DBMS

(DATABASE MANAGEMENT SYSTEM)

LESSON 6

Relational Algebra:

Relational Algebra is procedural query language, which takes Relation as input and generate relation as output. Relational algebra mainly provides theoretical foundation for relational databases and SQL.

Operators in Relational Algebra

Projection (π)

Projection is used to project required column data from a relation.

Example :

```
   R
 (A  B  C)
 ----------
  1  2  4
  2  2  3
  3  2  3
  4  3  4
```

```
π (BC)
B  C
-----
2  4
2  3
3  4
```

Note: By Default projection removes duplicate data.

Selection (σ)

Selection is used to select required tuples of the relations.

for the above relation

$\sigma (c>3)R$

will select the tuples which have c more than 3.

Note: selection operator only selects the required tuples but does not display them. For displaying, data projection operator is used.

For the above selected tuples, to display we need to use projection also.

$\pi (\sigma (c>3)R)$ will show following tuples.

A	B	C
1	2	4
4	3	4

Union (U)

Union operation in relational algebra is same as union operation in set theory, only constraint is for union of two relation both relation must have same set of Attributes.

Set Difference (-)

Set Difference in relational algebra is same set difference operation as in set theory with the constraint that both relation should have same set of attributes.

Rename (ρ)

Rename is a unary operation used for renaming attributes of a relation.

$\rho (a/b)R$ will rename the attribute ‘b’ of relation by ‘a’.

Cross Product (X)

Cross product between two relations let say A and B, so cross product between A X B will results all the attributes of A followed by each attribute of B. Each record of A will pairs with every record of B.

below is the example

A

Name	Age	Sex
Ram	14	M
Sona	15	F
kim	20	M

B

Id	Course
1	DS
2	DBMS

A X B

Name	Age	Sex	Id	Course
Ram	14	M	1	DS
Ram	14	M	2	DBMS
Sona	15	F	1	DS
Sona	15	F	2	DBMS
Kim	20	M	1	DS
Kim	20	M	2	DBMS

Note: if A has 'n' tuples and B has 'm' tuples then A X B will have 'n*m' tuples.

Natural Join (⋈):

Natural join is a binary operator. Natural join between two or more relations will result set of all combination of tuples where they have equal common attribute.

Let us see below example

Emp

(Name	Id	Dept_name)
A	120	IT
B	125	HR
C	110	Sale
D	111	IT

Dep

(Dept_name	Manager)
Sale	Y
Prod	Z
IT	A

Emp ⋈ Dep

Name	Id	Dept_name	Manager
A	120	IT	A
C	110	Sale	Y
D	111	IT	A

EXTENDED OPERATORS

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**

Intersection (∩): Intersection on two relations R1 and R2 can only be computed if R1 and R2 are **union compatible** (These two relation should have same number of attributes and corresponding attributes in two relations have same domain). Intersection operator when applied on two relations as R1∩R2 will give a relation with tuples which are in R1 as well as R2. Syntax:

Relation1 ∩ Relation2

Example: Find a person who is student as well as employee- **STUDENT ∩ EMPLOYEE**

In terms of basic operators (union and minus) :

STUDENT ∩ EMPLOYEE = STUDENT + EMPLOYEE - (STUDENT U EMPLOYEE)

RESULT:

ROLL_NO	**NAME**	**ADDRESS**	**PHONE**	**AGE**
1	RAM	DELHI	9455123451	18
4	SURESH	DELHI	9156768971	18

Conditional Join($\bowtie_c$): Conditional Join is used when you want to join two or more relation based on some conditions. Example: Select students whose ROLL_NO is greater than EMP_NO of employees

STUDENT⋈c STUDENT.ROLL_NO>EMPLOYEE.EMP_NOEMPLOYEE

In terms of basic operators (cross product and selection) :

σ (STUDENT.ROLL_NO>EMPLOYEE.EMP_NO)(STUDENT×EMPLOYEE)

RESULT:

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

Equijoin(⋈): Equijoin is a **special case of conditional join** where only equality condition holds between a pair of attributes. As values of two attributes will be equal in result of equijoin, only one attribute will be appeared in result.

Example:Select students whose ROLL_NO is equal to EMP_NO of employees

$$\text{STUDENT} \bowtie_{\text{STUDENT.ROLL\_NO=EMPLOYEE.EMP\_NO}} \text{EMPLOYEE}$$

In terms of basic operators (cross product, selection and projection) :

$\Pi_{\text{(STUDENT.ROLL\_NO, STUDENT.NAME, STUDENT.ADDRESS, STUDENT.PHONE, STUDENT.AGE EMPLOYEE.NAME, EMPLOYEE.ADDRESS, EMPLOYEE.PHONE, EMPLOYEE>AGE)}}(\sigma_{\text{(STUDENT.ROLL\_NO=EMPLOYEE.EMP\_NO)}}\ (\text{STUDENT} \times \text{EMPLOYEE}))$

RESULT:

ROLL_NO	NAME	ADDRESS	PHONE	AGE	NAME	ADDRESS	PHONE	AGE
1	RAM	DELHI	9455123451	18	RAM	DELHI	9455123451	18
4	SURESH	DELHI	9156768971	18	SURESH	DELHI	9156768971	18

Natural Join(⋈): It is a special case of equijoin in which equality condition hold on all attributes which have same name in relations R and S (relations on which join operation is applied). While applying natural join on two relations, there is no need to write equality condition explicitly. Natural Join will also return the similar attributes only once as their value will be same in resulting relation.

Example: Select students whose ROLL_NO is equal to ROLL_NO of STUDENT_SPORTS as:

STUDENT⋈STUDENT_SPORTS

In terms of basic operators (cross product, selection and projection) :

Π(STUDENT.ROLL_NO, STUDENT.NAME, STUDENT.ADDRESS, STUDENT.PHONE, STUDENT.AGE STUDENT_SPORTS.SPORTS)(σ (STUDENT.ROLL_NO=STUDENT_SPORTS.ROLL_NO) (STUDENT×STUDENT_SPORTS))

RESULT:

ROLL_NO	NAME	ADDRESS	PHONE	AGE	SPORTS
1	RAM	DELHI	9455123451	18	Badminton
2	RAMESH	GURGAON	9652431543	18	Cricket

2	RAMESH	GURGAON	9652431543	18	Badminton
4	SURESH	DELHI	9156768971	18	Badminton

Natural Join is by default inner join because the tuples which does not satisfy the conditions of join does not appear in result set. e.g.; The tuple having ROLL_NO 3 in STUDENT does not match with any tuple in STUDENT_SPORTS, so it has not been a part of result set.

Left Outer Join(⟕): When applying join on two relations R and S, some tuples of R or S does not appear in result set which does not satisfy the join conditions. But Left Outer Joins gives all tuples of R in the result set. The tuples of R which do not satisfy join condition will have values as NULL for attributes of S.

Example:Select students whose ROLL_NO is greater than EMP_NO of employees and details of other students as well

STUDENT&#10197$_{\text{STUDENT.ROLL\_NO>EMPLOYEE.EMP\_NO}}$**EMPLOYEE**

RESULT

ROLL_NO	NAME	ADDRESS	PHONE	AGE	EMP_NO	NAME	ADDRESS	PHONE	AGE

2	RAMESH	GURGAON	9652431543	18	1	RAM	DELHI	9455123451	18
3	SUJIT	ROHTAK	9156253131	20	1	RAM	DELHI	9455123451	18
4	SURESH	DELHI	9156768971	18	1	RAM	DELHI	9455123451	18
1	RAM	DELHI	9455123451	18	NULL	NULL	NULL	NULL	NULL

Right Outer Join(⟖): When applying join on two relations R and S, some tuples of R or S does not appear in result set which does not satisfy the join conditions. But Right Outer Joins gives all tuples of S in the result set. The tuples of S which do not satisfy join condition will have values as NULL for attributes of R.

Example: Select students whose ROLL_NO is greater than EMP_NO of employees and details of other Employees as well

$\text{STUDENT} ⟖_{\text{STUDENT.ROLL\_NO}>\text{EMPLOYEE.EMP\_NO}} \text{EMPLOYEE}$

RESULT:

ROLL_NO	NAME	ADDRESS	PHONE	AGE	EMP_NO	NAME	ADDRESS	PHONE	AGE
2	RAMESH	GURGAON	9652431543	18	1	RAM	DELHI	9455123451	18
3	SUJIT	ROHTAK	9156253131	20	1	RAM	DELHI	9455123451	18
4	SURESH	DELHI	9156768971	18	1	RAM	DELHI	9455123451	18
NULL	NULL	NULL	NULL	NULL	5	NARESH	HISAR	9782918192	22

NULL	NULL	NULL	NULL	NULL	6	SWETA	RANCHI	9852617621	21
NULL	NULL	NULL	NULL	NULL	4	SURESH	DELHI	9156768971	18

Full Outer Join(⟗): When applying join on two relations R and S, some tuples of R or S does not appear in result set which does not satisfy the join conditions. But Full Outer Joins gives all tuples of S and all tuples of R in the result set. The tuples of S which do not satisfy join condition will have values as NULL for attributes of R and vice versa.

Example:Select students whose ROLL_NO is greater than EMP_NO of employees and details of other Employees as well and other Students as well

STUDENT⟗STUDENT.ROLL_NO>EMPLOYEE.EMP_NOEMPLOYEE

RESULT:

ROLL_NO	**NAME**	**ADDRESS**	**PHONE**	**AGE**	**EMP_NO**	**NAME**	**ADDRESS**	**PHONE**	**AGE**

2	RAMESH	GURGAON	9652431543	18	1	RAM	DELHI	9455123451	18
3	SUJIT	ROHTAK	9156253131	20	1	RAM	DELHI	9455123451	18
4	SURESH	DELHI	9156768971	18	1	RAM	DELHI	9455123451	18
NULL	NULL	NULL	NULL	NULL	5	NARESH	HISAR	9782918192	22
NULL	NULL	NULL	NULL	NULL	6	SWETA	RANCHI	9852617621	21

NULL	NULL	NULL	NULL	NULL	4	SURESH	DELHI	9156768971	18
1	RAM	DELHI	9455123451	18	NULL	NULL	NULL	NULL	NULL

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:

TUPLE RELATIONAL CALCULUS

Tuple Relational Calculus is a **non-procedural query language** unlike relational algebra. Tuple Calculus provides only the description of the query but it does not provide the methods to solve it. Thus, it explains what to do but not how to do.

In Tuple Calculus, a query is expressed as

{t| P(t)}

where t = resulting tuples,

P(t) = known as Predicate and these are the conditions that are used to fetch t

Thus, it generates set of all tuples t, such that Predicate P(t) is true for t.

P(t) may have various conditions logically combined with OR (∨), AND (∧), NOT(¬).

It also uses quantifiers:

∃ t ∈ r (Q(t)) = "there exists" a tuple in t in relation r such that predicate Q(t) is true.

$\forall\ t \in r\ (Q(t))$ = Q(t) is true “for all” tuples in relation r.