My SQL Subqueries

Objectives

1. What is subqueries
2. Subquery with the SELECT statement
3. Subquery with the INSERT statement
4. Scalar subqueries
5. Table subqueries
6. Subqueries with EXISTS, NOT EXISTS

1. What is subqueries

A *subquery* is a query in a query. It is also called an **inner query** or a **nested query**. A subquery can be used anywhere an expression is allowed. It is a query expression enclosed in parentheses. Subqueries can be used with `SELECT`, `INSERT`, `UPDATE`, or `DELETE` statements.

There is more than one way to execute an SQL task. Many subqueries can be replaced by SQL joins. *SQL joins are usually faster*.

1. What is subqueries

We recapitulate what we have in the Customers and Reservations tables. Subqueries are often performed on tables, which have some relationship.

```
SELECT * FROM Customers;
+------------+-------------+
| CustomerId | Name        |
+------------+-------------+
|          1 | Tom Willis  |
|          2 | Terry Neils |
|          3 | Cindy Mason |
|          4 | Paul Novak  |
|          5 | Jack Fonda  |
+------------+-------------+
4 rows in set (0.00 sec)
```

```
SELECT * FROM Reservations;
+----+------------+------------+
| Id | CustomerId | Day        |
+----+------------+------------+
|  1 |          1 | 2009-11-22 |
|  2 |          2 | 2009-11-28 |
|  3 |          2 | 2009-11-29 |
|  4 |          1 | 2009-11-29 |
|  5 |          3 | 2009-12-02 |
+----+------------+------------+
5 rows in set (0.00 sec)
```

2. Subquery with the SELECT statement

To get all reservations that Paul Novak had made, we have to get Paul's Id, then use `WHERE` statement in Reservations table to find out:

```
mysql> SELECT Name, Day
    FROM Customers, Reservations
    WHERE Customers.CustomerId=Reservations.CustomerId
    AND Reservations.CustomerId=(SELECT CustomerId FROM Customers
                                 WHERE Name='Paul Novak');
```

Customers

CustomerId	Name
1	Tom Willis
2	Terry Neils
3	Cindy Mason
4	Paul Novak
5	Jack Fonda

Reservations

Id	CustomerId	Day
1	4	2009-11-22
2	2	2009-11-28
3	2	2009-11-29
4	4	2009-11-29
5	5	2009-12-02
6	2	2009-12-03
7	3	2009-12-04

---\
---/

Result

Name	Day
Paul Novak	2009-11-22
Paul Novak	2009-11-29

3. Subquery with the INSERT statement

We want to create a copy of the Cars table. Into another table called Cars2. We will create a subquery for this.

```
mysql> CREATE TABLE Cars2(
           Id INT NOT NULL PRIMARY KEY,
           Name VARCHAR(50) NOT NULL,
           Cost INT NOT NULL
       );
```

We create a new Cars2 table with the same columns and datatypes as the Cars table. To find out how a table was created, we can use the SHOW CREATE TABLE statement.

```
mysql> INSERT INTO Cars2 SELECT * FROM Cars;
```

This is a simple subquery. We insert all rows from the Cars table into the Cars2 table.

3. Subquery with the INSERT statement

The data was copied to a new Cars2 table.

```
mysql> INSERT INTO Cars2 SELECT * FROM Cars;
```

```
mysql> SELECT * FROM Cars;
+----+------------+--------+
| Id | Name       | Cost   |
+----+------------+--------+
|  1 | Audi       |  52642 |
|  2 | Mercedes   |  57127 |
|  3 | Skoda      |   9000 |
|  4 | Volvo      |  29000 |
|  5 | Bentley    | 350000 |
|  6 | Citroen    |  21000 |
|  7 | Hummer     |  41400 |
|  8 | Volkswagen |  21600 |
+----+------------+--------+
8 rows in set (0.00 sec)
```

Then INSERT INTO table Cars2 =>

```
+--------------------------+
| Cars2                    |
+----+------------+--------+
| Id | Name       | Cost   |
+----+------------+--------+
|  1 | Audi       |  52642 |
|  2 | Mercedes   |  57127 |
|  3 | Skoda      |   9000 |
|  4 | Volvo      |  29000 |
|  5 | Bentley    | 350000 |
|  6 | Citroen    |  21000 |
|  7 | Hummer     |  41400 |
|  8 | Volkswagen |  21600 |
+----+------------+--------+
```

4. Scalar subqueries

A scalar subquery returns a single value.

```
mysql> SELECT Name FROM Customers WHERE
    -> CustomerId=(SELECT CustomerId FROM Reservations WHERE Id=5);
+------------+
| Name       |
+------------+
| Jack Fonda |
+------------+
```

The query enclosed in parentheses is the subquery. It returns one single scalar value. The returned value is then used in the outer query. In this scalar subquery, we return the name of the customer from the Customers table, whose reservation has Id equal to 5 in the Reservations table.

5. Table subqueries

A table subquery returns a result table of zero or more rows.

```
mysql> SELECT Name FROM Customers WHERE CustomerId IN (SELECT DISTINCT CustomerId FROM Reservations);
```

```
SELECT DISTINCT CustomerId
FROM Reservations;
+------------+
| CustomerId |
+------------+
|          4 |
|          2 |
|          5 |
|          3 |
+------------+
```

=> {4, 2, 5, 3} =>

```
WHERE CustomerId IN {4, 2, 5, 3}
+-------------+
| Name        |
+-------------+
| Paul Novak  |
| Terry Neils |
| Jack Fonda  |
| Cindy Mason |
+-------------+
```

5. Table subqueries

The above query returns the names of the customers, who made some reservations. The inner query returns customer Ids from the Reservations table. We use the IN predicate to select those names of customers, who have their CustomerId returned from the inner select query.

```
mysql> SELECT DISTINCT Name FROM Customers JOIN Reservations ON Customers.CustomerId=Reservations.CustomerId;
```

```
+-------------+
| Name        |
+-------------+
| Paul Novak  |
| Terry Neils |
| Jack Fonda  |
+-------------+
```

The previous subquery can be rewritten using SQL join.

6. Subqueries with EXISTS

If a subquery returns any values, then the predicate EXISTS returns TRUE, and NOT EXISTS FALSE.

```
SELECT Name FROM Customers WHERE EXISTS (SELECT * FROM Reservations WHERE Customers.CustomerId=Reservations.CustomerId);
```

Customers

CustomerId	Name
1	Tom Willis
2	Terry Neils
3	Cindy Mason
4	Paul Novak
5	Jack Fonda

Reservations

Id	CustomerId	Day
1	4	2009-11-22
2	2	2009-11-28
3	2	2009-11-29
4	4	2009-11-29
5	5	2009-12-02
6	2	2009-12-03
7	3	2009-12-04

---\
---/

Result

Name
Paul Novak
Cindy Mason
Terry Neils
Jack Fonda

6. Subqueries with NOT EXISTS

In the above SQL statement we select all customers' names, which have an entry in the Reservations table..

```
mysql> SELECT Name FROM Customers WHERE NOT EXISTS (SELECT * FROM Reservations WHERE Customers.CustomerId=Reservations.CustomerId);
```

```
+------------+
| Name       |
+------------+
| Tom Willis |
+------------+
```

In this query, we return all customers that do not have an entry in the Reservations table. Both SQL queries are correlated queries.