# CS & IT ENGINEERING


## Database Management System

DBMS

Lecture No. 8

By- Ravindrababu Ravula Sir

# Recap of Previous Lecture

Topic

SQL

# Topics to be Covered

| | |
|---|---|
| Topic | Nested Queries |
| Topic | Function used for Nested Queries |
| Topic | Any && All |
| Topic | EXISTS and NOT EXITS |
| Topic | TRC vs RA vs SQL |

# Topic: Nested Queries

There are two format of nested queries:

1. Nested Query without Correlation.
2. Nested Query With Correlation.

# Topic: Nested Queries

## Nested Query WITHOUT Correlation

```
SELECT eid
FROM  Emp
WHERE sal = (SELECT max(sal)
             FROM Emp));
```

## Nested Query WITH Correlation

```
SELECT *
FROM  R
WHERE  (SELECT count(*)
        FROM S
        WHERE  R.A > S.B) <2;
```

# Topic: Nested Queries

## Nested Query WITHOUT Correlation

top

```
SELECT eid
FROM  Emp
WHERE sal = (SELECT max(sal)
             FROM Emp));
```

Executes first-bottom

## Nested Query WITH Correlation

```
SELECT *
FROM  R
WHERE  (SELECT count(*)
        FROM S
        WHERE  R.A > S.B) <2;
```

# Topic: Nested Queries

## Nested Query WITHOUT Correlation

top

```
SELECT eid
FROM  Emp
WHERE sal = (SELECT max(sal)
             FROM Emp));
```

Executes first-bottom

Select
FROM
WHERE
GROUP BY
HAVING
ORDER BY


## Nested Query WITH Correlation

```
SELECT *
FROM  R
WHERE  (SELECT count(*)
        FROM S
        WHERE  R.A > S.B) <2;
```

# Topic: Nested Queries

## Nested Query WITHOUT Correlation

top

```
SELECT eid
FROM  Emp
WHERE sal = (SELECT max(sal)
             FROM Emp));
```

Executes first-bottom

Inner Query is Independent.
First inner query will be resolved, scalar result of whose will then be used to execute top query.

## Nested Query WITH Correlation

```
SELECT *
FROM  R
WHERE  (SELECT count(*)
        FROM S
        WHERE  R.A > S.B) <2;
```

# Topic: Nested Queries

## Nested Query WITHOUT Correlation

top
```
SELECT eid
FROM  Emp
WHERE sal = (SELECT max(sal)
             FROM Emp));
```

Executes first-bottom

Inner Query is Independent.
First inner query will be resolved, scalar result of whose will then be used to execute top query.

## Nested Query WITH Correlation

```
SELECT *
FROM  R
WHERE  (SELECT count(*)
        FROM S
        WHERE R.A > S.B) <2;
```

Inner Query is Dependent
Inner query uses attributes which are defined in Outer query

# Topic: Nested Queries

## Nested Query WITHOUT Correlation

top

```
SELECT eid
FROM  Emp
WHERE sal = (SELECT max(sal)
             FROM Emp));
```

Executes first-bottom

Bottom Query executes first and then top

## Nested Query WITH Correlation

```
SELECT *
FROM  R
WHERE  (SELECT count(*)
        FROM S
        WHERE  R.A > S.B) <2;
```

Inner query runs for each record of outer query

# Topic: Nested Queries

## Nested Query WITHOUT Correlation

top

```
SELECT eid
FROM  Emp
WHERE sal = (SELECT max(sal)
             FROM Emp));
```

Executes first-bottom

## Nested Query WITH Correlation

```
SELECT *
FROM  R
WHERE  (SELECT count(*)
        FROM S
        WHERE  R.A > S.B) <2;
```

Emp

| eid | Sal |
|---|---|
| 1 | 160 |
| 2 | 140 |
| 4 | 330 |
| 3 | 350 |
| 5 | 410 |

# Topic: Nested Queries

## Nested Query WITHOUT Correlation

top

```
SELECT eid
FROM  Emp
WHERE sal = (SELECT max(sal)
             FROM Emp));
```

Executes first-bottom

410

| Emp | eid | Sal |
|---|---|---|
| | 1 | 160 |
| | 2 | 140 |
| | 4 | 330 |
| | 3 | 350 |
| | 5 | 410 |

## Nested Query WITH Correlation

```
SELECT *
FROM  R
WHERE  (SELECT count(*)
        FROM S
        WHERE  R.A > S.B) <2;
```

# Topic: Nested Queries

## Nested Query WITHOUT Correlation

top
```
SELECT eid
FROM  Emp
WHERE sal = (SELECT max(sal)
             FROM Emp));
```

Executes first-bottom

410

| Emp | eid | Sal |
|---|---|---|
| | 1 | 160 |
| | 2 | 140 |
| | 4 | 330 |
| | 3 | 350 |
| | 5 | 410 |

| eid |
|---|
| 5 |

## Nested Query WITH Correlation

```
SELECT *
FROM  R
WHERE  (SELECT count(*)
        FROM S
        WHERE  R.A > S.B) <2;
```

# Topic: Nested Queries

## Nested Query WITHOUT Correlation

top

```
SELECT eid
FROM  Emp
WHERE sal = (SELECT max(sal)
             FROM Emp));
```

Executes first-bottom

## Nested Query WITH Correlation

```
SELECT *
FROM  R
WHERE  (SELECT count(*)
        FROM S
        WHERE  R.A > S.B) <2;
```

R

| A |
|---|
| 60 |
| 40 |
| 30 |
| 50 |
| 10 |

S

| B |
|---|
| 50 |
| 10 |
| 60 |
| 40 |
| 30 |

60 > S.B

# Topic: Nested Queries

## Nested Query WITHOUT Correlation

top

```
SELECT eid
FROM  Emp
WHERE sal = (SELECT max(sal)
             FROM Emp));
```

Executes first-bottom

## Nested Query WITH Correlation

```
SELECT *
FROM  R
WHERE  (SELECT count(*)
        FROM S
        WHERE  R.A > S.B) <2;
```

4 < 2

| R | A |
|---|---|
| ✗ | 60 |
| | 40 |
| | 30 |
| | 50 |
| | 10 |

| S | B |
|---|---|
| | 50 |
| | 10 |
| | 60 |
| | 40 |
| | 30 |

60 > S.B

Count = 4 ✓

# Topic: Nested Queries

## Nested Query WITHOUT Correlation

top
```
SELECT eid
FROM  Emp
WHERE sal = (SELECT max(sal)
             FROM Emp));
```

Executes first-bottom

## Nested Query WITH Correlation

```
SELECT *
FROM  R
WHERE  (SELECT count(*)
        FROM S
        WHERE  R.A > S.B) <2;
```

R

| A |
|---|
| 60 |
| 40 |
| 30 |
| 50 |
| 10 |

S

| B |
|---|
| 50 |
| 10 |
| 60 |
| 40 |
| 30 |

60 > S.B

Count = 4

# Topic: Nested Queries

## Nested Query WITHOUT Correlation

top
```
SELECT eid
FROM  Emp
WHERE sal = (SELECT max(sal)
             FROM Emp));
```

Executes first-bottom

## Nested Query WITH Correlation

```
SELECT *
FROM  R
WHERE  (SELECT count(*)
        FROM S
        WHERE  R.A > S.B) <2;
```

2 < 2

| R | A |
|---|---|
| ✗ | 60 |
| → | 40 |
| | 30 |
| | 50 |
| | 10 |

| S | B |
|---|---|
| ✗ | 50 |
| ✓ | 10 |
| ✗ | 60 |
| ✗ | 40 |
| ✓ | 30 |

40 > S.B

Count = 2

# Topic: Nested Queries

## Nested Query WITHOUT Correlation

top
```
SELECT eid
FROM  Emp
WHERE sal = (SELECT max(sal)
             FROM Emp));
```

Executes first-bottom

## Nested Query WITH Correlation

```
SELECT *
FROM  R
WHERE  (SELECT count(*)
        FROM S
        WHERE  R.A > S.B) <2;
```

R

| A |
|---|
| 60 |
| 40 |
| 30 |
| 50 |
| 10 |

S

| B |
|---|
| 50 |
| 10 |
| 60 |
| 40 |
| 30 |

40 > S.B

Count = 2

# Topic: Nested Queries

## Nested Query WITHOUT Correlation

top
```
SELECT eid
FROM  Emp
WHERE sal = (SELECT max(sal)
             FROM Emp));
```

Executes first-bottom

## Nested Query WITH Correlation

```
SELECT *
FROM  R
WHERE  (SELECT count(*)
        FROM S
        WHERE  R.A > S.B) <2;
```

1<2

| R | A |
|---|---|
| ✗ | 60 |
| ✗ | 40 |
| → | 30 |
| | 50 |
| | 10 |

| S | B |
|---|---|
| ✗ | 50 |
| ✓ | 10 |
| ✗ | 60 |
| ✗ | 40 |
| ✗ | 30 |

30 > S.B

Count = 1

# Topic: Nested Queries

## Nested Query WITHOUT Correlation

top
```
SELECT eid
FROM  Emp
WHERE sal = (SELECT max(sal)
             FROM Emp));
```

Executes first-bottom

30

## Nested Query WITH Correlation

```
SELECT *
FROM  R
WHERE  (SELECT count(*)
        FROM S
        WHERE  R.A > S.B) <2;
```

| R | A |
|---|---|
| ✗ | 60 |
| ✗ | 40 |
| ✓ ← | 30 |
| | 50 |
| | 10 |

| S | B |
|---|---|
| | 50 |
| | 10 |
| | 60 |
| | 40 |
| | 30 |

30 > S.B

Count = 1

# Topic: Nested Queries

## Nested Query WITHOUT Correlation

top
```
SELECT eid
FROM  Emp
WHERE sal = (SELECT max(sal)
             FROM Emp));
```

Executes first-bottom

## Nested Query WITH Correlation

```
SELECT *
FROM  R
WHERE  (SELECT count(*)
        FROM S
        WHERE  R.A > S.B) <2;
```

3 < 2

| R | A |
|---|---|
| ✗ | 60 |
| ✗ | 40 |
| ✓ | 30 |
| → | 50 |
| | 10 |

| S | B |
|---|---|
| ✗ | 50 |
| ✓ | 10 |
| ✗ | 60 |
| ✓ | 40 |
| ✓ | 30 |

50 > S.B

Count = 3

# Topic: Nested Queries

## Nested Query WITHOUT Correlation

top
```
SELECT eid
FROM  Emp
WHERE sal = (SELECT max(sal)
             FROM Emp));
```

Executes first-bottom

## Nested Query WITH Correlation

```
SELECT *
FROM  R
WHERE  (SELECT count(*)
        FROM S
        WHERE  R.A > S.B) <2;
```

| R | A |
|---|---|
| ✗ | 60 |
| ✗ | 40 |
| ✓ | 30 |
| ✗ | 50 |
| | 10 |

| S | B |
|---|---|
| | 50 |
| | 10 |
| | 60 |
| | 40 |
| | 30 |

50 > S.B

Count = 3

# Topic: Nested Queries

## Nested Query WITHOUT Correlation

top
```
SELECT eid
FROM  Emp
WHERE sal = (SELECT max(sal)
             FROM Emp));
```

Executes first-bottom

## Nested Query WITH Correlation

```
SELECT *
FROM  R
WHERE  (SELECT count(*)
        FROM S
        WHERE  R.A > S.B) <2;
```

0<2

| R | A |
|---|---|
| ✗ | 60 |
| ✗ | 40 |
| ✓ | 30 |
| ✗ | 50 |
| → | 10 |

| S | B |
|---|---|
| ✗ | 50 |
| ✗ | 10 |
| ✗ | 60 |
| ✗ | 40 |
| ✗ | 30 |

10 > S.B

Count = 0

# Topic: Nested Queries

## Nested Query WITHOUT Correlation

top

```
SELECT eid
FROM  Emp
WHERE sal = (SELECT max(sal)
             FROM Emp));
```

Executes first-bottom

## Nested Query WITH Correlation

```
SELECT *
FROM  R
WHERE  (SELECT count(*)
        FROM S
        WHERE  R.A > S.B) <2;
```

| R | A |
|---|---|
| ✗ | 60 |
| ✗ | 40 |
| ✓ | 30 |
| ✗ | 50 |
| ✓ | 10 |

| S | B |
|---|---|
| | 50 |
| | 10 |
| | 60 |
| | 40 |
| | 30 |

10 > S.B

Count = 0

# Topic: Nested Queries

## Nested Query WITHOUT Correlation

top

```
SELECT eid
FROM  Emp
WHERE sal = (SELECT max(sal)
             FROM Emp));
```

Executes first-bottom

## Nested Query WITH Correlation

```
SELECT *
FROM  R
WHERE  (SELECT count(*)
        FROM S
        WHERE  R.A > S.B) <2;
```

If Inner query is written in WHERE clause, then the result of inner query then treated as one constant value for Outer Query WHERE clause

# Topic: Nested Queries

## Nested Query WITHOUT Correlation

top
```
SELECT eid
FROM  Emp
WHERE sal = (SELECT max(sal)
             FROM Emp));
```

Executes first-bottom

FROM ↓ Table

↓ constant

If Inner query is written in WHERE clause, then the result of inner query then treated as one constant value for Outer Query WHERE clause

## Nested Query WITH Correlation

```
SELECT *
FROM  R
WHERE  (SELECT count(*)
        FROM S
        WHERE  R.A > S.B) <2;
```

If Inner query is written in FROM clause the result of inner query is treated as table of Outer Query FROM class.

# Topic: Nested Queries

## Nested Query WITHOUT Correlation

top
```
SELECT eid
FROM  Emp
WHERE sal = (SELECT max(sal)
             FROM Emp));
```

Executes first-bottom

## Nested Query WITH Correlation

```
SELECT *
FROM  R
WHERE  (SELECT count(*)
        FROM S
        WHERE  R.A > S.B) <2;
```

- Execution flow is Top - Bottom - Top.

# Topic: Nested Queries

## Nested Query WITHOUT Correlation

top

```
SELECT eid
FROM  Emp
WHERE sal = (SELECT max(sal)
             FROM Emp));
```

Executes first-bottom

## Nested Query WITH Correlation

```
SELECT *
FROM  R
WHERE  (SELECT count(*)
        FROM S
        WHERE  R.A > S.B) <2;
```

If co-relation is in WHERE Clause Inner Query re-computes for each record of Outer Query.

# Topic: Nested Queries

## Nested Query WITHOUT Correlation

top

```
SELECT eid
FROM  Emp
WHERE sal = (SELECT max(sal)
             FROM Emp));
```

Executes first-bottom

## Nested Query WITH Correlation

```
SELECT *
FROM  R
WHERE  (SELECT count(*)
        FROM S
        WHERE  R.A > S.B) <2;
```

If co-relation is in WHERE Clause Inner Query re-computes for each record of Outer Query.

If correlation is in HAVING Clause, inner Query re-computes for each group of Outer Query.

# Topic: Nested Queries

## Nested Query WITHOUT Correlation

top

```
SELECT eid
FROM  Emp
WHERE sal = (SELECT max(sal)
             FROM Emp));
```

Executes first-bottom

## Nested Query WITH Correlation

```
SELECT *
FROM  R
WHERE  (SELECT count(*)
        FROM S
        WHERE  R.A > S.B) <2;
```

Correlated queries are preferred when we have lot of compilation, but if Queries are simple non-correlated are sufficient.

# Topic: Function used for Nested Queries

Functions used for Nested queries

1. IN / NOT IN
2. ANY
3. ALL
4. EXISTS / NOT EXISTS

# Topic: Function used for Nested Queries

**Example:**

```
Select *
FROM Emp
WHERE Sal > ( Select Sal FROM Emp WHERE dno = 5 )
```

# Topic: Function used for Nested Queries

**Example:**

```
Select *
FROM Emp
WHERE Sal > ( Select Sal FROM Emp WHERE dno = 5 )
```

Here, the inner query will give us a table of data, instead of one scalar value.

The result

| A |
|---|
| 50 |
| 1 |
| 10 |

# Topic: Function used for Nested Queries

**Example:**

```
Select *
FROM Emp
WHERE Sal > ( Select Sal FROM Emp WHERE dno = 5 )
```

Here, the inner query will give us a table of data, instead of one scalar value.

The result

| A |
|---|
| 50 |
| 1 |
| 10 |

Now, the WHERE is expecting a scalar comparison, that is comparing a single value, but we get a database table, this will lead to an ERROR.

# Topic: Function used for Nested Queries

Functions used for Nested queries

1. IN/ NOT IN
2. ANY
3. ALL
4. EXISTS / NOT EXISTS

**Example:**

```
Select *
FROM Emp
WHERE Sal > ( Select MAX(Sal) FROM Emp WHERE dno = 5 )
```

100

Works becoz it returns scalar

# Topic: Function used for Nested Queries

Functions used for Nested queries

1. IN / NOT IN
2. ANY
3. ALL
4. EXISTS / NOT EXISTS

**Example:**

```
Select *
FROM Emp
WHERE Sal = ( Select Sal FROM Emp WHERE dno = 5 )
```

Doesn't Work becoz it returns table

# Topic: Function used for Nested Queries

The > is expecting a single value for comparison. If > receives a set of values then we will get an error.

What if we want to compare set of values?
Use function, instead of using a mathematical comparator like >, <, =, <=

# Topic: Function used for Nested Queries

The > is expecting a single value for comparison. If > receives a set of values then we will get an error.

What if we want to compare set of values?
Use function, instead of using a mathematical comparator like >, <, =, <=

Among
- IN/NOT IN
- ANY
- ALL

→ best suitable for nested query without correlation

- EXISTS / NOT EXISTS

→ best suitable for correlated query

# Topic: Function used for Nested Queries

IN function is used for membership test


If X is there in the set of Y values, then IN Func will return True.

# Topic: Function used for Nested Queries

IN function is used for membership test


If X is there in the set of Y values, then IN Func will return True.

i.e. X ∈ Y Set ⇒ True

else ⇒ False

# Topic: Function used for Nested Queries

Example:

X IN Y

| X |
|---|
| 3 |

| Y |
|---|
| 2 |
| 3 |
| 5 |

$\Rightarrow$ True

$\because$ $3 \in Y$

# Topic: Function used for Nested Queries

Example:

X IN Y

| X |
|---|
| 3 |

| Y |
|---|
| 2 |
| 3 |
| 5 |

$\Rightarrow$ True

$\because 3 \in Y$

| X |
|---|
| 8 |

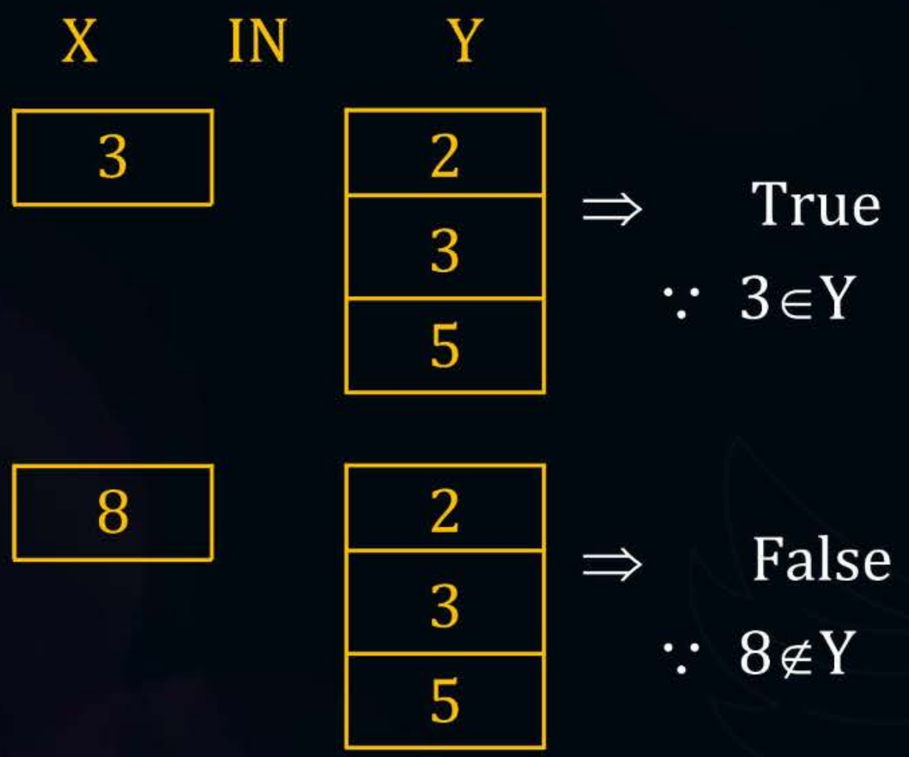
| Y |
|---|
| 2 |
| 3 |
| 5 |

$\Rightarrow$ False

$\because 8 \notin Y$

# Topic: Function used for Nested Queries

Example:




True if X value is member of Y Set

X value must be equal to some (at least one) value of Y set.

# Topic: Function used for Nested Queries

IN Function can be used as Join(⋈) queries

Say: $\Pi_{R.A}\ (R \bowtie_{R.A = S.B} S)$

- Retrieving A of the Relation R, whose A value equals to at least one B value of S.

⇒ A value should be present in B values of S.

# Topic: Function used for Nested Queries

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

**NOTE :** Natural Join is also equivalent to Equi-Join on common attr

Retrieving records of first relation, whose B value is equal to some (at least one) value of B of other relation.

These type of queries can be expressed using IN Function.

$R(A\,B) \quad S(B\,C) \quad \pi_A (R \bowtie S)$

# Topic: Function used for Nested Queries

R(A, B) and S(B, C)

1. $\prod_{B} (R \bowtie S)$

$\pi_B(R \bowtie S)$

# Topic: Function used for Nested Queries

R(A, B) and S(B, C)

1. $\prod_{B} (R \bowtie S)$

→ Select * FROM R
WHERE R.B IN (Select B FROM S)

= some

Retrieving the values of R.B that are equal to some(atleast one) of B in S ,
This is Natural Join

# Topic: Function used for Nested Queries

2. R(A) S(B)

$\prod(R \bowtie S)$

A R.A = S.B

R.A = S.B

we can use IN even if we don't have natural join like here we have with Equi-Join.

Retrieving attribute A values, which are equal to some B of relation S.

$\prod(R \bowtie S)$

A R.A = S.B

→ Select A
FROM R
WHERE A IN (Select B FROM S)

= some

# Topic: Function used for Nested Queries

2. R(A) S(B)

$\prod_{A}(R \underset{R.A = S.B}{\bowtie} S)$

we can use IN even if we don't have natural join like here we have with Equi-Join.

Retrieving attribute A values, which are equal to some B of relation S.

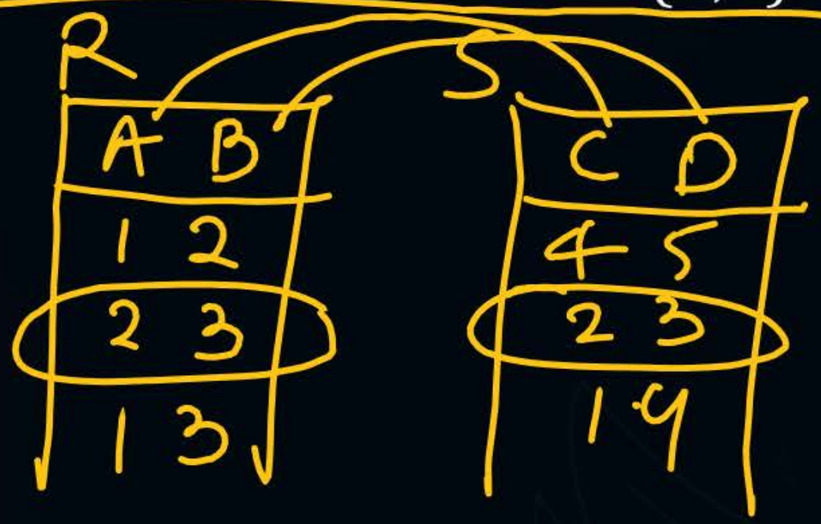
$\prod_{A}(R \underset{R.A = S.B}{\bowtie} S)$ ✓

→ Select DISTINCT A ✓
FROM R
WHERE A IN (Select B FROM S)
= or some

We use DISTINCT just in case we get duplicates

# Topic: Function used for Nested Queries

One More case: If we want to retrieve a Record R(A,B) which exists somewhere in S(C,D)




Relational Algebra:

3. R(A,B) S(C,D)

$\prod (R \bowtie S)$

AB R.A = S.C

$\wedge$ R.B = SD

The (A,B) pair of R should exist in the(C,D) pair set of S.

# Topic: Function used for Nested Queries

## Corresponding Query:

$$\prod_{AB}(R \bowtie_{\substack{R.A = S.C \\ \wedge\ R.B = S.D}} S)$$

→ Select A, B
FROM R
WHERE (A, B) IN (Select B FROM S)

A,B value = Some(atleast one) of S(C,D)

i.e. A,B value of R should be member of Record Set S of C,D.

# Inspiring Stories : DP Puglisi

***Background:*** **Priest in a Mafia-ruled area in Sicily.**

***Struggles:*** **Mafia controlled everything; fear ruled the streets.**

***Achievements:*** **Taught kids to standup for themselves.**

***Impact:*** **Killed for it, but inspired law and church to oppose crime.**

# Topic: Any && All

**ANY vs ALL :-**

Preceded by operator <,<=,>,>=,<>

X < ANY


Y set

TRUE if X value < SOME value of Y set
else FALSE

X < ALL


Y set

TRUE if X value < EVERY value of Y set
else FALSE

- WITH ANY/ALL, we can use any of the operator among <,<=,>,>=,<>
- ANY function can be used for Conditional Join ($\bowtie_c$) Queries.

# Topic: Any && All

1. R(A…) S(B…)

Retrieve values of R.A that are less than S.B

# Topic: Any && All

## 1. R(A...) S(B...)

Retrieve values of R.A that are less than S.B

Now, the above algebra $\prod_{A} (R \bowtie_{R.A<S.B} S)$

# Topic: Any && All

1. R(A…) S(B…)

Retrieve values of R.A that are less than S.B

Now, the above algebra $\prod_{A}(R \bowtie_{R.A<S.B} S)$

≡ Select distinct A
FROM R
WHERE R.A < ANY (Select B From S)

# Topic: Any && All

1. R(A…) S(B…)

Retrieve values of R.A that are less than S.B

Now, the above algebra $\prod_{A} (R \bowtie_{R.A<S.B} S)$

≡ Select distinct A
FROM R
WHERE R.A < ANY (Select B From S)

Can be expressed in queries as

≡ Select distinct R.A
FROM R, S
WHERE R.A < S.B

# Topic: Any && All

1. R(A…) S(B…)

Retrieve values of R.A that are less than S.B

Now, the above algebra $\prod_{A} (R \bowtie_{R.A<S.B} S)$

Can be expressed in queries as

≡ Select distinct A
FROM R
WHERE R.A < ANY ( Select B From S )

✓

≡ Select distinct R.A
FROM R, S
WHERE R.A < S.B

1. cross product of R and S,
2. WHERE condition is applied
3. Predicate Selection operator i.e. R.A < S.B
4. Projected distinct A value.

# Topic: Any && All

1. R(A…) S(B…)

Retrieve values of R.A that are less than S.B

Now, the above algebra $\prod_{A} (R \bowtie_{R.A<S.B} S)$

Can be expressed in queries as

≡ Select distinct A
FROM R
WHERE R.A < ANY ( Select B From S )

≡ Select distinct R.A
FROM R, S
WHERE R.A < S.B

All the above three gives same result.

1. cross product of R and S,
2. WHERE condition is applied
3. Predicate Selection operator i.e. R.A < S.B
4. Projected distinct A value.

# Topic: Any && All

2. R(A…) S(B...) Retrieve A values of R < every B value of S.

# Topic: Any && All

2. R(A…) S(B...) Retrieve A values of R < every B value of S.

   {A < every B} ≡ {all A values} - {A ≥ some B}

# Topic: Any && All

2. R(A…) S(B...) Retrieve A values of R < every B value of S.

   {A < every B} ≡ {all A values} - {A ≥ some B}

   Relational Algebra Query = $\prod_{A}(R) - \prod(R \bowtie_{R.A>=S.B} S)$

# Topic: Any && All

2. R(A…) S(B...) Retrieve A values of R < every B value of S.

   {A < every B} ≡ {all A values} - {A ≥ some B}

   Relational Algebra Query = $\prod_{A}(R) - \prod(R \bowtie_{R.A>=S.B} S)$

   except

   Select DISTINCT A

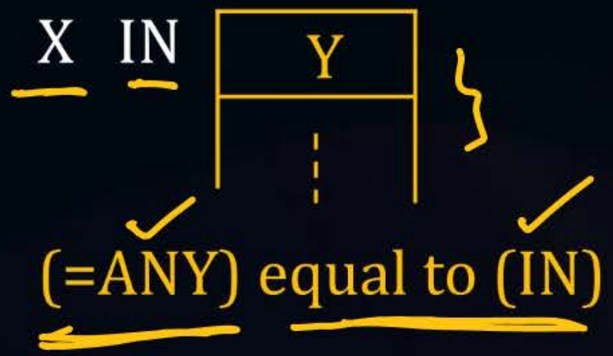
   From R

   WHERE R.A < ALL (Select B From S) ← SQL Query

- In RA, we used Set-difference of complement data, instead we can just use ALL in SQL.

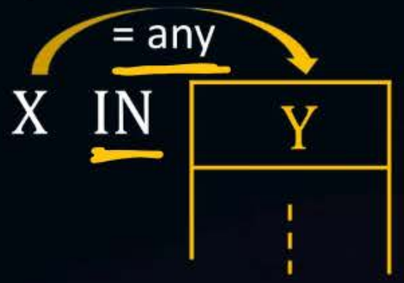
# Topic: Any && All

Say we have



# Topic: Any && All

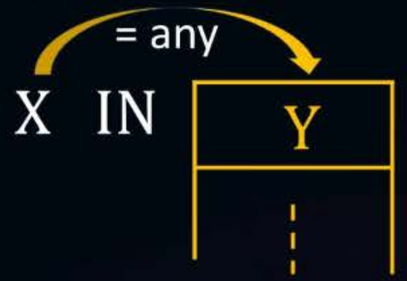
Say we have




(=ANY) equal to (IN)

# Topic: Any && All

Say we have



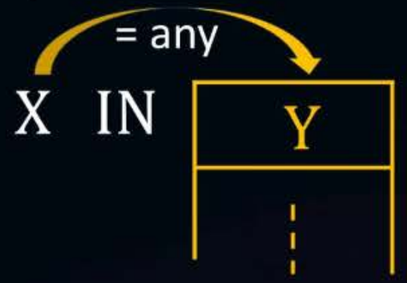

(=ANY) equal to (IN)

R(A) S(B)

$\Pi_{A}$ $(R \bowtie_{R.A = S.B} S)$ ⇒ Select DISTINCT A
From R
WHERE A IN Select B FROM S

# Topic: Any && All

Say we have




(=ANY) equal to (IN)

R(A) S(B)

$\prod_{A}$ $(R \bowtie_{R.A = S.B} S)$ ⇒ Select DISTINCT A
From R
WHERE A IN Select B FROM S
= ANY

# Topic: Any && All

If R(A,B) is compared with S(C,D)

# Topic: Any && All

If R(A,B) is compared with S(C,D)

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

$\prod_{AB} (R \bowtie_{R.A = S.C \wedge R.B=S.D} S)$

# Topic: Any && All

If R(A,B) is compared with S(C,D)

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

$\prod_{AB}$ ( R $\bowtie_{R.A = S.C \wedge R.B=S.D}$ S )

⇒ Select Distinct A, B
From R
WHERE (A,B) IN (Select C,D From S);

# Topic: Any && All

If R(A,B) is compared with S(C,D)

$$\prod_{AB} (R \bowtie_{R.A = S.C \wedge R.B=S.D} S)$$

⇒ Select Distinct A, B
From R
WHERE (A,B) IN (Select C,D From S);

= any ✗

Cant be Directly
replaced with =ANY

because "=" expects only one value, comparison happen only on one value.

# Topic: Any && All

**Advantage of IN**

Any number of attributes can be compared with any number of attribute not just single values

**Advantage of ANY**

It is not only for Equi-Join, but it is also suitable for other conditional Joins as well like <, >, =, <= etc.

# Topic: Any && All

X NOT IN

| Y |
|---|
| 2 |
| 4 |
| 6 |
| 8 |

# Topic: Any && All

X **NOT IN**

| Y |
|---|
| 2 |
| 4 |
| 6 |
| 8 |

<> All

here "**NOT IN**" can be replaced with "**<> All**"
⇒ ( <> ALL) equal to (NOT IN)

# Topic: Any && All

**

EXISTS Function:

Used to test whether the result of Inner Query is Empty or NOT empty

Exists (Inner Query)

TRUE

- if result of Inner Query is not empty.
- If result of Inner Query has at least one tuple(some or any)

This means that the result is conditional JOIN result ($\bowtie_c$)

# Topic: Any && All

EXISTS Function:

Used to test whether the result of Inner Query is Empty or NOT empty

Exists (Inner Query)

TRUE

- if result of Inner Query is not empty.
- If result of Inner Query has at least one tuple(some or any)

This means that the result is conditional JOIN result ($\bowtie_c$)

used for conditional JOIN Query in correlated Queries.

# Topic: Any && All

Example:

Emp (Eid dno Sal)  Retrieve Eid's whose Sal < some empSal of dept 5.

# Topic: Any && All

Example:

Emp (Eid dno Sal)  Retrieve Eid's whose Sal < some empSal of dept 5.

$$\prod_{eid}(Emp \bowtie_{Sal<S} \rho_{I,D,S} (\sigma_{dno=5} (Emp)))$$

# Topic: Any && All

Example:

Emp (Eid dno Sal)  Retrieve Eid's whose Sal < some empSal of dept 5.

$$\prod_{eid}(Emp \bowtie_{Sal<S} \rho_{I,D,S} (\sigma_{dno=5} (Emp)))$$

For this kind of conditional JOIN query  =>  **EXISTS**

# Topic: Any && All

Example:

Emp (Eid dno Sal)  Retrieve Eid's whose Sal < some empSal of dept 5.

$$\prod_{eid}(Emp \bowtie_{Sal<S} \rho_{I,D,S} (\sigma_{dno=5} (Emp)))$$

For this kind of conditional JOIN query  =>  **EXISTS**

Select Eid

1. From Emp $T_1$
3. WHERE EXISTS (2.Select *
   From Emp $T_2$
   WHERE $T_2$.dno = 5  and $T_1$.Sal < $T_2$.Sal);

# Topic: Any && All

Example:

Emp (Eid dno Sal)  Retrieve Eid's whose Sal < some empSal of dept 5.

$$\prod_{eid}(Emp \bowtie_{Sal<S} \rho_{I,D,S} (\sigma_{dno=5} (Emp)))$$

For this kind of conditional JOIN query  =>  **EXISTS**

Select Eid
1. From Emp $T_1$
3. WHERE EXISTS (2.Select *
   From Emp $T_2$
   WHERE $T_2$.dno = 5  and $T_1$.Sal < $T_2$.Sal);

Not empty

The above is correlated nested query.

⇒  EXISTS → TRUE ⇒ WHERE True.

# Topic: Any && All

```
Select Eid
1.    From Emp T1
3.   WHERE EXISTS (2.Select *
                   From Emp T2
                   WHERE T2.dno = 5 and T1.Sal < T2.Sal);
```

Example:-

Emp $T_1$

| Emp | dno | Sal |
|---|---|---|
| $e_1$ | 4 | 10 |
| $e_2$ | 5 | 20 |
| $e_3$ | 4 | 30 |
| $e_4$ | 5 | 40 |
| $e_5$ | 4 | 50 |

Emp $T_2$

| Emp | dno | Sal |
|---|---|---|
| $e_1$ | 4 | 10 |
| $e_2$ | 5 | 20 |
| $e_3$ | 4 | 30 |
| $e_4$ | 5 | 40 |
| $e_5$ | 4 | 50 |

# Topic: Any && All

```
Select Eid
1.    From Emp T1
3.   WHERE EXISTS (2.Select *
                   From Emp T2
                   WHERE T2.dno = 5 and T1.Sal < T2.Sal);
```

Example:-

Emp $T_1$

| Emp | dno | Sal |
|---|---|---|
| $e_1$ | 4 | 10 |
| $e_2$ | 5 | 20 |
| $e_3$ | 4 | 30 |
| $e_4$ | 5 | 40 |
| $e_5$ | 4 | 50 |

Emp $T_2$

| Emp | dno | Sal |
|---|---|---|
| $e_1$ | 4 | 10 |
| $e_2$ | 5 | 20 |
| $e_3$ | 4 | 30 |
| $e_4$ | 5 | 40 |
| $e_5$ | 4 | 50 |

# Topic: Any && All

```
Select Eid
1.    From Emp T1
3.   WHERE EXISTS (2.Select *
                   From Emp T2
                   WHERE T2.dno = 5 and T1.Sal < T2.Sal);
```

Example:-

Emp $T_1$

| Emp | dno | Sal |
|---|---|---|
| $e_1$ | 4 | 10 |
| $e_2$ | 5 | 20 |
| $e_3$ | 4 | 30 |
| $e_4$ | 5 | 40 |
| $e_5$ | 4 | 50 |

Emp $T_2$

| Emp | dno | Sal |
|---|---|---|
| $e_1$ | 4 | 10 |
| $e_2$ | 5 | 20 |
| $e_3$ | 4 | 30 |
| $e_4$ | 5 | 40 |
| $e_5$ | 4 | 50 |

# Topic: Any && All

```
Select Eid
1.    From Emp T1
3.   WHERE EXISTS (2.Select *
                    From Emp T2
                    WHERE T2.dno = 5 and T1.Sal < T2.Sal);
```

Example:-

Emp $T_1$

| Emp | dno | Sal |
|---|---|---|
| $e_1$ | 4 | 10 |
| $e_2$ | 5 | 20 |
| $e_3$ | 4 | 30 |
| $e_4$ | 5 | 40 |
| $e_5$ | 4 | 50 |

Emp $T_2$

| Emp | dno | Sal |
|---|---|---|
| $e_1$ | 4 | 10 |
| $e_2$ | 5 | 20 |
| $e_3$ | 4 | 30 |
| $e_4$ | 5 | 40 |
| $e_5$ | 4 | 50 |

# Topic: Any && All

```
Select Eid
1.    From Emp T1
3.   WHERE EXISTS (2.Select *
                    From Emp T2
                    WHERE T2.dno = 5 and T1.Sal < T2.Sal);
```

Example:-

Emp $T_1$

| Emp | dno | Sal |
|---|---|---|
| $e_1$ | 4 | 10 |
| $e_2$ | 5 | 20 |
| $e_3$ | 4 | 30 |
| $e_4$ | 5 | 40 |
| $e_5$ | 4 | 50 |

Emp $T_2$

| Emp | dno | Sal |
|---|---|---|
| $e_1$ | 4 | 10 |
| $e_2$ | 5 | 20 |
| $e_3$ | 4 | 30 |
| $e_4$ | 5 | 40 |
| $e_5$ | 4 | 50 |

# Topic: Any && All

```
Select Eid
1.    From Emp T1
3.   WHERE EXISTS (2.Select *
                   From Emp T2
                   WHERE T2.dno = 5 and T1.Sal < T2.Sal);
```

Example:-

Emp $T_1$

| Emp | dno | Sal |
|---|---|---|
| $e_1$ | 4 | 10 |
| $e_2$ | 5 | 20 |
| $e_3$ | 4 | 30 |
| $e_4$ | 5 | 40 |
| $e_5$ | 4 | 50 |

Emp $T_2$

| Emp | dno | Sal |
|---|---|---|
| $e_1$ | 4 | 10 |
| $e_2$ | 5 | 20 |
| $e_3$ | 4 | 30 |
| $e_4$ | 5 | 40 |
| $e_5$ | 4 | 50 |

# Topic: Any && All

```
Select Eid
1.    From Emp T1
3.   WHERE EXISTS (2.Select *
                    From Emp T2
                    WHERE T2.dno = 5 and T1.Sal < T2.Sal);
```

Example:-

Emp $T_1$

| Emp | dno | Sal |
|---|---|---|
| $e_1$ | 4 | 10 |
| $e_2$ | 5 | 20 |
| $e_3$ | 4 | 30 |
| $e_4$ | 5 | 40 |
| $e_5$ | 4 | 50 |

Emp $T_2$

| Emp | dno | Sal |
|---|---|---|
| $e_1$ | 4 | 10 |
| $e_2$ | 5 | 20 |
| $e_3$ | 4 | 30 |
| $e_4$ | 5 | 40 |
| $e_5$ | 4 | 50 |

# Topic: Any && All

```
Select Eid
1.    From Emp T1
3.   WHERE EXISTS (2.Select *
                   From Emp T2
                   WHERE T2.dno = 5 and T1.Sal < T2.Sal);
```

Example:-

Emp $T_1$

| | Emp | dno | Sal |
|---|---|---|---|
| ✓ | $e_1$ | 4 | 10 |
| ✓ | $e_2$ | 5 | 20 |
| ✓ | $e_3$ | 4 | 30 |
| ✗ | $e_4$ | 5 | 40 |
| | $e_5$ | 4 | 50 |

Emp $T_2$

| Emp | dno | Sal |
|---|---|---|
| $e_1$ | 4 | 10 |
| $e_2$ | 5 | 20 |
| $e_3$ | 4 | 30 |
| $e_4$ | 5 | 40 |
| $e_5$ | 4 | 50 |

# Topic: Any && All

```
Select Eid
1.    From Emp T1
3.   WHERE EXISTS (2.Select *
                   From Emp T2
                   WHERE T2.dno = 5 and T1.Sal < T2.Sal);
```

Example:-

Emp $T_1$

| Emp | dno | Sal |
|---|---|---|
| $e_1$ | 4 | 10 |
| $e_2$ | 5 | 20 |
| $e_3$ | 4 | 30 |
| $e_4$ | 5 | 40 |
| $e_5$ | 4 | 50 |

Emp $T_2$

| Emp | dno | Sal |
|---|---|---|
| $e_1$ | 4 | 10 |
| $e_2$ | 5 | 20 |
| $e_3$ | 4 | 30 |
| $e_4$ | 5 | 40 |
| $e_5$ | 4 | 50 |

# Topic: Any && All

```
Select Eid
1.    From Emp T1
3.   WHERE EXISTS (2.Select *
                   From Emp T2
                   WHERE T2.dno = 5 and T1.Sal < T2.Sal);
```

Example:-

Emp $T_1$

| Emp | dno | Sal |
|---|---|---|
| $e_1$ | 4 | 10 |
| $e_2$ | 5 | 20 |
| $e_3$ | 4 | 30 |
| $e_4$ | 5 | 40 |
| $e_5$ | 4 | 50 |

Emp $T_2$

| Emp | dno | Sal |
|---|---|---|
| $e_1$ | 4 | 10 |
| $e_2$ | 5 | 20 |
| $e_3$ | 4 | 30 |
| $e_4$ | 5 | 40 |
| $e_5$ | 4 | 50 |

# Topic: Any && All

```
Select Eid
1.    From Emp T1
3.   WHERE EXISTS (2.Select *
                   From Emp T2
                   WHERE T2.dno = 5 and T1.Sal < T2.Sal);
```

Example:-

Emp $T_1$

| | Emp | dno | Sal |
|---|---|---|---|
| ✓ | $e_1$ | 4 | 10 |
| ✓ | $e_2$ | 5 | 20 |
| ✓ | $e_3$ | 4 | 30 |
| ✗ | $e_4$ | 5 | 40 |
| ✗ | $e_5$ | 4 | 50 |

Emp $T_2$

| Emp | dno | Sal |
|---|---|---|
| $e_1$ | 4 | 10 |
| $e_2$ | 5 | 20 |
| $e_3$ | 4 | 30 |
| $e_4$ | 5 | 40 |
| $e_5$ | 4 | 50 |

**Conclusion :** EXISTS Clause behaves like Conditional ($\bowtie_c$) Join

# Topic: Any && All

#Q. Emp (eid, dno, sal) Retrieve eid's whose salary is less than every emp. of dept 5.

# Topic: Any && All

#Q. Emp (eid, dno, sal) Retrieve eid's whose salary is less than every emp. of dept 5.

$$\prod_{Eid}(emp) - \prod_{Eid}(Emp \bowtie_{sal >= S} \rho_{I,D,S}(\sigma_{dno=5}(Emp)))$$

# Topic: Any && All

#Q. Emp (eid, dno, sal) Retrieve eid's whose salary is less than every emp. of dept 5.

employes salary >= atleast one member of dept 5.

$$\prod_{Eid}(emp) - \prod_{Eid}(Emp \bowtie_{sal >= S} \rho_{I,D,S}(\sigma_{dno=5}(Emp)))$$

# Topic: Any && All

#Q. Emp (eid, dno, sal) Retrieve eid's whose salary is less than every emp. of dept 5.

employes salary >= atleast one member of dept 5.

$$\prod_{Eid}(emp) - \prod_{Eid}(Emp \bowtie_{sal >= S} \rho_{I,D,S}(\sigma_{dno=5}(Emp)))$$

SQL

```
Select eid
from Emp T1
WHERE NOT EXISTS (Select *
                  From Emp T2
                  WHERE T1.Sal >= T2.Sal
                  AND T2.dno = 5);
```

Not empty

# Topic: Any && All

#Q. Emp (eid, dno, sal) Retrieve eid's whose salary is less than every emp. of dept 5.

employes salary >= atleast one member of dept 5.

$$\prod_{Eid}(emp) - \prod_{Eid}(Emp \bowtie_{sal >= S} \rho_{I,D,S}(\sigma_{dno=5}(Emp)))$$

SQL

```
Select eid
from Emp T1
WHERE NOT EXISTS (Select *
                  From Emp T2
                  WHERE T1.Sal >= T2.Sal
                  AND T2.dno = 5);
```

Not empty

IF EXISTS IS TRUE, we get $T_1$.eid's whose Sal >= Some emp Sal of dept 5

Using NOT EXISTS ≡ $T_1$. eid's whose Sal < every emp Sal of dept 5

# Topic: EXISTS and NOT EXISTS

**EXISTS ⇔ SOME or Atleast one**

**NOT EXISTS ⇔ EVERY**

So, EXISTS returns true when result is not empty and

NON-EXISTS returns true when result is empty.

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways R(A. . .), S(B. . . )

Retrieve 'A' values of R which are more than some 'B' values of S.

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways R(A. . .), S(B. . . )

Retrieve 'A' values of R which are more than some 'B' values of S.

RA : $\Pi_A (R \bowtie_{R.A>S.B} S)$

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways R(A. . .), S(B. . . )

Retrieve 'A' values of R which are more than some 'B' values of S.

RA : $\Pi_A (R \bowtie_{R.A>S.B} S)$

**SQL:**

```
Select distinct R·A
From R, S
WHERE R·A > S · B;
```

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways R(A. . .), S(B. . . )

Retrieve 'A' values of R which are more than some 'B' values of S.

RA : $\Pi_A (R \bowtie_{R.A>S.B} S)$

**SQL:**

```
Select distinct R·A
From R, S
WHERE R·A > S · B;
```

```
(Select R · A
From R
WHERE R· A > Any (Select B from S);
```

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways R(A. . .), S(B. . . )

Retrieve 'A' values of R which are more than some 'B' values of S.

RA : $\Pi_A (R \bowtie_{R.A>S.B} S)$

**SQL:**

```
Select distinct R·A
From R, S
WHERE R·A > S · B;
```

```
(Select R · A
From R
WHERE R· A > Any (Select B from S);
```

```
(Select R · A
From R
WHERE EXISTS(SELECT *
             FROM S
             WHERE  R.A > S.B);
```

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways R(A. . .), S(B. . . )

Retrieve 'A' values of R which are more than some 'B' values of S.

RA : $\Pi_A (R \bowtie_{R.A>S.B} S)$

**SQL:**

```
Select distinct R·A
From R, S
WHERE R·A > S · B;
```

JOIN Query

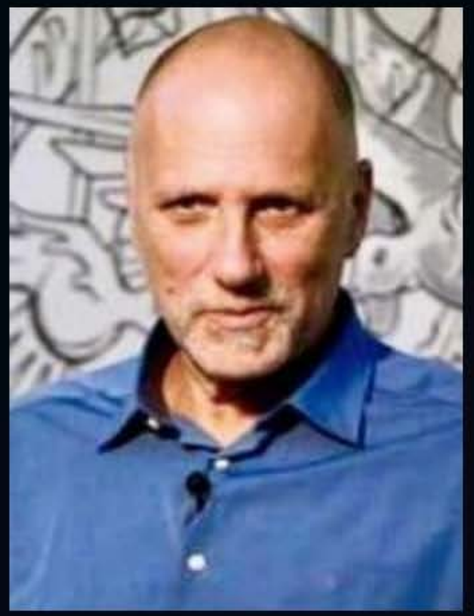
```
(Select R · A
From R
WHERE R· A > Any (Select B from S);
```

NESTED Query
Without Correlation

```
(Select R · A
From R
WHERE EXISTS(SELECT *
             FROM S
             WHERE  R.A > S.B);
```

NESTED Query
with correlation

# Inspiring Stories : Yossi Ghinsberg



***Background:*** **Explorer in the Amazon jungle.**

***Struggles:*** **Separated from group; survived alone for three weeks in wild forest.**

***Achievements:*** **Ate insects, walked through dangers, found help.**

***Impact:*** **Came back alive from one of Earth's toughest jungles.**

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways R(A. . .), S(B. . . )

Retrieve 'A' values of R which are more than every B values of S.

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways R(A. . .), S(B. . . )

> every <= Some

Retrieve 'A' values of R which are more than every B values of S.

RA : $\Pi_A(R) - \Pi_A (R \bowtie_{R.A<=S.B} S)$

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways R(A. . .), S(B. . . )

Retrieve 'A' values of R which are more than every B values of S.

RA : $\Pi_A(R) - \Pi_A (R \bowtie_{R.A<=S.B} S)$

except

**SQL:**

```
Select A
From R
     EXCEPT
Select A
From R, S
WHERE A <= B;
```

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways R(A. . .), S(B. . . )

Retrieve 'A' values of R which are more than every B values of S.

RA : $\Pi_A(R) \; - \; \Pi_A \, (R \bowtie_{R.A<=S.B} S)$

**SQL:**

```
Select A
From R
     EXCEPT
Select A
From R, S
WHERE A <=  B;
```

```
(Select A
From R
WHERE A > ALL (Select B from S);
```

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways R(A. . .), S(B. . . )

Retrieve 'A' values of R which are more than every B values of S.

RA : $\Pi_A(R) \; - \; \Pi_A (R \bowtie_{R.A<=S.B} S)$

**SQL:**

```
Select A
From R
     EXCEPT
Select A
From R, S
WHERE A <=  B;
```

```
(Select A
From R
WHERE A > ALL (Select B from S);
```

```
(Select  A
From R
WHERE NOT EXISTS(SELECT *
                 FROM S
                 WHERE  R.A <= S.B);
```

NOT( A<=Some B)
A > EVERY B

A<=Some B

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways R(A. . .), S(B. . . )

Retrieve 'A' values of R which are more than every B values of S.

RA : $\Pi_A(R) \; - \; \Pi_A (R \bowtie_{R.A<=S.B} S)$

**SQL:**

```
Select A
From R
     EXCEPT
Select A
From R, S
WHERE A <=  B;
```

JOIN Query ✓

```
(Select A
From R
WHERE A > ALL (Select B from S);
```

NESTED Query
Without Correlation ✓

NOT( A<=Some B)
A > EVERY B

A<=Some B

```
(Select  A
From R
WHERE NOT EXISTS(SELECT *
                 FROM S
                 WHERE  R.A <= S.B);
```

NESTED Query
with correlation ✓

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways EMP(eid,sal)

Retrieve eid's who get minimum salary (≤ every)

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways EMP(eid,sal)

Retrieve eid's who get minimum salary (≤ every)

RA : $\Pi_{eid}(Emp) - \Pi_{eid}(Emp \bowtie_{sal > S} \rho_{I,S}(Emp))$

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways EMP(eid,sal)

Retrieve eid's who get minimum salary (≤ every)

RA : $\Pi_{eid}(Emp) - \Pi_{eid}(Emp \bowtie_{sal > S} \rho_{I,S}(Emp))$

**SQL:**

```
(Select eid
From EMP
WHERE sal <= ALL (Select sal from EMP);
```

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways EMP(eid,sal)

Retrieve eid's who get minimum salary (≤ every)

RA : $\Pi_{eid}(Emp) - \Pi_{eid}(Emp \bowtie_{sal > S} \rho_{I,S}(Emp))$

**SQL:**

```
(Select eid
From EMP
WHERE sal <= ALL (Select sal from EMP);
```

```
(Select  eid
From EMP
WHERE NOT EXISTS(SELECT *
                 FROM EMP T2
                 WHERE  T1.sal > T2.sal);
```

NOT(T1.sal> Some T2.sal)
T1.sal<= EVERY T2.sal

T1.sal> Some T2.sal

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways EMP(eid,sal)

Retrieve eid's who get minimum salary (≤ every)

RA : $\Pi_{eid}(Emp) - \Pi_{eid}(Emp \bowtie_{sal > S} \rho_{I,S}(Emp))$

**SQL:**

```
(Select eid
From EMP
WHERE sal <= ALL (Select sal from EMP);
```

NESTED Query
Without Correlation ✓

NOT(T1.sal> Some T2.sal)
T1.sal<= EVERY T2.sal

T1.sal> Some T2.sal

```
(Select  eid
From EMP
WHERE NOT EXISTS(SELECT *
                 FROM EMP T2
                 WHERE  T1.sal > T2.sal);
```

NESTED Query
with correlation ✓

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways EMP(eid,sal, dept)

Retrieve eid's who get minimum salary (≤ every) for each dept

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways EMP(eid,sal, ~~dept~~ dno)

Retrieve eid's who get minimum salary (≤ every) for each dept

RA : $\Pi_{eid}(EMP) - \Pi_{eid}(EMP \bowtie_{sal > s \wedge dno. = D} \rho_{I,S,D}(EMP))$

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways EMP(eid,sal, dept)

Retrieve eid's who get minimum salary (≤ every) for each dept

RA : $\Pi_{eid}(EMP) - \Pi_{eid}(EMP \bowtie_{sal > s \wedge dno. = D} \rho_{I,S,D}(EMP))$

We will write using group by & Aggregation.

We need to group using deptno. We need Agg for sal

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways EMP(eid,sal, dept)

Retrieve eid's who get minimum salary (≤ every) for each dept

RA : $\Pi_{eid}(EMP) - \Pi_{eid}(EMP \bowtie_{sal > s \wedge dno. = D} \rho_{I,S,D}(EMP))$

We will write using group by & Aggregation.

We need to group using deptno. We need Agg for sal

But eid is selected after that

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways EMP(eid,sal, dept)

Retrieve eid's who get minimum salary (≤ every) for each dept

RA : $\Pi_{eid}(EMP) - \Pi_{eid}(EMP \bowtie_{sal > s \wedge dno. = D} \rho_{I,S,D}(EMP))$

We will write using group by & Aggregation.

We need to group using deptno. We need Agg for sal

But eid is selected after that

For GROUP BY every attribute which is chosen for GROUP BY must be in SELECT Clause

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways EMP(eid,sal, dept)

Retrieve eid's who get minimum salary (≤ every) for each dept

RA : $\Pi_{eid}$ (EMP)– $\Pi_{eid}$ (EMP $\bowtie_{sal > s \wedge dno. = D}$ $\rho_{I,S,D}$(EMP))

We will write using group by & Aggregation.

We need to group using deptno. We need Agg for sal

But eid is selected after that

For GROUP BY every attribute which is chosen for GROUP BY must be in SELECT Clause

Hence we used NESTED Queries

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways EMP(eid,sal, dept)

Retrieve eid's who get minimum salary (≤ every) for each dept

RA : $\Pi_{eid}(EMP) - \Pi_{eid}(EMP \bowtie_{sal > s \wedge dno. = D} \rho_{I,S,D}(EMP))$

**SQL**

```
Select eid
FROM Emp, (Select dept, min(sal) MSal
           FROM Emp
           GROUP by dno) temp
WHERE Emp · dno = temp · dno and Sal = MSal;
```

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways EMP(eid,sal, dept)

Retrieve eid's who get minimum salary (≤ every) for each dept

RA : $\Pi_{eid}(EMP) - \Pi_{eid}(EMP \bowtie_{sal > s \wedge dno. = D} \rho_{I,S,D}(EMP))$

**SQL**

```
Select eid
FROM Emp, (Select dept, min(sal) MSal
           FROM Emp
           GROUP by dno) temp
WHERE Emp · dno = temp · dno and Sal = MSal;
```

EMP

| Emp | sal | Dept |
|---|---|---|
| $e_1$ | 40 | 3 |
| $e_2$ | 70 | 3 |
| $e_3$ | 50 | 3 |
| $e_4$ | 30 | 4 |
| $e_5$ | 60 | 4 |

TEMP

| Dept | min(sal) |
|---|---|
| 3 | 40 |
| 4 | 30 |

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways EMP(eid,sal, dept)

Retrieve eid's who get minimum salary (≤ every) for each dept

RA : $\Pi_{eid}(EMP) - \Pi_{eid}(EMP \bowtie_{sal > s \wedge dno. = D} \rho_{I,S,D}(EMP))$

**SQL**

```
Select eid
FROM Emp, (Select dept, min(sal) MSal
           FROM Emp
           GROUP by dno) temp
WHERE Emp · dno = temp · dno and Sal = MSal;
```

EMP

| Emp | sal | Dept |
|---|---|---|
| $e_1$ | 40 | 3 |
| $e_2$ | 70 | 3 |
| $e_3$ | 50 | 3 |
| $e_4$ | 30 | 4 |
| $e_5$ | 60 | 4 |

= = TEMP

| Dept | min(sal) |
|---|---|
| 3 | 40 |
| 4 | 30 |

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways EMP(eid,sal, dept)

Retrieve eid's who get minimum salary (≤ every) for each dept

RA : $\Pi_{eid}(EMP) - \Pi_{eid}(EMP \bowtie_{sal > s \wedge dno. = D} \rho_{I,S,D}(EMP))$

**SQL**

```
Select eid
FROM Emp, (Select dept, min(sal) MSal
                FROM Emp
                GROUP by dno) temp
WHERE Emp · dno = temp · dno and Sal = MSal;
```

EMP

| Emp | sal | Dept |
|---|---|---|
| $e_1$ | 40 | 3 |
| $e_2$ | 70 | 3 |
| $e_3$ | 50 | 3 |
| $e_4$ | 30 | 4 |
| $e_5$ | 60 | 4 |

= = TEMP

| Dept | min(sal) |
|---|---|
| 3 | 40 |
| 4 | 30 |

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways EMP(eid,sal, dept)

Retrieve eid's who get minimum salary (≤ every) for each dept

RA : $\Pi_{eid}(EMP) - \Pi_{eid}(EMP \bowtie_{sal > s \wedge dno. = D} \rho_{I,S,D}(EMP))$

**SQL**

```
Select eid
FROM Emp, (Select dept, min(sal) MSal
                FROM Emp
                GROUP by dno) temp
WHERE Emp · dno = temp · dno and Sal = MSal;
```

EMP

| Emp | sal | Dept |
|---|---|---|
| $e_1$ | 40 | 3 |
| $e_2$ | 70 | 3 |
| $e_3$ | 50 | 3 |
| $e_4$ | 30 | 4 |
| $e_5$ | 60 | 4 |

= = TEMP

| Dept | min(sal) |
|---|---|
| 3 | 40 |
| 4 | 30 |

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways EMP(eid,sal, dept)

Retrieve eid's who get minimum salary (≤ every) for each dept

RA : $\Pi_{eid}(EMP) - \Pi_{eid}(EMP \bowtie_{sal > s \wedge dno. = D} \rho_{I,S,D}(EMP))$

**SQL**

```
Select eid
FROM Emp, (Select dept, min(sal) MSal
                FROM Emp
                GROUP by dno) temp
WHERE Emp · dno = temp · dno and Sal = MSal;
```

EMP

| | Emp | sal | Dept |
|---|---|---|---|
| ✓ | $e_1$ | 40 | 3 |
| ✗ | $e_2$ | 70 | 3 |
| → | $e_3$ | 50 | 3 |
| | $e_4$ | 30 | 4 |
| | $e_5$ | 60 | 4 |

= = TEMP

| Dept | min(sal) |
|---|---|
| 3 | 40 |
| 4 | 30 |

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways EMP(eid,sal, dept)

Retrieve eid's who get minimum salary (≤ every) for each dept

RA : $\Pi_{eid}(EMP) - \Pi_{eid}(EMP \bowtie_{sal > s \wedge dno. = D} \rho_{I,S,D}(EMP))$

**SQL**

```
Select eid
FROM Emp, (Select dept, min(sal) MSal
                FROM Emp
                GROUP by dno) temp
WHERE Emp · dno = temp · dno and Sal = MSal;
```

EMP

| | Emp | sal | Dept |
|---|---|---|---|
| ✓ | $e_1$ | 40 | 3 |
| ✗ | $e_2$ | 70 | 3 |
| ✗ | $e_3$ | 50 | 3 |
| | $e_4$ | 30 | 4 |
| | $e_5$ | 60 | 4 |

= = TEMP

| Dept | min(sal) |
|---|---|
| 3 | 40 |
| 4 | 30 |

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways EMP(eid,sal, dept)

Retrieve eid's who get minimum salary (≤ every) for each dept

RA : $\Pi_{eid}(EMP) - \Pi_{eid}(EMP \bowtie_{sal > s \wedge dno. = D} \rho_{I,S,D}(EMP))$

**SQL**

```
Select eid
FROM Emp, (Select dept, min(sal) MSal
                FROM Emp
                GROUP by dno) temp
WHERE Emp · dno = temp · dno and Sal = MSal;
```

EMP

| | Emp | sal | Dept |
|---|---|---|---|
| ✓ | $e_1$ | 40 | 3 |
| ✗ | $e_2$ | 70 | 3 |
| ✗ | $e_3$ | 50 | 3 |
| | $e_4$ | 30 | 4 |
| | $e_5$ | 60 | 4 |

= = TEMP

| Dept | min(sal) |
|---|---|
| 3 | 40 |
| 4 | 30 |

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways EMP(eid,sal, dept)

Retrieve eid's who get minimum salary (≤ every) for each dept

RA : $\Pi_{eid}(EMP) - \Pi_{eid}(EMP \bowtie_{sal > s \wedge dno. = D} \rho_{I,S,D}(EMP))$

**SQL**

```
Select eid
FROM Emp, (Select dept, min(sal) MSal
                FROM Emp
                GROUP by dno) temp
WHERE Emp · dno = temp · dno and Sal = MSal;
```

EMP

| | Emp | sal | Dept |
|---|---|---|---|
| ✓ | $e_1$ | 40 | 3 |
| ✗ | $e_2$ | 70 | 3 |
| ✗ | $e_3$ | 50 | 3 |
| ✓ | $e_4$ | 30 | 4 |
| | $e_5$ | 60 | 4 |

= = TEMP

| Dept | min(sal) |
|---|---|
| 3 | 40 |
| 4 | 30 |

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways EMP(eid,sal, dept)

Retrieve eid's who get minimum salary (≤ every) for each dept

RA : $\Pi_{eid}(EMP) - \Pi_{eid}(EMP \bowtie_{sal > s \wedge dno. = D} \rho_{I,S,D}(EMP))$

**SQL**

```
Select eid
FROM Emp, (Select dept, min(sal) MSal
                FROM Emp
                GROUP by dno) temp
WHERE Emp · dno = temp · dno and Sal = MSal;
```

EMP

| | Emp | sal | Dept |
|---|---|---|---|
| ✓ | $e_1$ | 40 | 3 |
| ✗ | $e_2$ | 70 | 3 |
| ✗ | $e_3$ | 50 | 3 |
| ✓ | $e_4$ | 30 | 4 |
| → | $e_5$ | 60 | 4 |

= = TEMP

| Dept | min(sal) |
|---|---|
| 3 | 40 |
| 4 | 30 |

# Topic: EXISTS and NOT EXISTS

We will write one query in different ways EMP(eid,sal, dept)

Retrieve eid's who get minimum salary (≤ every) for each dept

RA : $\Pi_{eid}(EMP) - \Pi_{eid}(EMP \bowtie_{sal > s \wedge dno. = D} \rho_{I,S,D}(EMP))$

**SQL**

```
Select eid
FROM Emp, (Select dept, min(sal) MSal
           FROM Emp
           GROUP by dno) temp
WHERE Emp · dno = temp · dno and Sal = MSal;
```

EMP

| | Emp | sal | Dept |
|---|---|---|---|
| ✓ | $e_1$ | 40 | 3 |
| ✗ | $e_2$ | 70 | 3 |
| ✗ | $e_3$ | 50 | 3 |
| ✓ | $e_4$ | 30 | 4 |
| ✗ | $e_5$ | 60 | 4 |

= = TEMP

| Dept | min(sal) |
|---|---|
| 3 | 40 |
| 4 | 30 |

# Topic: Query Language

Enroll(Sid, Cid)  Course(Cid, Instructor)

Retrieve Sids who enrolled in every course taught by KORTH

# Topic: Query Language

**SQL Query which is equal to division of RA:**

**Some** == ⋈

**Every** == /

# Topic: Query Language

Enroll(Sid, Cid)  Course(Cid, Instructor)

Retrieve Sids who enrolled in every course taught by KORTH

# Topic: Query Language

Enroll(Sid, Cid) Course(Cid, Instructor)

Retrieve Sids who enrolled in every course taught by KORTH

- RA: $\Pi_{Sid,\ Cid}(Enroll) / \Pi_{Cid}(\sigma_{Instr=KORTH}(Course))$

# Topic: Query Language

Enroll(Sid, Cid) Course(Cid, Instructor)

Retrieve Sids who enrolled in every course taught by KORTH

- RA: $\Pi_{Sid,\ Cid}(Enroll) / \Pi_{Cid}(\sigma_{Instr=KORTH}(Course))$

Course

| Cid | Instr |
|---|---|
| $C_1$ | KORTH |
| $C_2$ | KORTH |
| $C_3$ | KORTH |
| $C_4$ | NAVATHE |

Enroll T2

| Sid | Cid |
|---|---|
| $S_1$ | $C_1$ |
| $S_1$ | $C_2$ |
| $S_1$ | $C_3$ |
| $S_2$ | $C_1$ |
| $S_2$ | $C_4$ |
| $S_3$ | $C_4$ |

Enroll T1

| Sid | Cid |
|---|---|
| $S_1$ | $C_1$ |
| $S_1$ | $C_2$ |
| $S_1$ | $C_3$ |
| $S_2$ | $C_1$ |
| $S_2$ | $C_4$ |
| $S_3$ | $C_4$ |

# Topic: Query Language

Enroll(Sid, Cid) Course(Cid, Instructor)

Retrieve Sids who enrolled in every course taught by KORTH

- RA: $\Pi_{Sid,\ Cid}(Enroll) / \Pi_{Cid}(\sigma_{Instr=KORTH}(Course))$

Then the student is enrolled for all courses of KORTH

All Cids of KORTH Courses — Cids of T2 which are enrolled by T1.sid = $\phi$

Course

| Cid | Instr |
|---|---|
| $C_1$ | KORTH |
| $C_2$ | KORTH |
| $C_3$ | KORTH |
| $C_4$ | NAVATHE |

=

Enroll T2

| Sid | Cid |
|---|---|
| $S_1$ | $C_1$ |
| $S_1$ | $C_2$ |
| $S_1$ | $C_3$ |
| $S_2$ | $C_1$ |
| $S_2$ | $C_4$ |
| $S_3$ | $C_4$ |

Enroll T1

| Sid | Cid |
|---|---|
| $S_1$ | $C_1$ |
| $S_1$ | $C_2$ |
| $S_1$ | $C_3$ |
| $S_2$ | $C_1$ |
| $S_2$ | $C_4$ |
| $S_3$ | $C_4$ |

# Topic: Query Language

Enroll(Sid, Cid) Course(Cid, Instructor)

Retrieve Sids who enrolled in every course taught by KORTH

- RA: $\Pi_{Sid, Cid}(Enroll) / \Pi_{Cid}(\sigma_{Instr=KORTH}(Course))$

T1.Sid = $S_1$ $C_1, C_2, C_3$ — $C_1, C_2, C_3$ = $\phi$

$C_1, C_2, C_3$: All Cids of KORTH Courses

$C_1, C_2, C_3$: Cids of T2 which are enrolled by T1.sid

RESULT

| Sid |
|---|
| $S_1$ |

Course

| Cid | Instr |
|---|---|
| $C_1$ | KORTH |
| $C_2$ | KORTH |
| $C_3$ | KORTH |
| $C_4$ | NAVATHE |

=

Enroll T2

| Sid | Cid |
|---|---|
| $S_1$ | $C_1$ |
| $S_1$ | $C_2$ |
| $S_1$ | $C_3$ |
| $S_2$ | $C_1$ |
| $S_2$ | $C_4$ |
| $S_3$ | $C_4$ |

Enroll T1

| Sid | Cid |
|---|---|
| $S_1$ | $C_1$ |
| $S_1$ | $C_2$ |
| $S_1$ | $C_3$ |
| $S_2$ | $C_1$ |
| $S_2$ | $C_4$ |
| $S_3$ | $C_4$ |

# Topic: Query Language

Enroll(Sid, Cid) Course(Cid, Instructor)

Retrieve Sids who enrolled in every course taught by KORTH

- RA: $\Pi_{Sid, Cid}(Enroll) / \Pi_{Cid}(\sigma_{Instr=KORTH}(Course))$

T1.Sid = $S_2$

$C_1, C_2, C_3$ — All Cids of KORTH Courses

—

$C_1, C_4$ — Cids of T2 which are enrolled by T1.sid

=

$C_2, C_3$

RESULT

| Sid |
|---|
| $S_1$ |

Course

| Cid | Instr |
|---|---|
| $C_1$ | KORTH |
| $C_2$ | KORTH |
| $C_3$ | KORTH |
| $C_4$ | NAVATHE |

Enroll T2

| Sid | Cid |
|---|---|
| $S_1$ | $C_1$ |
| $S_1$ | $C_2$ |
| $S_1$ | $C_3$ |
| $S_2$ | $C_1$ |
| $S_2$ | $C_4$ |
| $S_3$ | $C_4$ |

Enroll T1

| Sid | Cid |
|---|---|
| $S_1$ | $C_1$ |
| $S_1$ | $C_2$ |
| $S_1$ | $C_3$ |
| $S_2$ | $C_1$ |
| $S_2$ | $C_4$ |
| $S_3$ | $C_4$ |

# Topic: Query Language

Enroll(Sid, Cid) Course(Cid, Instructor)

Retrieve Sids who enrolled in every course taught by KORTH

- RA: $\Pi_{Sid, Cid}(Enroll) / \Pi_{Cid}(\sigma_{Instr=KORTH}(Course))$

T1.Sid = $S_3$

$C_1, C_2, C_3$ — All Cids of KORTH Courses

—

$C_4$ — Cids of T2 which are enrolled by T1.sid

=

$C_1, C_2, C_3$

RESULT

| Sid |
|---|
| $S_1$ |

Course

| Cid | Instr |
|---|---|
| $C_1$ | KORTH |
| $C_2$ | KORTH |
| $C_3$ | KORTH |
| $C_4$ | NAVATHE |

Enroll T2

| Sid | Cid |
|---|---|
| $S_1$ | $C_1$ |
| $S_1$ | $C_2$ |
| $S_1$ | $C_3$ |
| $S_2$ | $C_1$ |
| $S_2$ | $C_4$ |
| $S_3$ | $C_4$ |

=

Enroll T1

| Sid | Cid |
|---|---|
| $S_1$ | $C_1$ |
| $S_1$ | $C_2$ |
| $S_1$ | $C_3$ |
| $S_2$ | $C_1$ |
| $S_2$ | $C_4$ |
| $S_3$ | $C_4$ |

# Topic: Query Language

Enroll(Sid, Cid) Course(Cid, Instructor)

Retrieve Sids who enrolled in every course taught by KORTH

- RA: $\Pi_{Sid,\ Cid}(Enroll) / \Pi_{Cid}(\sigma_{Instr=KORTH}(Course))$

RESULT

| Sid |
|---|
| $S_1$ |

Since the inner query has to return EMPTY, we use NOT EXISTS

All Cids of KORTH Courses — Cids of T2 which are enrolled by T1.sid

Course

| Cid | Instr |
|---|---|
| $C_1$ | KORTH |
| $C_2$ | KORTH |
| $C_3$ | KORTH |
| $C_4$ | NAVATHE |

Enroll T2 =

| Sid | Cid |
|---|---|
| $S_1$ | $C_1$ |
| $S_1$ | $C_2$ |
| $S_1$ | $C_3$ |
| $S_2$ | $C_1$ |
| $S_2$ | $C_4$ |
| $S_3$ | $C_4$ |

Enroll T1

| Sid | Cid |
|---|---|
| $S_1$ | $C_1$ |
| $S_1$ | $C_2$ |
| $S_1$ | $C_3$ |
| $S_2$ | $C_1$ |
| $S_2$ | $C_4$ |
| $S_3$ | $C_4$ |

# Topic: Query Language

Enroll(Sid, Cid) Course(Cid, Instructor)

Retrieve Sids who enrolled in every course taught by KORTH

- RA: $\Pi_{Sid,\ Cid}(Enroll) / \Pi_{Cid}(\sigma_{Instr=KORTH}(Course))$

RESULT

| Sid |
|---|
| $S_1$ |

Since the inner query has to return EMPTY, we use NOT EXISTS

All Cids of KORTH Courses — Cids of T2 which are enrolled by T1.sid

```
SELECT Cid
FROM Course
WHERE Instructor=KORTH
      EXCEPT
SELECT Cid
FROM Enroll T2
WHERE T2.Sid = T1.Sid);
```

# Topic: Query Language

Enroll(Sid, Cid) Course(Cid, Instructor)

Retrieve Sids who enrolled in every course taught by KORTH

- RA: $\Pi_{Sid,\ Cid}(Enroll) / \Pi_{Cid}(\sigma_{Instr=KORTH}(Course))$

RESULT

| Sid |
|---|
| $S_1$ |

Since the inner query has to return EMPTY, we use NOT EXISTS

All Cids of KORTH Courses — Cids of T2 which are enrolled by T1.sid

```
Select DISTINCT Sid
FROM Enroll T1
WHERE NOT EXISTS ( SELECT Cid
                   FROM Course
                   WHERE Instructor=KORTH
                         EXCEPT
                   SELECT Cid
                   FROM Enroll T2
                   WHERE T2.Sid = T1.Sid);
```

# Topic: Query Language

Enroll(Sid, Cid) Course(Cid, Instructor)

Retrieve Sids who enrolled in every course taught by KORTH

- RA: $\Pi_{Sid,\ Cid}(Enroll) / \Pi_{Cid}(\sigma_{Instr=KORTH}(Course))$

RESULT

| Sid |
|---|
| $S_1$ |

Since the inner query has to return EMPTY, we use NOT EXISTS

All Cids of KORTH Courses — Cids of T2 which are enrolled by T1.sid

```
Select DISTINCT Sid
FROM Enroll T1
WHERE NOT EXISTS ( SELECT Cid
                   FROM Course
                   WHERE Instructor=KORTH
                         EXCEPT
                   SELECT Cid
                   FROM Enroll T2
                   WHERE T2.Sid = T1.Sid);
```

SQL Correlated Query Equivalent to RA

# Topic: Query Language

WorksFor(eid, pid)  Project (pid, pname)

Retrieve eid's who works for every Project of pname DB.

if {all pid's of pname · DB} − {pid's works by $T_1$ . eid} = $\phi$ then eid is working for all DB projects

Project

| pid | pname |
|---|---|
| P1 | DB |
| P2 | DB |
| P3 | DB |
| P4 | CN |

Work $T_2$

| eid | pid |
|---|---|
| | |

=

Work $T_1$

| eid | Pid |
|---|---|
| | |

Like before, we need division for the relational algebra

# Topic: Query Language

WorksFor(eid, pid)  Project (pid, pname)

Retrieve eid's who works for every Project of pname DB.

$$\Pi_{Eid,\ pid}(W) \ / \ \Pi_{Pid}(\sigma_{Pname\ =\ DB}(P))$$

Project

| pid | pname |
|---|---|
| P1 | DB |
| P2 | DB |
| P3 | DB |
| P4 | CN |

Work $T_2$

| eid | pid |
|---|---|
| | |

=

Work $T_1$

| eid | Pid |
|---|---|
| | |

Like before, we need division for the relational algebra

# Topic: Query Language

WorksFor(eid, pid)  Project (pid, pname)

Retrieve eid's who works for every Project of pname DB.

$$\Pi_{Eid,\ pid}\ (W)\ /\ \Pi_{Pid}\ (\sigma_{Pname\ =\ DB}(P))$$

**SQL:**

```
Select distinct T1. eid
FROM works for T1
WHERE NOT EXISTS ( Select Pid
                   FROM Project
                   WHERE Pname = DB
                         EXCEPT
                   Select Pid
                   FROM works for T2
                   WHERE T2. eid = T1 . eid)
```

# Inspiring Stories : Mauro Prosperi

***Background:*** **Marathon runner in the Sahara.**

***Struggles:*** **Lost by a sandstorm, trapped in desert for nine days.**

***Achievements:*** **Survived on bats and bugs until rescued.**

***Impact:*** **Proved human grit can beat extreme nature.**

# Topic: Query Language

RA, SQL → Imp

TRC

## Tuple Relational Calculus (TRC)

There is not much difference between RA logic and TRC logic, the only fundamental difference is we will be using Calculus Operator instead of algebra operator.

Non-Procedural Query language, it means we only formulate what data we retrieve from DB, and we don't focus much on "HOW" it is retrieved from DB.

uses first order logic & & Predicate Calculus formulas. This topic is also in Discrete maths, the formulas are used in TRC.

- Predicate Statements are statement which are either True or False not ambiguous.
- P, Q are Predicate Statements.

## Basic formulas:

- $P \cup Q \rightarrow$ P or Q
- $P \cap Q \rightarrow$ P and Q
- $\neg$ P : NOT P
- $P \rightarrow Q$ : if P then Q
- $X \in$ Rel [x is variables]
- $\exists X \in$ Rel (P(X))

  For some X which belongs to Rel s.t. P(X) must be TRUE.
- $\forall X \in$ Rel(P(X))

  For every X of Rel s.t. P(X) must be True"

} Quantifiers

## Format of TRC Query:

The format is {T/P(T)}

T → Tuple Variable

P(T)→ Formula Over Tuple Variable T.

This retrieve set of tuples (T), those who satisfy P(T).

# Topic: Query Language

**EX.:** T/(T ∈ stud ∧ T.age ≥ 20}.

Means:

- Retrieve students whose age is more than 20.
- As we can see, we only know that P(T) [in this case T ∈ stud ∧T.age>20] gives us the result, we don't care about procedure.
- whatever we need, for that we need to write "First order Formula" for that. That's it, that's all we need to do.
- Only concern is develop P(T) formula, to get data that we want."

# Topic: Query Language

There are two types of Tuple variables,

1. Bounded Variable:

- Variable bounded by Quantifier [∃, ∀]

∃ X ∈ Rel

X is bounded Variable

∀ X ∈ Rel

∃ $T_1$ ∈ stud(...)

$T_1$ & $T_2$ are bounded Variable

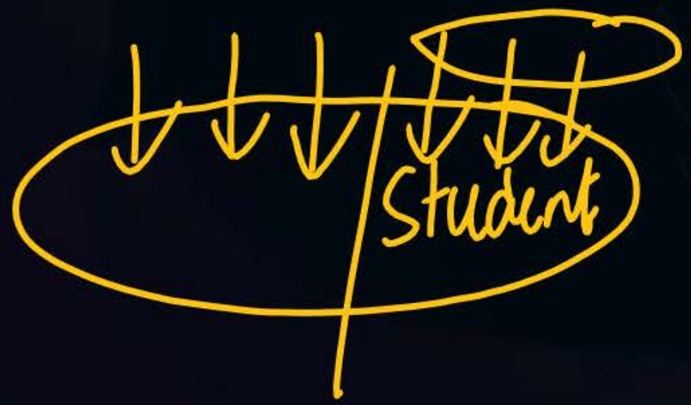
∀ $T_2$ ∈ course(...)

2. Free Variable

- Not bounded by Quantifier
- Universal Variable

Say $T \in$ Student

now, this T may or may not belong to student, so boundary of T is Universal within Relational Schema.



# Topic: Query Language

**NOTE:** That is {T/P(T)}

T → must be free variable remaining variable can be bounded, based on needs and requirements.

# Topic: Query Language

**Ex.** Suppliers (Sid, rating)

Parts (Pid, Pname, color)

Catalog (Sid, Pid, cost)

Retrieve Sid's of Suppliers who supplied some red parts.

# Topic: Query Language

**Ex.** Suppliers (Sid, rating)

Parts (Pid, Pname, color)

Catalog (Sid, Pid, cost)

Retrieve Sid's of Suppliers who supplied some red parts.

- There can be many Suppliers, and many Parts.
- Suppliers may/may not Supply
- In Catalog, we will have Suppliers who Supplied atleast one part.
- There may be Suppliers who didn't Supply part at all, they will be on Suppliers table but not in catalog table.
- Parts table will contain all parts.

# Topic: Query Language

**Ex.** Suppliers (Sid, rating)

Parts (Pid, Pname, color)

Catalog (Sid, Pid, cost)

Retrieve Sid's of Suppliers who supplied some red parts.

In terms of RA

We need Catalog table, and Parts table, becoz we need Sid of supplier who supplied red parts.

(Catalog(Sid Pid)) Parts(Pid Color)

Red.

# Topic: Query Language

**Ex.** Suppliers (Sid, rating)

Parts (Pid, Pname, color)

Catalog (Sid, Pid, cost)

Retrieve Sid's of Suppliers who supplied some red parts.

We need their cross product.

$\Pi_{Sid}(\sigma_{Catalog.pid = Parts.pid}(Catalog \times \sigma_{color = red}(Parts)))$

We are projecting Sid of the Catalog relation who supplied one of the red parts.

# Topic: Query Language

**Ex.** Suppliers (Sid, rating)

Parts (Pid, Pname, color)

Catalog (Sid, Pid, cost)

Retrieve Sid's of Suppliers who supplied some red parts.

Let's see corresponding T.R.C expression.

For TRC; $\exists\ T_1$ for Catalog

$\exists\ T_2$ for Parts

There should be one record in catalog table, and for that there should be one record in parts table with color red

# Topic: Query Language

**Ex.** Suppliers (Sid, rating)

Parts (Pid, Pname, color)

Catalog (Sid, Pid, cost)

Retrieve Sid's of Suppliers who supplied some red parts.

$T / \{\exists T_1 \in Catalog \; \exists T_2 \in Parts$

$(T_1 . pid = T_2 . pid \wedge T_2 .color = RED \wedge T = T_1 . sid)\}$

all the records which are satisfying P(T) are in result of Query

Basically if

1. $T_1 . pid = T_2 . pid$ && 2. $T_2 . color = Red$

Then from universal variable, the corresponding sid ($T = T_1 . sid$) will be retrieved."

**NOTE:**

- $\sigma_p(R \times S)$
- $R \bowtie_p S$
- $R \bowtie S$

All of this can be expressed using ∃ i.e. Existential Quantifier.

# Topic: Query Language

Supplier (Sid, rating)

Parts (Pid, pname, color)

Catalog (Sid, Pid, cost)

Retrieve Sid's of Suppliers who supplied at least two parts.

# Topic: Query Language

Supplier (Sid, rating)

Parts (Pid, pname, color)

Catalog (Sid, Pid, cost)

Retrieve Sid's of Suppliers who supplied at least two parts.

RA Query

- $\Pi_{Sid}(Catalog \bowtie_{Sid=S \wedge pid \neq p} \rho_{S,P,C}(catalog))$

TRC Query

- $T/\{\exists T_1 \in Catalog\ \exists T_2 \in Catalog\ (T_1 \cdot sid = T_2 . sid \wedge T_1 . pid \neq T_2 . pid \wedge T = T_1 . sid)\}$

$\Pi_{Sid}(Catalog \bowtie_{Sid=S \wedge pid \neq p} \rho_{S,P,C}(catalog))$

We can express this type of conditional JOIN query using EXISTENTIAL QUANTIFICATION.

- NOTE that whatever attribute we are trying to project for final result, we will use free variable for that.
- $T/\exists T_1 \in Catalog\ \exists T_2 \in Catalog$

$$(T_1 \cdot sid = T_2 . sid \wedge$$

$$T_1 . pid \neq T_2 . pid \wedge$$

$$\underbrace{T = T_1 . sid)\}}_{P(T)}$$

**Set Operators:**

OR

$$R \cup S = \{T \,/\, T \in R \cup T \in S\}$$

AND

$$R \cap S = \{T \,/\, T \in R \cap T \in S\}$$

But Not

$$R - S = \{T \,/\, T \in R \wedge \neg T \in S\}$$

# Topic: Query Language

Supplier (Sid, rating)

Parts (Pid, pname, color)

Catalog (Sid, Pid, cost)

Retrieve Supplier id's whose rating is maximum.

# Topic: Query Language

Supplier (Sid, rating)

Parts (Pid, pname, color)

Catalog (Sid, Pid, cost)

Retrieve Supplier id's whose rating is maximum.

**RA Query:** $\Pi_{sid}$ (Supplier)$-\Pi_{sid}$(Supplier $\bowtie_{rating < R}$ $\rho_{S,R}$(Supplier))

# Topic: Query Language

Supplier (Sid, rating)

Parts (Pid, pname, color)

Catalog (Sid, Pid, cost)

Retrieve Supplier id's whose rating is maximum.

**RA Query:** $\Pi_{sid}$ (Supplier) − $\Pi_{sid}$(Supplier $\bowtie_{rating < R}$ $\rho_{S,R}$(Supplier))

**TRC Query:**

$\{T / \exists T_1 \in Suppliers \nexists T_2 \in Suppliers\ (T_1 . rating \geq T_2 . rating \wedge T . Sid = T_1 . Sid)\}$

# Topic: Query Language

Supplier (Sid, rating)

Parts (Pid, pname, color)

Catalog (Sid, Pid, cost)

**SQL Query:**

```
Select Sid
FROM Supplier
WHERE rating ≥ All (Select rating From Supplier)
```

Retrieve Supplier id's whose rating is maximum.

**RA Query:** $\Pi_{sid}$ (Supplier) − $\Pi_{sid}$(Supplier $\bowtie_{rating < R}$ $\rho_{S,R}$(Supplier))

**TRC Query:**

$\{T / \exists T_1 \in Suppliers\ \forall T_2 \in Suppliers\ (T_1 . rating \geq T_2 . rating \wedge T = T_1 . Sid)\}$

RA ≅ TRC

## Expressive Power:

- Basic RA queries expressive power = Safe TRC queries expressive power.
- Everything that can be expressed by RA can also be expressed by TRC and vice- versa.


## Basic RA Queries:

Queries which can be expressed using

$\pi, \times, \sigma, \rho, \cup, -, \cap, \bowtie, \bowtie, \rtimes, \ltimes, \rtimes, /$ or $-$

operators, these queries fall under Basic RA Queries.

# Topic: Query Language

SQL > RA

**Queries cannot be expressed using basic RA:**

- Count of records / Count of attribute values ✗
- Sum of attribute values
- AVG of attribute values.

# Topic: Query Language

We have done total of 5 aggregation in SQL:

1. Count
2. Sum
3. Avg
4. Minimum
5. Maximum

Minimum and Maximum can be expressed using Basic RA, but not Count, Sum or Avg.

**Ordering of records:**

We can't retrieve data based on some pattern.

## Ordering of records:

We can't retrieve data based on some pattern.

**Ex:**

Retrieve employee name based on Alphabetical order or Retrieve data based on Ascending order of their rating.

This is not possible to express using Relational Algebra.

SQL > RA

**RA doesn't give DUPLICATE records**

That is why we say, the result has distinct records

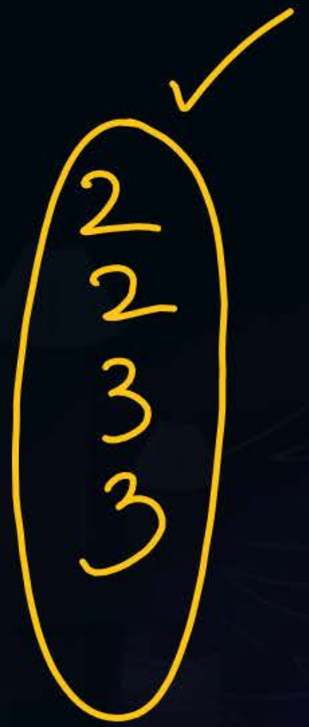
**EX:**

R

| A | B |
|---|---|
| 2 | 4 |
| 2 | 5 |
| 3 | 4 |
| 3 | 5 |

$\Pi_A(R)$ ⟶

| A |
|---|
| 2 |
| 3 |



# Inspiring Stories : Anthony Omari

***Background:*** **Kenyan man protecting orphans.**

***Struggles:*** **Raiders attacked the orphanage at night with knives.**

***Achievements:*** **Fought them off with just a hammer—twice—saving the children.**

***Impact:*** **Became a real-life guardian angel.**

# THANK - YOU