MySQL ALTER Command

MySQL **ALTER** command is very useful when you want to change a name of your table, any table field or if you want to add or delete an existing column in a table.

Let's begin with creation of a table called **testalter_tbl.**

```
mysql> create table testalter_tbl
    -> (
    -> i INT,
    -> c CHAR(1)
    -> );
Query OK, 0 rows affected (0.05 sec)

mysql> SHOW COLUMNS FROM testalter_tbl;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| i     | int(11) | YES  |     | NULL    |       |
| c     | char(1) | YES  |     | NULL    |       |
+-------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Dropping, Adding or Repositioning a Column:

Suppose you want to drop an existing column **i** from above MySQL table then you will use **DROP** clause along with **ALTER** command as follows:

```
mysql> ALTER TABLE testalter_tbl  DROP i;
```

A **DROP** will not work if the column is the only one left in the table.

To add a column, use ADD and specify the column definition. The following statement restores the **i** column to testalter_tbl:

```
mysql> ALTER TABLE testalter_tbl ADD i INT;
```

After issuing this statement, testalter will contain the same two columns that it had when you first created the table, but will not have quite the same structure. That's because new columns are added to the end of the table by default. So even though **i** originally was the first column in mytbl, now it is the last one.

```
mysql> SHOW COLUMNS FROM testalter_tbl;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| c     | char(1) | YES  |     | NULL    |       |
| i     | int(11) | YES  |     | NULL    |       |
+-------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

To indicate that you want a column at a specific position within the table, either use FIRST to make it the first column or AFTER col_name to indicate that the new column should be placed after col_name. Try the following ALTER TABLE statements, using SHOW COLUMNS after each one to see what effect each one has:

```
ALTER TABLE testalter_tbl DROP i;
ALTER TABLE testalter_tbl ADD i INT FIRST;
ALTER TABLE testalter_tbl DROP i;
ALTER TABLE testalter_tbl ADD i INT AFTER c;
```

The FIRST and AFTER specifiers work only with the ADD clause. This means that if you want to reposition an existing column within a table, you first must DROP it and then ADD it at the new position.

Changing a Column Definition or Name:

To change a column's definition, use **MODIFY** or **CHANGE** clause along with ALTER command. For example, to change column **c** from CHAR11 to CHAR1010, do this:

```
mysql> ALTER TABLE testalter_tbl MODIFY c CHAR(10);
```

With CHANGE, the syntax is a bit different. After the CHANGE keyword, you name the column you want to change, then specify the new definition, which includes the new name. Try out the following example:

```
mysql> ALTER TABLE testalter_tbl CHANGE i j BIGINT;
```

If you now use CHANGE to convert j from BIGINT back to INT without changing the column name, the statement will be as expected:

```
mysql> ALTER TABLE testalter_tbl CHANGE j j INT;
```

The Effect of ALTER TABLE on Null and Default Value Attributes:

When you MODIFY or CHANGE a column, you can also specify whether or not the column can contain NULL values and what its default value is. In fact, if you don't do this, MySQL automatically assigns values for these attributes.

Here is the example, where NOT NULL column will have value 100 by default.

```
mysql> ALTER TABLE testalter_tbl
    -> MODIFY j BIGINT NOT NULL DEFAULT 100;
```

If you don't use above command, then MySQL will fill up NULL values in all the columns.

Changing a Column's Default Value:

You can change a default value for any column using ALTER command. Try out the following example.

```
mysql> ALTER TABLE testalter_tbl ALTER i SET DEFAULT 1000;
mysql> SHOW COLUMNS FROM testalter_tbl;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| c     | char(1) | YES  |     | NULL    |       |
| i     | int(11) | YES  |     | 1000    |       |
+-------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

You can remove default constraint from any column by using DROP clause along with ALTER command.

```
mysql> ALTER TABLE testalter_tbl ALTER i DROP DEFAULT;
mysql> SHOW COLUMNS FROM testalter_tbl;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| c     | char(1) | YES  |     | NULL    |       |
| i     | int(11) | YES  |     | NULL    |       |
+-------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Renaming a Table:

To rename a table, use the **RENAME** option of the ALTER TABLE statement. Try out the following example to rename testalter_tbl to alter_tbl.

```
mysql> ALTER TABLE testalter_tbl RENAME TO alter_tbl;
```

You can use ALTER command to create and drop INDEX on a MySQL file. We will see this feature in next chapter.