Practical MySQL

Session 5

Subqueries

Session Overview

- Explain sub queries and their usage
- Outline the clauses and keywords used in subqueries
- Describe practical uses of `WHERE` and `FROM` clauses in subqueries
- Define `IN`, `NOT IN`, `EXISTS`, and `NOT EXISTS` keywords
- List and explain different types of subqueries

Subquery

A subquery is a nested query inside another query. It is also called an Inner Query or Nested Query.

It is also referred to as nested `SELECT` or `Sub-SELECT`.

Following is the basic syntax of a subquery:

```
SELECT column_name FROM table_1 WHERE column_name
expression operator (SELECT COLUMN_NAME FROM
TABLE_2 WHERE...);//Subquery
```

When to use Subquery?

Subqueries are used when some result must be fetched from the database that requires multiple query statements. In multiple queries, each query returns a subset of data from the table.

Following are guidelines to use subqueries:

- Subquery is used for membership questions.
- Subqueries are used when the user requires more than one query.
- Multiple queries can be formulated using subqueries

Clauses and Keywords in Subquery

There are several clauses and keywords, which are used with subqueries to solve complex queries.

`WHERE` and `FROM` form the basis of clauses used whereas, `IN`, `NOT IN`, `EXISTS`, and `NOT EXISTS` are the most used keywords.

For a better illustration of clauses and keywords in a subquery, consider two tables: **`Department`** and **`Employee`** tables.

The **`Department`** table must be created and data must be added before creating and adding data to the **`Employee`** table to avoid a foreign key constraint error.

WHERE Clause in a Subquery (1-2)

The WHERE clause in a sub-query is used to filter the rows from set of rows of the resultant set, depending on the condition.

It compares the column of the outer query with the results returned by the sub-query.

WHERE clause is used along with SELECT, INSERT, UPDATE, and DELETE.

WHERE Clause in a Subquery (2-2)

SELECT Statement: The WHERE clause compares the values returned by the subquery with the records in the table which is then displayed by the SELECT statement.

INSERT Statement: The WHERE clause compares the returned value by the subquery with the records in the table. It is selected by the SELECT statement and consequently inserted by the INSERT statement.

UPDATE Statement: The WHERE clause compares the values returned by the subquery with the records in the table. The records of the selected employees are then updated using the UPDATE statement.

DELETE Statement: The DELETE statement is used to delete the returned values of WHERE clause in the subquery.

FROM Clause

All the subqueries are placed inside the `WHERE` clause.

Subqueries used in `FROM` clause are useful for calculating aggregate values such as `SUM()`, `MAX()`, `MIN()`, `AVG()`, and `COUNT()`.

For example, in a corporate database system, a user wants to display the average expenses of the company department-wise.

IN and NOT IN keywords in the Subquery

The IN and NOT IN keywords are used to compare more than one value in a list with the records in the table. If a record matches with any value in the list (with IN clause), the particular row/s are returned as a result dataset.

IN Clause: Following query fetches the records of the employees who work in departments having a rank of either 1 or 2:

```
SELECT * FROM Employee WHERE Dept IN (SELECT DID FROM
Department WHERE DRanking = 1 OR DRanking = 2 OR DRanking
= 3);
```

NOT IN: Following query fetches the records of the employees who do not work in the departments located in Texas or Alaska:

```
SELECT * FROM Employee WHERE Dept NOT IN (SELECT DID FROM
Department WHERE DLocation = 'Texas' OR Dlocation =
'California');
```

EXISTS and NOT EXISTS

EXISTS and NOT EXISTS are operators that check whether rows exist or not in the database. It always returns a boolean value.

Exist

The EXISTS subquery returns TRUE if it returns a dataset containing one or more records. Following query finds the details of the departments which has at least one employee working under them:

```
SELECT * FROM Department d WHERE
EXISTS (SELECT * FROM Employee e
    WHERE e.Dept = d.DID);
```

Not Exist

The NOT EXISTS statement combines two operators: EXISTS and NOT. It is the opposite of EXISTS. The subquery returns true if it returns a dataset not containing any record.

```
SELECT * FROM Department d WHERE NOT
EXISTS (SELECT * FROM Employee e WHERE
e.Dept = d.DID);
```

Types of Subquery

Single Row Subquery: A single row subquery returns no record or a single record to the outer query.

A single row subquery can be formed by placing a `SELECT` statement in a `WHERE` clause, `FROM` clause, or `HAVING` clause of a statement. Such subqueries are always used with single row operators.

Following query displays the details of the employee having the highest salary in the table:

```
SELECT * FROM Employee WHERE Salary = (SELECT MAX(Salary)
FROM Employee);
```

Multiple Column Subquery

Following query displays the grouped dataset of maximum salary of each position (Clerk, Manager, and Salesman):

In multiple-column subqueries, the inner query returns a dataset of values of multiple columns to the outer query, which compares values of multiple columns with the dataset.

The dataset returned by the subquery is considered as a separate table. It is called a temporary table or inline view. If the subquery contains a `GROUP BY`, then the grouped data are treated as separate rows.

```
SELECT * FROM Employee WHERE (Position, Salary) IN (SELECT
Position,MAX(Salary) FROM Employee GROUP BY POSITION);
```

Correlated Sub-query

Correlated subqueries work in an opposite way to how generic subqueries work. In basic subqueries, the inner query returns values that are operated upon by the outer query.

In correlated subqueries, the subquery operates on the values provided by the outer query..

Correlated subqueries cannot be run independently. They contain references to table names mentioned in the outer queries and gives an error on execution.

Query to display the details of the employees with a salary higher than the average salary of their department:

```
SELECT * FROM Employee E WHERE Salary > (SELECT AVG(Salary) AS avgsal FROM Employee WHERE Dept = E.Dept);
```

Summary

- A subquery is a nested query.
- Basic subqueries return row/s to the outer query for comparison.
- Subqueries can be used with `WHERE`, `HAVING`, and `FROM` clauses.
- The `DELETE` statement is used to delete the returned values of the `WHERE` clause in the subquery.
- Subqueries used in `FROM` clause are useful for calculating aggregate values such as `SUM()`, `MAX()`, `MIN()`, `AVG()`, and `COUNT()`.
- Single row operators and multi-row operators are important as comparison operators.
- Correlated subqueries cannot run independently as they contain alias names of tables mentioned in the outer query.