UNIT-2

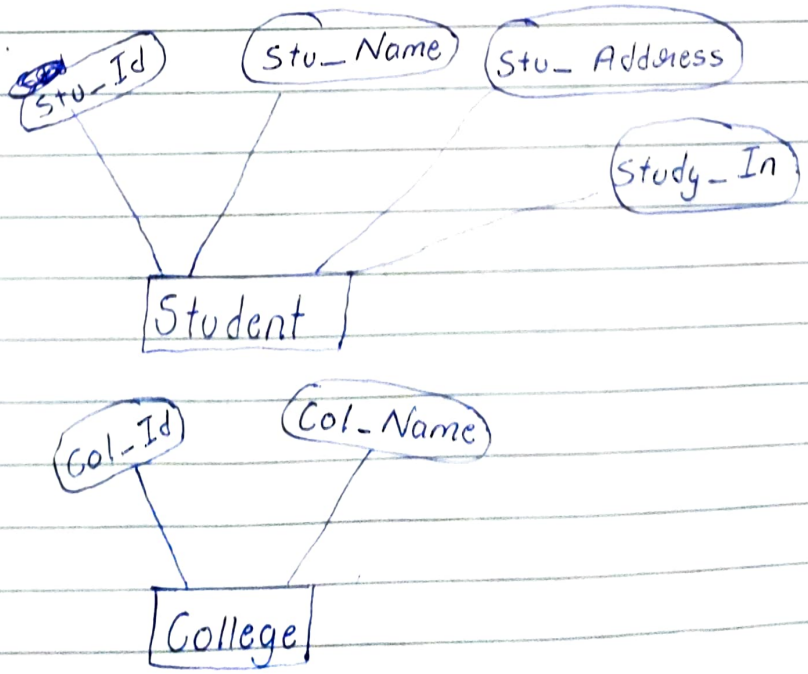
ER DIAGRAM
[Entity Relationship Model]

→ It describes the structure of Database with the help of a diagram, which is called as E-R diagram.

→ E-R model is a design or blue-print of Database that can be later implemented as Database.

→ Main Components of E-R model are -
- Entity set
- Relationship set [more than one group]

→ An entity is a set of a group of similar entity & having attributes.



ER Diagram

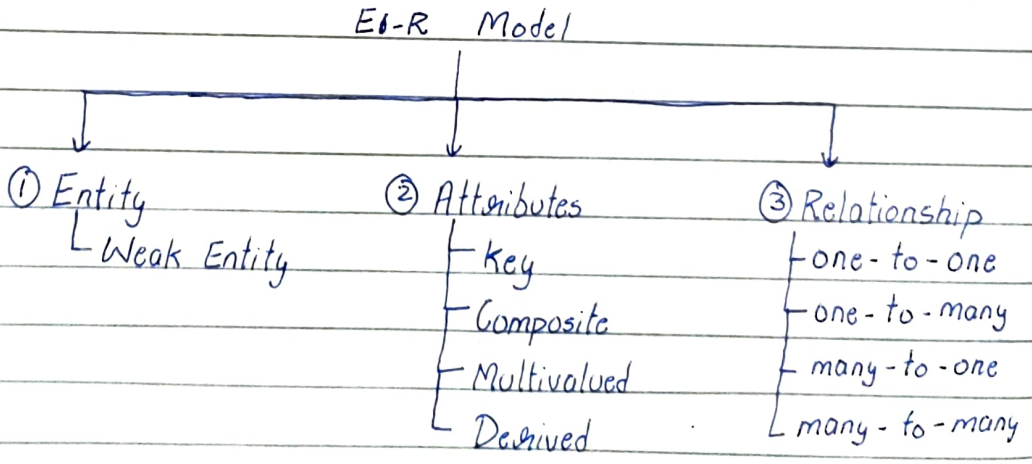
→ We have 2 entities
- Student
- College

→ Relationship between student & college is many to one.

Student entity have attributes
- Stu_Id
- Stu_Name
- Stu_Address

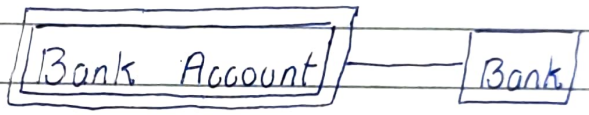
College entity have attributes
- Col_Id
- Col_Name

Components of E-R Model



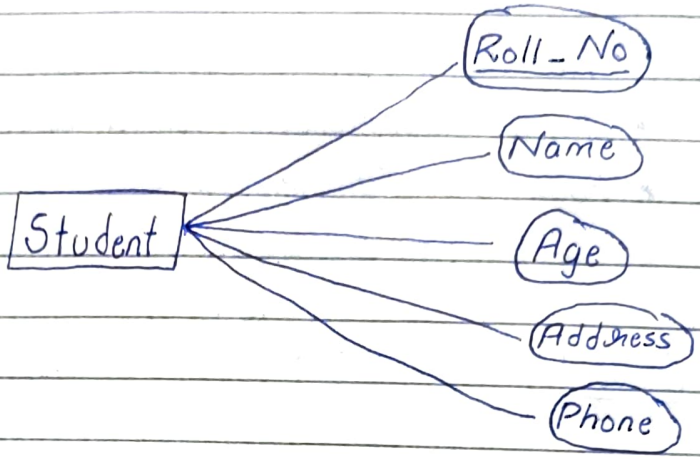
① Entity → * An entity is an object or component of data.
* It is represented by a Rectangle.
* For Ex – Two entities → Student
→ College

→ Weak Entity → An entity that can't be uniquely identified by its own attributes & relies on the relationship with other entity is called as weak entity.
It is represented by Double Rectangle.

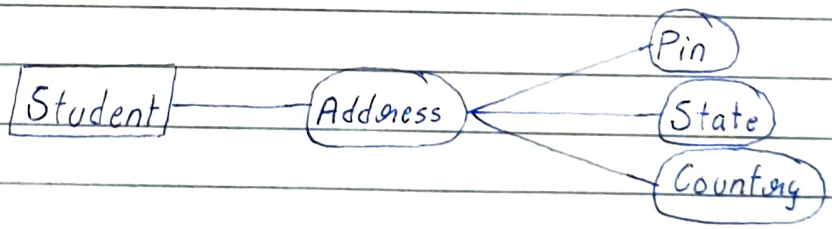


② Attributes → * It describes the property of an entity.
* It is represented by oval shape.
* 4 types of attributes → Key
Composite
Multivalued
Derived

A. Key Attribute → It can uniquely identify an entity from the entity set.
* Student Roll Number can uniquely identify a student from a set of students.
* Text of key attributes is underlined.

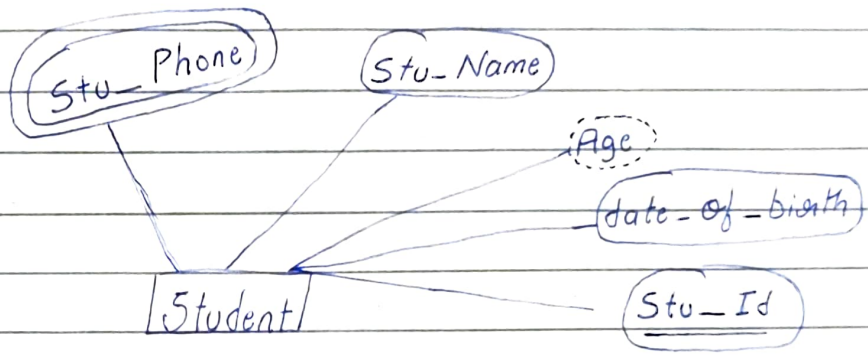


B. Composite Attribute → An attribute that is a combination of other attributes is known as Composite Attribute. For ex → In student entity, the student address is a composite attribute, as an address is composed of other attributes like pin code, state, country.



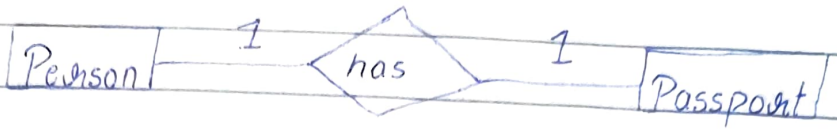
C. Multivalued Attribute → An attribute that can hold multiple values is known as multivalued attribute. It is represented with Double Ovals in a E-R diagram. For ex → a person can have more than one phone no, therefore, the phone no attribute is multivalued.

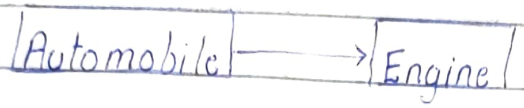
D. Derived Attribute → A derived attribute is one whose value is dynamic and derived from another attribute. It is represented by Dashed Oval in an E-R diagram. For ex → person's age is a derived attribute because it changes over time and can be derived from another attribute [DOB].

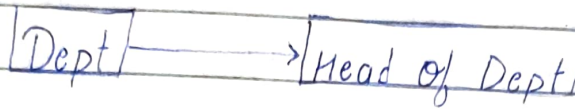


(3) Relationship → A relationship is represented by diamond shape in E-R diagram. It shows the relationship between different entities. There are basically 4 types of relationship →
- one-to-one
- one-to-many
- many-to-one
- many-to-many

A. One-to-One → When a single instance of an entity is associated with a single instance of another entity then it is called one-to-one Relationship. For ex → a person has only one passport and a single passport is given to one person only.

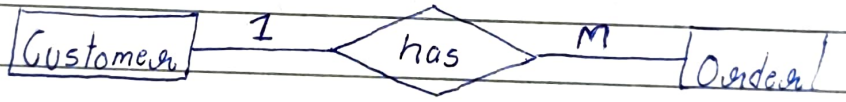
(a) 

(b) 

(c) 

B. One-to-Many → When a single instance of an entity is associated with more than one instances of another entity, then it is called as One-to-Many Relationship.

* For Ex → A customer can place many orders but a order can't be placed by many customers.

(a) 

(b) 

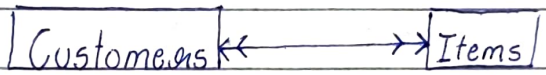
(c) 

C. Many-to-One → When more than one instances of an entity is associated with a single instance of another entity, then it is called as Many-to-One Relationship.

* For ex → Many students can study in a single college but a student can't study in many college at the same time.

D. Many - to - Many → When more than one instances of an entity is associated with more than one instance of another entity, then it is called as Many - to - Many Relationship.

* For ex → A student can be assigned too many projects and a project can be assigned to many students.

(a) 

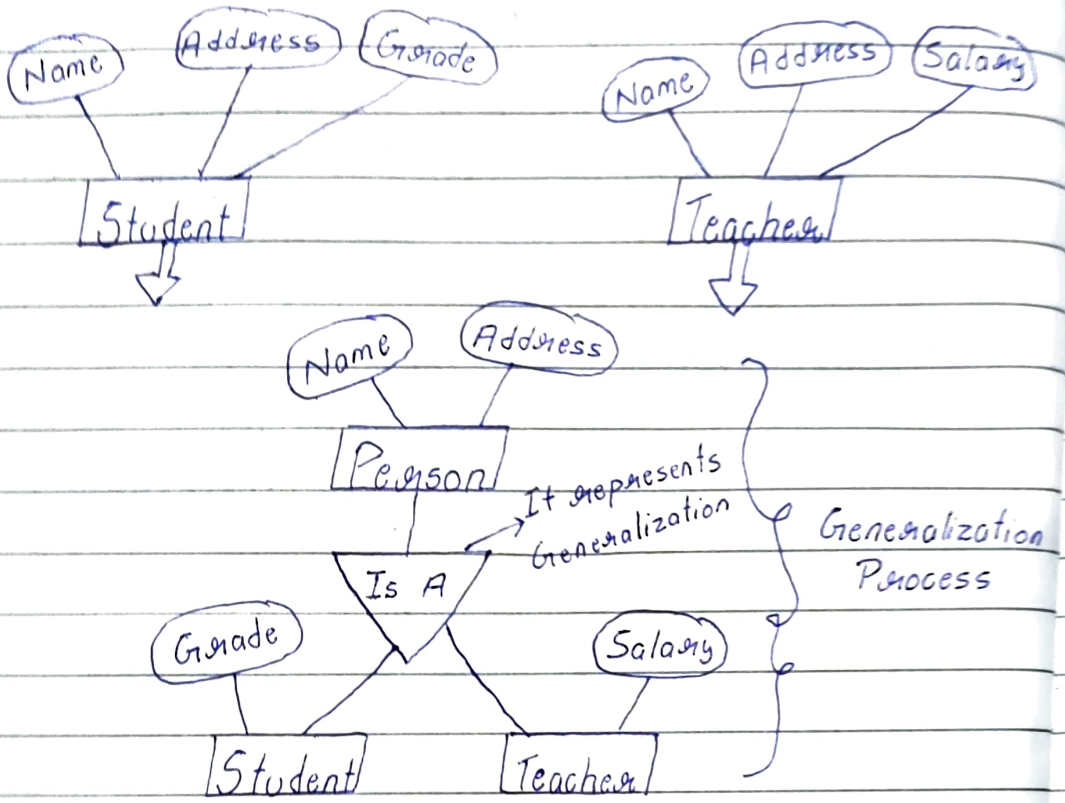
GENERALIZATION

It is a process in which the common attributes of more than one entities form a new entity. This newly formed entity is called as Generalized entity.

* For ex ↓
Let say we have 2 entities → Student, Teacher

Attributes of Student → Name, Address, Grade

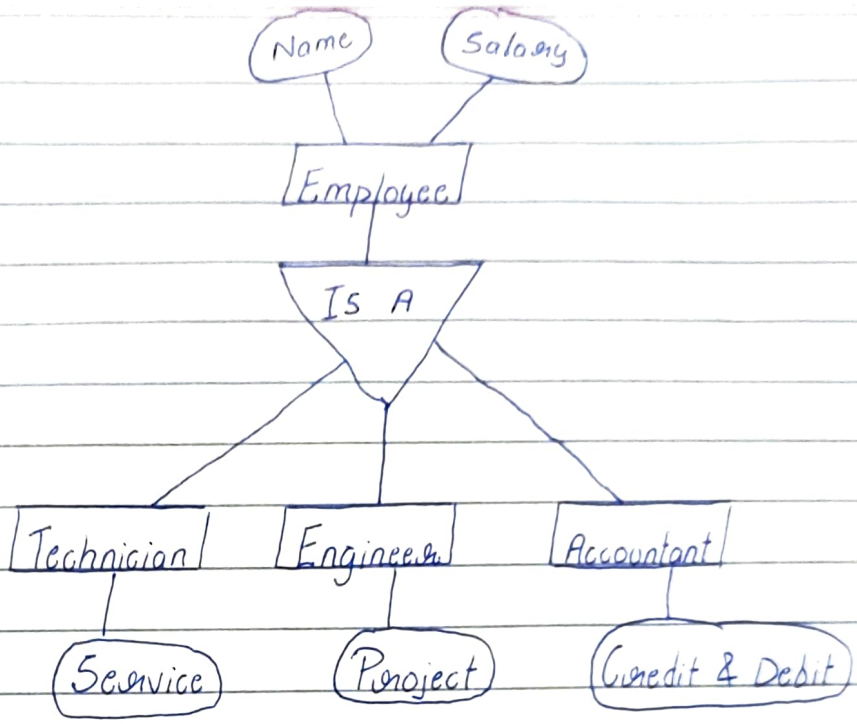
Attributes of Teacher → Name, Address, Salary



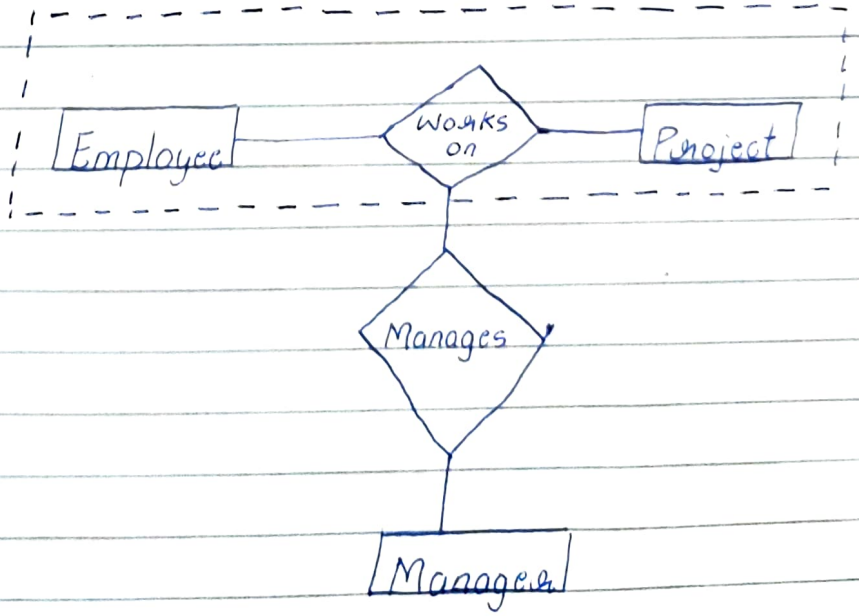
* It is depicted through a triangle component labelled "Is A".

SPECIALIZATION

* It is a process in which an entity is divided into sub-entities.
* It is a reverse process of generalization.
* It finds the subset of entities that have few distinguish attributes.
* For ex - Entity - Employee can be further classified as
  - Technician [Handle Service Request]
  - Engineer [Works on Project]
  - Accountant [Handle Credit & Debit Details]



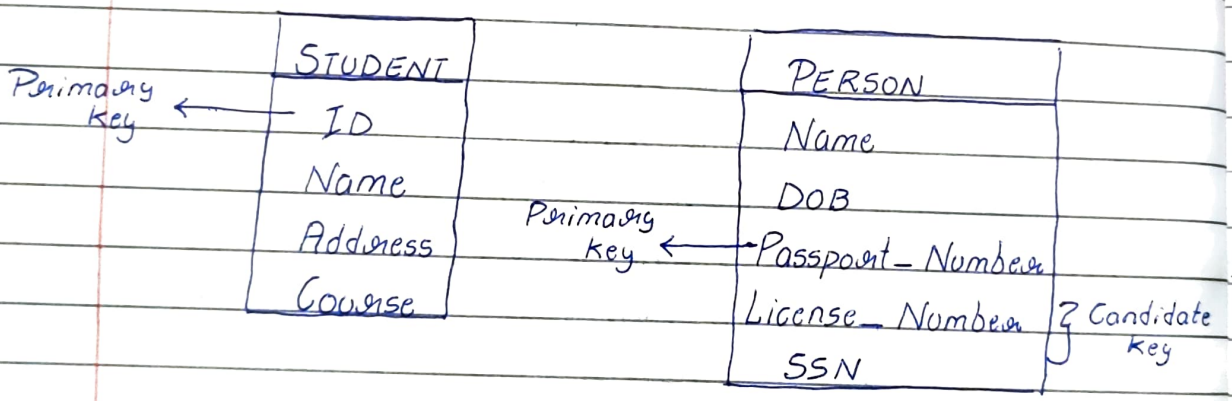
AGGREGRATION → It is a process in which a single entity alone is not able to make sense in a relationship. So, the relationship of two entities acts as one entity



KEYS → A Key is that data item that exclusively identifies a record.

* It is of 4 types →
  - Primary Key
  - Candidate Key
  - Super Key
  - Foreign Key

① Primary Key →



(1) PRIMARY KEY → * It is the first key which is used to identify one & only one instance of an entity uniquely. An entity can contain multiple keys as we have seen in PERSON table. The key which is most suitable from those list becomes a primary key.

* In the EMPLOYEE table, ID can be primary key since it is unique for each employee. In the EMPLOYEE table, we can even select licence_number, passport_number as primary key because they are also unique.

(2) CANDIDATE KEY → * A Candidate key is an attribute or set of an attributes which can uniquely identifies a tuple [row].

* The remaining attributes except for primary key are considered as a candidate key. The Candidate key are as strong as primary key.

(3) SUPER KEY → Super key is a set of attributes which can uniquely identify a tuple. Super key is a super set of a candidate key.

(4) FOREIGN KEY → FOREIGN KEY ARE THE COLUMN OF THE TABLE WHICH IS USED TO POINT TO THE primary key of another table.

Page No.
Date: / /

RELATIONAL ALGEBRA (R.A)

* Relational Algebra uses a procedural query language which is a collection of operations to manipulate relations.

* It consist of a set of such operations that take one or more relations as input & produce a new relation as their result.

Relational Algebra Operations →
- Select (σ)
- Project (π)
- Join
- Union
- Difference
- Intersection, etc.

① Select (σ)

* The Select operation extracts specific tuples from a relations.
* Sigma (σ) is a Greek Letter.

DEPOSIT RELATION

Branch-name	Account-Number	Customer-Name	Balance
Triveni Nagar	0001123	Gopal	5,000
Malviya Nagar	001445	Divya	10,000
Malviya Nagar	001566	Isha	2,000
Aadarsh Nagar	007733	Akshita	4,000
Triveni Nagar	001129	Yash	8,000

* For ex- We want to find tuples in which the balance amount is more than 7000.

$$\sigma_{Balance > 7000}(Deposit)$$

Relational Algebra Operations
1. Select
2. Project
3. Set Union

② Project (π) → It is a unary operation which extracts attributes from a relation.

* Denoted by Greek letter pi (π)
* For ex- Suppose we want to know the customer names and their bank balance amounts from deposit relation

DEPOSIT RELATION

Branch_name	Account_no	Customer_name	Balance
Triveni Nagar	0001123	Gopal	5,000
Malviya Nagar	001445	Divya	10,000
Malviya Nagar	001566	Isha	2,000
Aadarsh Nagar	007733	Akshita	4,000
Triveni Nagar	001129	Yash	8,000

$$\pi_{Customer\_name,\ Balance}(Deposit)$$

③ Set Union

* The result of union operation is denoted by the symbol (∪)
* Four ex → X∪Y is a relation that includes all the tuples are either in X or in Y or in both X & Y.

Resulted Relation ←

* Duplicate tuples will be eliminated, by using set union.
* X & Y are two Relations.
* X holds details of employee working on project J_1.
* Y holds employee working on project J_2.

X

Emp_No	Emp_Name
101	Rita
103	Ritu
104	Seta
106	Leela
107	Neena

Y

Emp_No	Emp_Name
101	Rita
105	Isha
107	Neena
112	Seema

Ex → To select all those employees who are working on either in Project J_1 & J_2 or in both projects :-

The results will be :- X∪Y

Emp_No	Emp_Name
101	Rita
103	Ritu
104	Seta
105	Isha
106	Leela
107	Neena
112	Seema

X∪Y →

(4) Set Intersection [used to find common records]

* It is denoted by $x \cap y$.
* It includes all types that are in both x and y relation.
* For ex:- To select those employees who are exclusively working on both projects J_1 and J_2.

Output → $x \cap y$ →

Emp-No	Emp-Name
101	Rita
107	Neena

(5) Set Difference

* Denoted by x-y
* It allows us to find tuples that are in one relation but are not in another relation.
* For Ex- To select all those employees who are exclusively working on project J_1 and J_2, is given by x-y.

Output → x-y →:

Emp-No	Emp-Name
103	Ritu
104	Setu
106	Lelela

(6) Cartesian Product (Cross Product)

multiply ↓

* Denoted by $X \times Y$ and returns a relation on tuples whose schema contains all fields of x [in the same

order they appear in X] followed by all fields of Y [in the same order they appear in Y].

* The result of X×Y contains one tuple (r,s) [the concatenation of tuples r and s] for each pair of tuples where r belongs to s [r∈s] and s belongs to s [s∈s].

X: Relation

Emp-No	Emp-Name
101	Rita
103	Ritu
104	Setu
106	Leela
107	Neena

Y: Relation

Project
A×10
A×11

For ex → The operation X×Y gives the following combination as follows:

Emp-No	Emp-Name	Project
101	Rita	A×10
101	Rita	A×11
103	Ritu	A×10
103	Ritu	A×11
104	Setu	A×10
104	Setu	A×11
106	Leela	A×10
106	Leela	A×11
107	Neena	A×10
107	Neena	A×11

⑦ Join Operation (⋈)

* It allows to combine two relations to form a single new relation.
* The tuples from the operand relations that participate in the operation and contribute to the result are related.
* The join operation allows the processing of relationship existing between the operand relations.

For Ex → Suppose we want to retrieve the names of the employees who work in department 10, then we will use the following query:-

Employee

Emp_No	Emp_Name	Project
1	Rita	A×10
2	Reetu	A×10
3	Seema	A×11
4	Leela	A×10

Emp_No	Dept_No
1	10
2	20
3	10
4	30

$$\Pi_{(Employee.Emp\_Name)} (Employee \bowtie_{Dept\_No} Department)$$

Types of Joins

① Equi - Join

* A join where the only comparison operator used in the join condition " = " is called Equi-Join.

* The result of equi-join is always has one or more pairs of attributes that have identical values in every tuple.

Employee

Name	Emp_ID	Dept_Name
Mansry	3415	Sales
Sally	2241	Sales
George	3401	Finance
Mansriot	2202	Production

Dept.

Dept_Name	Manager
Sales	Mansriot
Production	Charles

Employee ⋈ Dept OR Employee.Dept_Name = Dept.Dept_Name

Name	Emp_ID	Dept_Name	Dept_Name	Manager
Mansry	3415	Sales	Sales	Mansriot
Sally	2241	Sales	Sales	Mansriot
Mansriot	2202	Production	Production	Charles

(2) Theta-Join [Θ]

* The Θ-Join is a binary operator that is written as R ⋈ S where a and b are attribute names, a Θ b.

Θ is a binary relation in the set(<, <=, >, >=, =)

* The result of this operation consist of all combinations of tuples in R and S that satisfy the relation Θ.

* For Ex- Consider tables Car and Boat which list models of cars and boats and their respective prices. Suppose a customer wants to buy a car and a boat, but doesn't want to spend more money for the boat than for the car.

Car

Car Model	Car Price
Car A	2,00,000
Car B	3,00,000
Car C	5,00,000

Boat

Boat Model	Boat Price
Boat 1	1,00,000
Boat 2	4,00,000
Boat 3	6,00,000

Car ⋈ Boat (Car Price ≥ Boat Price)

Car Model	Car Price	Boat Model	Boat Price
Car A	2,00,000	Boat 1	1,00,000
Car B	3,00,000	Boat 1	1,00,000
Car C	5,00,000	Boat 1	1,00,000
Car C	5,00,000	Boat 2	4,00,000

Natural Join ⟶ Natural Join is same as Equi-Join but match the values in column of the same name and do not output those columns twice.

* The Natural Join is a binary operation that allows us to combine certain collection and cartesian product into one operation.
* This operation forms a cartesian product of its

two arguments performs a selection forcing equality on these attributes that appears in both relations and finally remove duplicate attributes.