How to Use EXISTS, UNIQUE, DISTINCT, and OVERLAPS in SQL Statements

Related Book
SQL For Dummies, 8th Edition
By Allen G. Taylor

Within the WHERE clause lies many possibilities for modifying your SQL statement. Among these possibilities are the EXISTS, UNIQUE, DISTINCT, and OVERLAPS predicates. Here are some examples of how to use these in your SQL statements.

EXISTS

You can use the EXISTS predicate in conjunction with a subquery to determine whether the subquery returns any rows. If the subquery returns at least one row, that result satisfies the EXISTS condition, and the outer query executes. Consider the following example:

```
SELECT FirstName, LastName
FROM CUSTOMER
WHERE EXISTS
  (SELECT DISTINCT CustomerID
   FROM (SELECT DISTINCT CustomerID
         FROM CUSTOMER));
```
Here the SALES table contains all of your company’s sales transactions. The table includes the `CustomerID` of the customer who makes each purchase, as well as other pertinent information. The CUSTOMER table contains each customer’s first and last names, but no information about specific transactions.

The subquery in the preceding example returns a row for every customer who has made at least one purchase. The outer query returns the first and last names of the customers who made the purchases that the SALES table records.

`EXISTS` is equivalent to a comparison of `COUNT` with zero, as the following query shows:

```sql
SELECT FirstName, LastName
FROM CUSTOMER
WHERE 0 <>
  (SELECT COUNT(*)
   FROM SALES
   WHERE SALES.CustomerID = CUSTOMER.CustomerID);
```

For every row in the SALES table that contains a `CustomerID` that’s equal to a `CustomerID` in the CUSTOMER table, this statement displays the `FirstName` and `LastName` columns in the CUSTOMER table. For every sale in the SALES table, therefore, the statement displays the name of the customer who made the purchase.
As you do with the `EXISTS` predicate, you use the `UNIQUE` predicate with a subquery. Although the `EXISTS` predicate evaluates to True only if the subquery returns at least one row, the `UNIQUE` predicate evaluates to True only if no two rows returned by the subquery are identical. In other words, the `UNIQUE` predicate evaluates to True only if all the rows that its subquery returns are unique.

Consider the following example:

```sql
SELECT FirstName, LastName
FROM CUSTOMER
WHERE UNIQUE
  (SELECT CustomerID FROM SALES
   WHERE SALES.CustomerID = CUSTOMER.CustomerID);
```

This statement retrieves the names of all new customers for whom the `SALES` table records only one sale. Because a null value is an unknown value, two null values aren’t considered equal to each other; when the `UNIQUE` keyword is applied to a result table that contains only two null rows, the `UNIQUE` predicate evaluates to True.

**DISTINCT**

The `DISTINCT` predicate is similar to the `UNIQUE` predicate, except in the way it treats nulls. If all the values in a result table are `UNIQUE`, then they’re also `DISTINCT` from each other.

However, unlike the result for the `UNIQUE` predicate, if the `DISTINCT` keyword is applied to a result table that contains only two null rows, the `DISTINCT` predicate evaluates to False. Two null
values are not considered distinct from each other, while at the same time they are considered to be unique.

This strange situation seems contradictory, but there’s a reason for it. In some situations, you may want to treat two null values as different from each other — in which case, use the `UNIQUE` predicate. When you want to treat the two nulls as if they’re the same, use the `DISTINCT` predicate.

**OVERLAPS**

You use the `OVERLAPS` predicate to determine whether two time intervals overlap each other. This predicate is useful for avoiding scheduling conflicts. If the two intervals overlap, the predicate returns a True value. If they don’t overlap, the predicate returns a False value.

You can specify an interval in two ways: either as a start time and an end time or as a start time and a duration. Here are some examples:

```
(TIME '2:55:00', INTERVAL '1' HOUR)
OVERLAPS
(TIME '3:30:00', INTERVAL '2' HOUR)
```

This first example returns a True because 3:30 is less than one hour after 2:55.

```
(TIME '9:00:00', TIME '9:30:00')
OVERLAPS
(TIME '9:29:00', TIME '9:31:00')
```
This example returns a True because you have a one-minute overlap between the two intervals.

(TIME '9:00:00', TIME '10:00:00') OVERLAPS (TIME '10:15:00', INTERVAL '3' HOUR)

This example returns a False because the two intervals don’t overlap.

(TIME '9:00:00', TIME '9:30:00') OVERLAPS (TIME '9:30:00', TIME '9:35:00')

This example returns a False because even though the two intervals are contiguous, they don’t overlap.