Formal Relational Languages

Relational Algebra: The basic set of operations for the relational model is the relational algebra. These operations enable a user to specify basic retrieval requests as relational algebra expressions.

The relational algebra provides a formal foundation for relational model operations. It is also used as a basis for implementing and optimizing queries in the query processing and optimization modules that are integral parts of RDMSs.

Operations of Relational Algebra:-

Select:- selects all tuples that satisfy the selection condition from a relation R.

Notation:- $\sigma_{\langle \text{selection condition} \rangle}(R)$

Eg:- Select the tuples for all employees who either work in dept 4 and make over 25,000 per year, or work in dept 5 and make over 35000.

$$\sigma_{(Dno = 4 \text{ and } Salary > 25000) \text{ or } (Dno = 5 \text{ and } Salary > 35000)}(emp)$$

Project:- produces ~~all tuples that satisfy~~ a new relation with only some of the attributes of R, and removes duplicate tuples.

Notation:- $\pi_{\langle \text{attribute list} \rangle}(R)$

Eg:- List each employee's first and last name and salary.

$$\pi_{Fname, Lname, Salary}(emp).$$

Rename :- Relation names and attribute names can be renamed using ρ operator.

Notation :- $\rho_S(R)$ or $\rho_{S(B_1, B_2, \ldots, B_n)}(R)$ or $\rho_{(B_1, B_2, \ldots, B_n)}(R)$

Eg :- ρ_{emp}(employee)

Relational algebra operations from Set Theory :-

UNION, INTERSECT, MINUS (SET DIFFERENCE)

UNION :- produces a relation that includes all the tuples in ~~both~~ R_1 ~~and~~ or R_2 or both R_1 and R_2; R_1 and R_2 must be union compatible - Same no. of attributes and each pair of attributes from same domain.

Notation :- $R_1 \cup R_2$

Eg :- Retrieve the SSNs of all employees who either work in Dept 5 or directly supervise an employee who works in Dept 5.

Dept5_emps $\leftarrow \sigma_{dno=5}$(emp)

Result1 $\leftarrow \pi_{ssn}$(Dept5_emps)

Result2(ssn) $\leftarrow \pi_{super\_ssn}$(Dept5_emps)

Result $\leftarrow$ Result1 $\cup$ Result2

Intersection :- produces a relation that includes all the tuples in both R_1 and R_2; R_1 and R_2 must be union compatible.

Notation :- $R_1 \cap R_2$

Eg :- Student $\cap$ Instructor

Difference :- produces a relation that includes all the tuples in R_1 that are not in R_2; R_1 and R_2 must be union compatible.

Notation :- $R_1 - R_2$

Eg :- Student - Instructor

Cartesian product :- produces a relation that has the attributes of R_1 and R_2 and includes ~~all~~ tuples all combinations of tuples from R_1 and R_2

Notation :- $R_1 \times R_2$

Binary relational operations JOIN and Division :-

The JOIN operation denoted by $\bowtie$, is used to combine related tuples from two relations into single "Longer" tuples.

notation :- $R \bowtie_{\langle join\text{-}condition \rangle} S$

The result of the join is a relation Q with n+m attributes $Q(A_1, A_2 \ldots A_n, B_1, B_2 \ldots B_m)$ in that order; Q has one tuple for each combination of tuples - one from R and one from S - whenever the combination satisfies the join condition.

Eg :- Retrieve the name of the manager of each department

$$Dept\_mgr \leftarrow Dept \bowtie_{mgr\text{-}ssn = ssn} Emp$$

$$Result \leftarrow \pi_{Dname,\, Lname,\, Fname} (Dept\_mgr)$$

The main difference between Cartesian product and join is:

In join, only combinations of tuples satisfying the join condition appear in the result, whereas in the Cartesian product all combinations of tuples are included in the result.

- A join operation with ~~such~~ a general join condition is called a Theta join (θ) $(\theta = \{=, <, \leq, >, \geq, \neq\})$

A join, where the only comparison operator used is =, is called an equi join.

A natural join denoted by * was created to get rid of the second attribute in an equijoin condition.

Eg :- proj_dept ← project * Dept

For natural join the two join attributes ~~ha~~ have to be same name in ~~two~~ both relations. If not, a renaming operation is applied first

eg :- Proj_dept ← project * $\rho_{(Dname, Dnum, mgr\_ssn)}$ (Department)

The attribute Dnum is the join attribute here.

A Complete set of Relational Algebra operations :-

The set of relational algebra operations $\{\sigma, \pi, \cup, \rho, -, \times\}$ is a Complete set ; that is, any of the other relational algebra operations Can be expressed as a sequence of operations from this set.

Eg:- $R \cap S = (R \cup S) - ((R-S) \cup (S-R))$

$R \bowtie_{<condition>} S = \sigma_{condition}(R \times S)$

The Division operation :-

The division operation is applied to two relations $R(Z) \div S(X)$, where the attributes of R are a subset of the attributes of S ; that is, $X \subseteq Z$

Eg:- Retrieve the names of employees who work on all the projects that 'John Smith' works on

$SMITH \leftarrow \sigma_{Fname='John' \text{ and } Lname='Smith'}(EMP)$

$SMITH\_PNOS \leftarrow \pi_{Pno}(WORKS\_ON \bowtie_{Essn=ssn} SMITH)$

$SSN\_PNOS \leftarrow \pi_{Essn, Pno}(WORKS\_ON)$

$SSNS \leftarrow SSN\_PNOS \div SMITH\_PNOS$

$RESULT \leftarrow \pi_{Fname, Lname}(SSNS * EMP)$

SSN_PNOS

Essn	Pno
123456789	1
123456789	2
666884444	3
453453453	1
453453453	2
333445555	2
333445555	3
333445555	10
999887777	30
999887777	10
987987987	30
987654321	20
888665555	20

SMITH_PNOS

Pno
1
2

SSNS

ssn
123456789
453453453

The division operation Can be expressed as a sequence of π, $\times$ and $-$ operations as follows:

$T_1 \leftarrow \pi_Y(R)$, $T_2 \leftarrow \pi_Y((S \times T_1) - R)$

$T \leftarrow T_1 - T_2$

Let Y be the set of attributes of R that are not attributes of S, i.e $Y = Z - X$

The result of Division is a relation $T(Y)$ that includes a tuple t if tuples t_R appear in R with $t_R[Y] = t$ and with $t_R[X] = t_s$ for every tuple t_s in S.

Eg:- $X = \{A\}$ $Y = \{B\}$ and $Z = \{A, B\}$

R

A	B
a_1	b_1
a_2	b_1
a_3	b_1
a_4	b_1
a_1	b_2
a_3	b_2
a_2	b_3
a_3	b_3
a_4	b_3

S

A
a_1
a_2
a_3

$R \div S \rightarrow$ T

B
b_1
b_4

For a tuple t to appear in the result T of the division, the values in t must appear in R in Combination with every tuple in S.