Unit-III
Database Design & the ER Model

Database Design Process

The processes here aren't the same as the agile model or iterative approach. They are defined steps to end up having a fully defined database, with its constraints, and structure.

There is no place for many changes because they are going to cost you a lot. So, you need to be specific and take things step by step. And, here's the steps:

1. Requirements Gathering

Understanding what you want to do, and what you have is essential before you can dive into designing a database. We'll look at the steps in this article.

2. Conceptual Design

We specify the entities, columns, and their relationship. We may use an entity relationship (ER) diagram to visualize the database.

The output is: A conceptual schema (described using a conceptual data model like ER model).

3. Logical Design

It's concerned about data model mapping; mapping a conceptual schema (like ER model) into logical schema to provide a much detailed description.

The output is: A logical schema (described using a logical data model specific to the DBMS like relational model).

4. Physical Design

It describes the details of how data is stored. You start by defining (already modeled) tables, how the data is stored, define relationships, … in DBMS.

This requires dealing with the DBMS, and could involve SQL.

The output is: An internal (physical) schema (described using a physical data model).

Explain the different phases of database design

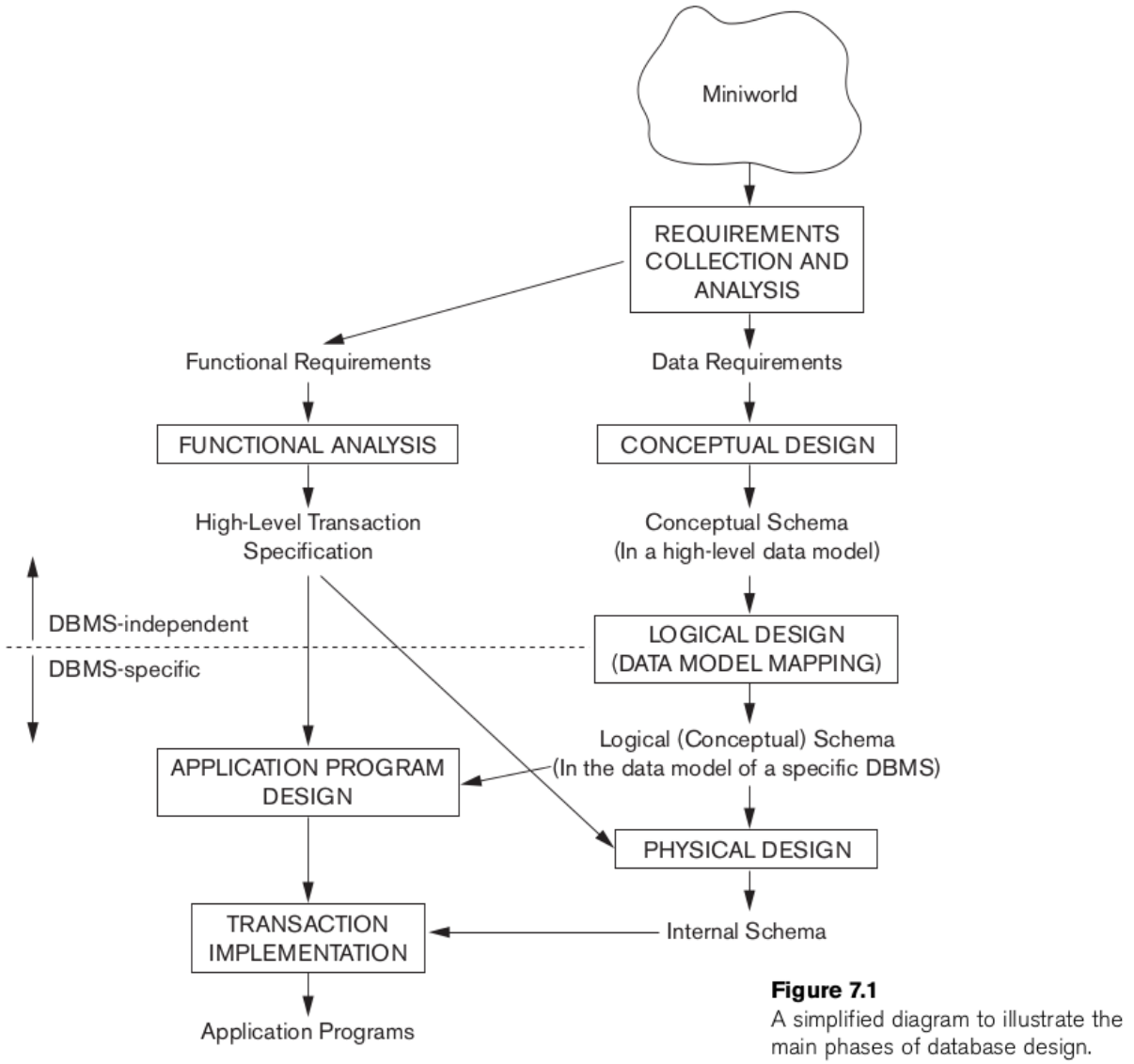


Figure 7.1
A simplified diagram to illustrate the main phases of database design.

1. **Requirement collection & analysis:**
   During this step, the database designers interview the database users to understand and document their data requirements.
   In parallel with specifying the data requirements, it is useful to specify the known functional requirements consisting of user defined operations or transactions that will be applied to the database.

2. **Conceptual database design:**
   Create a conceptual schema for the database using a high level conceptual data model. It describes the data requirements of the users and data types, relationship and constraints, these are expressed using the concepts provided by the high level data model. Because these concepts do not include any implementation details.
3. **Logical database design or data model mapping:**
   This step transforms the conceptual schema from the high level data model into the implementation data model.
4. **Physical database design phase:**
   In this step the internal storage structures and file organization for the database are specified. In parallel with these activity application programs were designed and implemented as database transactions corresponding to high-level transaction specifications.

What is an Entity relationship model in DBMS?

Entity relationship (ER) models are based on the real-world entities and their relationships.

It is easy for the developers to understand the system by simply looking at the ER diagram.

ER models are normally represented by ER-diagrams.

Components

ER diagram basically having three components:

- **Entities** − It is a real-world thing which can be a person, place, or even a concept. For Example: Department, Admin, Courses, Teachers, Students, Building, etc are some of the entities of a School Management System.
- **Attributes** − An entity which contains a real-world property called an attribute. For Example: The entity employee has the property like employee id, salary, age, etc.
- **Relationship** − Relationship tells how two attributes are related. For Example: Employee works for a department.

An entity has a real-world property called attribute and these attributes are defined by a set of values called domain.

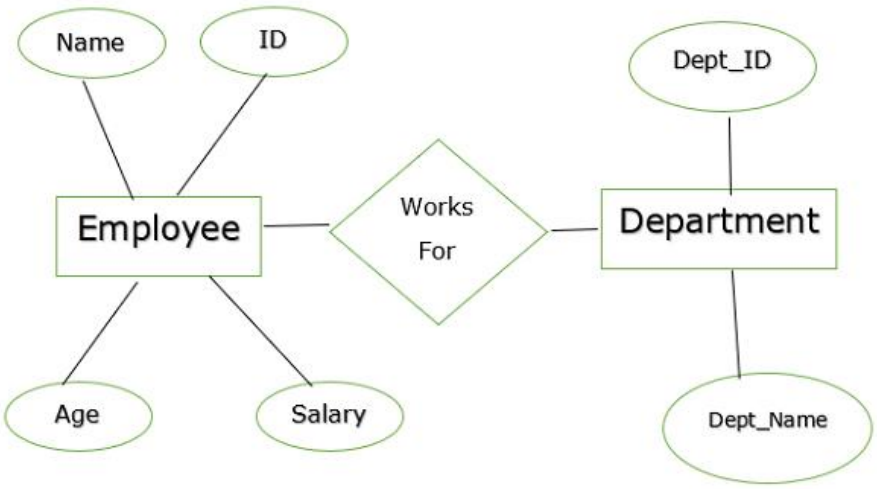
Example 1

In a university,

- A student is an entity,
- University is the database,
- Name and age and sex are the attributes.
- The relationships among entities define the logical association between entities.

Example 2

Given below is another example of ER:



In the above example,

Entities − Employee and Department.

Attributes −

- Employee − Name, id, Age, Salary
- Department − Dept_id, Dept_name

The two entities are connected using the relationship. Here, each employee works for a department.

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

Entity, Entity Type, Entity Set –

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.

An Entity is an object of Entity Type and set of all entities is called as entity set. e.g.; E1 is an entity having Entity Type Student and set of all students is called Entity Set. In ER diagram, Entity Type is represented as:

Entity Type

Entity Set

Example :

- The collection of all the students from the student table at a particular instant of time is an example of an entity set.
- The collection of all the employees from the employee table at a particular instant of time is an example of an entity set.

Note :

- Entity sets need not be disjoint. For example, the entity set of Article Writer (all content creators for GeeksforGeeks) and the entity set of Article Reader (all students who read the article of GeeksforGeeks) may have members in common.
- The collection of all the entities in the relation of RDBMS is called an entity set.

Relation With Table :

Consider a table student as follows :

Table Name : Student

Student_ID	Student_Name	Student_Age	Student_Gender
1	Avi	19	M
2	Ayush	23	M
3	Nikhil	21	M
4	Riya	16	F

Entity : Each row is an entity.
Example :

1 Avi 19 M

Entity Type : Each entity belongs to the student type. Hence, the type of entity here is a student.
Entity Set : The complete data set of all entities is called entity set. For the above table, the records with student id 1, 2, 3, 4 are the entity set.
Difference Table :

Entity	Entity Type	Entity Set
A thing in the real world with independent existence	A category of a particular entity	Set of all entities of a particular entity type.
Any particular row (a record) in a relation(table) is known as an entity.	The name of a relation (table) in RDBMS is an entity type	All rows of a relation (table) in RDBMS is entity set

Relationship in DBMS-

Example-

'Enrolled in' is a relationship that exists between entities **Student** and **Course**.

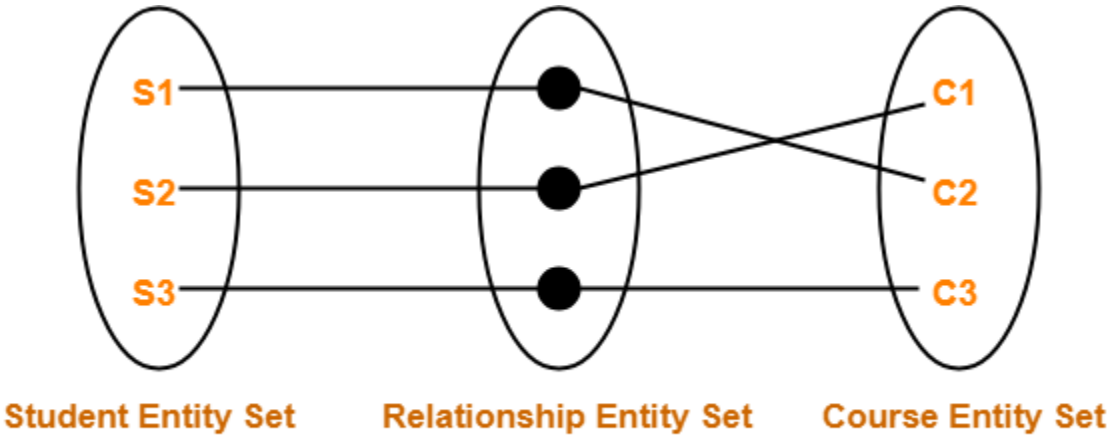
Also read- Entity Sets in DBMS

Relationship Set-

A relationship set is a set of relationships of same type.

Example-

Set representation of above ER diagram is-



Set Representation of ER Diagram

Degree of a Relationship Set-

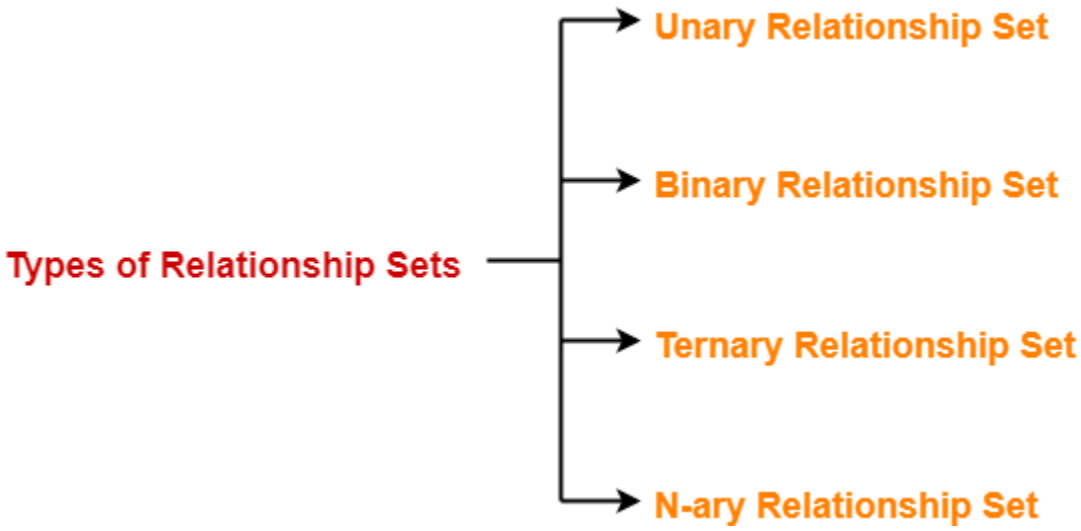
A relationship is defined as an association among several entities.

The number of entity sets that participate in a relationship set is termed as the degree of that relationship set. Thus,

Degree of a relationship set = Number of entity sets participating in a relationship set

Types of Relationship Sets-

On the basis of degree of a relationship set, a relationship set can be classified into the following types-



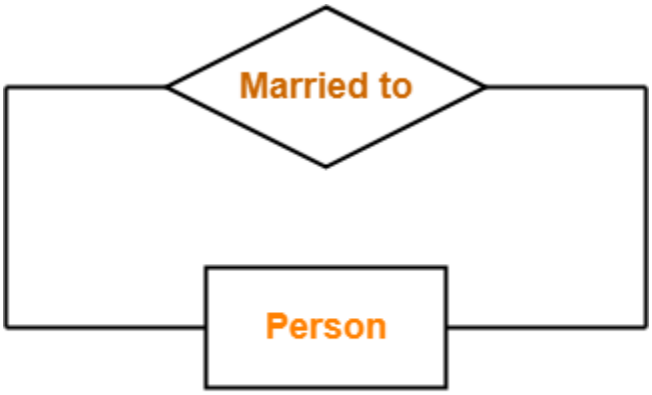
1. Unary relationship set
2. Binary relationship set
3. Ternary relationship set
4. N-ary relationship set

1. Unary Relationship Set-

Unary relationship set is a relationship set where only one entity set participates in a relationship set.

Example-

One person is married to only one person



Unary Relationship Set

2. Binary Relationship Set-

Binary relationship set is a relationship set where two entity sets participate in a relationship set.

Example-

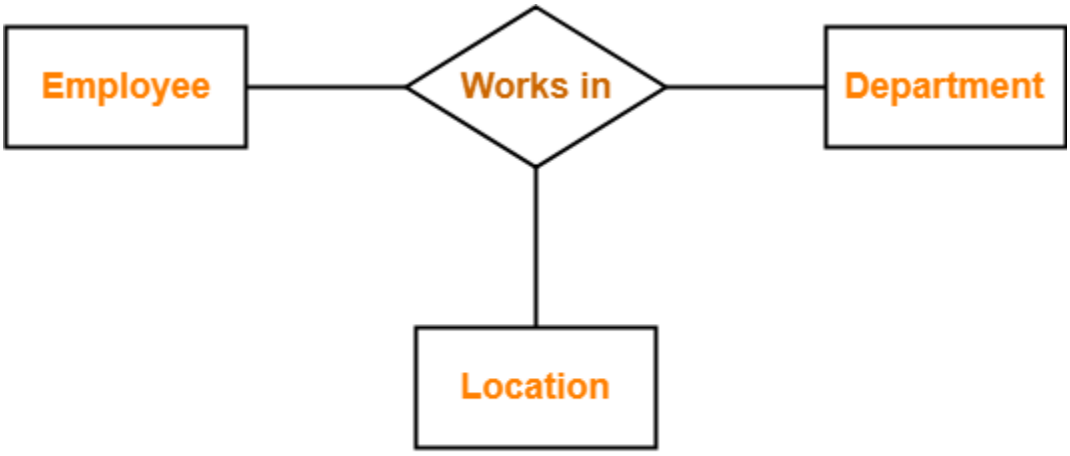
Student is enrolled in a Course

Binary Relationship Set

3. Ternary Relationship Set-

Ternary relationship set is a relationship set where three entity sets participate in a relationship set.

Example-



Ternary Relationship Set

4. N-ary Relationship Set-

N-ary relationship set is a relationship set where ‘n’ entity sets participate in a relationship set.

Keys:

Keys are very important part of Relational database model. They are used to establish and identify relationships between tables and also to uniquely identify any record or row of data inside a table.

A Key can be a single attribute or a group of attributes, where the combination may act as a key.

To avoid all this, **Keys** are defined to easily identify any row of data in a table.

student_id	name	phone	age
1	Akon	9876723452	17
2	Akon	9991165674	19
3	Bkon	7898756543	18
4	Ckon	8987867898	19
5	Dkon	9990080080	17

Let's take a simple **Student** table, with fields student_id, name, phone and age.

Super Key

Super Key is defined as a set of attributes within a table that can uniquely identify each record within a table. Super Key is a superset of Candidate key.

In the table defined above super key would include student_id, (student_id, name), phone etc.

Confused? The first one is pretty simple as student_id is unique for every row of data, hence it can be used to identity each row uniquely.

Next comes, (student_id, name), now name of two students can be same, but their student_id can't be same hence this combination can also be a key.

Similarly, phone number for every student will be unique, hence again, phone can also be a key.

So they all are super keys.

Candidate Key

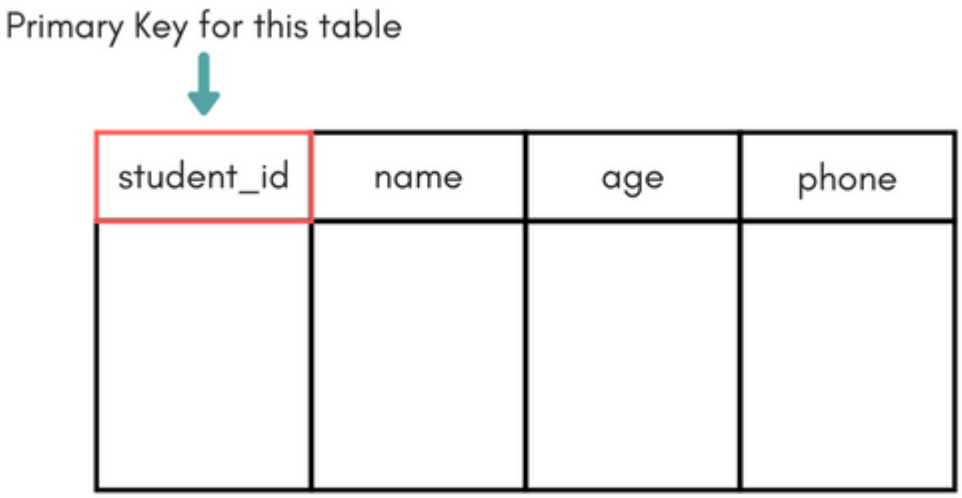
Candidate keys are defined as the minimal set of fields which can uniquely identify each record in a table. It is an attribute or a set of attributes that can act as a Primary Key for a table to uniquely identify each record in that table. There can be more than one candidate key.

In our example, student_id and phone both are candidate keys for table **Student**.

- A candiate key can never be NULL or empty. And its value should be unique.
- There can be more than one candidate keys for a table.
- A candidate key can be a combination of more than one columns(attributes).

Primary Key

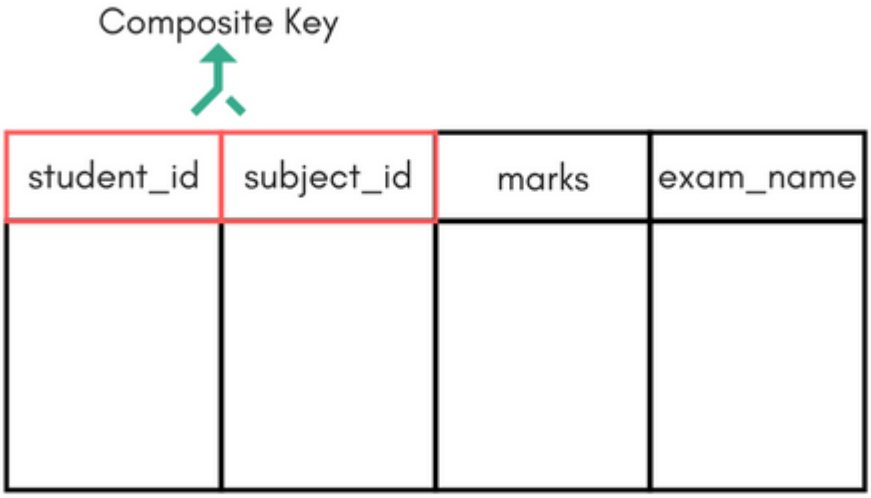
Primary key is a candidate key that is most appropriate to become the main key for any table. It is a key that can uniquely identify each record in a table.



For the table **Student** we can make the student_id column as the primary key.

Composite Key

Key that consists of two or more attributes that uniquely identify any record in a table is called **Composite key**. But the attributes which together form the **Composite key** are not a key independentely or individually.



Score Table - To save scores of the student for various subjects.

In the above picture we have a **Score** table which stores the marks scored by a student in a particular subject.

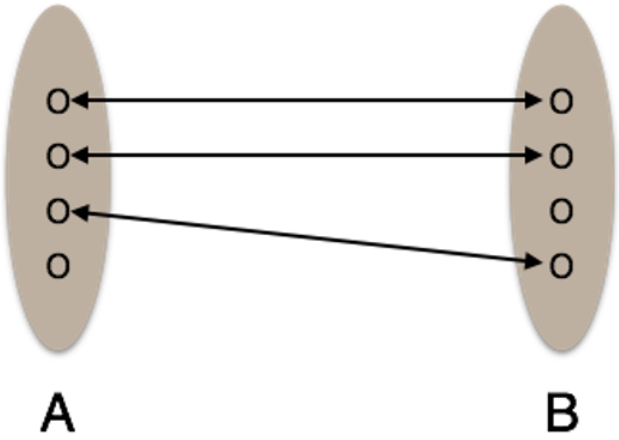
In this table student_id and subject_id together will form the primary key, hence it is a composite key.

Mapping Cardinalities or Mapping Constraints

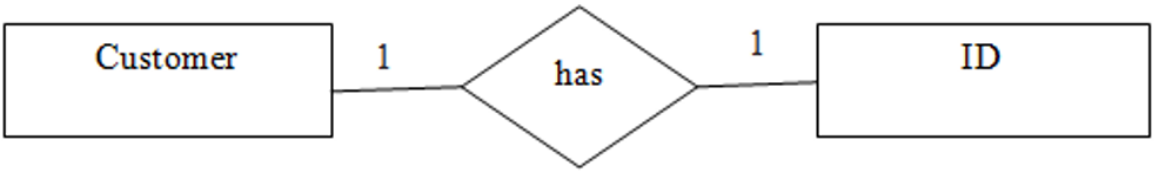
Mapping cardinalities defines the relationship between numbers of entities in one entity set with the number of entities to other entity sets.

One-to-one

As its name implies, it maps one entity of the first entity set with another one in the second one entity set. In the below image this has depicted that one entity from entity set A is associated with at most one entity of other one entity set B.



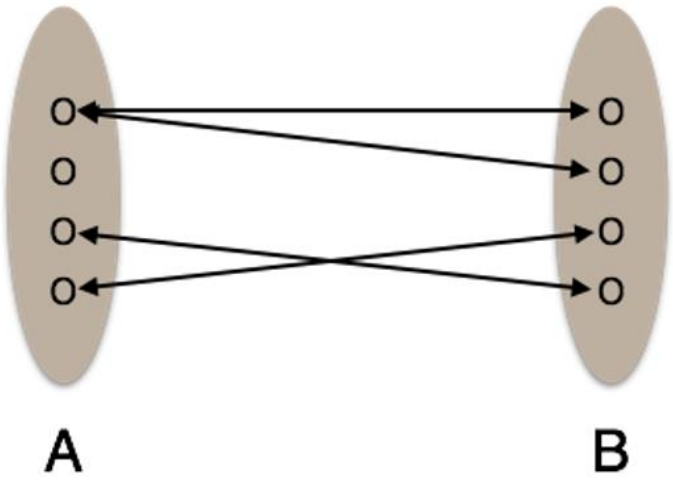
For example - a customer has only one ID.



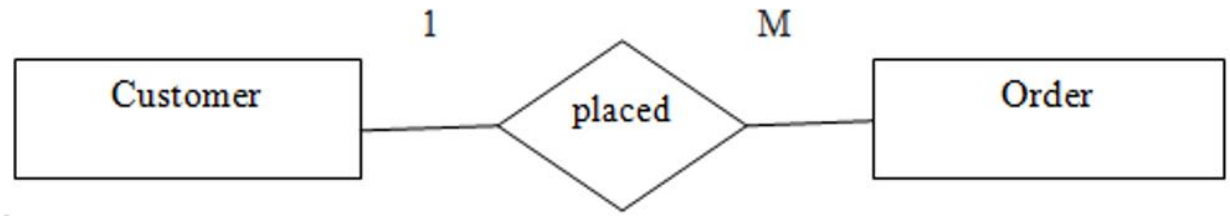
If a single instance of an entity is connected with one instances of another entity, then it is called a relationship between one and many.

One-to-many

An entity from set A can be aligned with more than one entity from set B, but an entity from set B can be associated with a limit of one entity.



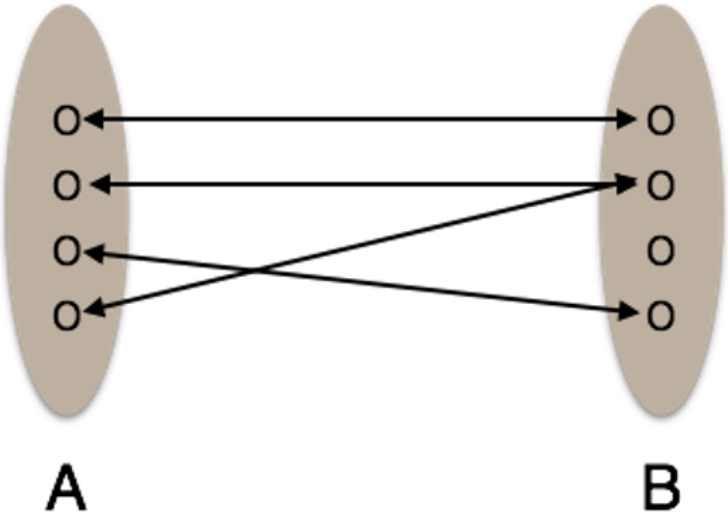
For example - a customer may place many orders, but many customers do not place an order.



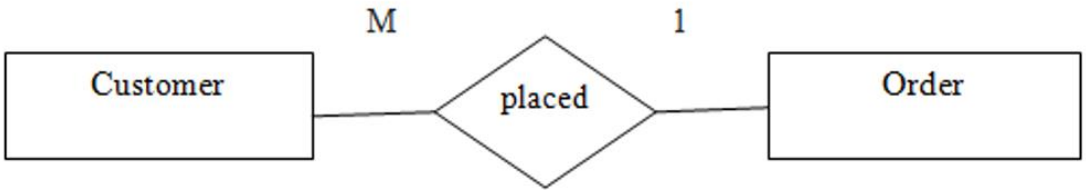
If a single instance of an entity is connected to more than one instances of another entity, then it is called a relationship between one and many.

Many-to-one

More than one entity from set A could be paired with a maximum of one entity from set B, but more than one entity from set B can be paired with a maximum of one entity from set A.



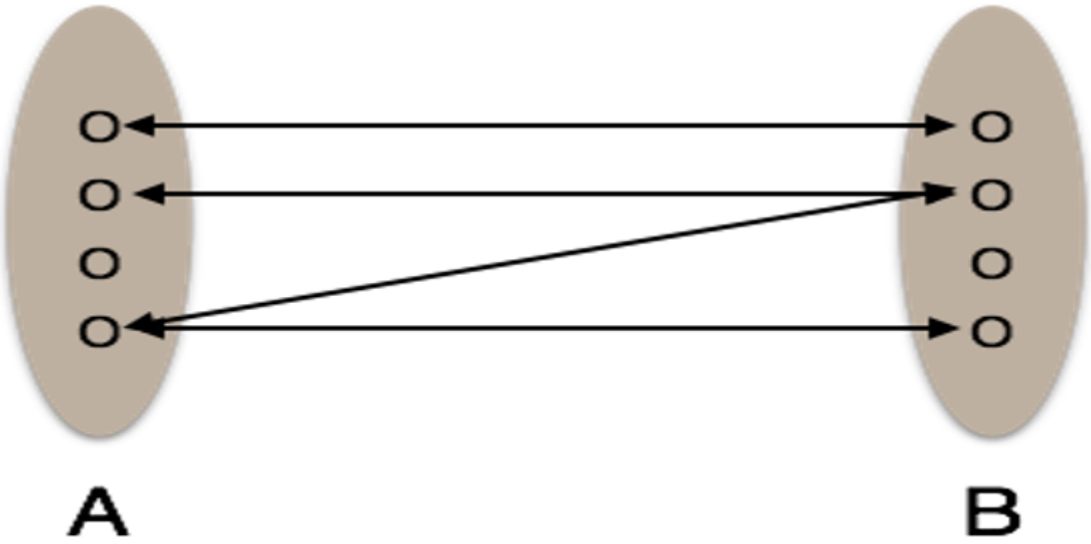
For example – many customers may place 1 order.



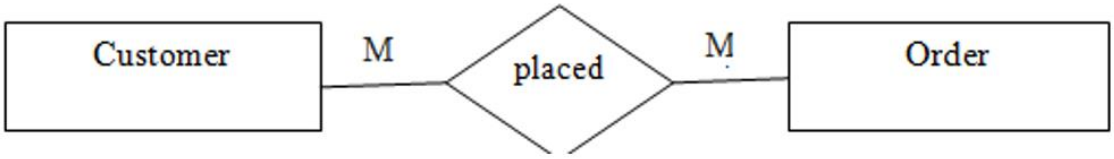
If a single instance of the second one entity is connected with more than one instance of the first entity, then it is referred to as many to one relationship.

Many-to-many

It is possible to connect one entity from set A with more than one entity from set B and vice versa.



For example – many customers may place many orders.



If more than one instance of the first entity is connected with more than one instance of the second entity, it is called a many to many relationships.

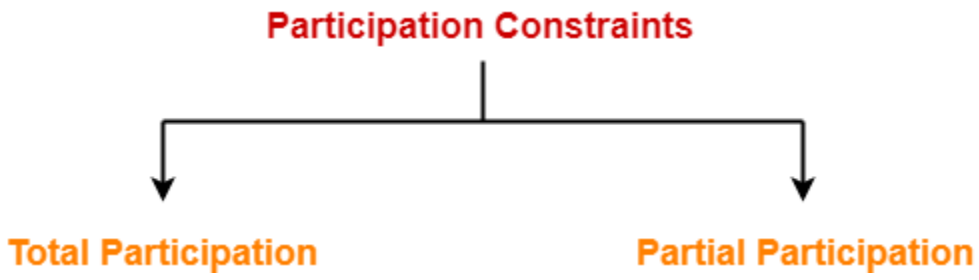
Participation Constraints-

Before you go through this article, make sure that you have gone through the previous article on **Introduction to ER Diagrams**.

Participation constraints define the least number of relationship instances in which an entity must compulsorily participate.

Types of Participation Constraints-

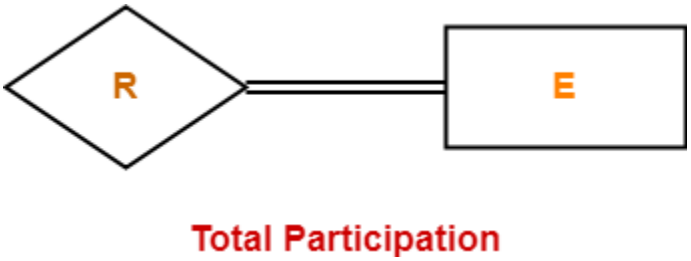
There are two types of participation constraints-



1. Total participation
2. Partial participation

1. Total Participation-

- It specifies that each entity in the entity set must compulsorily participate in at least one relationship instance in that relationship set.
- That is why, it is also called as **mandatory participation.**
- Total participation is represented using a double line between the entity set and relationship set.



Total Participation

Example-

Suppose an entity set Student related to an entity set Course through Enrolled relationship set.

The participation of entity set course in enrolled relationship set is partial because a course may or may not have students enrolled in. It is possible that only some of

the course entities are related to the student entity set through the enrolled relationship set.

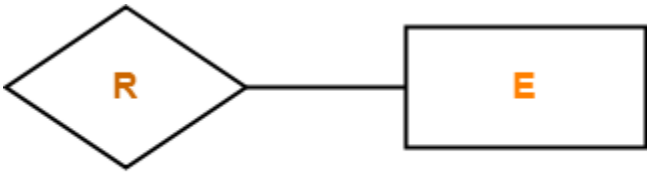
The participation of entity set student in enrolled relationship set is total because every student is expect to relate at least one course through the enrolled relationship set Participation in Enrolled relationship set: Partial Course Total Student

Here,

- Double line between the entity set “Student” and relationship set “Enrolled in” signifies total participation.
- It specifies that each student must be enrolled in at least one course.

2. Partial Participation-

- It specifies that each entity in the entity set may or may not participate in the relationship instance in that relationship set.
- That is why, it is also called as **optional participation.**
- Partial participation is represented using a single line between the entity set and relationship set.



Partial Participation

Example-

Here,

- Single line between the entity set “Course” and relationship set “Enrolled in” signifies partial participation.
- It specifies that there might exist some courses for which no enrollments are made.

Relationship between Cardinality and Participation Constraints-

Minimum cardinality tells whether the participation is partial or total.

- If minimum cardinality = 0, then it signifies partial participation.
- If minimum cardinality = 1, then it signifies total participation.

Maximum cardinality tells the maximum number of entities that participates in a relationship set.

Attributes are the properties which describe an entity.

Example

The attributes of student entity are as follows −

- Roll number
- Name
- Branch
- Age

Types of attributes

The different types of attributes are as follows −

Composite attribute

It can be divided into smaller sub parts, each sub part can form an independent attribute.

For example −

Name

FirstName MiddelName LastName

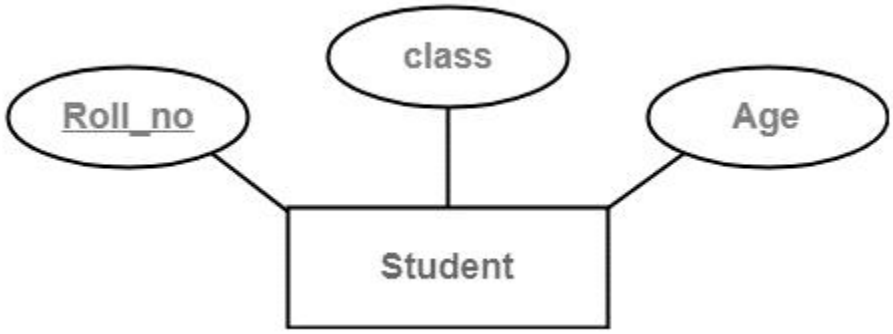
Simple or Atomic attribute

Attributes that cannot be further subdivided are called atomic attributes.

For example −

Phone number

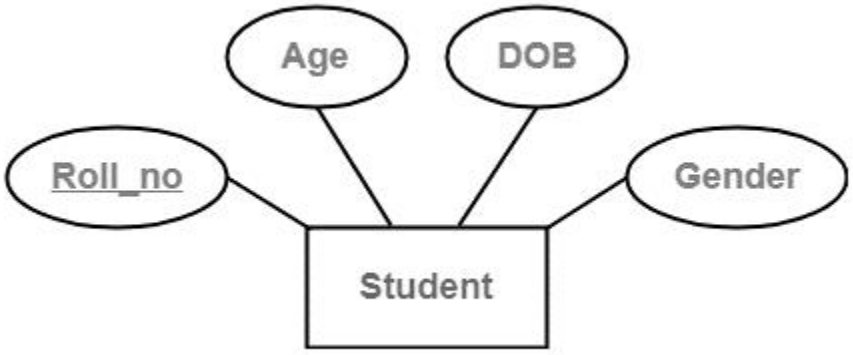
PIN code



Single valued Attribute

Attributes having a single value for a particular item is called a single valued attribute.

For example: Room Number



Multi-valued Attribute

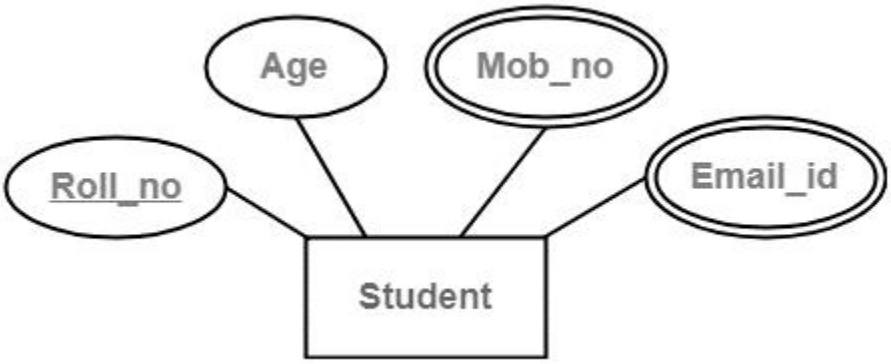
Attribute having a set of values for a single entity is called a multi-valued attribute.

For example −

e-mail

Tel.No

Hobbies

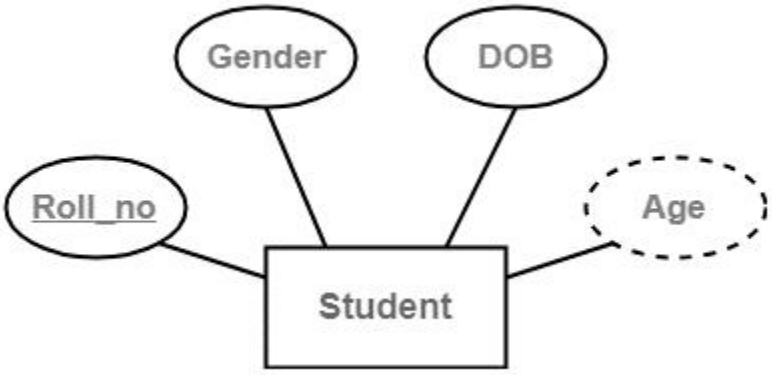


Derived Attributes or stored Attributes

When one attribute value is derived from the other is called a derived attribute.

For example: Age can be derived from date of birth, where,

- Age is the derived attribute.
- DOB is the stored attribute.

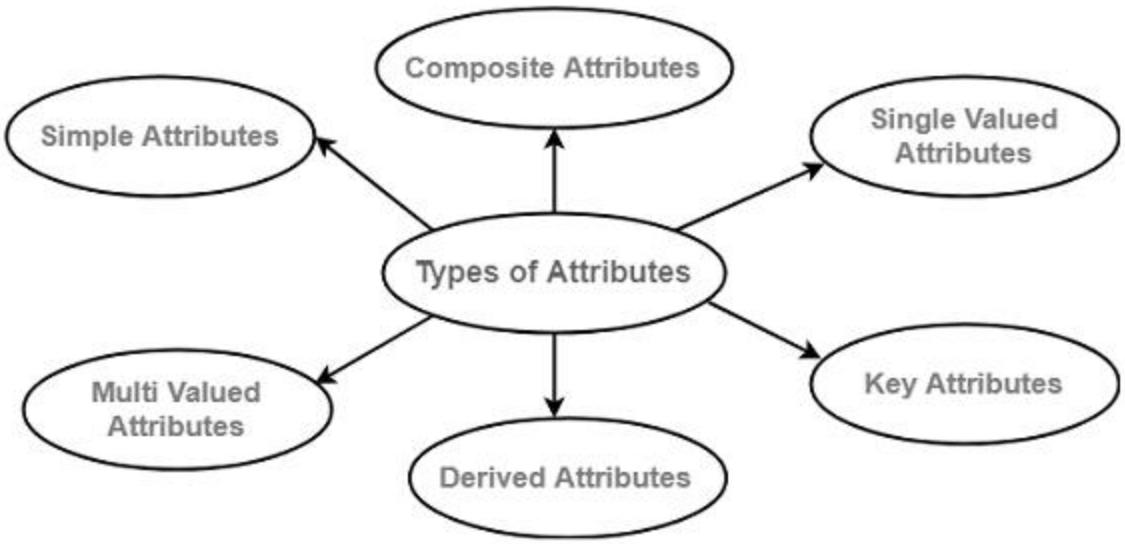


Complex Attribute

Nesting of composite and multi-valued attributes forms a complex attribute.

For example

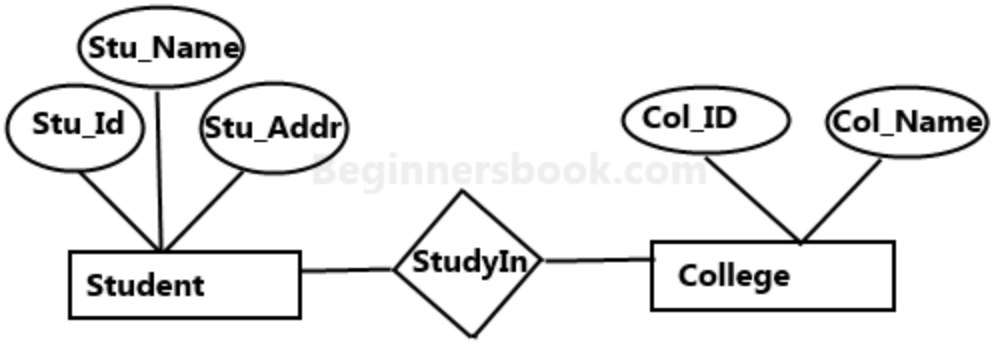
If a person has more than one house and each house has more than one phone. Then, that attribute phone is represented as a complex attribute.



What is an Entity Relationship Diagram (ER Diagram)?

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Lets have a look at a simple ER diagram to understand this concept.

A simple ER Diagram:



Sample E-R Diagram

In the following diagram we have two entities Student and College and their relationship.

The relationship between Student and College is many to one as a college can have many students however a student cannot study in multiple colleges at the same time.

Student entity has attributes such as Stu_Id, Stu_Name & Stu_Addr and College entity has attributes such as Col_ID & Col_Name.

Here are the geometric shapes and their meaning in an E-R Diagram.

We will discuss these terms in detail in the next section(Components of a ER Diagram) of this guide so don't worry too much about these terms now, just go through them once.

Rectangle: Represents Entity sets.
Ellipses: Attributes
Diamonds: Relationship Set
Lines: They link attributes to Entity Sets and Entity sets to Relationship Set
Double Ellipses: Multivalued Attributes
Dashed Ellipses: Derived Attributes
Double Rectangles: Weak Entity Sets
Double Lines: Total participation of an entity in a relationship set

Important Questions:

1. Explain Database Design Phases.
2. Explain Relationship Set in ER Model.
3. Explain Keys.
4. Explain Constraints in Mapping Cardinalities.
5. Explain Participation Constraints.