ORACLE

Native Support for New Data Types in MySQL

BOOLEAN, UUID, and ARRAY

Øystein Grøvlen

MySQL Optimizer Team Lead

Oracle

May 26, 2026

Øystein Grøvlen

MySQL Optimizer Team Lead, Oracle

Safe harbor statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

Proposals for new data types

BOOLEAN

UUID

ARRAY

BOOLEAN

BOOLEAN

Currently a synonym for TINYINT

```
mysql> CREATE TABLE t1(b BOOLEAN);
Query OK, 0 rows affected (0.051 sec)

mysql> SHOW CREATE TABLE t1;
+-------+--------------------------------------------------------------------------------------+
| Table | Create Table                                                                         |
+-------+--------------------------------------------------------------------------------------+
| t1    | CREATE TABLE `t1` (
  `b` tinyint(1) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci                           |
+-------+--------------------------------------------------------------------------------------+
1 row in set (0.017 sec)
```

Make BOOLEAN a native type

WL#3554 Boolean data type

First step

- Introduce a new type BOOLX

Functional requirements

- F1 - Create column with BOOLX data type, e.g. CREATE TABLE t(b BOOLX);
- F3 - Let BOOLX be an integer value with only two possible values: 0 (FALSE) and 1 (TRUE).
- F4 - SHOW CREATE TABLE must show BOOLX data types.
- F5 - DESCRIBE must show BOOLX data types.
- F6 - DUMP command must show BOOLX data types for backup purposes.
- F7 - Implicit CAST between integer types and BOOLX (non-standard)
- F8 - ALTER TABLE allows to convert between data types, according to the rules for CAST.

Existing contribution

- Bug#103845 — Contribution: WL#3554 Boolean data type in MySQL

Make BOOLEAN a native type

WL#15215 Switch behaviour of BOOL/BOOLEAN from TINYINT(1) to new boolean data type

Final step

- Support BOOLEAN according to the SQL standard
- Allow BOOL as synonym for BOOLEAN
- Introduce new system variable **`boolean_is_tinyint`** to switch between old and new behavior

Question

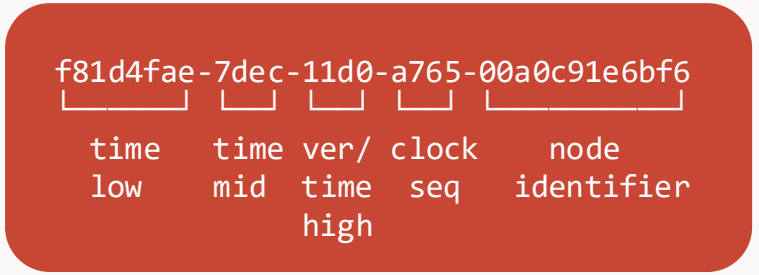
Is the two-step approach strictly necessary?

UUID

UUID

Universally Unique Identifier

- 128-bit number used as a unique identifier
- Current representation
  - CHAR(36)
    - `UUID()` function generates a new UUID version 1
    - `IS_UUID()` checks whether the supplied string is a valid UUID.
    - Disadvantage: Size
  - BINARY(16)
    - Functions to convert between character string and binary
      - `UUID_TO_BIN()`
      - `BIN_TO_UUID()`
      - Optional parameter to swap *time-low* and *time-high* parts to ensure chronological sorting



UUID

New data type?

```
CREATE TABLE t1 (id UUID)
```

- Stored as BINARY(16)
- Internal format ensures chronological sorting
- Automatic conversion from CHAR to UUID

```
INSERT INTO TABLE t1 VALUES ('f81d4fae-7dec-11d0-a765-00a0c91e6bf6');
```

- Built-in CAST

```
SELECT CAST('f81d4fae-7dec-11d0-a765-00a0c91e6bf6' AS UUID);
```

- Error if CHAR/BINARY value is not a valid UUID
- Easy Migration

```
CREATE TABLE t1 (id BINARY(16));
ALTER TABLE t1 MODIFY COLUMN id UUID;
```

New function?

- `UUID_v7()`

UUID data type

Main benefits:

- Avoids explicit use of conversion functions
- Efficient storage
- Automatic validation of UUID values

ARRAY

ARRAY

SQL Standard Data Type

```
CREATE TABLE t1 (id INTEGER, vals DOUBLE ARRAY[10])
```

- Maximum cardinality is optional
- Nested arrays are supported

```
CREATE TABLE t2 (vals2d VARCHAR(10) ARRAY ARRAY)
```

- Array constructor

```
INSERT INTO t1 VALUES (ARRAY[1.0, 1.4142, 2.7182, 3.14])
INSERT INTO t1 VALUES (ARRAY(SELECT …))
```

- Cardinality function

```
SELECT CARDINALITY(vals) FROM t1
```

- Array element reference

```
SELECT vals[1], vals[CARDINALITY(vals)] FROM t1
```

- Aggregate function

```
SELECT x, ARRAY_AGG(y) FROM t GROUP BY x
```

- Collection derived table

```
SELECT … FROM UNNEST(ARRAY(SELECT vals FROM t1)) WITH ORDINALITY AS dt(val, idx)
SELECT t1.id, dt.val FROM t1, LATERAL(UNNEST(t1.vals)) AS dt(val)
```

Alternative

Use JSON arrays

Summary

Bug#120450 Proposed new native data types

- BOOLEAN
  - Contribution exists
- UUID
  - Avoids explicit use of conversion functions
  - Efficient storage
  - Automatic validation of UUID values
- ARRAY
  - SQL standard type
  - Benefits compared to JSON arrays:
    - Strict typing
    - More compact syntax

Contributions are encouraged!

Thank you


ORACLE