COMPUTER SCIENCE

Database Management System

Query Language

Lecture_2

Vijay Agarwal sir


TOPICS TO BE COVERED
01
Basic Operators
02
Derived Operators

BASIC operator & Derived operator.

Selection [σ] ⟶ Tuples / Row

Projection [Π] ⟶ Attribute (or) Attribute List

Union

Set Difference

Intersection

① Arity of R & S must be same

② Range of Attribute must be similar.

R ⇒ M Tuple S ⇒ N Tuple.

UNION

Intersection

R - S

S - R

Clang JAVA

Assume Relation R & Relation S consist M & N Tuple Respectively

(1) Range of tuples in $R \cup S$ = Minimum $\max(M, N)$ to Maximum $M + N$

(2) Range of tuples in $R \cap S$ = ϕ to $\min(M, N)$

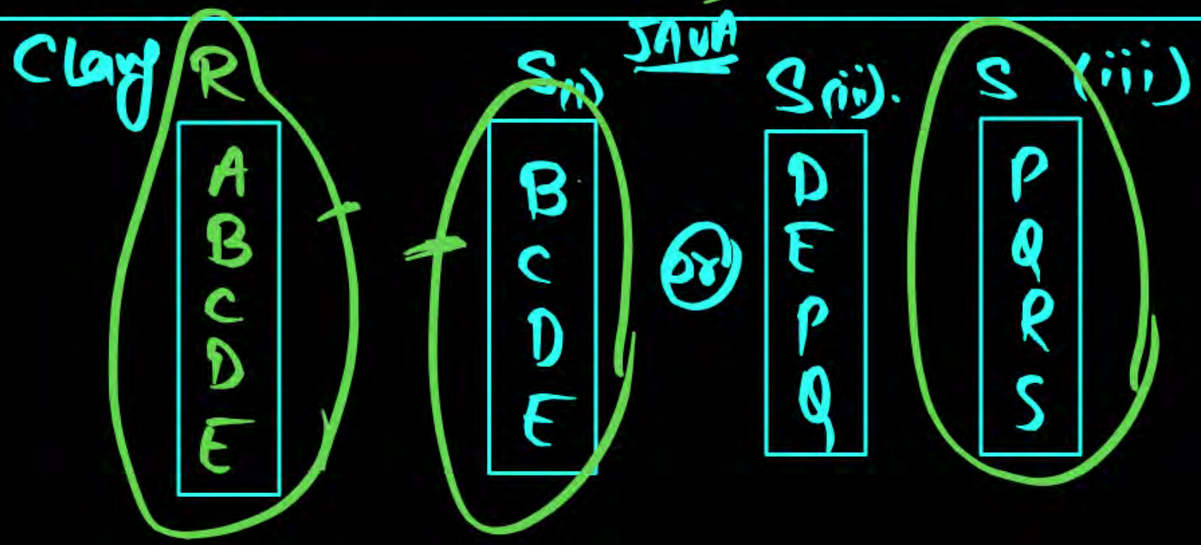
(3) Range of tuples in $R - S$ = ϕ to M

(4) Range of tuples in $S - R$ = ϕ to N

R(A B C D E)

S(A B C D E F G)

S(P Q R S T U V)



Union Operation

- Notation: $r \cup s$
- Defined as :

$$r \cup s = \{t | t \in r \text{ or } t \in s\}$$

- For $r \cup s$ to be valid.
  1. r, s must have the same arity (same number of attributes)
  2. The attribute domains must be compatible (example: 2nd column or r deals with the same type of values as does the 2nd column of s)

Example:

To find all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or in both.

$$\pi_{\text{course\_id}}(\sigma_{\text{semester = "Fall"} \wedge \text{ year = 2009}}(\text{section})) \cup$$
$$\pi_{\text{course\_id}}(\sigma_{\text{semester = "Spring"} \wedge \text{ year = 2010}}(\text{section}))$$

Set Difference Operation

- Notation: r – s
- Defined as :

$$r - s = \{t \mid t \in r \text{ and } t \notin s\}$$

- Set differences must be taken between compatible relations.
  - r and s must have the same arity
  - attribute domains of r and s must be compatible

Example:

Example: to find all courses taught in the Fall 2009 semester, but not in the Spring 2010 semester

$$\pi_{\text{course\_id}}(\sigma_{\text{semester} = \text{“Fall”} \wedge \text{year} = 2009}(\text{section})) -$$

$$\pi_{\text{course\_id}}(\sigma_{\text{semester} = \text{“Spring”} \wedge \text{year} = 2010}(\text{section}))$$

Example:

Sailors (S_1)

Sid	Sname	Rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

Sailors (S_2)

Sid	Sname	Rating	age
28	Yuppy	9	35.0
31	Lubber	8	55.5
44	Guppy	5	35.0
58	rusty	10	35.0

$S_1 - S_2$

22 dustin 7 45.0

$S_2 - S_1$

28 yuppy 9 35

44 Guppy 5 35

(1) Union

Sid	Sname	Rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0
44	guppy	5	35.0
28	yuppy	9	35.0

$S1 \cup S2$

(2) Set Difference

Sid	Sname	Rating	age
22	dustin	7	45.0

S1 – S2

(3) Intersection

Sid	Sname	Rating	age
31	lubber	8	55.5
58	rusty	10	35.0

$S1 \cap S2$

Selection [σ]

Projection [π]

UNION [$\cup$]

Set Difference ($-$)

Intersection [$\cap$].

Relational Algebra

Basic operators

- π : Projection operator
- σ : Selection operator
- $\times$: Cross-product operator
- $\cup$: Union
- $-$: Set difference
- ϱ : Rename operator

Relational Algebra

Derived operators

∩ : Intersection {using "_"}

⋈ : Join {using X, σ}

/ or ÷ : Division {using π, x, −}

CROSS Product

R X S

R	S
n_1 Tuple	n_2 Tuples
c_1 Attribute	c_2 Attribute

$$R \times S = \begin{matrix} n_1 \times n_2 \text{ Tuples} \\ c_1 + c_2 \text{ Attribute} \end{matrix}$$

R	S
3 Tuple	3 Tuples
2 Attribute	5 Attribute

$R \times S = 3 \times 3 = 9$ Tuples

$2 + 5 = 7$ Attributes.

Relational Algebra

Basic operators

II.Cross product (×): / Cartesian Product

- R × S : It result all attributes of R followed by all attributes of S, and each record of R paired with every record of S.
- Degree (R × S) = Degree (R) + Degree (S)
- |(R × S)| = |R| × |S|

R X S =

$C_1 + C_2$ Attribute

$n_1 * n_2$ Tuples.

NOTE:

- Relation R with n tuples and
- Relation S with 0 tuples then
- number of tuples in R × S = 0 tuples

Cross – Product

Reserves (R_1)

Sid	bid	day
22	101	10/10/96
58	103	11/12/96

Sailors (S_1)

Sid	Sname	Rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1.Sid

R_1 : 2 Tuple
3 Attribute

S_1 : 3 Tuple
4 Attribute

$R_1 \times S_1 = 2 \times 3 = 6$ Tuple

$3 + 4 = 7$ Attribute

R_1 S_1

2×3 = 6 Tuples
3+4 = 7 Attribute

$R_1 \times S_1$ =

R_1.Sid	R_1.bid	R_1.day	S_1.Sid	S_1.Sname	S.Rating	S_1.age
22	101	10/10/96	22	dustin	7	45.0
22	101	10/10/96	31	Lubber	8	55.5
22	101	10/10/96	58	Rusty	10	35.0
58	103	11/12/96	22	dustin	7	45.0
58	103	11/12/96	31	Lubber	8	55.5
58	103	11/12/96	58	Rusty	10	35.0

$R_1.Sid > S_1.sid$

S_1 R_1

$S_1 \times R_1 =$

(Sid)	Sname	Rating	age	(Sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

Q1 why CROSS product Required?

Sol 1 Wants to Perform JOIN operation Cross Product Required.

Q.2 If Don't Want to Perform JOIN Operation then Why CROSS Product Required ?
Why Need of CROSS Product ?

Sol 2 If we Require Something like $R_1.Sid > S_1.Sid$.

In Query language its Not possible to Fetch Directly.
Because at a time its Fetch a Tuple either from Relation R (or) FROM Relation S. So Not Possible to Compare.

Thats Why we are Maintaining a One Table R1XS1

To Convert them into One Table.

So Now $R_1.Sid > S_1.Sid$ Comparsion Make Possible.

So CROSS Product is Required.

JOIN ($\bowtie$) R $\bowtie$ S

It's Derived operator Performed in 3 steps.

	R	S
	n_1 Tuple	n_2 Tuple
	c_1 Attribute	c_2 Attribute

Step1: CROSS PRODUCT of R & S (RXS)

RXS =
$n_1 \times n_2$ Tuple
$c_1 + c_2$ Attribute

Step2: Select the Tuples which satisfy equality Condition on ALL Common Attribute of R & S (FROM RXS)

Step3: Projection of Distinct Attribute.

Natural Join ($R \bowtie S$)

Step 3 · Step 2 · Step 1

$$R \bowtie S = \pi_{\text{Distinct Attribute}}\left[\sigma_{\text{equality Condition on All Common Attributes}}(R \times S)\right]$$

(eg) $R(ABCDE)$ $S(DEFG)$

$$R \bowtie S = \pi_{ABCDEFG}\left[\sigma_{R.D=S.D \wedge R.E=S.E}(R \times S)\right]$$

Join Operations

(1) Conditional Join ($\bowtie_c$) ⇒ Any Condition.

(2) Equi join [Equality Condition]

(3) Natural join

(4) Left outer join

(5) Right outer join

(6) Full outer join

Conditional Join.

$$R \bowtie_c S = \pi_{\text{ALL Attribute}}\left[\sigma_{\text{Given Condition}}\left(R \times S\right)\right]$$

$$R \bowtie_{R_1.Sid > S_1.Sid} S = \pi_{\text{Sid Sname Rating Age Sid bid day}}\left[\sigma_{R_1.Sid > S_1.Sid}\left(R \times S\right)\right]$$

Conditional Join

$$R \bowtie_c S = \sigma_C (R \times S)$$

$R.sid > S.sid$

S_1 Sid

R_1 Sid

$R_1 \times S_1$

(Sid)	Sname	Rating	age	(Sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

Conditional Join

$S1 \bowtie_{S1.sid < R1.sid} R1$

R1.sid > S1.sid

output

(Sid)	Sname	Rating	age	(Sid)	bid	day
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	58	103	11/12/96

$S1 \bowtie_{S1.sid < R1.sid} R1$

- Result schema same as that of cross-product.
- Fewer tuples than cross – product, might be able to compute more efficiently.
- Sometimes called a theta –join.

Equi – Join

A special case of condition join where the correction c contains only equalities.

R_1 $R_1.Sid = S_1.Sid$ S_1

$R_1 \times S_1$

(Sid)	Sname	Rating	age	(Sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

Equi – Join

$S1 \bowtie_{sid} R1$

$S.sid = R1.sid$

Sid	Sname	Rating	age	bid	day
22	dustin	7	45.0	101	10/10/96
58	rusty	10	35.0	103	11/12/96

- Result schema similar to cross – product, but only one copy of fields for which equality is specified.
- Natural join: Equijoin on all common fields.

R(A B C D E) S(C D E F)

R Equi Join S © ≠ R.C = S.C (OR) R.A = S.F

Natural Join

R ⋈ S

R.C = S.C ∧
R.D = S.D ∧
R.E = S.E

Join (⋈)

I. Natural join (⋈)

step 2 ↑ step 1 ↑

$R \bowtie S \equiv \pi_{\text{distinct attributes}}(\sigma_{\text{equality between common attributes of R and S}}(R \times S))$

Example:

- T_1 (ABC) and T_2 (BCDE)

$$\therefore \quad T_1 \bowtie T_2 = \pi_{ABCDE}\left(\sigma_{\substack{T_1 \cdot B = T_2 \cdot B (T_1 \times T_2) \\ \cap T_1 \cdot C = T_2 \cdot C}}\right)$$

- T_1 (AB) and T_2 (CD)

$$\therefore \quad T_1 \bowtie T_2 \equiv T_1 \times T_2 = \pi_{ABCD}(T_1 \times T_2)$$

If No Common is Common then

$R \bowtie S \equiv R \times S$

NOTE:

Natural join equal to cross-product if join condition is empty.

⇓

No Common Attribute

Join (⋈)

II. Conditional Join ($\bowtie_c$)

- $R \bowtie_c S \equiv \sigma_c (R \times S)$

Join (⋈)

III. Outer Joins:

(a) LEFT OUTER JOIN

$R ⟕ S$: It produces

$(R ⋈ S) \cup$ {Records of R those are failed join condition with remaining attributes null}

(b) RIGHT OUTER JOIN (⟖)

$R ⟖ S$: It produces

$(R ⋈ S) \cup$ {Records of S those are failed join condition with remaining attributes null}

(C) FULL OUTER JOIN (⟗)

$R ⟗ S = (R ⟕ S) \cup (R ⟖ S)$

Natural Join ⋈

$R \bowtie S = \Pi_{ABCD}\left[\sigma_{R.B=S.B \wedge R.C=S.C}(R \times S)\right]$

R

A	B	C
1	2	4
3	2	6

S

B	C	D
2	4	8
2	7	4

$R \times S =$

R.A	R.B	R.C	S.B	S.C	S.D
1	2	4	2	4	8
1	2	4	2	7	4
3	2	6	2	4	8
3	2	6	2	7	4

2×2 = 4 Tuple

3+3 = 6 Attributes

$R \bowtie S =$

A	B	C	D
1	2	4	8

$$R \bowtie S = \pi_{ABCD} \left\{ \sigma_{\substack{R.B = S.B \\ R.C = S.C}} \wedge (R \times S) \right\}$$

$R \bowtie S =$

A	B	C	D
1	2	4	8

LEFT OUTER Join $R \overset{L}{\bowtie} S$: $R \overset{L}{\bowtie} S$ ll Tuples from Left side Relation [R]

which failed to satisfy Join condition.

RIGHT OUTER Join : $R \overset{R}{\bowtie} S$: $R \overset{R}{\bowtie} S$ ll Tuples from Right Side Relation (S)

those which failed to satisfy Join condition.

FULL OUTER JOIN $R \overset{F}{\bowtie} S$

$$R \overset{F}{\bowtie} S = R \overset{L}{\bowtie} S \cup R \overset{R}{\bowtie} S$$

Left Outer Join [⟕]

$(R \bowtie S)$

R

A	B	C
1	2	4
3	2	6

Failed to satisfy Join condition

S

B	C	D
2	4	8
2	7	4

$(R \bowtie S)$ =

A	B	C	D
1	2	4	8

$R ⟕ S$ =

A	B	C	D
1	2	4	8
3	2	6	Null

Left outer JOIN

Right outer JOIN

R ⟖ S =

A	B	C	D
1	2	4	8
NULL	2	7	4

Right Outer Join [⟖]

$R \rightouterjoin S =$

A	B	C	D
1	2	4	8
Null	2	7	4

Full Outer Join [⟗]

Full outer join = Left outer join Union Right outer join

$R ⟗ S = R ⟕ S \cup R ⟖ S$

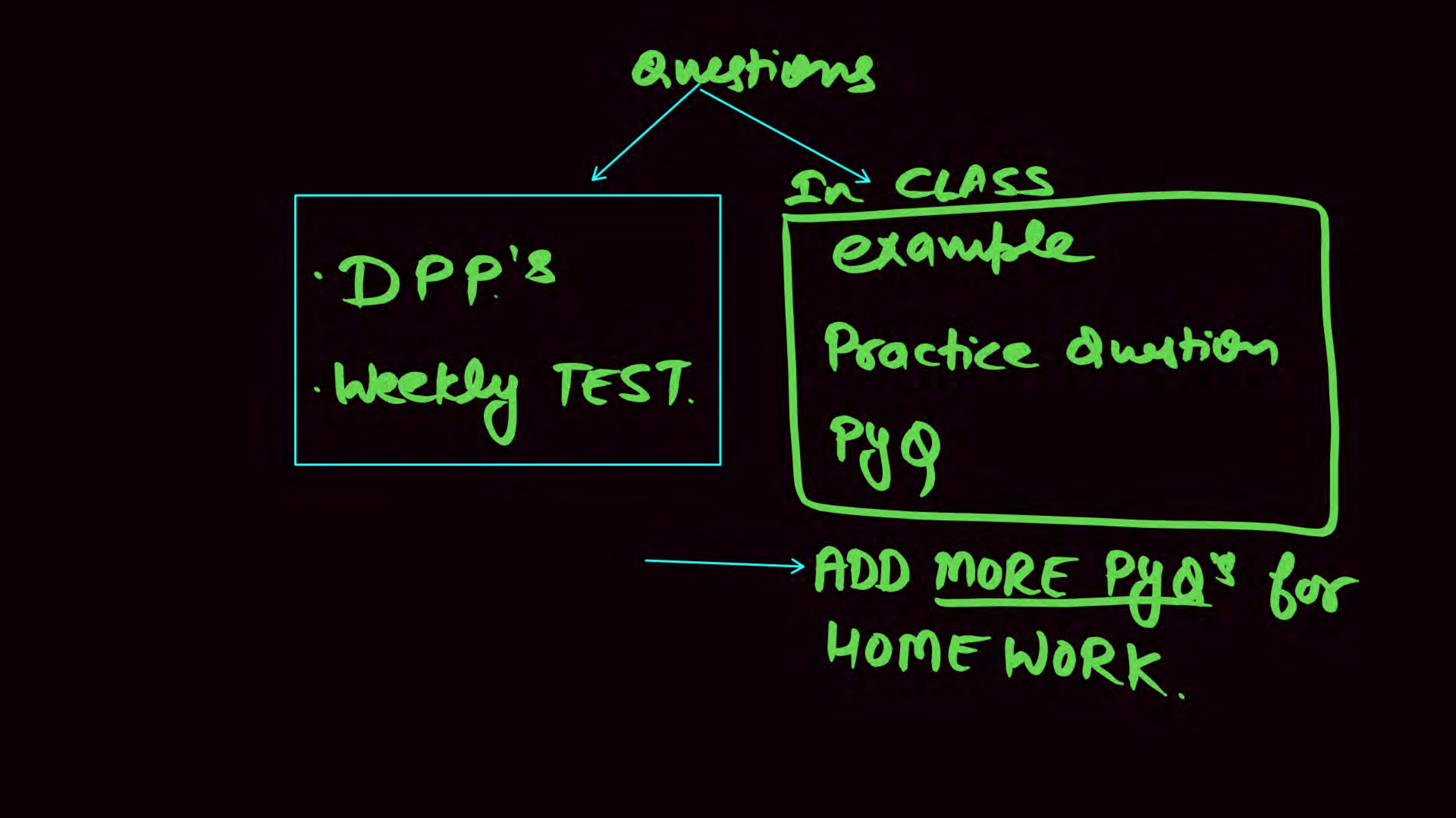
A	B	C	D
1	2	4	8
3	2	6	Null

$\cup$

A	B	C	D
1	2	4	8
Null	2	7	4

$R ⟗ S =$

A	B	C	D
1	2	4	8
3	2	6	Null
Null	2	7	4


Questions
In CLASS
example
Practice question
PYQ
· DPP's
· Weekly TEST.
ADD MORE PYQ's for HOME WORK.

Q.

Let R and S be two relations with the following schema

R(P, Q, R1, R2, R3)

S(P, Q, S1, S2)

Where {P, Q} is the key for both schemas. Which of the following queries are equivalent?

[GATE : 2 Marks
NIC : 3 Marks
Sci.

I. $\pi_P (R \bowtie S)$ R.P = S.P ∧ R.Q = S.Q.

II. $\pi_P(R) \bowtie \pi_P(S)$ ⇒ Here we get in which Q's are Not equal.

III. $\pi_P(\pi_{P,Q}(R) \cap \pi_{P,Q}(S))$ R ∩ S

IV. $\pi_P(\pi_{P,Q}(R) - (\pi_{P,Q}(R) - \pi_{P,Q}(S)))$ R - (R - S)

A Only I and II

B Only I and III

C Only I, II and III

D Only I, III and IV

Ans (D).

$$R \cap S = R - (R - S)$$

R

P	Q	R_1	R_2	R_3
1	2			
2	5			
4	6			
6	8			

P	Q	S_1	S_2
1	2		
2	7		
4	9		
6	8		

$R.P = S.P \wedge R.Q = S.Q$

$\Pi_P(R) \bowtie \Pi_P(S)$

P
1
2
4
6

$\bowtie$

P
1
2
4
6

$\cup$

P
1
2
4
6

(i) $\Pi_P(R \bowtie S)$

Π_P

P	Q
1	2
6	8

$\Rightarrow$

P
1
6

(ii) Π_P

P	Q
1	2
6	8

$\Rightarrow$

P
1
6

Rename operator (ϱ)

It is used to rename table name and attribute names for query processing.

Example:

(I) Stud (Sid, Sname, age)

ϱ(Temp, Stud) : Temp (Sid, Sname, age)

(II) $\varrho_{I, N, A}$ (Stud) : Stud (I, N, A)

All attributes renaming

(III) $\varrho_{\substack{Sid \to I \\ age \to A}}$ (Stud) : Stud (I, Sname, A)

Some attribute renaming

Division

- It is used to retrieve attribute value of R which has paired with every attribute value of other relation S.
- $\pi_{AB}(R)/\pi_{B}(S)$: It will retrieve values of attribute 'A' from R for which there must be pairing 'B' value for every 'B' of S.

Expansion of '/' by using basic operator

- Example: Retrieve sid's who enrolled every course.
- Result:

  $\pi_{sidcid}(Enroll) / \pi_{cid}(Course)$

  Step 1: Sid's not enrolled every course of course relation.

  (Sid's enrolled proper subset of course)

  $\pi_{sid}((\pi_{sid}(Enroll) \times \pi_{cid}(course)) - \pi_{sidcid}(Enroll))$
- Step 2:

  [sid's enrolled every course] = [sid's enrolled some course] – [sid's not enrolled every course]

$\therefore \quad \pi_{sidcid}(E) / \pi_{cid}(c) = \pi_{sid}(E) - \pi_{sid}((\pi_{sid}(E) \times \pi_{cid}(C) - \pi_{sidcid}(E))$

Division

Q. Retrieve all student who are Enrolled **Some course** or **Any course** or at least one course?

Solution Π_{Sid} (Enrolled)

Enrolled	
Sid	**Cid**
S_1	C_1
S_1	C_2
S_1	C_3
S_2	C_1
S_2	C_3
S_3	C_1

Course
Cid
C_1
C_1
C_3

Division

Q. Retrieve all student who are Enrolled every course?

Solution

$\Pi_{Sid.Cid}$ (Enrolled) / Π_{Cid} (Course)

Find

2^{nd} attribute must be same.

Enrolled	
Sid	**Cid**
S_1	C_1
S_1	C_2
S_1	C_3
S_2	C_1
S_2	C_3
S_3	C_1

Course
Cid
C_1
C_1
C_3

Division

$$\Pi_{Sid}(\text{Enrolled}) - \Pi_{Sid}[\Pi_{Sid}(\text{Enrolled}) \times \Pi_{Cid}(\text{Course}) - \text{Enrolled}]$$

Division

$$\Pi_{AB}(R) \,/\, \Pi_{B}(S) = \Pi_{A}(R) - \Pi_{A}\,[\Pi_{A}(R) \times \Pi_{B}(S) - R\,]$$

Find

Connection

$$\Pi_{ABCD}(R) \,/\, \Pi_{CD}(S) \Rightarrow \Pi_{AB}(R) - \Pi_{AB}\,[\Pi_{AB}(R) \times \Pi_{CD}(S) - R\,]$$

Q.

Consider the following three relations in a relational database:

Employee (eId, Name), Brand (bId, bName), Own(eId, bId)

Which of the following relational algebra expressions return the set of eIds who own all the brands? [GATE: 2022]

A $\pi_{eId}\,(\pi_{eId,\,bId}\,(Own / \pi_{bId}\,(Brand))$

B $\pi_{eId}\,(Own) - \pi_{eId}\,((\pi_{eId}\,(Own) \times \pi_{bId}\,(Brand)) - \pi_{eId,\,bId}\,(Own))$

C $\pi_{eId}\,(\pi_{eId,\,bId}\,(Own) / \pi_{bId}\,(Own))$

D $\pi_{eId}\,((\pi_{eId}\,(Own) \times \pi_{bId}\,(Own) / \pi_{bId}\,(Brand))$

Consider the two relation Suppliers and Parts are given below.

Suppliers

S_{no}	P_{no}
S_1	P_1
S_1	P_2
S_1	P_3
S_1	P_4
S_2	P_1
S_2	P_2
S_3	P_2
S_4	P_2
S_4	P_4

Parts

P_{no}
P_2
P_4

$\pi_{S_{no}\,P_{no}}$ (Suppliers) / $\pi_{P_{no}}$ (Parts)

The number of tuples are there in the result when the above relational algebra query executes is _____.

Consider the Database with relations:

S Supplier (Sid, Sname, Rating)

P Parts (Pid, Pname, Color)

S Catalog (Sid Pid, Cost)

Q. Find the Sid of Supplier whose Rating greater than 9?

Q.

Find the Pid of Red Color Parts?

Q.

Retrieve Sid of Supplier whose cost is greater than 20,000?

Q.

Retrieve Sid of Supplier who supplied some Red color parts?

Solution: $\Pi_{Sid}\left[\sigma_{\substack{P.Pid=C.Pid \\ P.Color=Red}} \wedge (Catalog \times Parts)\right]$

Note: Let an Attribute A belongs to R only then

$\sigma_{A = 'a'} (R \bowtie S) = \sigma_{A = 'a'} (R) \bowtie S \rightarrow$ More efficiency query

Note: Let an Attribute A belongs to R only and Attribute B belongs to S only then

$\sigma_{A = 'a' \wedge B = 'b'} (R \bowtie S) = \sigma_{A = 'a'} (R) \bowtie \sigma_{B = 'b'} (S)$

Q.

Consider the following relation schemas:

b-Schema = (b-name, b-city, assets)

a-Schema = (a-num, b-name, bal)

d-Schema = (c-name, a-number)

Let branch, account and depositor be respectively instances of the above schemas. Assume that account and depositor relations are much bigger than the branch relation.

Consider the following query:

$\Pi_{\text{c-name}}\ (\sigma_{\text{b-city} = \text{“agra”} \wedge\ \text{bal} < 0}\ (\text{branch} \bowtie \text{account} \bowtie \text{depositor}))$

Q.

Which one of the following queries is the most efficient version of the above query? [GATE-2007: 2 Marks]

A $\Pi_{\text{c-name}}\ (\sigma_{\text{bal} < 0}\ (\sigma_{\text{b-city} = \text{"Agra"}}\ \text{branch} \bowtie \text{account}) \bowtie \text{depositor})$

B $\Pi_{\text{c-name}}\ (\sigma_{\text{b-city} = \text{"Agra"}}\ \text{branch} \bowtie\ (\sigma_{\text{bal} < 0 =}\ \text{account}) \bowtie \text{depositor})$

C $\Pi_{\text{c-name}}\ (\sigma_{\text{b-city} = \text{"Agra"}}\ \text{branch} \bowtie \sigma_{\text{b-city} = \text{"Agra} \wedge \text{bal} < 0}\ \text{account} \bowtie \text{depositor})$

D $\Pi_{\text{c-name}}\ (\sigma_{\text{b-city} = \text{"Agra"}}\ \text{branch} \bowtie (\sigma_{\text{b-city} = \text{"Agra} \wedge \text{bal} < 0}\ \text{account} \bowtie \text{depositor}))$

Q.

Consider two relations $R_1(A, B)$ with the tuples (1, 5), (3, 7)and R_2 (A, C) = (1, 7)(4, 9)
Assume that R(A, B, C) is the full natural outer join of R_1 and R_2. Consider the following tuples of the form (A, B, C); a = (1, 5, null), b = (1, null, 7), c = (3, null, 9), d = (4, 7, null), e = (1, 5, 7), f = (3, 7, null), g = (4, null, 9). Which one of the following statements is correct? [GATE-2015: 1 Mark]

A. R contains a, b, e, f, g, but not c, d

B. R contains all of a, b, c, d, e, f, g

C. R contains e, f, g, but not a, b

D. R contains e but not f, g

Q.

Consider the following relations given below:

R

A	B
6	6
7	6
8	8

S

C	D
6	7
8	9
8	10

$\Pi_{AD}\,(R \times S) - P_{A \leftarrow B}\,(\Pi_{BD}\,(R \bowtie_{B=C} S))$

Number of tuples return by the above query when it is executed on the above instance of relation R and S is ____

Summary

OPERATION	PURPOSE	NOTATION
SELECT	Selects all tuples that satisfy the selection condition from a relation R.	$\sigma_{\text{<selection condition>}}(R)$
PROJECT	Produces a new relation with only some of the attributes of R, and removes duplicate tuples.	$\pi_{\text{<attribute list>}}(R)$
THETA JOIN	Produces all combinations of tuples from R_1 and R_2 that satisfy the join condition.	$R_1 \bowtie_{\text{<join condition>}} R_2$
EQUIJOIN	Produces all the combinations of tuples from R_1 and R_2 that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{\text{<join condition>}} R_2$, OR $R_1 \bowtie_{\text{(<join condition 1 >), (<join condition 1 >)}} R_2$
NATURAL JOIN	Same as EQUIJOIN except that the join attributes of R_2 are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.	$R_1 *_{\text{<join condition>}} R_2$, OR $R_1 *_{\text{(<join attributes 1 >), (<join attributes 2 >)}} R_2$ OR $R_1 * R_2$

OPERATION	PURPOSE	NOTATION
UNION	Produces a relation that includes all the tuples in R_1 or R_2 or both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cup R_2$
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.	$R_1 \cap R_2$
DIFFERENCE	Produces a relation that includes all the tuples in R_1 and that are not in R_2; R_1 and R_2 must be union compatible.	$R_1 - R_2$
CARTESIAN PRODUCT	Produces a relation that has the attributes of R_1 and R_2 and includes as tuples all possible combinations of tuples from R_1 and R_2.	$R_1 \times R_2$
DIVISION	Produces a relation R(X) that includes all tuples t[X] in $R_1(Z)$ that appear in R_1 in combination with every tuple from $R_2(Y)$, where $Z = X \cup Y$	$R_1(Z) \div R_2(Y)$

THANK YOU!