GLA UNIVERSITY

Accredited with A+ Grade by NAAC

Mathura | Greater Noida

Department of Computer Engineering & Applications

Institute of Engineering & Technology

Advanced SQL & Query Optimization

Dr. Medha Kirti

Agenda

- Nested Queries
- Correlated Subqueries
- Set Operations
    - Union
    - Intersect
    - Minus
- Query Optimization Basics
    - Indexes
    - Execution Plans

Nested Queries

Definition: A query within another SQL query.

Structure:

SELECT ... FROM ... WHERE column = (SELECT ... FROM ...);

Example:

```
SELECT employee_id, name
FROM employees
WHERE department_id IN (SELECT department_id FROM departments
WHERE location_id = 1000);
```

Nested Queries

Employees:

EmpID	Name	Salary
1	Alice	5000
2	Bob	7000
3	Carol	6000

Task: Find employees who earn **more than the average salary**.

```
SELECT Name, Salary
FROM Employees
WHERE Salary > (
   SELECT AVG(Salary)
   FROM Employees
);
```

Nested Queries

How it Works:

1. Inner Query:

```
SELECT AVG(Salary) FROM Employees;
```

→ Calculates average salary (say it's 6000)

2. Outer Query:

```
SELECT Name, Salary FROM Employees WHERE Salary > 6000;
```

→ Returns employees with salary above 6000 → **Bob**.

Output:

Name	Salary
Bob	7000

Correlated Subqueries

Definition: A subquery that references columns from the outer query.

Structure:

```
SELECT ...
FROM OuterTable AS o
WHERE EXISTS (
   SELECT ...
   FROM InnerTable AS i
   WHERE i.column = o.column
);
```

Correlated Subqueries

Employees:

EmpID	Name	DeptID	Salary
1	Alice	10	5000
2	Bob	10	7000
3	Carol	20	6000
4	Dave	20	5500

Task: Find employees who **earn more than the average salary of their department**.

```
SELECT Name, Salary, DeptID
FROM Employees AS E
WHERE Salary > (
   SELECT AVG(Salary)
   FROM Employees AS E2
   WHERE E2.DeptID = E.DeptID
);
```

Correlated Subqueries

How it Works:

- For **each employee**, the subquery calculates the **average salary** in their **own department**.
- Then compares their salary with that average.

Output:

Name	Salary	DeptID
Bob	7000	10

Interview Question

What is the difference between Normal Subquery and Correlated Subquery?

Normal (Non-Correlated) Subquery

🔶**How It Works:**

- The **inner query runs once**.
- The result is then used by the outer query.
- The inner query is **independent** of the outer query.

Correlated Subquery

🔶**How It Works:**

- The **inner query runs once for each row** of the outer query.
- The inner query **depends on the current row** of the outer query.
- Slower but more powerful and flexible.

Interview Question

Describe a real-life scenario where you would use a correlated subquery instead of a JOIN.

Scenario: Find employees earning more than the average of their own department.
Why not JOIN? Because we need the **average per department**, and compare it **row by row**.

JOINs are better for combining related tables, but correlated subqueries are better for row-wise comparisons.

Interview Question

Which of the following queries finds employees who earn more than the average salary in their own department? (Amazon)

1. A. Uses a simple GROUP BY
B. Uses a correlated subquery
C. Uses a JOIN only
D. Uses UNION

Answer: B — Requires a correlated subquery comparing each employee to their department's average.

Set Operations

Overview: Operations that combine results from two or more queries.

Types:

- Union
- Intersect
- Minus

⚠ Important Rules:

•The **number and order of columns must match** in both SELECT queries.

•The **data types** of the columns must be compatible

Union

Definition: Combines the results of two queries, removing duplicates.

Example:

```
SELECT name FROM employees
UNION
SELECT name FROM contractors;
```

Intersect

Definition: Returns only the rows that are present in both queries.

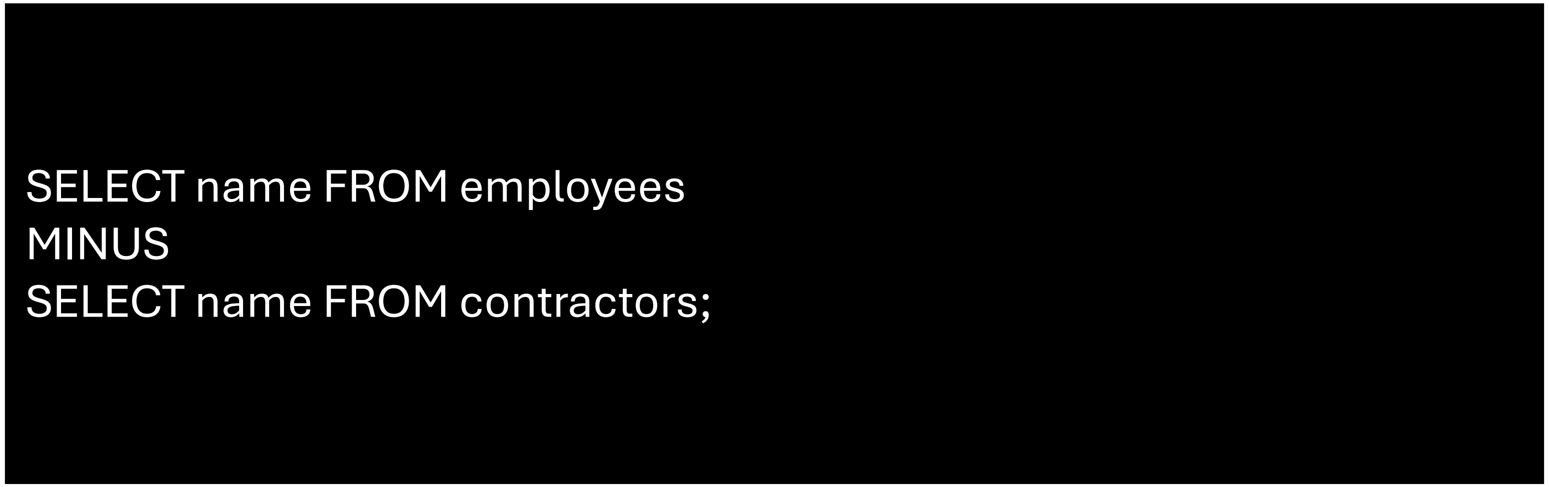
Example:

```
SELECT name FROM employees
INTERSECT
SELECT name FROM contractors;
```

Minus

Definition: Returns rows from the first query that are not in the second.

Example:



```
SELECT name FROM employees
MINUS
SELECT name FROM contractors;
```

Quick Summary Table

Operation	Purpose	Removes Duplicates?	Order of Columns Must Match?
UNION	Combine and return distinct values	Yes	Yes
UNION ALL	Combine and return all values	No	Yes
INTERSECT	Return common records	Yes	Yes
MINUS	Return records from first query only	Yes	Yes

Interview Question

Can we use INTERSECT and MINUS in all databases?

No, not all relational databases support INTERSECT and MINUS directly.

Database System	INTERSECT	MINUS / EXCEPT	Notes
Oracle	✅ Supported	✅ MINUS is used	Fully supported
PostgreSQL	✅ Supported	✅ EXCEPT is used	Fully supported
SQL Server	✅ Supported	✅ EXCEPT is used	Fully supported
MySQL (before v8.0)	❌ Not Supported	❌ Not Supported	Use JOIN, IN, or NOT EXISTS instead
MySQL 8.0+	❌ Not Supported	❌ Not Supported	Still lacks direct support
SQLite	✅ Supported	✅ EXCEPT is used	Works with some limitations

Interview Question

Explain a scenario where using INTERSECT is better than using INNER JOIN.

Use INTERSECT when you only need common values from identical columns in two tables without needing to combine additional columns. It is shorter and cleaner than a JOIN followed by DISTINCT.

🔷 Scenario:

Suppose you are analyzing customer behavior for an e-commerce platform (like Amazon or Flipkart). You want to find the customer_ids that appear in both the Online_Customers and Store_Customers tables — meaning customers who have shopped through both channels.

You do not need other details like names or addresses — just the matching IDs.

Using INTERSECT:

```
SELECT customer_id FROM
Online_Customers
INTERSECT
SELECT customer_id FROM
Store_Customers;
```

✅ This gives you only the common customer_ids.

✅ No duplicates (by default).

✅ Very readable and clear intent.

Using INNER JOIN:

```
SELECT DISTINCT o.customer_id
FROM Online_Customers o
INNER JOIN Store_Customers s
ON o.customer_id = s.customer_id;
```

✅ Same result, but requires:

- Explicit aliasing
- DISTINCT to remove duplicates (if needed)

Interview Question

What would happen if the columns selected in a set operation do not match in type or number?

The SQL engine will return an error. Set operations require the queries to return the **same number of columns** with **compatible data types**.

Query Optimization Basics

Importance: Enhances performance and efficiency of SQL queries.

Key Concepts:

- Indexes
- Execution Plans

Indexes

Definition: A database structure that improves the speed of data retrieval.

Types:

- Single-column Index
- Composite Index

```
CREATE INDEX idx_employee_name ON employees(name);
```

Execution Plans

Definition: A detailed breakdown of how a SQL query will be executed.

How to View:

- Use ‘EXPLAIN’ keyword before your query.

Example:

```
EXPLAIN SELECT * FROM employees WHERE department_id = 10;
```

Interview Question

What is the role of an index in query optimization?

An index speeds up data retrieval by allowing the database to avoid full table scans. It works like a lookup system and improves performance for SELECT queries.

Interview Question

Describe a scenario at Amazon where query optimization is crucial.

During product search, Amazon needs to retrieve millions of items fast. Indexes on product_name, rating, and price allow optimized search filtering.

Why Query Optimization Matters

- When a user searches or filters products (e.g., “wireless headphones under $50 with 4+ star ratings”), Amazon’s database must:
- Quickly fetch relevant results
- Sort and filter efficiently
- Handle millions of simultaneous queries

Interview Question

Explain a situation where adding an index can decrease performance.

1. Frequent INSERT, UPDATE, DELETE operations

Why?

- Every time a row is **inserted, updated, or deleted**, the database must **update the index** as well.
- This **adds overhead** and slows down write operations.

2. Too Many Indexes on a Table

Why?

- Each index must be maintained separately.
- More indexes = more disk I/O = slower performance during **data modification**.

Thank You