MySQLTUTORIAL

MySQL INTERSECT

Summary: in this tutorial, we will introduce you to the `INTERSECT` operator and show you how to emulate the MySQL `INTERSECT` operator.

> Note that MySQL does not support the `INTERSECT` operator. This tutorial introduces you to how to emulate the `INTERSECT` operator in MySQL using join clauses.

Introduction to the INTERSECT operator

The `INTERSECT` operator is a set operator that returns only distinct rows of two queries or more queries.

The following illustrates the syntax of the `INTERSECT` operator.

```
(SELECT column_list
FROM table_1)
INTERSECT
(SELECT column_list
FROM table_2);
```

The `INTERSECT` operator compares the result sets of two queries and returns the distinct rows that are output by both queries.

To use the `INTERSECT` operator for two queries, you follow these rules:

1. The order and the number of columns in the select list of the queries must be the same.
2. The data types of the corresponding columns must be compatible.

The following diagram illustrates the `INTERSECT` operator.

The left query produces a result set of (1,2,3).

The right query returns a result set of (2,3,4).

The `INTERSECT` operator returns the distinct rows of both result sets which include (2,3).

Unlike the `UNION (https://www.mysqltutorial.org/sql-union-mysql.aspx)` operator, the `INTERSECT` operator returns the intersection between two circles.

Note that the SQL standard has three set operators that include `UNION (https://www.mysqltutorial.org/sql-union-mysql.aspx)` , `INTERSECT` , and `MINUS (https://www.mysqltutorial.org/mysql-minus/)` .

Emulating INTERSECT in MySQL

Unfortunately, MySQL does not support the `INTERSECT` operator. However, you can emulate the `INTERSECT` operator.

Setting up sample tables

The following statements create tables (https://www.mysqltutorial.org/mysql-create-table/) `t1` and `t2` , and then insert data (https://www.mysqltutorial.org/mysql-insert-statement.aspx) into both tables.

```
CREATE TABLE t1 (
    id INT PRIMARY KEY
);

CREATE TABLE t2 LIKE t1;
```

```
INSERT INTO t1(id) VALUES(1),(2),(3);

INSERT INTO t2(id) VALUES(2),(3),(4);
```

The following query returns rows from the `t1` table.

```
SELECT id FROM t1;
```

```
id
----
1
2
3
```

The following query returns the rows from the `t2` table:

```
SELECT id
FROM t2;
```

```
id
---
2
3
4
```

1) Emulate INTERSECT using DISTINCT and INNER JOIN clause

The following statement uses `DISTINCT` (https://www.mysqltutorial.org/mysql-distinct.aspx) operator and `INNER JOIN` (https://www.mysqltutorial.org/mysql-inner-join.aspx) clause to return the distinct rows in both tables:

```
SELECT DISTINCT
   id
FROM t1
   INNER JOIN t2 USING(id);
```

```
id
----
2
3
```

How it works.

1. The `INNER JOIN` clause returns rows from both left and right tables.
2. The `DISTINCT` operator removes the duplicate rows.

2) Emulate INTERSECT using IN and subquery

The following statement uses the `IN` (https://www.mysqltutorial.org/mysql-basics/mysql-in/) operator and a subquery (https://www.mysqltutorial.org/mysql-subquery/) to return the intersection of the two result sets.

```
SELECT DISTINCT id
FROM t1
WHERE id IN (SELECT id FROM t2);
```

```
id
----
2
3
```

How it works.

1. The subquery returns the first result set.
2. The outer query uses the `IN` operator to select only values that exist in the first result set. The `DISTINCT` operator ensures that only distinct values are selected.

In this tutorial, you have learned a couple of ways to simulate the `INTERSECT` operator in MySQL.