Unit : 02 Relational Algebra

Date
Page

→ Cardinality of a table = Number of tuples (rows)
→ Degree of a table = Number of columns.
→ Domain is a set of all possible unique values for a specific column.

* Basic relational algebra operations :-

1. Selection (σ) sigma :- (Row / tuple)
→ Returns all tuples which satisfies the condition.
→ $\sigma_c(R)$
→ Ex :- $\sigma_{\text{salary} > 8000}(\text{Employee})$, $\sigma_{\text{name} = \text{'Smith'}}(\text{Employee})$
→ Condition can be $=, >, <, \geq, \leq, <>, \wedge, \vee$ (AND) (OR)
<> : not equal to

RA : $\sigma_{\text{gpa} > 3.5}(\text{students})$ SQL : select * from students where gpa > 3.5.

Example : Write down relational algebra for the following table :

Employee :

empid	name	dept	salary
101	Nilesh	Sales	10000
102	Mayur	HR	25000
103	Hardik	HR	15000
104	Ajay	Admin	20000.

1). Display the details of all employees.
→ RA : $\sigma(\text{Employee})$
→ SQL : select * from Employee

2) Display the details of employees whose salary is more than 10000.

→ R.A : $\sigma_{salary > 10000}(Employee)$

→ SQL : select * from Employee
where salary > 10000.

3) Display the details of employees belong to HR department having salary more than 20000.

→ RA : $\sigma_{dept = 'HR' \wedge salary > 20000}(Employee)$

→ SQL : select * from Employee
where dept = 'HR' and salary > 20000

4) Display the details of employees belong to either 'HR' or 'Admin' dept.

→ RA : $\sigma_{dept = 'HR' \vee dept = 'Admin'}(Employee)$

→ SQL : select * from Employee
where dept = 'HR' or dept = 'Admin'

5) Display the details of Employees whose salary lies between 10000 and 25000 and belong to 'HR' dept.

→ RA : $\sigma_{salary > 10000 \wedge salary < 25000 \wedge dept = 'HR'}(Employee)$

→ SQL : select * from Employee
where salary between 10000 and 25000
and dept = 'HR'.

2. Projection (π) pie :- (Column)

→ selects specified attributes / columns of a relation.
→ It removes duplicate tuples / rows / records from the result.
→ $\pi_{A1, A2, \ldots, An}(R)$

→ Ex : $\pi_{name}(Employee)$, $\pi_{no, name, id}(Employee)$

→ RA : $\pi_{sname, sno}(Students)$

→ SQL : select distinct sname, sno from students

distinct used to remove duplicates from the result.

Example : Write down relational algebra for the following table.

Students :

Rollno	Name	Branch	SPI
101	Raj	CE	6
102	Meet	ME	8
103	HARSH	EE	7
104	Punit	CE	9

1) Display Rollno, Name and SPI of all students.
→ RA : $\pi_{Rollno, Name, SPI}(Students)$

→ SQL : select distinct Rollno, Name, SPI from students

2) Display name and SPI of all students.

→ RA : $\sigma_{Name, SPI}$ (Students)

→ SQL : select distinct Name, SPI from students.

3) Display the name of all students.

→ RA : σ_{Name} (Students)

→ SQL : select distinct Name from students.

4) Display the name of all branches.

→ RA : σ_{Branch} (Student)

→ SQL : select distinct Branch from students.

* Combination of selection and projection operations :-

Example :- Write down relational algebra for the following table.

Faculty :

Faculty ID	Name	Branch	Salary.
101	B.M. Patel	CE	25000
102	A.M. Shah	ME	35000
103	J.B. Katariya	EE	18000
104	H.N. Shukla	CE	50000
105	V.K. Vyas.	ME	45000.

1) Display the name of faculties belong to 'CE' branch and having salary more than 25000.

→ RA : $\Pi_{Name}(\sigma_{Branch='CE' \wedge Salary > 25000}(Faculty))$

→ SQL : select Name from Faculty
where Branch = 'CE' and
Salary > 25000.

2) Display the name of all 'CE' and 'ME' branch's faculties.

→ RA : $\Pi_{Name}(\sigma_{Branch='CE' \vee Branch='ME'}(Faculty))$

→ SQL : select Name from Faculty
where Branch = 'CE' OR Branch = 'ME'

3) List the name of faculty with their salary who belongs to 'CE' OR 'ME' branch having salary more than 35000.

→ RA : $\Pi_{Name, Salary}(\sigma_{(Branch='CE' \vee Branch='ME') \wedge Salary > 35000}(Faculty))$

→ SQL : select Name, Salary from Faculty
where Branch = 'CE' OR
Branch = 'ME' and
Salary > 35000

relation_name . attribute_name

4) Display the name of faculty along with their branch name whose salary lies between 25000 and 50000.

→ RA : $\Pi_{Name, Branch}(\sigma_{Salary > 25000 \wedge Salary < 50000}(Faculty))$

3. Cartesian product / Cross product :

→ Symbol : X (cross)
→ Notation : Relation-1 X Relation-2
R1 X R2
→ Rare in practice, mainly used to express joins.
→ Operation : it will multiply each tuples of relation-1 to each tuples of relation-2.

* Attributes of Resultant Relation
= Attributes of R1 + Attributes of R2.
* Tuples of Resultant Relation
= Tuples of R1 * Tuples of R2.

→ Student :

RollNo	Name	Branch
101	Raj	CE
102	Meet	ME

→ Result :

RollNo	SPI
101	8
103	9

→ Student x Result :

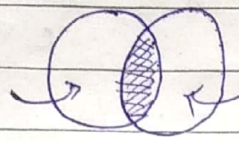
Student.RollNo	Name	Branch	Result.RollNo	SPI
101	Raj	CE	101	8
101	Raj	CE	103	9
102	Meet	ME	101	8
103	Meet	ME	103	9

→ To perform natural join, there must be a common attribute/column.

* Natural join (Inner join) :-

→ Symbol : ⋈

→ It will retrieve consistent data from multiple relations.



→ Steps : 1) Perform cartesian product
2) Delete inconsistent tuples.
3) Remove an attribute from duplicate attributes.

Ex :-

Student

RollNo	Name	Branch
101	Raj	CE
102	Meet	ME

Result

RollNo	SPI
101	8
103	9.

step 1: Perform a cartesian product.

Student x Result

Student.RollNo	Name	Branch	Result.RollNo	SPI
101	Raj	CE	101	8
101	Raj	CE	103	9
102	Meet	ME	101	8
102	Meet	ME	103	9

inconsistent

Step 2: Delete inconsistent tuples.

Student ⋈ Result

Student. RollNo	Name	Branch	Result. RollNo	SPI
101	Raj	CE	101	8.

duplicates

Step 3 : Remove duplicate attribute.

Student ⋈ Result

RollNo	Name	Branch	SPI
101	Raj	CE	8.

Example : Write down relational algebra for the following tables / relations :

1) Student (Rno , Sname , Address , City , Mobile)
2) Department (Did , Dname)
3) Academic (Rno , Did , SPI , CPI , Backlog)
4) Guide (Rno , Pname , Fid)
5) Faculty (Fid , Fname , Subject , Did , Salary)

⇨ List the name of students with their dept. name and SPI of all student belong to 'CE' dept.

Ans :- $\pi_{Sname, Dname, SPI}\left(\sigma_{Dname='CE'}\left(Student \bowtie \left(Department \bowtie Academic\right)\right)\right)$

⇨ Display the name of students with their project name whose guide is A.J. Shah.

Ans :- $\pi_{Sname, Pname}\left(\sigma_{Fname='A.J.\ Shah'}\left(Student \bowtie \left(Guide \bowtie Faculty\right)\right)\right)$

Date
Page

cu> List the name of students with their department name having backlog 0.

Ans :- $\Pi_{Sname, Dname}(\sigma_{Backlog='0'}(Student \bowtie (Department \bowtie Academic)))$

cu> List the name of faculties with their department name and salary having salary more than 25000 and belongs to CE department.

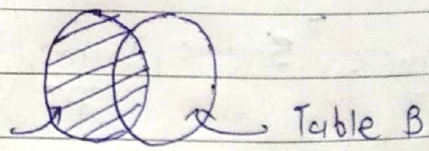
Ans :- $\Pi_{Fname, Dname, Salary}(\sigma_{Salary>25000 \wedge Dname='CE'}(Department \bowtie Faculty))$

cu> List the name of all faculties of 'CE' and 'ME' department whose salary is more than 50000.

Ans :- $\Pi_{Fname}(\sigma_{Dname='CE' \wedge Dname='ME' \wedge Salary>50000}(Faculty \bowtie Department))$

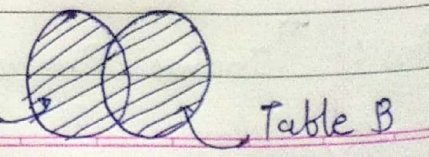
* Outer join :-

1) Left outer join : $\bowtie$



2) Right outer join : $\bowtie$

3) Full outer join : $\bowtie$



* Equi join ($\bowtie_{A=B}$) :-

→ For whatever join type (inner, outer, etc, ...), if we use only the equility operator (=), then we say that the join is an equi-join.

→ It's a theta join where θ is an equality.

→ $R_1 \bowtie_{A=B} R_2 = \sigma_{A=B}(R_1 \times R_2)$

→ MOST common join in practice.

→ Ex: SQL: select * from Students S, People P where sname = pname;

RA : $S \bowtie_{sname = pname} P$

* Theta join ($\bowtie_\theta$) :-

→ This is same as equi join but it allows all other operators like >, <, >=, ---.

→ This join envolves predicate.

→ $R_1 \bowtie_\theta R_2 = \sigma_\theta (R_1 \times R_2)$

→ Natural join = Theta join + projection.

→ Ex: SQL : select * from Students, People where θ;

RA : Students $\bowtie_\theta$ People.

* Set operators :-

1) Union
2) Intersect [Intersection]
3) Minus [Set Difference]

* Conditions to perfor set operations :-

→ Set operators will take two or more queries as input, which must be union-compatible.

1. Both queries should have same number of columns.
2. Corresponding attributes should have the same data type.

* Division operator :- (÷)

→ Condition : Attributes of relation : 2 is proper subset of attributes of relation : 1.

→ The output of the division operator will have

attributes = all attributes of relation : 1 − all attributes of relation : 2

tuples = tuples in relation 1, which are associated with all tuples of relation 2.

A		B1	B2
Sno	Pno	Pno	Pno
S1	P1	P2	P2
S1	P2		P4
S1	P3		
S1	P4		
S2	P1		
S2	P2		
S3	P2		
S4	P2		
S4	P4		
S5	P4		

same

same

→ A ÷ B1 =>

Sno
S1
S2
S3
S4

→ A ÷ B2 =>

Sno
S1
S4

* Rename operator :- (ρ) Rho

** Student :

Rno	Name	CPI
101	Raj	8
102	Meet	9
103	Suresh	7

1) → ρ_{Person} (Student) => Person :

Rno	Name	CPI
101	Raj	8
102	Meet	9
103	Suresh	7

2) → $\rho_{(RollNo, StudentName, CPI)}$ (Student)

=> Student :

RollNo	StudentName	CPI
101	Raj	8
102	Meet	9
103	Suresh	7

3) → $\rho_{Person(RollNo, StudentName)}(\pi_{Rno, Name}(Student))$

=> Person :

RollNo	StudentName
101	Raj
102	Meet
103	Suresh.

* Exercises :-

* Write down relational algebra for the following tables

Employee (person-name , street , city)
Works (person-name , company-name , salary)
Company (company-name , city)
Managers (person-name , manager-name)

1) Find the names of all the employees who work for TCS.

→ RA : $\pi_{person\text{-}name}(\sigma_{company\text{-}name='TCS'}(Employee \bowtie Works))$

2) Find the names and cities of residence of all employees who work for Infosys.

→ RA : $\pi_{person\text{-}name, city}(\sigma_{company\text{-}name = 'Infosys'}(Employee \bowtie Works))$

3) Find the names, street and city of residence of all employees who work for 'ITC' and earn more than $ 10,000 per annum

→ RA: $\Pi_{\text{person-name, street, city}}\left(\sigma_{\text{company-name} = \text{'ITC'} \wedge \text{salary} > 10000}(\text{Employee} \bowtie \text{Works})\right)$

4) Find the names of all employees in this database who live in the same city as the company for which they work.

→ RA: $\Pi_{\text{person-name}}\left(\Pi_{\text{city}}(\text{Employee}) \cap \Pi_{\text{city}}(\text{Company})\right)$

5) Find the names of all employees working in 'TCS' who earn more than 25000 and less than 40000.

→ RA: $\Pi_{\text{person-name}}\left(\sigma_{\text{company-name} = \text{'TCS'} \wedge \text{salary} > 25000 \wedge \text{salary} < 4000}(\text{Employee} \bowtie \text{Works})\right)$

6) Find the name of Employee whose manager is 'Ajay Patel' and salary is more than 50000

→ RA: $\Pi_{\text{person-name}}\left(\sigma_{\text{manager-name} = \text{'Ajay Patel'} \wedge \text{salary} > 50000}(\text{Works} \bowtie \text{Managers})\right)$

7) Display the name of employee with street, city, company-name, salary and manager name staying in 'Rajkot' and working in 'Ahmedabad'.

→ RA: $\Pi_{\text{person-name, street, city, company-name, salary, manager-name,}}\left(\text{Managers} \bowtie \left(\sigma_{\text{city} = \text{'Rajkot'}}(\text{Employee}) \cup \sigma_{\text{city} = \text{'Ahm'}}(\text{Company})\right)\right)$

8) Find maximum, minimum and average salary of all employee.

→ RA : $\mathcal{G}_{max(salary),\ min(salary),\ avg(salary)}(Works)$

9) Find out the total number of employees.

→ RA : $\mathcal{G}_{count(person\text{-}name)}(Employee)$

* Limitation of RA : Can not compute 'transitive closure'.

—— × ——