MySQL 8.0でMroonga

須藤功平

クリアコード

MyNA会 2018年7月
2018-07-23

MySQLのストレージエンジン

ストレージエンジン

- C++で実装
- MySQLのハンドラーAPIを使う
  - ハンドラーAPIっていう名前なの？

ハンドラーAPI

- すごくよく変わる
- メジャーバージョンアップ
  - 絶対変わる
- MyNAバージョンアップ
  - たまに変わる

MySQL 8.0対応

1. ビルドエラー修正
   - API変更に対応
2. 新機能対応
   - 新COLLATION対応とか
3. テストをパスするようにする
   - ビルドができても期待通りに動くとは限らない

制約

対応MySQL族で
Mroongaのテストを
すべてパスし続ける

対応MySQL族

- MySQL
  - 5.5, 5.6, 5.7
- MariaDB
  - 5.5, 10.0, 10.1, 10.2, 10.3
- Percona Server for MySQL
  - 5.6, 5.7

Travis CI

Build Jobs

✓ # 3794.1		C++	MYSQL_VERSION=mysql-system	4 min 9 sec
✓ # 3794.2		C++	MYSQL_VERSION=mysql-5.6	4 min 57 sec
✓ # 3794.3		C++	MYSQL_VERSION=mysql-5.7	6 min 22 sec
✓ # 3794.4		C++	MYSQL_VERSION=mariadb-5.5	4 min 41 sec
✓ # 3794.5		C++	MYSQL_VERSION=mariadb-10.0.35…	16 min 11 sec
✓ # 3794.6		C++	MYSQL_VERSION=mariadb-10.0.35…	21 min 34 sec
✓ # 3794.7		C++	MYSQL_VERSION=mariadb-10.1.34…	17 min 26 sec
✓ # 3794.8		C++	MYSQL_VERSION=mariadb-10.2.16…	22 min 49 sec
✓ # 3794.9		C++	MYSQL_VERSION=mariadb-10.2.16…	23 min 43 sec
✓ # 3794.10		C++	MYSQL_VERSION=mariadb-10.3.8 …	28 min 38 sec
✓ # 3794.11		C++	MYSQL_VERSION=mariadb-10.3.8 …	26 min 39 sec
✓ # 3794.12		C++	MYSQL_VERSION=percona-server-5.6	5 min 54 sec
✓ # 3794.13		C++	MYSQL_VERSION=percona-server-5.7	6 min 54 sec

AppVeyor

JOB NAME

Environment: CMAKE_GENERATOR_NAME=Visual Studio 14 2015, MARIADB_VERSION=10.1.34

Environment: CMAKE_GENERATOR_NAME=Visual Studio 14 2015 Win64, MARIADB_VERSION=10.1.34

Environment: CMAKE_GENERATOR_NAME=Visual Studio 14 2015, MARIADB_VERSION=10.2.16

Environment: CMAKE_GENERATOR_NAME=Visual Studio 14 2015 Win64, MARIADB_VERSION=10.2.16

Environment: CMAKE_GENERATOR_NAME=Visual Studio 14 2015, MARIADB_VERSION=10.3.8

Environment: CMAKE_GENERATOR_NAME=Visual Studio 14 2015 Win64, MARIADB_VERSION=10.3.8

MySQL 8.0対応の現状

1. ビルドエラー修正
   - 完了
2. 新機能対応
   - 完了
3. テストをパスするようにする
   - 未完（全然おわんねーの）

ビルドエラー修正までの道

90コミット

API変更パターン

1. ふつうのC++になった
   - すごくよいこと
2. 名前が変わった
3. 新しいなにかが増えた

ふつうのC++

- my_bool→bool
- 必要なヘッダーだけ#include
- HASH→std::unordered_map
- std::stringを使える！
- autoを使える！

名前が変わった

- 型：st_select_lex→SELECT_LEX
- API：
  - order->direction == ORDER_ASC
  - order->direction == ORDER::ORDER_ASC
  - order->asc

対応方針

- 上位互換APIを用意して使う
  - 使わない引数は単に無視
- コード中で#ifしない
  - メンテナンスできないコードのできあがり！ 

互換型

```
#if MYSQL_VERSION_ID >= 80011 && !defined(MRN_MARIADB_P)
  class SELECT_LEX;
  typedef SELECT_LEX mrn_select_lex;
#else
  typedef st_select_lex mrn_select_lex;
#endif

// mrn_select_lex *select_lex = table_list->select_lex;
```

互換API

```
#if MYSQL_VERSION_ID >= 80011
#  define MRN_ORDER_IS_ASC(order) \
     ((order)->direction == ORDER_ASC)
#elif MYSQL_VERSION_ID >= 50603
#  define MRN_ORDER_IS_ASC(order) \
     ((order)->direction == ORDER::ORDER_ASC)
#else
#  define MRN_ORDER_IS_ASC(order) ((order)->asc)
#endif

// if (MRN_ORDER_IS_ASC(order)) {...}
```

新しいなにかが増えた

- dd::*が増えた
  - data dictionaryだって
  - テーブル定義の取得方法が変わった
- handlerにメンバー関数追加
  - 実装して対応しないといけない

対応方針

説明が面倒に
なってきたので
省略

新機能対応

- 新COLLATION対応
  - utf8mb4_0900_ai_ci
  - utf8mb4_0900_as_ci
  - utf8mb4_0900_as_cs
  - utf8mb4_ja_0900_as_cs
  - utf8mb4_ja_0900_as_cs_ks

groonga-normalizer-mysql

1.1.4で全部対応！

テストデータ

```
CREATE TABLE x (
  a text,
  FULLTEXT INDEX (a)
) ENGINE=Mroonga;
INSERT INTO x VALUES ("はひふへほ");
INSERT INTO x VALUES ("ばびぶべぼ");
INSERT INTO x VALUES ("ハヒフヘホ");
INSERT INTO x VALUES ("バビブベボ");
INSERT INTO x VALUES ("ﾊﾋﾌﾍﾎ");
INSERT INTO x VALUES ("ﾊﾞﾋﾞﾌﾞﾍﾞﾎﾞ");
```

確認方法

```
SELECT * FROM x
 WHERE MATCH(a)
       AGAINST('+バビブベボ' IN BOOLEAN MODE);
```

utf8mb4_0900_ai_ci

```
ALTER TABLE x MODIFY COLUMN a text
  COLLATE utf8mb4_0900_ai_ci;
-- +--------------------------------+
-- | a                              |
-- +--------------------------------+
-- | はひふへほ                     |
-- | ばびぶべぼ                     |
-- | ハヒフヘホ                     |
-- | バビブベボ                     |
-- | ﾊﾋﾌﾍﾎ                          |
-- | ﾊﾞﾋﾞﾌﾞﾍﾞﾎﾞ                    |
-- +--------------------------------+
```

utf8mb4_0900_as_cs

```
ALTER TABLE x MODIFY COLUMN a text
  COLLATE utf8mb4_0900_as_cs;
-- +------------------+
-- | a                |
-- +------------------+
-- | バビブベボ       |
-- +------------------+
```

utf8mb4_ja_0900_as_cs

```
ALTER TABLE x MODIFY COLUMN a text
  COLLATE utf8mb4_ja_0900_as_cs;
-- +--------------------+
-- | a                  |
-- +--------------------+
-- | ばびぶべぼ         |
-- | バビブベボ         |
-- +--------------------+
```

utf8mb4_ja_0900_as_cs_ks

```
ALTER TABLE x MODIFY COLUMN a text
  COLLATE utf8mb4_ja_0900_as_cs_ks;
-- +------------------+
-- | a                |
-- +------------------+
-- | バビブベボ       |
-- +------------------+
```

Mroongaの正規化

- MySQLとはステージが違う
- 濁点とかひらがなとかカタカナとか小文字とか絵文字とかそういうステージじゃないから

使用Unicodeのバージョン

- MySQL: 9.0.0
- Mroonga: 10.0.0

2018-07時点の最新Unicodeバージョン：11.0.0

使い方

```
CREATE TABLE x (
  a text,
  FULLTEXT INDEX (a)
-- MariaDBだとNORMALIZER='NormalizerNFKC100'と書ける！
    COMMENT='normalizer "NormalizerNFKC100"'
) ENGINE=Mroonga;
```

デフォルト

```
AGAINST('+バビブベボ' IN BOOLEAN MODE)
-- +--------------------+
-- | a                  |
-- +--------------------+
-- | バビブベボ         |
-- | ﾊﾞﾋﾞﾌﾞﾍﾞﾎﾞ         |
-- +--------------------+
```

unify_kana

```
-- COMMENT='normalizer
--   "NormalizerNFKC100(\'unify_kana\', true)"'
AGAINST('+ﾊﾞﾋﾞﾌﾞﾍﾞﾎﾞ' IN BOOLEAN MODE)
-- +-----------------+
-- | a               |
-- +-----------------+
-- | ばびぶべぼ      |
-- | バビブベボ      |
-- | ﾊﾞﾋﾞﾌﾞﾍﾞﾎﾞ      |
-- +-----------------+
```

unify_voiced_sound_mark

```
-- COMMENT='normalizer
--   "NormalizerNFKC100(\'unify_voiced_sound_mark\', true)"'
AGAINST('+バビブベボ' IN BOOLEAN MODE)
-- +--------------------------------+
-- | a                              |
-- +--------------------------------+
-- | ハヒフヘホ                     |
-- | バビブベボ                     |
-- | ﾊﾋﾌﾍﾎ                          |
-- | ﾊﾞﾋﾞﾌﾞﾍﾞﾎﾞ                    |
-- +--------------------------------+
```

unify_kana_case

```
INSERT INTO x VALUES ("やゆよ");
INSERT INTO x VALUES ("ゃゅょ");
INSERT INTO x VALUES ("ヤユヨ");
INSERT INTO x VALUES ("ャュョ");
INSERT INTO x VALUES ("ﾔﾕﾖ");
INSERT INTO x VALUES ("ｬｭｮ");
```

unify_kana_case

```
-- COMMENT='normalizer
--   "NormalizerNFKC100(\'unify_kana_case\', true)"'
AGAINST('+ヤユヨ' IN BOOLEAN MODE)
-- +-----------+
-- | a         |
-- +-----------+
-- | ヤユヨ    |
-- | ヤユヨ    |
-- | ﾔﾕﾖ       |
-- | ﾔユヨ     |
-- +-----------+
```

unify_hyphen

- ハイフンっぽい文字を
  - ‐‑‒–—―⁃⁻₋−
- ハイフンへ
  - - (U+002D)
- ユースケース：電話番号検索

unify_prolonged_sound_mark

- 長音記号っぽい文字を
  - ー﹘—‐－
- 長音記号へ
  - ー (U+30FC)
- ユースケース：電話番号検索

unify_hyphen_and_prolonged_sound_mark

- ハイフンっぽい文字と
  - ‐‑‒–—―⁃－
- 長音記号っぽい文字を
  - ー﹘－
- ハイフンへ
  - - (U+002D)

unify_middle_dot

- 中点っぽい文字を
  - ･･･•●･ ・・
- 中点へ
  - ・ (U+00B7)
- ユースケース：外来語検索

unify_katakana_v_sounds

- ヴァヴィヴヴェヴォを
- バビブベボへ
- ユースケース：外来語検索（ワイン名）

unify_katakana_bu_sound

- ヴァヴィヴヴェヴォを
- ブへ
- ユースケース：外来語検索（ワイン名）

オプションは組み合わせ可能

```
-- ワイン名検索用
-- COMMENT='normalizer
--   "NormalizerNFKC100(
--     \'unify_middle_dot\', true,
--     \'unify_katakana_bu_sound\', true)"'
```

トークナイザー

- MySQLとはステージが違う
- ゆるく検索するモード
  - 電話番号・ワイン名検索に便利

ゆるく検索

- loose_symbol
  - 記号の有無に関係なくマッチ
- loose_blank
  - 空白文字の有無に関係なくマッチ

書き方

```
-- MariaDBだと↓と書ける！
--   TOKENIZER='TokenNgram("loose_symbol", true)'
-- COMMENT='tokenizer
--   "TokenNgram(\'loose_symbol\', true)"'
```

電話番号検索例

```
INSERT INTO x VALUES ('(123)4567-8901');
INSERT INTO x VALUES ('123-4567-8901');
INSERT INTO x VALUES ('12345678901');
INSERT INTO x VALUES (' (１２３) ４５６７-８９０１');
INSERT INTO x VALUES ('123 4567 8901');
```

電話番号検索例

```
-- NormalizerNFKC100('unify_hyphen_and_prolonged_sound_mark', true)
-- TokenNgram('loose_symbol', true,
--            'loose_blank', true)
AGAINST('+１２３4567−8901' IN BOOLEAN MODE)
-- +-----------------------------------------+
-- | a                                       |
-- +-----------------------------------------+
-- | (123)4567-8901                          |
-- | 123-4567-8901                           |
-- | 12345678901                             |
-- |  （１２３）４５６７−８９０１            |
-- | 123 4567 8901                           |
-- +-----------------------------------------+
```

ワイン名検索例

```
INSERT INTO x VALUES ('セーヴェル エ メーヌ');
INSERT INTO x VALUES ('セブルエメーヌ');
INSERT INTO x VALUES ('セーブル・エ・メーヌ');
INSERT INTO x VALUES ('セーヴル　エメーヌ');
```

ワイン名検索例

```
-- NormalizerNFKC100('unify_middle_dot', true,
--                   'unify_hyphen_and_prolonged_sound_mark', true,
--                   'unify_katakana_bu_sound', true)
-- TokenNgram('loose_symbol', true,
--            'loose_blank', true)
AGAINST('+セーヴェルエメーヌ' IN BOOLEAN MODE)
-- +--------------------------------+
-- | a                              |
-- +--------------------------------+
-- | セーヴェル エ メーヌ           |
-- | セブルエメーヌ                 |
-- | セーブル・エ・メーヌ           |
-- | セーヴル　エメーヌ             |
-- +--------------------------------+
```

テストがパスしない原因

- そもそも動かない
- 動くけどクラッシュする
- 動くけど期待通りじゃない
- 動くし期待通りだけど期待通りじゃない

そもそも動かない例

- mysql-test/include/*が減った
  - have_innodb.inc：InnoDB必須だから
  - not_embedded.inc：libmysqldを辞めたから
- 互換.incを用意

動くけどクラッシュする例

- JSON型のカラムの更新
- ラッパーモード
  - 使っている人いる？
  - 削除していい？

動くけど期待通りじゃない例

- FOREIGN KEYの制約チェック
  - dd::*からうまく主キー情報を取得できていない

動くし期待通りだけど期待通りじゃない例

- デフォルト値変更→期待結果も変わる
  - 例：COLLATIONの変更
  - SHOW CREATE TABLEの結果が変わる
- テスト方法・結果の変更で対応
  - 例：非デフォルトCOLLATIONを使う
  - ただただ面倒

テストがまだパスしていない要因

- テスト実行が遅くなった
- JSONのやつがなんで動かなくなったかわからん
- dd::*の使い方がわからん

テスト実行が遅い

- mysql-test-runが遅くなった
  - テスト実行開始まで30秒くらい
  - メモリー3GB食うようになった
  - Mroongaのテストは約770個
- gdbの起動が遅くなった
  - mysqldのシンボル読込に15秒くらい

テストがパスするための支援方法

- クリアコードに開発案件を発注
- 仕事の時間で開発できる！

次回リリース

- 8月か9月の肉の日かなぁ
- db tech showcase Tokyo 2018の前にリリースできるといいなぁ