MySQL UUID Smackdown: UUID vs. INT for Primary Key

Summary: This tutorial introduces you to MySQL UUID, shows you how to use it as the primary key for a table, and discusses the pros and cons of using it as the primary key.

Introduction to MySQL UUID

UUID stands for Universally Unique IDentifier. UUID is defined based on RFC 4122, "a Universally Unique Identifier (UUID) URN Namespace".

UUID is designed as a number that is unique globally in space and time. Two UUID values are expected to be distinct, even if they are generated on two independent servers.

In MySQL, a UUID value is a 128-bit number represented as a utf8 string of five hexadecimal numbers in the following format:

```
aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee
```

To generate a UUID value, you use the `UUID()` function as follows:

```
UUID()
```

The `UUID()` function returns a UUID value in compliance with UUID version 1 described in the RFC 4122.

For example, the following statement uses the `UUID()` function to generate a UUID value:

```
SELECT UUID();
```

Output:

```
+--------------------------------------+
| UUID()                               |
+--------------------------------------+
| e5f96fc2-a700-11ee-9e18-0a0027000003 |
+--------------------------------------+
1 row in set (0.04 sec)
```

MySQL UUID vs. Auto-Increment INT as the primary key

Pros

Using UUID for a primary key has the following advantages:

- UUID values are unique across tables, databases, and even servers that allow you to merge rows from different databases or distribute databases across servers.
- UUID values do not expose the information about your data so they are safer to use in a URL. For example, if a customer with ID 10 accesses his account via `http://www.example.com/customers/10/` URL, it is easy to guess that there is a customer 11, 12, etc., and this could be a target for an attack.
- UUID values can be generated anywhere which can help avoid a round trip to the database server. It also simplifies logic in the application. For example, to insert data into a parent table and child tables, you have to insert into the parent table first, get the generated ID, and then insert data into the child tables. By using UUID, you can generate the primary key value of the parent table up front and insert rows into both parent and child tables at the same time within a transaction.

Cons

Besides the advantages, UUID values also come with some disadvantages:

- Storing UUID values (16 bytes) takes more storage than integers (4 bytes) or even big integers(8 bytes).
- Debugging seems to be more difficult, imagine the expression `WHERE id = 'df3b7cb7-6a95-11e7-8846-b05adad3f0ae'` instead of `WHERE id = 10`
- Using UUID values may cause performance issues due to their size and not being ordered.

MySQL UUID solution

In MySQL, you can store UUID values in a compact format (`BINARY`) and display them in human-readable format (`VARCHAR`) with the help of the following functions:

- `UUID_TO_BIN`
- `BIN_TO_UUID`
- `IS_UUID`

> Notice that `UUID_TO_BIN()` , `BIN_TO_UUID()` , and `IS_UUID()` functions are only available in MySQL 8.0 or later.

The `UUID_TO_BIN()` function converts a UUID from a human-readable format (`VARCHAR`) into a compact format (BINARY) format for storing and the `BIN_TO_UUID()` function converts UUID from the compact format (`BINARY`) to human-readable format (`VARCHAR`) for displaying.

The `IS_UUID()` function returns 1 if the argument is a valid string-format UUID. If the argument is not valid string format UUID, the `IS_UUID` function returns 0. In case the argument is `NULL` , the `IS_UUID()` function returns `NULL` .

The following are the valid string-format UUIDs in MySQL:

```
aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee
aaaaaaaabbbbccccddddeeeeeeeeeeee
{aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee}
```

MySQL UUID examples

Let's take a look at an example of using UUID as the primary key.

1) Basic MySQL UUID example

First, create a new table called `customers`:

```
CREATE TABLE customers (
    id BINARY(16) PRIMARY KEY,
    name VARCHAR(255)
);
```

Second, insert UUID values into the `id` column using the `UUID()` and `UUID_TO_BIN()` functions as follows:

```
INSERT INTO customers(id, name)
VALUES(UUID_TO_BIN(UUID()),'John Doe'),
      (UUID_TO_BIN(UUID()),'Will Smith'),
      (UUID_TO_BIN(UUID()),'Mary Jane');
```

Third, query data from a UUID column and use `BIN_TO_UUID()` function to convert binary format to human-readable format:

```
SELECT
    BIN_TO_UUID(id) id,
    name
FROM
    customers;
```

Output:

```
+--------------------------------------+------------+
| id                                   | name       |
+--------------------------------------+------------+
| 6ab66253-a701-11ee-9e18-0a0027000003 | John Doe   |
| 6ab66917-a701-11ee-9e18-0a0027000003 | Will Smith |
| 6ab66b0e-a701-11ee-9e18-0a0027000003 | Mary Jane  |
```

```
+--------------------------------------+------------+
3 rows in set (0.01 sec)
```

2) Using UUID as the default value for the primary key

First, create a new table called `vendors`:

```
CREATE TABLE vendors(
    id BINARY(16) DEFAULT (UUID_TO_BIN(UUID())),
    name VARCHAR(255),
    PRIMARY KEY(id)
);
```

In the `vendors` table, we use the result of the expression `UUID_TO_BIN(UUID())` as the default value for the primary key column. Therefore, we don't need to specify the value for the `id` column whenever we insert a new row into the table.

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

```
INSERT INTO vendors(name)
VALUES('ABC Inc.');
```

Third, retrieve data from the `vendors`:

```
SELECT
    BIN_TO_UUID(id) id,
    name
FROM
    vendors;
```

Output:

```
+--------------------------------------+----------+
| id                                   | name     |
+--------------------------------------+----------+
| 2c56abc6-a704-11ee-9e18-0a0027000003 | ABC Inc. |
+--------------------------------------+----------+
1 row in set (0.00 sec)
```

In this tutorial, you have learned about MySQL UUID and how to use it for the primary key column.

Was this tutorial helpful?

PREVIOUSLY
MySQL ENUM

UP NEXT
MySQL Character Sets

Search ...

GETTING STARTED

Load Sample Database

QUERYING DATA

SELECT FROM

SELECT

ORDER BY

WHERE

SELECT DISTINCT

AND

OR

IN

NOT IN

BETWEEN

LIKE

LIMIT

IS NULL

Table & Column Aliases

Joins

INNER JOIN

LEFT JOIN

RIGHT JOIN

Self Join

CROSS JOIN

GROUP BY

HAVING

HAVING COUNT

ROLLUP

Subquery

Derived Tables

EXISTS

EXCEPT

INTERSECT

MANAGING DATABASES

Select a Database

Create Databases

Drop 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

ABOUT MYSQL TUTORIAL 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

LATEST TUTORIALS

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

SITE LINKS

Donation ❤️

Contact Us

About

Privacy Policy

OTHERS

MySQL Cheat Sheet

MySQL Resources

MySQL Books