MySQL Temporal Enhancements

Refactoring and new data types

Roy Lyseng
Senior Principal Engineer
Oracle MySQL
21. May, 2026

MySQL - Temporal data type work

Motivation

- Tackle the Year 2038 problem with TIMESTAMP data type
- SQL standard compliance (new data types)
- Improved code consistency and type safety
- Improved efficiency
- Avoid time zone gap problem (conversion between data types is lossy)
- Fix data type consistency problems

MySQL - Temporal data type work

Some bug reports that may be handled by this work

- Bug#95054: TIMESTAMP WITH TIME ZONE
- Bug#14031: TIMESTAMP conflicts with Standard SQL 92
- Bug#118267: Can we extend timestamp range, 2038 limit is approaching
- Bug#88465: JOIN is performed incorrectly on TIMESTAMP
- Bug#95797: Erratic lossy roundtrip conversion of timestamps when copying columns
- Bug#96608: Mishandled timezone conversions on timestamps
- Bug#96609: Corrupted timestamp data on alter
- Bug#108787: UNIX_TIMESTAMP(NOW()) is lossy
- Bug#58583: UTC_TIMESTAMP inserted in TIMESTAMP column uses time_zone setting

MySQL - Temporal data type work

WL#16790 - Refactor TIME handling in server

- Implement a dedicated Time_val class for processing of TIME values
- Small and efficient: 8 bytes instead of 48 bytes
- Implemented in 9.5.0
- Load/store time reduced to approximately half

MySQL - Temporal data type work

WL#16895 - Refactor DATE handling in server

- Implement a dedicated Date_val class for processing of DATE values
- Implemented range: 0000-01-01 to 9999-12-31, plus 0000-00-00
- Small and efficient: 4 bytes instead of 48 bytes
- Implemented in 9.7.0
- Some basic operations twice as efficient
- 9 consistency problems fixed

MySQL - Temporal data type work

WL#16896 - Refactor DATETIME handling in server

- Implement a dedicated Datetime_val class for processing of DATETIME and TIMESTAMP values
- Small and efficient: 16 bytes instead of 48 bytes
- Two classes of values: UTC-based and local time based
- Operations may convert between value classes as needed
- Pure UTC-based operations avoid time zone gap problem
- Supports existing DATETIME and TIMESTAMP data types
- Supports future TIMESTAMP compatible type with full date time range
- Supports future date time type with explicitly stored time zone displacement
- Will potentially fix 20-25 consistency problems
- Work in progress

MySQL - Temporal data type work

WL#13922 - DATETIME WITH TIME ZONE

- New data type: DATETIME WITH LOCAL TIME ZONE
    - Range 0000-01-01 to 9999-12-31, plus 0000-00-00
    - Replacement for current TIMESTAMP data type
    - UTC-based timestamp value, with time zone displacement based on current session
- New data type: DATETIME WITH TIME ZONE
    - Range 0000-01-01 to 9999-12-31, plus 0000-00-00
    - UTC-based timestamp value, stores explicit time zone displacement

MySQL - Temporal data type work

Implications for databases and applications

- Existing system tables will be converted to use DATETIME WITH LOCAL TIME ZONE instead of TIMESTAMP
- Some system functions like CURRENT_TIMESTAMP and UTC_TIMESTAMP may be changed to return a different data type
- Automatic conversions will neutralize most potential problems around type handling
- In-place conversion of user data from TIMESTAMP to DATETTIME WITH [LOCAL] TIME ZONE will be provided

Suggestions and contributions are welcome

Thank you