DBMS | Architectures

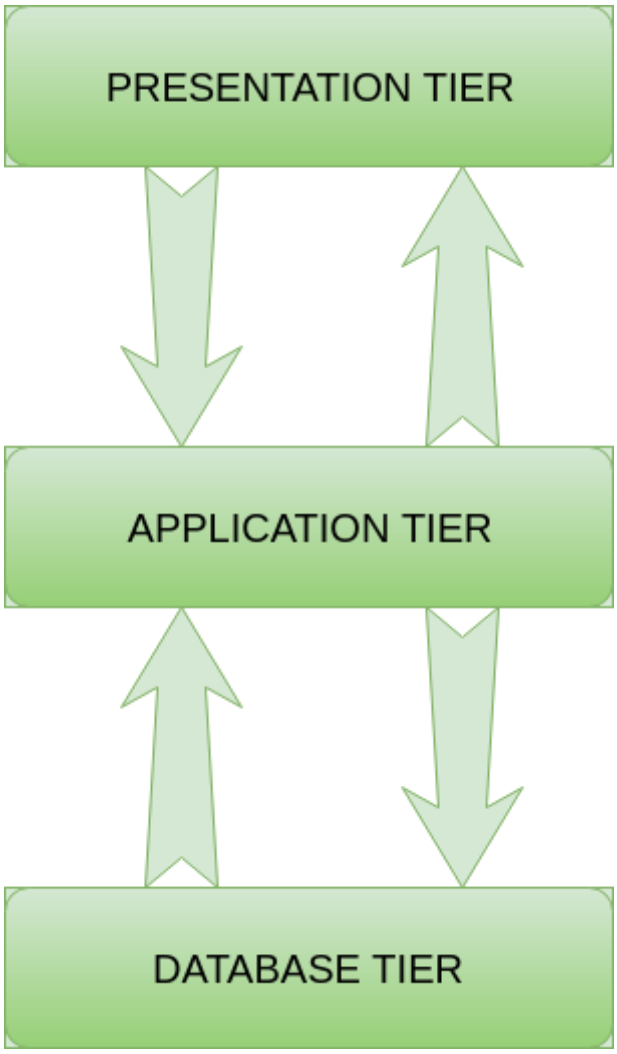
Two Level Architecture

Two level architecture is similar to a basic **client-server** model. The application at the client end directly communicates with the database at the server side. API's like ODBC,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, compatible with existing systems. However, this model gives poor performance when there are a large number of users.

DBMS 3-tier Architecture

DBMS 3-tier architecture divides the complete system into three inter-related but independent modules as shown in Figure below.



Presentation or User Tier: This tier is presented before the user of who knows nothing regarding the other existing complex databases underneath. This tier can provide multiple views of the databases, generated by the application residing in the application tier.

Application Tier: This is the middle lined tier between the Database and the Presentation tier and acts as a connection between the end-user and the database. This tier holds the programs and the application server along with the programs that could access the database. This is the last layer that the users are made aware of. It provides a complete abstraction to the database layer.

Database Tier: This tier holds the primary database along with all the data and the query languages for processing. The relations of data with there constraints are also defined in this level.

Advantages:

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

Disadvantages:
Increased complexity of implementation and communication. It becomes difficult for this sort of interaction to take place due to presence of middle layers.

Data Independence

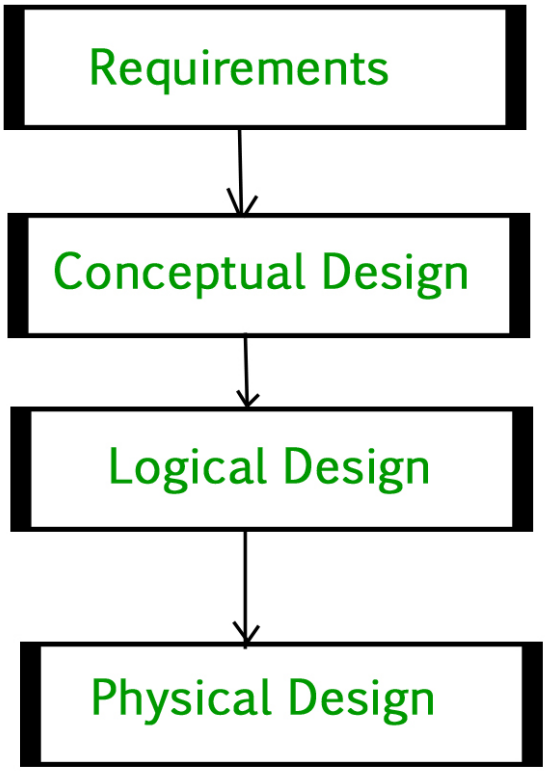
Data independence means a change of data at one level should not affect another level. Two types of data independence are present in this architecture:

Physical Data Independence: Any change in the physical location of tables and indexes should not affect the conceptual level or external view of data. This data independence is easy to achieve and implemented by most of the DBMS.

Conceptual Data Independence: The data at conceptual level schema and external level schema must be independent. This means, change in conceptual schema should not affect external schema. e.g.; Adding or deleting attributes of a table should not affect the user's view of table. But this type of independence is difficult to achieve as compared to physical data independence because the changes in conceptual schema are reflected in user's view.

Phases of database design

Database designing for a real world application starts from capturing the requirements to physical implementation using DBMS software which consists of following steps shown in Figure.



Conceptual Design: The requirements of database are captured using high level conceptual data model. For Example, ER model is used for conceptual design of database.

Logical Design: Logical Design represents data in the form of relational model. ER diagram produced in the conceptual design phase is used to convert the data into the Relational Model.

Physical Design: In physical design, data in relational model is implemented using commercial DBMS like Oracle, DB2.

- DBMS | ER - Model

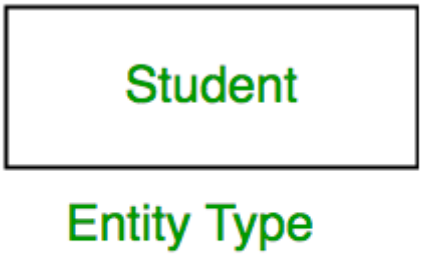
ER Model is used to model the logical view of the system from the data perspective which consists of these components:

Entity, Entity Type, Entity Set

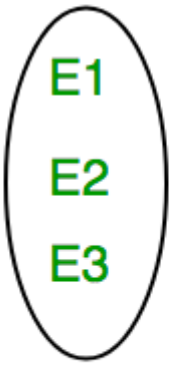
An **Entity** may be an object with a physic ence - a particular person, car, house, or

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

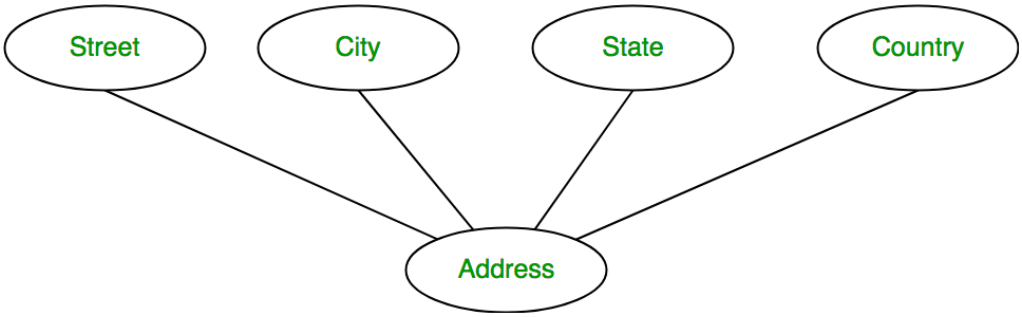


Entity Set

Attribute(s) Attributes are the **properties which define the entity type**. For example, Roll_No, Name, DOB, Age, Address, Mobile_No are the attributes which defines entity type Student. In ER diagram, attribute is represented by an oval.

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

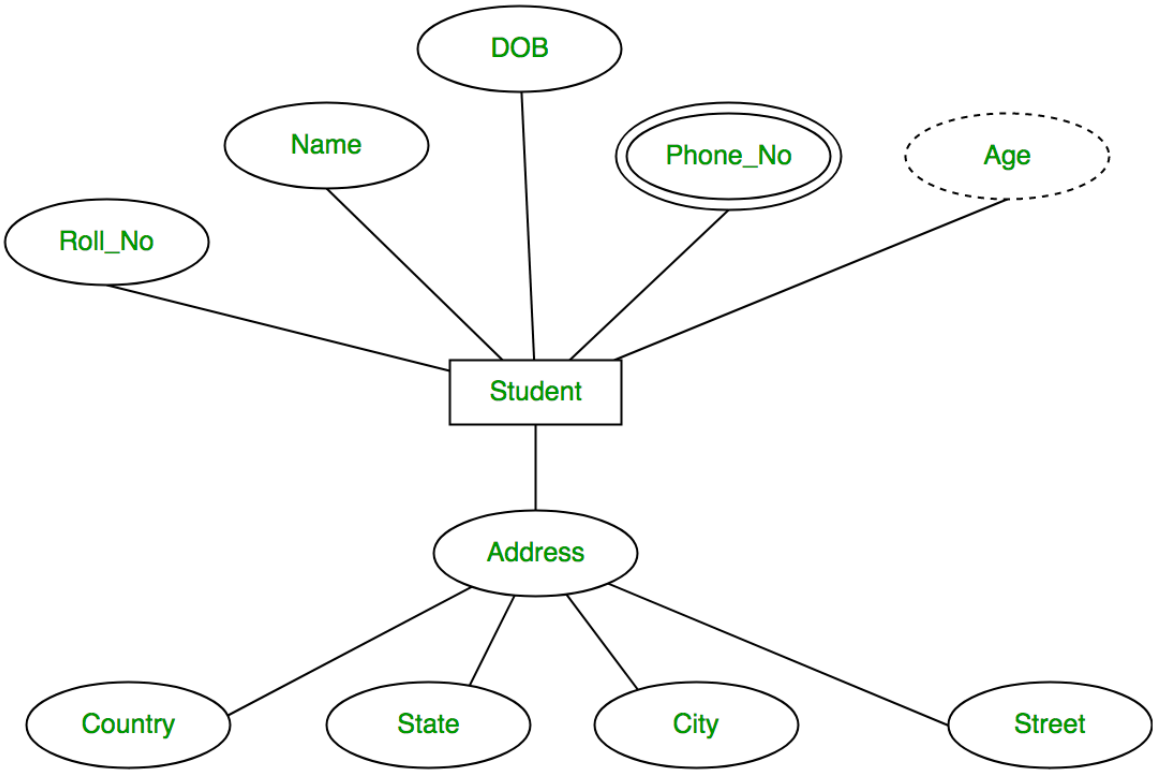
2. **Composite Attribute -** An attribute **composed of many other attribute** is called as composite attribute. For example, Address attribute of student Entity type consists of Street, City, State, and Country. In ER diagram, composite attribute is represented by an oval comprising of ovals.



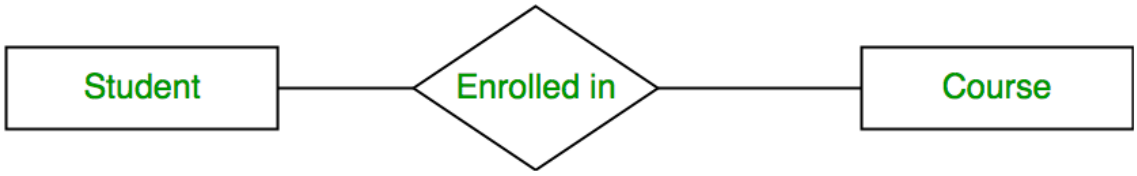
3. **Multivalued Attribute -** An attribute consisting **more than one value** for a given entity. For example, Phone_No (can be more than one for a given student). In ER diagram, multivalued attribute is represented by double oval.

4. **Derived Attribute -** An attribute which can be **derived from other attributes** of the entity type is known as derived attribute. e.g.; Age (can be derived from DOB). In ER diagram, derived attribute is represented by dashed oval.

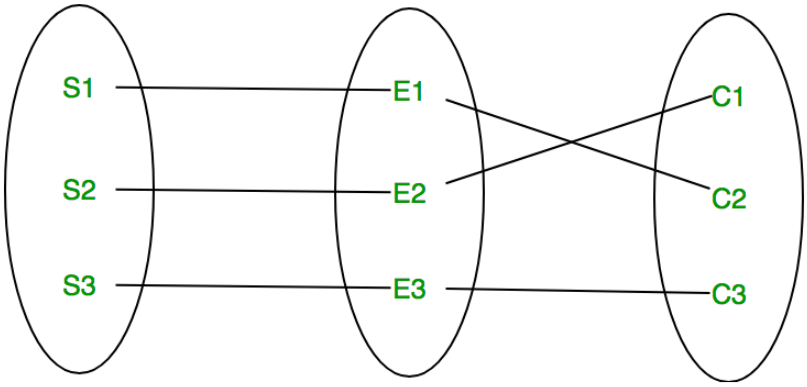
The complete entity type **Student** with its attributes can be represented as:



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, relationship type is represented by a diamond and connecting the entities with lines.

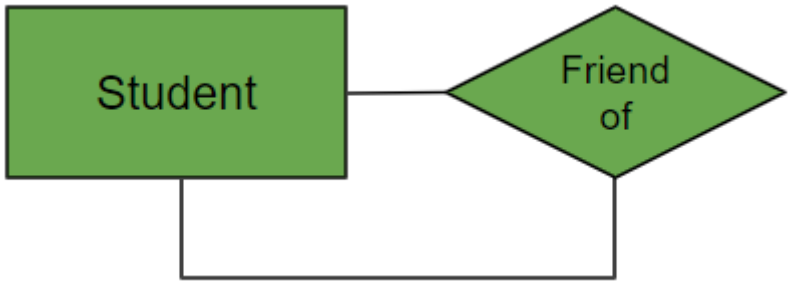


A set of relationships of the same type is known as a relationship set. The following relationship set depicts S1 is enrolled in C2, S2 is enrolled in C1 and S3 is enrolled in C3.

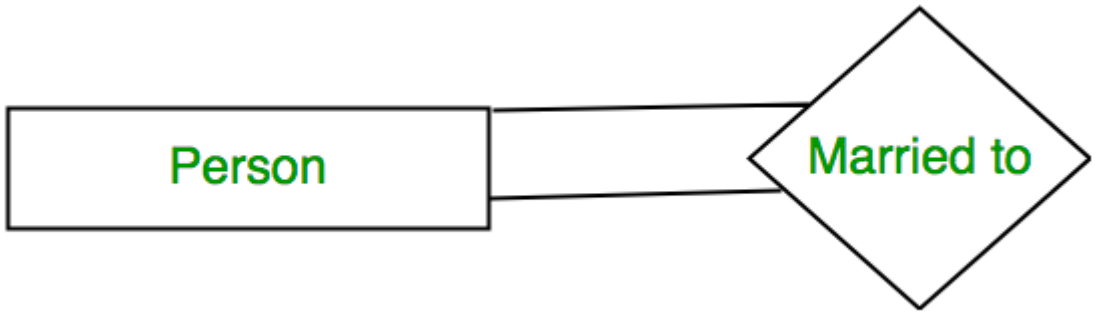


Degree of a relationship set: The number of different entity sets **participating in a relationship** set is called as degree of a relationship set.

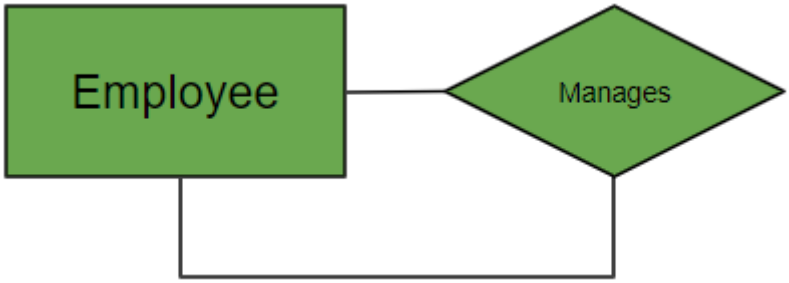
1. **Unary Relationship -** When there is **only ONE entity set participating in a relation**, the relationship is called as unary relationship.
   Example 1: A student is a friend of itself.



Example 2: A person is married to a person.

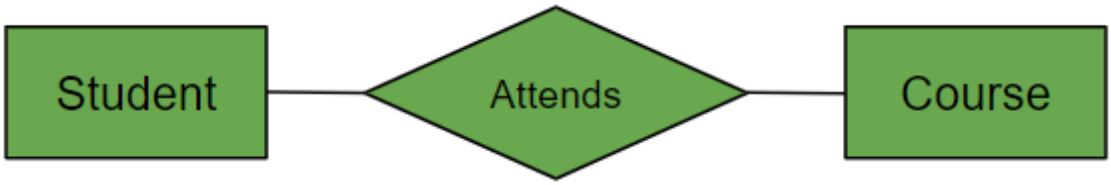


Example 3: An employee manages an employee.



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

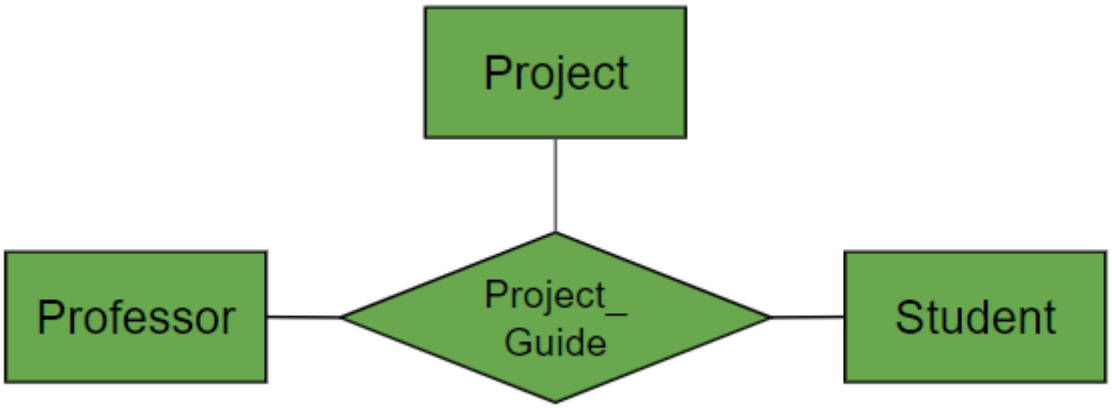
   Example 1: A student attends a course.



   Example 2: A supplier supplies item.

   Example 3: A Profssor teaches subject.

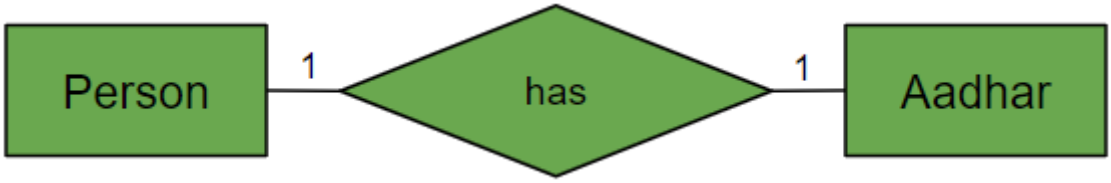
3. **n-ary Relationship -** When there are n entities set participating in a relation, the relationship is called as n-ary relationship.
   Example: A Professor, student and Project is related to a Project_Guide.



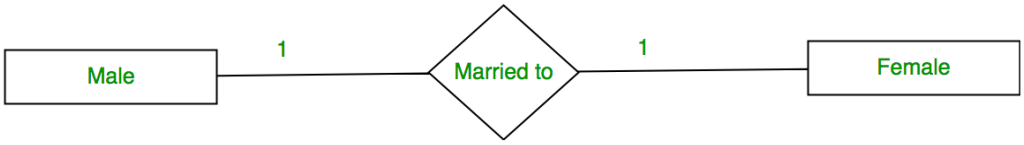
Cardinality The **number of times an entity of an entity set participates in a relationship** set is known as cardinality. Cardinality can be of different types:

1. **One to One -** When each entity in each entity set can take part **only once in the relationship**, the cardinality is one to one.
   Example 1: Let us assume that a driver can drive one carand a cat can be driven by the same driver. So the relationship will be one to one.

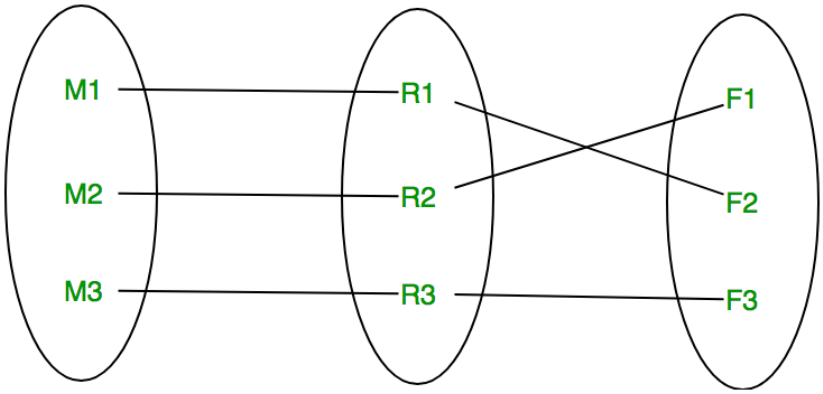
   Example 2: A person can have only one Aadhar card and one Aadhar card can belong to only one person.



   Example 3: Let us assume that a male can marry to one female and a female can marry to one male. So the relationship will be one to one.



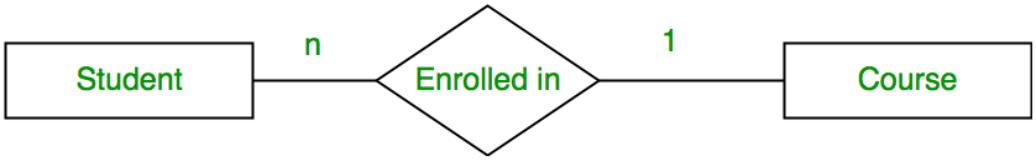
Using Sets, it can be represented as:



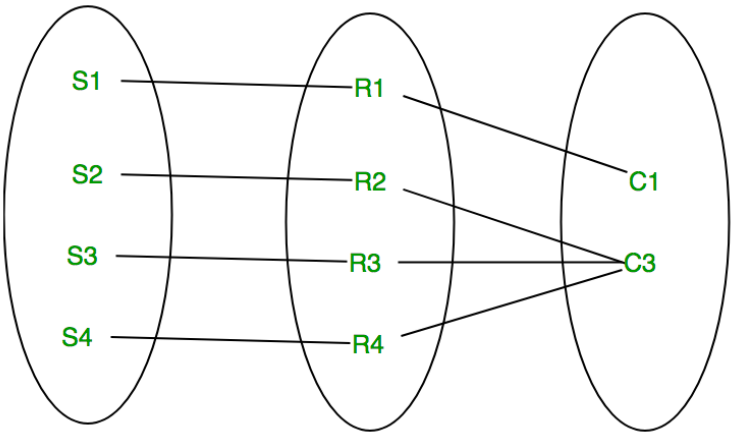
2. **One to Many -** When entities in one entity set **can take part only once in the relationship set and entities in other entity sets can take part more than once in the relationship set,** cardinalities is one to many. Many to one also come under this category.
   Example 1: A professor teaching many courses

Example 2: Many employees working for one department.

Example 3: 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.



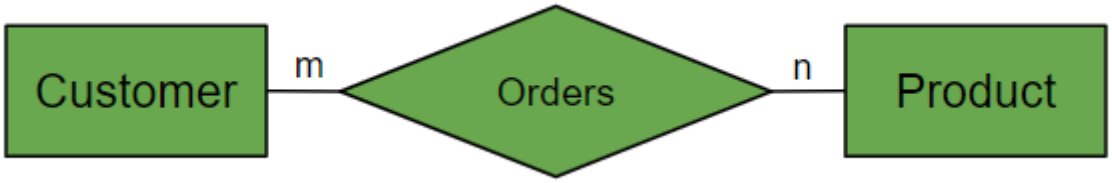
Using Sets, it can be represented as:



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

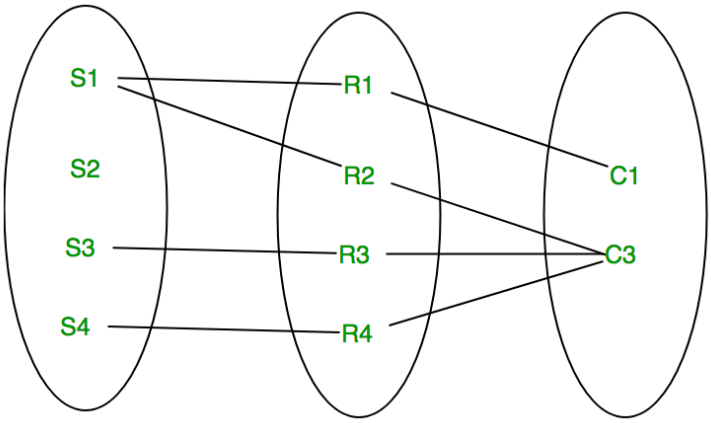
3. **Many to many -** When entities in all entity sets can **take part more than once in the relationship** cardinality is many to many.
   Example 1: Any number of student can take any number of subjects.

Example 2: Any number of customer can order any number of products.



Example 3: 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.

Using sets, it can be represented as:



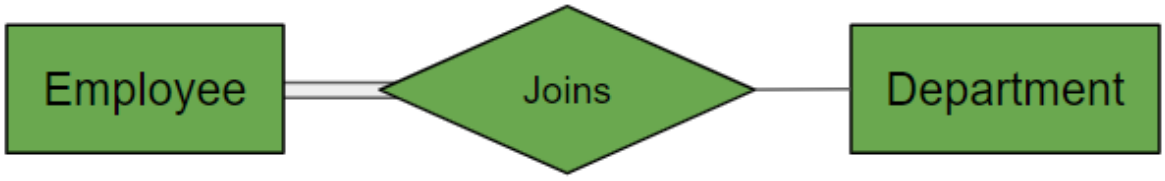
In this example, student S1 is enrolled in C1 and C3 and Course C3 is enrolled by S1, S3, and S4. So it is many to many relationships.

Participation Constraint: Participation Constraint is applied on the entity participating in the relationship set.

1. **Total Participation -** Each entity in the entity set **must participate** in the relationship.

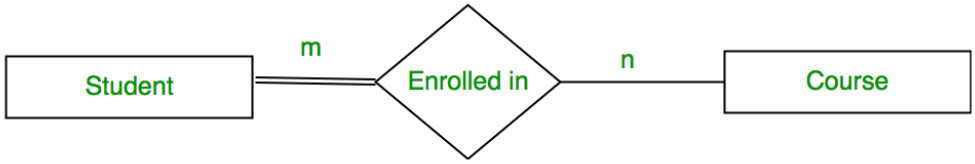
   Example 1: If each student must attend a course, the participation of student will be total. Total participation is shown by double line in ER diagram.

   Example 2: Each employee must join a department.

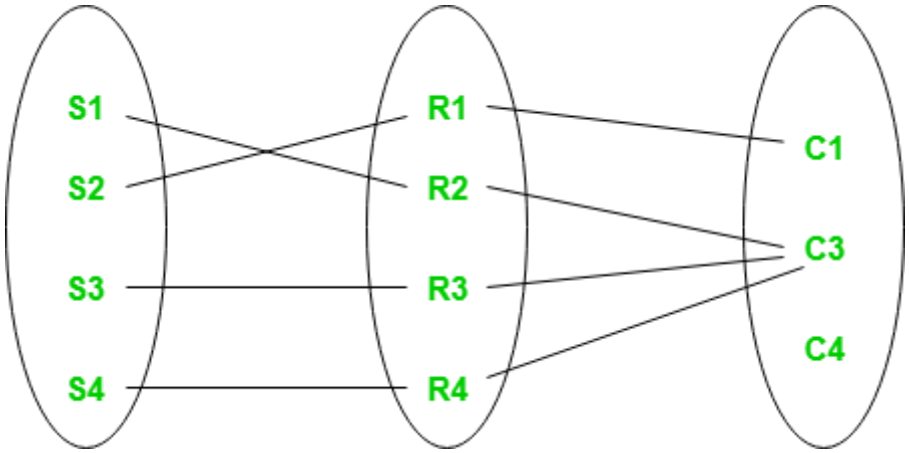


2. **Partial Participation -** The entity in the entity set **may or may NOT participat**e in the relationship. If some courses are not enrolled by any of the student, the participation of course will be partial.

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



Using set, it can be represented as,

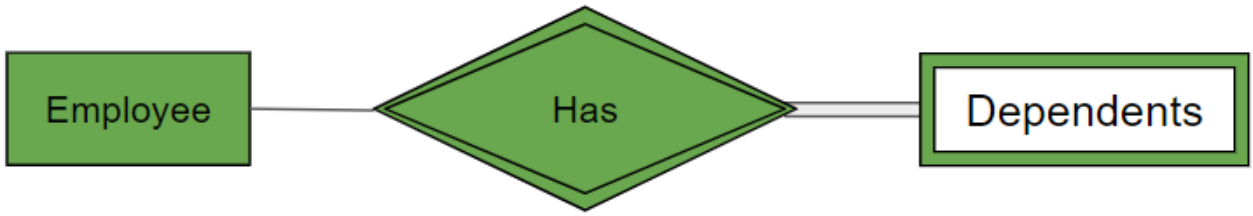


Every student in Student Entity set is participating in a relationship but there exists a course C4 which is not taking part in the relationship.

Weak Entity Type and Identifying Relationship As discussed before, an entity type has a key attribute that uniquely identifies each entity in the entity set. But there exists **some entity type for which key attribute can't be defined**. These are called the Weak Entity type. A weak entity type is represented by a double rectangle. The participation of a weak entity type is always total. The relationship between a weak entity type and its identifying strong entity type is called an identifying relationship and it is represented by a double diamond.

Example 1: a school might have multiple classes and each class might have multiple sections. The section cannot be identified uniquely and hence they do not have any primary key. Though a class can be identified uniquely and the combination of a class and section is required to identify each section uniquely. Therefore the section is a weak entity and it shares total participation with the class.

Example 2: A company may store the information of dependants (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 the Employee will be Identifying Entity type for Dependant.



Example 3: In case of multiple hosts, the login id cannot become primary key as it is dependent upon the hosts for its identity.

DBMS | Relational Model

Relational Model was proposed by E.F. Codd to model data in the form of relations or tables. After designing the conceptual model of Database using the ER diagram, we need to convert the conceptual model in the relational model which can be implemented using any RDBMS languages like Oracle SQL, MySQL, etc. So we will see what the Relational Model is.

What is Relational Model?

Relational Model represents how data is stored in Relational Databases. A relational database stores data in the form of relations (tables). Consider a relation STUDENT with attributes ROLL_NO, NAME, ADDRESS, PHONE and AGE shown in Table 1.

STUDENT

ROLL_NO	NAME	ADDRESS	PHONE	AGE
1	RAM	DELHI	9455123451	18
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	DELHI		18

IMPORTAN[illegible]MINOLOGIES

1. **Attribute:** Attributes are the properties that define a relation. e.g.; **ROLL_NO**, **NAME**
2. **Relation Schema:** A relation schema represents name of the relation with its attributes. e.g.; STUDENT (ROLL_NO, NAME, ADDRESS, PHONE and AGE) is relation schema for STUDENT. If a schema has more than 1 relation, it is called Relational Schema.
3. **Tuple:** Each row in the relation is known as tuple. The above relation contains 4 tuples, one of which is shown as:

1	RAM	DELHI	9455123451	18

4. **Relation Instance:** The set of tuples of a relation at a particular instance of time is called as relation instance. Table 1 shows the relation instance of STUDENT at a particular time. It can change whenever there is insertion, deletion or updation in the database.
5. **Degree:** The number of attributes in the relation is known as degree of the relation. The **STUDENT** relation defined above has degree 5.
6. **Cardinality:** The number of tuples in a relation is known as cardinality. The **STUDENT** relation defined above has cardinality 4.
7. **Column:** Column represents the set of values for a particular attribute. The column **ROLL_NO** is extracted from relation STUDENT.

ROLL_NO
1
2
3
4

8. **NULL Values:** The value which is not known or unavailable is called NULL value. It is represented by blank space. e.g.; PHONE of STUDENT having ROLL_NO 4 is NULL.

Constraints in Relational Model

While designing Relational Model, we define some conditions which must hold for data present in database are called Constraints. These constraints are checked before performing any operation (insertion, deletion and updation) in database. If there is a violation in any of constrains, operation will fail.

Domain Constraints: These are attribute level constraints. An attribute can only take values which lie inside the domain range. e.g; If a constrains AGE>0 is applied on STUDENT relation, inserting a negative value of AGE will result in failure.

Key Integrity: Every relation in the database should have at least one set of attributes that defines a tuple uniquely. Those set of attributes is called key. e.g.; ROLL_NO in

STUDENT is a key. No two students can have the same roll number. So a key has two properties:

- It should be unique for all tuples.
- It can't have NULL values.

Referential Integrity: When one attribute of a relation can only take values from other attribute of same relation or any other relation, it is called referential integrity. Let us suppose we have 2 relations

STUDENT

ROLL_NO	NAME	ADDRESS	PHONE	AGE	BRANCH_CODE
1	RAM	DELHI	9455123451	18	CS
2	RAMESH	GURGAON	9652431543	18	CS
3	SUJIT	ROHTAK	9156253131	20	ECE
4	SURESH	DELHI		18	IT

BRANCH

BRANCH_CODE	BRANCH_NAME
CS	COMPUTER SCIENCE
IT	INFORMATION TECHNOLOGY
ECE	ELECTRONICS AND COMMUNICATION ENGINEERING
CV	CIVIL ENGINEERING

BRANCH_CODE of STUDENT can only take the values which are present in BRANCH_CODE of BRANCH which is called referential integrity constraint. The relation which is referencing to other relation is called REFERENCING RELATION (STUDENT in this case) and the relation to which other relations refer is called REFERENCED RELATION (BRANCH in this case).

ANOMALIES

An anomaly is an irregularity or something which deviates from the expected or normal state. When designing databases, we identify three types of anomalies: Insert, Update and Delete.

Insertion Anomaly in Referencing Relation:
We can't insert a row in REFERENCING RELATION if the referencing attribute's value is not present in the referenced attribute value. e.g.; Insertion of a student with BRANCH_CODE 'ME' in STUDENT relation will result in an error because 'ME' is not present in BRANCH_CODE of BRANCH.

Deletion/ Updation Anomaly in Referenced Relation:
We can't delete or update a row from REFERENCED RELATION if the value of REFERENCED ATTRIBUTE is used in the value of REFERENCING ATTRIBUTE. e.g; if we try to delete tuple from BRANCH having BRANCH_CODE 'CS', it will result in error because 'CS' is referenced by BRANCH_CODE of STUDENT, but if we try to delete the row from BRANCH with BRANCH_COD will be deleted as the value is not been

used by referencing relation. It can be handled by the following method:

ON DELETE CASCADE: It will delete the tuples from REFERENCING RELATION if a value used by REFERENCING ATTRIBUTE is deleted from REFERENCED RELATION. e.g; if we delete a row from BRANCH with BRANCH_CODE 'CS', the rows in STUDENT relation with BRANCH_CODE CS (ROLL_NO 1 and 2 in this case) will be deleted.

ON UPDATE CASCADE: It will update the REFERENCING ATTRIBUTE in REFERENCING RELATION if attribute value used by REFERENCING ATTRIBUTE is updated in REFERENCED RELATION. Ex: if we update a row from BRANCH with BRANCH_CODE 'CS' to 'CSE', the rows in STUDENT relation with BRANCH_CODE CS (ROLL_NO 1 and 2 in this case) will be updated with BRANCH_CODE 'CSE'.

SUPER KEYS: Any set of attributes that allows us to identify unique rows (tuples) in a given relationship are known as superkeys. Out of these super keys, we can always a proper subset that can be used as a primary key. Such keys are known as Candidate keys. If there is a combination of two or more attributes that are being used as the primary key then we call it as a Composite key.

- DBMS | Keys in Relational Model

We have considered a Student and Course Relational Model for our Reference Examples.

Example Student Course Relational Model

STUDENT

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	Rajsthan	India	18
4	SURESH		Punjab	India	21

Table 1

STUDENT_COURSE

STUD_NO	COURSE_NO	COURSE_NAME
1	C1	DBMS
2	C2	Computer Networks
1	C2	Computer Networks

Table 2

Candidate Key: The minimal set of attribute which can uniquely identify a tuple is known as candidate key. For **Example**, STUD_NO in STUDENT relation.

- The value of Candidate Key is unique and non-null for every tuple.
- There can be more than one candidate key in a relation. For Example, STUD_NO as well as STUD_PHONE both are candidate keys for relation STUDENT.
- The candidate key can be simple (having only one attribute) or composite as well. For **Example**, {STUD_NO, COURSE_NO} is a composite candidate key for relation STUDENT_COURSE.

Note - In Sql Server a unique constraint that has a nullable column, **allows** the value '**null**' in that column **only once**. That's why STUD_PHONE attribute as candidate here, but can not be 'null' values in primary key attribute.

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.

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

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 both, are candidate keys for relation STUDENT but STUD_NO can be chosen as the primary key (only one out of many candidate keys).

Alternate Key: The candidate key other than the primary key is called an alternate key. For **Example**, STUD_NO, as well as STUD_PHONE both, are candidate keys for relation STUDENT but STUD_PHONE will be alternate key (only one out of many candidate keys).

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 and the relation which refers to the referenced relation is called referencing relation and the corresponding attribute is called referencing attribute. A referenced attribute of the referenced relation should be the primary key for it. For **Example**, STUD_NO in STUDENT_COURSE is a foreign key to STUD_NO in STUDENT relation.

It may be worth noting that unlike, Primary Key of any given relation, Foreign Key can be NULL as well as may contain duplicate tuples i.e. it need not follow uniqueness constraint.

For **Example**, STUD_NO in STUDENT_COURSE relation is not unique. It has been repeated for the first and third tuple. However, the STUD_NO in STUDENT relation is a primary key and it needs to be always unique and it cannot be null.

Report An Issue

If you are facing any issue on this page. Please let us know.

LIVE BATCHES

GeeksforGeeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

Company

About Us

Careers

Privacy Policy

Contact Us

Terms of Service

Learn

Algorithms

Data Structures

Languages

CS Subjects

Video Tutorials

Practice

Courses

Company-wise

Topic-wise

How to begin?

Contribute

Write an Article

Write Interview Experience

Internships

Videos

▲