Relational Algebra

Complete Summary

Part – 2

Instructor:

Deepak Poonia

MTech, IISc Bangalore

GATE CSE AIR 53; AIR 67;
AIR 107; AIR 206; AIR 256

DBMS Complete Course:

https://www.goclasses.in/courses/Database-Management-Systems

NOTE :

Complete Discrete Mathematics & Complete Engineering

Mathematics Courses, by GO Classes, are **FREE** for ALL learners.

Visit here to watch : https://www.goclasses.in/s/store/

SignUp/Login on Goclasses website for free and start learning.

GO CLASSES

FREE

COMPLETE COURSE

Engineering Mathematics

GATE CS IT

Freely Available Features

FREE

- All Video Lectures
- Quizzes
- Homeworks, Practice Sets
- Annotated Notes
- Toppers' Hand Written Notes
- GATE PYQs Video Solutions

SACHIN MITTAL SIR

www.goclasses.in

/ GoClasses

GO CLASSES

FREE

COMPLETE COURSE

Discrete Mathematics

GATE CS IT

Freely Available Features

FREE

- All Video Lectures
- Quizzes, Homeworks
- Annotated Notes
- Toppers' Hand Written Notes
- Summary Lectures
- GATE PYQs Video Solutions

DEEPAK POONIA SIR

www.goclasses.in

/ GoClasses

GO CLASSES

FREE

COMPLETE COURSE

Linear Algebra

GATE DA

Freely Available Features

FREE

- All Video Lectures
- Quizzes
- Homeworks, Practice Sets
- Annotated Notes
- Toppers' Hand Written Notes
- GATE PYQs Video Solutions

SACHIN MITTAL SIR

www.goclasses.in

/ GoClasses

We are on Telegram. **Contact us** for any help.

Link in the Description!!

Join GO Classes **Doubt Discussion** Telegram Group :

Username:

@GATECSE_GOCLASSES

We are on Telegram. **Contact us** for any help.

Join GO Classes Telegram Channel, Username: **@GOCLASSES_CSE**

Join GO Classes **Doubt Discussion** Telegram Group :

Username: **@GATECSE_Goclasses**

(Any doubt related to Goclasses Courses can also be asked here.)

Join GATEOverflow Doubt Discussion Telegram Group :

Username: **@GateOverflow_CSE**

GO CLASSES


GO CLASSES
GO CLASSES
GO CLASSES
GO CLASSES
GO CLASSES
GO CLASSES

Relational Algebra

Complete Summary

Part – 2

Next Topic:

Writing RA Queries

Consider the Twitter database Table:

- Follows (pname1, pname2) - Person pname1 *follows* person pname2

T

T (uid1, uid2)

A → B
B → A
B → C
A → C
C → B

A follows B
B follows A
A follows C
C follows B

Q 1: In the Twitter database, find all people whom "A" follows.

T (uid1, uid2)

A → B
B → A
B → C
A → C
C → B

Q 1: In the Twitter database, find all people whom "A" follows.

Idea: T (uid1 uid2) → Project

fixed

A	?

$$\pi_{uid2}\left(\sigma_{uid1='A'}(T)\right)$$

T (uid1, uid2)

A → B
B → A
B → C
A → C
C → B

Q 2: In the Twitter database, find all people who follow “A”.

T (uid1, uid2)

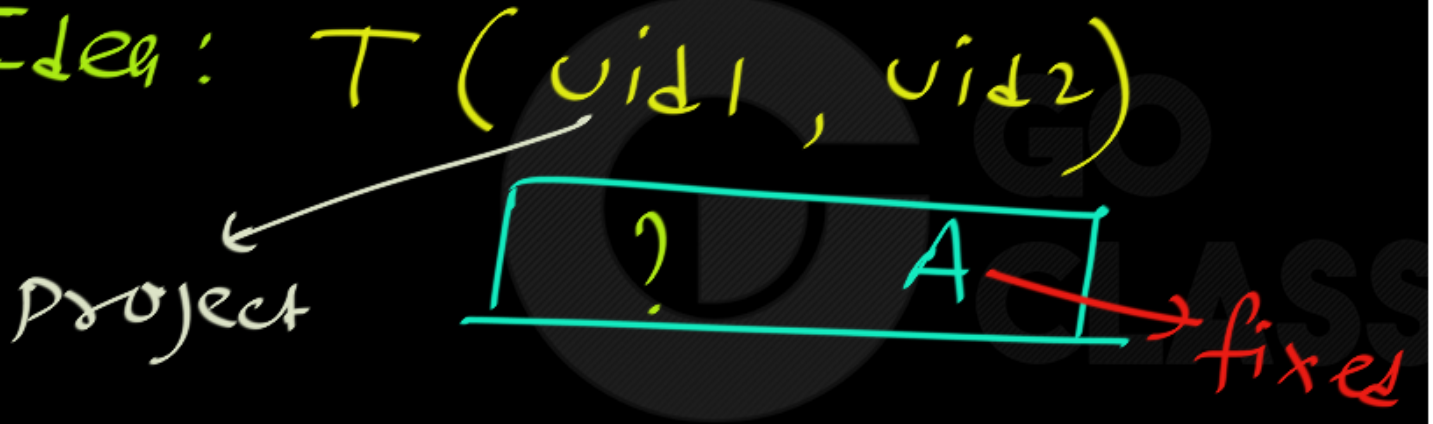
$A \rightarrow B$
$B \rightarrow A$
$B \rightarrow C$
$A \rightarrow C$
$C \rightarrow B$

Q 2: In the Twitter database, find all people who follow "A".

T (uid1, uid2)

A → B
B → A
B → C
A → C
C → B

Idea : T (uid1, uid2)



$$\Pi_{uid1}\left(\sigma_{uid2='A'}(T)\right)$$

Q 3: In the Twitter database, find all people who follow at least two different people.

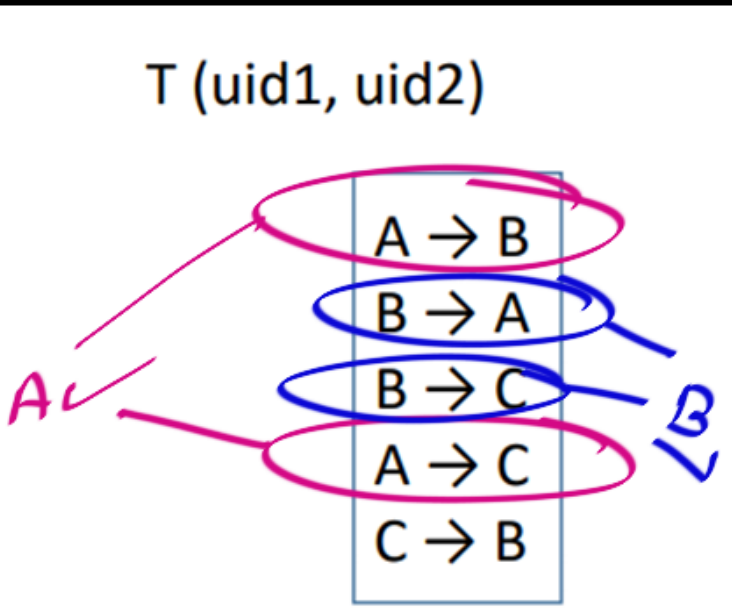
T (uid1, uid2)

$A \rightarrow B$
$B \rightarrow A$
$B \rightarrow C$
$A \rightarrow C$
$C \rightarrow B$

Q 3: In the Twitter database, find all people who follow at least two different people.

How many Tuples Give this Info ? 2 Tuples

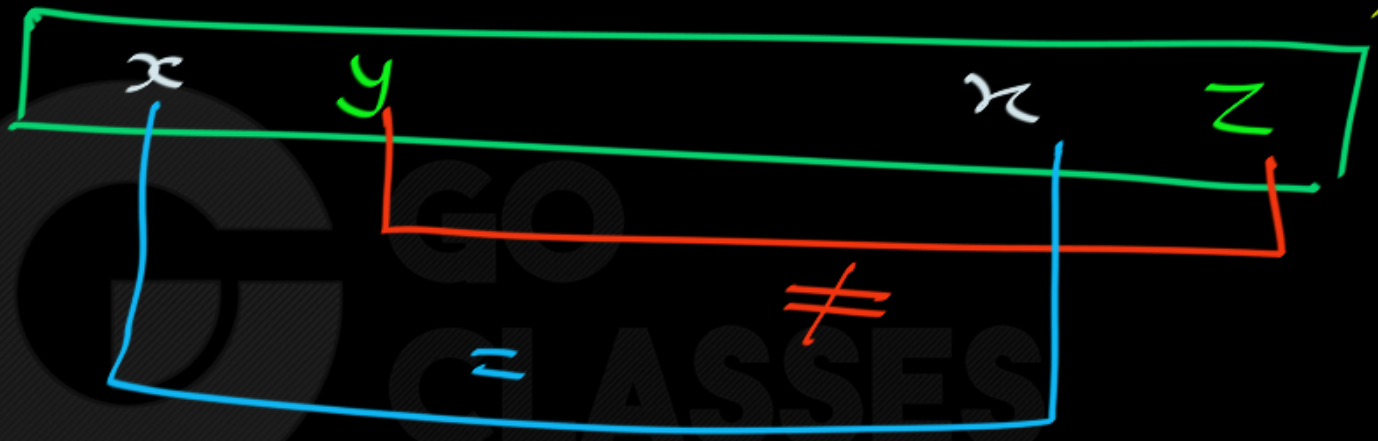
Ans: A, B



Ex:

$T_1(vid1, vid2) \times T_2(vid1, vid2)$

Project



$$\Pi_{T_1.vid1}\left(\sigma_{(T_1.vid1 = T_2.vid1) \wedge (T_1.vid2 \neq T_2.vid2)}\left(\rho_{T_1}(T) \times \rho_{T_2}(T)\right)\right)$$

Q 4: In the Twitter database, find all people who follow at least three different people.

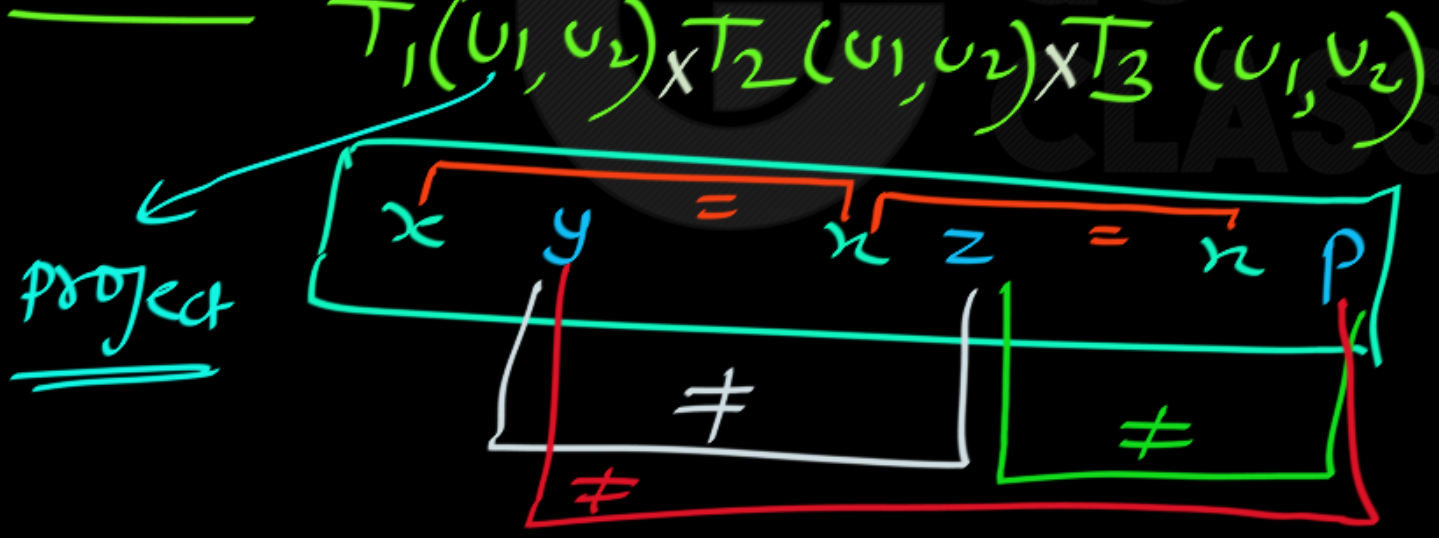
T (uid1, uid2)

$A \rightarrow B$
$B \rightarrow A$
$B \rightarrow C$
$A \rightarrow C$
$C \rightarrow B$

$A \rightarrow D$

Q 4: In the Twitter database, find all people who follow at least three different people.

Idea:



T (uid1, uid2)

A → B
B → A
B → C
A → C
C → B

A → D

$$\pi_{T_1.uid1}\left(\sigma_{(T_1.uid1 = T_2.uid1) \wedge (T_2.uid1 = T_3.uid1) \wedge (T_1.uid2 \neq T_2.uid2) \wedge (T_2.uid2 \neq T_3.uid2) \wedge (T_1.uid2 \neq T_3.uid2)}\left(\rho_{T_1}(T) \times \rho_{T_2}(T) \times \rho_{T_3}(T)\right)\right)$$

Q 5: In the Twitter database, find all pairs of people who follow each other.

T (uid1, uid2)

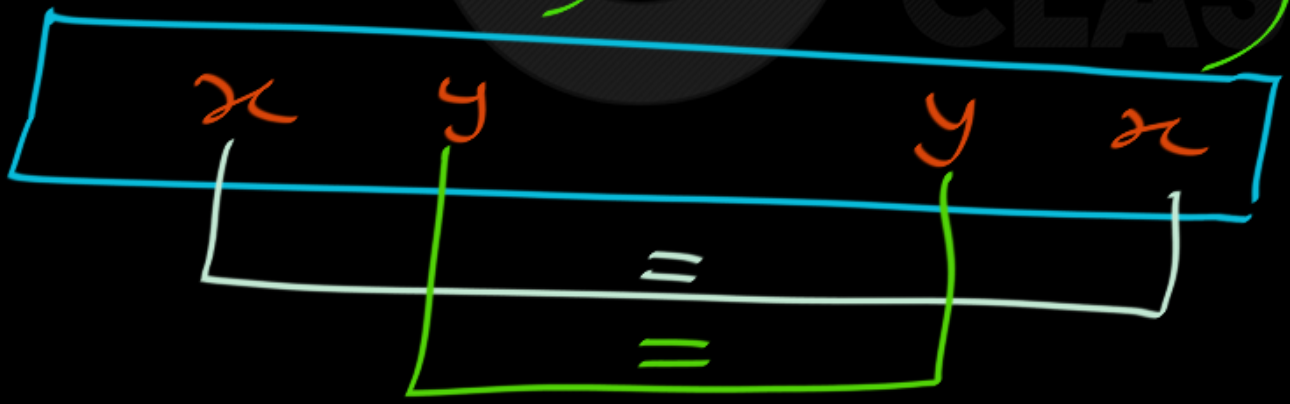
A → B
B → A
B → C
A → C
C → B

Q 5: In the Twitter database, find all pairs of people who follow each other.

Project ←

Given by two Tuples ↙

$T_1(uid1, uid2) \times T_2(uid1, uid2)$



T (uid1, uid2)

A → B
B → A
B → C
A → C
C → B

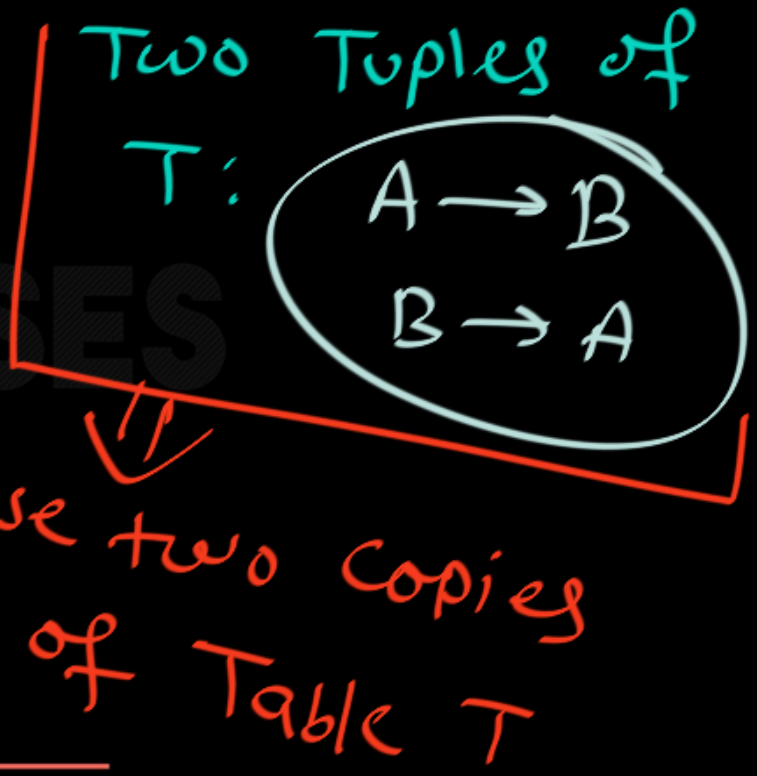
X

Ans:-

A	B
B	A
B	C
C	B

Simple Guideline:

Check How many rows you need to see to get the desired answer.



$$\Pi_{T_1.uid1,\ T_1.uid2}\left(\sigma_{(T_1.uid1 = T_2.uid2) \wedge (T_1.uid2 = T_2.uid1)}\left(\rho_{T_1}(T) \times \rho_{T_2}(T)\right)\right)$$

Renaming: example

T (uid1, uid2)

A → B
B → A
B → C
A → C
C → B

- Find all true friends in twitter dataset

who follow each other

Renaming: example

T (uid1, uid2)

A → B
B → A
B → C
A → C
C → B

- Find all true friends in twitter dataset
- By renaming T we created two identical relations R and S, and we now extract all tuples where for each pair X → Y in R there is a pair Y → X in S

$$\pi_{R.uid1,\ R.uid2}\ \sigma_{R.uid1=S.uid2\ AND\ R.uid2\ =\ S.uid1}(\rho_R\ (T)\ x\ \rho_S\ (T))$$

GATE CSE 2024 | Set 2 | Question: 35

asked Feb 16 • retagged Apr 27 by **Arjun**

7

The relation schema, Person (pid, city), describes the city of residence for every person uniquely identified by pid. The following relational algebra operators are available: selection, projection, cross product, and rename.

To find the list of cities where at least 3 persons reside, using the above operators, the minimum number of cross product operations that must be used is

A. 1
B. 2
C. 3
D. 4

GATE CSE 2024 | Set 2 | Question: 35

asked Feb 16 • retagged Apr 27 by **Arjun**

4,053 views

FD

7

The relation schema, Person (pid, city), describes the city of residence for every person uniquely identified by pid. The following relational algebra operators are available: selection, projection, cross product, and rename. ρ

σ π $\times$

To find the list of cities where at least 3 persons reside, using the above operators, the minimum number of cross product operations that must be used is

A. 1
B. 2
C. 3
D. 4

Silly

I need 3 Copies of Table Person

3 Tuples give this info

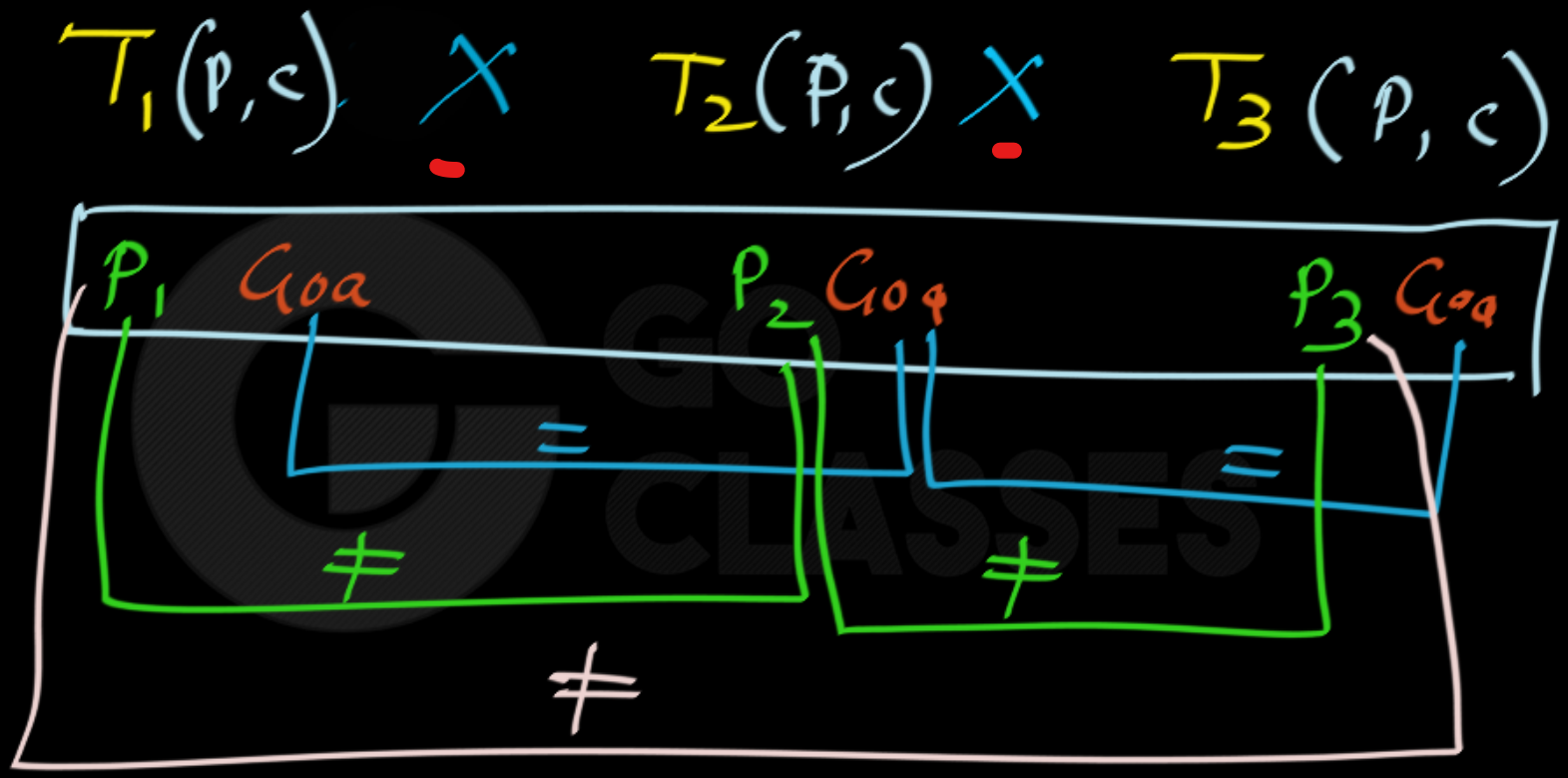
Person (pid, city)

P_1	C
P_2	C
P_3	C

Mistake

P_4 C

Idea:



$$\pi_{T_1.city}\left(\sigma_{(T_1.city = T_2.city) \wedge (T_2.city = T_3.city) \wedge (T_1.pid \neq T_2.pid) \wedge (T_2.pid \neq T_3.pid) \wedge (T_1.pid \neq T_3.pid)}\left(\rho_{T_1}(person) \times \rho_{T_2}(person) \times \rho_{T_3}(person)\right)\right)$$

Person (pid, city)

Next Topic:

Algebra Operators

Derived Operators

Relational Algebra

- Procedural language
- Queries in relational algebra are applied to relation instances, result of a query is again a relation instance
- Six basic operators in relational algebra:

select	σ	selects a subset of tuples from reln
project	π	deletes unwanted columns from reln
Cartesian Product	$\times$	allows to combine two relations
Set-difference	$-$	tuples in reln. 1, but not in reln. 2
Union	$\cup$	tuples in reln 1 plus tuples in reln 2
Rename	ρ	renames attribute(s) and relation

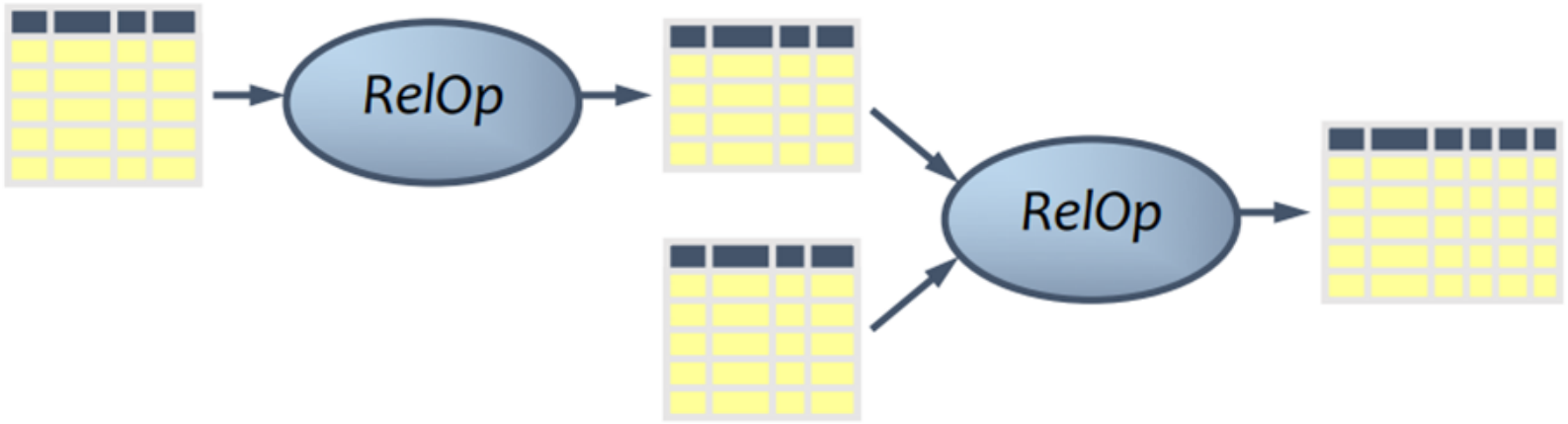
- The operators take one or two relations as input and give a new relation as a result (relational algebra is "closed").

Additional Operators

These operators do not add any power (expressiveness) to the relational algebra but simplify common (often complex and lengthy) queries.

Set-Intersection	$\cap$	
Natural Join	$\bowtie$	
Condition Join	$\bowtie_C$	(also called Theta-Join)
Division	$\div$	

Relational algebra Operators



- Core operators:
  - Selection, projection, cross product, union, difference, and renaming
- Additional, derived operators:
  - Join, natural join, intersection, etc.
- Compose operators to make complex queries

Next Topic:

Join Operations

Join ≡ Inner Join

(Inner) Join :

① Conditional Join (Theta Join)

$\bowtie_C \equiv \bowtie_\theta$

1.1 : Equi-Join

② Natural Join ($\bowtie$)

Join Operations:

Join == Inner Join

- Conditional Join (Theta Join / Theta Inner Join)
  - Equi-Join (Equi Inner Join)
- Natural Join (Natural Inner Join)

Join Operations in Relational Algebra:

Conditional (Inner) Join
(Theta Join)

Syntax:

$$R \bowtie_C S \equiv \sigma_C (R \times S)$$

$$R \bowtie_\theta S \equiv \sigma_\theta (R \times S)$$

Tuple selection Condition

$$R \bowtie_C S \equiv \sigma_C (R \times S)$$

Evaluation: $R \bowtie_C S$

Tuples of R will match with those tuples of S such that condition C is satisfied.

Combining Cross-product with selection

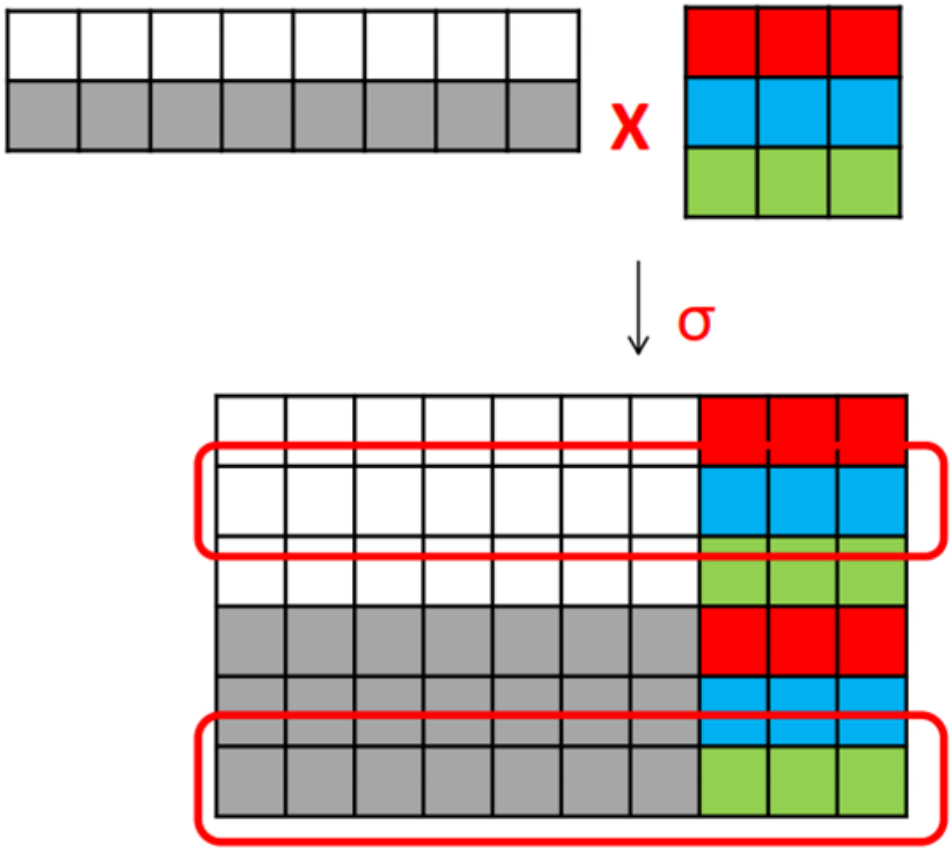
$R \bowtie_C S$ → Tuple selection condition

1.The result is constructed as follows:

a)Take the Cartesian product of **R** and **S**.

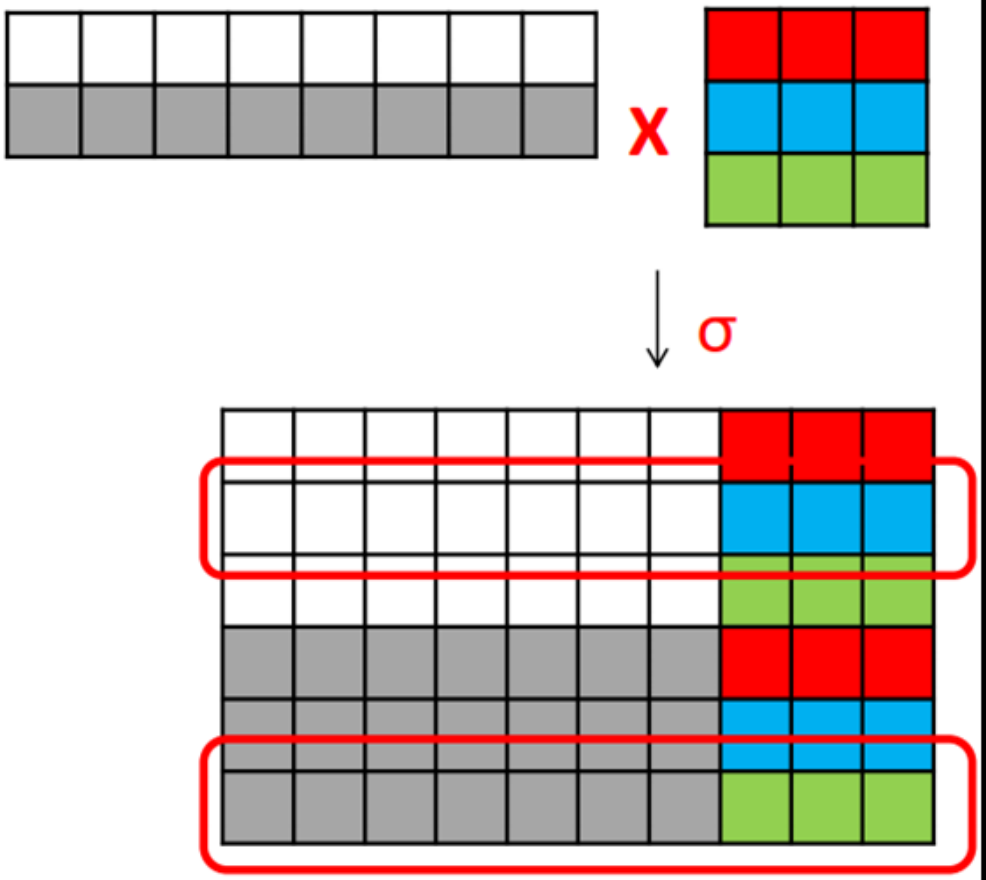
b) Select from the product only those tuples that satisfy the condition **C**.

2.Schema for the result is the union of the schema of **R** and **S,** with **"R"** or **"S"** prefix as necessary.



$T=\sigma_{condition}\ (R \times S)$

Cross-product with selection



$T=\sigma_{condition}\ (R \times S)$

- Example: given two relations r, s

r

A	B	C
1	2	3
4	5	6
7	8	9

s

D	E
3	1
6	2

Dangling Tuple

⇓

NOT matching with anyone

$r \bowtie_{B<D} s$

Theta Join (Condition Join)

$\theta \equiv$ Condition

A	B	C	D	E
1	2	3	3	1
1	2	3	6	2
4	5	6	6	2

$\equiv \sigma_{B<D}$

$(r \times s)$

- Example: given two relations r, s

r

A	B	C
1	2	3
4	5	6
7	8	9

s

D	E
3	1
6	2

$r \bowtie_{B<D} s$

A	B	C	D	E
1	2	3	3	1
1	2	3	6	2
4	5	6	6	2

Note: $R(A, B)$ $S(B, C)$

Schema of $R \bowtie_{\theta} S \equiv \sigma_{\theta}(R \times S)$

$\equiv$ Same as Schema of $R \times S$

$R \bowtie_{\theta} S$:

• A	R.B	S.B	C

Condition Joins

The most general version of the join operation accepts a *join condition* c and a pair of relation instances as arguments, and returns a relation instance. The *join condition* is identical to a *selection condition* in form. The operation is defined as follows:

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

Thus $\bowtie$ is defined to be a cross-product followed by a selection. Note that the condition c can (and typically *does*) refer to attributes of both R and S. The reference to an attribute of a relation, say R, can be by position (of the form $R.i$) or by name (of the form $R.name$).

Dangling Tuple:

A tuple that fails to pair with any tuple of the other relation in a join is said to be a dangling tuple.

Dangling Tuples: None

A	B	C
1	2	3
6	7	8
9	7	8

(a) Relation U

B	C	D
2	3	4
2	3	5
7	8	10

(b) Relation V

A	U.B	U.C	V.B	V.C	D
1	2	3	2	3	4
1	2	3	2	3	5
1	2	3	7	8	10
6	7	8	7	8	10
9	7	8	7	8	10

Figure 2.17: Result of $U \bowtie_{A<D} V$

Conditional Join

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0

Figure 4.1 Instance $S1$ of Sailors

sid	*bid*	*day*
22	101	10/10/96
58	103	11/12/96

Figure 4.3 Instance $R1$ of Reserves

Tuples in o/p ??

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

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0

Figure 4.1 Instance $S1$ of Sailors

sid	*bid*	*day*
22	101	10/10/96
58	103	11/12/96

Dangling Tuples

Figure 4.3 Instance $R1$ of Reserves

(*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

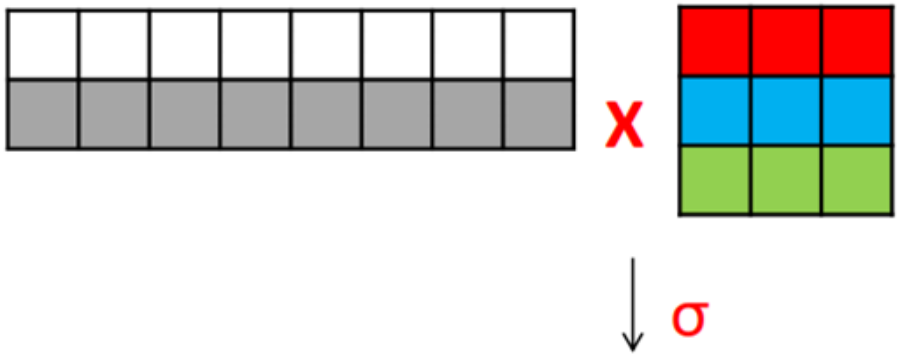
Figure 4.12 $S1 \bowtie_{S1.sid<R1.sid} R1$

Shortcut: Theta-join

1.The result of this operation is constructed as follows:

a)Take the Cartesian product of **R** and **S**.

b) Select from the product only those tuples that satisfy the condition **C**.



2.Schema for the result is the union of the schema of **R** and **S,** with **"R"** or **"S"** prefix as necessary.

$$T = R \bowtie_{condition} S$$

Shortcut for

$$T = \sigma_{condition} (R \times S)$$

7. Suppose relation R(A,B) has the tuples:

A	B
1	2
3	4
5	6

and the relation S(B,C,D) has tuples:

B	C	D
2	4	6
4	6	8
4	7	9

Compute the theta-join of R and S with the condition R.A < S.C AND R.B < S.D. Then, identify from the list below one of the tuples in $R \mid\!><\!\mid_{R.A < S.C \text{ AND } R.B < S.D} S$. You may assume the schema of the result is (A, R.B, S.B, C, D).

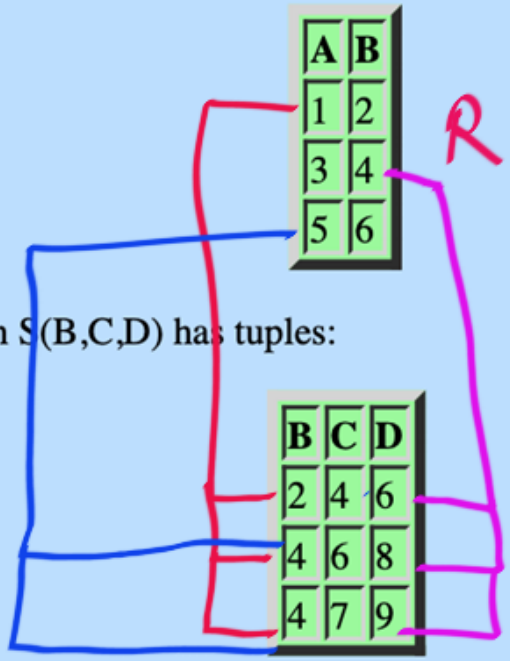
- a) (3,4,4,6,8)
- b) (3,4,5,7,9)
- c) (1,2,4,4,6)
- d) (1,2,2,6,8)

7. Suppose relation R(A,B) has the tuples:

A	B
1	2
3	4
5	6

and the relation S(B,C,D) has tuples:

B	C	D
2	4	6
4	6	8
4	7	9



$R \bowtie S$

$(R.A < S.C \text{ and } R.B < S.D)$

$\rightarrow \emptyset$

$\Rightarrow R(A,B) < S(C,D)$

Compute the theta-join of R and S with the condition R.A < S.C AND R.B < S.D. Then, identify from the list below one of the tuples in $R \mid\!><\!\mid_{R.A < S.C \text{ AND } R.B < S.D} S$. You may assume the schema of the result is (A, R.B, S.B, C, D).

- a) (3,4,4,6,8) ✓
- b) (3,4,5,7,9) ✗
- c) (1,2,4,4,6) ✓
- d) (1,2,2,6,8) ✗

#Tuples in o/p:

3 + 3 + 2 = 8 ✓

No Dangling Tuple

Join Operations in Relational Algebra:

Equi Join

(Join with Equality Condition)

Equi-Join : Special Case of Conditional Join

- Only "=" Comparison is used
- Can't Use $<, \leq, \leq, \geq, \neq$

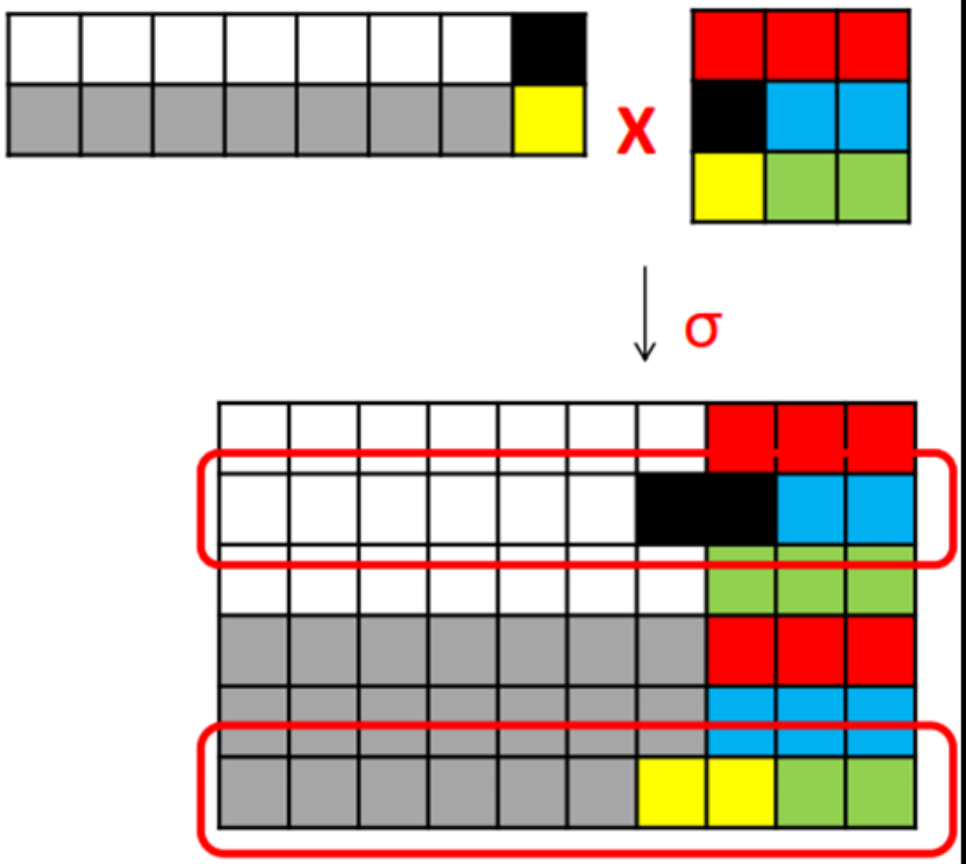
Condition Join

- Notation: $r \bowtie_C s$

If C involves only the comparison operator "$=$", the condition join is also called *Equi-Join*.

Subtype of theta-join: Equijoin

1.Equijoin is a subset of theta-joins where the join condition is equality



$T = R \bowtie_{R.A = S.B} S$

Shortcut for

$T = \sigma_{R.A = S.B} (R \times S)$

r

A	B	C
4	5	6
7	8	9

s

C	D
6	8
10	12

$\Rightarrow$ S

SC	D
6	8
10	12

renaming operation for attributes of S

$$r \bowtie_{C=SC} (\rho_{S(SC,D)}(s))$$

Conditional Join

Equi Join

A	B	C	SC	D
4	5	6	6	8

r

A	B	C
4	5	6
7	8	9

s

C	D
6	8
10	12

$r \bowtie_{C=SC} (\rho_{S(SC,D)}(s))$

A	B	C	SC	D
4	5	6	6	8

Consider three tables T1, T2 and T3 below. Show the results of the following operations:
(**Remember, every Relational Algebra expression results into a table. For each question, you need to write down the column (attribute) names on the top, similar to tables below**)

T1

P	R	C
10	a	5
15	b	8
25	a	6
15	b	5
10	b	6

T2

A	B	C
10	b	6
25	c	3
10	b	5
25	c	6

T3

A	B	C
10	b	6
30	c	3
25	c	6

f) *T1 equi-join (T1.C = T3.C) T3* [equi-join is done on the specified attributes]
(i.e., get all combination of tuples that have equal values on join condition attributes, T1.C and T3.C; attribute list is the same as the Cartesian Product)

Consider three tables T1, T2 and T3 below. Show the results of the following operations:
(**Remember, every Relational Algebra expression results into a table. For each question, you need to write down the column (attribute) names on the top, similar to tables below**)

T1 (P	R	C)
10	a	5
15	b	8
25	a	6
15	b	5
10	b	6

T3 (A	B	C)
10	b	6
30	c	3
25	c	6

Dangling

f) *T1 equi-join (T1.C = T3.C) T3* [equi-join is done on the specified attributes]
(i.e., get all combination of tuples that have equal values on join condition attributes, T1.C and T3.C; attribute list is the same as the Cartesian Product)

$T_1 \bowtie T_3$

$T_1.C = T_3.C$

P	R	$T_1.C$	A	B	$T_3.C$
25	a	6	10	b	6
25	a	6	25	c	6
10	b	6	10	b	6
10	b	6	25	c	6

Consider three tables T1, T2 and T3 below. Show the results of the following operations:
(**Remember, every Relational Algebra expression results into a table. For each question, you need to write down the column (attribute) names on the top, similar to tables below**)

T1 (P	R	C)
10	a	5
15	b	8
25	a	6
15	b	5
10	b	6

T2 (A	B	C)
10	b	6
25	c	3
10	b	5
25	c	6

T3 (A	B	C)
10	b	6
30	c	3
25	c	6

h) *T1 equi-join (T1.P = T3.A) T3* [equi-join is done on the specified attributes]
(i.e., get all combination of tuples that have equal values on join condition attributes, T1.P and T3.A; attribute list is the same as the Cartesian Product)

Join Operations in Relational Algebra:

Natural Join

(Equality of ALL Common Attributes)

Natural Join : (Very Important)

$R \bowtie S$

1. No Conditional Specifies
2. Implicitly Condition : Equality of ALL Common attributes
3. Result will NOT have Redundant Columns

A	B
1	2
3	4

(a) Relation R

B	C	D
2	5	6
4	7	8
9	10	11

(b) Relation S

$R \bowtie S$

One Copy

A	B	C	D
1	2	5	6
3	4	7	8

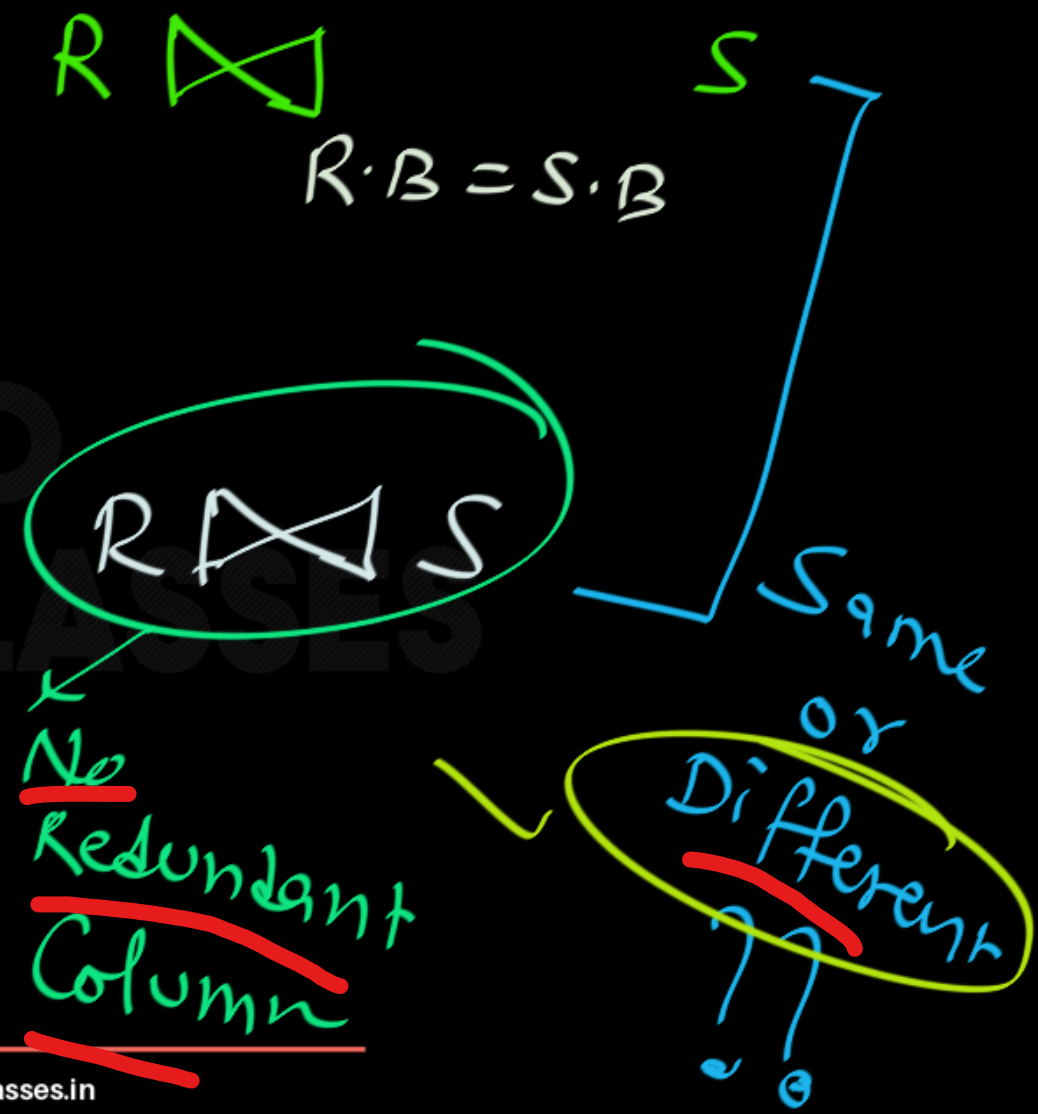
Dangling Tuple

A	B
1	2
3	4

(a) Relation R

B	C	D
2	5	6
4	7	8
9	10	11

(b) Relation S

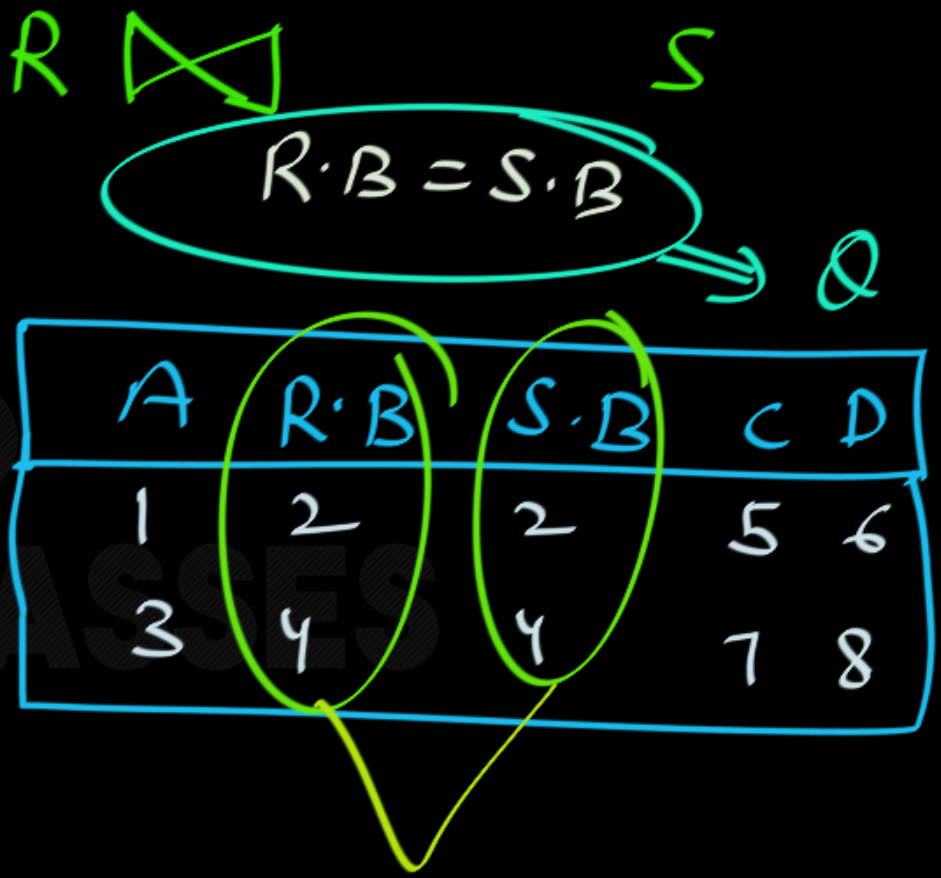


A	B
1	2
3	4

(a) Relation R

B	C	D
2	5	6
4	7	8
9	10	11

(b) Relation S



A	B
1	2
3	4

(a) Relation R

B	C	D
2	5	6
4	7	8
9	10	11

(b) Relation S

Example 2.13: The natural join of the relations R and S from Fig. 2.14(a) and (b) is

A	B	C	D
1	2	5	6
3	4	7	8

The only attribute common to R and S is B. Thus, to pair successfully, tuples need only to agree in their B components. If so, the resulting tuple has components for attributes A (from R), B (from either R or S), C (from S), and D (from S).

A	B	C
1	2	3
6	7	8
9	7	8

(a) Relation U

B	C	D
2	3	4
2	3	5
7	8	10

(b) Relation V

$U(BC) = V(BC)$

Implicit Condition

A	B	C	D
1	2	3	4
1	2	3	5
6	7	8	10
9	7	8	10

(c) Result $U \bowtie V$

Figure 2.16: Natural join of relations

No Dangling Tuple

Natural Join

- A useful join variant (*natural* join) connects two relations by:
  - Equating attributes of the same name, and
  - Projecting out one copy of each pair of equated attributes.
- Denoted R3 := R1 ⋈ R2.

Natural Join in terms of Basic opⁿ:

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

R ⋈ S :

$$\Pi_{A, R.B, R.C, D}\left(\sigma_{R.B=S.B \wedge R.C=S.C}(R \times S)\right)$$

Projecting unique attributes

Special case of equijoin:
Natural Join

$R \bowtie S$

Let $A_1, A_2, \ldots, A_n$ be the attributes in both the schema of **R** and the schema of **S**.

Then a tuple ***r*** from **R** and a tuple ***s*** from **S** are successfully paired if and only if ***r*** and ***s*** agree on each of their common attributes $A_1, A_2, \ldots, A_n$.

Still the same meaning as:

$T = \sigma_{R.A = S.A} (R \times S)$,

but common attributes are not duplicated as in Cartesian Product

Derived operator: natural join

- Input: two tables R and S
- Notation: $R \bowtie S$
- Purpose: relate rows from two tables, and
  - Enforce equality between identically named columns
  - Eliminate one copy of identically named columns
- Shorthand for $\pi_L(R \bowtie_p S)$, where
  - p equates each pair of columns common to R and S
  - L is the union of column names from R and S (with duplicate columns removed)

Q: R(A, B) S(C, D)

R ⋈ S ?

Q: R(A, B) S(C, D)

R ⋈ S ?

$$\Pi_{A,B,C,D}\left(\sigma_{\text{No Condition}}(R \times S)\right) \equiv R \times S$$

Natural Join if no common attributes:

If there are no attributes in common between two relations and you perform a natural join, it will return the cartesian product of the two relations.

A	B	C
1	2	3
6	7	8
9	7	8

(a) Relation U

B	C	D
2	3	4
2	3	5
7	8	10

(b) Relation V

#Tuples in o/p ?

#Dangling Tuples ?

$$U \bowtie_{A<D \text{ AND } U.B \neq V.B} V$$

A	B	C
1	2	3
6	7	8
9	7	8

(a) Relation U

B	C	D
2	3	4
2	3	5
7	8	10

(b) Relation V

A	U.B	U.C	V.B	V.C	D
1	2	3	7	8	10

$$U \bowtie_{A<D \text{ AND } U.B \neq V.B} V$$

Tuples in o/p = 1

Dangling Tuples = 4

Example 2.16: Here is a theta-join on the same relations U and V that has a more complex condition:

$$U \bowtie_{A<D \text{ AND } U.B \neq V.B} V$$

That is, we require for successful pairing not only that the A component of the U-tuple be less than the D component of the V-tuple, but that the two tuples disagree on their respective B components. The tuple

A	$U.B$	$U.C$	$V.B$	$V.C$	D
1	2	3	7	8	10

is the only one to satisfy both conditions, so this relation is the result of the theta-join above. □

Next Topic:

Outer Join

Part of Extended Relation Algebra

NOT Part of Original Relational Algebra

Outer Join:

Outer joins are NOT considered part of the classical relational algebra discussed so far.

These operations enhance the expressive power of the original relational algebra.

Increase Expressive Power of Relation Algebra

In GATE

Ullman Book

(Original) Relational Algebra :

(1) No Concept of NULL

(2) Basic : $\sigma, \Pi, \times, \cup, -, \rho$

Derived : $\cap, \bowtie, \bowtie_c, \div$

Extended Rel. Alg : Outer Join

Outer Join:

Outer Join

=

Inner Join + Dangling Tuples with NULLs

Outer Join:

Outer Join =
Inner Join + Dangling Tuples with NULLs

Left Outer Join = Inner Join + Dangling Tuples of Left Table with NULLs

Right Outer Join = Inner Join + Dangling Tuples of Right Table with NULLs

Full Outer Join = Inner Join + Dangling Tuples of both Tables with NULLs

Outer Join:

Outer Join =
Inner Join + Dangling Tuples with NULLs

- Conditional Join
- Natural Join (by Default, if no Condition is Given)

Outer Join :

① Left outer Join :


,

,


$R \bowtie S \equiv R \text{ LOJ } S$

≡ first Do $R \triangleright S$ then add Dangling Tuples of R, using Null values remaining Column of S

Outer Join :

(2) Right outer Join : $\overset{\circ}{\bowtie}$, $\bowtie$, $\overset{\circ}{\bowtie}_R$

$R \overset{\circ}{\bowtie} S \equiv R \overset{\circ}{\bowtie}_R S$

$\equiv$ first do $R \overset{\circ}{\bowtie} S$ then add Dangling Tuples of S using Null Values of Remaining Column of R

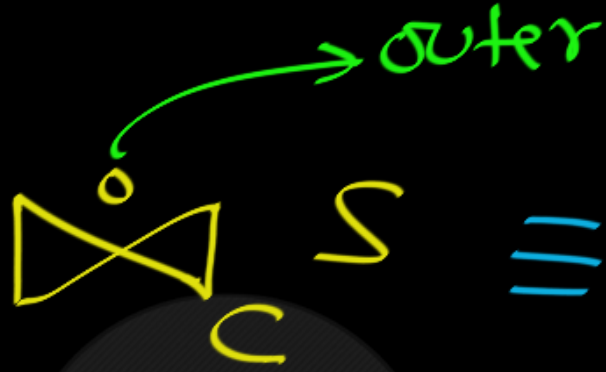
Outer Join : By Default; Outer Join ≡ FOJ

(3) Full outer Join : ⟗ , ⟕

$R \text{ ⟗ } S \equiv R \text{ FOJ } S$

≡ first do ⋈, add Dangling tuples of both Tables using Null Values of Remaining Columns.

Note :



$\equiv$ first do $R \bowtie_C S$ then add Dangling Tuples of both tables using Null values.

A	B	C
1	2	3
4	5	6
7	8	9

(a) Relation U

B	C	D
2	3	10
2	3	11
6	7	12

(b) Relation V

first

$U \bowtie V$

then

add

Dangling

Tuples

of both

Tables

A	B	C	D
1	2	3	10
1	2	3	11
4	5	6	⊥ → Null
7	8	9	⊥
⊥	6	7	12

(c) Result $U ⟗ V \equiv$ $U \text{ FOJ } V \equiv U \bowtie U$

Outer Join

Figure 5.6: Outerjoin of relations

There are many variants of the basic (natural) outerjoin idea. The *left outerjoin* $R \overset{\circ}{\bowtie}_L S$ is like the outerjoin, but only dangling tuples of the left argument R are padded with $\perp$ and added to the result. The *right outerjoin* $R \overset{\circ}{\bowtie}_R S$ is like the outerjoin, but only the dangling tuples of the right argument S are padded with $\perp$ and added to the result.

Example 5.14: If U and V are as in Fig. 5.6, then $U \overset{\circ}{\bowtie}_L V$ is:

A	B	C
1	2	3
4	5	6
7	8	9

(a) Relation U

B	C	D
2	3	10
2	3	11
6	7	12

(b) Relation V

A	B	C	D
1	2	3	10
1	2	3	11
4	5	6	$\perp$
7	8	9	$\perp$

Outer

Right

Null

LOJ

Dangling Tuples of Left Table

and $U \overset{\circ}{\bowtie}_R V$ is:

ROJ

A	B	C	D
1	2	3	10
1	2	3	11
$\perp$	6	7	12

Null

Dangling Tuple of Right Table

A	B	C
1	2	3
4	5	6
7	8	9

(a) Relation U

B	C	D
2	3	10
2	3	11
6	7	12

(b) Relation V

$$U \overset{\circ}{\bowtie}_{A>V.C} V$$

A	B	C
1	2	3
4	5	6
7	8	9

(a) Relation U

B	C	D
2	3	10
2	3	11
6	7	12

(b) Relation V

$$U \overset{\circ}{\bowtie}_{A>V.C} V$$

Outer

Condition

= first do inner Conditional Join ; then add Dangling Tuples of both Tables with Nulls

A	B	C
1	2	3
4	5	6
7	8	9

(a) Relation U

B	C	D
2	3	10
2	3	11
6	7	12

(b) Relation V

$$U \bowtie_{A>V.C} V$$

$U \bowtie_{A>V.C} V$

A	U.B	U.C	V.B	V.C	D
4	5	6	2	3	10
4	5	6	2	3	11
7	8	9	2	3	10
7	8	9	2	3	11

A	B	C
1	2	3
4	5	6
7	8	9

(a) Relation U

B	C	D
2	3	10
2	3	11
6	7	12

(b) Relation V

$$U \overset{\circ}{\bowtie}_{A>V.C} V$$

① $= U \bowtie_{A>V.C} V$ ② add Dangling Tuples

A	U.B	U.C	V.B	V.C	D
4	5	6	2	3	10
4	5	6	2	3	11
7	8	9	2	3	10
7	8	9	2	3	11
1	2	3	Null	Null	Null
Null	Null	Null	6	7	12

A	B	C
1	2	3
4	5	6
7	8	9

(a) Relation U

B	C	D
2	3	10
2	3	11
6	7	12

(b) Relation V

$$U \overset{\circ}{\bowtie}_{A>V.C} V$$

A	$U.B$	$U.C$	$V.B$	$V.C$	D
4	5	6	2	3	10
4	5	6	2	3	11
7	8	9	2	3	10
7	8	9	2	3	11
1	2	3	$\perp$	$\perp$	$\perp$
$\perp$	$\perp$	$\perp$	6	7	12

Figure 5.7: Result of a theta-outerjoin

Left Outer Join

R=

name	phone
A	B
C	D

S=

name	email
A	F
G	H

- If we do a left outer join on R and S, and we match on the first column, the result is:

name	phone	email
A	B	F
C	D	-

Right Outer Join

R=

name	phone
A	B
C	D

S=

name	email
A	F
G	H

- If we do a right outer join on R and S, and we match on the first column, the result is:

name	phone	email
A	B	F
G	-	H

Full Outer Join

R=

name	phone
A	B
C	D

S=

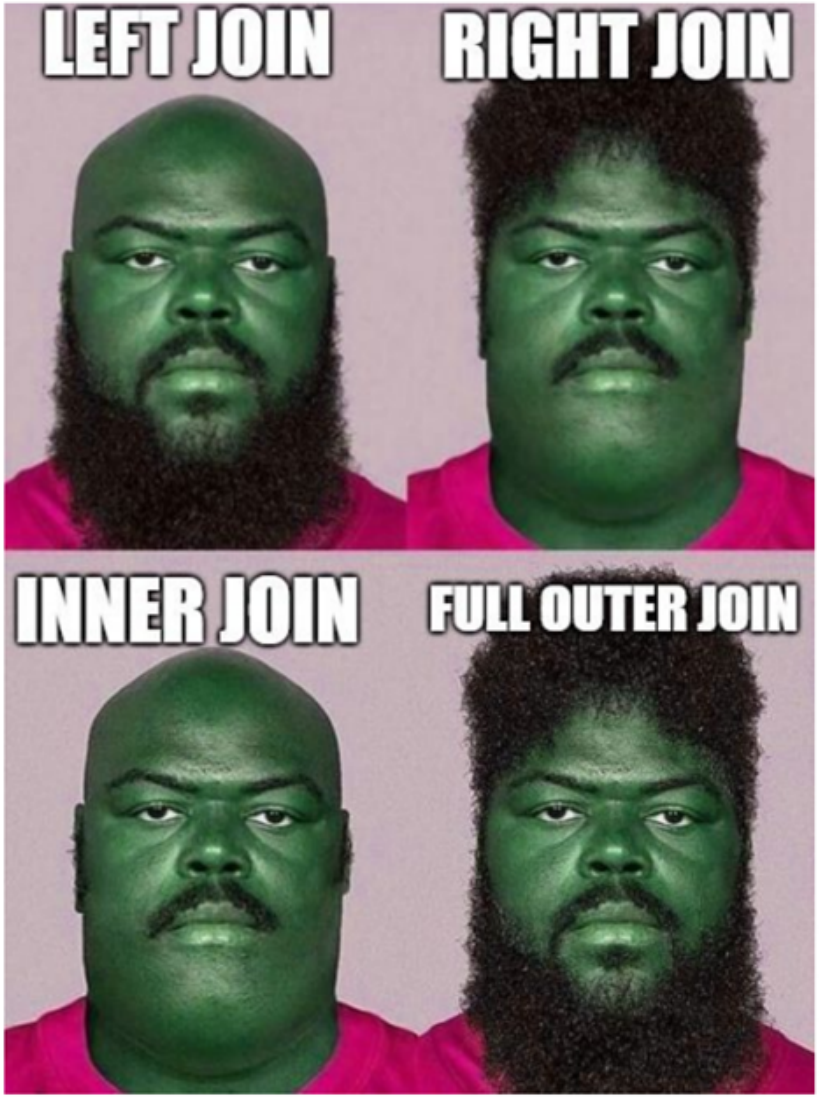
name	email
A	F
G	H

- If we do a full outer join on R and S, and we match on the first column, the result is:

name	phone	email
A	B	F
C	D	-
G	-	H

Join operations – The interpretation

LoL



Cartesian product / Cross join

Notation: $R_1 \times R_2$, where R_1, R_2 are relational algebra expressions.

Description: Returns the Cartesian product of two relations, thereby producing a relation with attributes $\mathcal{A}(R_1) \cup \mathcal{A}(R_2)$ and $\mathcal{T}(R_1) * \mathcal{T}(R_2)$ number of tuples.

Join operations – The interpretation

LOL

Outer join

Motivation

- Suppose we join $R \bowtie S$.
- A tuple of R which doesn't join with any tuple of S is said to be *dangling*.
  - Similarly for a tuple of S.
  - **Problem**: We loose dangling tuples.

Outerjoin

- Preserves dangling tuples by padding them with a special **NULL** symbol in the result.

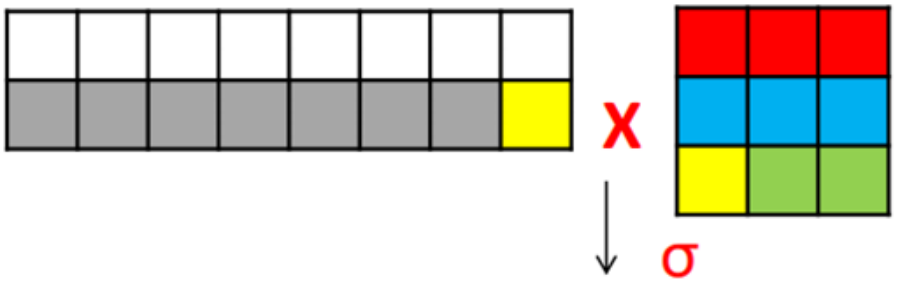
Types of outer join

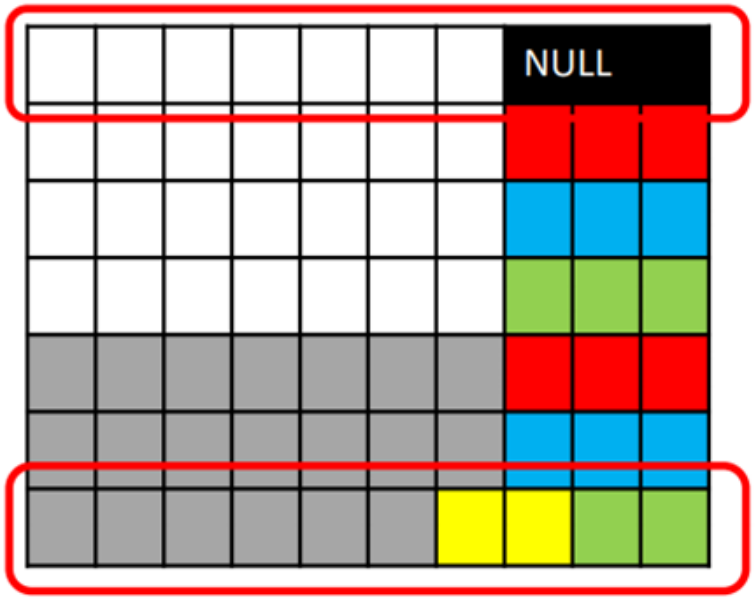
- $R \overset{\times}{\bowtie}_C S$ – This is the **full outerjoin**: Pad dangling tuples from both tables.
- $R \overset{\times}{\triangleright}_C S$ – **left outerjoin**: Only pad dangling tuples from the left table.
- $R \overset{\times}{\triangleleft}_C S$ – **right outerjoin**: Only pad dangling tuples from the right table.

Left outer join

1. For each tuple in R, include all tuples in S which satisfy join condition, but include also tuples of R that do not have matches in S

2. For this case, pair tuples of R with NULL





$T = R \bowtie_{condition} S$

Outer join: example

Anonymous patient P

age	zip	disease
54	99999	heart
20	44444	flue
33	66666	lung

Anonymous occupation O

age	zip	job
54	99999	lawyer
20	44444	cashier

$T = P \bowtie O$

age	zip	disease	job
54	99999	heart	lawyer
20	44444	flue	cashier
33	66666	lung	**NULL**

Outer Join

- An extension of the join operation that avoids loss of information.
- Computes the join and then adds tuples form one relation that does not match tuples in the other relation to the result of the join.
- Uses *null* values:
  - *null* signifies that the value is unknown or does not exist

Natural Join

- Given the schemas R(A, B, C, D), S(A, C, E), what is the schema of R ⋈ S ?
- Given R(A, B, C), S(D, E), what is R ⋈ S ?
- Given R(A, B), S(A, B), what is R ⋈ S ?

Natural Join

- Given the schemas R(A, B, C, D), S(A, C, E), what is the schema of R ⋈ S ? (A, B, C, D, E)

- Given R(A, B, C), S(D, E), what is R ⋈ S ?

  R ⋈ S ≡ R x S

  (A, B, C, D, E)

- Given R(A, B), S(A, B), what is R ⋈ S ?

  R ⋈ S (A B)

φ : $R(A, B) \bowtie S(A, B) = ?$

$R(A, B) \bowtie S(A, B) \equiv R \cap S$

Equality of all Common attributes

$R.A = S.A$ &
$R.B = S.B$

Question 4 (3 points) ✓ *Saved*

If we have two relations, A and B, A RIGHT OUTER JOIN B ON A.id = B.id returns:

- All rows in A and matching rows in B, where there is no match in B null values will appear
- All rows in B and matching rows in A, where there is no match in A, null values will appear
- All rows in both tables, where there is no match in a relation null values will appear
- None of these
- Only matching rows in A and B

Question 4 (3 points) ✓ *Saved*

If we have two relations, A and B, A RIGHT OUTER JOIN B ON A.id = B.id returns:

- All rows in A and matching rows in B, where there is no match in B null values will appear
- All rows in B and matching rows in A, where there is no match in A, null values will appear
- All rows in both tables, where there is no match in a relation null values will appear
- None of these
- Only matching rows in A and B

R ⟖ S

↳ All Tuples of S Come in O/P.

Next Topic:

Division Operation

Relational Algebra

- Procedural language
- Queries in relational algebra are applied to relation instances, result of a query is again a relation instance
- Six basic operators in relational algebra:

select	σ	selects a subset of tuples from reln
project	π	deletes unwanted columns from reln
Cartesian Product	$\times$	allows to combine two relations
Set-difference	$-$	tuples in reln. 1, but not in reln. 2
Union	$\cup$	tuples in reln 1 plus tuples in reln 2
Rename	ρ	renames attribute(s) and relation

- The operators take one or two relations as input and give a new relation as a result (relational algebra is "closed").

Additional Operators (Derived operators)

These operators do not add any power (expressiveness) to the relational algebra but simplify common (often complex and lengthy) queries.

Set-Intersection	$\cap$	
Natural Join	$\bowtie$	
Condition Join	$\bowtie_C$	(also called Theta-Join)
Division	$\div$	

Syntax:

$R(A\ B) \div S(B) = Q(A)$

- $R(A\ B)$ → Dividend
- $S(B)$ → Divisor
- $Q(A)$ → Quotient

Containing those values of R.A which with Every value of S.B, in R.

Syntax: $R/S \equiv R \div S$

$R(X) \div S(Y)$

X must be proper superset of Y.

$X \supset Y$ or $Y \subset X$

Valid

- $R(AB) \div S(A)$
- $R(AB) \div S(B)$
- $R(ABCD) \div S(BC)$
- $R(ABCD) \div S(ACD)$

Invalid

$R(A) \div S(A)$

$R(AB) \div S(BC)$

$R(AB) \div S(CD)$

$r_1 \div r_2$

$r_1(a\ b) / r_2(b)$

Valid

a	b
1	1
1	4
2	1
2	2
2	3
2	4
3	1
3	3
3	4

(a) r_1 (dividend)

b
1
3

(b) r_2 (divisor)

$r_1(ab) \div r_2(b) = Q(a)$

a	b
1	1
1	4
2	1
2	2
2	3
2	4
3	1
3	3
3	4

(a) r_1 (dividend)

Q

a
1 ✗
2 ✓
3 ✓

⇒ Q

a
2
3

✓

b
1
3

(b) r_2 (divisor)

$r_1(ab) / r_2(b) = r_3(a)$

a	b
1	1
1	4
2	1
2	2
2	3
2	4
3	1
3	3
3	4

(a) r_1 (dividend)

b
1
3

(b) r_2 (divisor)

a
2
3

(c) r_3 (quotient)

Figure 1. Division: $r_1 \div r_2 = r_3$

Note :

$R(A\ B) / S(B) = Q(A)$

$Q(A)$ = those values of R.A which Pair with Every Value of S.B

DIVISION Operation

- denoted by $\div$
- useful for retrieval requests that contain a universal quantifier ($\forall$)
- binary operation, defined as follows:

  $R(Z) \div S(X) = T(Y)$ where $X \subset Z$ and where $Y = Z - X$ (Y denotes the set of attributes of R that are not attributes of S)

  a tuple t appears in the result $T(Y)$ if the Y value(s) of t appears in R combined with **every** X value(s) of S.

A

sno	*pno*
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B1

pno
p2

A/B1

B2

pno
p2
p4

A/B2

B3

pno
p1
p2
p4

A/B3

Figure 4.14 Examples Illustrating Division

A

sno	*pno*
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B1

pno
p2

A/B1

$A(Sno, Pno) \div B_1(Pno)$

Valid

$= \varphi(Sno)$

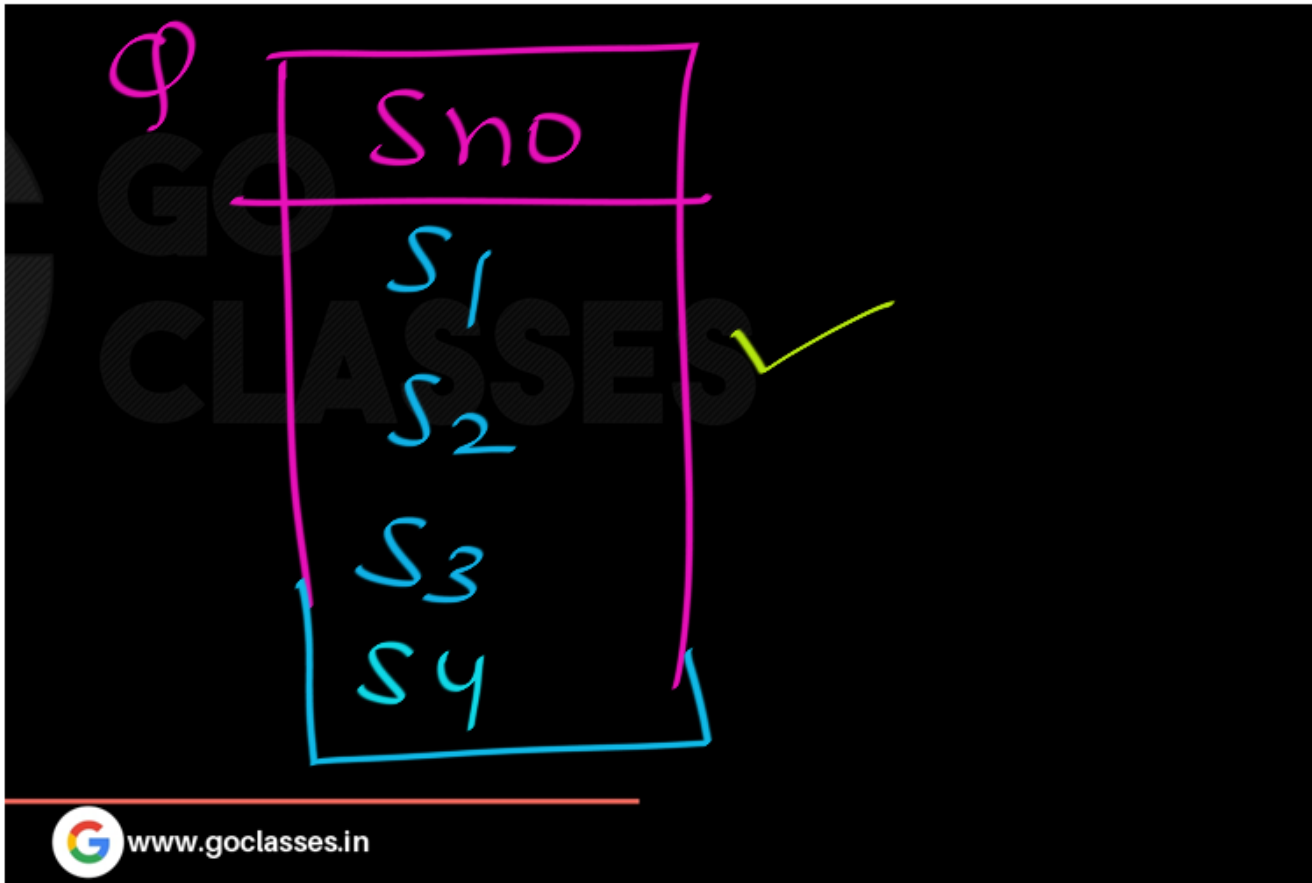
A

sno	*pno*
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B1

pno
p2

A/B1



A

sno	*pno*
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B2

pno
p2
p4

A/B2

φ

Sno
s_1
s_4

A

sno	*pno*
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B3

pno
p1
p2
p4

A/B3

q

Sno
S1

A

sno	*pno*
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B1

pno
p2

B2

pno
p2
p4

B3

pno
p1
p2
p4

A/B1

sno
s1
s2
s3
s4

A/B2

sno
s1
s4

A/B3

sno
s1

Figure 4.14 Examples Illustrating Division

Note: $R(A\ B) \div S(A) = Q(B)$

Those R.B values which Pair with ALL S.A Values, in R.

We discuss division through an example. Consider two relation instances A and B in which A has (exactly) two fields x and y and B has just one field y, with the same domain as in A. We define the *division* operation A/B as the set of all x values (in the form of unary tuples) such that for *every* y value in (a tuple of) B, there is a tuple $\langle x,y \rangle$ in A.

Another way to understand division is as follows. For each x value in (the first column of) A, consider the set of y values that appear in (the second field of) tuples of A with that x value. If this set contains (all y values in) B, the x value is in the result of A/B.

R

A	B
a1	b1
a2	b1
a3	b1
a4	b1
a1	b2
a3	b2
a2	b3
a3	b3
a4	b3
a1	b4
a2	b4
a3	b4

S

A
a1
a2
a3

T

B
b1
b4

R / S ?

R / T ?

R

A	B
a1	b1
a2	b1
a3	b1
a4	b1
a1	b2
a3	b2
a2	b3
a3	b3
a4	b3
a1	b4
a2	b4
a3	b4

S

A
a1
a2
a3

$R(A\,B) / S(A) = Q(B)$

Q

B
b_1 ✓
b_2 ✗
b_3 ✗
b_4 ✓

$\Rightarrow$ Q

B
b_1
b_4

R

A	B
a1	b1
a2	b1
a3	b1
a4	b1
a1	b2
a3	b2
a2	b3
a3	b3
a4	b3
a1	b4
a2	b4
a3	b4

T

B
b1
b4

$R(AB) \div T(B) = Q(A)$

Q

A
a_1
a_2
a_3

--- Division Operations: Example 2

A	B	C	D	E
α	a	α	a	1
α	a	γ	a	1
α	a	γ	b	1
β	a	γ	a	1
β	a	γ	b	3
γ	a	γ	a	1
γ	a	γ	b	1
γ	a	β	b	1

r

D	E
a	1
b	1

S

r ÷ S ⟶

--- Division Operations: Example 2

$r(ABCDE) \div s(D,E) = q(ABC)$

A	B	C	D	E
α	a	α	a	1
α	a	γ	a	1
α	a	γ	b	1
β	a	γ	a	1
β	a	γ	b	3
γ	a	γ	a	1
γ	a	γ	b	1
γ	a	β	b	1

r

D	E
a	1
b	1

S

r ÷ S ⟶

(A, B, C)

q

A	B	C
α	a	γ
γ	a	γ

--- Division Operations: Example 2

A	B	C	D	E
α	a	α	a	1
α	a	γ	a	1
α	a	γ	b	1
β	a	γ	a	1
β	a	γ	b	3
γ	a	γ	a	1
γ	a	γ	b	1
γ	a	β	b	1

r

D	E
a	1
b	1

S

r ÷ S ⟶

A	B	C
α	a	γ
γ	a	γ

a	b
1	1
1	4
2	1
2	2
2	3
2	4
3	1
3	3
3	4

(a) r_1 (dividend)

$r_1(ab) \div r_2(bc)$

Invalid ✓

b	c
1	1
2	1
4	1
1	2
3	2

(b) r_2 (divisor)

HW

Let R(A,B,C) be the relation below:

A	B	C
0	0	2
0	0	3
0	1	2
0	1	3
0	1	4
1	0	3
1	1	2
1	1	4

There are three other relations S1(C), S2(C), and S3(C). S1(C) has only the tuple (2); that is, S1(C) = {(2)}. S2(C) has the tuples (2) and (3); that is, S2(C) = {(2), (3)}.
Likewise, S3(C) = {(2), (3), (4)}.

Compute the results of dividing R by each of S1, S2, and S3: R÷S1, R÷S2, and R÷S3. Then, identify the true statement from the list below. Note, each tuple should be interpreted as a tuple over attributes A and B, in that order.

A. (0,1) is in R÷S3.

B. (0,0) is not in R÷S2.

C. (1,0) is in R÷S3.

D. (1,1) is in R÷S2.

Consider a database that has the relation schema CR(StudentName, CourseName). An instance of the schema CR is as given below.

StudentName	CourseName
SA	CA
SA	CB
SA	CC
SB	CB
SB	CC
SC	CA
SC	CB
SC	CC
SD	CA
SD	CB
SD	CC
SD	CD
SE	CD
SE	CA
SE	CB
SF	CA
SF	CB
SF	CC

The following query is made on the database.

- $T1 \leftarrow \pi_{CourseName}\ (\sigma_{StudentName=SA}\ (CR))$
- $T2 \leftarrow CR \div T1$

The number of rows in $T2$ is _____ .

The following query is made on the database.

(1) • $T1 \leftarrow \pi_{CourseName}\ (\sigma_{StudentName=SA}\ (CR))$

(2) • $T2 \leftarrow CR \div T1$

The number of rows in $T2$ is _____ 4 ✓ .

$CR(SN, CN) \div T_1(CN) = T_2(SN)$

Result: Those Student Names who have taken Every Course taken by SA.

Consider a database that has the relation schema CR(StudentName, CourseName). An instance of the schema CR is as given below.

CR

StudentName	CourseName
SA	CA
SA	CB
SA	CC
SB	CB
SB	CC
SC	CA
SC	CB
SC	CC
SD	CA
SD	CB
SD	CC
SD	CD
SE	CD
SE	CA
SE	CB
SF	CA
SF	CB
SF	CC

T_1:

$\pi_{CN}(\sigma_{SN=SA}(CR))$

T_1:

Courses taken by student SA.

T_1

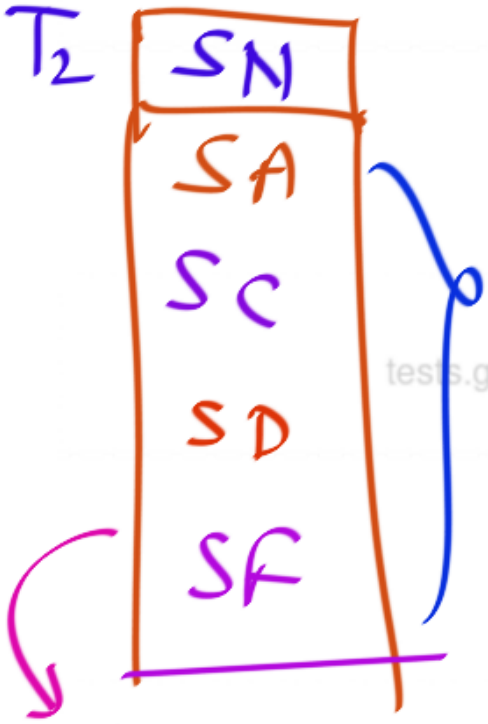
CourseName
CA
CB
CC

Consider a database that has the relation schema CR(StudentName, CourseName). An instance of the schema CR is as given below.

StudentName	CourseName
SA	CA
SA	CB
SA	CC
SB	CB
SB	CC
SC	CA
SC	CB
SC	CC
SD	CA
SD	CB
SD	CC
SD	CD
SE	CD
SE	CA
SE	CB
SF	CA
SF	CB
SF	CC

T_2 :

$CR \div T_1$



those students who took all Courses taken by SA

Next Sub-topic *(Important)*:

Division Operation In terms of Basic Operators

R(Kid , Game) ; S(Game)

R	
a	f
a	c
a	J
a	T
b	c

S
f
c
J

R/S = Q(kid) : those kids who have played every S.game.

Implementation of Division using Basic Operators:

Idea: $R(A\,B) \div S(B) = Q(A)$

Those R.A values which Pair with Every S.B values in R.

(All R.A values) − (Disqualified R.A values)

those R.A values which do not Pair with some S.B value.

$R(A,B) \div S(B)$

Disqualified R.A values :

$$\pi_A(\text{IDEALLY} - \text{Actual})$$

Those R.A values which ideally should have been there, but not Actually there $\Rightarrow$ So; Disqualified R.A values.

IDEA: $R(AB) \div S(B) = Q(A)$

Qualified R.A values = All R.A values − Disqualified R.A values

Disqualified R.A values =

π_A (IDEALLY − Actual)

A

sno	*pno*
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B2

pno
p2
p4

A/B2

$A(Sno, Pno) \div B_2(Pno)$

$= Q(Sno)$

S1
S4

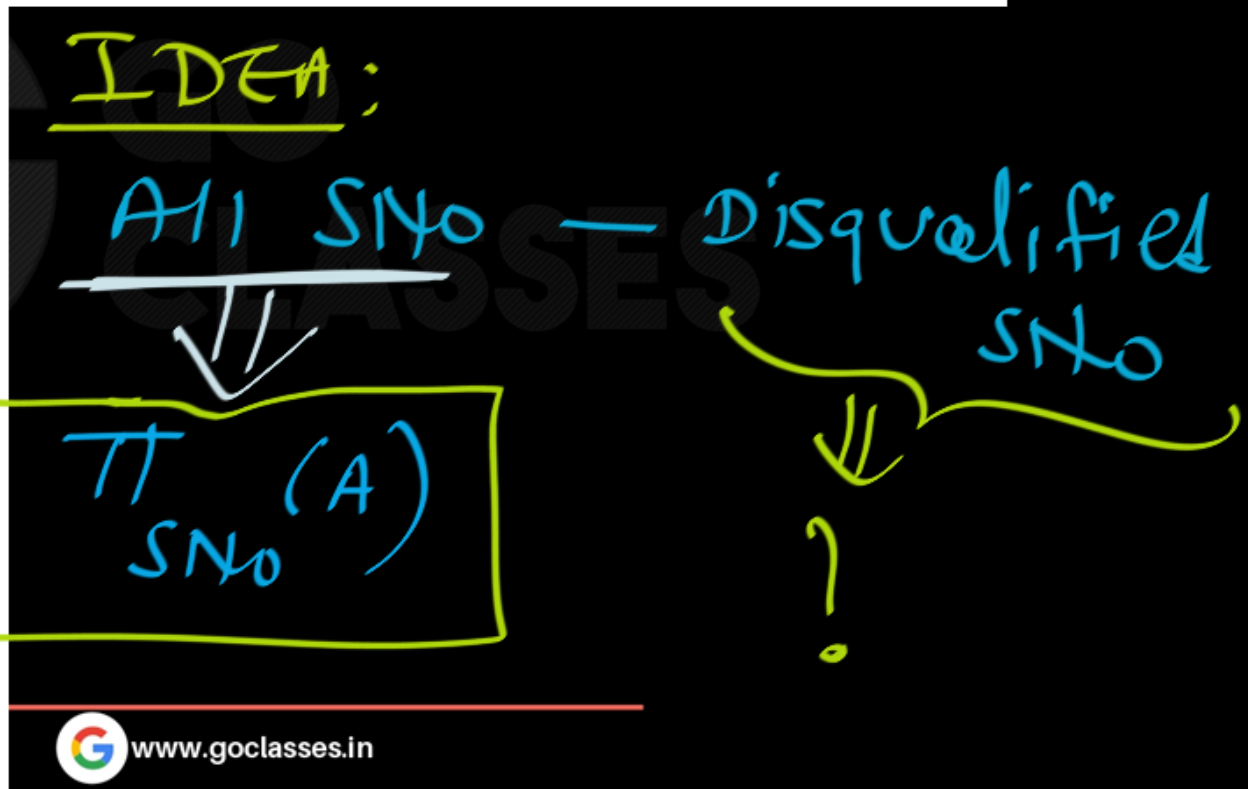
A

sno	*pno*
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B2

pno
p2
p4

A/B2



A

sno	*pno*
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B2

pno
p2
p4

A/B2

Disqualified SNo =

Ideally − Actual

$$\pi_{SNo}\left[\left(\pi_{SNo}(A) \times B_2\right) - A\right]$$

A

sno	*pno*
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B2

pno
p2
p4

A/B2

$$A/B_2 = \underline{\text{All SNo}} - \text{Disqualified SNo}$$

$$= \underbrace{\pi_{SNo}(A) - \pi_{SNo}\left[\left(\pi_{SNo}(A) \times B_2\right) - A\right]}$$

Note: $R(AB) \div S(B) = \Phi(A)$

Idea: All A values — Disqualified A values

$$\Pi_A(R) - \Pi_A\left[\left(\Pi_A(R) \times S\right) - R\right]$$

IDEAL Pairs − Actual Pairs

A

sno	*pno*
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B3

pno
p1
p2
p4

A/B3

GO Classes

$A \div B_3 = Q(sno)$

All SNo : $\Pi_{SNo}(A)$

A

sno	*pno*
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B3

pno
p1
p2
p4

A/B3

GO Classes

$A \div B_3 = Q(sno)$

Disqualified sno :

Π_{sno} (IDEAL pairs − Actual pairs)

A

sno	*pno*
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B3

pno
p1
p2
p4

A/B3

$A \div B_3 = Q(sno)$

IDEAL Pairs:

$\underbrace{\Pi_{SNo}(A) \times B_3}$

A

sno	*pno*
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B3

pno
p1
p2
p4

A/B3

$A \div B_3 = Q(sno)$

Actual Pairs: A

A

sno	*pno*
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B3

pno
p1
p2
p4

A/B3

GO Classes

$A \div B_3 = Q(SNo)$

Disqualified SNo :

$$\pi_{SNo}\left[\left(\pi_{SNo}(A) \times B_3\right) - A\right]$$

A

sno	*pno*
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B3

pno
p1
p2
p4

A/B3

GO Classes

$A \div B_3 = Q(SNo)$

$= \text{All } SNo - \text{Disqualified } SNo$

$= \pi_{SNo}(A) - \pi_{SNo}\left[\left(\pi_{SNo}(A) \times B\right) - A\right]$

--- Division Operations: Example 2

A	B	C	D	E
α	a	α	a	1
α	a	γ	a	1
α	a	γ	b	1
β	a	γ	a	1
β	a	γ	b	3
γ	a	γ	a	1
γ	a	γ	b	1
γ	a	β	b	1

r

D	E
a	1
b	1

S

r ÷ S ⟶

A	B	C
α	a	γ
γ	a	γ

$r(ABCDE) \div s(DE) = Q(A,B,C)$

= All ABC values − Disqualifies ABC values

$= \Pi_{ABC}(r) - \Pi_{ABC}\left[\left(\Pi_{ABC}(r) \times s\right) - r\right]$

Conclusion: $R(AB) \div S(B) = \Phi(A)$

$R \div S =$

$$\underbrace{\pi_A(R)}_{\text{All } R.A \text{ values}} - \underbrace{\pi_A\left[\left(\pi_A(R) \times S\right) - R\right]}_{\text{Disqualified } R.A \text{ values}}$$

Conclusion: $R(AB) \div S(B) = \Phi(A)$

Those R.A Values which Do Not pair with Some S.B Values in R.

$\in \{ \pi_A [(\pi_A(R) \times S) - R] \}$

Conclusion: $R \div S$ **valid** (Schema(R) ⊃ Schema(S))

$$R \div S = \Phi_{(R-S)}$$

$$= \Pi_{R-S}(r) - \Pi_{R-S}\left[\left(\Pi_{R-S}(r) \times S\right) - r\right]$$

$R-S$ → Schema

r → instance

Generalized Implementation of Division using Basic Operators:

DEFINITION 2 (HEALY'S DIVISION): $r_1 \div r_2 = \pi_A(r_1) - \pi_A((\pi_A(r_1) \times r_2) - r_1)$

We discuss division through an example. Consider two relation instances A and B in which A has (exactly) two fields x and y and B has just one field y, with the same domain as in A. We define the *division* operation A/B as the set of all x values (in the form of unary tuples) such that for *every* y value in (a tuple of) B, there is a tuple $\langle x,y \rangle$ in A.

Another way to understand division is as follows. For each x value in (the first column of) A, consider the set of y values that appear in (the second field of) tuples of A with that x value. If this set contains (all y values in) B, the x value is in the result of A/B.

Expressing A/B in terms of the basic algebra operators is an interesting exercise, and the reader should try to do this before reading further. The basic idea is to compute all x values in A that are not *disqualified*. An x value is *disqualified* if by attaching a

y value from B, we obtain a tuple $\langle x,y \rangle$ that is not in A. We can compute disqualified tuples using the algebra expression

$$\pi_x((\pi_x(A) \times B) - A)$$

Thus we can define A/B as

$$\pi_x(A) - \pi_x((\pi_x(A) \times B) - A)$$

5.13.10 Relational Calculus: GATE CSE 2009 | Question: 45

Let R and S be relational schemes such that $R = \{a, b, c\}$ and $S = \{c\}$. Now consider the following queries on the database:

1. $\pi_{R-S}(r) - \pi_{R-S}\left(\pi_{R-S}(r) \times s - \pi_{R-S,S}(r)\right)$

$R \div S$

$R \div S = \Phi(R-S)$

Example: **DIVISION** (1st abstract example)

$R \div S = T$

Dividend

R	A	B
	a_1	b_1
	a_2	b_1
	a_3	b_1
	a_4	b_1
	a_1	b_2
	a_3	b_2
	a_2	b_3
	a_3	b_3
	a_4	b_3
	a_1	b_4
	a_2	b_4
	a_3	b_4

Divisor

S	A
	a_1
	a_2
	a_3

T	B
	b_1
	b_4

Quotient

HW

Which tuple(s) must we add to R so that b_2 belongs to the result T?

Which tuple(s) must we take out from R so that b_4 does not belong to the result T?

(NOT Asked in GATE Yet...)

Another Implementation of Division:

A

sno	*pno*
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B2

pno
p2
p4

A/B2

GO Classes

SNo pairing with P_2
$= \{S_1, S_2, S_3, S_4\} = R_2$

SNo pairing with P_4
$= \{S_1, S_4\} = R_4$

$A/B_2 = R_2 \cap R_4$

A

sno	*pno*
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B2

pno
p2
p4

A/B2

SNo pairing with P_2

∩

SNo pairing with P_4

R

A	B
a1	b1
a2	b1
a3	b1
a4	b1
a1	b2
a3	b2
a2	b3
a3	b3
a4	b3
a1	b4
a2	b4
a3	b4

S

A
a1
a2
a3

$R(AB) \div S(A) = Q(B)$

B values pairing with a_1 $= \{ b_1, b_2, b_4 \}$

B values pairing with a_2 $= \{ b_1, b_3, b_4 \}$

B values pairing with a_3 $= \{b_1, b_2, b_3, b_4\}$

R

A	B
a1	b1
a2	b1
a3	b1
a4	b1
a1	b2
a3	b2
a2	b3
a3	b3
a4	b3
a1	b4
a2	b4
a3	b4

S

A
a1
a2
a3

$R(AB) \div S(A) = Q(B)$

B values pairing with a_1 ∩

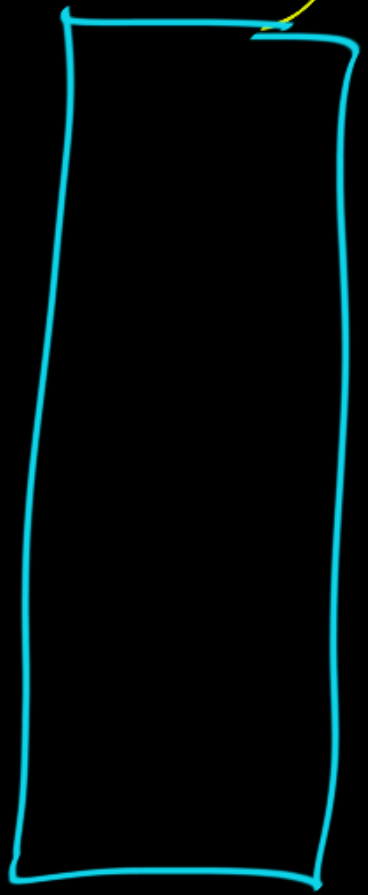
B values pairing with a_2 ∩

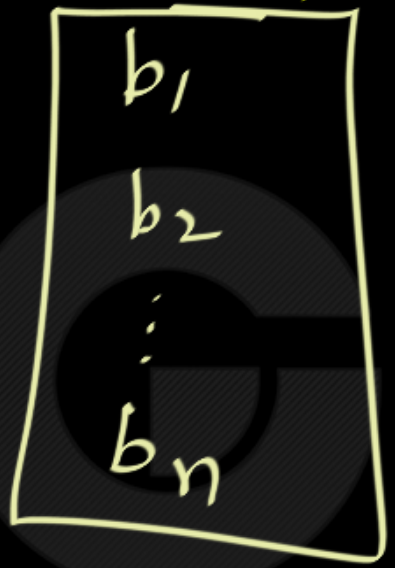
B values pairing with a_3

$R \div S$

$= \{b_1, b_4\}$

$R(A\ B) \div S(B) = Q(A)$





All A values pairing with b_1

∩

All A values pairing with b_2

∩

All A values pairing with b_3

∩

All A values pairing with b_n

$R(A\ B) \div S(B) = Q(A)$

?	b_1
?	b_1
?	b_1
?	b_1

b_1
b_2
⋮
b_n

All A values pairing with b_1

$$= \pi_A\left(\sigma_{B=b_1}(R)\right)$$

$R(A\ B) \div S(B) = Q(A)$

?	b_2
?	b_2
?	b_2
?	b_2

b_1
b_2
⋮
b_n

All A values pairing with b_2

$$= \pi_A\left(\sigma_{B=b_2}(R)\right)$$

$$R \div S = \bigcap_{b_i \in B} \pi_A \left(\sigma_{B = b_i} (R) \right)$$

Conclusion : $R(A\,B) \div S(B) = Q(A)$

$$= \bigcap_{b_i \in B} \left[\Pi_{A} \left(\sigma_{B = b_i} (R) \right) \right]$$

→ Done for every $b_i \in S$

→ Then take intersection of all

Conclusion : $R(A\,B) \div S(B) = Q(A)$

$$= \bigcap_{b_i \in B} \Pi_{A} \left(\sigma_{B = b_i} (R) \right)$$

DEFINITION 3 (MAIER'S DIVISION): $r_1 \div r_2 = \bigcap_{t \in r_2} \pi_A (\sigma_{B=t} (r_1))$

Max/Min Cardinality of Division:

Q: R(A,B) ; S(B)

Assume both R,S are Non-Empty,

|R| = n ; |S| = m

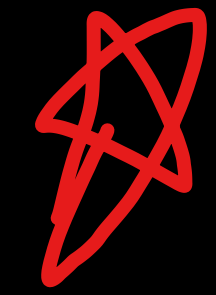
What is minimum & maximum number of tuples in R/S ?

$R(A\ B)$ $S(B)$

both R, S: NonEmpty

min $|R \div S| = 0$

when No A value pairs with all S values.



R (A	B)
a_1	b_1
a_1	b_3
a_2	b_2
a_3	b_3

S (B)
b_1
b_2

$R \div S = Q(A)$

$Q(A)$

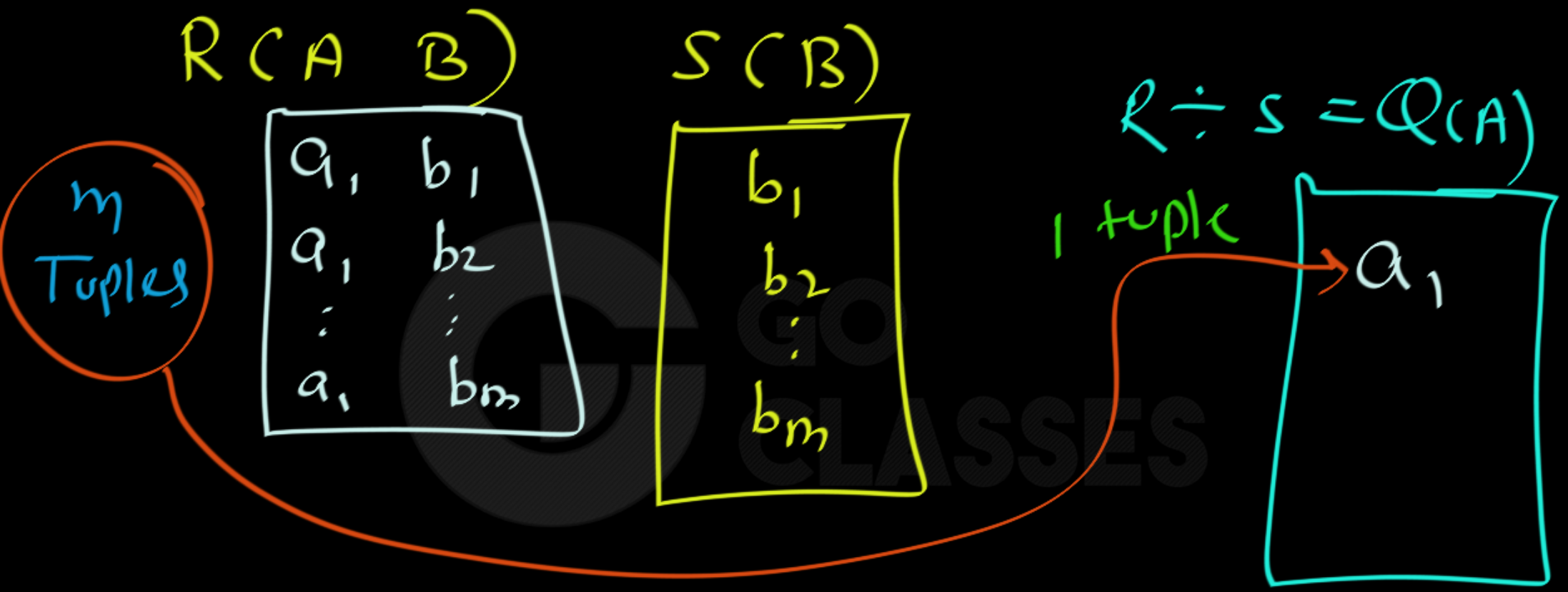
Empty Table

$R(A\ B)$ $S(B)$ both R, S: NonEmpty

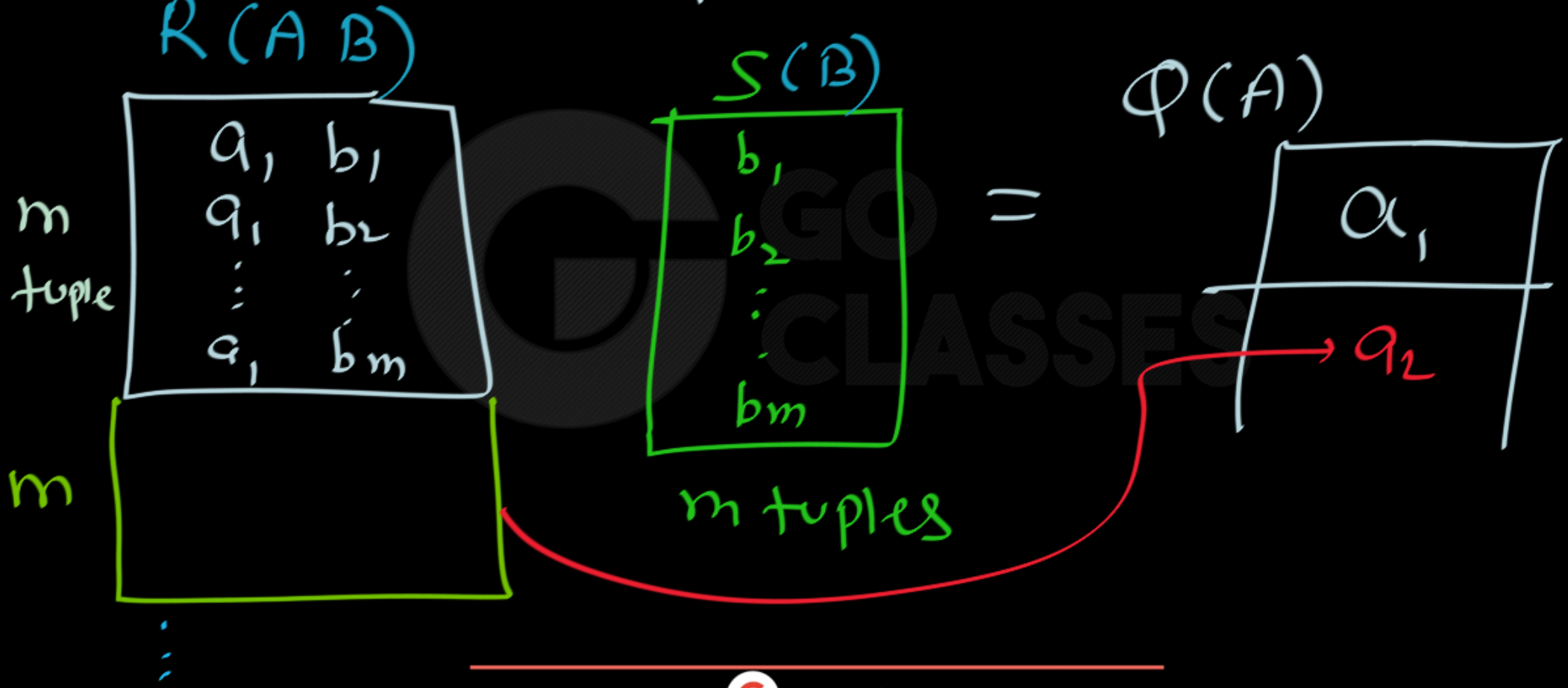
$$\max |R \div S| = \left\lfloor \frac{|R|}{|S|} \right\rfloor$$

If $|R| = n$; $|S| = m$

then $\max |R \div S| = \left\lfloor \frac{n}{m} \right\rfloor$


R(A B)
a1 b1
a1 b2
a1 bm
S(B)
b1
b2
bm
m Tuples
R ÷ S = Q(A)
1 tuple
a1

I want max. (R ÷ S)



To get max $|R \div S|$

$|S| = m$

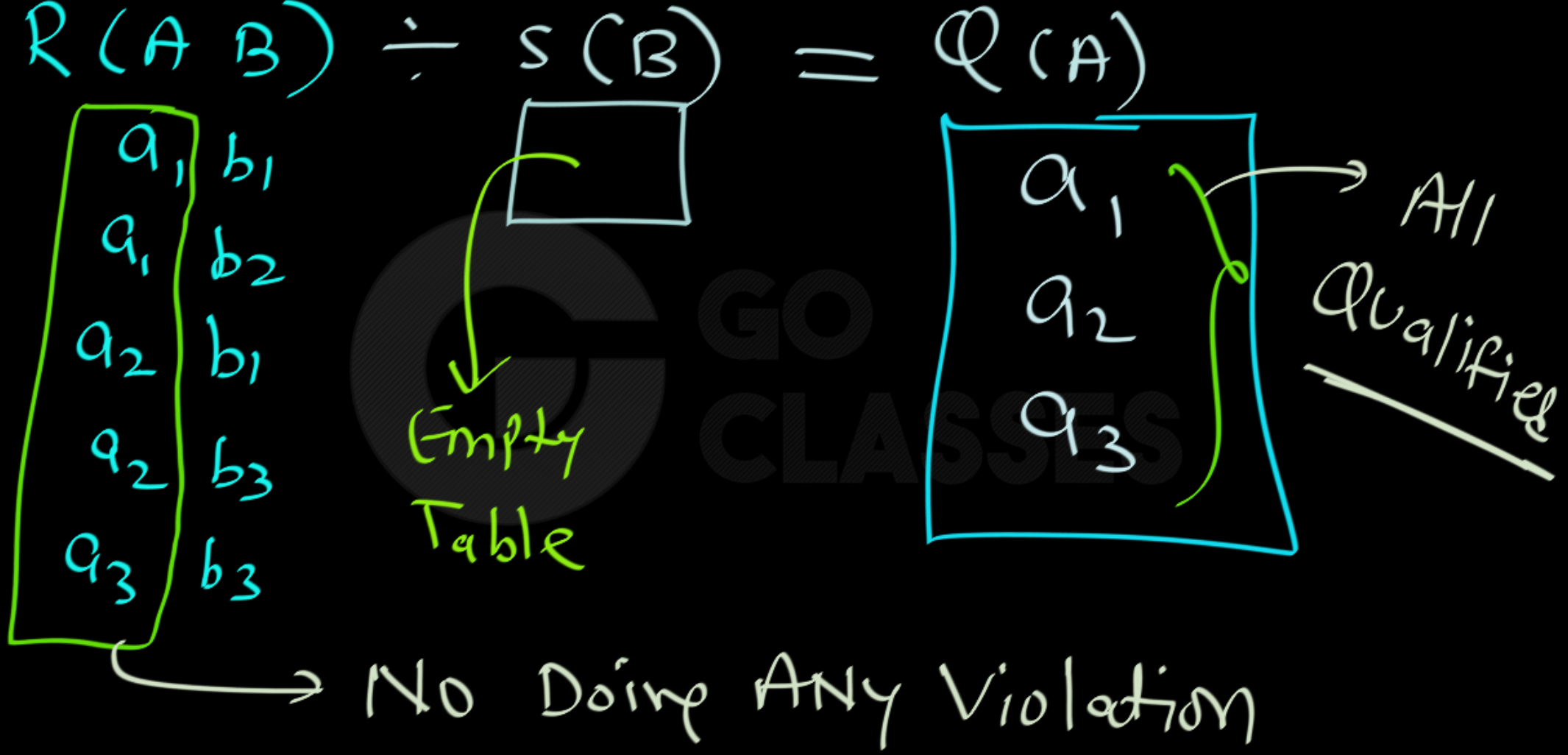
R(AB) S(B)

Q(A)

Q: R(A,B) ; S(B)

Assume S is Empty. |R| = n

What is minimum & maximum number of tuples in R/S ?


R(A B) ÷ S(B) = Q(A)
a1 b1
a1 b2
a2 b1
a2 b3
a3 b3
Empty Table
a1
a2
a3
All Qualified
No Doing ANY Violation

Note :

If S Empty then

$R \times S = \phi$

$|R \times S| = |R| |S|$

If $|S| = 0$ then $|R \times S| = 0$

$R(AB) \div S(B) = Q(A)$

Given; S: Empty

Healy Equation: All A's — Disqualifies A's

$= \pi_A(R) - \pi_A[(\pi_A(R) \times S) - R]$

$= (\pi_A(R))$ Empty Empty Empty

(1) $\phi - R = \phi$

(2) $R - \phi = R$

Note: $R(AB) \div S(B) = Q(A)$

If S is Empty then

$R \div S = \Pi_A(R) =$ All A's of R

$\max |R \div s| = |R|$ if all a's are unique

$\min |R \div s| = 0$ if R empty

Note: $R(AB) \div S(B) = Q(A)$

If R empty then $R \div S$ = Empty

(because there is No A value at all)

Theoretical Definition of Division

(Why it is called "Division")

Mathematical Division :

$18 / 4 = 4$ ⟹ Largest integer such that $4 \times 4 \leq 18$

Quotient

$16 / 8 = 2$ ⟶ Largest int s.t. $8 \times 2 \leq 16$

Mathematical Division :

$7 / 3 = 2$

↓ Quotient

$14 / 3 = 4$

$\frac{m}{n} = q$

q is largest integer such that $nq \leq m$

Division opⁿ in Rel. Algebra:

$R(A\,B) \div S(B) = Q(A)$

Largest Relation

s.t. $Q \times S \subseteq R$

Do you understand the "Division" operation in mathematical algebra??

For integers A and B,
A/B is the largest integer Q such that
$Q * B \leq A$.

$$\frac{A}{B} = Q$$

Division Operation:

For relation instances A and B, A/B is the largest relation instance Q such that $Q \times B \subseteq A$.

Cross Product

Division again

Another point of view:
Analogy with integer division.

$7/3 = 2$ because $2 \cdot 3 \leq 7$

A/B is the maximum integer Q s.t. $Q \cdot B \leq A$

$A \div B$ is the largest relation Q s.t. $Q \times B \subseteq A$

Cross Product

Division Operator

- Given relations r(R) and s(S), such that $S \subset R$, $r \div s$ is the largest relation t(R-S) such that

$$t \times s \subseteq r$$

Q:

For any relations R, S with Disjoint Schema;

What is (RxS)/S?

Q:

For any relations R, S with Disjoint Schema; $R(AB)$; $S(C)$

What is $(R \times S)/S$? $= R$

$(R \times S)$ → (ABC)

S → (C)

Q:

For any relations R, S with Disjoint Schema,

What is $(R \times S)/R$? $= S$

Q: True/False?

For any relations R(A,B), S(B);

If R/S = Q

Then is QxS is subset of R.

Q: True/False?

For any relations R(A,B), S(B);

If R/S = Q

Then is QxS is subset of R. Yes

$$\frac{R(AB)}{S(B)} = Q(A)$$

Q is Largest Relation s.t. $Q \times S \subseteq R$

Q: True/False?

For any relations R(A,B), S(B);

If Y is a relation such that

$Y \times S$ is subset of R

then Y = (R/S) ?

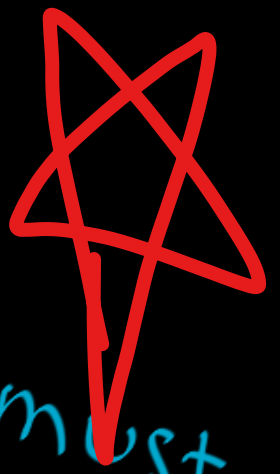
Q: True/False?

For any relations R(A,B), S(B);

If Y is a relation such that

Y × S is subset of R

then Y = (R/S) ? No



Y must be Largest

Q: True/False?

For any relations R(A,B), S(B);

If Y is the largest relation such that Y*S is subset of R

then Y = (R/S) ? True