Chapter-3 (Relational Algebra and SQL)

- The relational algebra
- Fundamental operation in relational algebra
- Unary relational operation : Select
  Project
  Rename
- Union , set intersection , set difference, cartesia product, Assignment , Division operation.
- Natural join
- Aggregation
- Set comparison

The relational algebra: The relational algebra is a procedural query language. It consists of a set of operations that takes one or two relations as input and produce a new relation as their result.

Fundamental relation operation in relational algebra:

Select (σ) → unary operation

Project (π) → Unary operation

Union ($\cup$) → Binary

Set difference ($-$) → Binary

Cartesian product ($\times$) → Binary

Rename (ρ) → unary operation.

Unary operation – which operate on one relation.

Binary operation – which operate on pairs.

The additional operation:

Set intersection ($\cap$) → Binary operation

Assignment operation ($=$) → Binary operation

Natural join ($\bowtie$) → Binary

Division operation ($\div$) → Binary

Outer join
- left outer join (⟕)
- Right outer join (⟖)
- Full order join (⟗)

Unary Relational operation:
- SELECT (symbol: σ (sigma))
- PROJECT (symbol: π (pi))
- RENAME (symbol: ρ (rho))

The select operation: The select operation selects tuples that satisfy a given predicate. Donated by sigma (σ)

syntax: $\sigma_p(r)$ p = predicate / condition, r = relation

: $\sigma_{condition}(relation)$ OR $\sigma_{selection\_criteria}(Input)$ → relation instance

Comparison operator ($=, \neq, <, \leq, >, \geq$)

connections And ($\wedge$), OR ($\vee$), Not ($\neg$)

Ex: ① To select those tuples of the instructor relation where the instructor is in the 'physics' department.

$\sigma_{dept\_name = 'physics'}(instructor)$

② To find all instructors with salary greater than $90,000

$\sigma_{salary > 90000}(instructor)$

③ select the instructors in physics with a salary greater than $90,000 (and ($\wedge$), or ($\vee$) and not ($\neg$))

$\sigma_{dept\_name = "physics" \wedge salary > 90000}(instructor)$

Table

instructor (id, name, dept_name, salary)
department (dept_name, building, budget)
section (course_id, sec_id, semester, year, building, room_number, time_slot_id)
teaches (ID, course_id, sec_id, semester, year)

The project operation: The project operation is a unary operation that returns its argument relation, with certain attributes left out

* Projection is denoted by uppercase Greek letter pi (Π)
* In the result, the duplicate rows are eliminated
* Syntax: $\Pi_{attributes, attributes\ 2, \ldots}(Relation)$

OR $\Pi_{fields\ (Input)}$ → relation → columns

Ex: ① Show the ID, name, salary

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

② Show the all instructor name who are take the 'physics'

$\Pi_{name}(\sigma_{dept\_name = "physics"}(instructor))$

③ Show the instructor name, salary, tax(10%)

$\Pi_{name, salary, (salary * 10/100)\ as\ tax}(instructor)$

④ To list each employee's first and last name and salary,

$\Pi_{LNAME, FNAME, SALARY}(EMPLOYEE)$

Rename operator:

It is useful to be able to give the relational algebra expressions a name.

Denoted by greek operator rho (ρ),

Syntax: $\rho_x(E)$

$$\rho_x(E) = \rho_{x(A_1, A_2, \ldots A_n)}(E)$$

x ↓ relation name

E ↳ Relational algebra expression

⇒ The column attributes names to $A_1, A_2, \ldots A_n$

Example: ⇒ $\rho_{LastName, SocSocNo}(Employee)$

Output schema: Answer (LastName, SocSocNo)

[changes the schema not the instance]

The union operation: It is a set of all objects that are member of A or B or both

⇒ Duplicated row are eliminated, It is a binary operation.

⇒ denoted by ∪

⇒ Syntax : $\Pi_{column}(Relation-1) \cup \Pi_{column}(Relation-2)$

For, a union operation r ∪ s to be valid, we require that two conditions hold:

① The relations r and s must be of the same arity. That is they must have the same number of attributes.

② The domains of the ith attribute of r and the ith attribute of s must be the same.

Ex: ① Consider a query to find the set of all courses taught in the Fall 2009 semester, the spring 2010 semester. or both,

⇒ To find the set of all courses taught in the Fall 2009 semester.

$\Pi_{course\_id}(\sigma_{semester = "Fall" \wedge year = 2009}(section))$

⇒ To find the set of all courses taught in the spring 2010 semester,

$\Pi_{course\_id}(\sigma_{semester = "spring" \wedge year = 2010}(section))$

Output:

$\Pi_{course\_id}(\sigma_{semester = "Fall" \wedge year = 2009}(section)) \cup \Pi_{course\_id}(\sigma_{semester = "spring" \wedge year = 2010}(section))$

Set-Intersection operation: It is a set of all objects that are a member of both A and B.

→ denoted by ∩

→ Syntax: $\Pi_{column}(Relation\_1) \cap \Pi_{column}(Relation\_2)$

Ex: To find the set of all courses taught in both the Fall 2009 and the spring 2010 semesters.

Output: $\Pi_{course\_id}(\sigma_{semester = "Fall" \wedge year = 2009}(section)) \cap \Pi_{course\_id}(\sigma_{semester = "Spring" \wedge year\ 2010}(section))$

Set-Difference operation: To find tuples that are in one relation but are not in another

→ Denoted by (−)

Ex: Find all the courses taught in the Fall 2009 semester but not in Spring 2010 semester.

Output: $\Pi_{course\_id}(\sigma_{semester = "FALL" \wedge year = 2009}(section)) - \Pi_{course\_id}(\sigma_{semester = "SPRING" \wedge year = 2010}(section))$

Cartesian-product operation:

⇒ Denoted by a cross (X)

⇒ allows us to combine information from any two relations.

⇒ Cartesian product of relations R1 and R2 as R1 X R2

Ex: Find the teacher name, and course_id who taken course in CSE department.

output: $\Pi_{name, course\_id}(\sigma_{instructor.id = teaches.id}(\sigma_{dept\_name = 'CSE'}(instructor \times teaches)))$

Assignment operation: A relational algebra expression by assigning parts of it to temporary relation variables.

⇒ The assignment operation works like assignment in a programming language

⇒ Denoted by ← or =

⇒ To illustrate this operation, consider the definition of natural join operation.

we could write $R * S$ as

temp 1 ← R × S

temp 2 ← $\sigma_{A_1 = S.A_1 \wedge R.A_2 = S.A_2 \wedge \dots \wedge R.A_n = S.A_n}$ (temp 1)

result = π_{RUS} (temp 2)

Division operation: The division operation is used for queries which involve the 'all'

⇒ Denoted by ÷ sign

⇒ Represented by R_1 / R_2 or $R_1 \div R_2$ where R_1 and R_2 are relations.

Ex: Retrieve the name of the subject that is taught in all courses.

R_1

Name	course
System	Btech
Database	Mtech
Database	Btech
Algebra	Btech

R_2

course
Btech
Mtech

output: $R_1 \div R_2$

Name
database

⇒ Can write $R \div S$ as

temp 1 ← $\pi_{R-S}(h)$

temp 2 ← π_{R-S} (Temp1 x S) − $\pi_{R-S,S}(h)$

result = temp1 − temp2

Natural join ($\bowtie$): Natural join can be performed if there is at least one common attribute that exists between two relation.

⇒ The attributes must have the same name and domain

⇒ denoted by the join symbol $\bowtie$.

⇒ Natural join is the inner join.

Syntax: $R \bowtie R$

natural join = cross product + condition

Ex: R = A B

x y

(*) $\pi \bowtie S = \pi_{RUS}(\sigma_{\pi.A_1 = S.A_1 \wedge \pi.A_2 = S.A_2 \wedge \ldots \wedge \pi.A_n = S.A_n}(\pi \times S))$

Ex: Instructor table

ID	Name	Dept_name	Salary (k)
10101	X	CSE	56
10102	Y	EEE	60
10103	Z	CSE	67

Teaches table

ID	course~~nam~~_id	sec_id	semester
10101	CSE-2423	A	4
10102	~~EEE~~-1221	B	1

~~for~~ instructor X teaches is

~~for~~	id	name	dept_name	Salary(k)	course id	sec id	semester
~~for~~	10101	X	CSE	56	CSE_2423	A	4
	10102	Y	EE	60	EEE_1221	B	1

Ex: Find the names of all instructors together with the course_id of all courses they taught".

$\Pi_{name, course\_id}$ (instructor $\bowtie$ teaches)

Left Outer Join(R ⟕ S):

Instructor ⟕ Teaches

Id	Name	Dept_name	Salary(k)	course_id	sec_id	semester	year
10101	Salam	CSE	56	CSE-2423	A	4	2022
10102	Rafiq	EEE	65	EEE-1221	B	1	2022
10103	Jabbor	CSE	67	- -	- -	- -	- -

Right Outer Join(R ⟖ S):

Instructor ⟖ Teaches

Id	Name	Dept_name	Salary (k)	course_id	Id	course_id	sec_id	semester	year
10101	Salam	CSE	56	CSE-2423	10101	CSE-2423	A	4	2022
10102	Rafiq	EEE	65	EEE-1221	10102	EEE-1221	B	1	2022

Full Outer Join: (R ⟗ S)

Instructor ⟗ Teaches

Id	Name	Dept_name	Salary (k)	course_id	Id	course_id	sec_id	semester	year
10101	Salam	CSE	56	CSE-2423	10101	CSE-2423	A	4	2022
10102	Rafiq	EEE	65	EEE-1221	10102	EEE-1221	B	1	2022
10103	Jabbor	CSE	67	- -	- -	- -	- -	- -	- -

Theta join: ⇒ A join involves a predicate

⇒ $R1 \mid x \mid_{\theta} R_2 = \sigma_{\theta} (R1 \times R2)$ [θ can be any condition]

⇒

Eq-join: Equijoin is a special case of conditional join where only equality condition holds between a pair of attributes

⇒ A theta join where θ is an equality

⇒ $R1 \mid x \mid_{A=B} R2 = \sigma_{A=B} (R1 \times R2)$

⇒ Ex: $Employee \mid x \mid_{SSN=SSN} Dependents$

Semi join: $R \mid x \mid S = \pi_{A1, \ldots An} (R \mid x \mid S)$

where $A_1, \ldots A_n$ are the attributes in R

Ex: Employee |x Dependents

⇒ $R \triangleright_F S$

⇒ Can rewrite semijoin using projection and join:

$$-R \triangleright_F S = \pi_A (R \bowtie_F S)$$

Aggregation: Aggregate function take a collection of values and return a single value as a result

⇒ Aggregate operation $\mathcal{G}$ (denoted by $\mathcal{G}$) which permits the use of aggregate functions such as min or average, on sets of values.

⇒ Some aggregate function are sum, avg, max, min, count.

avg: average value

min: minimum value

max: maximum value

sum: sum of values

count: number of values

Syntax: <grouping-attributes> $\mathcal{G}$ <function_list> (R)

Ex: Find the number of the salary from instruction table.

$\mathcal{G}$ sum(salary) (instructor)

$G_1, G_2 \ldots G_n \mathcal{G} F_1(A_1), F_2(A_2 \ldots F_n(A_n)$ (E)

E is any relational algebra expression

⇒ $G_1, G_2 \ldots G_n$ is a list of attributes on which to group (can be empty)

⇒ Each F_i is an aggregate function

⇒ Each A_i is an attribute name

Insert: Example- Add an instructor Borkhot is an instructor table salary in 54k and he also take course where course_code = CSE-1224, sec_id = 2, semester = 2, year = 2023

⇒ Instructor_id ← instructor ∪ {("Borkhot", 54)}

Teaches ← {(Instructor_id, "CSE-1224", 2, 2, 2023)}

Delete:

Example: Delete all the course which are teach on 2019

⇒ Teaches ← Teaches − $\sigma_{year = 2019}$ (Teacher)

update:

Example: salary of all instructor over 55K increase 7% whereas all other receive 5%

⇒ Instructor ← $\pi_{id, name, dept\_name, salary * 1.07}$ ($\sigma_{salary >= 55}$ (Instructor))

∪ $\pi_{id, name, dept\_name, salary * 1.07}$ ($\sigma_{salary < 55}$ (Instructor))

View:

Create a view name as <query expression>

Example: Find the teacher name and course id who are taken course in CSE department. Create view all CSE_dept_instructor.

⇒ $\pi_{name, course\_id}$ ($\sigma_{instructor.id = teacher.id}$ ($\sigma_{dept\_name = "CSE"}$ (instructor × teacher)))

Insert into view:

Create view dept_short details as

$\Pi_{id, name, dept\_name}$ (Instructor)

Insert into dept_short details view:

dept_short details $\leftarrow$ dept_short_details $\cup$ {('34', 'x')}

Set comparison: Set comparison sql

→ <some, <= some, >some, >= some, =some and <>some

→ <all, <= all, >= all, = all and <>all

Example: Find the name of all branches that have assets greater than those of at least one branch of located at 'chittagong'

```
Select
    branch_name
from branch
    branch
where
    assets> some (select assets
                  from branch
                  where branch_city = 'chittagong'
```