COMPUTER SCIENCE

Database Management System

Query Language

Lecture_2

Vijay Agarwal sir


TOPICS TO BE COVERED
01
Basic Operators
02
Derived Operators

Relational Algebra

Basic operators

1. π : Projection operator
2. σ : Selection operator

× : Cross-product operator

U : Union

– : Set difference

ϱ : Rename operator

Relational Algebra

Derived operators

∩ : Intersection {using "_"}

⋈ : Join {using X, σ}

/ or ÷ : Division {using π, x, –}

① Selection [σ] → Select tuples Based on Specified Condition.

② Projection [π]

↳ It select Attribute or Attribute list from Relation.

Important points

① $\sigma_{C_1}(\sigma_{C_2}(\sigma_{C_3}(R))) \equiv \sigma_{C_2}(\sigma_{C_3}(\sigma_{C_1}(R)))$

Commutative.

GATE

② $\sigma_{C_1 \wedge C_2 \wedge C_3}(R) \equiv \sigma_{C_1}(\sigma_{C_2}(\sigma_{C_3}(R)))$

In general

C_1: CGPA
A: Name

③ $\pi_A[\sigma_{C_1}(R)] \neq \sigma_{C_1}[\pi_A(R)]$

④ $\sigma_{C_1}[\pi_{A_1}(R)] \xrightarrow{\text{L.H.S transformed into R.H.S}} \pi_{A_1}[\sigma_{C_1}(R)]$

L.H.S — R.H.S

C_1: Condition 1
C_2: Condition 2
C_3: Condition 3

(5) $\pi_A(R) \equiv \pi_A\left(\pi_{AB}(R)\right)$

(6) $\pi_{Attribute}(R) \equiv \pi_{Attribute}\left(\pi_{Attributes''}(R)\right)$

If Attribute $\subseteq$ Attributes'

Special Case (6)

$$\pi_{Att.}\left(\sigma_C(R)\right) = \sigma_C\left(\pi_{Att.}(R)\right)$$

Iff Att. includes all Attributes used in C[Condition]

Now SET OPERATORS $(\cup, \cap, -)$

Set operators [∪, ∩, −]

To Apply Set operators Relation must be **Union Compitable** (**Type Compitable**)

Relation R & Relation S is **Union (Type) Compitable**

iff

1. Arity (Degree (# Attributes) of R & S Must be Same.
2. Range (Domain) of Attribute Must be Similar.

Set operator

U: Union operator

– : Except or minus

∩ : Intersection operator

- To apply set operations relations must be union compatible. (Type)
- R and S relations are union compatible
- If and only if- (iff

(i) Arity of R equal to Arity of S and

(ii) Domain of attributes of R must be same as domain of attributes of s respectively.

(1)

R

A	B	C

(Arity) Degree = 3

S

–	–

#Attributes

(Arity) Degree = 2

$\cup$ $\cap$ $-$ Not possible to Apply.

∵ Arity (Degree | #Attribute) Not Same.

(2)

R

Name	Branch

Name: String, Branch: String

S

CGPA	Contact

CGPA: Num, Contact: integer

$\cup$ $\cap$ $-$ Not possible to Apply

∵ Bcz Domain Not Same.

Registration
C Lang.

RollNo	Name

Registration
JAVA Lang.

RollNo	Name

→ Now we can Apply Set operators.

① Arity (2) of R & S are Same.
② Domain is Similar.

Note: Same Domain with Different Name allowed.

A	B
1	2
3	4

U
∩
—

↑ Apply.

C	D
5	6
7	8

Relation R & Relation S

Note

$R(A_1\ A_2\ A_3 \ldots\ldots A_n)$ & $S(B_1\ B_2\ B_3 \ldots\ldots B_n)$ is union (type) compatable iff.

① Arity of R = Arity of S.

② Domain (A_i) = Domain (B_i) for $1 \le i \le n$

↳ Corresponding pair of Attributes having same Domain.

① Arity(R) = 4 = Arity(S) = 4

② Domain (A_1) = Domain (B_1)
Domain (A_2) = Domain (B_2)
Domain (A_3) = Domain (B_3)
Domain (A_4) = Domain (B_4)

R

A_1	A_2	A_3	A_4

S

B_1	B_2	B_3	B_4

Example

Example 1:

$\pi_{Sid\ Sname}(\ldots\ldots\ldots) \cap \pi_{Sid}(\ldots\ldots\ldots)$

{Arity not same so, set operation not allowed}

Example 2:

String

Numerical

$\pi_{Sid\ Sname}(\ldots\ldots\ldots) \cap \pi_{Sid\ Marks}(\ldots\ldots\ldots)$

{Arity same but Sname domain is different from marks so, not allowed}

Example

$\pi_{Sid\ Sname}(\ldots\ldots\ldots) \cap \pi_{Stud\ ID\ \ Stud\ name}(\ldots\ldots\ldots)$

{Arity and domains are same so, allowed for set operation}

1. Set operation on relation:

R

A
2
2
2
3

S

B
2
2
4

RUS = 2, 3, 4

R-S: Present in R But Not Present in S.

RUS: $\{x/x \in R \vee x \in S\} \equiv$

A
2
3
4

$R - S: \{x/x \in R \wedge x \notin S\} \equiv$

A
3

$R \cap S: \{x/x \in R \wedge x \in S\} \equiv$

A
2

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

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

Minimum

Maximum

Clear:

5 Tuple (M=5)

R

A
B
C
D
E

U

S (JAVA Lang.) (4 Tuple N=4)

(i)

A
B
C
D

OR

(ii)

C
D
P
Q

OR

(iii)

P
Q
R
S

R U S(i) = 5 Max(M, N)

R U S(ii) = 7

R U S(iii) = 9 M+N

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.

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

Set Difference Operation | MINUS | EXCEPT

- 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:

Raghuram Ramakrishnan Textbook

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

1. $S_1 \cup S_2$ & $S_2 \cup S_1$
2. $S_1 - S_2 \longrightarrow$ 1 tuple [22 dustin 7 45.0]
3. $S_2 - S_1 \rightarrow$ 2 Tuple 28 yuppy 9 35.0, 44 Guppy 5 35.0

(1) Union S1 ∪ S2

(1) Union

$$S_1 \cup S_2 \equiv S_2 \cup S_1$$

↳ Commutative.

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 ∪ S2

$S_2 \cup S_1$.

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

$S_2 \cup S_1$

(2) Set Difference

$S_1 - S_2$

Sid	Sname	Rating	age
22	dustin	7	45.0

$S_1 - S_2 \neq S_2 - S_1$

Not Commutative

S1 - S2

$S_2 - S_1 \Rightarrow$

Sid	Sname	Rating	age
28	yuppy	9	35.0
44	guppy	5	35.0

(3) Intersection

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

$S1 \cap S2$ & $S_2 \wedge S_1$

$$S_1 \wedge S_2 \equiv S_2 \wedge S_1$$

Commutative

Korth Book

R

A	B
α	1
α	2
β	1

S

C	B
α	2
β	3

UNION

① R ∪ S =

A	B
α	1
α	2
β	1
β	3

② S ∪ R =

C	B
α	2
β	3
α	1
β	1

Commutative.

$R \cup S = S \cup R$

② MINUS

$R - S \Rightarrow$

α	1
β	1

Ans

$S - R \Rightarrow$

β	3

Ans

$R - S \neq S - R$ Not Commutative.

③ $R \cap S$, $S \cap R$

α	2

$R \cap S = S \cap R$ Commutative

Till Now

- Selection
- Projection
- Union
- Intersection
- Set Difference / MINUS / EXCEPT

CROSS / Cartesian Product (R X S)

R

n_1 Tuple

C_1 Attribute

S

n_2 Tuple

C_2 Attribute

$$R \times S = \begin{matrix} n_1 \times n_2 \text{ Tuple} \\ C_1 + C_2 \text{ Attribute} \end{matrix}$$

Relational Algebra

Basic operators

II. Cross 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) → $C_1 + C_2$ Attribute
- |(R × S)| ⇒ |R| × |S| → $n_1 \times n_2$ Tuples.

① R(A B C) S(D E F)

R: 2 Tuple
3 Attribute

S: 2 Tuple
3 Attribute

A	B	C
1	2	3
4	5	6

D	E	F
7	8	9
10	11	13

RXS = 2X2 = 4 Tuple
3+3 = 6 Attribute

RXS =

A	B	C	D	E	F
1	2	3	7	8	9
1	2	3	10	11	13
4	5	6	7	8	9
4	5	6	10	11	13

Now Variations

(if in Relation R & Relation S Attribute having Same Name [Common Attribute] then

R(A B C)

S(C E F)

① R(A B C) S(C E F)

R: 2 Tuple
3 Attribute

S: 2 Tuple
3 Attribute

A	B	C
1	2	3
4	5	6

C	E	F
7	8	9
10	11	13

Type 1

R X S = 2 X 2 = 4 Tuple
3 + 3 = 6 Attribute

R X S =

A	B	R.C	S.C	E	F
1	2	3	7	8	9
1	2	3	10	11	13
4	5	6	7	8	9
4	5	6	10	11	13

∴

① R(A B C) S(C E F)

R: 2 Tuple
3 Attribute

S: 2 Tuple
3 Attribute

A	B	C
1	2	3
4	5	6

C	E	F
7	8	9
10	11	13

Type2

RXS = 2X2 = 4 Tuple
3+3 = 6 Attribute

RXS =

A	B	3	4	E	F
1	2	3	7	8	9
1	2	3	10	11	13
4	5	6	7	8	9
4	5	6	10	11	13

① R(A B C) S(C E F)

R: 2 Tuple
3 Attribute

S: 2 Tuple
3 Attribute

A	B	C
1	2	3
4	5	6

C	E	F
7	8	9
10	11	13

Type 3
Best
Approach

R X S = 2 X 2 = 4 Tuple
3 + 3 = 6 Attribute

RXS =

R.A	R.B	R.C	S.C	S.E	S.F
1	2	3	7	8	9
1	2	3	10	11	13
4	5	6	7	8	9
4	5	6	10	11	13

R(ABCD) S(CDEF)

Relational Algebra

Basic operators

II.Cross product (×): /CROSS JOIN/ 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|

NOTE:

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

Note If in R & S Any Table is Empty then R X S = Empty.

R X S = S X R Data

Note

for Set operator [∪, ∩, −] Both Relations
MUST be Union (Type) Compitable.

Note

for CROSS | Cartesian Product [CROSS JOIN] Relation
Not Required to be Union (Type) Compitable.

Cross – Product

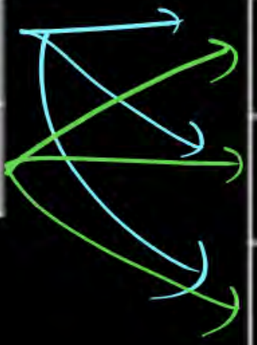
Reserves (R_1) Arity(3)

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

Arity(4)

Sailors (S_1)

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



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

$R_1 \times S_L$:

R_1.Sid	R_1.bid	R_1.day	S_1.Sid	S_1.Sname	S_1.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

S_1 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

$S_1 \times R_1$:

Why CROSS PRODUCT Required?

Soln ① for Performing the Join operations. (Inner Join & outer Join)

② But if we don't wants to Perform Join operations. then why CROSS Product is Required ?

Soln If we Required Something like R.C < S.D

Why CROSS PRODUCT Required?

R

A	B	C
1	2	4
3	2	6

S

B	C	D
2	4	8
2	7	4

If we Required

R.C < S.D ?

If Require R.C < S.D

It is Not possible in Query Language Directly. Because, Query Executed on a DB Table Tuple By Tuple, One Tuple at a time.

So it Fetch a Tuple either from Relation R (or) From Relation S at a time

So Not possible to Compare. that Why we do RXS to Convert them into single Relation.

Why CROSS PRODUCT Required?

R: 2 Tuple
3 Attribute

S: 2 Tuple
3 Attribute

R

A	B	C
1	2	4
3	2	6

S

B	C	D
2	4	8
2	7	4

If we Require

R.C < S.D

R X S = 2X2 = 4 Tuple
3+3 = 6 Attribute

R.C < S.D

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

R X S =

But

R: 500 Tuple

S: 400 Tuple

$R \times S = 500 \times 400 = 2,00,000$ Tuple (2 lakh)

Only Cross Product is Not Meaningful untill unless we Not Applying Any Condition [Selection] & Not Selecting (Project) Any Attribute.

So JOIN

Join Operations

(1) Conditional Join ($\bowtie_c$) [Theta Join]

(2) Equi join

(3) Natural join

Inner Join.

(4) Left outer join

(5) Right outer join

(6) Full outer join

OUTER JOIN.

Inner Join

1. Conditional (Theta) Join.
2. Equi Join
3. Natural Join

OUTER JOIN

1. Left Outer JOIN
2. Right Outer JOIN
3. FULL OUTER JOIN

JOIN OPERATION ⋈

① Natural JOIN ($R \bowtie S$) : It's Derived operator Performed in 3 steps.

Step1 : Cross [Cartesian] Product of R & S.

Step2 : Select the tuple which Satisfy equality Condition on All Common Attributes of R & S (FROM R X S).

Step3 Projection of Distinct Attribute.

JOIN OPERATION ⋈

Natural JOIN $(R \bowtie S)$: It's Derived operator Performed in 3 Steps.

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

Step 3 → Distinct Attribute
Step 2 → equality Condition on ALL Common Attribute
Step 1 → $(R \times S)$

Eg) $R(A\underline{BC})$ $S(\underline{BC}\,D)$

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

Conditional Join

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

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

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

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 condition c contains only equalities.

R_1 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

$S1 \bowtie_{sid} R1$

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.

Join (⋈)

I. Natural join (⋈)

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

Example:

- T_1 (ABC) and T_2 (BCDE)

$$\therefore \quad T_1 \bowtie T_2 = \pi_{ABCDE}\left(\begin{array}{c}\sigma_{T_1 \cdot B} = T_{2 \cdot B}(T_1 \times T_2) \\ \cap T_1 \cdot C = T_2 \cdot C\end{array}\right)$$

- T_1 (AB) and T_2 (CD)

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

NOTE:

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

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

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

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

$R \bowtie S =$

A	B	C	D
1	2	4	8

Left Outer Join [⟕]

$(R ⟕ S)$

R

A	B	C
1	2	4
3	2	6

S

B	C	D
2	4	8
2	7	4

$(R ⟕ S)$ =

A	B	C	D
1	2	4	8

$R ⟕ S$ =

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

Right Outer Join [⟖]

$R \bowtie S =$

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

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?

I. $\pi_P (R \bowtie S)$

II. $\pi_P(R) \bowtie \pi_P(S)$

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

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

A Only I and II

B Only I and III

C Only I, II and III

D Only I, III and IV

THANK YOU!