MySQLTUTORIAL

MySQL Sequence

Summary: in this tutorial, you will learn how to use **MySQL sequence** to automatically generate unique numbers for ID columns of tables.

Creating MySQL sequence

In MySQL, a sequence is a list of integers (https://www.mysqltutorial.org/mysql-int/) generated in the ascending order i.e., 1,2,3... Many applications need sequences to generate unique numbers mainly for identification e.g., customer ID in CRM, employee numbers in HR, and equipment numbers in the services management system.

To create a sequence in MySQL automatically, you set the `AUTO_INCREMENT` attribute for a column, which typically is a primary key (https://www.mysqltutorial.org/mysql-primary-key/) column.

The following rules are applied when you use the `AUTO_INCREMENT` attribute:

- Each table has only one `AUTO_INCREMENT` column whose data type is typically the integer (https://www.mysqltutorial.org/mysql-int/) .
- The `AUTO_INCREMENT` column must be indexed, which means it can be either `PRIMARY KEY` (https://www.mysqltutorial.org/mysql-primary-key/) or `UNIQUE` (https://www.mysqltutorial.org/mysql-unique/) index.
- The `AUTO_INCREMENT` column must have a `NOT NULL` constraint (https://www.mysqltutorial.org/mysql-not-null-constraint/) . When you set the `AUTO_INCREMENT` attribute to a column, MySQL automatically adds the `NOT NULL` constraint to the column implicitly.

Creating MySQL sequence example

The following statement creates a table (https://www.mysqltutorial.org/mysql-create-table/) named `employees` that has the `emp_no` column is an `AUTO_INCREMENT` column:

```
CREATE TABLE employees (
    emp_no INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50),
```

```
    last_name VARCHAR(50)
);
```

How MySQL sequence works

The `AUTO_INCREMENT` column has the following attributes:

- The starting value of an `AUTO_INCREMENT` column is 1 and it is increased by 1 when you insert a `NULL` (https://www.mysqltutorial.org/mysql-null/) value into the column or when you omit its value in the `INSERT` (https://www.mysqltutorial.org/mysql-insert-statement.aspx) statement.
- To obtain the last generated sequence number, you use the `LAST_INSERT_ID()` (https://www.mysqltutorial.org/mysql-last_insert_id.aspx) function. We often use the last insert ID for the subsequent statements e.g., insert data into the tables. The last generated sequence is unique across sessions. In other words, if another connection generates a sequence number, from your connection you can obtain it by using the `LAST_INSERT_ID()` function.
- If you insert a new row into a table (https://www.mysqltutorial.org/mysql-insert-statement.aspx) and specify a value for the sequence column, MySQL will insert the sequence number if the sequence number does not exist in the column or issue an error if it already exists. If you insert a new value that is greater than the next sequence number, MySQL will use the new value as the starting sequence number and generate a unique sequence number greater than the current one for the next usage. This creates gaps in the sequence.
- If you use the `UPDATE` (https://www.mysqltutorial.org/mysql-update-data.aspx) statement to update values in the `AUTO_INCREMENT` column to a value that already exists, MySQL will issue a duplicate-key error if the column has a unique index. If you update an `AUTO_INCREMENT` column to a value that is greater than the existing values in the column, MySQL will use the next number of the last insert sequence number for the next row. For example, if the last insert sequence number is 3, you update it to 10, the sequence number for the new row is 4.
- If you use the `DELETE` (https://www.mysqltutorial.org/mysql-delete-statement.aspx) statement to delete the last inserted row, MySQL may or may not reuse the deleted sequence number depending on the storage engine (https://www.mysqltutorial.org/understand-mysql-table-types-innodb-myisam.aspx) of the table. A MyISAM table does not reuse the deleted sequence numbers if you delete a row e.g., the last insert id in the table is 10, if you remove it, MySQL still generates the next sequence number which is 11 for the new row. Similar to MyISAM tables, InnoDB tables do not reuse sequence number when rows are deleted.

Once you set the `AUTO_INCREMENT` attribute for a column, you can reset the auto increment (https://www.mysqltutorial.org/mysql-reset-auto-increment) value in various ways e.g., by using the `ALTER TABLE` (https://www.mysqltutorial.org/mysql-alter-table.aspx) statement.

Let's take a look at some examples to get a better understanding of the MySQL sequence.

First, insert two new rows into the `employees` table:

```
INSERT INTO employees(first_name,last_name)
VALUES('John','Doe'),
      ('Mary','Jane');
```

Second, select data from the `employees` table:

```
SELECT * FROM employees;
```

Third, delete the second employee whose `emp_no` is 2:

```
DELETE FROM employees
WHERE emp_no = 2;
```

Fourth, insert a new employee:

```
INSERT INTO employees(first_name,last_name)
VALUES('Jack','Lee');
```

Because the storage engine of the `employees` table is InnoDB, it does not reuse the deleted sequence number. The new row has `emp_no` 3.

Fifth, update an existing employee with `emp_no` 3 to 1:

```
UPDATE employees
SET
    first_name = 'Joe',
    emp_no = 1
WHERE
    emp_no = 3;
```

MySQL issued an error of duplicate entry for the primary key. Let's fix it.

```
UPDATE employees
SET
    first_name = 'Joe',
    emp_no = 10
WHERE
    emp_no = 3;
```

Sixth, insert a new employee after updating the sequence number to 10:

```
INSERT INTO employees(first_name,last_name)
VALUES('Wang','Lee');
```

The next sequence number of the last insert is number 4, therefore, MySQL uses number 4 for the new row instead of 11.

In this tutorial, you have learned how to use MySQL sequence to generate unique numbers for a primary key column by assigning the `AUTO_INCREMENT` attribute to the column.