Relational Algebra

Dr. Seema Gupta Bhol

Relational Algebra

- Relational Algebra is a procedural query language.
- Relational algebra mainly provides a theoretical foundation for relational databases and SQL.

Fundamental Operators

- These are the fundamental operators used in Relational Algebra.
- Selection(σ)
- Projection(π)
- Union(U)
- Set Difference(-)
- Set Intersection(∩)
- Rename(ρ)
- Cartesian Product(X)

Selection(σ)

It is used to select requred tuples of the relations.
e.g. consider relation R

A	B	C
1	2	4
2	2	3
3	2	3
4	3	4

For the above relation, $\boldsymbol{\sigma_{(c>3)}R}$ will select the tuples which have c more than 3.

A	B	C
1	2	4
4	3	4

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

Projection(π)

It is used to project required column data from a relation. Suppose we want columns B and C from Relation R.

$\pi_{B,C}(R)$ will show following columns.

By Default, projection removes duplicate data.

B	C
2	4
2	3
2	3
3	4

Union(U)

- Union operation in relational

algebra is the same as union operation in set theory.

- Consider the following table of Students having different optional subjects in their course.

FRENCH

Student_Name	Roll_Number
Ram	01
Mohan	02
Vivek	13
Geeta	17

GERMAN

Student_Name	Roll_Number
Vivek	13
Geeta	17
Shyam	21
Rohan	25

$$\pi_{Student\_Name}(FRENCH) \cup \pi_{Student\_Name}(GERMAN)$$

Student_Name
Ram
Mohan
Vivek
Geeta
Shyam
Rohan

In the union of two relations ,both relations must have the same set of Attributes.

Set Intersection(∩)

- Set Intersection in relational algebra is the same set intersection operation in set theory.
- **Example:** From the above table of FRENCH and GERMAN, the Set Intersection is used as follows

$$\pi_{Student\_Name}(FRENCH) \cap \pi_{Student\_Name}(GERMAN)$$

Student_Name
Vivek
Geeta

- In the Intersection of two relations ,both relations must have the same set of Attributes.

Set Difference(-)

- Set Difference in relational algebra is the same set difference operation as in set theory.

- list of students studying French not German :

$\pi_{Student\_Name}(FRENCH) - \pi_{Student\_Name}(GERMAN)$

- list of students studying German not French:

$\pi_{Student\_Name}(GERMAN) - \pi_{Student\_Name}(FRENCH)$

Rename(ρ)

- The **rename operator** *ρ* is one of the unary operators in relational algebra
- It is used to rename relations in a DBMS.
- In other words, some relations or attributes may have complex names and can be changed to make writing queries easier.

$\rho_{new\_relation\_name}(new\_attribute\_name\_list)(R)$

e.g.

$\rho_{FinalYrStudents}(\sigma_{\text{year='final'}}\ (Students))$

Cross Product(X)

- Cross-product between two relations.
- Let A and B are two relations
- The cross product between A X B will result in all the attributes of A followed by each attribute of B. Each record of A will pair with every record of B.
- If A has n tuples and B has m tuples then A X B will have n*m tuples.

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

Example

Employee-Schema = { Emp-id, Name }
Project-Schema = { Proj-name }

R = Employee X Project

Emp-Id	Name
101	Sachin
103	Rahul
104	Omkar
106	Sumit
107	Ashish

Proj-name
DBMS 1
DBMS 2

mp-Id	Name	Proj-name
101	Sachin	DBMS 1
101	Sachin	DBMS 2
103	Rahul	DBMS 1
103	Rahul	DBMS 2
104	Omkar	DBMS 1
104	Omkar	DBMS 2
106	Sumit	DBMS 1
106	Sumit	DBMS 2
107	Amit	DBMS 1
107	Amit	DBMS 2

Natural Join($\bowtie$)

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

EMP ⋈ DEPT

Emp table

Name	ID	Dept_Name
A	120	IT
B	125	HR
C	110	Sales
D	111	IT

DEPT

Dept_Name	Manager
Sales	Y
Production	Z
IT	A

Natural join between EMP and DEPT with condition :

EMP.Dept_Name = EPT.Dept_Name

List manager of employee A

$\pi_{manager} (\sigma_{name='A'} (EMP \bowtie DEPT))$

Conditional Join

- Conditional join works similarly to natural join. In natural join, by default condition is equal between common attributes while in conditional join we can specify any condition such as greater than, less than, or not equal.

Join between R and S with condition

R.marks >= S.marks

R

ID	Sex	Marks
1	F	45
2	F	55
3	F	60

S

ID	Sex	Marks
10	M	20
11	M	22
12	M	59

R.ID	R.Sex	R.Marks	S.ID	S.Sex	S.Marks
1	F	45	10	M	20
1	F	45	11	M	22
2	F	55	10	M	20
2	F	55	11	M	22
3	F	60	10	M	20
3	F	60	11	M	22
3	F	60	12	M	59

Example1 :Players

Player Id	Team Id	Country	Age	Runs
1001	101	India	25	10000
1004	101	India	28	20000
1006	101	India	22	15000
1005	101	India	21	12000
1008	101	India	22	15000
1009	103	England	24	6000
1010	104	Australia	35	1300
1011	104	Australia	29	3530
1012	105	Pakistan	28	1421
1014	105	Pakistan	21	3599

Queries in Relation Algebra

- Find all tuples from player relation for which country is India.

$$\sigma_{\text{country = “India”}}(\text{Player})$$

- Select all the tuples for which runs are greater than or equal to 15000.

$$\sigma_{\text{runs>=1500}}(\text{Player})$$

- Select all the players whose runs are greater than or equal to 6000 and age is less than 25

$$\sigma_{\text{runs>=1500 ^ age < 25}}(\text{Player})$$

- List all the countries in Player relation

$$\pi_{\text{country}} (\text{Player})$$

- List all the team ids and countries in Player Relation

$$\pi_{\text{country, team\_id}} (\text{Player})$$

- List id of all the players whose runs are greater than or equal to 6000 and age is less than 25

$$\pi_{\text{country}} (\sigma_{\text{runs>=1500 ^ age < 25}} (\text{Player}))$$

- Create new relation called Indian players

$$\rho \text{ Indian\_players}(\sigma_{\text{country = “India”}}(\text{Player})$$

Example 2

Deposit Relation	
Acc. No.	Cust-name
A 231	Rahul
A 432	Omkar
R 321	Sachin
S 231	Raj
T 239	Sumit

Borrower Relation	
Loan No.	Cust-name
P-3261	Sachin
Q-6934	Raj
S-4321	Ramesh
T-6281	Anil

Example 2

- Find all the customers having an account but not the loan.

$$\pi_{cust\text{-}name}(Depositor) - \pi_{cust\text{-}name}(Borrower)$$

Result:

Cust-name
Rahul
Omkar
Sumit

- Find all the customers having a load but not the account.

$$\pi_{cust\text{-}name}(Borrower) - \pi_{cust\text{-}name}(Depositor)$$

Result:

Cust-name
Ramesh
Anil

Example 2

- Find all the customers having an account but not the loan.

$$\pi_{cust\text{-}name}(Depositor) - \pi_{cust\text{-}name}(Borrower)$$

Result:

Cust-name
Rahul
Omkar
Sumit

- Find all the customers having a loan but not the account.

$$\pi_{cust\text{-}name}(Borrower) - \pi_{cust\text{-}name}(Depositor)$$

Result:

Cust-name
Ramesh
Anil

Example 2

- Find all the customers having account or loan.

$$\pi_{cust\text{-}name}(Depositor) \cup \pi_{cust\text{-}name}(Borrower)$$

- Find all the customers having a both loan and account.

$$\pi_{cust\text{-}name}(Borrower) \cap \pi_{cust\text{-}name}(Depositor)$$

Exercises

Consider the following Schema:

- Sellers(sID, sName, address)
- Products(pID, pName, ptype)
- Orders(sID, pID, price)

Solve the following queries

1.Find the names of all products of the type nuts .

$$\pi_{pname}(\sigma_{\text{type = “nuts”}}(\text{Products}))$$

2. Find price of the products sold by smith

$\pi_{price}\ (\sigma_{\text{sname = “smith”}}\ (\text{Sellers})) \bowtie \text{Orders})$

3. Find address of sellers , selling nuts

$\pi_{address}(((\sigma_{\text{type = “nuts”}}\ (\text{Products})) \bowtie \text{Orders}) \bowtie \text{sellers})\ )$

4. Find all prices for products that are nuts or bolts.

$\pi_{price}(\sigma_{\text{type = “nuts” v type = “bolts”}}(\text{Products})) \bowtie \text{Orders})$

5. Find the sIDs of all sellers who supply a product that is nuts or bolts.

$\pi_{sid}(\sigma_{\text{type = “nuts” v type = “bolts”}}(\text{Products})) \bowtie \text{Orders})$

6. Find the sIDs of all sellers who supply a product that is nuts and bolts.

   Trick question. Each tuple has only one type, and each product has only one tuple (since pID is a key), so no product can be recorded as both nuts and bolts.

7. Find the names of all sellers who supply a product that is nuts or bolts.

$\pi_{sname}(\sigma_{\text{type = “nuts” v type = “bolts”}}(\text{Products})) \bowtie \text{Orders}) \bowtie \text{Sellers})$

8. Find the Sids of sellers who supply some nuts and some bolts .

$\pi_{sid}(\sigma_{\text{type = “nuts”}}(\text{Products})) \bowtie \text{Orders}) \cap \pi_{sid}(\sigma_{\text{type = “bolts”}}(\text{Products})) \bowtie \text{Orders})$

9.Find the Sids of the seller who supply some nuts or bolts .

$\pi_{sid}(\sigma_{\text{type = “nuts”}}(\text{Products})) \bowtie \text{Orders}) \text{ U } \pi_{sid}(\sigma_{\text{type = “bolts”}}(\text{Products})) \bowtie \text{Orders})$

10. Find the sellers who supply nuts only but no bolts.

$\pi_{sid}(\sigma_{\text{type = “nuts”}}(\text{Products})) \bowtie \text{Orders}) - \pi_{sid}(\sigma_{\text{type = “bolts”}}(\text{Products})) \bowtie \text{Orders})$