Unit-2(Relational Model)

Introduction to Relational Model:

=>The Relational model is the theoretical basis of relational databases.

=>The relational model of data is based on the concept of relations or table.

=>A Relation is a mathematical concept based on the ideas of sets.

=>The Relational model was proposed by E.F .Codd for IBM in 1970 to model data in the form of relations or tables.

Relational Model:

=>Relational model represents how data is stored in Relational Databases. A relational database stores data in the form of relations(tables).

=>After designing the conceptual model of database using ER diagram, we need to convert the conceptual model in the relational model which can be implemented using any RDBMS languages.

=>RDBMS languages: MySql , Oracle, SQL server, DB2

=>RDBMS stands for Relational database management system is based on the relational model.

=>Relational model can be represented as a table with columns and rows.

=> tuple :Each row is known as tuple.

=> attribute :Each table of the column is known as attribute.

=>Relation: table is called Relation.

=>Domain: A domain is the set of allowable values for one or more attributes

=>Degree: The total number of columns or attributes in the relation or table.

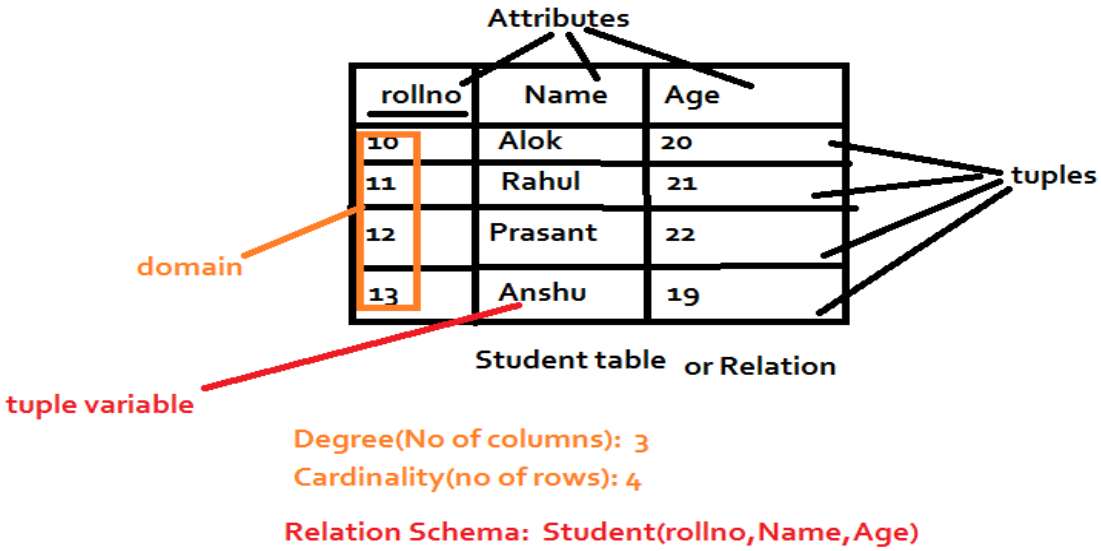
=>Cardinality: Total number of rows present in the table.

=>Relation Schema: A relation schema represents the name of relation with its attributes.

=>Relation instance(state): Relation instance is a finite set of tuples. Relation instance never have duplicates tuples.

=>Relation Key: Every row has one or multiple attributes that can uniquely identify the row the relation,which is called relation key(primary key).

=>Tuple variable: It is the data stored in a record of the table.



Properties of Relational Model:

1. Each Relation has unique name.
2. Each tuple/row is unique : No duplicate row.
3. Entries in any column have the same domain.
4. Each attribute/column has a unique name.
5. Order of the column rows is irrelevant i.e relations are unordered.
6. Each cell of relation contains exactly one value i.e attribute values are required to be atomic.

Integrity Constraints or Relational Constraints or Constraints in DBMS -

- Integrity constraints are the restrictions imposed on the database contents and operations.
- Integrity constraints are used to ensure accuracy and consistency of the data in a relational database.
- Integrity constraints are set of rules that the database is not permitted to violate.
- Integrity constraints may apply to each attribute or they may apply to relationships between tables.
- Integrity constraints ensures that changes (update, delete, insert) made to the database by authorized user do not result in a loss of data consistency.

Thus, Integrity constraints guard against accidental damage to the database.

Types of Integrity Constraints :

1.Domain Constraints
2.Key Constraints
3.Entity Integrity constraint
4.Referential Entity Integrity constraint

1. Domain Constraint-

=> Domain constraint defines the domain or the valid set of values for an attribute.
=>It specifies that the value taken by the attribute must be the atomic value from its domain.

Example-
Consider the following Student table-

rollno	Name	Age
10	Alok	20
11	Rahul	21
12	Prasant	22
13	Anshu	A

Not allowed.Because Age is an integer value

Student table or Relation

2.Key Constraints:

Key constraint specifies that in any relation-

=>All the values of primary key must be unique.

Example-
Consider the following Student table-

This relation does not satisfy the key constraint as here all the values of primary key are not unique.

rollno	Name	Age
10	Alok	20
10	Rahul	21
12	Prasant	22
13	Anshu	19

Student table or Relation

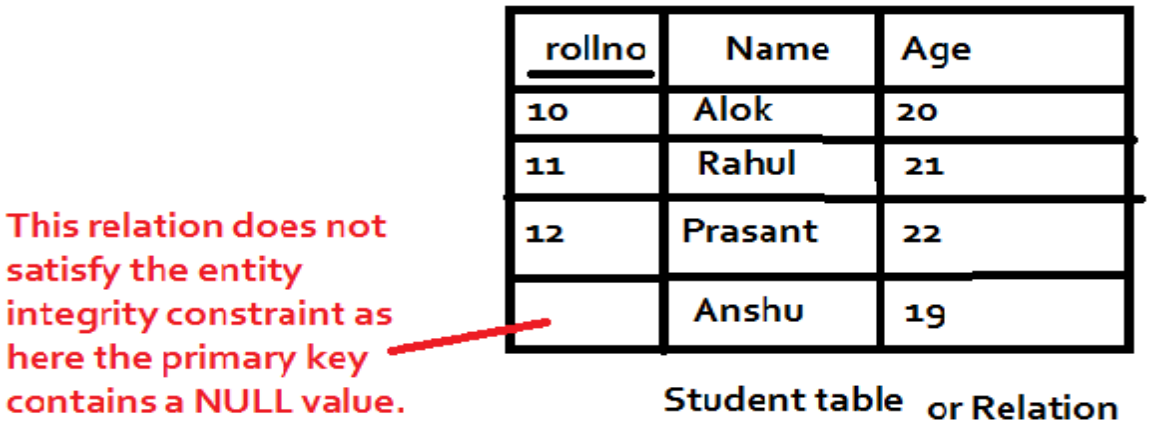
3. Entity Integrity Constraint-

=>Entity integrity constraint specifies that no attribute of primary key must contain a null value in any relation.

=>This is because the presence of null value in the primary key violates the uniqueness property.

Example-

Consider the following Student table-



rollno	Name	Age
10	Alok	20
11	Rahul	21
12	Prasant	22
	Anshu	19

Student table or Relation

4. Referential Integrity Constraint-

=>This constraint is enforced when a foreign key references the primary key of a relation.
=>It specifies that all the values taken by the foreign key must either be available in the relation of the primary key or be null.

Example-

Consider the following two relations- 'Student' and 'Department'.
Here, relation 'Student' references the relation 'Department'.

rollno	Name	Age	Dept_Id
10	Alok	20	D10
11	Rahul	21	D10
12	Prasant	22	D11
13	Anshu	19	D14

Student table or Relation

Dept_id	Dept_name
D10	MCA
D11	CS
D12	ECE
D13	ME

Department table

Here,
The relation 'Student' does not satisfy the referential integrity constraint.
This is because in relation 'Department', no value of primary key specifies department no. 14.
Thus, referential integrity constraint is violated.

Query Languages:

Query Language is language in which user requests information from the database.

Types of Query Language:

1.Procedural Query Language

2.Non-Procedural Query Language

1.Procedural Query Language:

=>In Procedural Query language, user instructs the system to perform series of operation to produce the desired results.

=>User tells what data to be retrieved from database and how to retrieve it.

2.Non-Procedural (or Declarative)Query Language:

=>In Non-procedural query language,user instructs the system to produce the desired result without telling the step by step process.

=>User tells what data to be retrived from database but does not tell how to retrieve it.

Two Pure Query languages or Two Mathematical Query language

1.Relational Algebra

2.Relational Calculus:

a. Tuple Relational Calculus

b.Domain Relational Calculus

1.Relational Algebra:

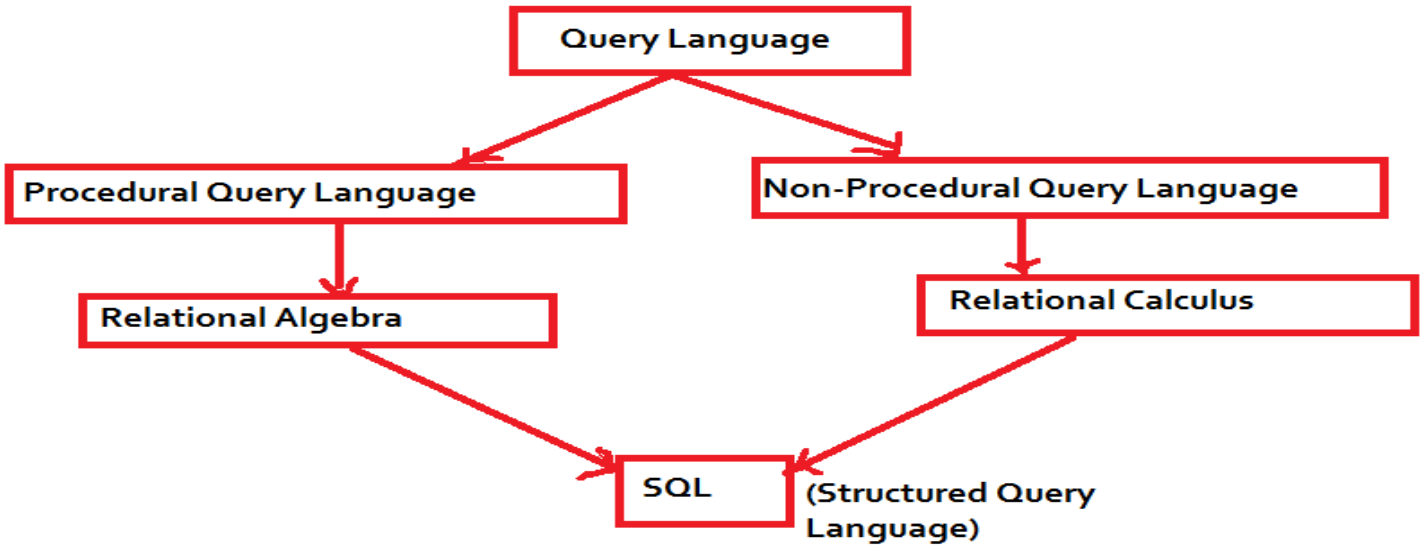
=>Relational Algebra is a Procedural Query language.
=>It is more operational, very usefull for representing execution plan.
=>Procedural: what data is required and how to get those data.

2.Relational Calculus: This is two type

a. Tuple Relational Calculus

b.Domain Relational Calculus

=>Relational Calculus is a Non-Procedural language.
=>It is non-operational or Declarative
=> **Non-Procedural** :what data to be retrived from database but does not tell how to retrieve it.

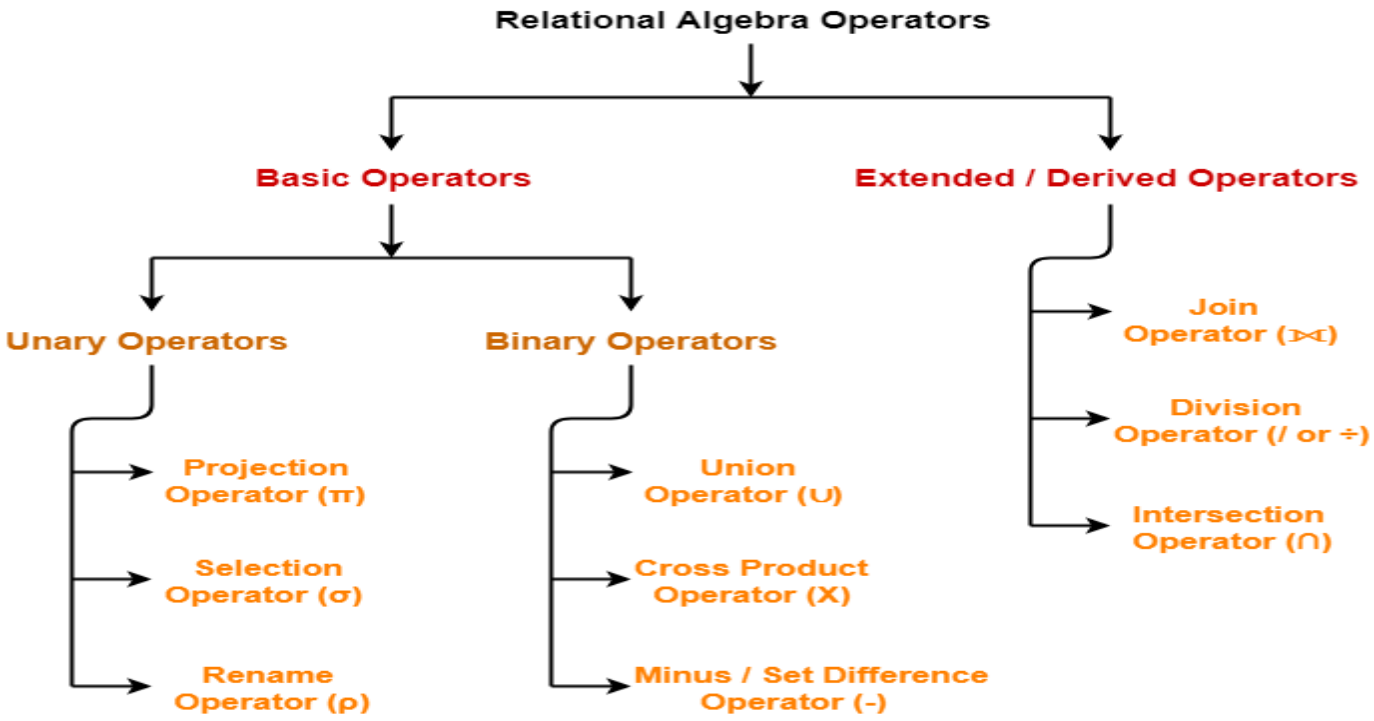


Relational Algebra:

Relational Algebra is a procedural query language which takes a relation as an input and generates a relation as an output.

It uses operators to perform queries. An operator may be unary or binary.

Relational Algebra Operators-



Characteristics-

Following are the important characteristics of relational operators-
=> Relational Operators always work on one or more relational tables.
=>Relational Operators always produce another relational table.
=>The table produced by a relational operator has all the properties of a relational model.

Selection Operator((σ))-

=>Selection Operator denoted by sigma (σ) is a unary operator in relational algebra that performs a selection operation.

=>It selects those rows or tuples from the relation that satisfies the selection condition.

Syntax- : $\sigma_P(R)$

Here, σ represents select predicate

R for relation

P for proposition logics like – $=, \neq, \geq, <, >, \leq$. with us of connectors like OR, AND, or NOT .

or

$\sigma_{<selection\_condition>}$(Relation name)

Example:

Select tuples from a relation “Books” where subject is “database”

$\sigma_{subject = \text{“database”}}$ (Books)

Select tuples from a relation “Books” where subject is “database” and price is “450”

$\sigma_{subject = \text{“database”} \wedge price = \text{“450”}}$ (Books)

Select tuples from a relation “Books” where subject is “database” and price is “450” or have a publication year after 2010

$\sigma_{subject = \text{“database”} \wedge price = \text{“450”} \vee year > \text{”2010"}}$ (Books)

Important Points-

Point-01:

We may use logical operators like $\wedge$, $\vee$, ! and relational operators like = , ≠ , > , < , <= , >= with the selection condition.

Point-02:

=> Selection operator only selects the required tuples according to the selection condition.

=>It does not display the selected tuples.

=>To display the selected tuples, projection operator is used.

Point-03:

=>Selection operator always selects the entire tuple. It can not select a section or part of a tuple.

Point-04:

=>The **where** clause of a **SQL Command** corresponds to relational algebra selection operator.

Ex: SQL: **Select * from R where condition.**

Example:

rollno	Name	Age
10	Alok	20
11	Rahul	21
12	Prasant	22
13	Anshu	19

Student table or Relation

Query: Select Student whose rollno is 12.

$\sigma_{rollno=12}(Student)$

rollno	Name	Age
12	Prasant	22

Query: Select Student whose name is Rahul.

$\sigma_{name=Rahul}(Student)$

rollno	Name	Age
11	Rahul	21

Query: Select Student whose age is greater than 20

$\sigma_{age>20}(Student)$

rollno	Name	Age
11	Rahul	21
12	Prasant	22

Projection Operator(π):

- Projection Operator denoted by PIE (π) is a unary operator in relational algebra that performs a projection operation.

- It displays the columns of a relation or table based on the specified attributes.
- It delete columns that are not in projection list.
- Duplicate rows are automatically eliminated from result.

Syntax- $\pi_{<attribute\ list>}(R)$ or $\prod_{A1,A2,...An}(R)$

Here,

1. ∏ represents Project predicate
2. R for relation
3. A1, A2, A3 for selection from columns for projection

The SQL command corresponds to relational project operator is

SQL: SELECT A1,A2,……An FROM R;

Example- Consider the following student relation

ID	Name	Subject	Age
100	Ashish	Maths	19
200	Rahul	Science	20
300	Naina	Physics	20
400	Sameer	Chemistry	21

Student

Ex- Display the Name and age from student relation.

Query: $\pi_{Name,\ Age}(Student)$

Result of Query:

Name	Age
Ashish	19
Rahul	20
Naina	20
Sameer	21

Ex: Display the Id and Name from student relation.

Query: $\pi_{ID, Name}(Student)$

Result:

ID	Name
100	Ashish
200	Rahul
300	Naina
400	Sameer

Ex: Display or project the id and name of students whose age is greater than 19.

Query:

$$\pi_{ID, Name}(\sigma_{age>19}(Student))$$

Result:

ID	Name
200	Rahul
300	Naina
400	Sameer

Note:

There is only one difference between projection operator of relational algebra and SELECT operation of SQL.

Projection operator does not allow duplicates while SELECT operation of SQL allows duplicates.

To avoid duplicates in SQL, we use “distinct” keyword and write SELECT distinct.

Set Theory Operators-

Union, intersection and difference, are binary operators as they takes Two input relations.

To use set theory operators on two relations:

The two relations must be compatible.

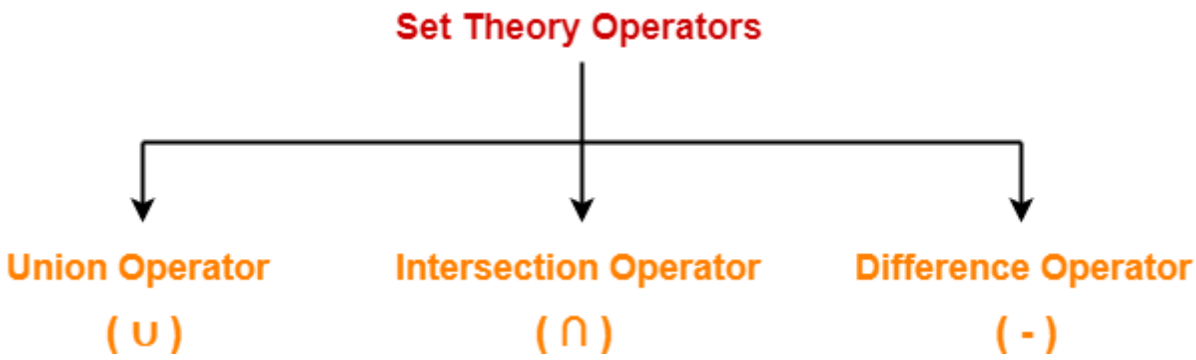
Two relations are compatible if:

a.Both the relation must have same number of attributes(columns).

b.Corresponding attribute (or column) have the same domain (or type).

c.Duplicate tuples are automatically eliminated.

Following operators are called as set theory operators-



1. Union Operator (∪)
2. Intersection Operator (∩)
3. Difference Operator (-)

1.Union Operator(∪):

Suppose R and S are two relations. The Union operation selects all the tuples that are either in relations R or S or in both relations R & S.

It eliminate the duplicate tuples.

Example-

Consider the following two relations R and S-

ID	Name	Subject
100	Ankit	English
200	Pooja	Maths
300	Komal	Science

Relation R

ID	Name	Subject
100	Ankit	English
400	Kajol	French

Relation S

Then, R ∪ S is-

ID	Name	Subject
100	Ankit	English
200	Pooja	Maths
300	Komal	Science
400	Kajol	French

Relation R ∪ S

Ex: project the name from both the R and S table.

Query: $\Pi_{name}(R) \cup \Pi_{name}(S)$

Result:

Name
Ankit
Pooja
Komal
Kajol

Ex: Find the names of authors who have either written a book or an article or both.

$$\Pi_{author}(Book) \cup \Pi_{author}(article)$$

2. Intersection Operator (∩)-

Let R and S be two relations.

Then-

- R ∩ S is the set of all tuples belonging to both R and S.

- In R ∩ S, duplicates are automatically removed.
- Intersection operation is both commutative and associative.

Example-

Consider the following two relations R and S-

ID	Name	Subject
100	Ankit	English
200	Pooja	Maths
300	Komal	Science

Relation R

ID	Name	Subject
100	Ankit	English
400	Kajol	French

Relation S

Then, R ∩ S is-

ID	Name	Subject
100	Ankit	English

Relation R ∩ S

Ex: find the name of students who has subject English in both R and S relation.

Query: $\prod_{name}(\sigma_{subject=English}(R)) \cap \prod_{name}(\sigma_{subject=English}(S))$

Result:

Name
Ankit

Ex: Find the names of the authors who have written a book and an article both.

Query:

$\prod_{author}(Books) \cap \prod_{author}(Articles)$

3. Difference Operator (-):

Let R and S be two relations.

Then-

- R – S is the set of all tuples belonging to R and not to S.
- In R – S, duplicates are automatically removed.
- Difference operation is associative but not commutative.

Example-

Consider the following two relations R and S-

ID	Name	Subject
100	Ankit	English
200	Pooja	Maths
300	Komal	Science

Relation R

ID	Name	Subject
100	Ankit	English
400	Kajol	French

Relation S

Then, R – S is:

ID	Name	Subject
200	Pooja	Maths
300	Komal	Science

Relation R – S

Ex: Find the names of those students who are in R table but not in S table.

Query: $\prod_{name}(R) - \prod_{name}(S)$

Name
Pooja
Komal

Example: Find the names of the authors who have written books but not articles.

Author	bookName
Rahul	java
Nitin	DBMS
Sunil	OS

Book

Auther	Article Name
Rahul	about CO
Sushil	Internet
Amit	WWW

Article

Query: $\Pi_{author}(Books) - - - - \Pi_{author}(Articles)$

Result:

Author
Nitin
Sunil

Question: Consider the following relation:

Branch(branch_name , branch_city, assets)

Account(account_no, branch_name, balance)

Depositor(customer_name, account_no)

Customer(cust_name, cust_street, cust_city)

Loan(loan_no, branch_name, amount)

Borrower(customer_name, loan_no)

Use Relational algebra to answer the following:

(i).Find those account number where balance is less than<1500.
(ii).Find those loan number which are from branch no=101 with amount>2000.
(iii).Find branch name and branch city with assets more than 300000.
(iv).Find the name of the customer who have loan or an account or both.
(v).Find the name of a branch who have accounts but not loan.
(vi).Find the name of a customer who neither have a loan or an account.

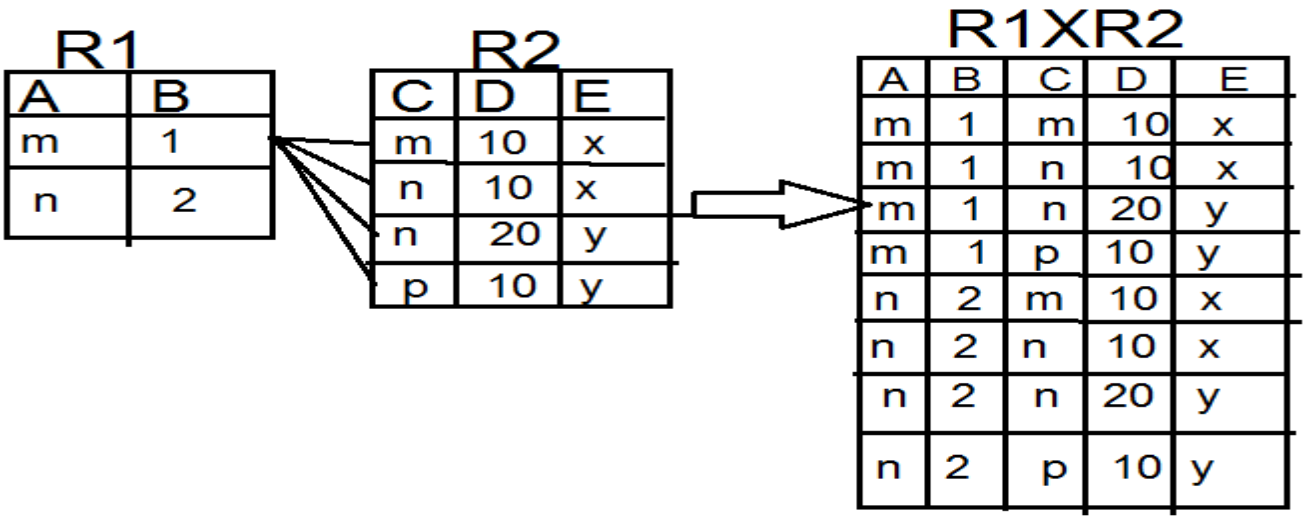
Solution:

1. $\Pi_{account\_no}\ \sigma_{balance<1500}(Account)$
2. $\Pi_{loan\_no}\ \sigma_{branch\_no=101\ \wedge\ amount>2000}(Loan)$
3. $\Pi_{branch\_name,branch\_city}\ \sigma_{assets>300000}(Branch)$
4. $\Pi_{customer\_name}(Depositor)\ \cup\ \Pi_{customer\_name}(Borrower)$
5. $\Pi_{branch\_name}(Account)\ -\ \Pi_{branch\_name}(Loan)$
6. $\Pi_{customer\_name}(Customer)\ -\Pi_{customer\_name}(Depositor)\ \cup\ \Pi_{customer\_name}(Borrower)$

Cartesian Product or Cross Product or Cross Join:

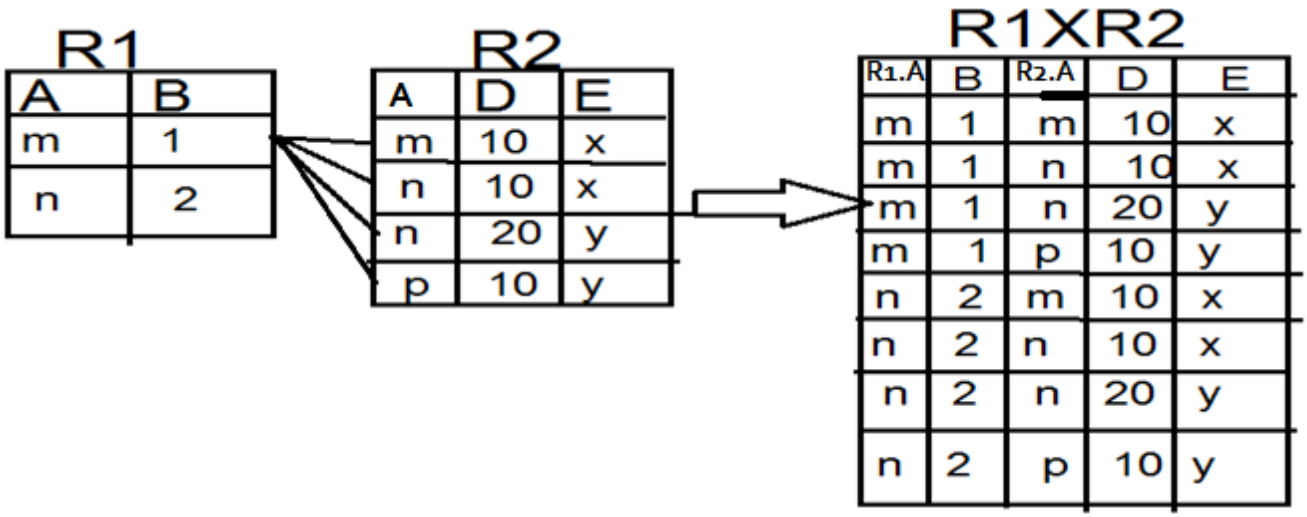
=>Cartesian Product combines information of two different relations into one.
=>It is basic and binary operator.
=>It is also called Cross Product.
=>Generally, a Cartesian Product is never a meaningful operation when it is performed alone.
However, it becomes meaningful when it is followed by other operations.
=>Generally, it is followed by select operations.
Symbol: X
Notation: R1 X R2

Example: Let R1 and R2 are two relation then Cartesian Product of two will be



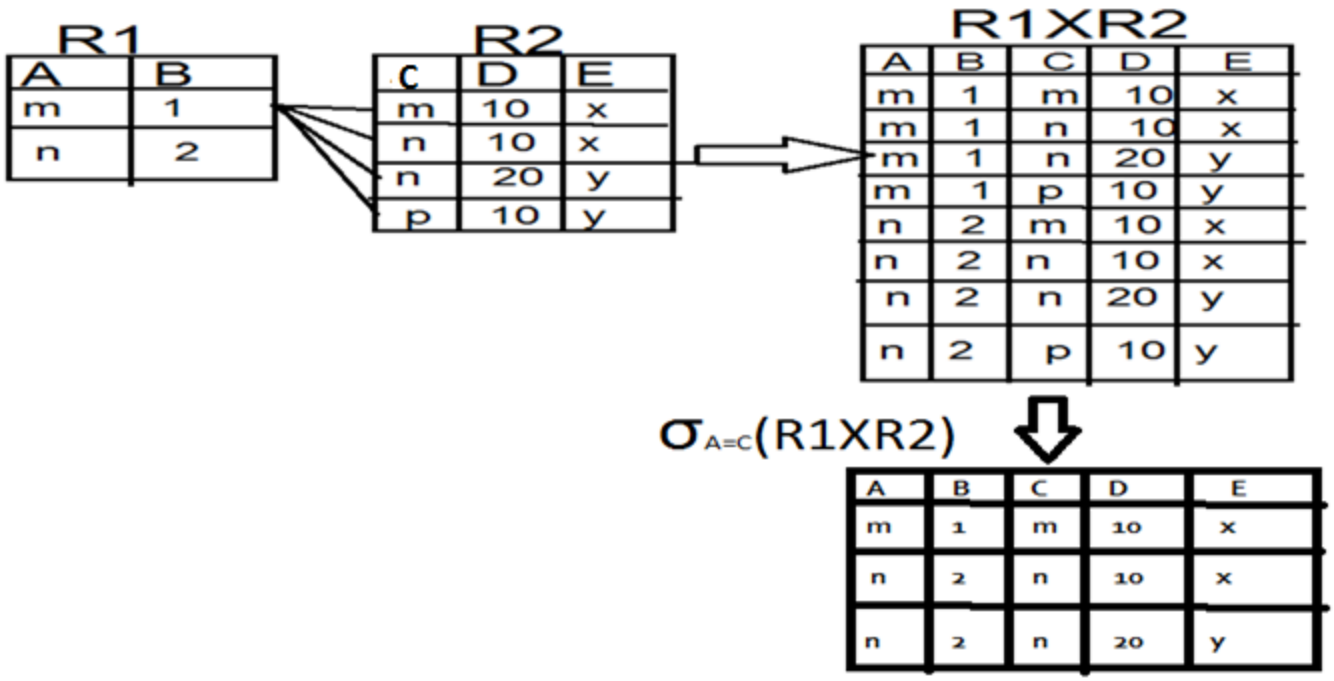
Characteristics of Cartesian Product:

1. If Relation **R1** and **R2** have **A1** and **A2** attributes respectively, then resulting relation will have **A1+A2** attributes from both the input relation.
   Ex:In the above example table R1 has two attribute(or column) A and B and table R2 has three attribute C,D,E
   So, Resulting relation R1XR2 have total 2+3=5 attributes(A,B,C,D,E)
2. If Relation **R1** and **R2** have **T1** and **T2** tuples respectively, then resulting relation will have **T1xT2** tuples.
   Ex: In the above example relation R1 has two tuples(or row) and relation R2 has four tuples , so resulting relation has Total 2x4=8 tuples.
3. If both input relation have some attribute having same name, change the name of the attribute with the name of the relation **"Relation_name.attribute_name"**
   Example:
   If both relation have some attribute having same name, change the name of the attribute(or column) with the
   Name of relation(or table) as table_name.column_name as shown in below

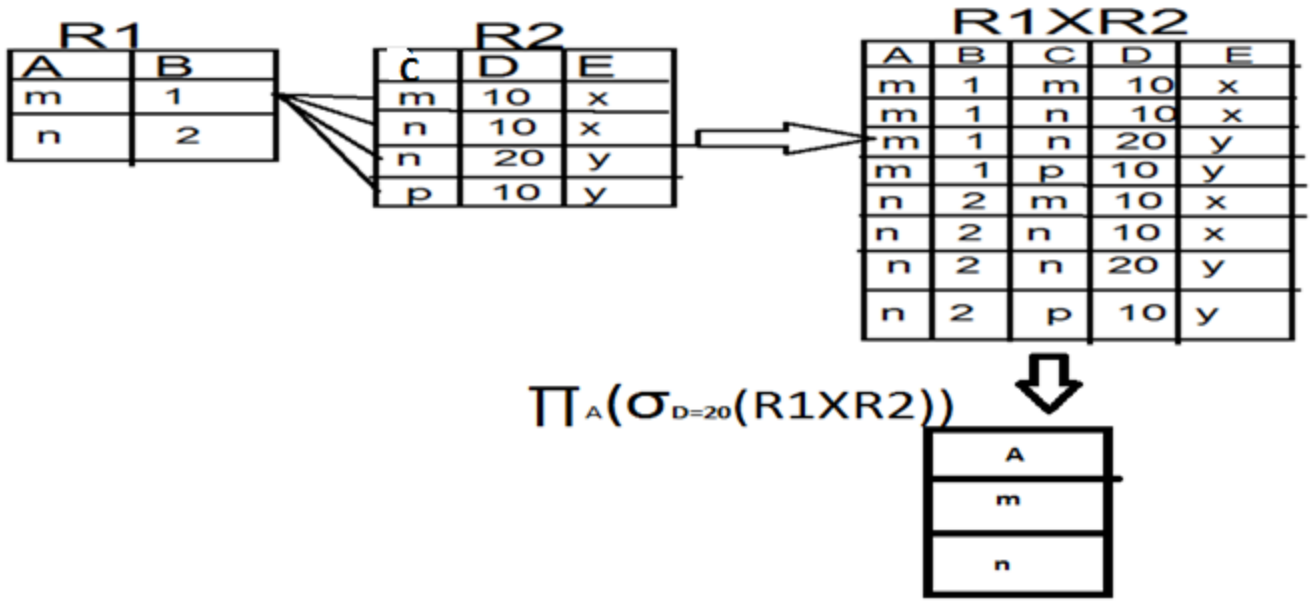


Composition of operation:

Example: $\sigma_{A=C}(R1XR2)$ in the following relation.



Example: $\prod_A(\sigma_{D=20}(R1XR2))$ in the following relation.



RENAME (ρ) Operation :

The RENAME operation is used to rename the output of a relation. Since the results of relation algebra is also relations but without any name.

Sometimes it is simple and suitable to break a complicated sequence of operations and rename it as a relation with different names. Reasons to rename a relation can be many, like –

- We may want to save the result of a relational algebra expression as a relation so that we can use it later.
- We may want to join a relation with itself, in that case, it becomes too confusing to specify which one of the tables we are talking about, in that case, we rename one of the tables and perform join operations on them.

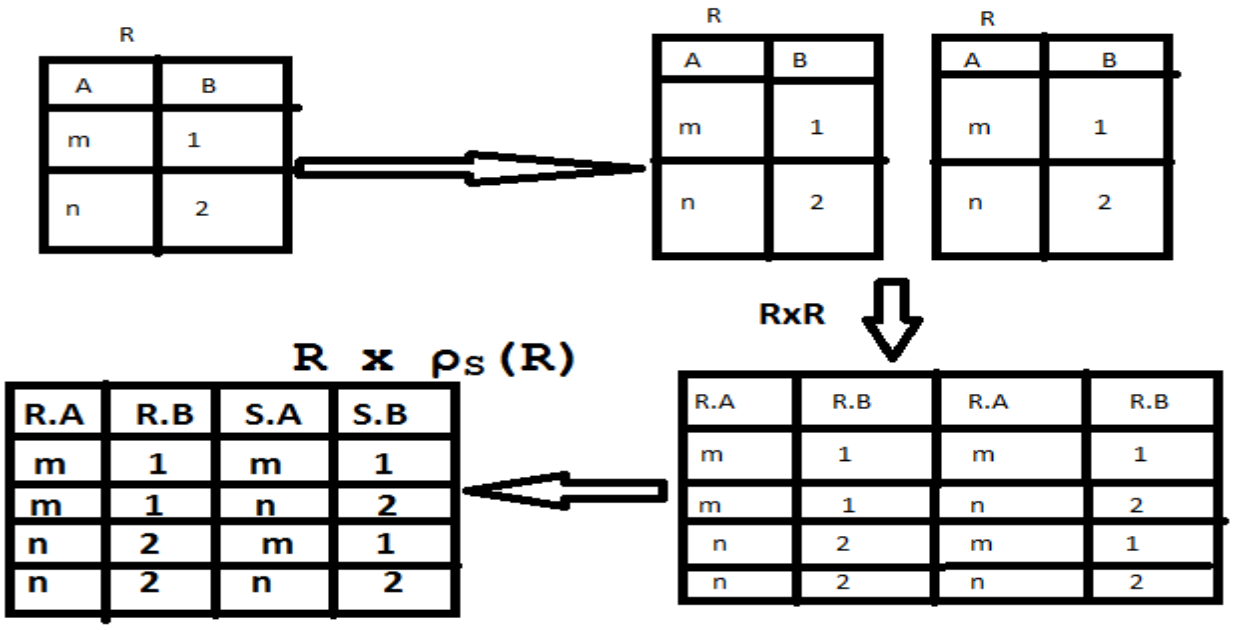
Notation:

ρ x (R)

where the symbol **rho 'ρ'** is used to denote the RENAME operator and **R** is the result of the sequence of operation or expression which is saved with the name **X**.

Example: Suppose we want to do Cartesian product between same table then one of the table should be renamed with another name.

RxR (ambiguity will be there)



Notation1: R x $\rho_S(R)$(Rename R to S)

Notation2: $\rho_{x(A1,A2,A3...An)}(E)$

It returns the result of expression E under the name X, and with attributes renamed to A1,A2,A3…An.

Notation3: $\rho_{(A1,A2,A3...An)}(E)$

It return the result of expression E with the attributes renamed to A1,A2,A3…An.

Example: find the female students from student relation and rename the relation Student as FemaleStudent

And the attribute of Student-RollNo, SName as Sno, Name.

Studnet

RollNo	SName	Gender
1	Rohit	M
2	Nisha	F
3	Arun	M
4	Tanisha	F
5	Jyoti	F

FemaleStudent

SNo	Name
2	Nisha
4	Tanisha
5	Jyoti

$$\rho_{FemaleStudent(SNo,Name)}(\prod_{RollNo,\ SName}(\sigma_{Gender='F'}(Student)))$$

- **Example-2:** Query to rename the attributes Name, Age of table Department to A,B.

  $\rho_{(A,B)}$ (Department)

- **Example-3:** Query to rename the table name Project to Pro and its attributes to P, Q, R.

  $\rho_{Pro(P,Q,R)}$ (Project)

- **Example-4:** Query to rename the first attribute of the table Student with attributes A, B, C to P.

  $\rho_{(P,B,C)}$ (Student)