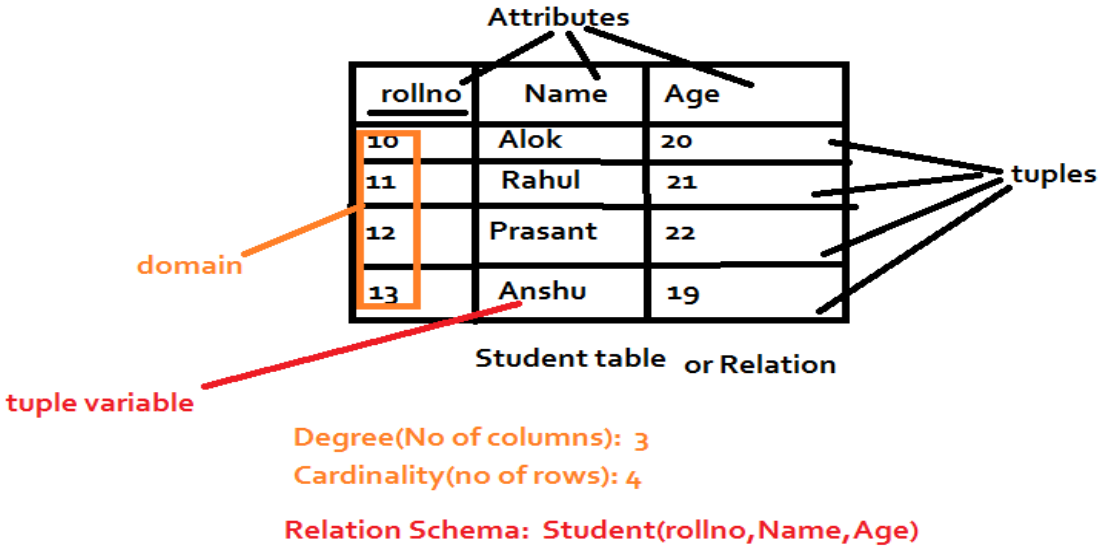
Unit-2(Relational Model)

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-

This relation does not satisfy the entity integrity constraint as here the primary key contains a NULL value.

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

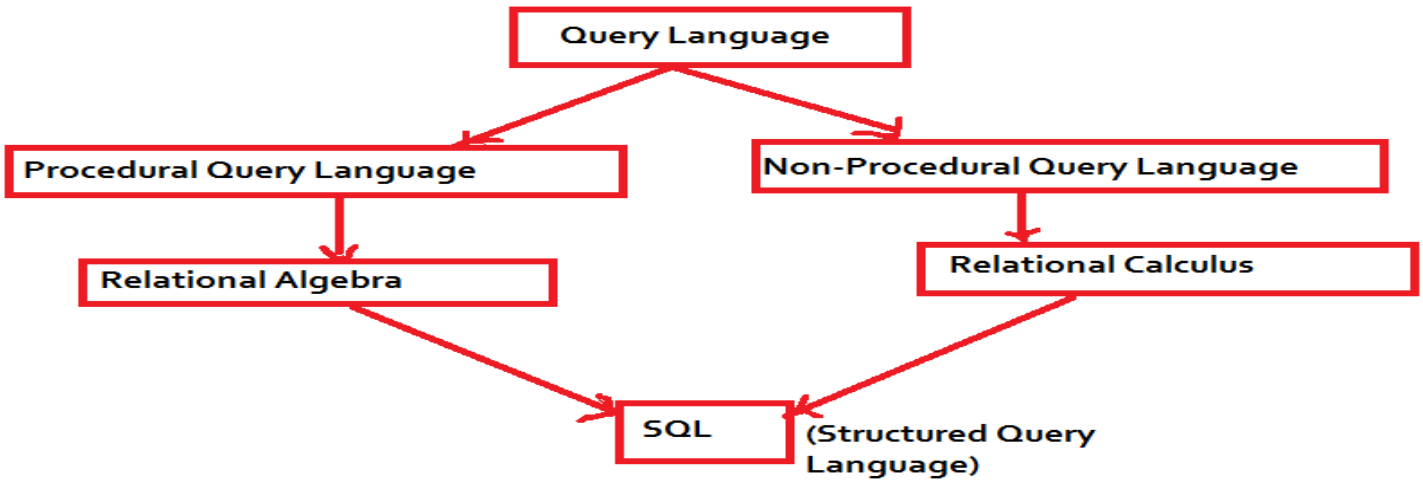
1.Relational Algebra
2.Relational Calculus: Relational tuple are also of two types
a. Tuple Relational Calculus
b.Domain Relational Calculus

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

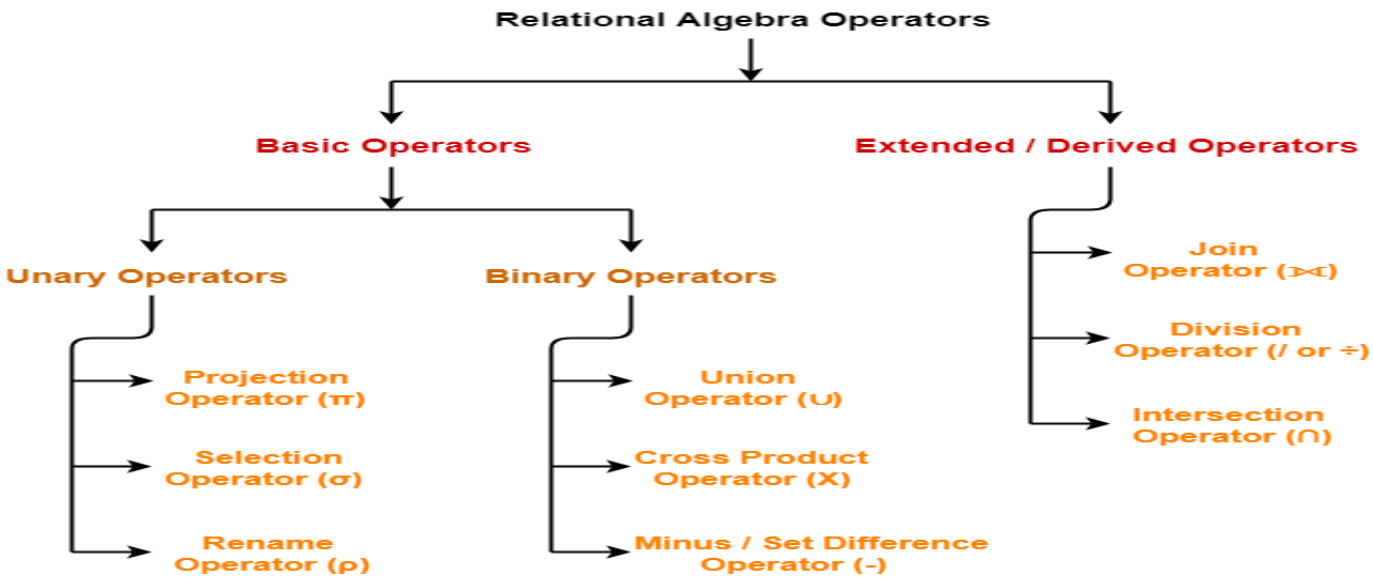
a. Tuple Relational Calculus
b.Domain Relational Calculus
=>Relational Calculus is a Non-Procedural language.
=>It is non-operational or Declarative
=> **Non-Procedural** :what data to be retrived from database but does not tell how to retrieve it.



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 – – =, ≠, ≥, < , >, ≤. with us of connectors like OR, AND, or NOT .

or

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

Important Points-

Point-01:

We may use logical operators like ∧ , ∨ , ! 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

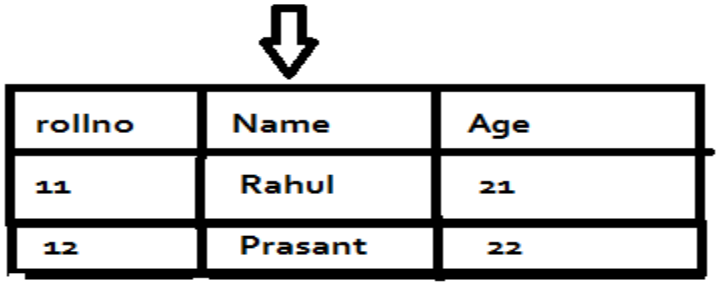
Query: Select Student whose name is Rahul.

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

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

Example:

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

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

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

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

3.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)

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.

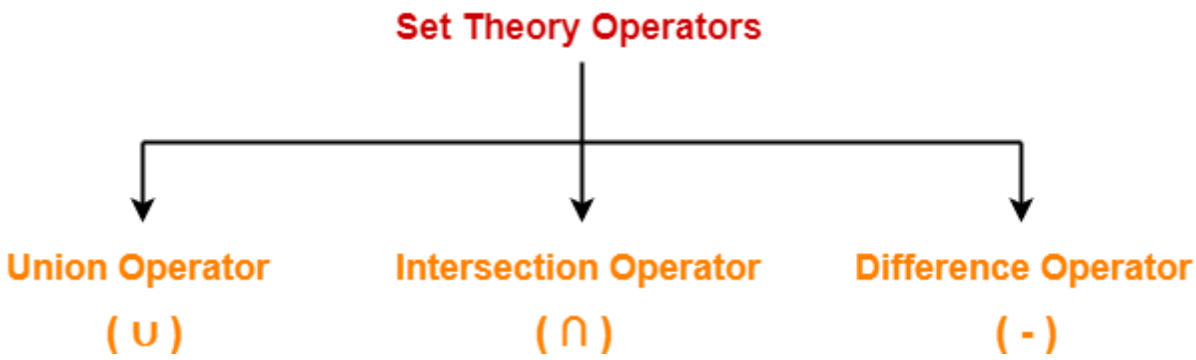
Two relations are compatible if:

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
400	Kajol	French

Relation S

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

Relation R

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: $\prod_{author}(Books) - - - - \prod_{author}(Articles)$

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 name='noida' 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. ∏ account_no (σ balance<1500 (Account))
2. ∏ loan_no (σ branch_name='noida' ^ amount>2000 (Loan))
3. ∏ branch_name,branch_city (σ assets>300000(Branch))

4.∏ customer_name (Depositor) U ∏ customer_name (Borrower)

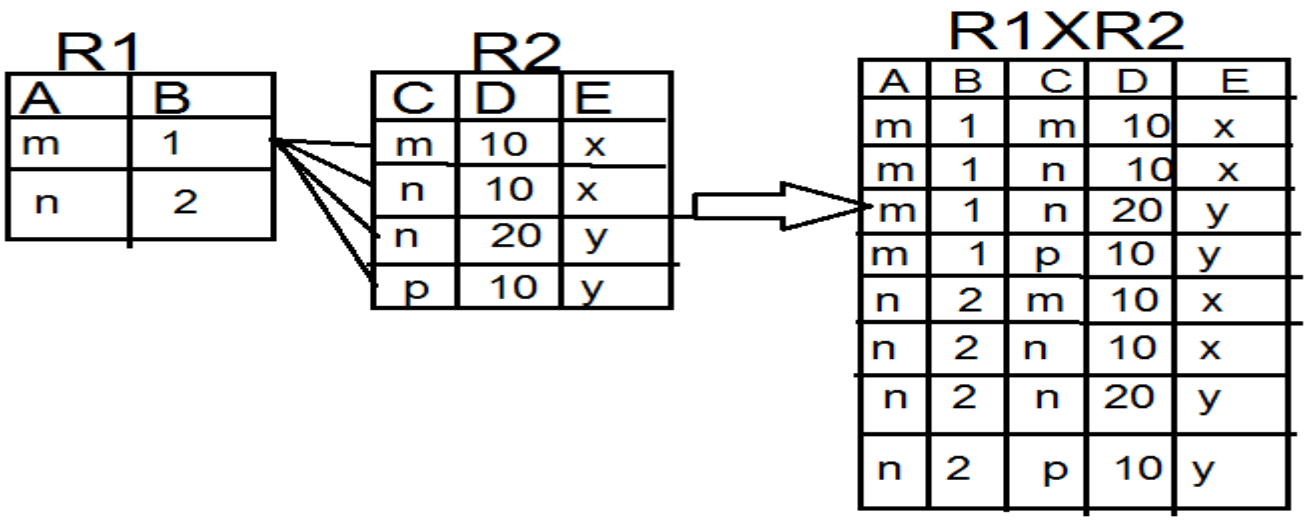
5. ∏ branch_name (Account) - ∏branch_name (Loan)

6. ∏customer_name (Customer) -(∏ customer_name (Depositor) U ∏ 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



R1

A	B
m	1
n	2

R2

C	D	E
m	10	x
n	10	x
n	20	y
p	10	y

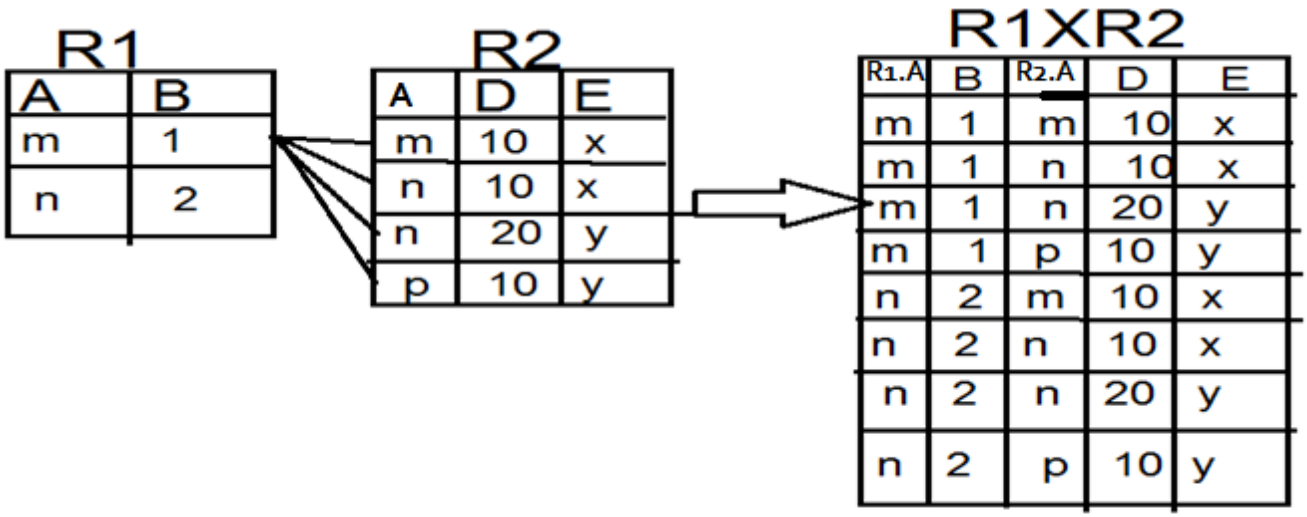
R1XR2

A	B	C	D	E
m	1	m	10	x
m	1	n	10	x
m	1	n	20	y
m	1	p	10	y
n	2	m	10	x
n	2	n	10	x
n	2	n	20	y
n	2	p	10	y

Characteristics of Cartesian Product:

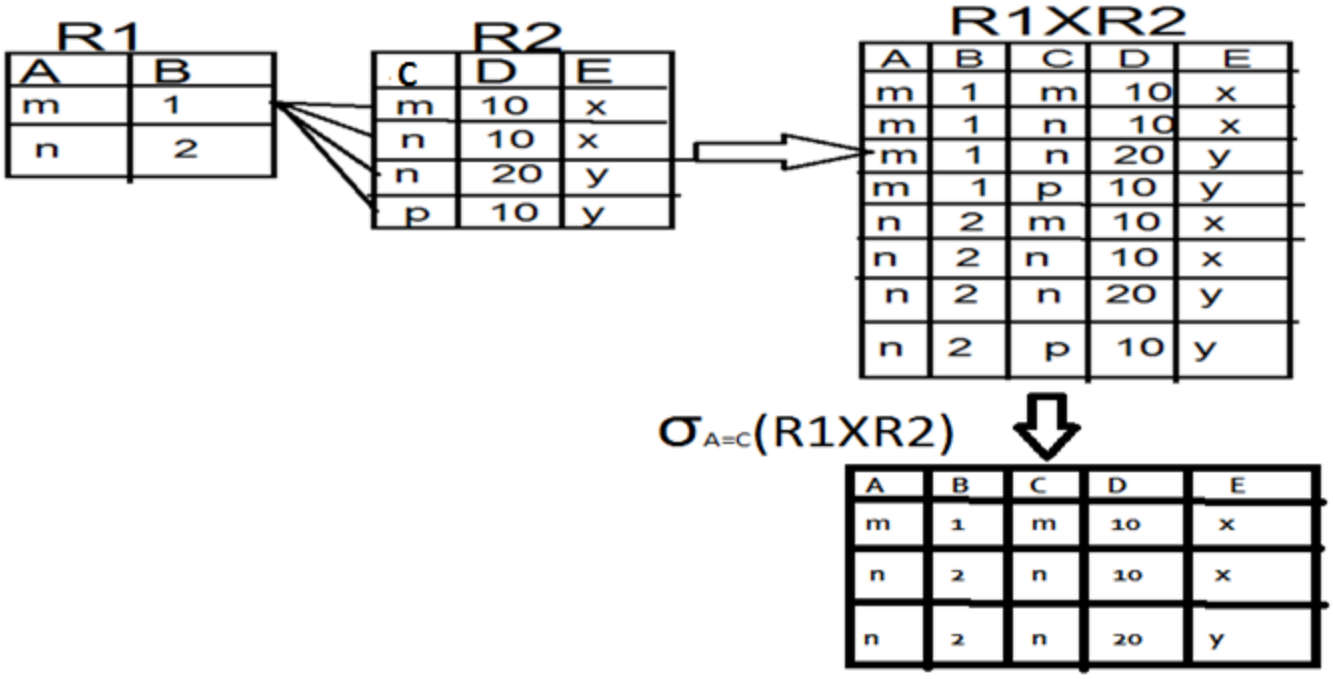
1. If Relation **R1** and **R2** have **A1** and **A2** attributes respectively, then resulting relation will have **A1+A2** attributes from both the input relation.
   Ex:In the above example table R1 has two attribute(or column) A and B and table R2 has three attribute C,D,E
   So, Resulting relation 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

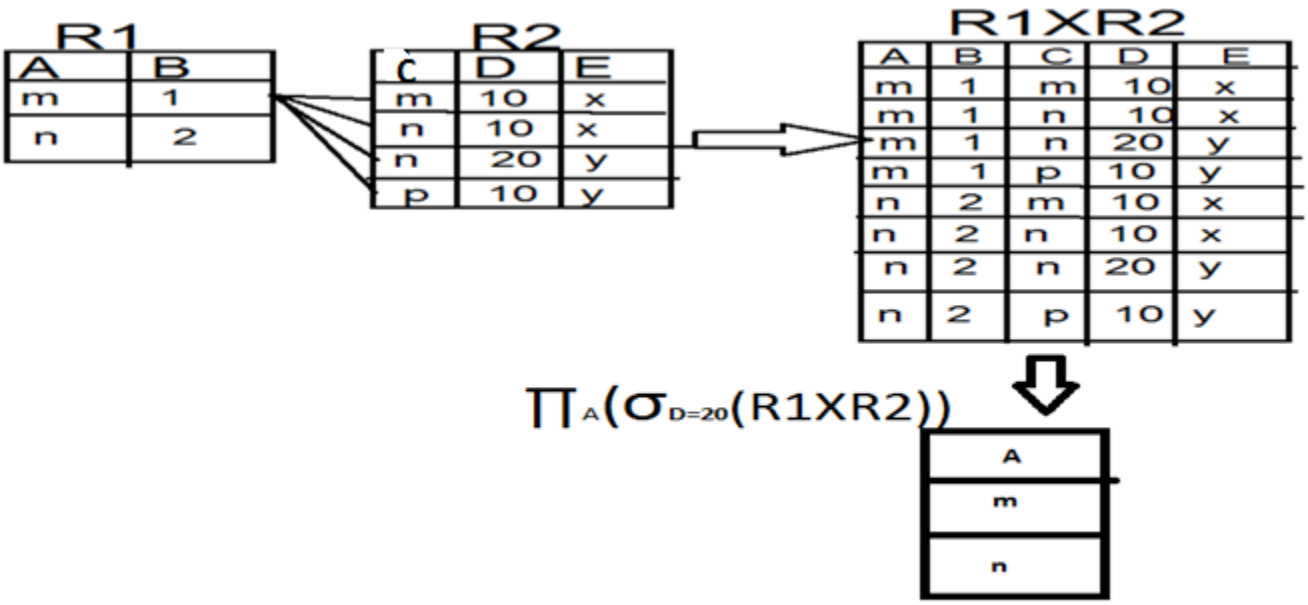


Composition of operation:

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.

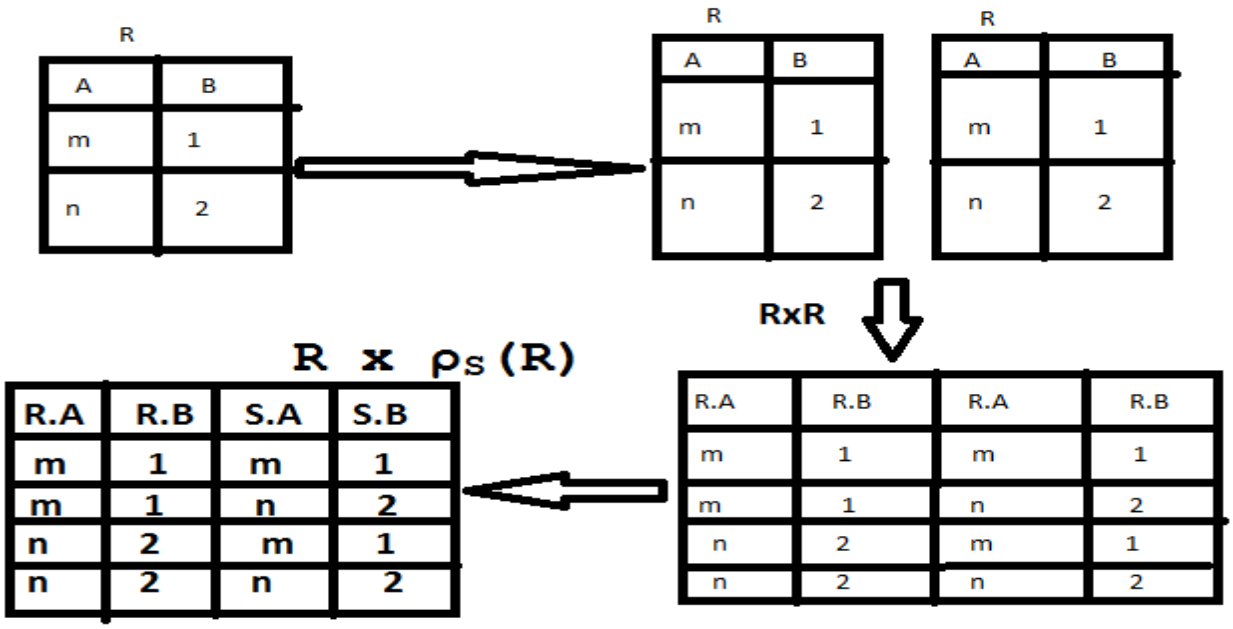
Notation:

$$\rho_x(R)$$

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

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

- **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)

Join Operator:

=>Cartesian product of two relation(AXB), gives us all the possible tuples that are paired together.
=>But it might not be feasible in certain cases to take a Cartesian product where we encounter huge relations with thousands of tuples having a considerable large number of attribute.

Join Operation (⋈):

=>Join is a combination of a Cartesian product followed by a selection process.

Join=Cartesian Product+Selection

=>Join is an Derived operator which simplify the queries.
=>A join operation pairs two tuples from different relation,if and only If a given join condition is satisfied.

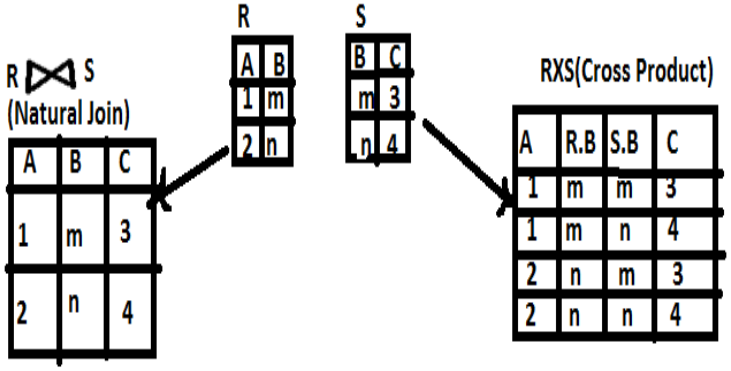
$$A \bowtie_c B = \sigma_c(A \times B)$$

Join (⋈)

1. Combination of tuples that satisfy the matching conditions
2. It has Fewer tuples then cross product and can be compute efficiently.

Cartesian Product/Cross Product/Cross Join(⋈)

1. It has All possible combination of tuples from the relations.
2. Huge number of tuples and costly to manage.



Types of Join:

1. Inner Join

2. Outer Join

1. Inner Join:

=>An inner join Contains only those tuples that satisfy the matching condition. While rest of the tuple are excluded.

=>Inner join is of three types

a. Theta (θ)Join
b. Equi Join
c. Natural Join

a. Theta (θ)Join/Conditional Join:

=>If we want to join two or more relation based on some conditions then it is known as Theta (θ)Join.

=>The join condition is denoted by the Theta (θ) symbol.

=>It uses all kind of relational operator like <,>,<=,>=,=

Notation: $R \bowtie_{\theta} S$

Where θ is a predicate/condition. It can use any comparison operator <,>,<=,>=,=

It is also equivalent to

$$R \bowtie_{\theta} S = \sigma_{\theta}(RXS)$$

Example:

R

rollno	name	age	marks
10	Mohit	19	67
11	Subodh	21	70
12	Rahul	20	72

S

rollno	city	mobile
10	Noida	9838393
12	Delhi	99393933

$R \bowtie_{R.rollno<S.rollno} S$

R.rollno	name	age	marks	S.rollno	city	mobile
10	Mohit	19	67	12	Delhi	99393933
11	Subodh	21	70	12	Delhi	99393933

Equivalent to: $\sigma_{R.rollno<S.rollno}(RXS)$

b. Equi Join:

Equi join is special case of Theta join where condition contains equalities =

Notaion: $R \bowtie R.a1=S.a1 \wedge R.b1=S.b1 \wedge \ldots\ldots R.an=S.bn S$

R

rollno	name	age	marks
10	Mohit	19	67
11	Subodh	21	70
12	Rahul	20	72

S

rollno	city	mobile
10	Noida	9838393
12	Delhi	99393933

$R \bowtie_{R.rollno=S.rollno} S$

R.rollno	name	age	marks	S.rollno	city	mobile
10	Mohit	19	67	10	Noida	9838393
12	Rahul	20	72	12	Delhi	99393933

Equivalent to: $\sigma_{R.rollno=S.rollno}(RXS)$

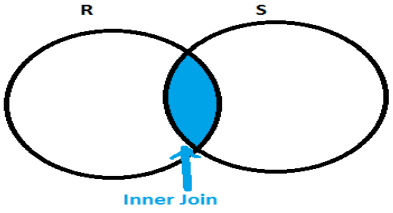
c. Natural join:

=>Natural join can only perform if there is at least one common attribute(column)that exist between two relations i.e the attributes must have same name and domain.
=>Natural join does not use any Comparison operator.
=>It is same as Equi join which occurs implicitly by comparing all the common attributes(columns) in both relation, but difference is that in natural join the Common attributes appears only once. The resulting schema will change.
=>The Result of Natural join is the set of all combinations of tuples in two relations R and S that are equal on their common attributes names.
=>The Natural join of two relation can be obtained by applying a Projection operation to Equi join of two
Relations. In terms of basic operator:

Natural join=Cartesian product + Selection + Projection

=>Natural join is by default inner join because the input tuples which does not satisfy the conditions of join does not appear in the result.

Notaion: $R \bowtie S$



Example:

R

rollno	name	age	marks
10	Mohit	19	67
11	Subodh	21	70
12	Rahul	20	72

S

rollno	city	mobile
10	Noida	9838393
12	Delhi	99393933

$R \bowtie S$

R.rollno	name	age	marks	City	mobile
10	Mohit	19	67	Noida	9838393
12	Rahul	20	72	Delhi	99393933

Equivalent to: $\prod_{R.rollno, name, age, marks, city,} (\sigma_{R.rollno=S.rollno}(RXS))$

Example:

EMP_CODE	EMP_NAME
101	Stephan
102	Jack
103	Harry

Employee

EMP_CODE	SALARY
101	50000
102	30000
103	25000

Salary

$\prod_{EMP\_NAME, SALARY} (EMPLOYEE \bowtie SALARY)$

EMP_NAME	SALARY
Stephan	50000
Jack	30000
Harry	25000

2. Outer Join:

=>Outer Join Contains matching tuples that satisfy the matching condition, along with some or all tuples that do not Satisfy the matching condition.
=>It is based on both matched and unmatched tuple.
=>It Contains all rows from either one or both relation.

=> It uses NULL values i.e the value in unknown or does not exist.

Outer Join=Natural join + Extra Information(from left table, right table or both table)

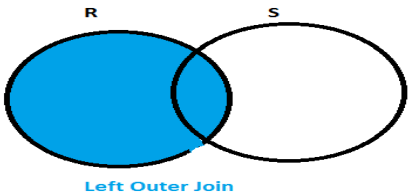
Outer Join is of three types:

i. Left Outer Join
ii. Right Outer Join
iii. Full Outer Join

i.Left Outer Join:

=>In Left outer join, all the tuples from the left relation R1 are included in the resulting relation. The tuples of R1 which do not satisfy join condition will have values as NULL for attributes of R2.
=>it contains All records from left table.
=>Only matching records from right table and rest record will be NULL.
Symbol:

Notation: R  S



Example: Employee

EMP_NAME	STREET	CITY
Ram	Civil line	Mumbai
Shyam	Park street	Kolkata
Ravi	M.G. Street	Delhi
Hari	Nehru nagar	Hyderabad

Fact_workers

EMP_NAME	BRANCH	SALARY
Ram	Infosys	10000
Shyam	Wipro	20000
Kuber	HCL	30000
Hari	TCS	50000

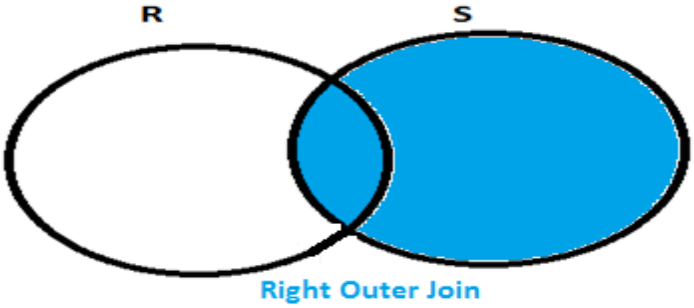
EMPLOYEE ⋈ FACT_WORKERS

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehru street	Hyderabad	TCS	50000
Ravi	M.G. Street	Delhi	NULL	NULL

ii. Right Outer Join:

=>In Right Outer join, all the tuples from the right relation S are included in the resulting relation. The tuples of S which do not satisfy join condition will have values as NULL for attributes of R.
=>It has all record from right table.
=>Only matching records from left table and NULL values for rest records.
Symbol: ⟖

Notation: R ⟖ S



Right Outer Join

EMPLOYEE

EMP_NAME	STREET	CITY
Ram	Civil line	Mumbai
Shyam	Park street	Kolkata
Ravi	M.G. Street	Delhi
Hari	Nehru nagar	Hyderabad

EMP_NAME	BRANCH	SALARY
Ram	Infosys	10000
Shyam	Wipro	20000
Kuber	HCL	30000
Hari	TCS	50000

FACT_WORKERS

⇨

EMPLOYEE ⟖ FACT_WORKERS

EMP_NAME	BRANCH	SALARY	STREET	CITY
Ram	Infosys	10000	Civil line	Mumbai
Shyam	Wipro	20000	Park street	Kolkata
Hari	TCS	50000	Nehru street	Hyderabad
Kuber	HCL	30000	NULL	NULL

iii.Full outer join:

- Full outer join is like a left or right join except that it contains all rows from both tables.
- In full outer join, tuples in R that have no matching tuples in S and tuples in S that have no matching tuples in R in their common attribute name.
- It is denoted by ⟗.
- **Example:** Using the above EMPLOYEE table and FACT_WORKERS table

  EMPLOYEE ⟗ FACT_WORKERS

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehru street	Hyderabad	TCS	50000
Ravi	M.G. Street	Delhi	NULL	NULL
Kuber	NULL	NULL	HCL	30000

Division Operator:

=>Division operator is a Derived operator. It is suited to queries that include the keyword **all** or **every** like **" at all", "for all "or " in all", "at every", "for every" or " in every".**

Example:
a. Find the person that has account in **all** the banks of a particular city.
b. Find Employees who works on **all** projects of company.
c. Find students who have registered for **every** course.
d. Find sailors who have reserved **all** boats.
=>In all these queries, the description after the keyword **"all"** or **"every"** defines a set which contains some elements and the **final result** contains those records who satisfy these requirements.

Notation: A/B or A÷B

Division operator A÷B can be applied if and only if:

- Attributes of B is proper subset of Attributes of A.
- The relation returned by division operator will have attributes = (All attributes of A – All Attributes of B)
- The relation returned by division operator will return those tuples from relation A which are associated to every B's tuple.

  Example: Let there are two table Enroll and Course

Sid	Cid
S1	C1
S2	C1
S1	C2
S3	C2

Enroll

Cid
C1
C2

Course

Query: Retrieve Sid of students who enrolled in every course.

Sol: ∏Sid(Enroll) -(∏ Sid (∏Sid(Enroll) X ∏Cid(Course)-Enroll))
Explanation:
Enroll(Sid/Cid)/Course(Cid)=S1(result)which is enrolled in both courses c1 and c2.
Now, in above query first we did the projection of Sid from Enroll table and Cid from Course table and we get.

Sid
S1
S2
S1
S3

Cid
C1
C2

Course

Then we perform cross product between Enroll table and course table and Set difference with Enrolled table so we get non enrolled student by doing the projection of sid of result

Sid	Cid
S1	C1
S1	C2
S2	C1
S2	C2
S3	C1
S3	C2

Sid	Cid
S1	C1
S1	C2
S2	C1
S2	C2
S3	C1
S3	C2

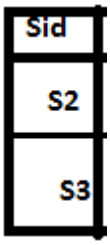
−

Sid	Cid
S1	C1
S2	C1
S1	C1
S3	C2

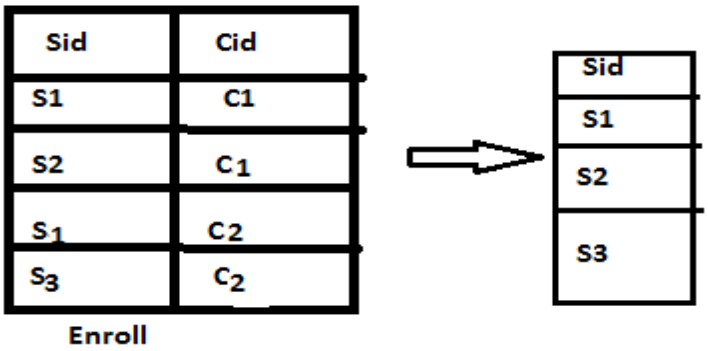
=

Sid	Cid
S2	C2
S3	C1

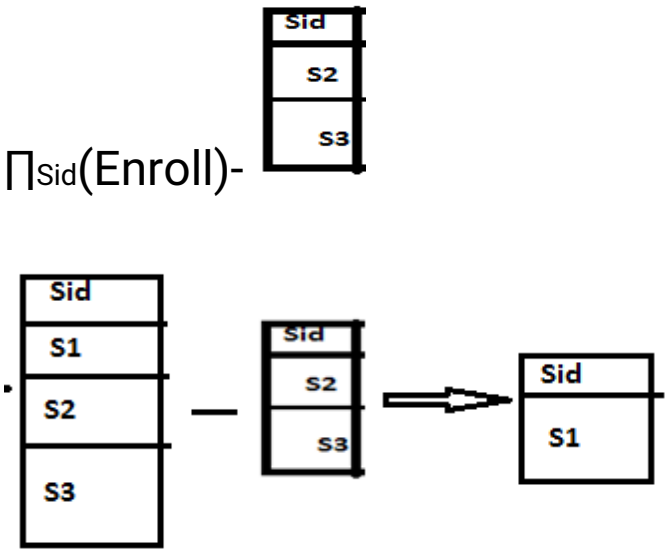
After projection we get Sid of those students who are not enrolled in both courses.



Π_{Sid}(Enroll):



Π_{Sid}(Enroll)-



Question: Consider the following relation:

Branch (branch_name , branch_city, assets)

Account (account_no, branch_name, balance)

Depositor (cust_name, account_no)

Customer (cust_name, cust_street, cust_city)

Loan (loan_no, branch_name, amount)

Borrower (cust_name, loan_no)

Use Relational algebra to answer the following:

Q1. Find the names of all customers who have a loan at the bank, along with the loan_number and the loan amount.

Q2. Find the names of all branches with customers who have an account in the bank and who live in Greater

noida.

Q3. Find all customers who have both a loan and an account at the bank.

Q4. Find the names of all customers who have a loan at the Greater noida branch.

Q5.Find the names of all customers who have a loan at Greater noida branch, but does not have an account at any branch
of bank.

Sol(1). Find the names of all customers who have a loan at the bank, along with the loan_number and the loan amount.

Using Cartesian product:

$$\prod_{\text{customer\_name,loan.loan\_number,amount}} \sigma_{\text{loan.loan\_no=borrower.loan\_no}}(\text{BorrowerXLoan})$$

Using Natural join:

$$\prod_{\text{customer\_name,loan.loan\_number,amount}} (\text{Borrower} \bowtie \text{Loan})$$

Sol2. Find the names of all branches with customers who have an account in the bank and who live in Greater noida.

$$\prod_{\text{branch\_name}} (\text{Customer} \bowtie \text{Depositor} \bowtie \text{Account})$$

Sol3. Find all customers who have both a loan and an account at the bank.

$$\prod_{\text{customer\_name}}(\text{Borrower}) \cap \prod_{\text{customer\_name}}(\text{Depositor})$$

Or

Using Natural join:

$$\prod_{\text{customer\_name}} (\text{Borrower} \bowtie \text{Depositor})$$

Sol(4). $\prod_{\text{customer\_name}}(\sigma_{\text{branch\_name='Greater Noida'}} (\text{Borrower} \bowtie \text{Loan}))$

Sol(5). $\prod_{\text{customer\_name}}(\sigma_{\text{branch\_name='Greater Noida'}} (\text{Borrower} \bowtie \text{Loan}) - \prod_{\text{customer\_name}}(\text{Depositor})$

Question:Consider the following relations:

Student(ssn, name, address, major)
Course(code,title)
Registered(ssn,code)

Use relational algebra to answer the following:

i.List the codes of course in which at least one student is registered(registered courses)

ii. List the title of registered courses.

(ii).List the codes of courses for which no student is registered.
(iv). The titles of courses for which no student is registered.
(v).Names of Students and titles of courses they registered to.
(vi).SSNs of Students who are registered for both 'Database System' and 'Analysis of Algorithm'.
(vii).The name of Students who are registered for both 'Database System' and 'Analysis of Algorithm'.
(viii).List of courses in which all students are registered.
(ix).List of courses in which all 'ECMP' major students are registered.
Sol:

(i). ∏ codes(Course ⋈ Registered)

Or

∏ codes(Registered)

(ii). ∏ title(Course ⋈ Registered)

(iii). ∏ codes(Course)- ∏ codes(Registered)

(iv). ∏title (∏ codes(Course)- ∏ codes(Registered)) ⋈ course)

(v). ∏name,title (Student ⋈ Registered ⋈ course)

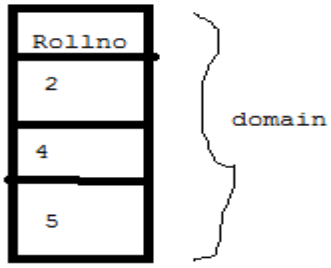
(vi).∏ssn (Student ⋈ Registered ⋈ (σ title='Database Systems' (Course)) ∪ ∏ssn (Student ⋈ Registered ⋈ (σ title='Analysis of Algorithms' (Course))

Relational Calculus

=>It is a Non-procedural query language(declarative language).
=>It uses mathematical predicate calculus(or first order logic) instead of algebra.
=>Relational Calculus tells what to do but never explain how to do.
=>Relational calculus provides description about the query to get the result where as Relational algebra gives the method to get the result.
There ate two type of Relational Calculus:

1. Tuple Relational Calculus: it works on tuples(rows)

2. Domain Relational Calculus: It works on domain of attributes(or columns)



1.Tuple Relational Calculus:

=>It is a Non-procedural query language(declarative language).
=>Tuple Relational calculus is used for selecting the tuples(rows) in a relation(table) that satisfy the given condition(or predicate). The result of the relation can have one or more tuples.
=> A Query in Tuple Relational Calculus is expressed as:

{t|P(t)}

Where t denotes resulting tuple and P(t) denote predicate(condition) used to fetch the tuple t.
Result of Query: It is the set of all tuples t such that predicate P is true for t.

Notation used:

t is tuple variable
t[A] denotes the value of tuple t on attribute A.
t Ɛ r denotes that tuple t is in relation r.

P is a formula similar to that of the predicate calculus.

Predicate Calculus Formula:

1. Set of attributes and constants.
2. Set of Comparision operators:<,>,<=,>=,=,‡
3. Set of Connectives: AND(^),OR(V),NOT(¬)
4. Implication(=>): x=>y, if x is true ,then y is true
5. Quantifiers: there Exists(∃) and for all(∀)

ex: ∃ t Ɛ r(Q(t))=”there exists” a tuple t in relation r such that predicate Q(t) is true.

ex: ∀ t Ɛ r(Q(t))=”For all” tuples t in r relation Predicate Q(t) is true.

Free and Bound variables:

=>The use of quantifiers ∃x and ∀x in a formula is said to bound x in the formula.

=> A variable that is not bound is free variable.

=>There is an important restriction:

The variable t that appears to the left | must be the only free variable in the formula P(t). in other words all other tuple variables must be bound using a quantifier.

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)

Example: find the loan-number,branch-name,and amount for all loans of over 1200.

$$\{ t \mid \{ t \in loan \wedge t[amount] > 1200 \}$$

Example:Find the loan number for each loan of an amount greater than 1500.

$$\{ t \mid \exists s \in loan\ (t[loan\text{-}number] = s[loan\text{-}number]) \wedge s[amount] > 1500) \}$$

it select the set of tuples t such that there exist a tuple s in relation loan for which the values of t & s for the loan-number attribute are equal and the value of s for the amount attribute is greater than 1500

2. Domain Relational Calculus

=>Domain Relational Calculus is a non-procedural query language.
=>In Domain Relational Calculus the records are filtered on the domains.
=>It uses list of attributes to be selected from relation based on the condition or(predicate)/
=>In Domain Relational Calculus, each query is an expression of the form:
{<a1,a2,……..,an>|P(a1,a2,……,an)}
where, a1,a2,……,an represent domain variables
P represent a predicate similar to that of the predicate calculus.
<a1,a2,……..,an> Ɛ r, where r is relation on n attributes and a1,a2,…,an are domain variables.

Result of query: *It is the set of all tuples a1,a2,….,an such that predicate P is true for* a1,a2,…,an tuples.

Predicate Calculus Formula:

1. Set of attributes and constants.
2. Set of Comparision operators:<,>,<=,>=,=,‡
3. Set of Connectives: AND(^),OR(V),NOT(¬)
4. Implication(=>): x=>y, if x is true ,then y is true

5.Quantifiers: there Exists(∃) **and for all(** ∀)

ex: ∃ x (P(x)) and ∀ x (P(x))

Example:

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)

(1) . Find the loan-number,branch-name, and amount for loan of over 50000.

Sol: **{<l,b,a>|<l,b,a> Ɛ loan^a>50000}**

2.Find the loan-number for each loan of an amount greater than 60000.

Sol: **{ <l> | ∃ b,a(<l,b,a> Ɛ loan^a>60000) }**

SQL(Structured Query Language)

- SQL stands for **Structured Query Language**.
- It is designed for managing data in a relational database management system (RDBMS).
- It is pronounced as S-Q-L or sometime **See-Qwell**.
- SQL is a database language, it is used for database creation, deletion, fetching rows, and

modifying rows, etc.
- SQL is domain-specific language.
- SQL is Declarative language i.e a non-procedural language. i.e SQL allows to declare What we want to do ,but not how to do. In SQL we can specify through queries that what data we want but does not specify how to get those data.
- SQL is based on relational algebra and tuple relational calculus.

All DBMS like MySQL, Oracle, MS Access, Sybase, Informix, PostgreSQL, and SQL Server use SQL as standard database language.

Characteristics of SQL

- SQL is easy to learn.
- SQL is used to access data from relational database management systems.
- SQL can execute queries against the database.
- SQL is used to describe the data.
- SQL is used to define the data in the database and manipulate it when needed.
- SQL is used to create and drop the database and table.
- SQL is used to create a view, stored procedure, function in a database.
- SQL allows users to set permissions on tables, procedures, and views.

Advantages of SQL

There are the following advantages of SQL:

High speed:
Using the SQL queries, the user can quickly and efficiently retrieve a large amount of records from a database.

No coding needed:
In the standard SQL, it is very easy to manage the database system. It doesn't require a substantial amount of code to manage the database system.

Well defined standards:
Long established are used by the SQL databases that are being used by ISO and ANSI.

Portability:
SQL can be used in laptop, PCs, server and even some mobile phones.

Interactive language:
SQL is a domain language used to communicate with the database. It is also used to receive answers to the complex questions in seconds.

Multiple data view:
Using the SQL language, the users can make different views of the database structure.

SQL Datatype:

SQL Datatype is used to define the values that a column can contain.
Every column is required to have a name and data type in the database table.

Data type of SQL:

1. Binary Datatypes

There are Three types of binary Datatypes which are given below:

Data Type	Description
binary	It has a maximum length of 8000 bytes. It contains fixed-length binary data.
varbinary	It has a maximum length of 8000 bytes. It contains variable-length binary data.
image	It has a maximum length of 2,147,483,647 bytes. It contains variable-length binary data.

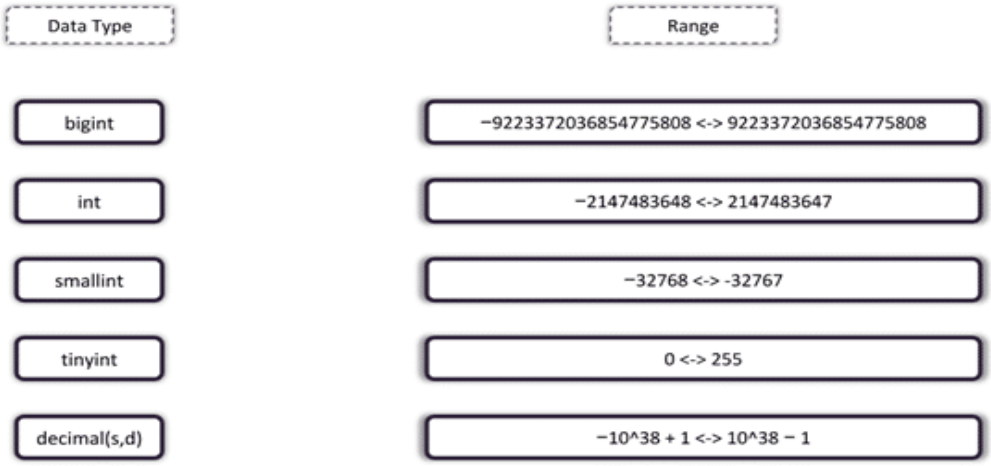
2. Character String Datatype

Data type	Description
char	It has a maximum length of 8000 characters. It contains Fixed-length non-unicode characters.
varchar	It has a maximum length of 8000 characters. It contains variable-length non-unicode characters.
text	It has a maximum length of 2,147,483,647 characters. It contains variable-length non-unicode characters.

Char data type takes in one argument and has fixed length. For example, consider the size of the value to be 20. This would mean that you cannot give any value having more than 20 characters. Keeping in mind the fact that char has fixed length, i.e., if the value size is to be 30 characters, but information assigned to it is of 3 characters, then the memory consumed is of 30 characters.
Varchar data type also takes in size as the argument. But here, it is a variable length data type, unlike char. So here if the value size is to be 30 characters, and you give only 3 characters, the memory consumed would be only of 3 characters.
Text data type can take in a string with a maximum length of 65,535 characters.

3. Exact Numeric Datatype:

Numeric data types store all numerical values or integer values.

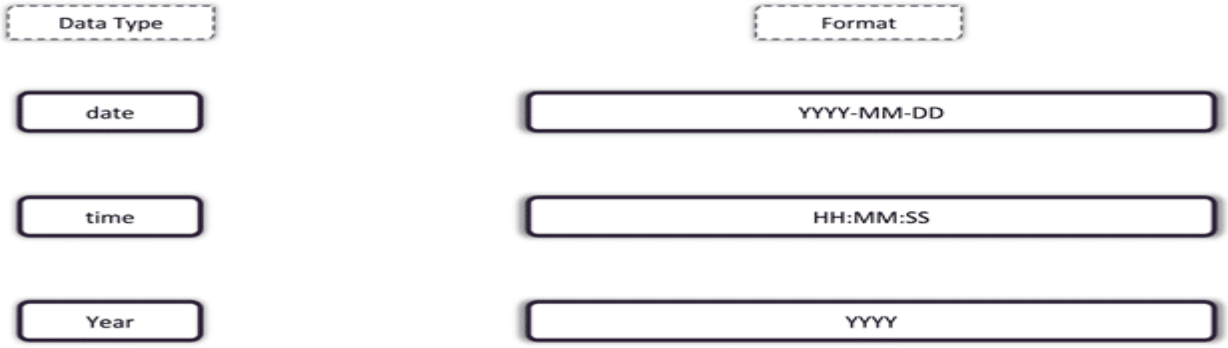


4. Approximate Numeric Datatype :

Data type	From	To	Description
float	-1.79E + 308	1.79E + 308	It is used to specify a floating-point value e.g. 6.2, 2.9 etc.
real	-3.40e + 38	3.40E + 38	It specifies a single precision floating point number

5.Date and time Datatypes:

Date and Time data types store a date or a date/time value.



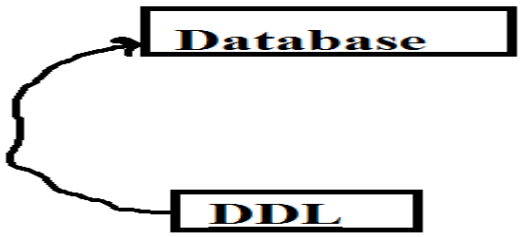
Date data type in SQL helps us specify the date in a format. Let's say, if we want to store the date, 2 January 2019, then first we will give the year which would be 2 0 1 9, then the month which would be 0 1, and finally, the day which would be 0 2.
Time data type helps us specify the time represented in a format. Let's say, we want to store the time 8:30:23 a.m. So, first we'll specify the hour which would be 0 8, then the minutes which would be 3 0, and finally the seconds which would be 2 3.
Year data type holds year values such as 1995 or 2011

Types of SQL commands:

SQL commands are categorized into four group:
1.DDL(Data Definition Language)
2.DML(Data Manipulation Language)
3.TCL(Transaction control Language)
4.DCL(Data Control Language)

1.DDL(Data Definition Language):

=>DDL Commands deal with the structure of the object.
=>DDL commands interact with the database directly.



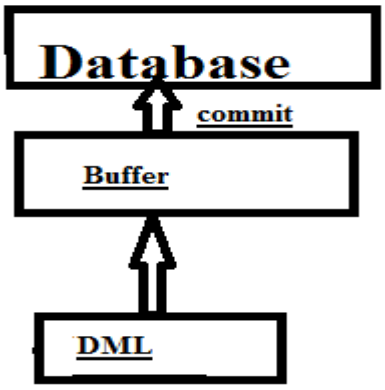
=>DDL Commands enforce an implicit commit before and after statement by database server.

=>implicit commit happens before commands irrespective either success or failure and if command success then only after commit will performed by database server.

=>cannot undo(Rollback) the changes because the implicit commit by database server.
=>DDL commands are faster as compared to DML commands in performance.

DDL Commands:
CREATE
ALTER
TRUNCATE
DROP

DML(Data Manipulation Language):
=>DML commands deals with the data only not with structure.
=>DML commands interact with buffer first and then Database on commit.



=>When we do commit then buffer data goes to the database.
=>We can Undo(Rollback) the changes before commit and buffer data will removed.
=>DML commands are slower in performance due to buffer involvement.
DML Commands:
DML commands are of two types:
Read Operation (Ex: SELECT Command)
Write Operation(Ex: INSERT, UPDATE, DELETE commands)

TCL(Transaction Control language):
=>TCL commands deal with the transaction only.
=>TCL commands are valid within transaction only.
Transaction: Starting from DML statement to COMMIT/ROLLBACK(implicit/explicit)
i.e A set of DML commands with commit or rollback are called transaction.
=>Every transaction start with DML write operation and commit or rollback are ending of transaction.
TCL commands:
Commit: to make the changes permanent.
Rollback: undo operation
Savepoint: temporary saving point within the transaction

DCL(Data Control language):

=>DCL commands deal with privileges only.i.e permission.
=>DCL Commands enforce an implicit commit before and after the statement and interact directly with database. So these are faster.
=>can not Undo(Rollback) the changes due to implicit commit by database server.

DCL commands:

1.Grant
2.Revoke
3.Set Role

1.DDL Commands:

1. Create:

Create table:

Syntax: CREATE TABLE table_name(column1 datatype[NULL|NOT NULL] ,
Column2 datatype[NULL|NOT NULL] ,

Column_n datatype[NULL|NOT NULL]);

Ex: CREATE TABLE customers(customer_id number(10) NOT NULL,
Customer_name varchar2(50) NOT NULL
City varchar2(40)
);