MySQLTUTORIAL

MySQL EXISTS

Summary: in this tutorial, you will learn how to use the MySQL `EXISTS` operator and when to use it to improve the performance of the queries.

Introduction to MySQL EXISTS operator

The `EXISTS` operator is a Boolean operator that returns either true or false. The `EXISTS` operator is often used to test for the existence of rows returned by the subquery (https://www.mysqltutorial.org/mysql-subquery/) .

The following illustrates the basic syntax of the `EXISTS` operator:

```
SELECT
    select_list
FROM
    a_table
WHERE
    [NOT] EXISTS(subquery);
```

If the subquery returns at least one row, the `EXISTS` operator returns true, otherwise, it returns false.

In addition, the `EXISTS` operator terminates further processing immediately once it finds a matching row, which can help improve the performance of the query.

The `NOT` operator negates the `EXISTS` operator. In other words, the `NOT EXISTS` returns true if the subquery returns no row, otherwise it returns false.

Note that you can use `SELECT *`, `SELECT column`, `SELECT a_constant`, or anything in the subquery. The results are the same because MySQL ignores the select list appeared in the `SELECT (https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx)` clause.

MySQL EXISTS operator examples

Let's take some examples of using the `EXISTS` operator to understand how it works.

MySQL SELECT EXISTS examples

Consider the following `customers` and `orders` tables in the sample database (https://www.mysqltutorial.org/mysql-sample-database.aspx) .

The following statement uses the `EXISTS` operator to find the customer who has at least one order:

```
SELECT
    customerNumber,
    customerName
FROM
    customers
WHERE
    EXISTS(
        SELECT
            1
        FROM
            orders
        WHERE
            orders.customernumber
                = customers.customernumber);
```

Try It Out

In this example, for each row in the `customers` table, the query checks the `customerNumber` in the `orders` table. If the `customerNumber`, which appears in the `customers` table, exists in the `orders` table, the subquery returns the first matching row. As a result, the `EXISTS` operator returns true and stops examining the `orders` table. Otherwise, the subquery returns no row and the `EXISTS` operator returns false.

The following example uses the `NOT EXISTS` operator to find customers who do not have any orders:

```
SELECT
    customerNumber,
    customerName
FROM
    customers
WHERE
    NOT EXISTS(
        SELECT
            1
        FROM
            orders
        WHERE
            orders.customernumber = customers.customernumber
        );
```

Try It Out

MySQL UPDATE EXISTS examples

Suppose that you have to update the phone's extensions of the employees who work at the office in San Francisco.

The following statement finds employees who work at the office in `San Franciso`:

```
SELECT
    employeenumber,
    firstname,
    lastname,
    extension
FROM
    employees
WHERE
    EXISTS(
        SELECT
            1
        FROM
            offices
        WHERE
            city = 'San Francisco' AND
           offices.officeCode = employees.officeCode);
```

Try It Out

This example adds the number 1 to the phone extension of employees who work at the office in San Francisco:

```
UPDATE employees
SET
    extension = CONCAT(extension, '1')
WHERE
    EXISTS(
        SELECT
            1
        FROM
            offices
        WHERE
            city = 'San Francisco'
                AND offices.officeCode = employees.officeCode);
```

How it works.

- First, the `EXISTS` (https://www.mysqltutorial.org/mysql-exists/) operator in the `WHERE` (https://www.mysqltutorial.org/mysql-where/) clause gets only employees who works at the office in San Fransisco.
- Second, the `CONCAT()` (https://www.mysqltutorial.org/sql-concat-in-mysql.aspx) function concatenate the phone extension with the number 1.

MySQL INSERT EXISTS example

Suppose that you want to archive customers who don't have any sales order in a separate table. To do this, you use these steps:

First, create a new table (https://www.mysqltutorial.org/mysql-create-table/) for archiving the customers by copying (https://www.mysqltutorial.org/mysql-copy-table-data.aspx) the structure from the `customers` table:

```
CREATE TABLE customers_archive
LIKE customers;
```

Second, insert customers who do not have any sales order into the `customers_archive` table using the following `INSERT (https://www.mysqltutorial.org/mysql-insert-statement.aspx)` statement.

```
INSERT INTO customers_archive
SELECT *
FROM customers
WHERE NOT EXISTS(
   SELECT 1
   FROM
       orders
   WHERE
       orders.customernumber = customers.customernumber
);
```

Try It Out

Third, query data (https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx) from the `customers_archive` table to verify the insert operation.

```
SELECT * FROM customers_archive;
```

Try It Out

MySQL DELETE EXISTS example

One final task in archiving the customer data is to delete the customers that exist in the `customers_archive` table from the `customers` table.

To do this, you use the `EXISTS` operator in `WHERE` clause of the `DELETE` `(https://www.mysqltutorial.org/mysql-delete-statement.aspx)` statement as follows:

```
DELETE FROM customers
WHERE EXISTS(
    SELECT
        1
    FROM
        customers_archive a

    WHERE
        a.customernumber = customers.customerNumber);
```

MySQL EXISTS operator vs. IN operator

To find the customer who has placed at least one order, you can use the `IN` `(https://www.mysqltutorial.org/mysql-basics/mysql-in/)` operator as shown in the following query:

```
SELECT
    customerNumber,
    customerName
FROM
    customers
WHERE
    customerNumber IN (
        SELECT
            customerNumber
        FROM
            orders);
```

Try It Out

Let’s compare the query that uses the `IN` operator with the one that uses the `EXISTS` operator by using the `EXPLAIN` statement.

```
EXPLAIN SELECT
    customerNumber,
    customerName
FROM
    customers
WHERE
    EXISTS(
        SELECT
            1
        FROM
            orders
        WHERE
            orders.customernumber = customers.customernumber);
```

Now, check the performance of the query that uses the `IN (https://www.mysqltutorial.org/mysql-basics/mysql-in/)` operator.

```
SELECT
    customerNumber, customerName
FROM
    customers
WHERE
```

```
    customerNumber IN (SELECT
            customerNumber
        FROM
            orders);
```

Try It Out

The query that uses the `EXISTS` operator is much faster than the one that uses the `IN` operator.

The reason is that the `EXISTS` operator works based on the “at least found” principle. The `EXISTS` stops scanning the table when a matching row found.

On the other hands, when the `IN` operator is combined with a subquery, MySQL must process the subquery first and then uses the result of the subquery to process the whole query.

The general rule of thumb is that if the subquery contains a large volume of data, the `EXISTS` operator provides better performance.

However, the query that uses the `IN` operator will perform faster if the result set returned from the subquery is very small.

For example, the following statement uses the `IN` operator selects all employees who work at the office in San Francisco.

```
SELECT
    employeenumber,
    firstname,
    lastname
FROM
    employees
WHERE
    officeCode IN (SELECT
            officeCode
        FROM
            offices
        WHERE
            offices.city = 'San Francisco');
```

Try It Out

Let's check the performance of the query.

It is a little bit faster than the query that uses the `EXISTS` operator that we mentioned in the first example. See the performance of the query that uses the `EXIST` operator below:

In this tutorial, you have learned how to use the MySQL `EXISTS` operator to test for the existence of rows returned by a subquery.