Relational and E-R Models

Unit 2

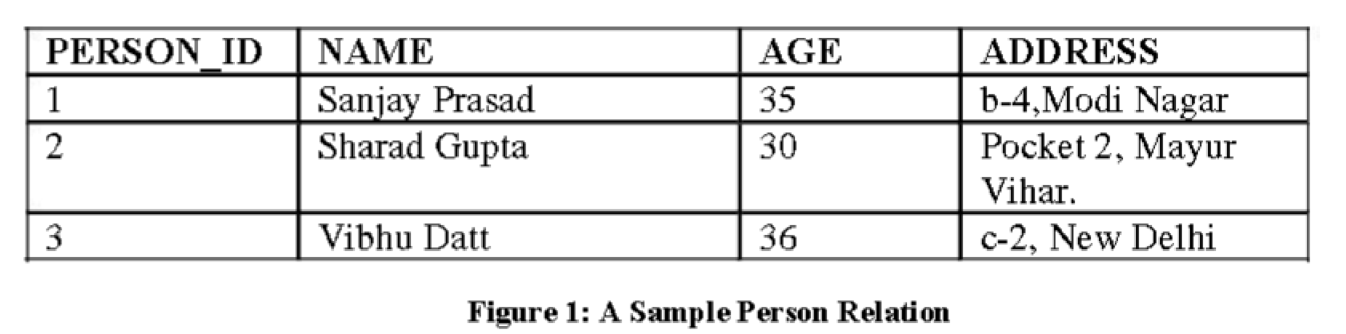
Red: indicates important

Contents

- Relational Model
- Entity Relationship (ER) Model
- E-R diagram
- Conversion of E-R diagram to Relational Database

The Relational Model

- A model in a database defines the structure or organization of data and a set of operations on that data.
- Relational Model :
  - It's a simple model in which the database is represented as a collection of relations, where each relation is represented by two dimensional table.
  - Because of its simplicity it's the most commonly used model.
  - Example :



PERSON_ID	NAME	AGE	ADDRESS
1	Sanjay Prasad	35	b-4,Modi Nagar
2	Sharad Gupta	30	Pocket 2, Mayur Vihar.
3	Vibhu Datt	36	c-2, New Delhi

Figure 1: A Sample Person Relation

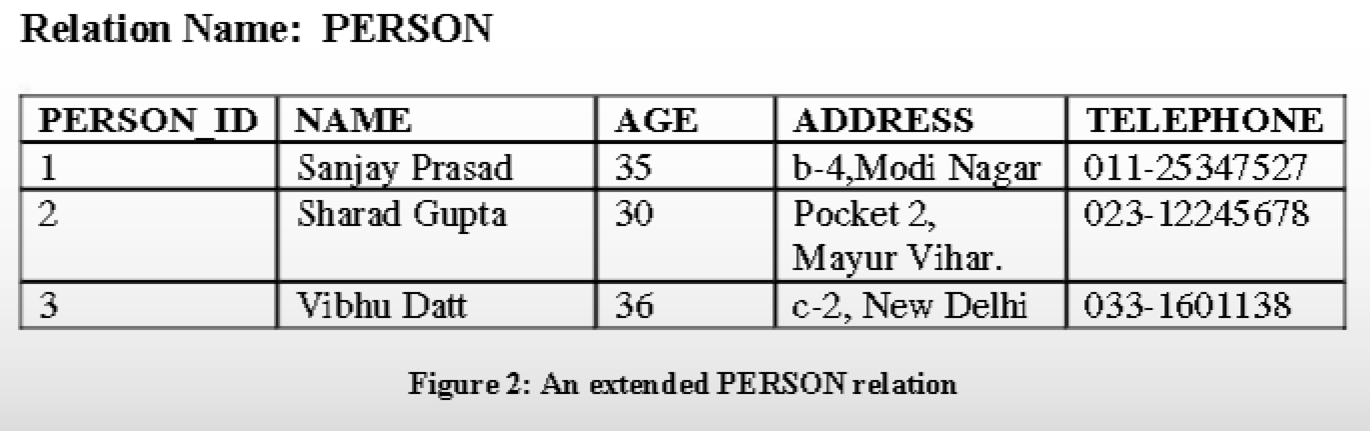
- Advantages of Relational Model :
  - Ease of use : The simple tabular representation of database helps the user define and query the database conveniently. For example, we can easily find the age of a person whose name is “Sharad Gupta” (Figure 1).
  - Flexibility : Database is a collection of tables. New data can be added and deleted easily. Manipulation of data can be done easily using basic operations. For example we can add a new field telephone to the table in Figure 1.
  - Accuracy : Manipulation of database is done by using algebraic operations. These are mathematical operations and ensure accuracy as compared to other models.
- Let us define some basic terms associated with relational model : Domains, Attributes, Tuples and Relation.

Tuple : Each row in a table represents a record and is called a tuple. A table containing ‘n’ attributes in a record is called n-tuple.

Attribute : The name of each column in a table is called an attribute. Its used to interpret its meaning. Each table in a database is called a relation.

Figure 2 represents a relation PERSON. The columns PERSON_
ID, NAME, AGE, ADDRESS, TELEPHONE are called attributes of PERSON.

Each row in a table represents a separate tuple (record).



Relation Name: PERSON

PERSON_ID	NAME	AGE	ADDRESS	TELEPHONE
1	Sanjay Prasad	35	b-4,Modi Nagar	011-25347527
2	Sharad Gupta	30	Pocket 2, Mayur Vihar.	023-12245678
3	Vibhu Datt	36	c-2, New Delhi	033-1601138

Figure 2: An extended PERSON relation

Domain : A domain is a set of permissible values that can be given to an attribute. Every attribute in a table has a specific domain. Values of these attributes cannot be beyond this domain. Domains can also be contiguous.

Example : If domain of AGE is between 1 and 150 then a value beyond this is invalid.

Relation : Each table in a database is called a relation.

Relation consists of :

1. Relational Schema
2. Relational instance

1. Relational Schema :A relational schema specifies the relations name, its attributes and the domain of each attribute. If R is a relation and A1, A2, A3,….An is a list of attributes, then R(A1,A2,A3,……An) is called a relational schema.

   Relational schema of PERSON as in Figure 1 will be :

   PERSON(PERSON_ID: integer, NAME: string, AGE: integer, ADDRESS: string)

   Total number of attributes in a relation denotes the degree of a relation.

   Therefore, degree of PERSON as in Figure 1 is 4.

2. Relation Instance or Relation State : A relation instance denoted by r is a collection of tuple for a given relational schema at a specific point of time.

A relation state r of the relation schema R(A1, A2,A3,....An) also denoted by r(R) is a set of n-tuples

r={t1,t2,....tm}

Where each n-tuple is a order list of n values

T=<v1, v2,......, vn>

where each vi belongs to the domain or contains null values.

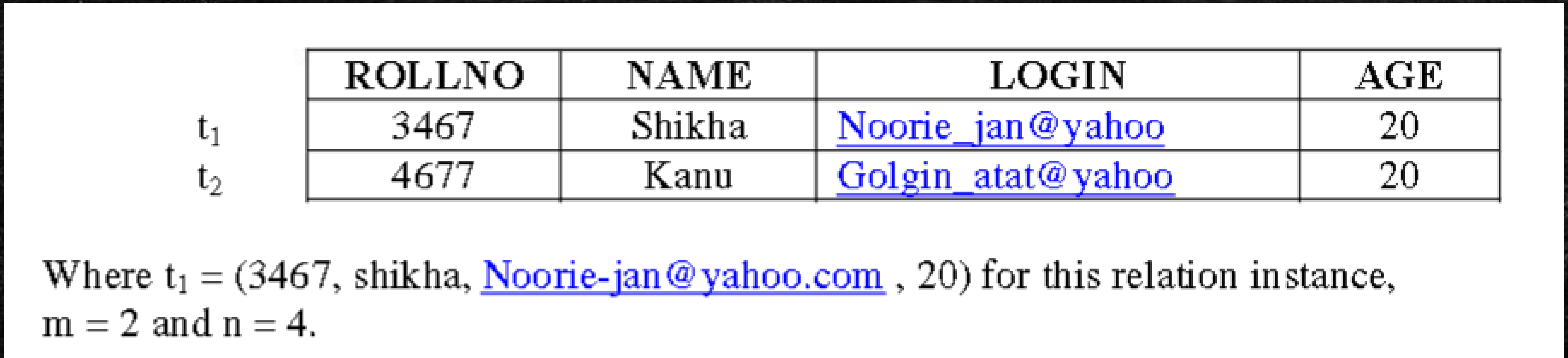
The relation schema is called intension and relation instance is called extension.

Examples :

Relation schema for STUDENT :

STUDENT(ROLLNO: string, NAME: string, LOGIN: string, AGE: integer)

Relation instance :



	ROLLNO	NAME	LOGIN	AGE
t_1	3467	Shikha	Noorie_jan@yahoo	20
t_2	4677	Kanu	Golgin_atat@yahoo	20

Where t_1 = (3467, shikha, Noorie-jan@yahoo.com , 20) for this relation instance, m = 2 and n = 4.

Ordering of tuples:

- Tuples are not inserted in any specific order.
- Ordering of tuples in not defined as an integral part of a relation definition.
- Records can be organized later according to an attribute value in the storage system.
- Relational model does not contain duplicate tuples.

Keys for the relations

- Keys are used to uniquely identify a tuple in a relation.
- Different keys used are : Super key, Candidate keys, Primary key.
- Super keys : Its an attribute or set of attributes used to identify the records uniquely in a relation.

For example, in the PERSON relation discussed earlier PERSON_ID is a super key as PERSON_ID is unique for each person.

Similarly, (PERSON_ID, AGE) and (PERSON_ID, NAME) are also super keys as their combination is also unique for each record.

Super keys can contain extra attributes.

- Candidate keys : Candidate keys are minimal super keys. So they do not contain extra attributes. An attribute is said to be extraneous if even after removing it from the key, makes the remaining attribute still has the properties of a super key.

  Properties of a candidate key :

  - A candidate key must be unique.
  - A candidate key's value must exist i.e. it cannot be null. This property is also known as entity integrity rule.
  - A candidate key is a minimal set of attributes.
  - The value of candidate key must be stable i.e. its value cannot change.

  A relation can have more than one candidate key and one can be chosen as primary key. In the relation PERSON discussed earlier PERSON_ID and NAME are the candidate keys. PERSON_ID is chosen as the primary key.

Relational Constraints

- Database constraints are restrictions on the contents of the database or on database operations

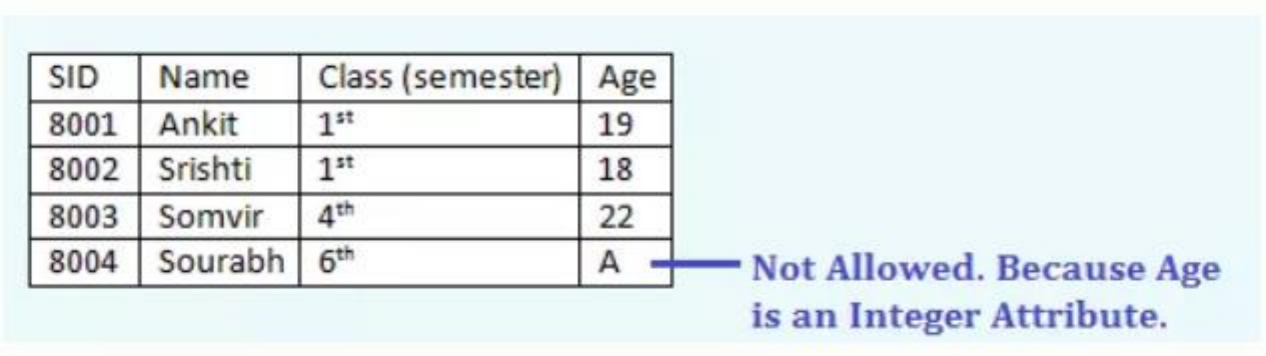
- Three types of constraints on relational database are:
  1. Domain Constraints
  2. Primary Key Constraints
  3. Integrity Constraints

1. Domain Constraints :

- It specifies that each attribute in a relation must contain atomic value only from the corresponding domains.

- Data types of RDBMS are :
  1. numeric data types for integers (such as short-integer, integer, long integer)
  2. Real numbers (float, double precision floats)
  3. Characters
  4. Fixed length strings and variable length strings.
- Domain constraints specify what set of values an attribute can take.

Example :



SID	Name	Class (semester)	Age
8001	Ankit	1st	19
8002	Srishti	1st	18
8003	Somvir	4th	22
8004	Sourabh	6th	A

Not Allowed. Because Age is an Integer Attribute.

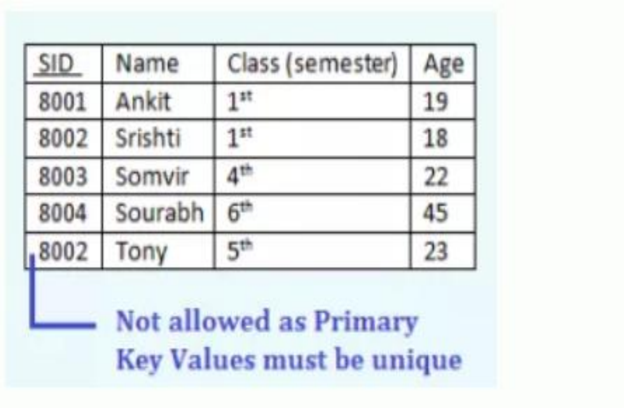
2. Key Constraints :

- It states that the key attribute value in each tuple must be unique, i.e., no two tuples can have the same value for the key attribute. This is because the value of the primary key is used to identify the tuples in the relation.

  Example 1 : In the PERSON relation (discussed earlier), PERSON_ID is a primary key so PERSON_ID cannot be given as the same for two persons.

  Example 2:

Here SID is the primary key.



SID	Name	Class (semester)	Age
8001	Ankit	1st	19
8002	Srishti	1st	18
8003	Somvir	4th	22
8004	Sourabh	6th	45
8002	Tony	5th	23

Not allowed as Primary Key Values must be unique

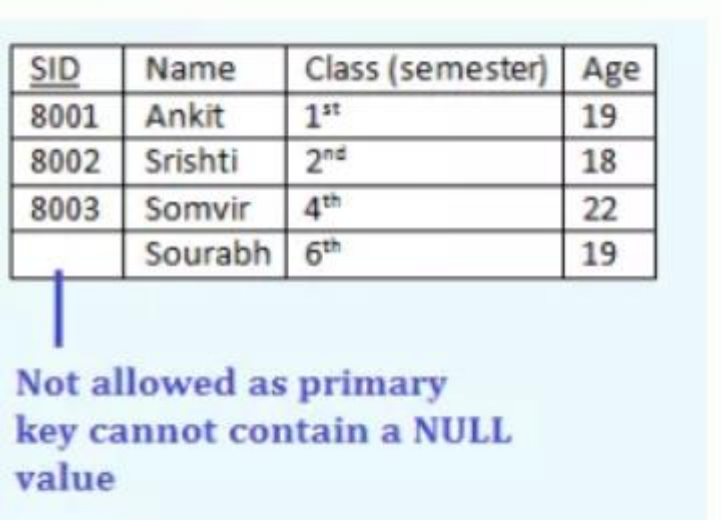
3. Integrity Constraints :

- Integrity means to maintain consistency of data.
- Integrity constraints in a database ensure that changes made to the database by authorized users do not compromise data consistency.
- Two types of integrity constraints :
    1. Entity integrity constraints
    2. Referential integrity constraints

Entity integrity constraints :

- It states that no primary key value can be null. Its because primary key is used to identify individual tuple in the relation. So we will not be able to identify the records uniquely containing null values for the primary key attributes.

- This constraint is specified on one individual relation.
- Consider the below relation:



SID	Name	Class (semester)	Age
8001	Ankit	1st	19
8002	Srishti	2nd	18
8003	Somvir	4th	22
	Sourabh	6th	19

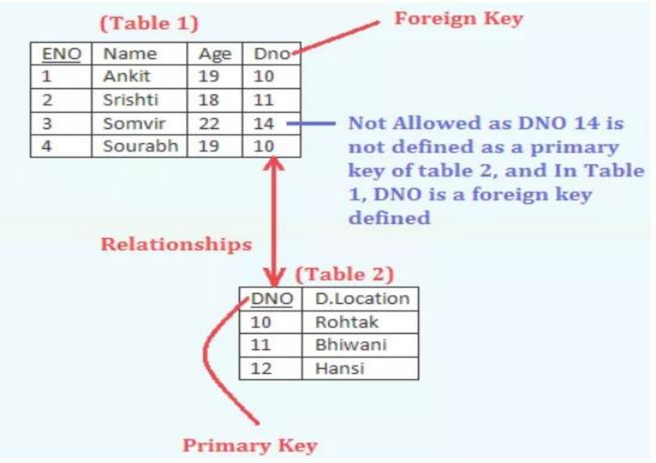
Not allowed as primary key cannot contain a NULL value

The above relation instance is an invalid instance.

Referential integrity constraints :

- It states that tuple in one relation that refers to another relation must refer to an existing tuple in that relation.
- This constraint is specified on two relations.
- It uses the concept of foreign key.

Example :



(Table 1)

ENO	Name	Age	Dno
1	Ankit	19	10
2	Srishti	18	11
3	Somvir	22	14
4	Sourabh	19	10

Foreign Key

Not Allowed as DNO 14 is not defined as a primary key of table 2, and In Table 1, DNO is a foreign key defined

Relationships

(Table 2)

DNO	D.Location
10	Rohtak
11	Bhiwani
12	Hansi

Primary Key

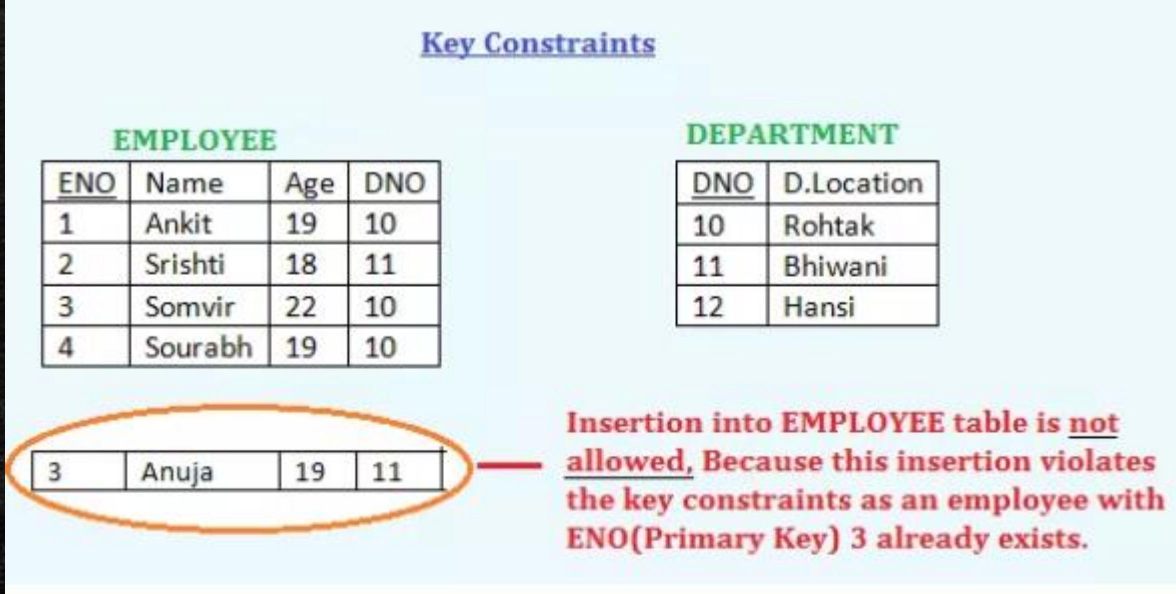
Dealing with constraint violation

- There are three basic operations on a relation :
  1. Insertion
  2. Deletion
  3. Update

1. Insertion : The insert operation allows us to insert a new tuple in a relation. When we try to insert a new record then any of the following four types of constraints can be violated :

- Domain constraint : If the value given to an attribute lies outside the domain of that attribute.

- Key constraint : If the value of the key attribute in new tuple t is the same as in the existing tuple in relation R.



Key Constraints

EMPLOYEE

ENO	Name	Age	DNO
1	Ankit	19	10
2	Srishti	18	11
3	Somvir	22	10
4	Sourabh	19	10

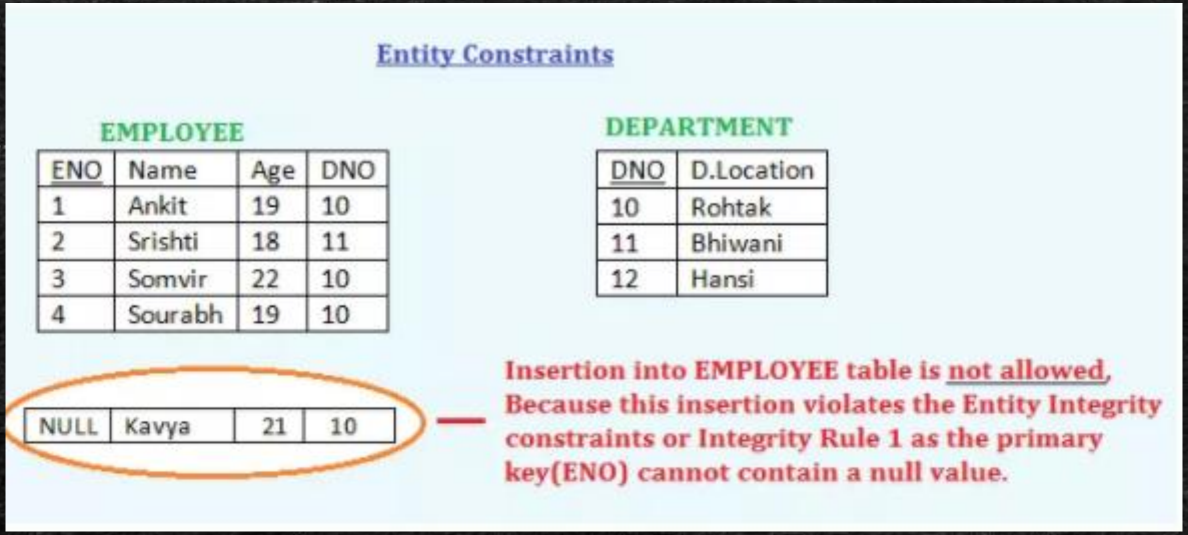
DEPARTMENT

DNO	D.Location
10	Rohtak
11	Bhiwani
12	Hansi

3	Anuja	19	11

Insertion into EMPLOYEE table is not allowed, Because this insertion violates the key constraints as an employee with ENO(Primary Key) 3 already exists.

- Entity Integrity constraint : If the primary key attribute value of new tuple t is given as null.



Entity Constraints

EMPLOYEE

ENO	Name	Age	DNO
1	Ankit	19	10
2	Srishti	18	11
3	Somvir	22	10
4	Sourabh	19	10

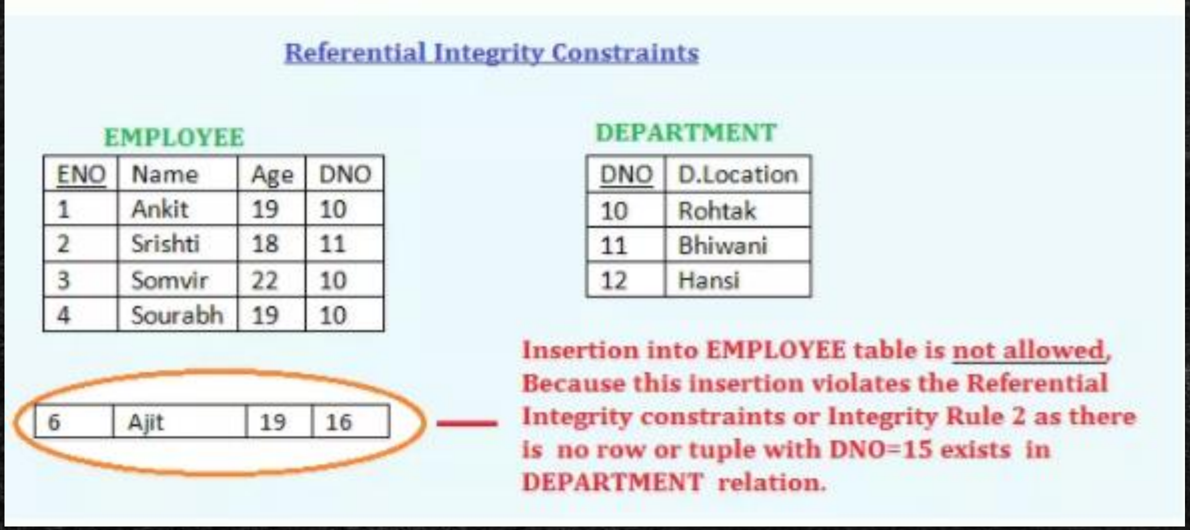
DEPARTMENT

DNO	D.Location
10	Rohtak
11	Bhiwani
12	Hansi

NULL	Kavya	21	10

Insertion into EMPLOYEE table is not allowed, Because this insertion violates the Entity Integrity constraints or Integrity Rule 1 as the primary key(ENO) cannot contain a null value.

- Referential Integrity constraint : If the value of the foreign key in t refers to a tuple that doesn't appear in the referenced relation.



Referential Integrity Constraints

EMPLOYEE

ENO	Name	Age	DNO
1	Ankit	19	10
2	Srishti	18	11
3	Somvir	22	10
4	Sourabh	19	10

DEPARTMENT

DNO	D.Location
10	Rohtak
11	Bhiwani
12	Hansi

6	Ajit	19	16

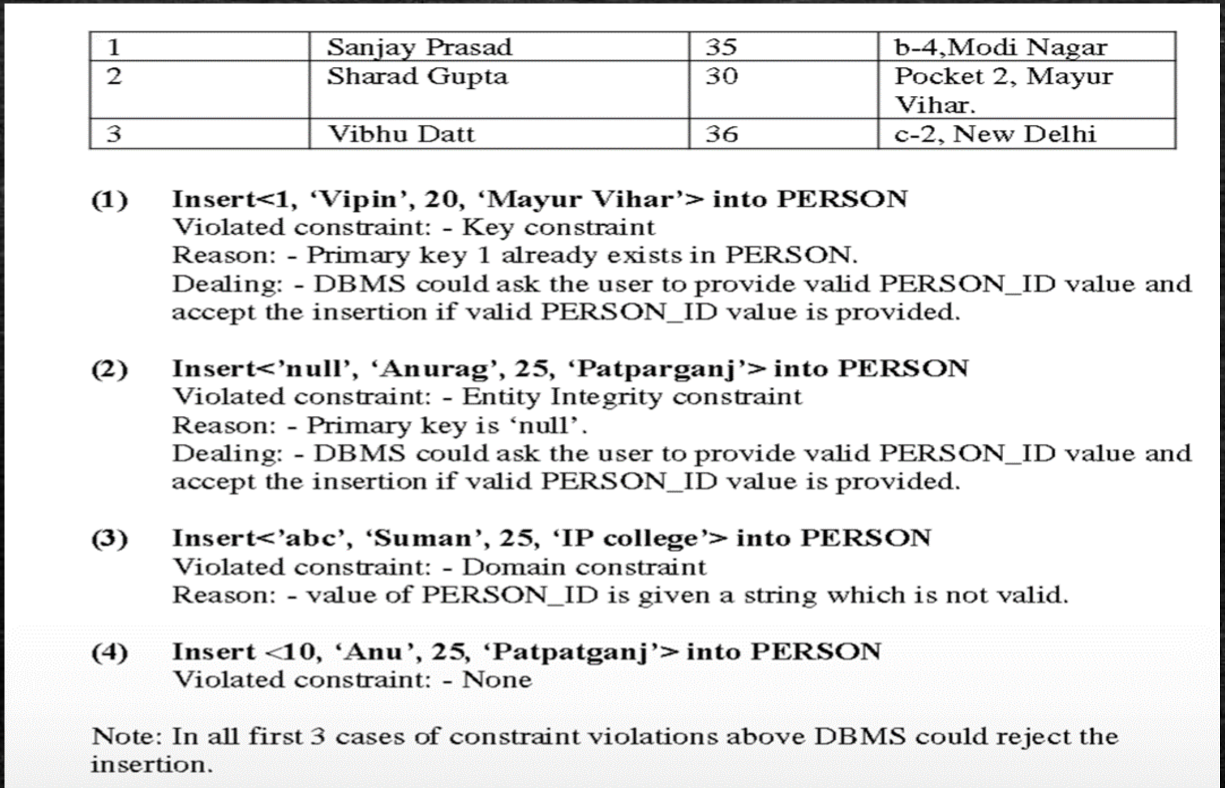
Insertion into EMPLOYEE table is not allowed, Because this insertion violates the Referential Integrity constraints or Integrity Rule 2 as there is no row or tuple with DNO=15 exists in DEPARTMENT relation.

Dealing with constraints violation during insertion :

If the insertion violates one or more constraints, then two options are available :

- Default option : Insertion can be rejected and the reason of rejection can also be explained to the user by DBMS.
- Ask the user to correct the data, resubmit, also give the reason for rejecting the insertion.l

Example :



1	Sanjay Prasad	35	b-4,Modi Nagar
2	Sharad Gupta	30	Pocket 2, Mayur Vihar.
3	Vibhu Datt	36	c-2, New Delhi

(1) **Insert<1, 'Vipin', 20, 'Mayur Vihar'> into PERSON**
Violated constraint: - Key constraint
Reason: - Primary key 1 already exists in PERSON.
Dealing: - DBMS could ask the user to provide valid PERSON_ID value and accept the insertion if valid PERSON_ID value is provided.

(2) **Insert<'null', 'Anurag', 25, 'Patparganj'> into PERSON**
Violated constraint: - Entity Integrity constraint
Reason: - Primary key is 'null'.
Dealing: - DBMS could ask the user to provide valid PERSON_ID value and accept the insertion if valid PERSON_ID value is provided.

(3) **Insert<'abc', 'Suman', 25, 'IP college'> into PERSON**
Violated constraint: - Domain constraint
Reason: - value of PERSON_ID is given a string which is not valid.

(4) **Insert <10, 'Anu', 25, 'Patpatganj'> into PERSON**
Violated constraint: - None

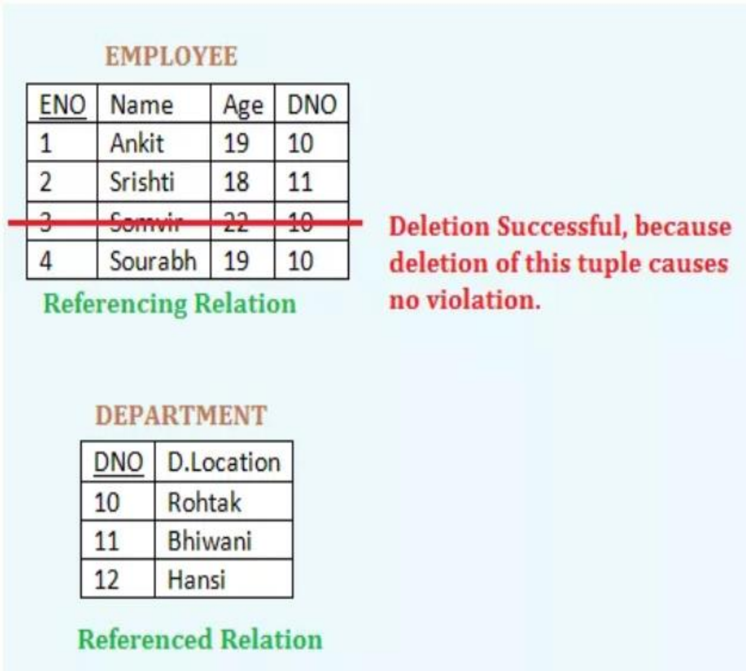
Note: In all first 3 cases of constraint violations above DBMS could reject the insertion.

2. Deletion :

- The delete operation is used to delete some exiting records from a relation.

- To delete some specific records from the database a condition is specified based on which records can be selected for deletion.

- Only one type of constraint can be violated during deletion, it is referential integrity constraint. It occurs when you want to delete a record in the table where it is referenced by the foreign key of another table.



EMPLOYEE

ENO	Name	Age	DNO
1	Ankit	19	10
2	Srishti	18	11
~~3~~	~~Somvir~~	~~22~~	~~10~~
4	Sourabh	19	10

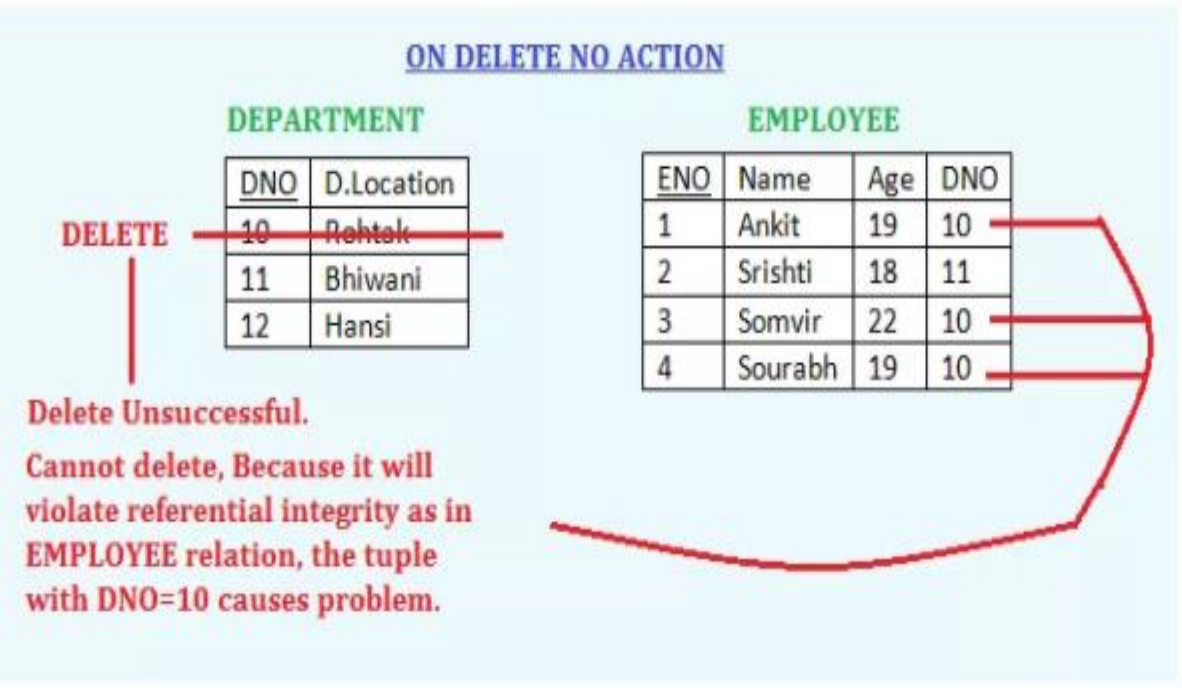
Referencing Relation

Deletion Successful, because deletion of this tuple causes no violation.

DEPARTMENT

DNO	D.Location
10	Rohtak
11	Bhiwani
12	Hansi

Referenced Relation



ON DELETE NO ACTION

DEPARTMENT

DNO	D.Location
~~10~~	~~Rohtak~~
11	Bhiwani
12	Hansi

EMPLOYEE

ENO	Name	Age	DNO
1	Ankit	19	10
2	Srishti	18	11
3	Somvir	22	10
4	Sourabh	19	10

DELETE

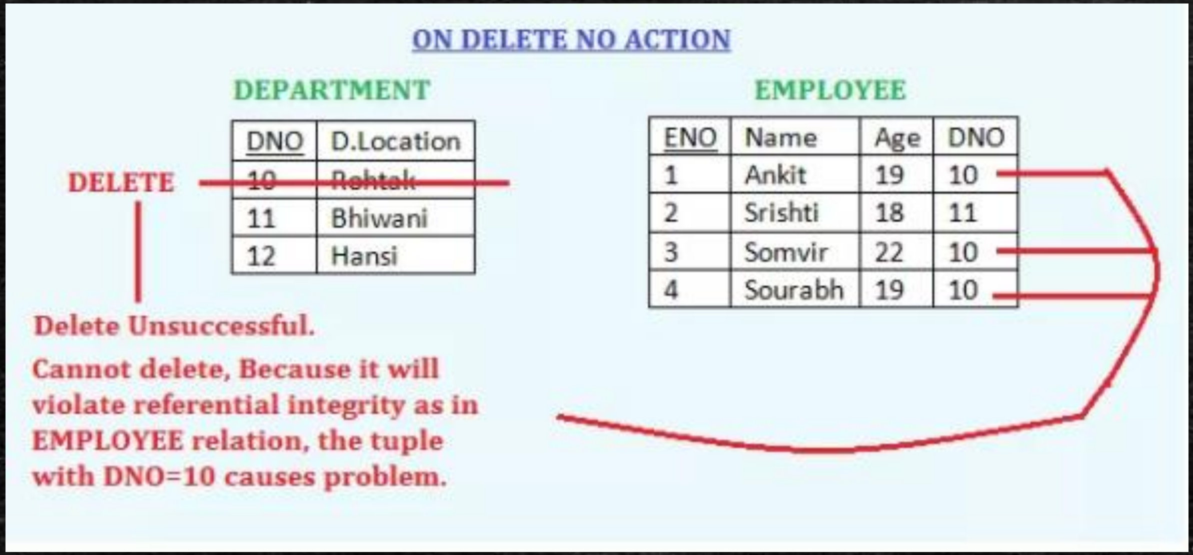
Delete Unsuccessful.

Cannot delete, Because it will violate referential integrity as in EMPLOYEE relation, the tuple with DNO=10 causes problem.

Dealing with constraints violation :

If the deletion violates referential integrity constraint, then three options are available :

- Default option : Reject the option. It is the job of the DBMS to explain to the user why the deletion was rejected.



ON DELETE NO ACTION

DEPARTMENT

DNO	D.Location
~~10~~	~~Rohtak~~
11	Bhiwani
12	Hansi

EMPLOYEE

ENO	Name	Age	DNO
1	Ankit	19	10
2	Srishti	18	11
3	Somvir	22	10
4	Sourabh	19	10

Delete Unsuccessful.

Cannot delete, Because it will violate referential integrity as in EMPLOYEE relation, the tuple with DNO=10 causes problem.

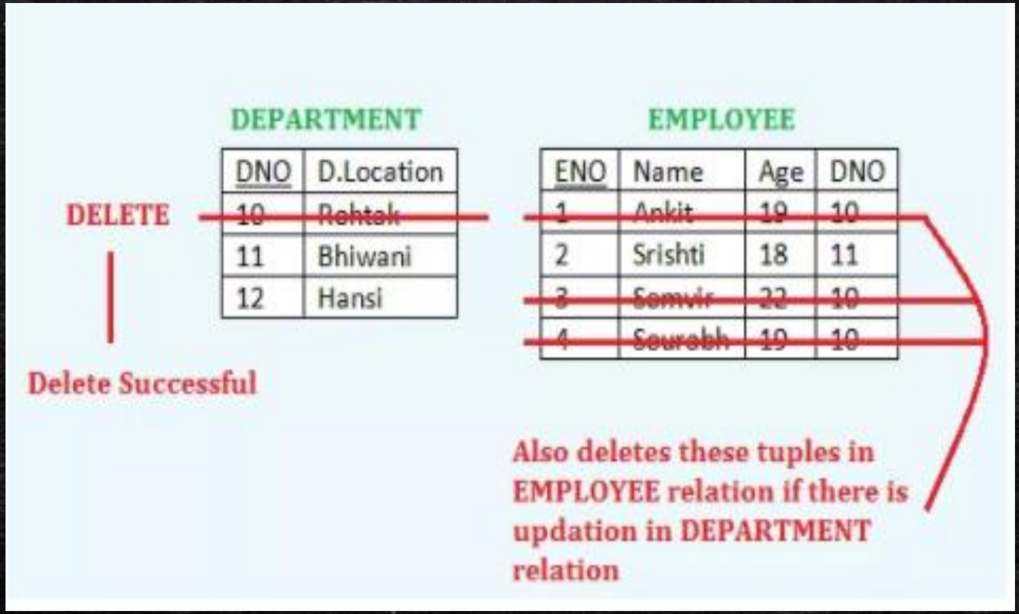
- Attempt to cascade the deletion by deleting tuples that reference the tuple that is being deleted.

Deletion of **<10, ‘Rohtak’>** from DEPARTMENT relation will delete the following tuples in EMPLOYEE relation :

<1, ‘Ankit’, 19, 10>

<3, ‘Somvir’, 22, 10>

<4, ‘Sourabh’, 19, 10>



DEPARTMENT

DNO	D.Location
~~10~~	~~Rohtak~~
11	Bhiwani
12	Hansi

EMPLOYEE

ENO	Name	Age	DNO
~~1~~	~~Ankit~~	~~19~~	~~10~~
2	Srishti	18	11
~~3~~	~~Somvir~~	~~22~~	~~10~~
~~4~~	~~Sourabh~~	~~19~~	~~10~~

Also deletes these tuples in EMPLOYEE relation if there is updation in DEPARTMENT relation

- Change the value of referencing attribute that causes the violation.

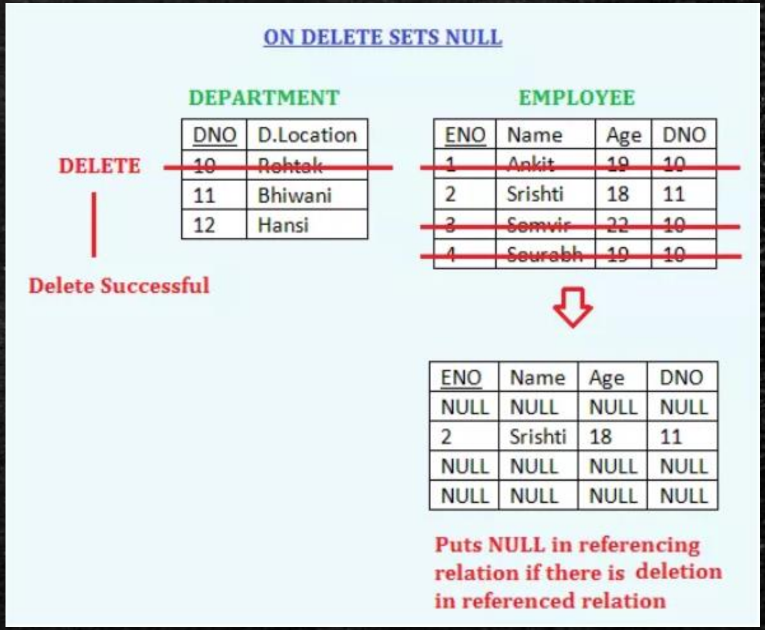
For example, Deletion of **<10, ‘Rohtak’>** from DEPARTMENT relation will delete the following tuples in EMPLOYEE relation :

<1, ‘Ankit’, 19, 10>

<3, ‘Somvir’, 22, 10>

<4, ‘Sourabh’, 19, 10>

and null values will be filled in the EMPLOYEE relation on that place.



ON DELETE SETS NULL

DEPARTMENT

DNO	D.Location
~~10~~	~~Rohtak~~
11	Bhiwani
12	Hansi

EMPLOYEE

ENO	Name	Age	DNO
~~1~~	~~Ankit~~	~~19~~	~~10~~
2	Srishti	18	11
~~3~~	~~Somvir~~	~~22~~	~~10~~
~~4~~	~~Sourabh~~	~~19~~	~~10~~

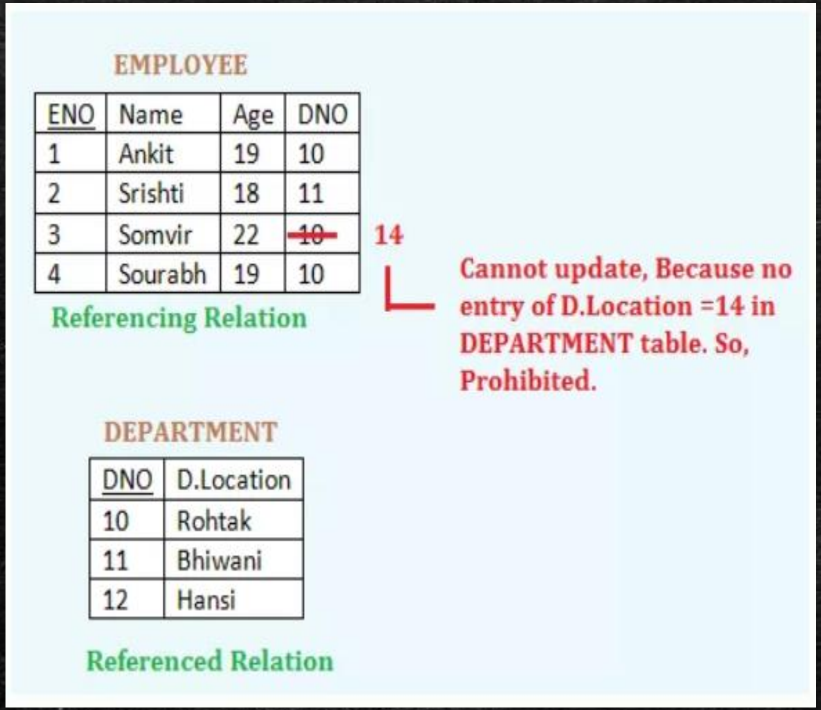
ENO	Name	Age	DNO
NULL	NULL	NULL	NULL
2	Srishti	18	11
NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL

Puts NULL in referencing relation if there is deletion in referenced relation

Update :

- Update is done to modify database values. It is necessary to specify a condition on the attributes of the relation to select the tuple (or tuples) to be modified.
- The constrain violations faced here are same as that of insertion and deletion.
- Domain constraint : If the value of an attribute is updated to a value that lies outside the domain of that attribute.
- Key constraint : If the value of the key attribute of a tuple t is updated to value that already exists in another tuple in relation R.
- Entity Integrity constraint : If the primary key attribute value of a tuple t is updated as null.
- Referential Integrity constraint : If the value of the foreign key in t refers to a tuple that doesn't appear in the referenced relation.

Examples : Referential integrity



EMPLOYEE

ENO	Name	Age	DNO
1	Ankit	19	10
2	Srishti	18	11
3	Somvir	22	~~10~~
4	Sourabh	19	10

14

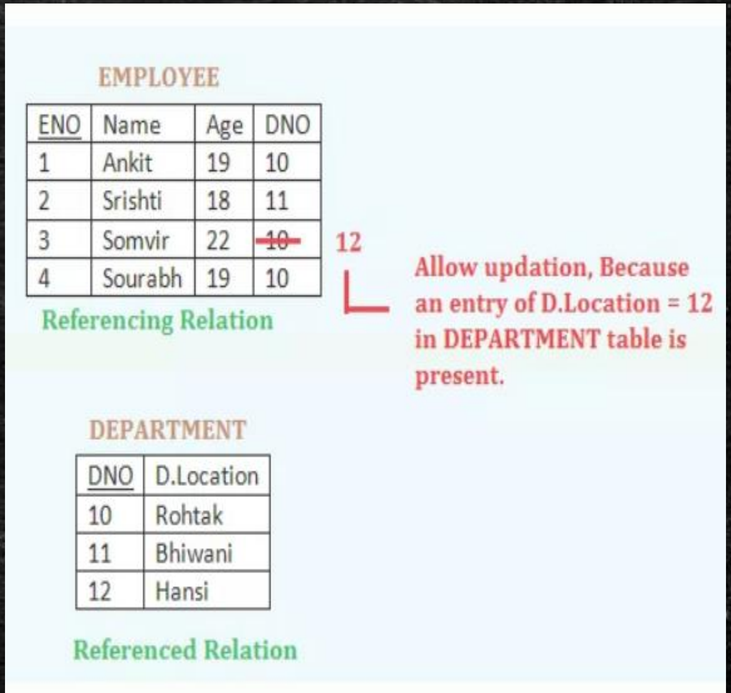
Cannot update, Because no entry of D.Location =14 in DEPARTMENT table. So, Prohibited.

Referencing Relation

DEPARTMENT

DNO	D.Location
10	Rohtak
11	Bhiwani
12	Hansi

Referenced Relation



EMPLOYEE

ENO	Name	Age	DNO
1	Ankit	19	10
2	Srishti	18	11
3	Somvir	22	~~10~~
4	Sourabh	19	10

12

Allow updation, Because an entry of D.Location = 12 in DEPARTMENT table is present.

Referencing Relation

DEPARTMENT

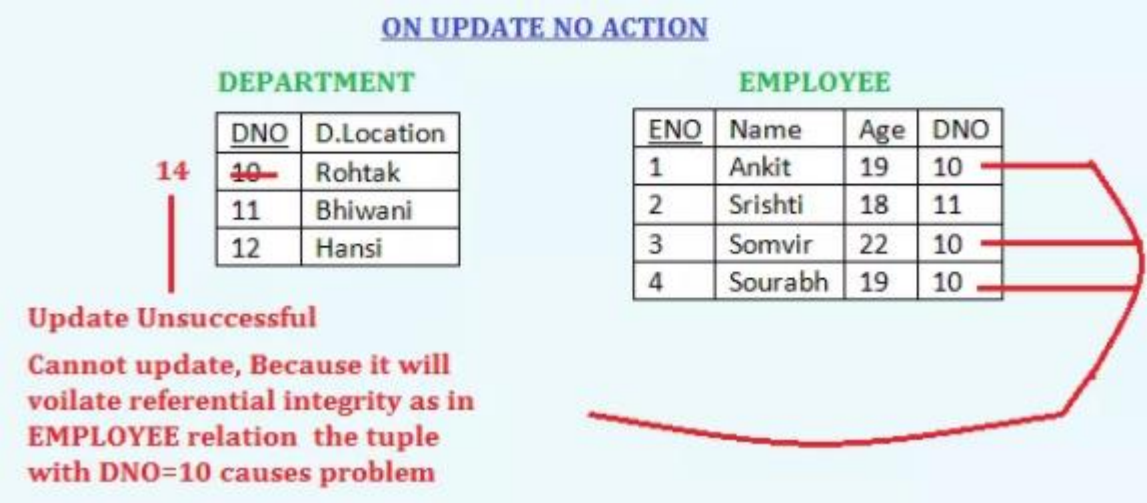
DNO	D.Location
10	Rohtak
11	Bhiwani
12	Hansi

Referenced Relation

Dealing with constraints :

If the deletion violates one or more constraints, then three options are available :

- Default option : Reject the option. It is the job of the DBMS to explain to the user why the update process was rejected.



ON UPDATE NO ACTION

DEPARTMENT

14

DNO	D.Location
~~10~~	Rohtak
11	Bhiwani
12	Hansi

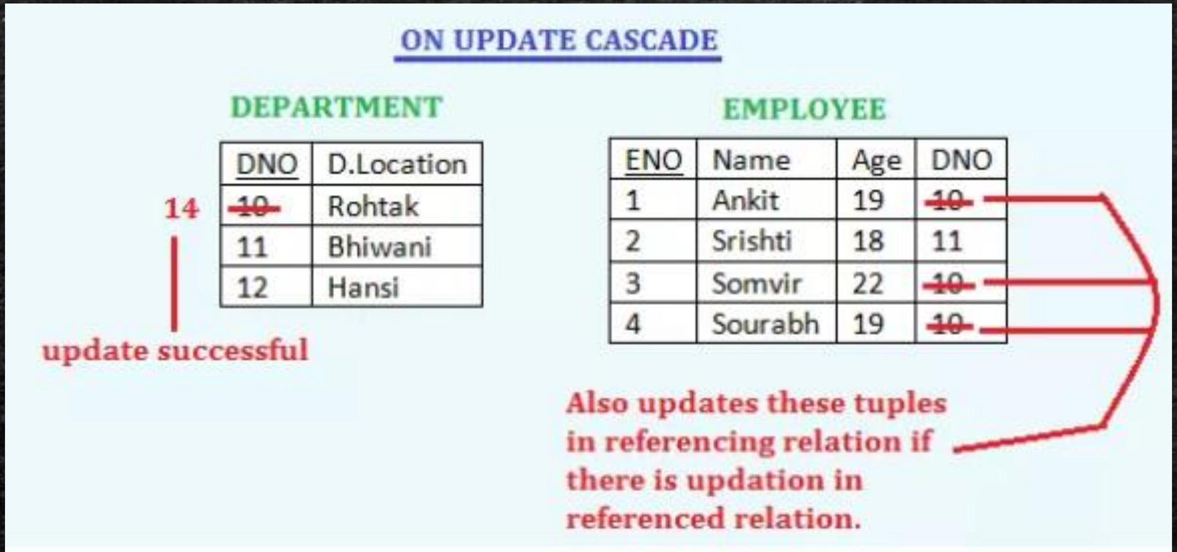
EMPLOYEE

ENO	Name	Age	DNO
1	Ankit	19	10
2	Srishti	18	11
3	Somvir	22	10
4	Sourabh	19	10

Update Unsuccessful

Cannot update, Because it will voilate referential integrity as in EMPLOYEE relation the tuple with DNO=10 causes problem

- Cascade **:** If update causes integrity violation, then update in both the table i.e. if the tuples are updated from the referenced table, then the tuple will also be updates from the referencing relation that is being updated.



ON UPDATE CASCADE

DEPARTMENT

DNO	D.Location
~~10~~ 14	Rohtak
11	Bhiwani
12	Hansi

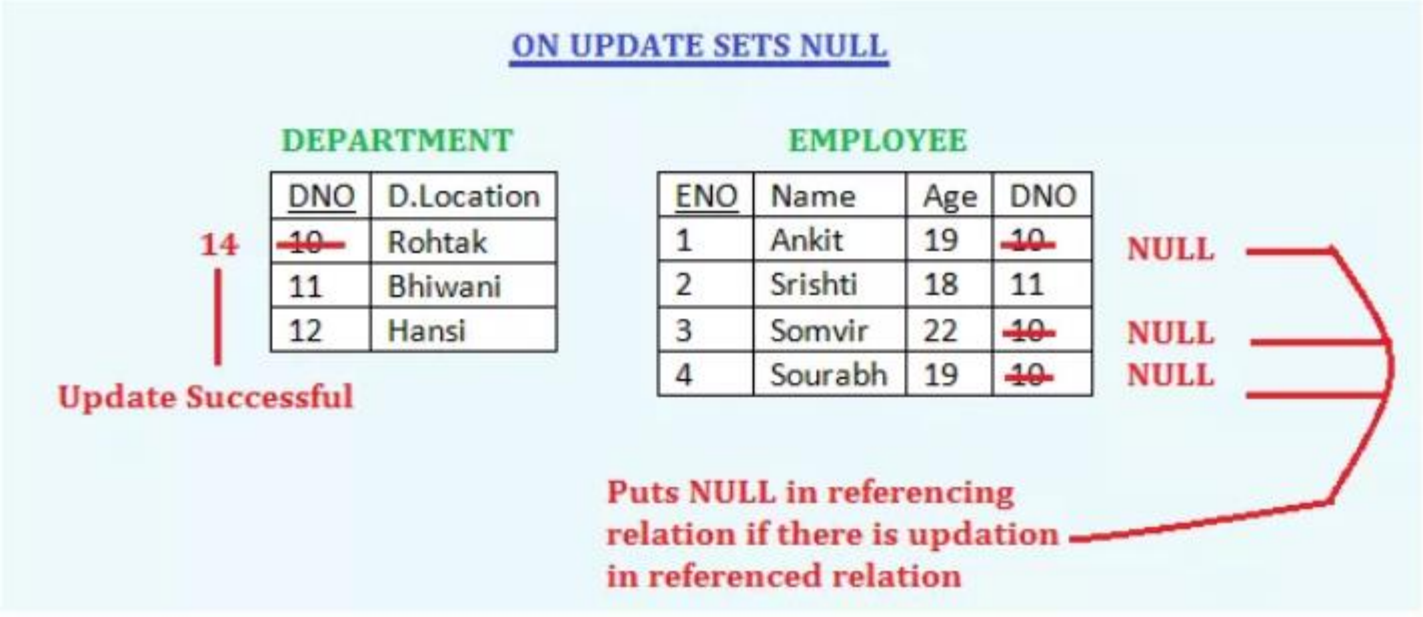
update successful

EMPLOYEE

ENO	Name	Age	DNO
1	Ankit	19	~~10~~
2	Srishti	18	11
3	Somvir	22	~~10~~
4	Sourabh	19	~~10~~

Also updates these tuples in referencing relation if there is updation in referenced relation.

- Modify the referencing Attributes **:** sets null value or some valid value in the foreign key field for corresponding updating referenced value. i.e. changing/updating the referencing attribute values that cause the violation either null or another valid value.
  If there is no restriction or constraint applied for putting the NULL value in the referencing relation – then allow to update from referenced relation otherwise prohibited.



ON UPDATE SETS NULL

DEPARTMENT

DNO	D.Location
~~10~~	Rohtak
11	Bhiwani
12	Hansi

EMPLOYEE

ENO	Name	Age	DNO
1	Ankit	19	~~10~~
2	Srishti	18	11
3	Somvir	22	~~10~~
4	Sourabh	19	~~10~~

Puts NULL in referencing relation if there is updation in referenced relation

Relational Algebra

- Relational algebra is a set of basic operations used to manipulate the data in relational model.

- Result is a new relation formed from one or more relations.

- These operations can be classified into two :
  - Basic Set Operations
    1. UNION
    2. INTERSECTION
    3. SET DIFFERENCE
    4. CARTESIAN PRODUCT
  - Relational Operations
    1. SELECT
    2. PROJECT
    3. JOIN
    4. DIVISION

- <u>Basic Set Operations :</u>

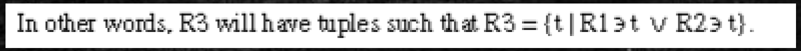
- These are binary operations i.e. applied on two sets or relations.

- These two relations should be union compatible except the Cartesian Product.

- Two relations R(A1, A2, A3,…… An) and S(B1, B2, B3,…Bn) are said to be union compatible if they have the same degree n and domains of the corresponding attributes are also the same (Domain (Ai)=Domain (Bi) for 1<=i<=n).

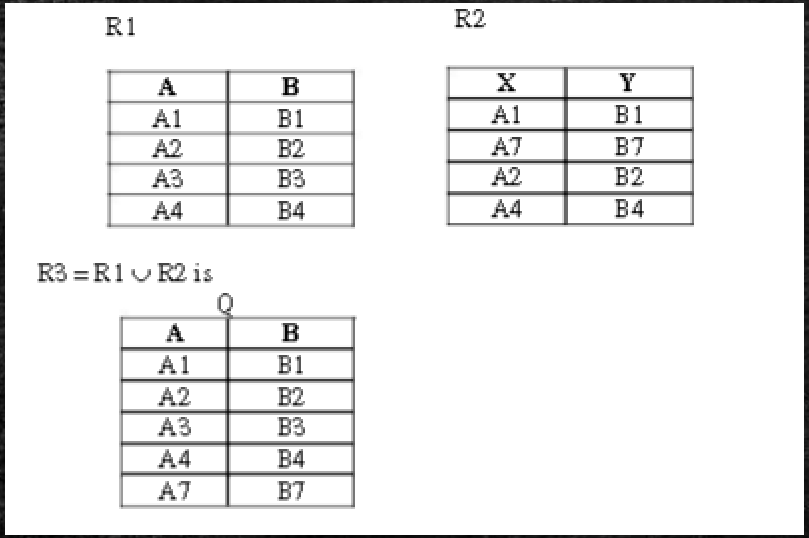
1. UNION :

- If R1 and R2 are two union compatible relations then R3=R1 U R2 is the relation that contains tuples that are either in R1 or in R2 or in both.



In other words, R3 will have tuples such that $R3 = \{t \mid R1 \ni t \lor R2 \ni t\}$.

- Example :



R1

A	B
A1	B1
A2	B2
A3	B3
A4	B4

R2

X	Y
A1	B1
A7	B7
A2	B2
A4	B4

R3 = R1 ∪ R2 is

Q

A	B
A1	B1
A2	B2
A3	B3
A4	B4
A7	B7

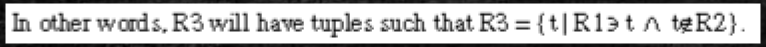
- Properties of union operation :
  1. Union is a commutative operation i.e.,

     $R \cup S = S \cup R$
  2. Union is an associative operation i.e.,

     $R \cup (S \cup T) = (R \cup S) \cup T$

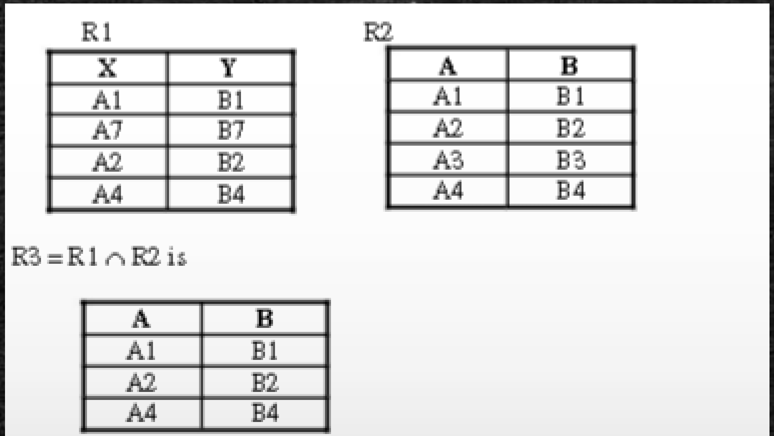
2. Intersection

- If R1 and R2 are two union compatible functions or relations, then the result of $R_3 = R_1 \cap R_2$ is the relation that includes all tuples that are both in both the relations.



In other words, R3 will have tuples such that $R3 = \{ t \mid R1 \ni t \wedge t \notin R2 \}$.

- Example :



R1

X	Y
A1	B1
A7	B7
A2	B2
A4	B4

R2

A	B
A1	B1
A2	B2
A3	B3
A4	B4

R3 = R1 ∩ R2 is

A	B
A1	B1
A2	B2
A4	B4

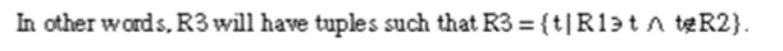
- Properties of intersection operation :
  1. Intersection is a commutative operation i.e.,

     $R1 \cap R2 = R2 \cap R1$
  2. Intersection is an associative operation i.e.,

     $R1 \cap (R2 \cap R3)=(R1 \cap R2) \cap R3$

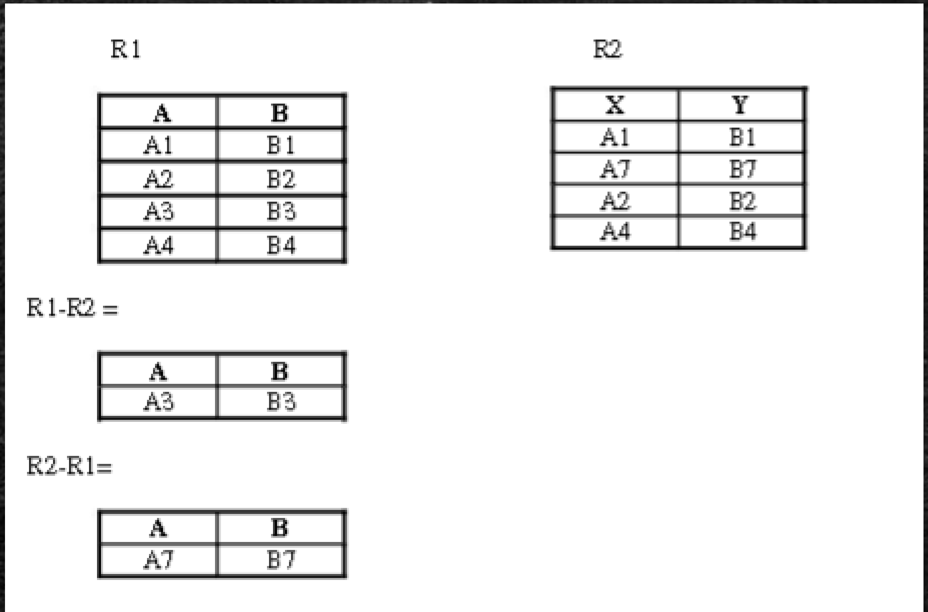
3. Set difference :

- If R1 and R2 are two union compatible functions or relations then result of R3=R1-R2 is the relation that includes only those tuples that are in R1 but not in R2.



In other words, R3 will have tuples such that $R3 = \{ t \mid R1 \ni t \wedge t \notin R2 \}$.

- Example :



R1

A	B
A1	B1
A2	B2
A3	B3
A4	B4

R2

X	Y
A1	B1
A7	B7
A2	B2
A4	B4

R1-R2 =

A	B
A3	B3

R2-R1=

A	B
A7	B7

- Properties of difference operation :
  1. Difference operation is not commutative, i.e.,

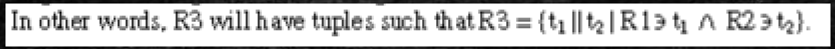
     R1-R2 ≠ R2-R1
  2. Difference operation is not associative, i.e.,

     R1-(R2-R3) ≠ (R1-R2)-R3

4. Cartesian Product :

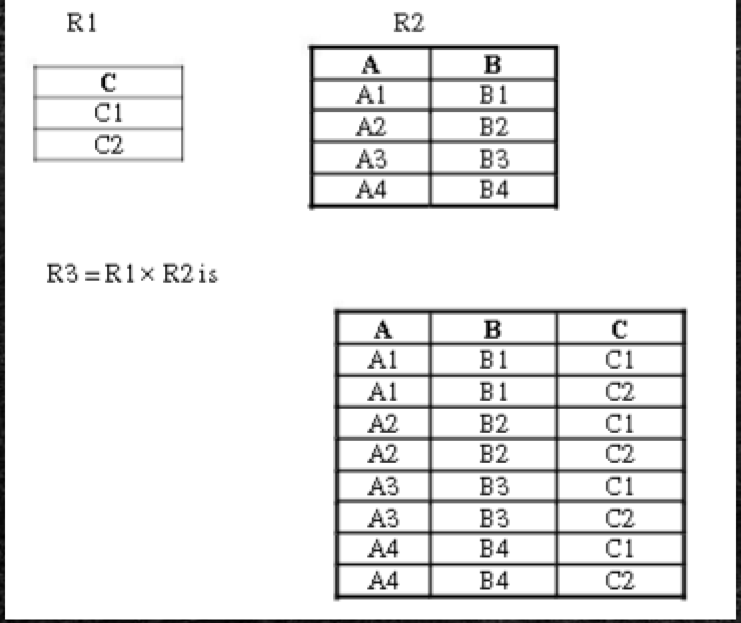
- If R1 and R2 are two functions or relations, then the result of R3 = R1 X R2 is the combination of tuples that are in R1 and R2. The product is commutative and associative.

- Degree(R3)=Degree(R1)+Degree(R2)



In other words, R3 will have tuples such that R3 = $\{t_1 \| t_2 \mid R1 \ni t_1 \wedge R2 \ni t_2\}$.

- Example :



R1

C
C1
C2

R2

A	B
A1	B1
A2	B2
A3	B3
A4	B4

R3 = R1 × R2 is

A	B	C
A1	B1	C1
A1	B1	C2
A2	B2	C1
A2	B2	C2
A3	B3	C1
A3	B3	C2
A4	B4	C1
A4	B4	C2

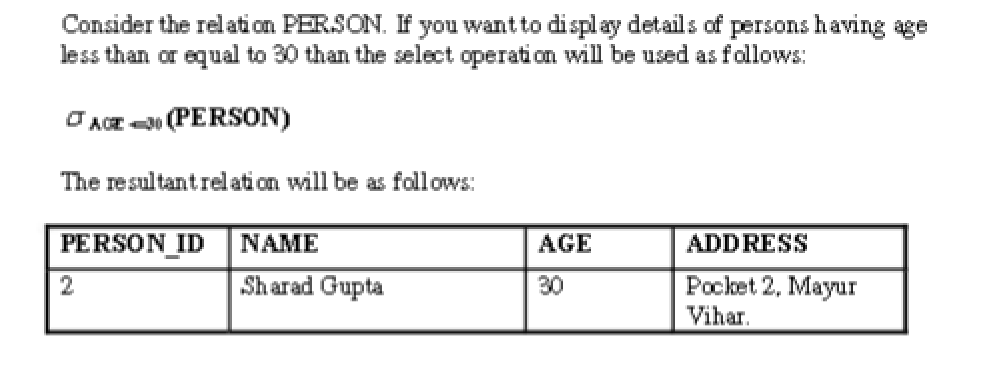
- Relational Operations :

1. SELECT :

- The select operation is used to select some specific records from the database based on some criteria.
- It's a unary operation i.e. its applied on one set or relation.
- Its denoted by σ .
- Syntax : σ <Selection condition> (Relation)

Selection condition is specified in <Select condition>. It is made up of a number of clauses of the form : <attribute name><comparison operator><constant value> or <attribute name><comparison operator><attribute name>

Comparison operators in the set {≤,≥,≠,=,<,>} apply to the attributes whose domains are ordered value like integer.



Consider the relation PERSON. If you want to display details of persons having age less than or equal to 30 than the select operation will be used as follows:

$\sigma_{AGE <=30}$ (PERSON)

The resultant relation will be as follows:

PERSON_ID	NAME	AGE	ADDRESS
2	Sharad Gupta	30	Pocket 2, Mayur Vihar.

- Properties of SELECT operation :

1. Select operation is commutative i.e.,

$$\sigma_{<condition1>}(\sigma_{<condition2>}(R)) = \sigma_{<condition2>}(\sigma_{<condition1>}(R))$$

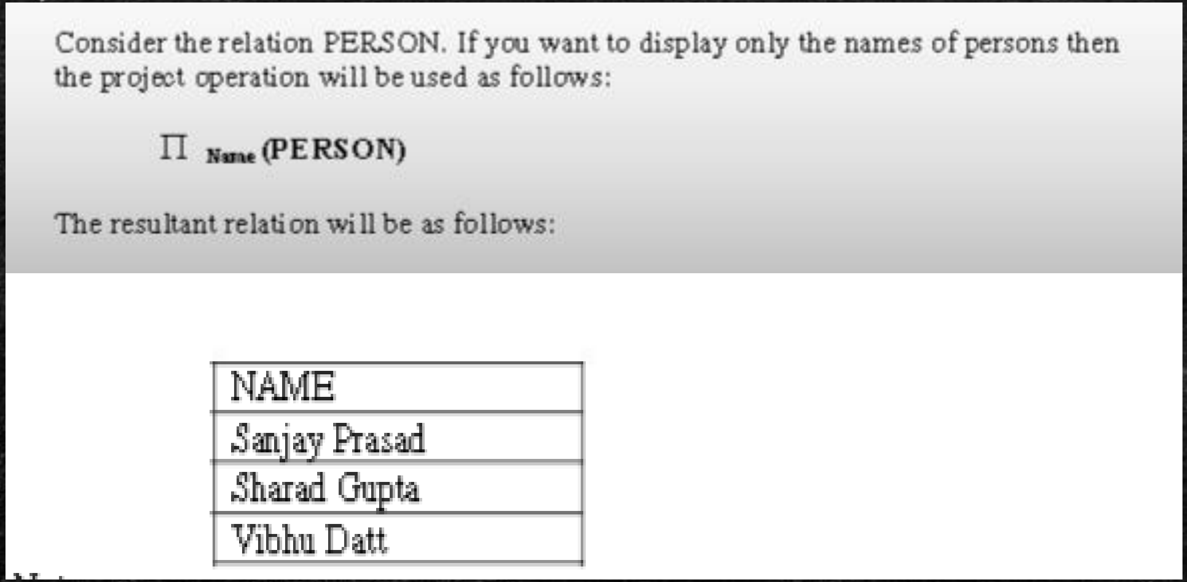
2. More than one condition can be applied using Boolean operators AND & OR etc.

2. PROJECT :

- The project operation is used to select the records with specified attributes while discarding the others based on some specific criteria.

- Its denoted by ∏.

- Syntax : $\prod_{\text{list of attributes for project}}$ (Relation)
- Example :



Consider the relation PERSON. If you want to display only the names of persons then the project operation will be used as follows:

$$\Pi_{\text{Name}}\,\textbf{(PERSON)}$$

The resultant relation will be as follows:

NAME
Sanjay Prasad
Sharad Gupta
Vibhu Datt

- Properties :

1. $\prod_{<list1>} (\prod_{<list2>}(R))= \prod_{<list\ 1>} (R)$ As long as <list2> contains attribute in <list1>.

3. JOIN operation :

- Its applied on two relations.
- A JOIN clause is used to combine rows from two or more tables, based on a related column between them.
- Its denoted by ⋈.
- Syntax : R1 ⋈<join condition> R2 .

<join condition> is of the form :

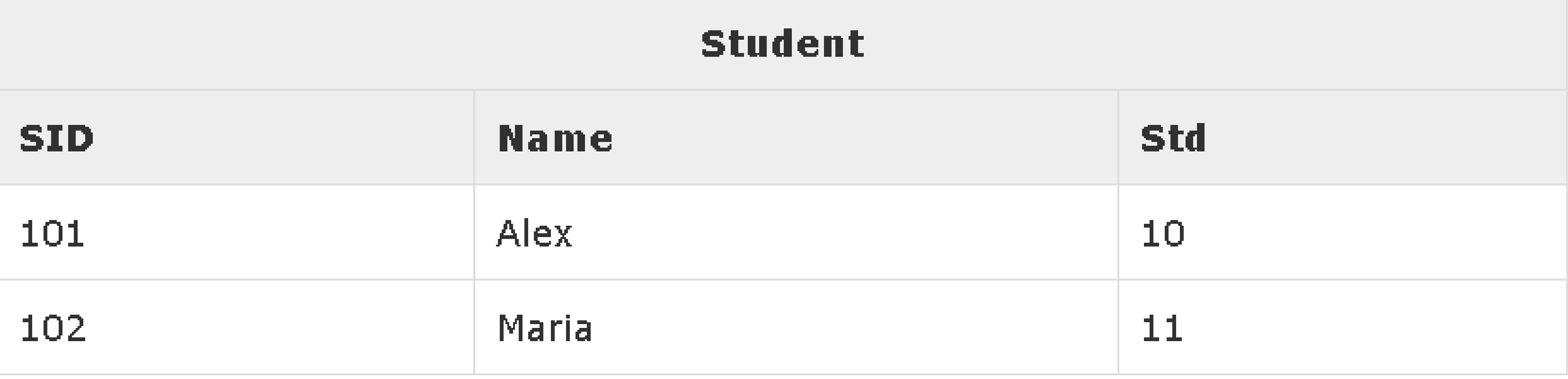
<condition> AND <condition> AND…….. AND<condition>.

- Degree of relation :

Degree(R1 ⋈<join condition> R2) <= Degree (R1) + Degree (R2)

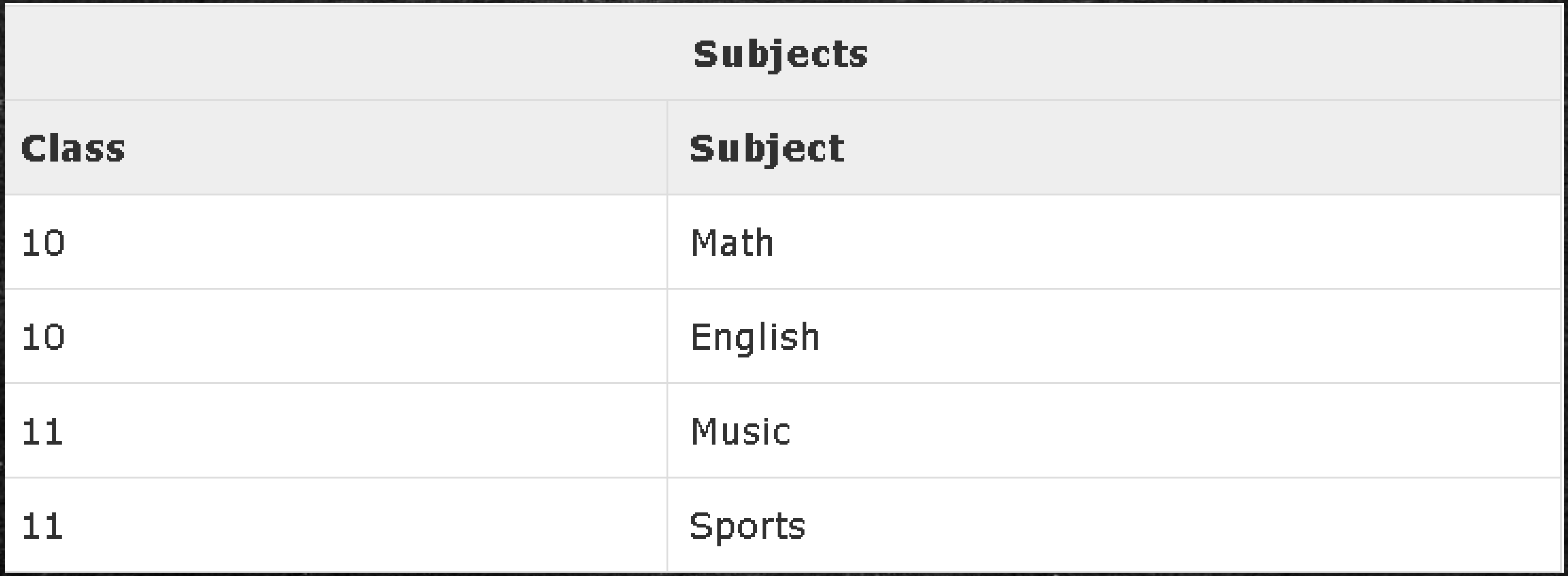
- Three types of joins are :

1. Theta join : When each condition is of the form AθB, A is an attribute of R1 and B is an attribute of R2 and have the same domain, and θ is one of the comparison operators {≥,≤,≠,=,<,>}.

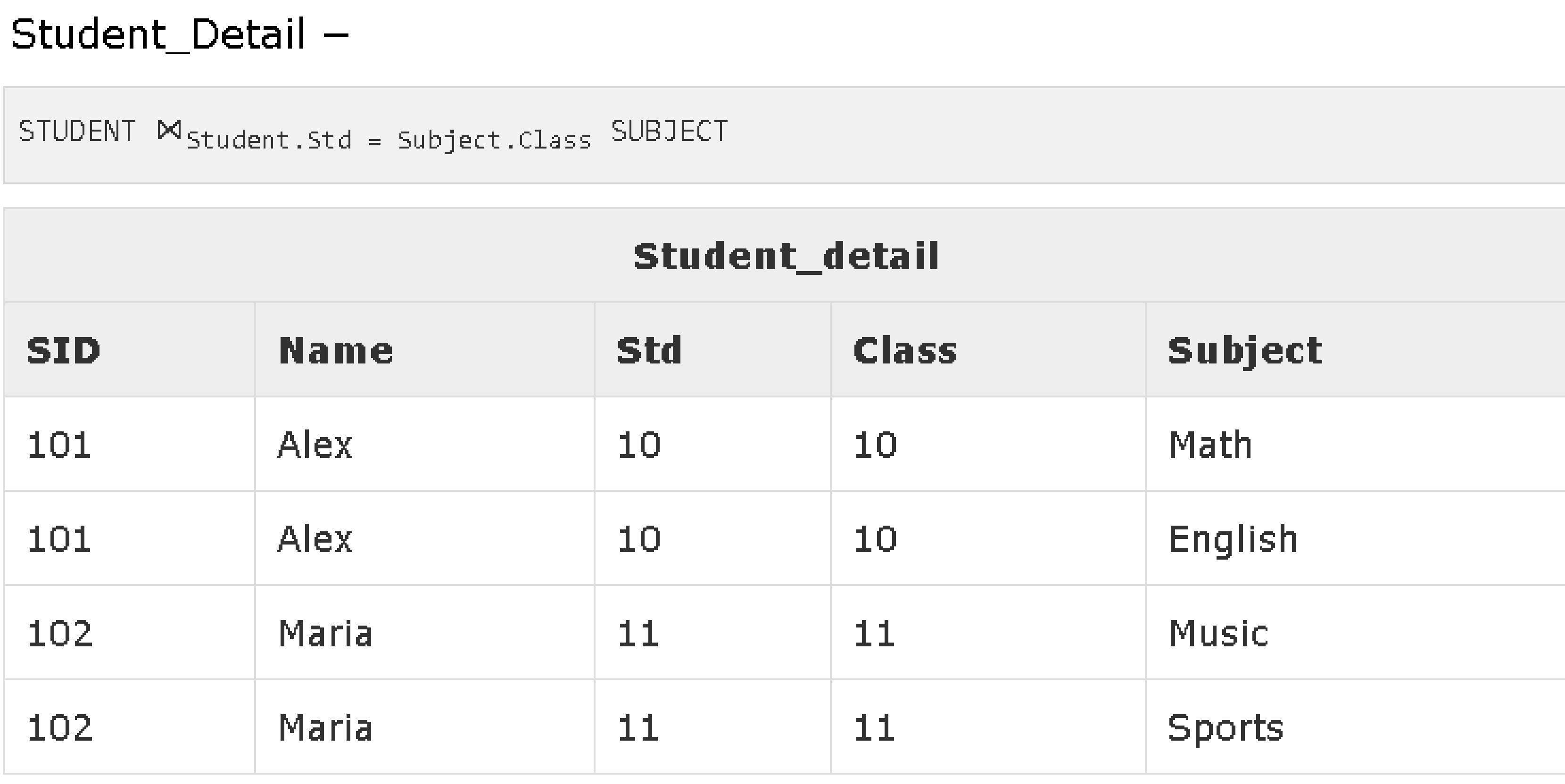


Student

SID	Name	Std
101	Alex	10
102	Maria	11



Subjects	
Class	Subject
10	Math
10	English
11	Music
11	Sports



Student_Detail –

$\text{STUDENT} \bowtie_{\text{Student.Std = Subject.Class}} \text{SUBJECT}$

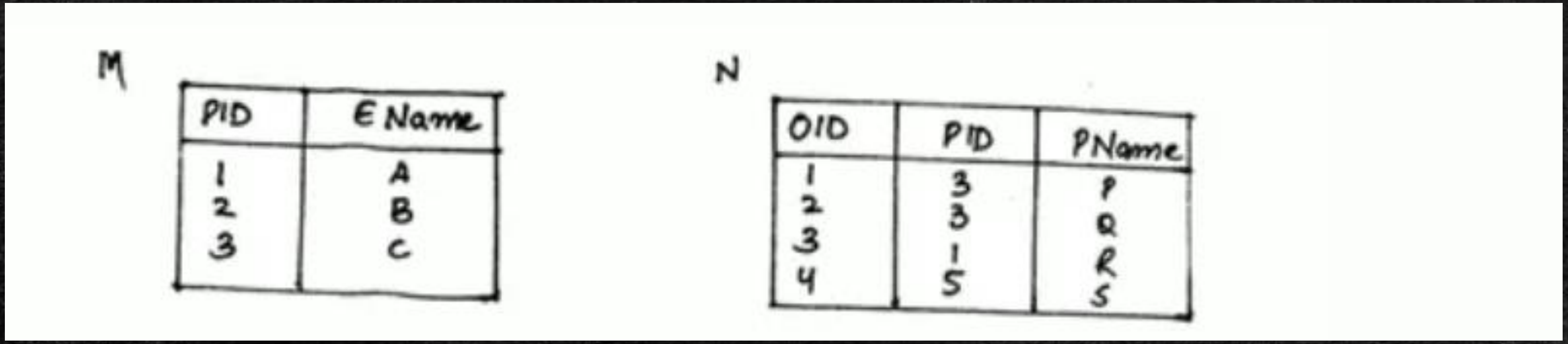
Student_detail

SID	Name	Std	Class	Subject
101	Alex	10	10	Math
101	Alex	10	10	English
102	Maria	11	11	Music
102	Maria	11	11	Sports

2. Equijoin : When each condition appears with equality condition (=) only.

Syntax : SELECT col1,col2,col3 from table1,table2 where table1.col1 = table2.col2

Example :

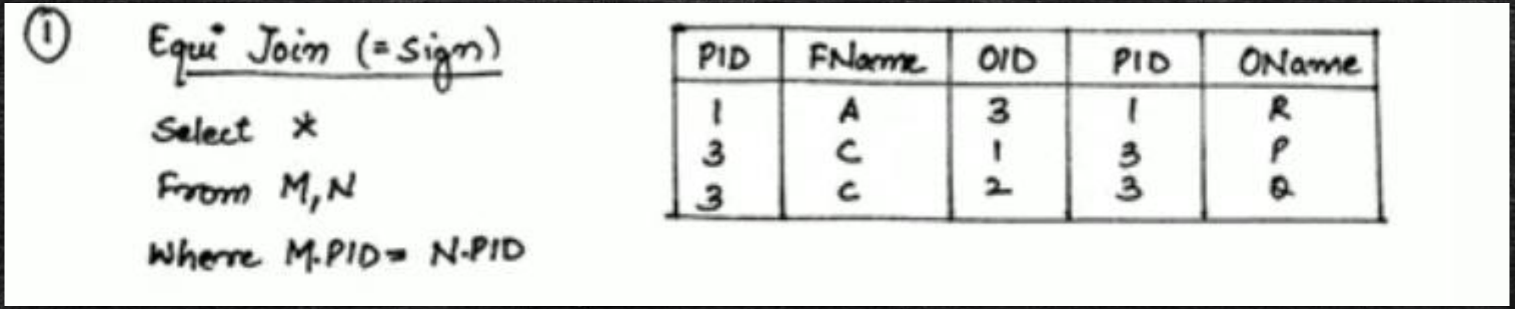


M

PID	EName
1	A
2	B
3	C

N

OID	PID	PName
1	3	P
2	3	Q
3	1	R
4	5	S



① Equi Join (= sign)

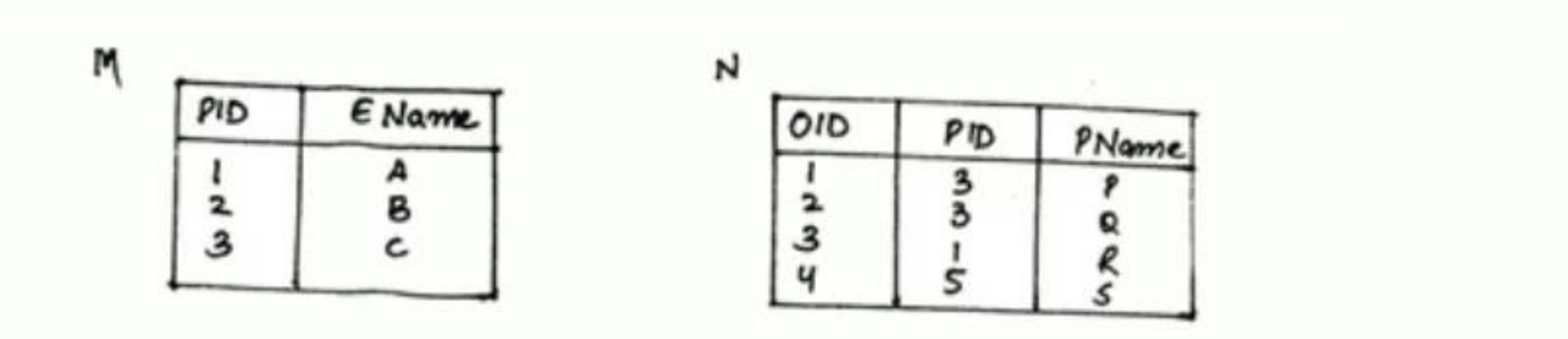
```
Select *
From M,N
Where M.PID= N.PID
```

PID	FName	OID	PID	OName
1	A	3	1	R
3	C	1	3	P
3	C	2	3	Q

3. Natural join : The natural join is a type of Equi Join and is structured in such a way that, columns with same name of associated tables will appear once only. Its denoted by R * S.

Syntax : SELECT * FROM table1 NATURAL JOIN table2;

Example :

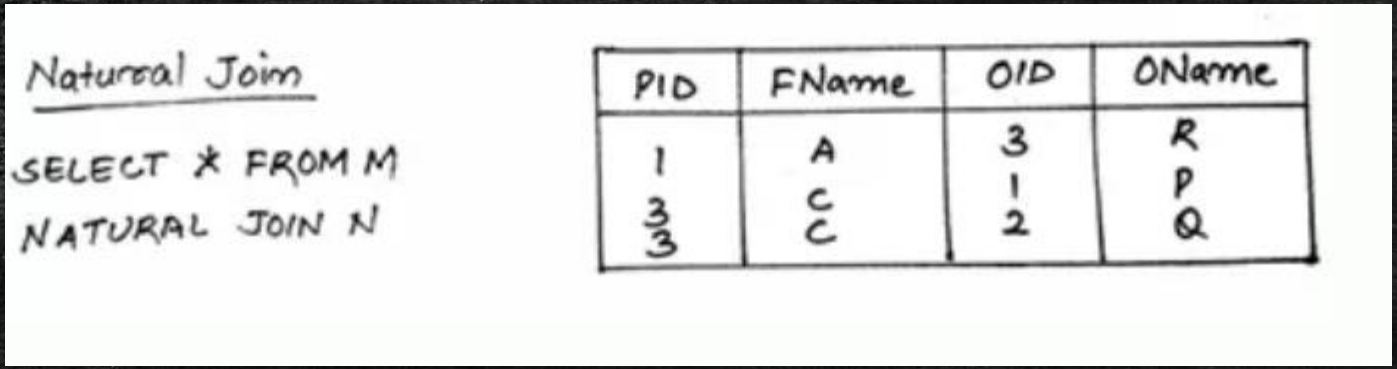


M

PID	EName
1	A
2	B
3	C

N

OID	PID	PName
1	3	P
2	3	Q
3	1	R
4	5	S



Natural Join

SELECT * FROM M
NATURAL JOIN N

PID	FName	OID	OName
1	A	3	R
3	C	1	P
3	C	2	Q

4. Self join : It's a join operation where the table is joined with itself. n this case, it is necessary to ensure that the join statement defines an ALIAS name for both the of copies of the tables to avoid column ambiguity.

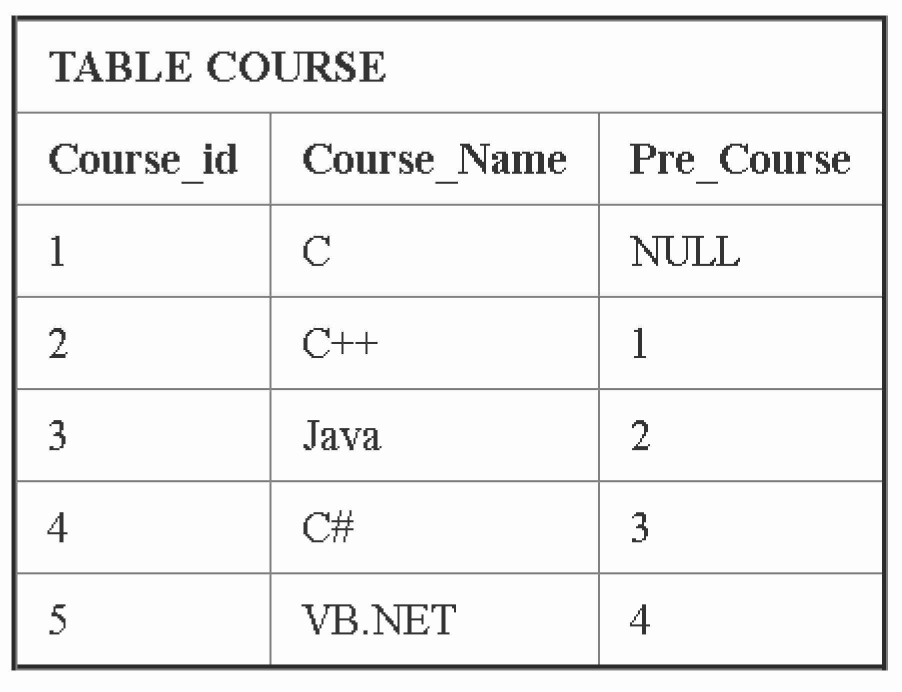
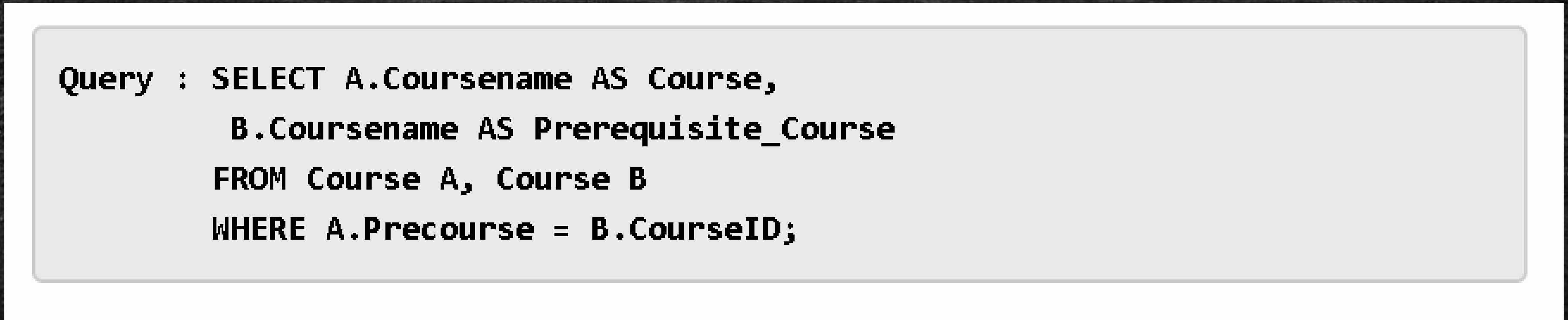
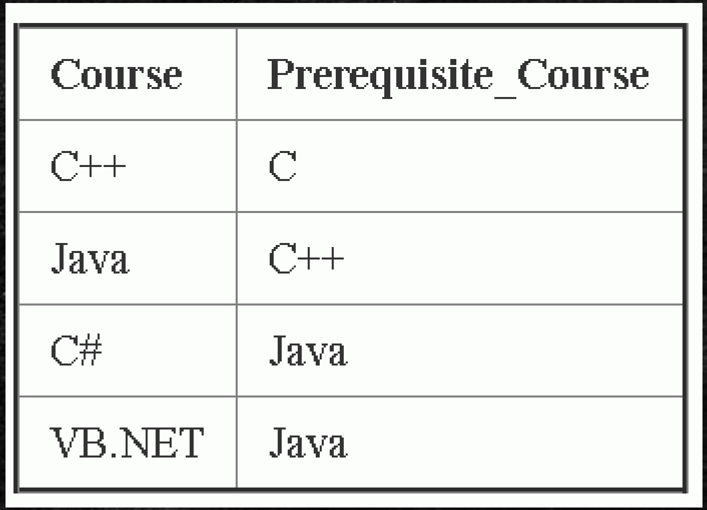


TABLE COURSE

Course_id	Course_Name	Pre_Course
1	C	NULL
2	C++	1
3	Java	2
4	C#	3
5	VB.NET	4



```
Query : SELECT A.Coursename AS Course,
         B.Coursename AS Prerequisite_Course
        FROM Course A, Course B
        WHERE A.Precourse = B.CourseID;
```



Course	Prerequisite_Course
C++	C
Java	C++
C#	Java
VB.NET	Java

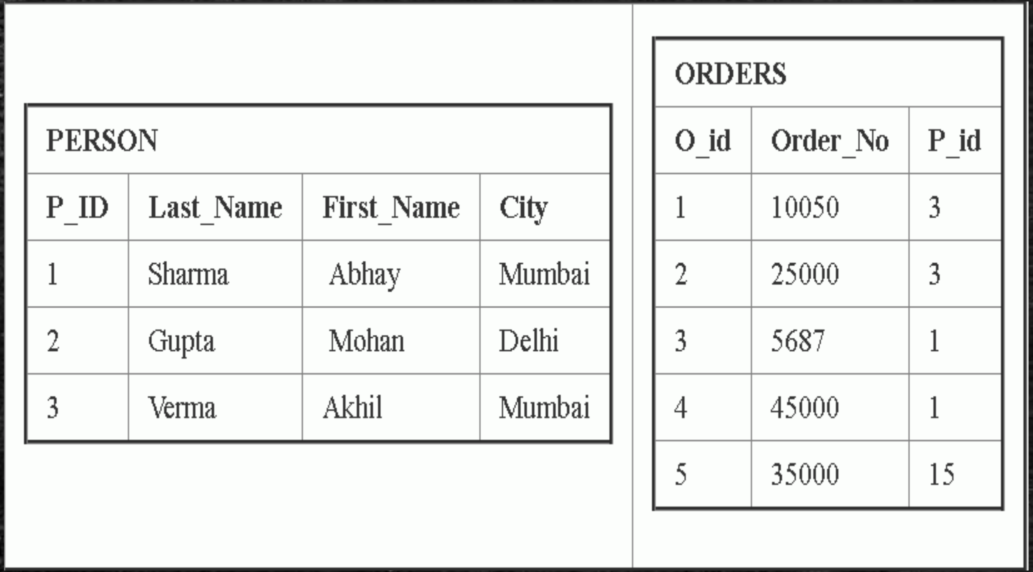
5. Outer join : The use of outer join is that it even joins those tuples that do not have matching values in common columns are also included in the result table. Outer join places null values in columns where there is not a match between tables. A condition with an outer join cannot use the IN operator or cannot be linked to another condition by the OR operator.

   Types of outer join : left outer join, right outer join and complete outer join.

Left outer join : The left outer join returns all the rows from the left table, even if there are no matches in the right table.

Syntax : SELECT col1,col2 FROM table1 LEFT JOIN table2 ON table1.col_name=table2.col_name;

Example :



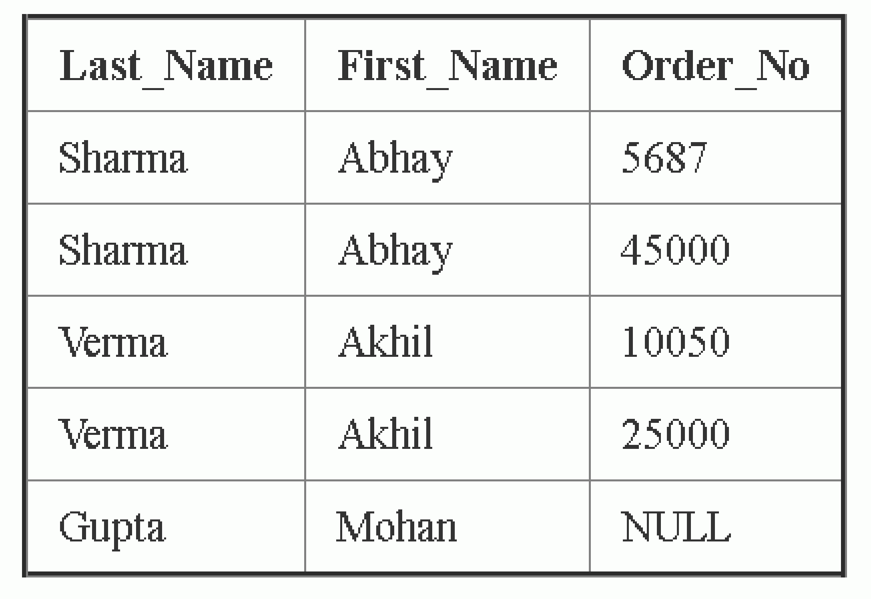
PERSON

P_ID	Last_Name	First_Name	City
1	Sharma	Abhay	Mumbai
2	Gupta	Mohan	Delhi
3	Verma	Akhil	Mumbai

ORDERS

O_id	Order_No	P_id
1	10050	3
2	25000	3
3	5687	1
4	45000	1
5	35000	15

SELECT P.Lname, P.Fname, O.Orderno FROM PERSON P LEFT JOIN Orders O ON P.PID = O.PID ORDER BY P.Fname



Last_Name	First_Name	Order_No
Sharma	Abhay	5687
Sharma	Abhay	45000
Verma	Akhil	10050
Verma	Akhil	25000
Gupta	Mohan	NULL

Right-outer join : The right outer join return all rows from the right table, even if there are no matches in the left table.

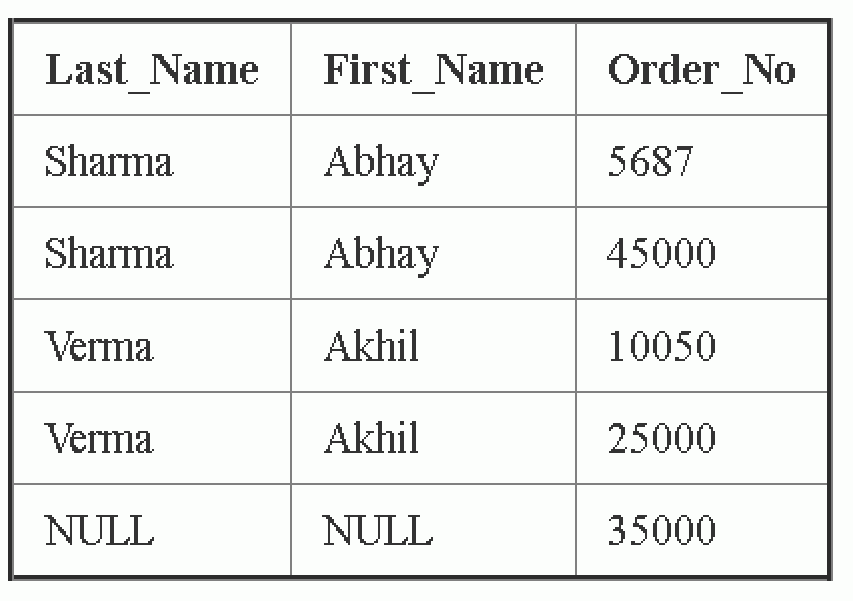
Syntax : SELECT col1,col2 FROM table1 RIGHT JOIN table2 ON table1.col_name=table2.col_name;

Example :

SELECT P.Lname, P.Fname, O.Orderno FROM

Person P RIGHT JOIN Order O ON P.PID = O.PID

ORDER BY P.Lname;

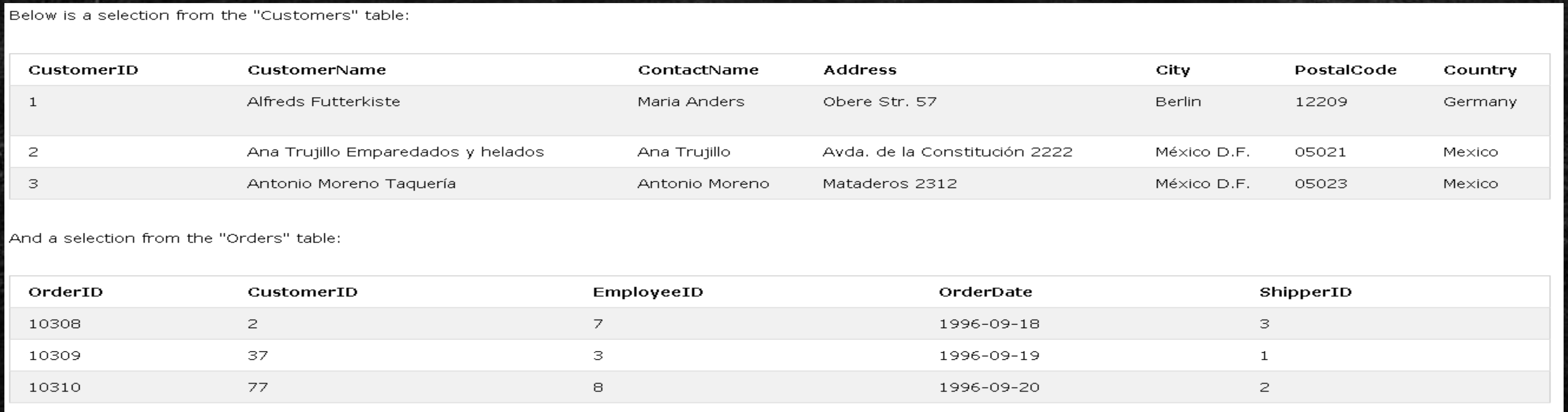


Last_Name	First_Name	Order_No
Sharma	Abhay	5687
Sharma	Abhay	45000
Verma	Akhil	10050
Verma	Akhil	25000
NULL	NULL	35000

Complete outer join : The COMPLETE OUTER JOIN keyword return all records when there is a match in either left (table1) or right (table2) table records. Also known as full outer join.

Syntax : SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2 ON table1.column_name = table2.column_name;

Example :



Below is a selection from the "Customers" table:

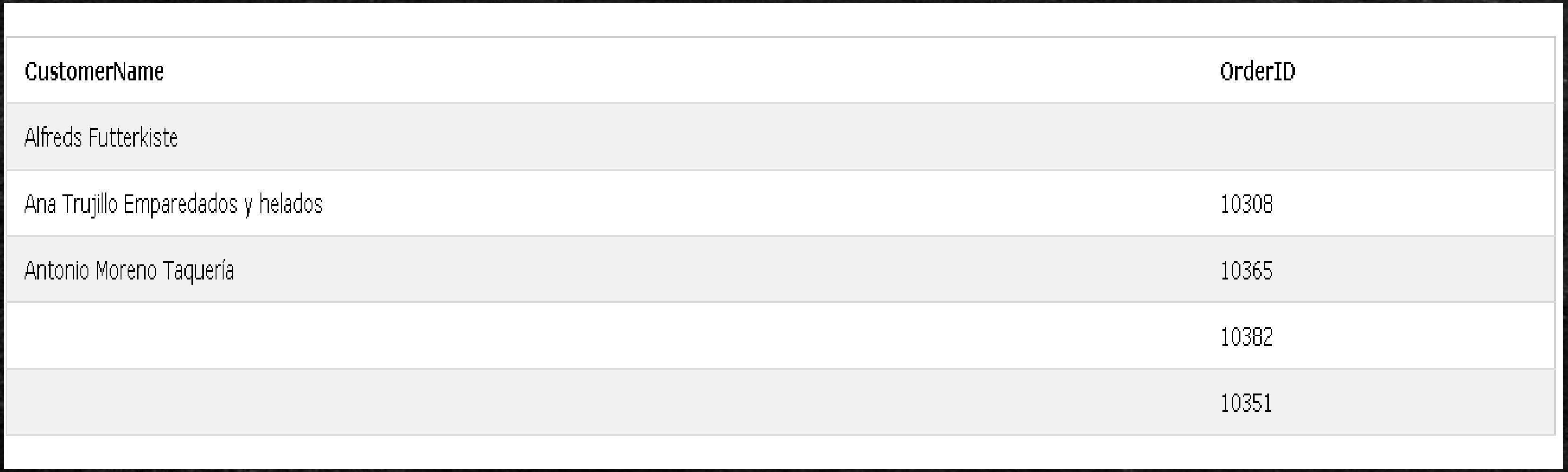
CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

And a selection from the "Orders" table:

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	2	7	1996-09-18	3
10309	37	3	1996-09-19	1
10310	77	8	1996-09-20	2

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
```

Result :

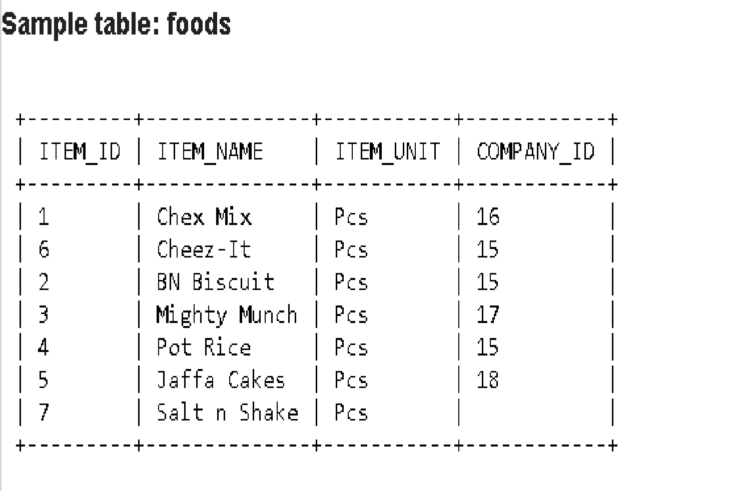


CustomerName	OrderID
Alfreds Futterkiste	
Ana Trujillo Emparedados y helados	10308
Antonio Moreno Taquería	10365
	10382
	10351

6. Inner join : The Inner joins returns us the rows which are common in both the tables. i.e. gives the intersection of two tables.

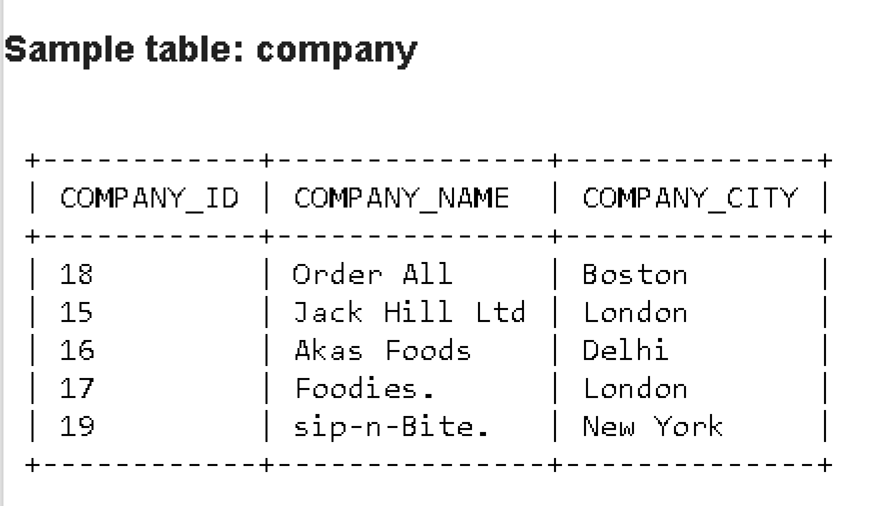
Syntax :

SELECT col1,col2 FROM table1 INNER JOIN table2 ON table1.column_name = table2.column_name;



Sample table: foods

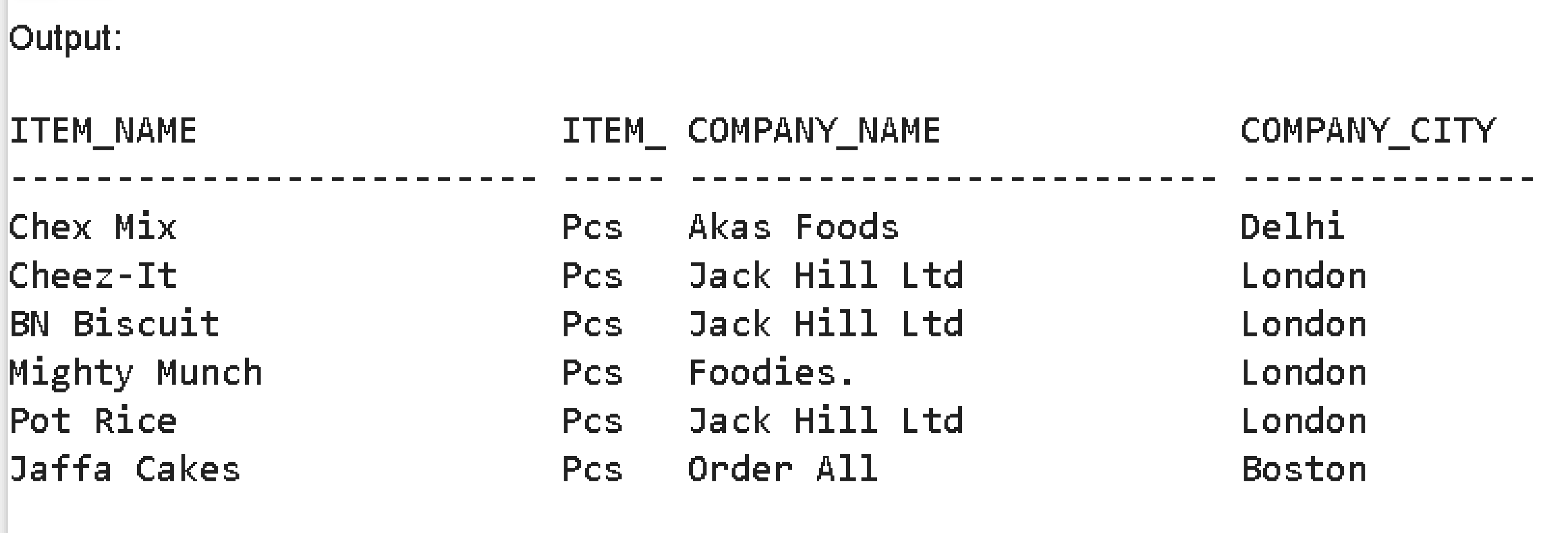
```
+---------+--------------+-----------+------------+
| ITEM_ID | ITEM_NAME    | ITEM_UNIT | COMPANY_ID |
+---------+--------------+-----------+------------+
| 1       | Chex Mix     | Pcs       | 16         |
| 6       | Cheez-It     | Pcs       | 15         |
| 2       | BN Biscuit   | Pcs       | 15         |
| 3       | Mighty Munch | Pcs       | 17         |
| 4       | Pot Rice     | Pcs       | 15         |
| 5       | Jaffa Cakes  | Pcs       | 18         |
| 7       | Salt n Shake | Pcs       |            |
+---------+--------------+-----------+------------+
```



Sample table: company

```
+------------+---------------+--------------+
| COMPANY_ID | COMPANY_NAME  | COMPANY_CITY |
+------------+---------------+--------------+
| 18         | Order All     | Boston       |
| 15         | Jack Hill Ltd | London       |
| 16         | Akas Foods    | Delhi        |
| 17         | Foodies.      | London       |
| 19         | sip-n-Bite.   | New York     |
+------------+---------------+--------------+
```

SELECT foods.item_name,foods.item_unit, company.company_name,company.company_city
FROM foods INNER JOIN company ON foods.company_id =company.company_id;

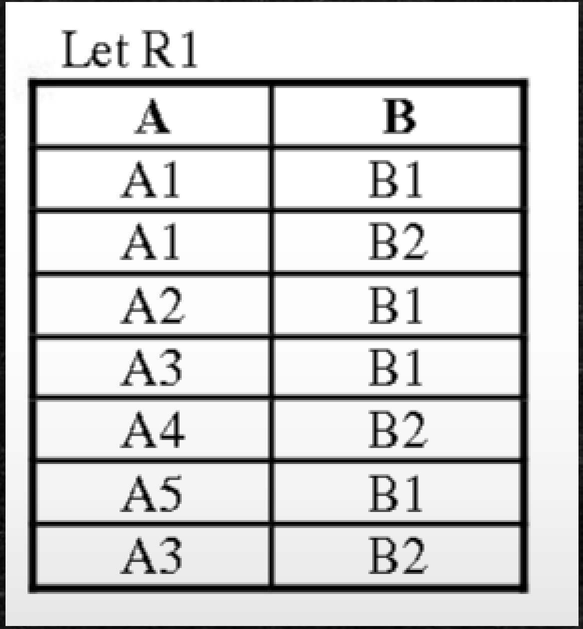


Output:

```
ITEM_NAME                 ITEM_ COMPANY_NAME              COMPANY_CITY
------------------------- ----- ------------------------- --------------
Chex Mix                  Pcs   Akas Foods                Delhi
Cheez-It                  Pcs   Jack Hill Ltd             London
BN Biscuit                Pcs   Jack Hill Ltd             London
Mighty Munch              Pcs   Foodies.                  London
Pot Rice                  Pcs   Jack Hill Ltd             London
Jaffa Cakes               Pcs   Order All                 Boston
```

4. Division operator : To perform division operation R1 ÷ R2, R2 should be a proper subset of R1. In the example below, R1 contains attributes A and B and R2 contains only attribute B so R2 is a proper subset of R1. If we perform R1 ÷ R2 then the resultant relation will contain those values of A from R1 that are related to all values of B present in R2.

Example :



Let R1

A	B
A1	B1
A1	B2
A2	B1
A3	B1
A4	B2
A5	B1
A3	B2

If R2 is :

B
B1
B2

R3 = R1 ÷ R2

A
A1
A3

If R2 is :

B
B1

R3=R1 ÷ R2

A
A1
A2
A3
A5

If R2 is :

A

R3=R1 ÷ R2

B
B1
B2
B3

If R2 is :

B

R3=R1 ÷ R2

A
A1
A2
A3
A4
A5

Entity Relationship (ER) Model

- Main features of ER model :
- It's a high-level conceptual data model.
- It describes the data involved in a real-world enterprise in terms of objects and their relationships.
- Its widely used to develop an initial design of a database.
- It provides a set of useful concepts that make it convenient for a developer to move from a basic set of information to a detailed and precise description of information that can be easily implemented in a database system.
- It describes data as a collection of entities, relationships and attributes.

1. Entities :

- An entity is an object of concern used to represent the things in the real world, e.g., car, table, book etc..

- Entity need not be a physical entity, it can also be a concept in real world, e.g. project, loan etc..

- It represents a class of things, not any one instance.

Entity set :

A collection of similar kind of entities is called an entity set or entity type.

Collection of student entities is an entity set.

Entity identifier key attributes :

- An entity type usually has an attribute whose values are distinct for each individual entity in the collection. Such an attribute is called key attribute. Its values are used to identify each entity uniquely.

Strong entity set :

- The entity type containing a key attribute is called a strong entity types or regular entity types.
- Example : the student entity has a key attribute RollNo which uniquely identifies it, hence it’s a strong entity.

2. Attributes :

- An attribute is a property used to describe the specific feature of the entity.
- For example : a student entity has the attributes name, age, address, etc..

Domains : Each simple attribute of an entity type contains a possible set of values that can be attached to it. This is called domain of an attribute. An attribute cannot contain value outside its domain.

Types of attributes :

Simple : Attribute that cannot be further divided into smaller parts and represents the basic meaning is called a simple attribute.

Example : First name, last name , age are simple attributes.

Composite :

Attributes that can be further divided into smaller units and each individual unit contains a specific meaning.

Example : Name attribute of an employee entity can be sub-divided into first name, last name and middle name.

Single valued :

Attributes having a single value for a particular entity.

Example : Age is a single valued attribute.

Multivalued :

Attributes that have more than one values for a particular entity is called a multivalued attribute. Different entities can have different number of values for these kind of attributes. We must also specify the minimum and maximum number of values that can be attached.

Example : Phone number is a multivalued attribute.

Stored :

Attributes that are directly stored in the data base.

Example : Birth date of a person.

Derived :

Attributes that are not stored directly but can be derived from stored attributes are called derived attributes.

Example : The years of services of a person entity can be calculated from the current date and the date of joining of the person.

3. Relationships :

A relationship can be defined as :

- a connection or set of associations, or
- a rule for communication among entities

Example : In the college database, the association between student and course entity, i.e., “Student opts course” is a relationship.

Relationship set :

It’s a set of relationships of the same type. Also known as relationship type.

Example : Consider the relationship between two entity sets student and course. Collection of all the instances of relationship opts forms a relationship set called relationship type.

Degree : Degree of a relation type is the number of participating entity types.

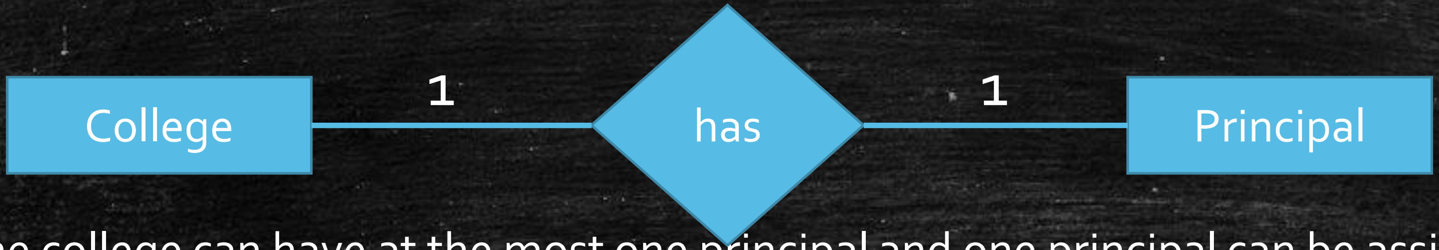
Relationship between two entities is called binary relationship. Relationship between three entities is called ternary relationship.

Relationship between n entities is called n-ry relationship.

Relationship cardinality :

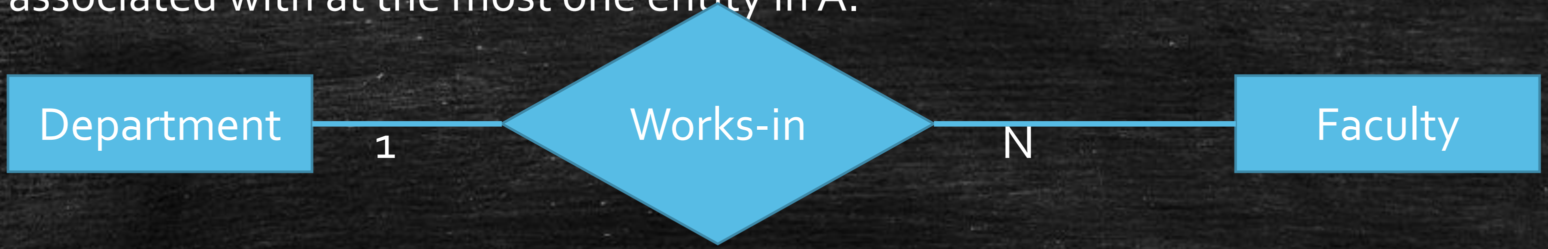
Cardinality specifies the number of instances of an entity associated with another entity participating in a relationship. Based on cardinality binary relationship can be divided into :

- One-to-one : An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.



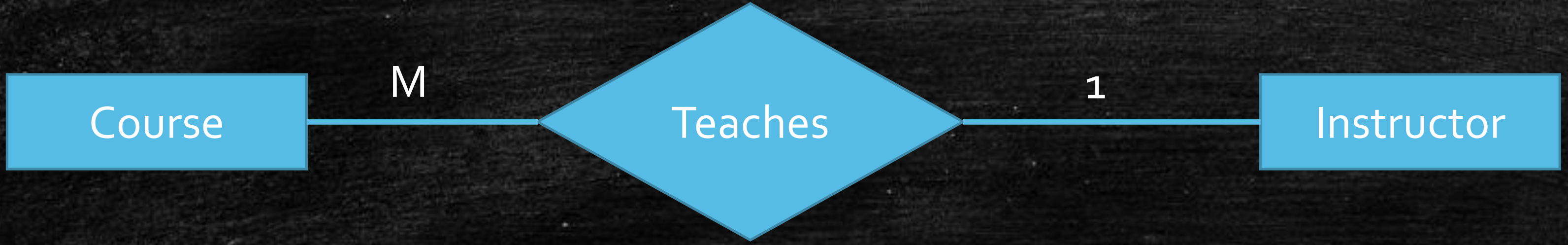
One college can have at the most one principal and one principal can be assigned to only one college.

- One-to-many : An entity in A is associated with any number of entities in B. An entity in B is associated with at the most one entity in A.



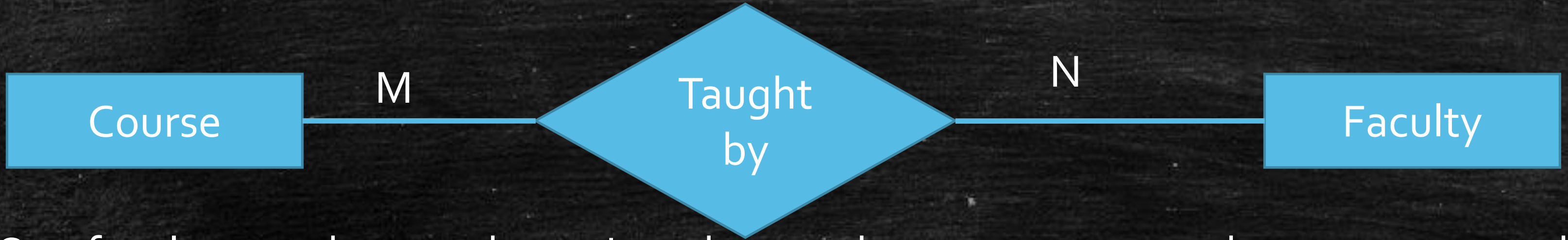
One department can appoint any number of faculty members but a faculty member can be assigned to only one department.

- Many-to-one : An entity in A is associated with at most one entity in B. An entity in B is associated with any number in A.



An instructor can teach various courses but a course can be taught only by one instructor. This is just an assumption.

- Many-to-many :Entities in A and B are associated with any number of entities from each other.



One faculty member can be assigned to teach many courses and one course may be taught by many faculty members.

Recursive relationships :

When the same entity type participates more than once in a relationship type in different roles, the relationship types are called recursive relationships.

Participation constraints :

The participation constraints specify whether the existence of an entity depends on its being related to another entity via the relationship type. There are two types of participation constraints :

Total : When all the entities from an entity set participate in a relationship type, is called total participation.

Example : The participation of the entity set student in the relationship set must 'opts' is said to be total because every student enrolled must opt for a course.

Partial : When it is not necessary for all the entities from an entity set to participate in a relationship type, it is called partial participation.

Example : The participation of the entity set student in 'represents' is partial , since not every student in a class is a class representative.

Weak entity :

- Entity types that do not contain any key attribute, and hence cannot be identified independently, are called week entity types. In a relational database, a **weak entity** is an **entity** that cannot be uniquely identified by its attributes alone; therefore, it must use a foreign key in conjunction with its attributes to create a primary key.

- The following restrictions must hold :

  - The owner entity set and the weak entity set must participate in one-to-many relationship set. This relationship is called the identifying relationship set of the weak entity set.

- The weak entity set must have total participation in the identifying relationship.

Extended E-R features :

Three enhanced features are :

1. Generalization
2. Specialization
3. Aggregation

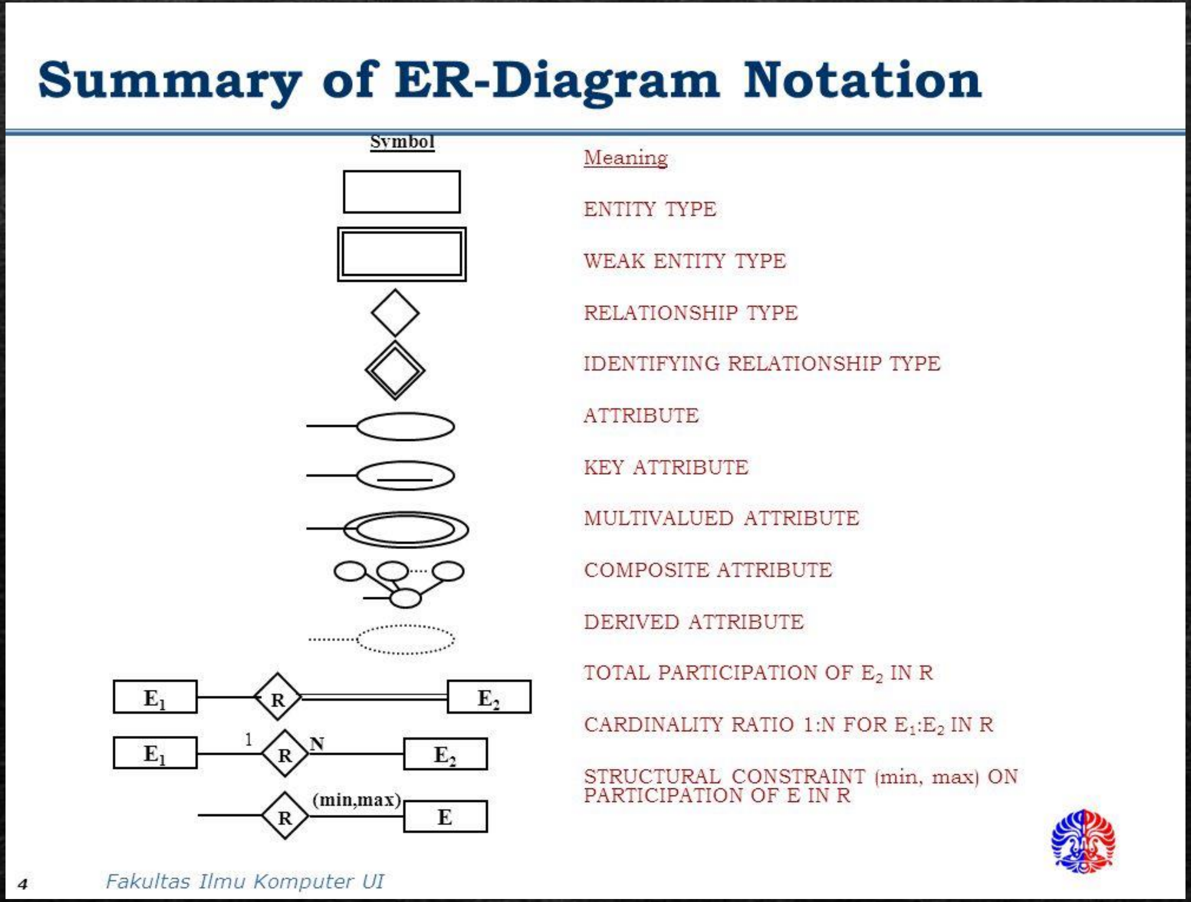
E-R Diagram

- E-R diagram it expresses the overall logical structure of a database using ER model graphically with the help of an E-R diagram.
- ER diagram consists of :

○ Rectangles : representing entity sets.

○ Ellipses : representing attributes.

○ Diamonds : representing relationship sets.



Summary of ER-Diagram Notation

Symbol

Meaning

ENTITY TYPE

WEAK ENTITY TYPE

RELATIONSHIP TYPE

IDENTIFYING RELATIONSHIP TYPE

ATTRIBUTE

KEY ATTRIBUTE

MULTIVALUED ATTRIBUTE

COMPOSITE ATTRIBUTE

DERIVED ATTRIBUTE

TOTAL PARTICIPATION OF E_2 IN R

CARDINALITY RATIO 1:N FOR E_1:E_2 IN R

STRUCTURAL CONSTRAINT (min, max) ON PARTICIPATION OF E IN R

Conversion of ER Diagram to Relational Database

- To convert the ER model to relational model there are 9 steps to be followed, which are:
- Conversion of entity sets :

1. Conversion of Strong Entities
2. Conversion of Weak Entities

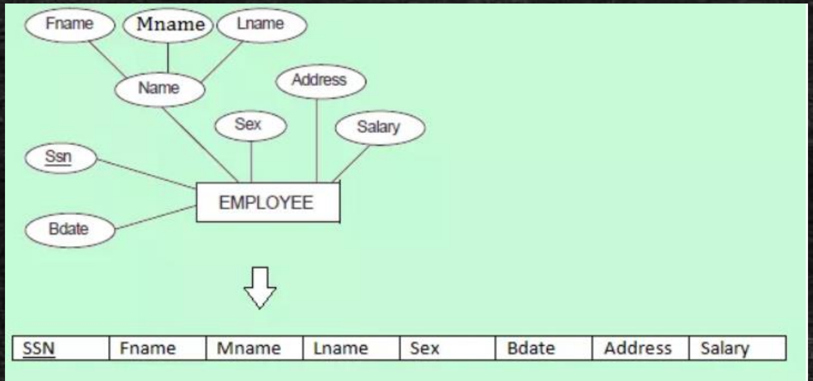
- Conversion of relationship sets :

3. Conversion of one to one Relationships
4. Conversion of One to Many Relationships
5. Conversion of Many to Many Relationships
6. Conversion of Many to Many Relationships
7. Conversion of n-ary Relationships
8. Conversion of Multivalued Attribute
9. Converting generalization/ specialization

1. Conversion of Strong Entities

For each strong entity type E in the ER diagram, we create a relation R containing all the simple attributes of E. The primary key of the relation R will be one of the key attributes of R.

Example :

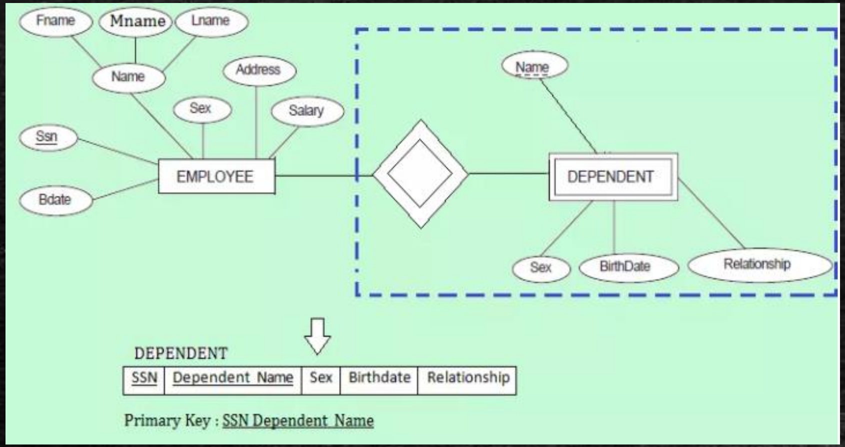


SSN	Fname	Mname	Lname	Sex	Bdate	Address	Salary

2. Conversion of Weak Entities

For each weak entity type W in the E R Diagram, we create another relation R that contains all simple attributes of W. If E is an owner entity of W then key attribute of E is also included in R.s

The key attribute of R is set as a foreign key attribute of R. The combination of primary key attribute of owner entity type and partial key of weak entity type will form the key of the weak entity type.



DEPENDENT

SSN	Dependent_Name	Sex	Birthdate	Relationship

Primary Key : SSN Dependent_Name

3. Conversion of One to One Relationships

For each 1:1 relationship type R in the ER diagram involving two entities E1 and E2 we choose one of entities preferably with total participation and add primary key attribute of another attribute of another entity E2 as a foreign key attribute in the table of entity (E1). We include all the simple attributes of relationship type R in E1 if any.

4. Conversion of one-to-many relationships

For each 1:n relationship type R involving two entities E1 and E2, we identify the entity type at the n-side of the relationship type R and include primary key of the entity on the other side of the relation as a foreign key attribute in the table of E1. We include all the simple attributes of R in E1.

5. Conversion of many-to-many

For each m:n relationship type R, we create a new table to Represent R. We also include the primary key attributes of both the participating entity types as a foreign key attribute in S. Any simple attribute of the m:n relationship type is also included as attributes of S.

6. Conversion of n-ary relationships

For each n-ary relationship type R where n>2, we create a new table S to represent R. We include foreign key attributes in S the primary keys of the relations that represent the participating entity types. We also include any simple attributes of the n-ary relationship type as attributes of S. The primary key of S is usually a combination of all the foreign keys that reference the relations representing the participating entity types.

7. Conversion of multivalued attributes

For each multivalued attribute 'A', we create a new relation R that includes an attribute corresponding to plus the primary key attribute k of the relation that represents the entity type or relationship type that has an attribute. The primary key of R is then combination of A and k.

8. Converting generalization /specialization :

A simple rule to this conversion may be to decompose all the specialized entities into tables they are disjoint.