7-March : Indexing (9.00 PM)

: Class Update :

Shifting

Shifted

Fri (8 March) → Thu (7 March) 9.00 PM

Mon (11 March) → X OFF

Wed (13 March) → Resume

[Fri - Tue] ⇒ No class

Agenda :

- Subqueries
  - IN Clause
  - FROM Clause
- ALL , ANY
- Correlated Subsquery
- EXISTS
- Subquery in WHERE Clause
- Views

Sub-query

↳ breakdown a query into small queries and then combine the result to get complete answer.

students

id	name	psp	batchid

Q) Find out all students who have psp > max psp of all students in batch 2.

Algo:

$x \rightarrow$ max psp of all student in batch 2

ans = [ ]

$O(N + N)$

```
for s in students :
        if ( s.psp > x )
                ans.add(s)

return ans;
```

SQL :

```
SELECT MAX(psp)
FROM   students
WHERE  batch_id = 2
```

```
SELECT *
FROM students
WHERE psp > x
```

x

Combine

Final query

```
SELECT *
FROM students
WHERE psp > (SELECT MAX(psp)
             FROM students
             WHERE batch_id = 2),
```

may return a diff for every row

correlated subquery.

Theoretically $O(N^2)$

internal optimisation $O(N)$ possibly

TODO:

Q) Find out students whose psp > psp of student with id = 18.

```
SELECT *
FROM Students
WHERE psp > ( SELECT psp
              FROM students
              WHERE s_id = 18;
              )
```

-- Todo: Sakila Database
-- For the film table, find out all the years where
-- average rental_rate of films of that year was greater >=
-- than average rental_rate of all films

M1	2006	100
M2	2008	200
M3	2005	300
M4	2006	400

Avg Rental → 250/-

grouping acc to year

→ 2006 ⇒ 250 RS
→ ~~2008 ⇒ 200 RS~~
→ 2005 ⇒ 300 RS

filter out the groups

```
SELECT    AVG(rental_rate) AS avg, release_year
FROM      film
GROUP BY  release_year
HAVING    avg >= ( SELECT AVG(rental_rate)
                   FROM film );
```

→ Single value

→ Multiple values?

Concept - II **Subqueries and IN Clause**

⇒ Users can be either student or TA but not both.

id	name	is_student	is_TA
1	Varun	T	F
5	Deeksha	F	T
2	Deeksha	T	F
6	Varun	T	F
3	Rohan	F	T
7	Varun	F	T
4	Varun	F	T

(1,7)
(1,4)
~~(1,6)~~
~~(4,1)~~

Q) Find names of students that are also the names of TA.

{ Varun, Deeksha, Varun, } ⇒ { Varun, Deeksha }

TODO: Self-Join

```
SELECT DISTINCT S.name
FROM   users S
JOIN   users T
ON     S.name = T.name AND
       S.is_student = true AND
       T.is_TA = true;
```

Subquery:

```
SELECT DISTINCT name
FROM users U
WHERE U.is_student = true
AND U.name IN (SELECT DISTINCT name
               FROM users U
               WHERE U.is_TA = TRUE);
```

list

list of names

(ABC, XYZ, Varun)

Concept - III Subqueries in FROM Clause

Q) Find all students whose psp is not less than the smallest psp of any batch.

```
SELECT * FROM Students WHERE psp > (27
    SELECT max(psp) FROM (
        SELECT min(psp), batch-id
        FROM students
        GROUP BY batch-id
    ) min psp Table
```

B1	26
B2	18
B3	25

min psp Table

);

10.20

CONCEPT - IV

ALL and ANY

Find if a val is greater than all values in the set

AND

$X > \text{ALL}\{10, 20, 30, 60\}$

if all comparisons ret True

→ 55 NO

→ 65 yes.

$X > \text{ANY}\{10, 20, 30, 60\}$

T

OR

→ x = 15 Yes

→ x = 5 NO

27

```
SELECT * FROM students
WHERE psp >
ALL (SELECT min(psp)
     FROM students
     GROUP BY batch-id);
```

26
18
25

Smallest psp of every batch

What is output of `x = ANY(a, b, c)` same as

21 users have participated

A	=	5%
B	ANY	38%
C	IN	48%

Concept - V

Correlated Subqueries

Q) Find out all students whose psp > averge psp of students of their batch.

S1	B1	10	> 20	NO
S2	B1	20	> 30	NO
S3	B1	30	> 20	yes
S4	B2	40	> 30	yes
S5	B1	20	> 20	NO

	Avg
B1 →	20
B2 →	30

S3
S4

```
SELECT * FROM students S
WHERE psp > (SELECT avg(psp)
             FROM student
             WHERE batch_id = S.batch_id
            )
```

$$\frac{10 + 30 + 20}{3} \Rightarrow 20$$

EXISTS Clause

Students

id	name	psp
6	A	30

tas

id	name	st_id
3	A	6
4	B	null

can be NULL if TA is not student

Q) Find out all students who are also TAs.

I

```
SELECT * FROM students
WHERE id IN
(SELECT st_id
 FROM tas
 WHERE st_id IS NOT NULL;)
```

(5, 7, 32, 64)

II

```
SELECT *
FROM students S
WHERE EXISTS
( SELECT st_id
  FROM tas
  WHERE tas.st_id = S.id );
```

optimal query in terms of memory and space

atleast 1 Row then student from outer query will be selected.

Lookup faster (Indexing)

III

```
SEECT  *
FROM   students S
JOIN   tas
 ON    S.id = tas.st_id
```

More memory