MySQL ENUM

Summary: in this tutorial, you will learn how to use MySQL `ENUM` data type for defining columns that store enumeration values.

Introduction to MySQL ENUM data type

In MySQL, an `ENUM` is a string object whose value is chosen from a list of permitted values defined at the time of column creation.

To define an `ENUM` column, you use the following syntax:

```
column_name ENUM('value1', 'value2', ..., 'valueN')
```

In this syntax:

- `column_name` : This is the name of the column that uses the `ENUM` data type
- `'value1'` , `'value2'` , ... `'valueN'` : These are the list of values that the column can hold. The values are separated by commas.

MySQL ENUM data type example

Suppose you have to store ticket information with the priority: low, medium, and high. To assign

these string values to the `priority` column, you can use the `ENUM` data type.

First, create a new table that includes a `priority` column with the `ENUM` type:

```
CREATE TABLE tickets (
    id INT PRIMARY KEY AUTO_INCREMENT,
    title VARCHAR(255) NOT NULL,
    priority ENUM('Low', 'Medium', 'High') NOT NULL
);
```

The `priority` column accepts only three values `Low`, `Medium` and `High`.

Behind the scenes, MySQL maps each enumeration member to a numeric index. In this case, it maps the Low, Medium, and High values to 1, 2, and 3 respectively.

Second, insert a new row into the `tickets` table:

```
INSERT INTO tickets(title, priority)
VALUES('Scan virus for computer A', 'High');
```

In this example, we use the predefined value `'High'` to insert into the `priority` column.

Besides the enumeration values, you can use the numeric index of the enumeration member to insert data into an `ENUM` column.

Third, insert a new row into the `tickets` table using a numeric index value instead of the predefined values:

```
INSERT INTO tickets(title, priority)
VALUES('Upgrade Windows OS for all computers', 1);
```

In this example, instead of using the `Low` enumeration value, we used value 1. Since `Low` is mapped to 1, it is acceptable.

Fourth, insert multiple rows into the `tickets` table:

```
INSERT INTO tickets(title, priority)
VALUES('Install Google Chrome for Mr. John', 'Medium'),
      ('Create a new user for the new employee David', 'High');
```

Because we define the `priority` as a `NOT NULL` column, when you insert a new row without specifying the value for the `priority` column, MySQL will use the first enumeration member as the default value. For example:

```
INSERT INTO tickets(title)
VALUES('Refresh the computer of Ms. Lily');
```

The contents of the `tickets` table are as follows:

```
+----+-----------------------------------------+----------+
| id | title                                   | priority |
+----+-----------------------------------------+----------+
|  1 | Scan virus for computer A               | High     |
|  2 | Upgrade Windows OS for all computers    | Low      |
|  3 | Install Google Chrome for Mr. John      | Medium   |
|  4 | Create a new user for the new employee David | High     |
|  5 | Refresh the computer of Ms. Lily        | Low      |
+----+-----------------------------------------+----------+
5 rows in set (0.00 sec)
```

In the non-strict SQL mode, if you insert an invalid value into an `ENUM` column, MySQL will use an empty string `''` with the numeric index `0` for inserting.

If you enable the SQL strict mode and you attempt to insert an invalid `ENUM` value, you will get an error. For example:

```
INSERT INTO tickets(title, priority)
VALUES('Invalid ticket',-1);
```

Error:

```
ERROR 1265 (01000): Data truncated for column 'priority' at row 1
```

> Note that an `ENUM` column can accept `NULL` values if you define it as a nullable column.

Filtering MySQL ENUM values

The following statement retrieves all the tickets with high priority:

```
SELECT
  *
FROM
  tickets
WHERE
  priority = 'High';
```

Output:

```
+----+-----------------------------------------------+----------+
| id | title                                         | priority |
+----+-----------------------------------------------+----------+
|  1 | Scan virus for computer A                     | High     |
|  4 | Create a new user for the new employee David | High     |
+----+-----------------------------------------------+----------+
```

Because the enumeration member `'High'` is mapped to 3, the following query returns the same result set:

```
SELECT
  *
FROM
  tickets
WHERE
  priority = 3;
```

Sorting MySQL ENUM values

MySQL sorts `ENUM` values based on their index numbers. Therefore, the order of members depends on how they were defined in the enumeration list.

The following query selects the tickets and sorts them by priority from `High` to `Low`:

```
SELECT
  title,
  priority
FROM
  tickets
ORDER BY
  priority DESC;
```

Output:

```
+-----------------------------------------------+----------+
| title                                         | priority |
+-----------------------------------------------+----------+
| Scan virus for computer A                     | High     |
| Create a new user for the new employee David  | High     |
| Install Google Chrome for Mr. John            | Medium   |
| Upgrade Windows OS for all computers          | Low      |
| Refresh the computer of Ms. Lily              | Low      |
+-----------------------------------------------+----------+
5 rows in set (0.01 sec)
```

It's a good practice to define the enumeration values in the order that you want to sort when you create the `ENUM` column.

Advantages of ENUM data type

- **Data Validation:** `ENUM` data types provide strong data validation because they restrict column values to a predefined set of options. This helps maintain data integrity.
- **Readability:** `ENUM` values are human-readable and self-explanatory, making it easy to understand the data in the column.
- **Space Efficiency:** `ENUM` values are stored as integers, which are more space-efficient than storing strings.

Limitations of ENUM data type

- **Limited Flexibility:** Once `ENUM` values are defined, they cannot be easily changed or extended. If you need to add or remove values, you may need to alter the table

structure, which can be a complex operation.

- **Portability:** The `ENUM` data type is specific to MySQL and may not be supported in other database systems.
- **Maintenance:** `ENUM` values can make the schema harder to maintain as the application evolves, as adding or removing values can be complex.

Summary

- Use MySQL `ENUM` for defining columns with a limited set of allowed values.

PREVIOUSLY
MySQL BLOB

UP NEXT
MySQL UUID Smackdown: UUID vs. INT for Primary Key

Search ...

GETTING STARTED

What Is MySQL?

Install MySQL Database Server

Connect to MySQL Server

Download MySQL Sample Database

Load Sample Database

QUERYING DATA

SELECT FROM

SELECT

ORDER BY

WHERE

SELECT DISTINCT

AND

OR

IN

NOT IN

BETWEEN

LIKE

MANAGING DATABASES

MANAGING TABLES

Create Tables

AUTO_INCREMENT

Rename Tables

Add Columns

Drop Columns

Drop Tables

Temporary Tables

Generated Columns

MYSQL CONSTRAINTS

Primary Key

Foreign Key

Disable Foreign Key Checks

UNIQUE Constraint

NOT NULL Constraint

DEFAULT Constraint

CHECK Constraint

INSERT DATA

Insert Into

Insert Multiple Rows

INSERT INTO SELECT

Insert On Duplicate Key Update

INSERT IGNORE

Insert DateTimes

Insert Dates

UPDATE DATA

UPDATE

UPDATE JOIN

DELETE DATA

DELETE JOIN

ON DELETE CASCADE

TRUNCATE TABLE

MYSQL TRANSACTIONS

Table Locking

MYSQL DATA TYPES

BIT

INT

BOOLEAN

DECIMAL

DATETIME

TIMESTAMP

DATE

TIME

CHAR

VARCHAR

TEXT

BINARY

VARBINARY

ENUM

BLOB

MYSQL GLOBALIZATION

MySQL Character Sets

MySQL Collation

MYSQL IMPORT & EXPORT

Import a CSV File Into a Table

Export a Table to a CSV File

WEBSITE

MySQLTutorial.org helps you master MySQL quickly, easily, and with enjoyment. Our tutorials make learning MySQL a breeze.

All MySQL tutorials are clear, practical and easy-to-follow.

More About Us

MySQL Port

MySQL Commands

innodb_dedicated_server: Configure InnoDB Dedicated Server

innodb_flush_method: Configure InnoDB Flush Method

innodb_log_buffer_size: Configure InnoDB Log Buffer Size

innodb_buffer_pool_chunk_size: Configure Buffer Pool Chunk Size

innodb_buffer_pool_instances: Configuring Multiple Buffer Pool Instances for Improved Concurrency in MySQL

innodb_buffer_pool_size: Configure InnoDB Buffer Pool Size

MySQL InnoDB Architecture

How to Kill a Process in MySQL

Donation ❤️

Contact Us

About

Privacy Policy

OTHERS

MySQL Cheat Sheet

MySQL Resources

MySQL Books