MySQLTUTORIAL

How To Rename Table Using MySQL RENAME TABLE Statement

Summary: in this tutorial, you will learn how to rename tables using MySQL RENAME TABLE statement and ALTER TABLE statement.

Introduction to MySQL RENAME TABLE statement

Because business requirements change, we need to rename the current table to a new one to better reflect the new situation. MySQL provides us with a very useful statement that changes the name of one or more tables.

To change one or more tables, we use the `RENAME TABLE` statement as follows:

```
RENAME TABLE old_table_name TO new_table_name;
```

The old table (`old_table_name`) must exist, and the new table (`new_table_name`) must not. If the new table `new_table_name` does exist, the statement will fail.

In addition to the tables, we can use the `RENAME TABLE` statement to rename view (https://www.mysqltutorial.org/mysql-views-tutorial.aspx) s.

Before we execute the `RENAME TABLE` statement, we must ensure that there is no active transactions or locked tables (https://www.mysqltutorial.org/mysql-table-locking/) .

> Note that you cannot use the `RENAME TABLE` statement to rename a temporary table (https://www.mysqltutorial.org/mysql-temporary-table/) , but you can use the ALTER TABLE statement (https://www.mysqltutorial.org/mysql-alter-table.aspx) to rename a temporary table.

In terms of security, any existing privileges that we granted to the old table (https://www.mysqltutorial.org/mysql-grant.aspx) must be manually migrated to the new table.

Before renaming a table, you should evaluate the impact thoroughly. For example, you should investigate which applications are using the table. If the name of the table changes, so the application code that refers to the table name needs to be changed as well. In addition, you must manually adjust other database objects such as views (https://www.mysqltutorial.org/mysql-views-tutorial.aspx) , stored procedures (https://www.mysqltutorial.org/mysql-stored-procedure-tutorial.aspx) , triggers (https://www.mysqltutorial.org/mysql-triggers.aspx) , foreign key constraints (https://www.mysqltutorial.org/mysql-foreign-key/) , etc., that reference to the table. We will discuss this in more detail in the following examples.

MySQL RENAME TABLE examples

First, we create a new database (https://www.mysqltutorial.org/mysql-create-drop-database.aspx) named `hr` that consists of two tables: `employees` and `departments` for the demonstration.

```
CREATE DATABASE IF NOT EXISTS hr;
```

```
CREATE TABLE departments (
    department_id INT AUTO_INCREMENT PRIMARY KEY,
    dept_name VARCHAR(100)
);

CREATE TABLE employees (
    id int AUTO_INCREMENT primary key,
    first_name varchar(50) not null,
    last_name varchar(50) not null,
    department_id int not null,
    FOREIGN KEY (department_id)
        REFERENCES departments (department_id)
);
```

Second, we insert sample data (https://www.mysqltutorial.org/mysql-insert-statement.aspx) into both `employees` and `departments` tables:

```
INSERT INTO departments(dept_name)
VALUES('Sales'),('Markting'),('Finance'),('Accounting'),('Warehouses'),('Production');
```

```
INSERT INTO employees(first_name,last_name,department_id)
VALUES('John','Doe',1),
                ('Bush','Lily',2),
                ('David','Dave',3),
                ('Mary','Jane',4),
                ('Jonatha','Josh',5),
                ('Mateo','More',1);
```

Third, we review our data in the `departments` and `employees` tables:

```
SELECT
    department_id, dept_name
FROM
    departments;
```

```
SELECT
    id, first_name, last_name, department_id
FROM
    employees;
```

Renaming a table referenced by a view

If the table that you are going to rename is referenced by a view (https://www.mysqltutorial.org/mysql-views-tutorial.aspx) , the view will become invalid if you rename the table, and you have to adjust the view manually.

For example, we create a view named `v_employee_info` based on the `employees` and `departments` tables as follows:

```
CREATE VIEW v_employee_info as
    SELECT
        id, first_name, last_name, dept_name
    from
        employees
            inner join
        departments USING (department_id);
```

The views use the inner join (https://www.mysqltutorial.org/mysql-inner-join.aspx) clause to join `departments` and `employees` tables.

The following SELECT statement (https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx) returns all data from the `v_employee_info` view.

```
SELECT
    *
FROM
    v_employee_info;
```

Now we rename the `employees` to `people` table and query data from the `v_employee_info` view again.

```
RENAME TABLE employees TO people;
```

```
SELECT
    *
FROM
    v_employee_info;
```

MySQL returns the following error message:

```
Error Code: 1356. View 'hr.v_employee_info' references invalid table(s) or
column(s) or function(s) or definer/invoker of view lack rights to use them
```

We can use the `CHECK TABLE` statement to check the status of the `v_employee_info` view as follows:

```
CHECK TABLE v_employee_info;
```

We need to manually change the `v_employee_info` view so that it refers to the `people` table instead of the `employees` table.

> Renaming a table that referenced by a stored procedure

In case the table that you are going to rename is referenced by a stored procedure (https://www.mysqltutorial.org/mysql-stored-procedure-tutorial.aspx) , you have to manually adjust it like you did with the view.

First, rename the `people` table back to the `employees` table.

```
RENAME TABLE people TO employees;
```

Then, create a new stored procedure (https://www.mysqltutorial.org/getting-started-with-mysql-stored-procedures.aspx) named `get_employee` that refers to the `employees` table.

```
DELIMITER $$

CREATE PROCEDURE get_employee(IN p_id INT)

BEGIN
        SELECT first_name
                ,last_name
                ,dept_name
        FROM employees
        INNER JOIN departments using (department_id)
        WHERE id = p_id;
END $$

DELIMITER;
```

Next, we execute the `get_employee` table to get the data of the employee with id 1 as follows:

```
CALL get_employee(1);
```

After that, we rename the `employees` to the `people` table again.

```
RENAME TABLE employees TO people;
```

Finally, we call the `get_employee` stored procedure to get the information of employee with id 2:

```
CALL get_employee(2);
```

MySQL returns the following error message:

```
Error Code: 1146. Table 'hr.employees' doesn't exist
```

To fix this, we must manually change the `employees` table in the stored procedure to `people` table.

Renaming a table that has foreign keys referenced to

The `departments` table links to the `employees` table using the `department_id` column. The `department_i` d column in the `employees` table is the foreign key (https://www.mysqltutorial.org/mysql-foreign-key/) that references to the `departments` table.

If we rename the `departments` table, all the foreign keys that point to the `departments` table will not be automatically updated. In such cases, we must drop and recreate the foreign keys manually.

```
RENAME TABLE departments TO depts;
```

We delete a department with id 1, because of the foreign key constraint, all rows in the `people` table should be also deleted. However, we renamed the `departments` table to the `depts` table without updating the foreign key manually, MySQL returns an error as illustrated below:

```
DELETE FROM depts
WHERE
    department_id = 1;
```

```
Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`hr`
```

Renaming multiple tables

We can also use the `RENAME TABLE` statement to rename multiple tables at a time. See the following statement:

```
RENAME TABLE old_table_name_1 TO new_table_name_2,
             old_table_name_2 TO new_table_name_2,...
```

The following statement renames the `people` and `depts` tables to `employees` and `departments` tables:

```
RENAME TABLE depts TO departments,
             people TO employees;
```

Note the `RENAME TABLE` statement is not atomic. It means that if any errors occurred, MySQL does a rollback all renamed tables to their old names.

Renaming tables using ALTER TABLE statement

We can rename a table using the `ALTER TABLE` statement as follows:

```
ALTER TABLE old_table_name
RENAME TO new_table_name;
```

The `ALTER TABLE` statement can rename a temporary table while the `RENAME TABLE` statement cannot.

Renaming temporary table example

First, we create a temporary table (https://www.mysqltutorial.org/mysql-temporary-table/) that contains all unique last names which come from the `last_name` column of the `employees` table:

```
CREATE TEMPORARY TABLE lastnames
SELECT DISTINCT last_name from employees;
```

Second, we use the `RENAME TABLE` to rename the `lastnames` table:

```
RENAME TABLE lastnames TO unique_lastnames;
```

MySQL returns the following error message:

```
Error Code: 1017. Can't find file: '.\hr\lastnames.frm' (errno: 2 - No such file or directory
```

Third, we use the `ALTER TABLE` statement to rename the `lastnames` table.

```
ALTER TABLE lastnames
RENAME TO unique_lastnames;
```

Fourth, we query data from the `unique_lastnames` temporary table:

```
SELECT
    last_name
FROM
    unique_lastnames;
```

In this tutorial, we have shown you how to rename tables using MySQL RENAME TABLE and ALTER TABLE statements.