph
B009	001	Mike	Brunton

This concept is also known as Referential Integrity.

What is the Compound key?

COMPOUND KEY has two or more attributes that allow you to uniquely recognize a specific record. It is possible that each column may not be unique by itself within the database. However, when combined with the other column or columns the combination of composite keys become unique. The purpose of the compound key in database is to uniquely identify each record in the table.
Example:

OrderNo	PorductID	Product Name	Quantity
B005	JAP102459	Mouse	5
B005	DKT321573	USB	10
B005	OMG446789	LCD Monitor	20
B004	DKT321573	USB	15
B002	OMG446789	Laser Printer	3

In this example, OrderNo and ProductID can’t be a primary key as it does not uniquely identify a record. However, a compound key of Order ID and Product ID could be used as it uniquely identified each record.

What is the Composite key?

COMPOSITE KEY is a combination of two or more columns that uniquely identify rows in a table. The combination of columns guarantees uniqueness, though individually uniqueness is not guaranteed. Hence, they are combined to uniquely identify records in a table.

The difference between compound and the composite key is that any part of the compound key can be a foreign key, but the composite key may or maybe not a part of the foreign key.

What is a Surrogate key?

SURROGATE KEYS is An artificial key which aims to uniquely identify each record is called a surrogate key. This kind of partial key in dbms is unique because it is created when you don't have any natural primary key. They do not lend any meaning to the data in the table. Surrogate key in DBMS is usually an integer. A surrogate key is a value generated right before the record is inserted into a table.

Fname	Lastname	Start Time	End Time
Anne	Smith	09:00	18:00
Jack	Francis	08:00	17:00
Anna	McLean	11:00	20:00
Shown	Willam	14:00	23:00

Above, given example, shown shift timings of the different employee. In this example, a surrogate key is needed to uniquely identify each employee.

Surrogate keys in sql are allowed when

- No property has the parameter of the primary key.
- In the table when the primary key is too big or complicated.

Difference Between Primary key & Foreign key

Following is the main difference between primary key and foreign key:

Primary Key	Foreign Key
Helps you to uniquely identify a record in the table.	It is a field in the table that is the primary key of another table.
Primary Key never accept null values.	A foreign key may accept multiple null values.
Primary key is a clustered index and data in the DBMS table are physically organized in the sequence of the clustered index.	A foreign key cannot automatically create an index, clustered or non-clustered. However, you can manually create an index on the foreign key.

You can have the single Primary key in a table.	You can have multiple foreign keys in a table.