Joins AND SubQuery

By- Saurabh Kumar Sharma

Course Objective

- To understand Joins and its types
- To understand Subquery and its types

Session Objective

- Joins
  - Inner Join
  - Self Join
  - Outer Join
    - Left outer join
    - Right outer join
- Subquery
  - Single row subquery
  - Multiple row subquery

Obtaining Data from Multiple Tables

Employee

Employee_ID	Last_Name	Department_ID
1001	Lanson	10
1002	Scott	20
1003	Toyota	30
1004	Hunda	10

Department

Department_ID	Department_Name
10	Admin
20	Fin
30	HR
40	RMG

Employee_ID	Department_ID	Department_Name
1001	10	Admin
1002	20	Fin
1003	30	HR
1004	10	Admin

Types Of Joins

- **(INNER) JOIN**: Returns records that have matching values in both tables

- **LEFT (OUTER) JOIN**: Return all records from the left table, and the matched records from the right table

- **RIGHT (OUTER) JOIN**: Return all records from the right table, and the matched records from the left table

Syntax:

SELECT ***table1.column, table2.column***

FROM ***table1, table2***

WHERE ***table1.column1 = table2.column2;***

Tables used for joins –Order Table

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	2	7	1996-09-18	3
10309	37	3	1996-09-19	1
10310	77	8	1996-09-20	2

Tables used for joins - Customer Table

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

Inner Join

SELECT *column_name(s)*
FROM *table1*
INNER JOIN *table2* ON *table1.column_name* = *table2.column_name*;

Example:

```
SELECT employees.employee_id, employees.last_name,employees.department_id,
departments.department_id,departments.location_id
FROM employees, departments
WHERE employees.department_id =  departments.department_id
```

Creating Joins with the ON Clause

- **The join condition for the natural join is basically an equijoin of all columns with the same name.**
- **To specify arbitrary conditions or specify columns to join, the ON clause is used.**
- **The join condition is separated from other *search* conditions.**
- **The ON clause makes code easy to understand.**

Example :

```
SELECT employee_id, city, department_name
FROM employees e
JOIN departments d
ON d.department_id = e.department_id
JOIN locations l
ON d.location_id = l.location_id;
```

Qualifying Ambiguous Column Names

- **Use table prefixes to qualify column names that are in multiple tables.**
- **Improve performance by using table prefixes.**
- **Distinguish columns that have identical names but reside in different tables by using column aliases.**

Example:

```
SELECT Orders.OrderID, Customers.CustomerName
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

Join – More than two table

```
SELECT Orders.OrderID, Customers.CustomerName,
Shippers.ShipperName
FROM ((Orders
INNER JOIN Customers ON Orders.CustomerID =
Customers.CustomerID)
INNER JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID);
```

Left join

The LEFT JOIN keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side, if there is no match.

Syntax:

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2 ON table1.column_name = table2.column_name;
```

Example :

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.CustomerName;
```

Right join

The RIGHT JOIN keyword returns all records from the right table (table2), and the matched records from the left table (table1). The result is NULL from the left side, when there is no match.

Syntax:

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2 ON table1.column_name = table2.column_name;
```

Example :

```
SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
FROM Orders
RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
ORDER BY Orders.OrderID;
```

Self Joins

A self JOIN is a regular join, but the table is joined with itself.

Employee (Worker)

Employee_ID	Last_Name	Manager_ID
1001	Lanson	1002
1002	Scott	1003
1003	Toyota	1002
1004	Hunda	1001

Employee (Manager)

Employee_ID	Last_Name
1001	Lanson
1002	Scott
1003	Toyota
1004	Hunda

Manager_ID in the WORKER is Equal to Employee_ID in the MANAGER table

Example:

```
SELECT worker.last_name || ' works for '
|| manager.last_name
FROM employees worker join employees manager
on (worker.manager_id = manager.employee_id) ;
```

Sub Query

Using a Sub query to Solve a Problem

Who has Salary greater than Lanson's salary?

Main Query

Which employees have salaries greater than Lanson's salary?

Sub Query

What is Lanson's salary?

Sub Query

A Subquery or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause.

A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.

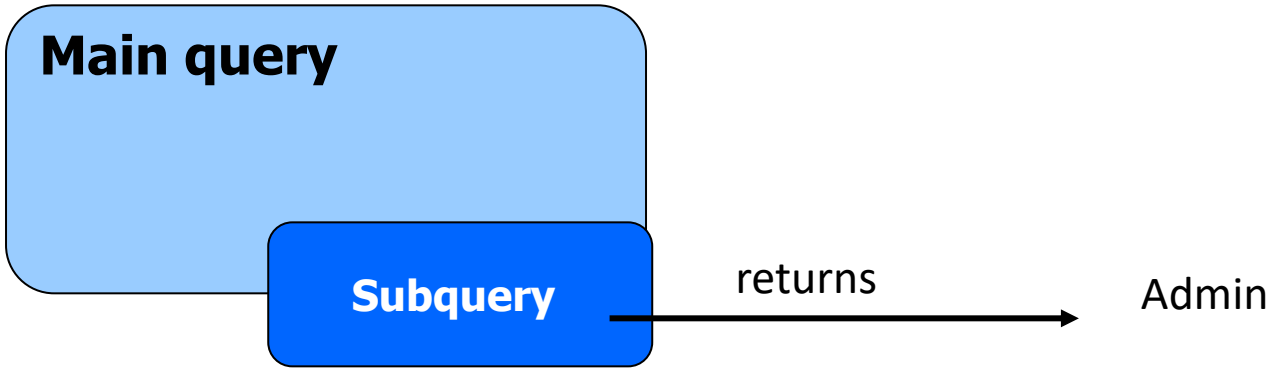
Guidelines

- Subqueries must be enclosed within parentheses.
- A subquery can have only one column in the SELECT clause, unless multiple columns are in the main query for the subquery to compare its selected columns.
- An ORDER BY command cannot be used in a subquery, although the main query can use an ORDER BY. The GROUP BY command can be used to perform the same function as the ORDER BY in a subquery.
- Subqueries that return more than one row can only be used with multiple value operators such as the IN operator.
- The SELECT list cannot include any references to values that evaluate to a BLOB, ARRAY, CLOB, or NCLOB.
- A subquery cannot be immediately enclosed in a set function.
- The BETWEEN operator cannot be used with a subquery. However, the BETWEEN operator can be used within the subquery.

Types of Sub queries

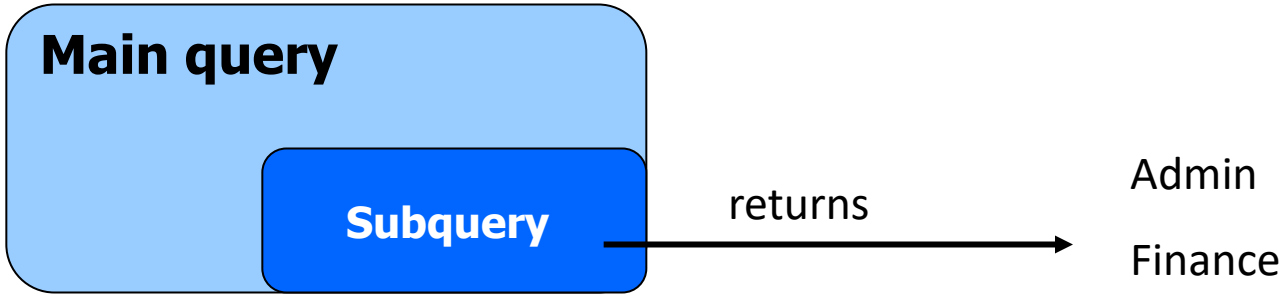
Single-row subqueries:

Queries that return only one row from the inner SELECT statement



Multiple-row subqueries:

Queries that return more than one row from the inner SELECT statement



Sub Query

The subquery (inner query) executes once before the main query.
The result of the subquery is used by the main query (outer query).

Syntax:

SELECT *select_list*
FROM *table*
WHERE *expr operator* (SELECT *select_list* FROM *table*);

Example:

```
SELECT last_name
FROM employees
WHERE salary >(SELECT salary FROM employees WHERE  last_name = 'Abel') ;
```

Single Row Sub query

- Returns only one row
- Use single row comparison operators
- sub queries and use multiple-row operators with multiple-row sub queries.

= , > , >= ,< , <= , <>

Single Row Sub query

Example 1:

```
SELECT last_name, job_id
FROM employees
WHERE job_id =(SELECT job_id FROM employees
                WHERE employee_id= 141);
```

Example 2:

```
SELECT last_name, job_id, salary
FROM employees
WHERE job_id = (SELECT job_id FROM employees
                WHERE employee_id = 141)
AND salary >     (SELECT salary FROM employees
                WHERE employee_id = 143);
```

Subqueries with the SELECT Statement

```
SELECT *
  FROM CUSTOMERS
  WHERE ID IN (SELECT ID
       FROM CUSTOMERS
       WHERE SALARY > 4500) ;
```

Group Functions in a Sub query

Example:

```
SELECT last_name, job_id, salary
FROM employees
WHERE salary =(SELECT MIN(salary)
                FROM employees);
```

The HAVING Clause with Sub queries

Example:

```
SELECT department_id, MIN(salary)
FROM employees
GROUP BY department_id
HAVING MIN(salary) > (SELECT MIN(salary)
                      FROM employees
                      WHERE department_id = 50);
```

Multiple-Row Sub queries

- Return more than one row
- Use multiple-row comparison operators:

IN : Equal to any member in the list

ANY : Compare value to each value returned by the subquery

ALL : Compare value to every value returned by the subquery

Multiple-Row Subqueries-IN Operator

Example:

```
SELECT last_name, salary, department_id
FROM employees
WHERE salary IN (2500, 4200, 4400, 6000, 7000, 8300, 8600, 17000);
(SELECT MIN(salary)
FROM employees
GROUP BY department_id);
```

Multiple-Row Subqueries-ANY Operator

Example:

```
SELECT employee_id,
last_name,job_id, salary
FROM employees
WHERE salary < ANY
(SELECT salary
FROM employees
WHERE job_id = 'IT_PROG')
AND job_id <> 'IT_PROG';
```

Inner query result= 9000,6000, 4200

<ANY means less than the maximum.

>ANY means more than the minimum.

Multiple-Row Sub queries-ALL Operator

Example:

```
SELECT employee_id, last_name,
job_id, salary
FROM employees
WHERE salary <ALL(SELECT salary
FROM employees
WHERE job_id = 'IT_PROG')
AND job_id <> 'IT_PROG';
```

Inner query result= 9000,6000, 4200

<ALL means less than the minimum.

>ALL means more than the maximum