MySQLTUTORIAL

How To Use The MySQL Generated Columns

Summary: in this tutorial, you will learn how to use the MySQL generated columns to store data computed from an expression or other columns.

Introduction to MySQL generated column

When you create a new table, you specify the table columns in the `CREATE TABLE (https://www.mysqltutorial.org/mysql-create-table/)` statement. Then, you use the `INSERT (https://www.mysqltutorial.org/mysql-insert-statement.aspx)` , `UPDATE` (https://www.mysqltutorial.org/mysql-update-data.aspx) , and `DELETE (https://www.mysqltutorial.org/mysql-delete-statement.aspx)` statements to modify directly the data in the table columns.

MySQL 5.7 introduced a new feature called the *generated* column. Columns are generated because the data in these columns are computed based on predefined expressions.

For example, you have the `contacts` with the following structure:

```
DROP TABLE IF EXISTS contacts;

CREATE TABLE contacts (
    id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) NOT NULL
);
```

To get the full name of a contact, you use the `CONCAT() (https://www.mysqltutorial.org/sql-concat-in-mysql.aspx)` function as follows:

```
SELECT
    id,
    CONCAT(first_name, ' ', last_name),
    email
```

```
FROM
    contacts;
```

This is not the most beautiful query yet.

By using the MySQL generated column, you can recreate the `contacts` table as follows:

```
DROP TABLE IF EXISTS contacts;

CREATE TABLE contacts (
    id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    fullname varchar(101) GENERATED ALWAYS AS (CONCAT(first_name,' ',last_name)),
    email VARCHAR(100) NOT NULL
);
```

The `GENERATED ALWAYS as (expression)` is the syntax for creating a generated column.

To test the `fullname` column, you insert (https://www.mysqltutorial.org/mysql-insert-statement.aspx) a row into the `contacts` table.

```
INSERT INTO contacts(first_name,last_name, email)
VALUES('john','doe','john.doe@mysqltutorial.org');
```

Now, you can query data (https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx) from the `contacts` table.

```
SELECT
    *
FROM
    contacts;
```

The values in the `fullname` column are computed on the fly when you query data from the `contacts` table.

MySQL provides two types of generated columns: stored and virtual. The virtual columns are calculated on the fly each time data is read whereas the stored column are calculated and stored physically when the data is updated.

Based on this definition, the `fullname` column that in the example above is a virtual column.

MySQL generated column's syntax

The syntax for defining a generated column is as follows:

```
column_name data_type [GENERATED ALWAYS] AS (expression)
   [VIRTUAL | STORED] [UNIQUE [KEY]]
```

First, specify the column name and its data type.

Next, add the `GENERATED ALWAYS` clause to indicate that the column is a generated column.

Then, indicate whether the type of the generated column by using the corresponding option: `VIRTUAL` or `STORED`. By default, MySQL uses `VIRTUAL` if you don't specify explicitly the type of the generated column.

After that, specify the expression within the braces after the `AS` keyword. The expression can contain literals, built-in functions with no parameters, operators, or references to any column within the same table. If you use a function, it must be scalar and deterministic.

Finally, if the generated column is stored, you can define a unique constraint (https://www.mysqltutorial.org/mysql-unique-constraint/) for it.

MySQL stored column example

Let's look at the `products` table in the sample database (https://www.mysqltutorial.org/mysql-sample-database.aspx) .

The data from `quantityInStock` and `buyPrice` columns allow us to calculate the stock's value per SKU using the following expression:

```
quantityInStock * buyPrice
```

However, we can add a stored generated column named `stock_value` to the `products` table using the following `ALTER TABLE ...ADD COLUMN` (https://www.mysqltutorial.org/mysql-add-column/) statement:

```
ALTER TABLE products
ADD COLUMN stockValue DOUBLE
GENERATED ALWAYS AS (buyprice*quantityinstock) STORED;
```

Typically, the `ALTER TABLE` statement requires a full table rebuild, therefore, it is time-consuming if you change the big tables. However, this is not the case for the virtual column.

Now, we can query the stock value directly from the `products` table.

```
SELECT
    productName,
    ROUND(stockValue, 2) stock_value
FROM
    products;
```

In this tutorial, you have learned how to use the MySQL generated column to store data computed from an expression or other columns.