#FOSDEM #MySQLDay #MySQL8isGreat

ORACLE MySQL

ticketsolve

Friends let real friends use MySQL 8.0

30th January 2020
BRUSSELS

Saverio Miroddi
Senior Engineer at ticketsolve


FRIENDS DON'T LET
REAL FRIENDS USE
RELATIONAL DATABASES
mongoDB.
14' x 48'
CLEAR CHANNEL


FRIENDS DON'T LET
REAL FRIENDS USE
RELATIONAL DATABASES
MYSQL 8
MySQL
14' x 48'
CLEAR CHANNEL

Introduction: Who are I and Ticketsolve

- Lead of invisible things at Ticketsolve
- Ticketsolve is a primary player in Ireland/UK in the events ticketing market, and it has an internal open source program
- Efficiency is one of our key values; this reflect in the presentation's tooling and general approach

Introduction: Presentation and target audience

- We've been using MySQL for 10+ years
- We start each migration process around 6 months after GA
- This presentation describes our experience in migrating to MySQL 8.0; it covers the most important concepts of what a sysadmin needs to know and do, if "tomorrow they need to migrate production to 8.0"
- The most important subject is utf8mb4 (and its default collation)
- There is no target level, although the audience is system/database administrators

Preparing MySQL: setup and tooling

- Use tarballs for convenience, and a fixed path with symlink
- I'm using several scripts to speed up operations and access frequently used terms
- PS. I'm running fully on ZFS!!

Primary problems to tackle when migrating to MySQL 8.0:

- Conversion to utf8mb4 with the **correct** collation (tables/triggers)
- Trailing whitespace handled differently
- GROUP BY not sorted anymore
- information_schema not updated realtime anymore
- Schema migration tools might not be compatible

The larger the scale, the more aspects will need to be considered, e.g.:

- Dirty page cleaning parameters
- Data dictionary contention
- etc.

Migrating to utf8mb4: Summary

- The new collation is the crucial aspect to consider
- What `utf8mb4_0900_ai_ci` is and what it brings
- Unfortunately, the documentation available in the web is outdated - it uses the prerelease default collation (`utf8mb4_general_ci`)

How the charset parameters work

- What happens when a client connects to the server?
- Which are the database object charset-related defaults?

Strategy for a migration; collation coercion, and issues `general` <> `0900_ai`

Change the connection settings, then migrate per-table.
I'll simulate all the cases of inter-charset comparisons.

- Comparisons `utf8_general_ci` columns <> literals (see next)
- Comparisons `utf8_general_ci` columns <> columns
  - Moving a schema from `utf8mb4_general_ci to` `utf8mb4_0900_ai_ci` is not trivial!

Collation Coercibility in Expressions

0: An explicit COLLATE clause (not coercible at all)
1: The concatenation of two strings with different collations
2: The collation of a column or a stored routine parameter or local variable
3: A “system constant” (the string returned by functions such as USER() or VERSION())
4: The collation of a literal
5: The collation of a numeric or temporal value
6: NULL or an expression that is derived from NULL

(From https://dev.mysql.com/doc/refman/8.0/en/charset-collation-coercibility.html)

Issues with `0900_ai` collation padding

- How are trailing whitespaces handled?
- What defines such handling? (see next)

The following are the core string comparison rules from the SQL (2003) standard (section 8.2):

3) The comparison of two character strings is determined as follows:

a) Let CS be the collation [...]
b) **If the length in characters of X is not equal to the length in characters of Y, then the shorter string is effectively replaced, for the purposes of comparison, with a copy of itself that has been extended to the length of the longer string by concatenation on the right of one or more pad characters**, where the pad character is chosen based on CS. **If CS has the NO PAD characteristic, then the pad character is an implementation-dependent character** different from any character in the character set of X and Y that collates less than any string under CS. Otherwise, the pad character is a space.

Triggers

- Triggers are fairly easy to handle (see next), as they can be dropped/rebuilt - just make sure to consider comparisons in the trigger body

```
SHOW CREATE TRIGGER enqueue_comments_update_instance_event\G

-- SQL Original Statement:
CREATE TRIGGER `enqueue_comments_update_instance_event`
AFTER UPDATE ON `comments`
FOR EACH ROW
trigger_body: BEGIN
  SET @changed_fields := NULL;

  IF NOT (OLD.description <=> NEW.description COLLATE utf8_bin AND CHAR_LENGTH(OLD.description) <=> CHAR_LENGTH(NEW.description)) THEN
    SET @changed_fields := CONCAT_WS(',', @changed_fields, 'description');
  END IF;

  IF @changed_fields IS NOT NULL THEN
    SET @old_values := NULL;
    SET @new_values := NULL;

    INSERT INTO instance_events(created_at, instance_type, instance_id, operation, changed_fields, old_values, new_values)
    VALUES(NOW(), 'Comment', NEW.id, 'UPDATE', @changed_fields, @old_values, @new_values);
  END IF;
END
--   character_set_client: utf8mb4
--   collation_connection: utf8mb4_0900_ai_ci
--     Database Collation: utf8mb4_0900_ai_ci
```

Behavior with indexes

- How does an `utf8mb4` index behave when comparing with an `utf8` literal?
- How does it behave on joins? What does one see in the standard and extended `EXPLAIN`s?

Consequences of the increase in (potential) size of char columns

- utf8mb4 characters will take 33% more, which must stay withing the InnoDB index limit, which is however (as of 8.0 default), high (3072 bytes).

Remember:

- `[VAR]CHAR(n)` refers to the number of characters; therefore, the maximum requirement is `4 * n` bytes
- `TEXT` fields refer to the number of bytes

`information_schema_stats_expiry` introduction

- When querying the `information_schema`, beware that stats are updated only once a day!
- If required, invoke `ANALYZE TABLE` (or reduce the setting value)

GROUP BY is now unsorted (not implicitly sorted)

- Developers will need to go through the codebase and check each query with GROUP BY
- It's impossible to use a one-size-fits-all approach

Very generic approaches:

- grep -A
- mini script checking every pair of lines

Schema migration tool issues

- There's a known [breaking bug](github/gh-ost/issues/687) on the latest Gh-ost release, which may prevent operations from succeeding on MySQL 8
  - There's a test fix PR, it would be great if somebody with scale could test it!
- Use trigger-based tools, like `pt-online-schema-change` v3.1.1 or v3.0.x (but v3.1.0 has a breaking bug) or Facebook's OnlineSchemaChange

Conclusion

- Over the next weeks, I will expand this subject into a series of articles in my professional blog
- This presentation is hosted (with git history preserved) at https://github.com/saveriomiroddi/prefosdem-2020-presentation
- Those who'd like to give ZFS a spin can check out my installer: https://github.com/saveriomiroddi/zfs-installer

Extra: comparing the global system variables between major releases

- Although we read the release notes for every release (including patch versions), approaching major version changes is very intimidating
- Comparing system variables is a good starting point
- Check out the bulky system variables, and filter them out

Extra: Mac Homebrew default collation is `utf8mb4_general_ci`!

- When MySQL is installed via Homebrew, the default collation is `utf8mb4_general_ci`.
- My fix PR has been merged into master
- Interim solution 1: edit the formula and rebuild mysql
- Interim solution 2: disable the server handshake