SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)

Re-accredited by NAAC with A+ grade, Accredited by NBA(CSE, IT, ECE, EEE & Mechanical)
Approvedy by AICTE, New Delhi, Recognized by UGC, Affiliated to Anna University, Chennai

Department of MCA

DBMS Introduction

Course Name : 19CAT609 - DATA BASE MANAGEMENT SYSTEM

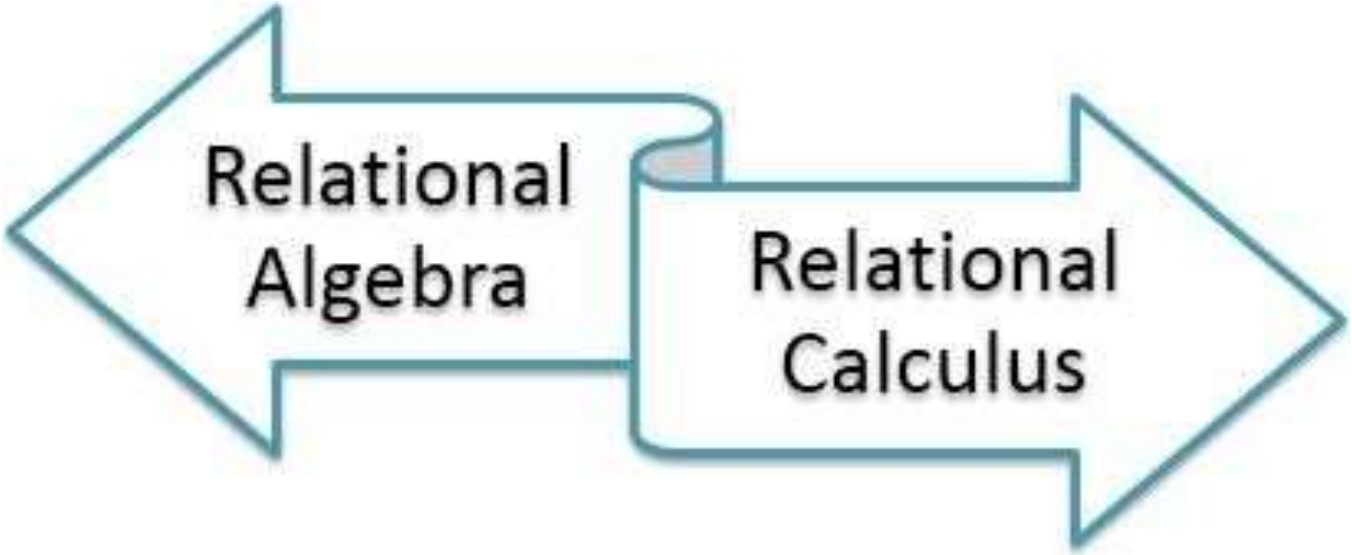
Class : I Year / II Semester

Unit II – Relational algebra and Calculus

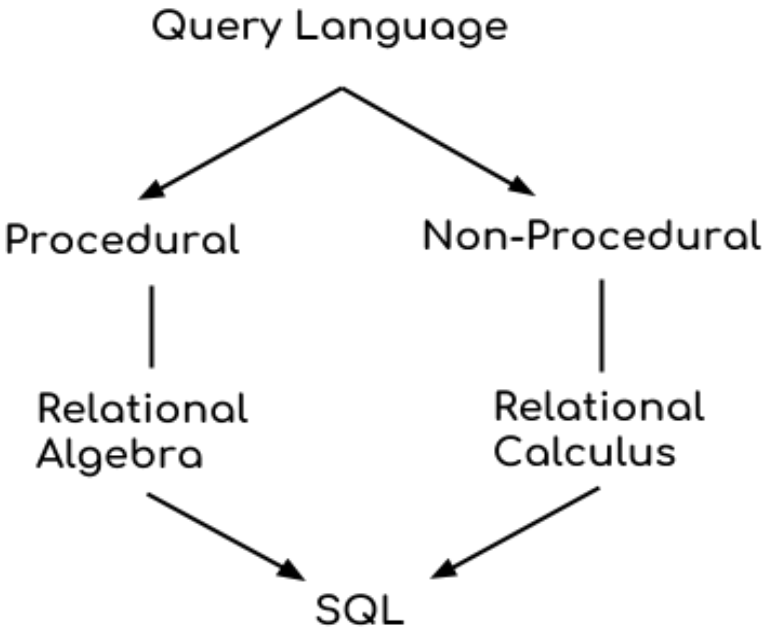
Agenda

- Introduction - Relational algebra and Calculus
- What is relational algebra?
- Relational Algebra and it’s Operations

Relational Algebra and Relational Calculus



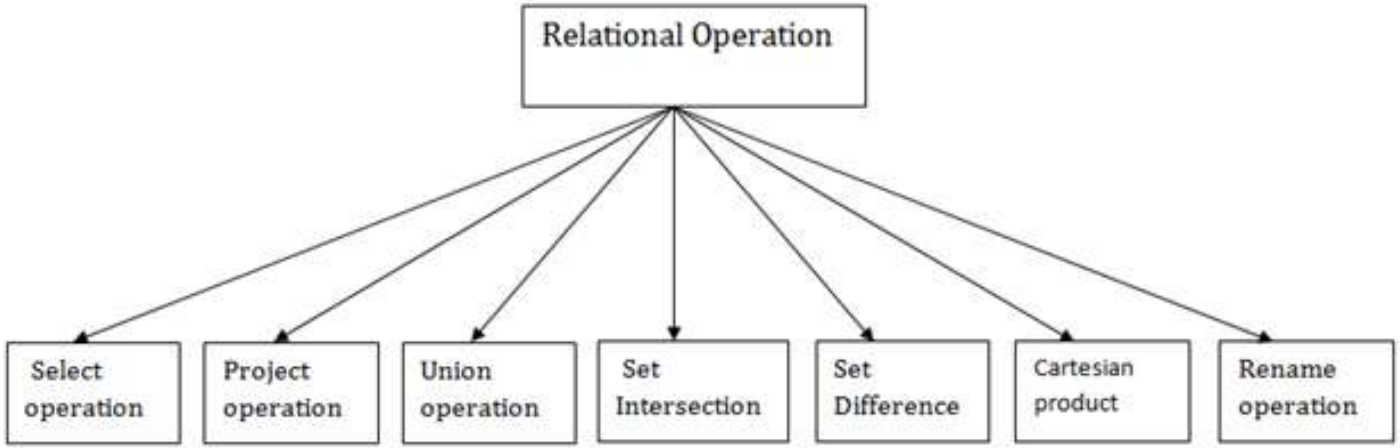
Relational algebra and Calculus



Relational Algebra

Relational algebra is a procedural query language. It gives a step by step process to obtain the result of the query. It uses operators to perform queries.

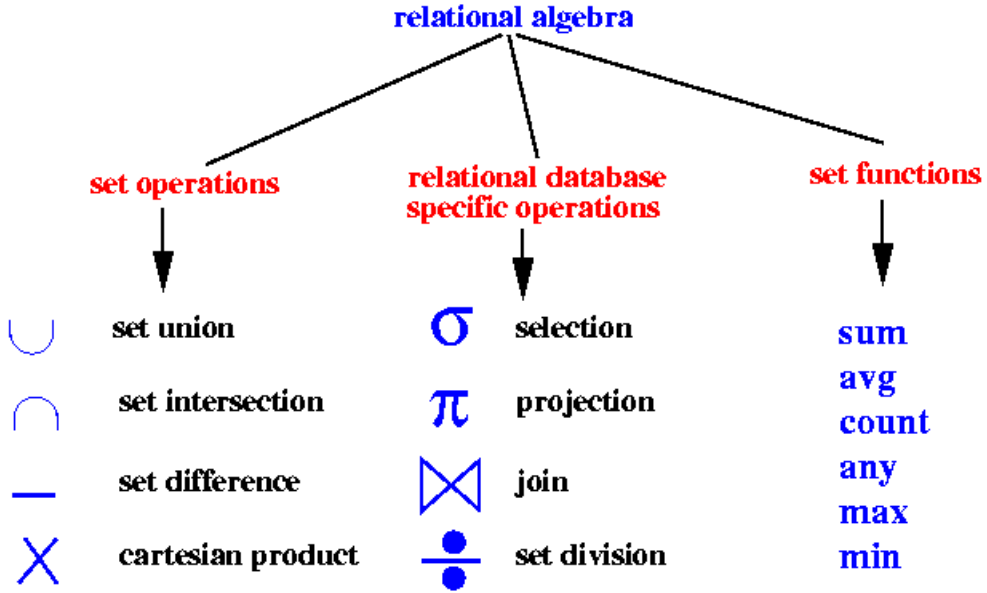
Types of Relational operation



Relational Algebra

Relational operation

Relational algebra is a procedural query language. It gives a step by step process to obtain the result of the query. It uses operators to perform queries.



Relational Algebra

Types of operations in relational algebra

We have divided these operations in two categories:

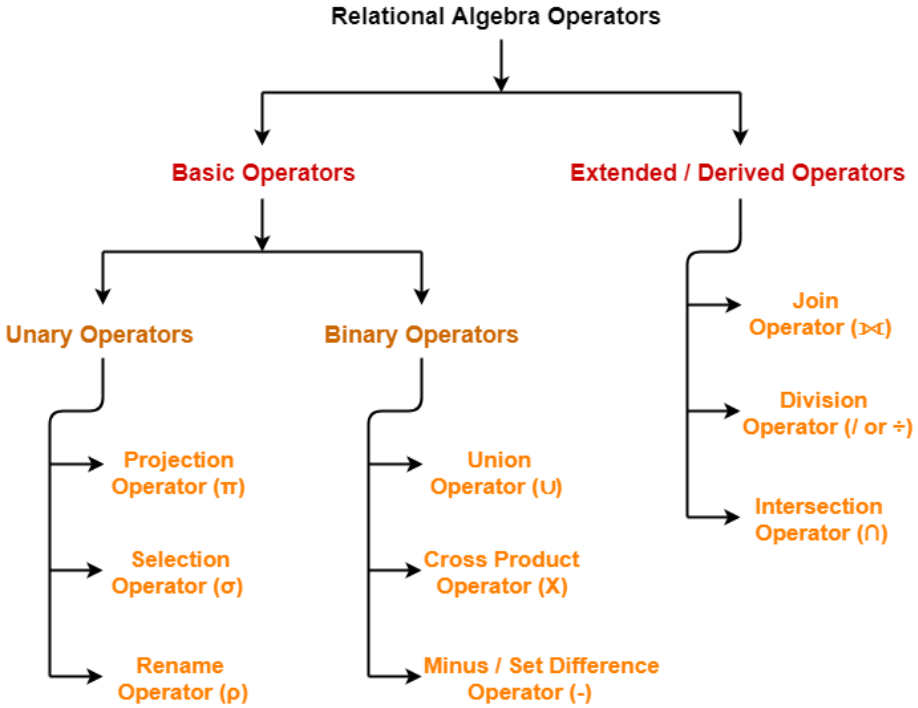
1. Basic Operations
2. Derived Operations

Basic/Fundamental Operations:

1. Select (σ)
2. Project (∏)
3. Union (∪)
4. Set Difference (-)
5. Cartesian product (X)
6. Rename (ρ)

Derived Operations:

1. Natural Join (⋈)
2. Left, Right, Full outer join (⟕, ⟖, ⟗)
3. Intersection (∩)
4. Division (÷)



Select Operator (σ)

•Select Operator is denoted by sigma (σ) and it is used to find the tuples (or rows) in a relation (or table) which satisfy the given condition.

•If you understand little bit of SQL then you can think of it as a **where clause in SQL**, which is used for the same purpose.

Relational Algebra

Syntax of Select Operator (σ)

$\sigma_{\text{Condition/Predicate}}(\text{Relation/Table name})$

Select Operator (σ) Example

Table: CUSTOMER

Query:

$\sigma_{\text{Customer\_City="Agra"}}(\text{CUSTOMER})$

Output:

Customer_Id	Customer_Name	Customer_City
C10100	Steve	Agra
C10111	Raghu	Agra

Table: CUSTOMER

Customer_Id	Customer_Name	Customer_City
C10100	Steve	Agra
C10111	Raghu	Agra
C10115	Chaitanya	Noida
C10117	Ajeet	Delhi
C10118	Carl	Delhi

Project Operator (∏)

Project operator is denoted by ∏ symbol and it is used to select desired columns (or attributes) from a table (or relation).

Project operator in relational algebra is similar to the **Select statement in SQL**.

Syntax of Project Operator (∏)

$$\prod_{column\_name1, column\_name2, ..., column\_nameN}(table\_name)$$

Project Operator (∏) Example

In this example, we have a table CUSTOMER with three columns, we want to fetch only two columns of the table, which we can do with the help of Project Operator ∏.

Table: CUSTOMER

Customer_Id	Customer_Name	Customer_City
C10100	Steve	Agra
C10111	Raghu	Agra
C10115	Chaitanya	Noida
C10117	Ajeet	Delhi
C10118	Carl	Delhi

Query:

```
∏ Customer_Name, Customer_City (CUSTOMER)
```

Output:

Customer_Name	Customer_City
Steve	Agra
Raghu	Agra
Chaitanya	Noida
Ajeet	Delhi
Carl	Delhi

Union Operator (U)

•Union operator is denoted by U symbol and it is used to select all the rows (tuples) from two tables (relations).

•Lets discuss union operator a bit more. Lets say we have two relations R1 and R2 both have same columns and we want to select all the tuples(rows) from these relations then we can apply the union operator on these relations.

•**Note:** The rows (tuples) that are present in both the tables will only appear once in the union set. In short you can say that there are no duplicates present after the union operation.

•**Syntax of Union Operator (U)**

•table_name1 U table_name2

Union Operator (U) Example

Table 1: COURSE

Course_Id	Student_Name	Student_Id
C101	Aditya	S901
C104	Aditya	S901
C106	Steve	S911
C109	Paul	S921
C115	Lucy	S931

Table 2: STUDENT

Student_Id	Student_Name	Student_Age
S901	Aditya	19
S911	Steve	18
S921	Paul	19
S931	Lucy	17
S941	Carl	16
S951	Rick	18

Query:

```
∏ Student_Name (COURSE) ∪ ∏ Student_Name (STUDENT)
```

OUTPUT

Student_Name
Aditya
Carl
Paul
Lucy
Rick
Steve

Note: As you can see there are no duplicate names present in the output even though we had few common names in both the tables, also in the COURSE table we had the duplicate name itself.

Intersection Operator (∩)

•Intersection operator is denoted by ∩ symbol and it is used to select common rows (tuples) from two tables (relations).

•Lets say we have two relations R1 and R2 both have same columns and we want to select all those tuples(rows) that are present in both the relations, then in that case we can apply intersection operation on these two relations R1 ∩ R2.

•**Note:** Only those rows that are present in both the tables will appear in the result set.

•**Syntax of Intersection Operator (∩)**

•table_name1 ∩ table_name2

Intersection Operator (∩)

Table 1: COURSE

Course_Id	Student_Name	Student_Id
C101	Aditya	S901
C104	Aditya	S901
C106	Steve	S911
C109	Paul	S921
C115	Lucy	S931

Table 2: STUDENT

Student_Id	Student_Name	Student_Age
S901	Aditya	19
S911	Steve	18
S921	Paul	19
S931	Lucy	17
S941	Carl	16
S951	Rick	

Query:

$\prod$ Student_Name (COURSE) ∩ $\prod$ Student_Name (STUDENT)

Student_Name
Aditya
Steve
Paul
Lucy

Set Difference (-)

•Set Difference is denoted by – symbol. Lets say we have two relations R1 and R2 and we want to select all those tuples(rows) that are present in Relation R1 but **not** present in Relation R2, this can be done using Set difference R1 – R2.

•**Syntax of Set Difference (-)**

•table_name1 - table_name2

Set Difference (-)

Query:

Lets write a query to select those student names that are present in STUDENT table but not present in COURSE table.

Π Student_Name (STUDENT) - Π Student_Name (COURSE)

Output:

```
Student_Name
------------
Carl
Rick
```

Cartesian product (X)

•Cartesian Product is denoted by X symbol. Lets say we have two relations R1 and R2 then the cartesian product of these two relations (R1 X R2) would combine each tuple of first relation R1 with the each tuple of second relation R2.

•I know it sounds confusing but once we take an example of this, you will be able to understand this.

•**Syntax of Cartesian product (X)**

•R1 X R2

Cartesian product (X)

- Table 1: R

Col_A	Col_B
AA	100
BB	200
CC	300

Table 2: S

Col_X	Col_Y
XX	99
YY	11
ZZ	101

output:

Col_A	Col_B	Col_X	Col_Y
AA	100	XX	99
AA	100	YY	11
AA	100	ZZ	101
BB	200	XX	99
BB	200	YY	11
BB	200	ZZ	101
CC	300	XX	99
CC	300	YY	11
CC	300	ZZ	101

Query:

Lets find the cartesian product of table R and S.

R X S

Note: The number of rows in the output will always be the cross product of number of rows in each table. In our example table 1 has 3 rows and table 2 has 3 rows so the output has 3×3 = 9 rows.

Cartesian product (X)

- Table 1: R

Col_A	Col_B
AA	100
BB	200
CC	300

Table 2: S

Col_X	Col_Y
XX	99
YY	11
ZZ	101

output:

Col_A	Col_B	Col_X	Col_Y
AA	100	XX	99
AA	100	YY	11
AA	100	ZZ	101
BB	200	XX	99
BB	200	YY	11
BB	200	ZZ	101
CC	300	XX	99
CC	300	YY	11
CC	300	ZZ	101

Query:

Lets find the cartesian product of table R and S.

R X S

Note: The number of rows in the output will always be the cross product of number of rows in each table. In our example table 1 has 3 rows and table 2 has 3 rows so the output has 3×3 = 9 rows.

Rename (ρ)

Rename (ρ) operation can be used to rename a relation or an attribute of a relation. **Rename (ρ) Syntax:**

ρ(new_relation_name, old_relation_name)

Rename (ρ) Example

Lets say we have a table customer, we are fetching customer names and we are renaming the resulted relation to CUST_NAMES.

Table: CUSTOMER

Customer_Id	Customer_Name	Customer_City
C10100	Steve	Agra
C10111	Raghu	Agra
C10115	Chaitanya	Noida
C10117	Ajeet	Delhi
C10118	Carl	Delhi

ρ(CUST_NAMES, ∏(Customer_Name)(CUSTOMER))

CUST_NAMES
Steve
Raghu
Chaitanya
Ajeet
Carl

Relational Algebra **Join Operations**

Extended operators are those operators which can be derived from basic operators. There are mainly three types of extended operators in Relational Algebra:

- **Join**
- **Intersection**
- **Divide**

Example Consider the following tables.

Table A	
column 1	column 2
1	1
1	2

Table B	
column 1	column 2
1	1
1	3

Join Operations

- Join Operations
- Inner Join:
- Theta Join:
- EQUI join:
- NATURAL JOIN (⋈)
- OUTER JOIN
- Left Outer Join(A B)
- Right Outer Join: (A B)
- Full Outer Join: (A B)

Join Operations

Join Operations

Join operation is essentially a cartesian product followed by a selection criterion.

Join operation denoted by ⋈.

JOIN operation also allows joining variously related tuples from different relations.

Types of JOIN:

Various forms of join operation are:

Inner Joins:

Theta join

EQUI join

Natural join

Outer join:

Left Outer Join

Right Outer Join

Full Outer Join

Inner Join

Inner Join:

In an inner join, only those tuples that satisfy the matching criteria are included, while the rest are excluded. Let's study various types of Inner Joins:

Theta Join:

The general case of JOIN operation is called a Theta join. It is denoted by symbol **θ**

Example

$A \bowtie_{\theta} B$

Theta join can use any conditions in the selection criteria.

For example:

$A \bowtie_{A.column\ 2 > B.column\ 2} (B)$

A ⋈ A.column 2 > B.column 2 (B)	
column 1	column 2
1	2

EQUI join

EQUI join:

When a theta join uses only equivalence condition, it becomes a equi join.

For example:

$A \bowtie_{A.column\ 2 = B.column\ 2} (B)$

A ⋈ A.column 2 = B.column 2 (B)	
column 1	column 2
1	1

EQUI join is the most difficult operations to implement efficiently using SQL in an RDBMS and one reason why RDBMS have essential performance problems.

NATURAL JOIN (⋈)

Natural join can only be performed if there is a common attribute (column) between the relations. The name and type of the attribute must be same.

Example: Consider the following two tables

	D
Num	Cube
2	8
3	27

	C
Num	Square
2	4
3	9

C ⋈ D

	C ⋈ D	
Num	Square	Cube
2	4	8
3	9	27

Left Outer Join

OUTER JOIN

In an outer join, along with tuples that satisfy the matching criteria, we also include some or all tuples that do not match the criteria.

Left Outer Join(A B)

In the left outer join, operation allows keeping all tuple in the left relation. However, if there is no matching tuple is found in right relation, then the attributes of right relation in the join result are filled with null values.

Left Outer Join

LOJ

	A
Num	Square
2	4
3	9
4	16

	B
Num	Cube
2	8
3	18
5	75

Join(A B)

	A ⋈ B	
Num	Square	Cube
2	4	8
3	9	18
4	16	–

Right Outer Join: (A⟖B)

In the right outer join, operation allows keeping all tuple in the right relation. However, if there is no matching tuple is found in the left relation, then the attributes of the left relation in the join result are filled with null values.

Right Outer Join

Right Outer Join: (A ⟖ B)

	A
Num	Square
2	4
3	9
4	16

	B
Num	Cube
2	8
3	18
5	75

Join(A ⟖ B)

	A ⋈ B	
Num	Cube	Square
2	8	4
3	18	9
5	75	–

Full Outer Join

Full Outer Join: (A⟗ B)

In a full outer join, all tuples from both relations are included in the result, irrespective of the matching condition.

Full Outer Join

Full Outer Join: (A⟗ B)

	A
Num	Square
2	4
3	9
4	16

	B
Num	Cube
2	8
3	18
5	75

Join(A ⟗B)

		A ⋈ B
Num	Cube	Square
2	4	8
3	9	18
4	16	–
5	–	75

Intersection

Intersection

An intersection is defined by the symbol ∩

A ∩ B

Defines a relation consisting of a set of all tuple that are in both A and B. However, A and B must be union-compatible.

Example:

A ∩ B

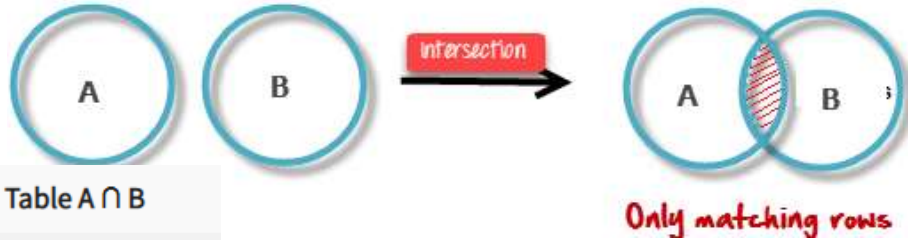


Table A ∩ B

column 1	column 2
1	1

Division Operator

Division Operator (÷): 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.
Consider the relation STUDENT_SPORTS and ALL_SPORTS given in Table 2 and Table 3 above.
To apply division operator as

STUDENT_SPORTS÷ ALL_SPORTS

- The operation is valid as attributes in ALL_SPORTS is a proper subset of attributes in STUDENT_SPORTS.
- The attributes in resulting relation will have attributes {ROLL_NO,SPORTS}-{SPORTS}=ROLL_NO
- The tuples in resulting relation will have those ROLL_NO which are associated with all B's tuple {Badminton, Cricket}. ROLL_NO 1 and 4 are associated to Badminton only. ROLL_NO 2 is associated to all tuples of B. So the resulting relation will be:

Division Operations

STUDENT

ROLL_NO	NAME	ADDRESS	PHONE	AGE
1	RAM	DELHI	9455123451	18
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	DELHI	9156768971	18

Table 3

EMPLOYEE

EMP_NO	NAME	ADDRESS	PHONE	AGE
1	RAM	DELHI	9455123451	18
5	NARESH	HISAR	9782918192	22
6	SWETA	RANCHI	9852617621	21
4	SURESH	DELHI	9156768971	18

Table 1

STUDENT_SPORTS

ROLL_NO	SPORTS
1	Badminton
2	Cricket
2	Badminton

Table 2

ALL_SPORTS

SPORTS
Badminton
Cricket

STUDENT_SPORTS÷ ALL_SPORTS

ROLL_NO

2

Summary

Operation(Symbols)	Purpose
Select(σ)	The SELECT operation is used for selecting a subset of the tuples according to a given selection condition
Projection(π)	The projection eliminates all attributes of the input relation but those mentioned in the projection list.
Union Operation(∪)	UNION is symbolized by symbol. It includes all tuples that are in tables A or in B.
Set Difference(-)	– Symbol denotes it. The result of A – B, is a relation which includes all tuples that are in A but not in B.
Intersection(∩)	Intersection defines a relation consisting of a set of all tuple that are in both A and B.
Cartesian Product(X)	Cartesian operation is helpful to merge columns from two relations.
Inner Join	Inner join, includes only those tuples that satisfy the matching criteria.
Theta Join(θ)	The general case of JOIN operation is called a Theta join. It is denoted by symbol θ.
EQUI Join	When a theta join uses only equivalence condition, it becomes a equi join.

Summary

Operation(Symbols)	Purpose
Natural Join(⋈)	Natural join can only be performed if there is a common attribute (column) between the relations.
Outer Join	In an outer join, along with tuples that satisfy the matching criteria.
Left Outer Join( )	In the left outer join, operation allows keeping all tuple in the left relation.
Right Outer join()	In the right outer join, operation allows keeping all tuple in the right relation.
Full Outer Join()	In a full outer join, all tuples from both relations are included in the result irrespective of the matching condition.

Company Database

- Company Database Entities that would be represented
- employee
- department
- salary
- management
- inventory
- and other such similar entities

References

1. https://www.javatpoint.com/dbms-relational-algebra
2. https://beginnersbook.com/2019/02/dbms-relational-algebra/
3. https://www.tutorialspoint.com/dbms/relational_algebra.htm