My SQL Create - Alter - Drop Table

Objectives

1. Create and drop table
2. Alter Table
3. Inserting Data
4. Update Data
5. Delete Data

1. Create & Drop Table

CREATE

The CREATE statement is used to create tables. It is also used to create indexes, views, events, routines, and triggers.

To create a table, we give a name to a table and to its columns. Each column has a data type. We have covered various MySQL data types in the previous chapter. Choosing the correct data type for the columns is part of the initial design of the database.

```
CREATE TABLE Testing(Id INTEGER);
```

We create a simple Testing table with the CREATE TABLE statement. The table name is Testing. The table has one column called Id. And column's data type is INTEGER.

1. Create & Drop Table

DROP

The DROP TABLE statement drops a table from the database.

```
mysql> DROP TABLE Testing;
Query OK, 0 rows affected (0.00 sec)
mysql> SHOW TABLES LIKE 'T%';
Empty set (0.00 sec)
```

```
mysql> CREATE TABLE Testing(Id INT NOT NULL) ENGINE=MEMORY CHARACTER SET='utf8'
    -> COLLATE='utf8_slovak_ci';
```

We recreate the Testing table. The INT is a synonym for INTEGER. The database engine is explicitly set to MEMORY. We also specify the character set and collation.

2. Alter Table

The ALTER TABLE statement changes the structure of an existing table. It is possible to add a new column, delete a column, rename column and table or change the type of the table. In the following examples, we will demonstrate some of the possibilities.

```
mysql> ALTER TABLE Testing RENAME TO TestTable;
```

```
mysql> SHOW TABLES LIKE 'T%';
+---------------------+
| Tables_in_mydb (T%) |
+---------------------+
| TestTable           |
+---------------------+
```

We use the RENAME TO clause to rename the Testing table to TestTable.

2. Alter Table

```
mysql> ALTER TABLE TestTable ADD iValues INT;
```

We add a new column named iValues to the table.

```
mysql> SHOW COLUMNS FROM TestTable;
+---------+---------+------+-----+---------+-------+
| Field   | Type    | Null | Key | Default | Extra |
+---------+---------+------+-----+---------+-------+
| Id      | int(11) | NO   |     | NULL    |       |
| iValues | int(11) | YES  |     | NULL    |       |
+---------+---------+------+-----+---------+-------+
```

The statement shows available columns in the table. We see the newly added column.

It is possible to add constraints to the table.

```
mysql> ALTER TABLE TestTable ADD PRIMARY KEY (Id);
```

We add a PRIMARY KEY constraint to the TestTable.

2. Alter Table

```
mysql> DESCRIBE TestTable;
+---------+---------+------+-----+---------+-------+
| Field   | Type    | Null | Key | Default | Extra |
+---------+---------+------+-----+---------+-------+
| Id      | int(11) | NO   | PRI | NULL    |       |
| iValues | int(11) | YES  |     | NULL    |       |
+---------+---------+------+-----+---------+-------+
```

The DESCRIBE is a synonym for SHOW COLUMNS FROM. We can see under the Key column that the primary key constraint is set for the Id column.

```
mysql> ALTER TABLE TestTable CHANGE COLUMN iValues iValues1
INT;
```

In this SQL statement we change the column name from iValues to iValues1.

```
mysql> ALTER TABLE TestTable MODIFY COLUMN iValues1 MEDIUMINT;
```

```
mysql> DESCRIBE TestTable;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| Id       | int(11)      | NO   | PRI | NULL    |       |
| iValues1 | mediumint(9) | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
```

We use the above SQL statement to modify the column definition. We change the column datatype from INTEGER to MEDIUM INTEGER.

```
mysql> ALTER TABLE TestTable DROP COLUMN iValues1;
mysql> DESCRIBE TestTable;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| Id    | int(11) | NO   | PRI | NULL    |       |
+-------+---------+------+-----+---------+-------+
```

In our last example, we drop an existing column from the table. In this part of the MySQL tutorial, we were creating, altering and dropping tables.

3. Inserting Data

The INSERT statement is used to insert data into tables.

We will create a new table, where we will do our examples.

```
mysql> CREATE TABLE Books(Id INTEGER PRIMARY KEY, Title VARCHAR(100),
    -> Author VARCHAR(60));
```

We create a new table Books, with Id, Title and Author columns.

```
mysql> INSERT INTO Books(Id, Title, Author) VALUES(1, 'War and Peace', 'Leo Tolstoy');
```

=> This is the classic INSERT SQL statement. We have specified all column names after the table name and all values after the VALUES keyword. We add our first row into the table.

```
mysql> SELECT * FROM Books;
+----+---------------+-------------+
| Id | Title         | Author      |
+----+---------------+-------------+
|  1 | War and Peace | Leo Tolstoy |
+----+---------------+-------------+
```

Now we will insert second row into the Books table:

```
mysql> INSERT INTO Books(Title, Author) VALUES ('The Brothers Karamazov',
    -> 'Fyodor Dostoyevsky');
```

4. Update Data

The UPDATE statement is used to change the value of columns in selected rows of a table.

We recreate the table Books. These are the rows:

```
mysql> SELECT * FROM Books;
+----+-----------------------------+--------------------+
| Id | Title                       | Author             |
+----+-----------------------------+--------------------+
|  1 | War and Peace               | Leo Tolstoy        |
|  2 | The Brothers Karamazov      | Fyodor Dostoyevsky |
|  3 | Paradise Lost               | John Milton        |
|  4 | The Insulted and Humiliated | Fyodor Dostoyevsky |
|  5 | Cousin Bette                | Honore de Balzac   |
+----+-----------------------------+--------------------+
```

3. Inserting Data

Say we wanted to change 'Leo Tolstoy' to 'Lev Nikolayevich Tolstoy' table. The following statement shows, how to accomplish this.

```
mysql> UPDATE Books SET Author='Lev Nikolayevich Tolstoy'
    -> WHERE Id=1;
```

=> The SQL statement sets the author column to 'Lev Nikolayevich Tolstoy' for the column with Id=1.

Now check again:

```
mysql> SELECT * FROM Books WHERE Id=1;
+----+---------------+--------------------------+
| Id | Title         | Author                   |
+----+---------------+--------------------------+
|  1 | War and Peace | Lev Nikolayevich Tolstoy |
+----+---------------+--------------------------+
```

The row is correctly updated.
In this part of the MySQL tutorial, we have inserted, deleted, and updated data in database tables.

5. Delete Data

In MySQL, we can delete data using the DELETE and TRUNCATE statements. The TRUNCATE statement is a MySQL extension to the SQL specification. First, we are going to delete one row from a table. We will use the Books2 table that we have created previously

```
mysql> DELETE FROM Books2 WHERE Id=1;
```

=> We delete a row with Id=1.

```
mysql> DELETE FROM Books2;
mysql> TRUNCATE Books2;
```

These two SQL statements delete all data in the table.

Now we verify the data:

```
mysql> SELECT * FROM Books2;
+----+------------------------+--------------------+
| Id | Title                  | Author             |
+----+------------------------+--------------------+
|  2 | The Brothers Karamazov | Fyodor Dostoyevsky |
|  3 | Paradise Lost          | John Milton        |
+----+------------------------+--------------------+
```