Relational Algebra

by

Dr. Pratik Roy

Introduction

- Relational algebra consists of a set of operations that take one or two relations as input and produce a new relation as their result.
- Six basic operations
  - select: σ
  - project: Π
  - union: $\cup$
  - set difference: –
  - Cartesian product: x
  - rename: ρ
- These operations enable a user to specify **basic retrieval requests** (or **queries**).
- Result of an operation is a *new relation*, which may have been formed from one or more *input* relations.
  - This property makes the algebra "closed" (all objects in relational algebra are relations).

Select Operation

- **select** operation selects tuples that satisfy a given predicate.
- Notation: $\sigma_p(r)$
- p is called **selection predicate**.
- **Example**

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

instructor relation

Select those tuples of *instructor* relation where instructor is in "Physics" department

Query

$$\sigma_{dept\_name = \text{“Physics”}}(instructor)$$

Result

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
33456	Gold	Physics	87000

Select Operation

- We allow comparisons using

  $=, \neq, >, \geq. <. \leq$

  in the selection predicate.

- We can combine several predicates into a larger predicate by using connectives:

  $\wedge$ (**and**), $\vee$ (**or**), $\neg$ (**not**)

- Example: Find the instructors in Physics with a salary greater $90,000, we write:

  $$\sigma_{dept\_name=\text{“Physics”} \wedge salary > 90,000} (instructor)$$

- select predicate may include comparisons between two attributes.
  - Example: find all departments whose name is same as their building name:
  - $\sigma_{dept\_name=building}$ (*department*)

SELECT Operation Properties

- SELECT operation $\sigma_{<\text{selection condition}>}(R)$ produces a relation S that has the same schema (same attributes) as R.
- SELECT σ is commutative:
  - $\sigma_{<\text{condition1}>}(\sigma_{<\text{condition2}>}(R)) = \sigma_{<\text{condition2}>}(\sigma_{<\text{condition1}>}(R))$
- Because of commutativity property, a cascade (sequence) of SELECT operations may be applied in any order:
  - $\sigma_{<\text{cond1}>}(\sigma_{<\text{cond2}>}(\sigma_{<\text{cond3}>}(R))) = \sigma_{<\text{cond2}>}(\sigma_{<\text{cond3}>}(\sigma_{<\text{cond1}>}(R)))$
- A cascade of SELECT operations may be replaced by a single selection with a conjunction of all the conditions:
  - $\sigma_{<\text{cond1}>}(\sigma_{<\text{cond2}>}(\sigma_{<\text{cond3}>}(R))) = \sigma_{<\text{cond1}> \text{ AND } <\text{cond2}> \text{ AND } <\text{cond3}>}(R))$
- Number of tuples in the result of a SELECT is less than (or equal to) the number of tuples in the input relation R

Project Operation

- A unary operation that returns its argument relation, with certain attributes left out.
- Notation:

$$\Pi_{A_1, A_2, A_3 \ldots A_k} (r)$$

  where A_1, A_2, …, A_k are attribute names and r is a relation name.
- Result is defined as the relation of k columns obtained by erasing columns that are not listed
- Duplicate rows removed from result, since relations are sets.
  - Mathematical sets *do not allow* duplicate elements.
- PROJECT creates a vertical partitioning.
  - List of specified columns (attributes) is kept in each tuple.
  - Other attributes in each tuple are discarded.

Project Operation

- **Example**: eliminate the *dept_name* attribute of *instructor*
- Query*:*

$$\Pi_{ID,\ name,\ salary}\ (instructor)$$

- Result:

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

ID	*name*	*salary*
10101	Srinivasan	65000
12121	Wu	90000
15151	Mozart	40000
22222	Einstein	95000
32343	El Said	60000
33456	Gold	87000
45565	Katz	75000
58583	Califieri	62000
76543	Singh	80000
76766	Crick	72000
83821	Brandt	92000
98345	Kim	80000

PROJECT Operation Properties

- Number of tuples in the result of projection $\pi_{<list>}(R)$ is always less or equal to the number of tuples in R
  - If the list of attributes includes a *key* of R, then the number of tuples in the result of PROJECT is *equal* to the number of tuples in R
- PROJECT is *not* commutative
  - $\pi_{<list1>}(\pi_{<list2>}(R)) = \pi_{<list1>}(R)$ as long as <list2> contains the attributes in <list1>

Composition of Relational Operations

- Result of a relational-algebra operation is relation and therefore relational-algebra operations can be composed together into a **relational-algebra expression**.
- Consider the query - Find the names of all instructors in Physics department.

$$\Pi_{name}(\sigma_{dept\_name = \text{“Physics”}} (instructor))$$

- Instead of giving the name of a relation as argument of the projection operation, we give an expression that evaluates to a relation.

Cartesian-Product Operation

- Cartesian-product operation (denoted by X) allows us to combine information from any two relations.
- On applying cartesian product on two relations that is on two sets of tuples, it will take every tuple one by one from the left set(relation) and will pair it up with all the tuples in the right set(relation).
- So, CROSS PRODUCT of two relation A(R1, R2, R3, …, Rp) with degree p, and B(S1, S2, S3, …, Sn) with degree n, is a relation C(R1, R2, R3, …, Rp, S1, S2, S3, …, Sn) with degree p + n attributes.

Cartesian-Product Operation

- Consider two relations STUDENT(SNO, FNAME, LNAME) and DETAIL(ROLLNO, AGE) below:

SNO	FNAME	LNAME
1	ABC	XYZ
2	PQR	DEF

ROLLNO	AGE
5	18
9	21

- On applying CROSS PRODUCT on STUDENT and DETAIL:

STUDENT × DETAIL

SNO	FNAME	LNAME	ROLLNO	AGE
1	ABC	XYZ	5	18
1	ABC	XYZ	9	21
2	PQR	DEF	5	18
2	PQR	DEF	9	21

Cartesian-Product Operation

- Cardinality (number of tuples) of resulting relation from a Cross Product operation is equal to number of attributes(say m) in first relation multiplied by number of attributes in second relation(say n).

```
Cardinality = m*n
```

- When same attribute appear in both relations we distinguish between these attribute by attaching to the attribute the name of the relation from which the attribute originally came.

Join Operation

- Join operation allows us to combine a selection and a Cartesian product into a single operation.
- Consider relations r(R) and s(S), and let θ be a predicate on attributes in the schema R ∪ S. The join operation $r \bowtie_\theta s$ is defined as follows:

$$r \bowtie_\theta s = \sigma_\theta(r \times s)$$

- The general form of JOIN operation on two relations R(A1, A2, ..., An) and S(B1, B2, ..., Bm) is

$$R \bowtie_{\text{<join condition>}} S$$

- **Conditional Join** is used when we want to join two or more relation based on some conditions. It is also called **theta join**. Theta join combines tuples from different relations provided they satisfy theta condition.
- Theta join can use all kinds of comparison operators {=, <, ≤, >, ≥, ≠}.

Theta Join Operation

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

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

Select students whose ROLL_NO is greater than EMP_NO of employees

$$STUDENT \bowtie_{STUDENT.ROLL\_NO > EMPLOYEE.EMP\_NO} EMPLOYEE$$

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

$$\sigma_{(STUDENT.ROLL\_NO > EMPLOYEE.EMP\_NO)}(STUDENT \times EMPLOYEE)$$

Theta Join Operation

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 or theta 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

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

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

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

Equijoin

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.
- Natural join does not use any comparison operator except =.
- We can perform a Natural Join only if there is at least one common attribute that exists between two relations. Common attributes must have the same name and domain.
- For NATURAL JOIN operation between two tables R and S, only tuples from R that have matching tuples in S, and vice versa, appear in result.

Natural Join

❑Example

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

STUDENT_SPORTS

ROLL_NO	SPORTS
1	Badminton
2	Cricket
2	Badminton
4	Badminton

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

$$STUDENT \bowtie STUDENT\_SPORTS$$

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

$$\Pi_{(STUDENT.ROLL\_NO,\ STUDENT.NAME,\ STUDENT.ADDRESS,\ STUDENT.PHONE,\ STUDENT.AGE\ STUDENT\_SPORTS.SPORTS)}(\sigma_{(STUDENT.ROLL\_NO=STUDENT\_SPORTS.ROLL\_NO)}(STUDENT \times STUDENT\_SPORTS))$$

Natural Join

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

Types of JOIN

- Inner Joins:
  - Theta join
  - EQUI join
  - Natural join
- Outer joins:
  - Left Outer Join
  - Right Outer Join
  - Full Outer Join

- **Inner Join**
  - An inner join includes only those tuples with matching attributes and the rest are discarded in the resulting relation.

Outer Joins

- 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.
- The outer join operation is an extension of the join operation. It is used to deal with missing information.
- In outer join, along with tuples that satisfy the matching criteria, we also include some or all tuples that do not match the criteria.
- We need to use outer joins to include all the tuples from the participating relations in the resulting relation.

Left outer join

- Operation allows keeping all tuple in the left relation.
- If there is no matching tuple is found in right relation, then attributes of right relation in join result are filled with null values.

- All tuples from the Left relation A are included in the resulting relation.
- If there are tuples in A without any matching tuple in the Right relation B, then B-attributes of the resulting relation are made NULL.
- Notation: A ⟕ B

Left outer join

❑Example

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

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

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

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

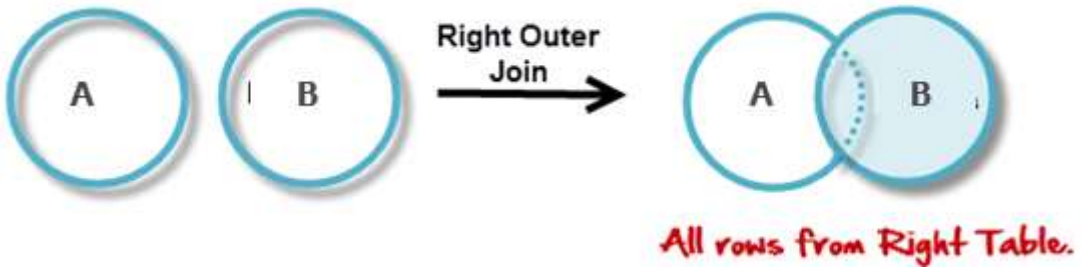
Left outer join

❑ **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

- Operation allows keeping all tuple in the right relation.
- If there is no matching tuple is found in the left relation, then attributes of left relation in join result are filled with null values.



- If we are applying right outer join on two relations A and B, it gives all tuples of B in the result set.
- If there are tuples in B without any matching tuple in A, then A-attributes of resulting relation are made NULL.
- Notation: A⟖B

Right Outer Join

❑Example

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

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

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

$$STUDENT \bowtie_{STUDENT.ROLL\_NO > EMPLOYEE.EMP\_NO} EMPLOYEE$$

Right Outer Join

❑ **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

- All tuples from both relations are included in the result, irrespective of the matching condition.
- When applying full outer join on two relations R and S, it 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.
- Notation: R ⟗ S

Full Outer Join

❑Example

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

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

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

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

Full Outer Join

❑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

Rename Operation

- RENAME operator is denoted by ρ (rho)
- In some cases, we may want to *rename* the attributes of a relation or the relation name or both
  - Useful when a query requires multiple operations
- General RENAME operation ρ can be expressed by any of the following forms:
  - $\rho_{S\ (B1,\ B2,\ \ldots,\ Bn\ )}(R)$ changes both:
    - the relation name to S, *and*
    - the column (attribute) names to B1, B1, …..Bn
  - $\rho_S(R)$ changes:
    - the *relation name* only to S
  - $\rho_{(B1,\ B2,\ \ldots,\ Bn\ )}(R)$ changes:
    - the *column (attribute) names* only to B1, B1, …..Bn
- If we write:
  - RESULT $\leftarrow \pi_{\text{FNAME, LNAME, SALARY}}(R)$
    - RESULT will have the *same attribute names* as R.

UNION

- Binary operation, denoted by ∪
- Result of R ∪ S, is a relation that includes all tuples that are either in R or in S or in both R and S.
- Duplicate tuples are eliminated.
- Two operand relations R and S must be “type compatible” (or UNION compatible)
  - R and S must have same number of attributes.
  - Each pair of corresponding attributes must be type compatible (have same or compatible domains).
- R1(A1, A2, ..., An) and R2(B1, B2, ..., Bn) are type compatible if:
  - they have the same number of attributes, and
  - domains of corresponding attributes are type compatible (i.e. dom(Ai)=dom(Bi) for i=1, 2, ..., n).
- Resulting relation for R1∪R2 has same attribute names as *first* operand relation R1 (by convention).

UNION

❑Example

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

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

Find person who are either student or employee or bot

STUDENT U EMPLOYEE

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
5	NARESH	HISAR	9782918192	22
6	SWETA	RANCHI	9852617621	21

INTERSECTION

- Set-intersection operation allows us to find tuples that are in both the input relations.
- INTERSECTION is denoted by ∩
- Result of the operation R ∩ S, is a relation that includes all tuples that are in both R and S.
  - Attribute names in the result will be same as the attribute names in R.
- Two operand relations R and S must be “type compatible”.

INTERSECTION

- **Example**

DEPOSITOR

CUSTOMER_NAME	ACCOUNT_NO
Johnson	A-101
Smith	A-121
Mayes	A-321
Turner	A-176
Johnson	A-273
Jones	A-472
Lindsay	A-284

BORROWER

CUSTOMER_NAME	LOAN_NO
Jones	L-17
Smith	L-23
Hayes	L-15
Jackson	L-14
Curry	L-93
Smith	L-11

Find the customers who are both depositor and borrower:

∏ CUSTOMER_NAME (BORROWER) ∩ ∏ CUSTOMER_NAME (DEPOSITOR)

CUSTOMER_NAME
Smith
Jones

SET DIFFERENCE

- The set-difference operation allows us to find tuples that are in one relation but are not in another.
- SET DIFFERENCE (also called MINUS or EXCEPT) is denoted by –
- Result of R – S, is a relation that includes all tuples that are in R but not in S.
  - Attribute names in the result will be the same as the attribute names in R
- Two operand relations R and S must be “type compatible”.

SET DIFFERENCE

❑ **Example**

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

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

Find person who are student but not employee:

STUDENT - EMPLOYEE

ROLL_NO	NAME	ADDRESS	PHONE	AGE
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20