

Oracle ACEが語る！

# 「ここがヘンだよMySQL & ここがスゴイよMySQL」

日本MySQLユーザー会

<http://www.mysql.gr.jp>



# 本日の登壇者

#mysql\_jp

- 平塚 貞夫 NTTコムウェア株式会社  
(Oracle ACE Alumni)
- @yoku0825 (Oracle ACE)
- Ryuta Kamizono (Oracle ACE Alumni)
- 三谷 智史 ヤフー株式会社 (Oracle ACE)
- とみたまさひろ 日本MySQLユーザ会代表  
(Oracle ACE Associate)



MySQL  
Nippon  
Association



# ここがヘンだったよMySQL

## Oracle Code Tokyo 2019

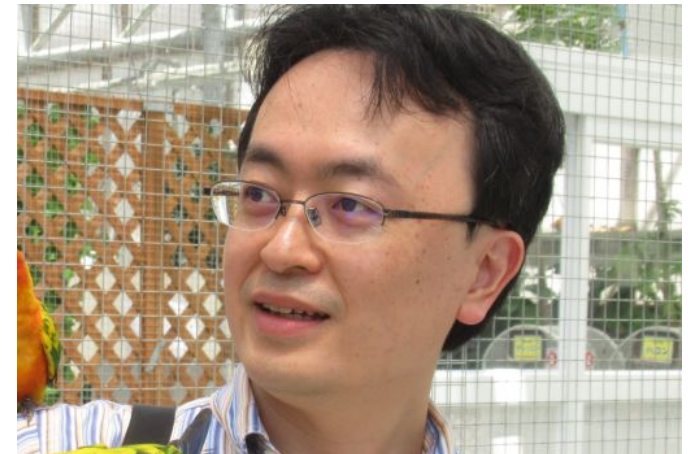
2019/05/17

NTTコムウェア株式会社

平塚貞夫

# 自己紹介

- システムインテグレータで開発/運用の技術支援をしています。
  - Oracle DatabaseとMySQLが主な守備範囲です。
  - 長い間DB専門でやっていましたが、最近は少し手を広げています。
- Twitter : @sh2nd
- はてな : sh2
- Oracle ACE Alumni (アラムナイと読むそうです)



# ここがヘンだったよMySQL

MySQLのヘンなところを挙げるときりがありませんね！

私はSQLより下のレイヤを中心に、過去ヘンだったけれど現在は良くなったところを3点ピックアップしてみました。

- CPUコア数を増やしても性能が上がりなかった
- バイナリログ形式のデフォルトがSTATEMENTだった
- トランザクションログの同期書き込みをサボっていた

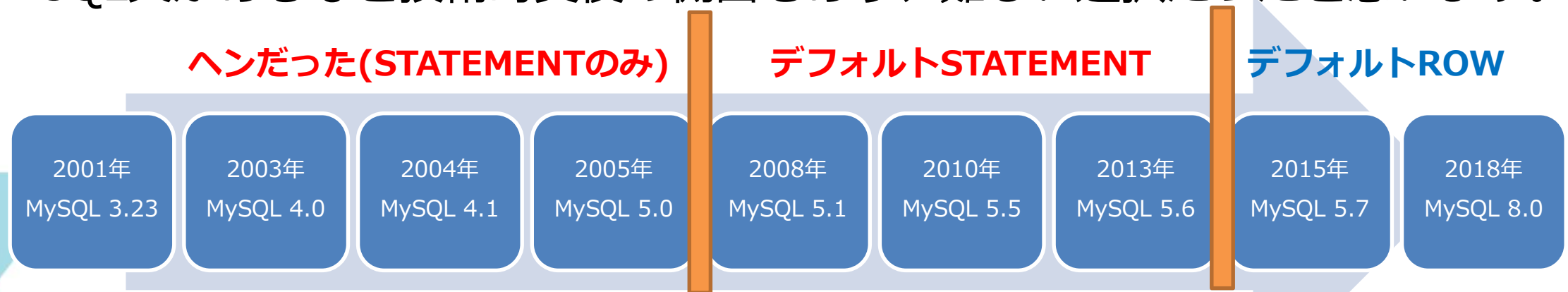
# CPUコア数を増やしても性能が上がりなかった

- MySQL 5.0まではInnoDBがバッファキャッシュを操作する際にジャイアントロックを取得しており、CPUコア数を増やしても性能がほとんど上がりませんでした。むしろ下がることも…。
- Xeonプロセッサがクアッドコアになったのが2006年なので、そのあたりでこの課題が目立ってきたようです。
- MySQL 5.1に導入されたInnoDB Pluginで大幅に手が入り、MySQL 5.5では標準のInnoDBで十分なCPUスケーラビリティが得られるようになりました。



# バイナリログ形式のデフォルトがSTATEMENTだった

- MySQLはバイナリログ形式をSTATEMENT(SQL文)とROW(行データ)から選択できます。MySQL 5.6まではデフォルトがSTATEMENTでした。
- これは他のRDBMSとは逆のスタンスでした。例えばOracle Databaseではフィジカル・スタンバイを基本とし、特別な要件に対してロジカル・スタンバイを利用します。
- MySQLは最初にSTATEMENTを選択したことでスケールアウト構成が容易になり、MySQLが広く普及する要因の一つになりました。ただ、扱いに困るSQL文があるなど技術的負債の側面もあり、難しい選択だったと思います。



# トランザクションログの同期書き込みをサボっていた

- RDBMSとしてDurabilityを担保するために、COMMITされたデータは即時ストレージに同期してほしいところです。
- これはInnoDBのinnodb\_flush\_log\_at\_trx\_commit、バイナリログのsync\_binlogを両方1(有効)にすることで実現できるのですが、sync\_binlogのデフォルト値が1になったのはMySQL 5.7からでした。
- またMySQL 5.5まではsync\_binlogを1にした際の性能低下が大きかったため、性能と信頼性のトレードオフを慎重に検討する必要がありました。



# 所感

私がMySQL 5.0を触り始めたときに感じた違和感は、ほぼ解消しました。  
(もちろん新たな違和感が出てくることもありますが)

かれこれ20年以上開発が継続しているのは、本当に素晴らしいことです。

MySQLは、Oracle Corporation及びその子会社、関連会社の米国及びその他の国における登録商標です。文中の社名、商品名等は各社の商標または登録商標である場合があります。

# HUGっと！ Oracle ACE(MySQL)

ここは変だよ、ここも変だよMySQL

2019/05/17

yoku0825

Oracle Code Tokyo 2019

# Safe Harbor Statement

- この資料はyoku0825の独断と偏見によるものであり、所属する組織、所属しない組織またはNULLの意見を一切代表する訳がありません



# HUGっと! Oracle ACE(MySQL)

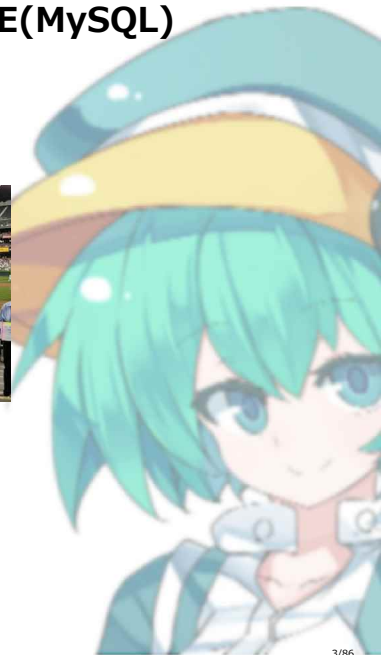


# HUGっと！ Oracle ACE(MySQL)

すごいプリキュア（語彙）  
キュア松信さん

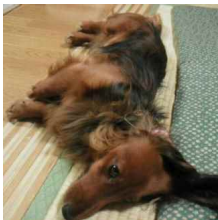


- <https://twitter.com/matsunobu>
- [Open database life](#)

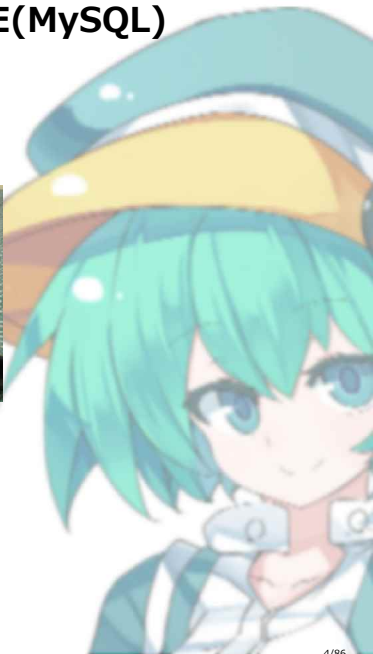


# HUGっと！ Oracle ACE(MySQL)

InnoDBのプロキュア  
キュアSH2さん

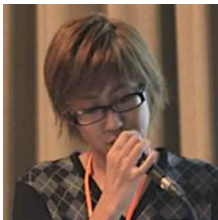


- <https://twitter.com/sh2nd>
- [SH2の日記](#)

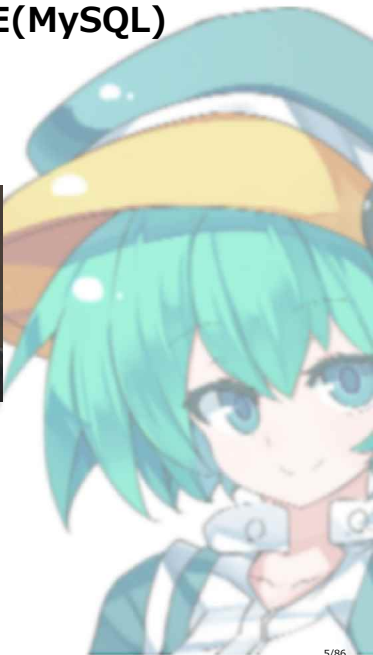


# HUGっと！ Oracle ACE(MySQL)

ActiveRecordのプリオー  
キュアkamipoさん



- <https://twitter.com/kamipo>
- [かみぽわーる](#)



# HUGっと！ Oracle ACE(MySQL)

運用のプロキュア  
キュアmita2さん



- <https://twitter.com/mita2>
- [mita2 DB メモ](#)

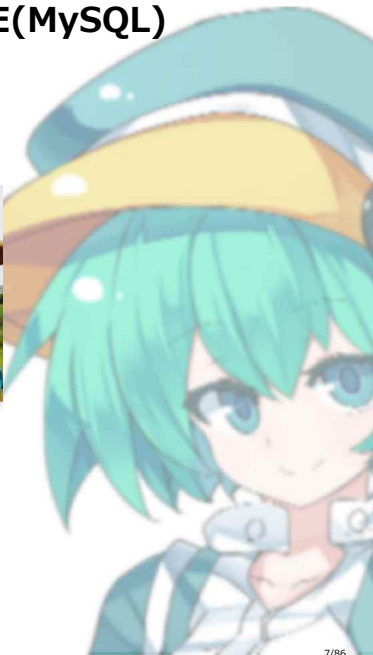


# HUGっと！ Oracle ACE(MySQL)

文字化けのプリキュア  
キュアとみたさん



- <https://twitter.com/tmtms>
- [@tmtms のメモ](#)



そしてこ  
の時間は

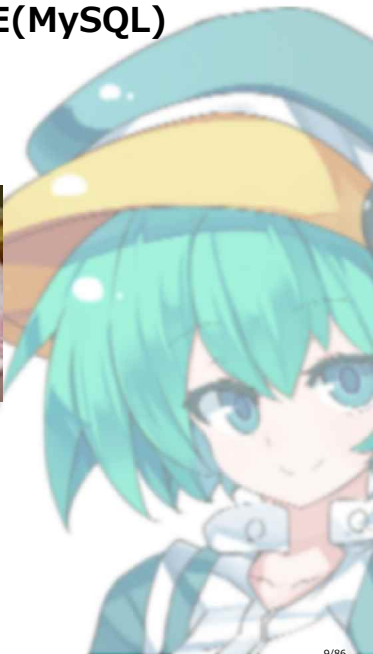


# HUGっと！ Oracle ACE(MySQL)

地雷のプリキュア  
キュアyoku0825



- <https://twitter.com/yoku0825>
- [日々の覚書](#)



がお送り  
します



# ＼こんにちは／

- yoku0825@とある企業のDBA

- オラクれない
- ポスグレない
- マイエスキューエる

- 生息域

- Twitter: [@yoku0825](https://twitter.com/yoku0825)
- Blog: [日々の覚書](#)
- [日本MySQLユーザ会](#)
- [MySQL Casual](#)



ここがヘンだよ  
MySQL & ここが  
スゴイよMySQL

# ここがヘンだよMySQL & ここがスゴイよMySQL

- yoku0825@とある企業のDBA
  - オラクれない
  - ポスグれない
  - マイエスキューエる
- MySQL.Optimizer < Oracle.Optimizer ?
- MySQL.Replication > PostgreSQL.Replication ?



# ここがヘンだよMySQL & ここがスゴイよMySQL

- yoku0825@とある企業のDBA
  - **オラクれない**
  - ポスグれない
  - マイエスキューエる
- MySQL. Optimizer < Oracle. Optimizer ?
- MySQL. Replication > PostgreSQL. Replication ?



# ここがヘンだよMySQL & ここがスゴイよMySQL

- yoku0825@とある企業のDBA
  - **オラクれない**
  - ポスグレない
  - マイエスキューエる
- MySQL. Optimizer < **NULL**
- MySQL. Replication > PostgreSQL. Replication ?



# ここがヘンだよMySQL & ここがスゴイよMySQL

- yoku0825@とある企業のDBA
  - オラクれない
  - **ポスグレない**
  - マイエスキューエる
- MySQL. Optimizer < NULL
- MySQL. Replication > PostgreSQL. Replication ?



# ここがヘンだよMySQL & ここがスゴイよMySQL

- yoku0825@とある企業のDBA
  - オラクれない
  - **ポスグレない**
  - マイエスキューエる
- MySQL.Optimizer < NULL
- MySQL.Replication > **NULL**



ここがヘンだよ  
MySQL & ここが  
スゴイよMySQL  
**IS NULL**



# 糸冬

=====

制作・著作：

yoku0825



という訳で気  
楽に話します



# Do you love MySQL 8.0?

```
mysql80 160> SELECT CURDATE() /* JST */;
```

CURDATE()
2018-04-20

1 row in set (0.00 sec)

```
mysql80 160> SELECT @@version;
```

@@version
8.0.11

1 row in set (0.00 sec)



# Do you love MySQL 8.0?

```
mysql80 101493> SELECT CURDATE() /* JST */;
```

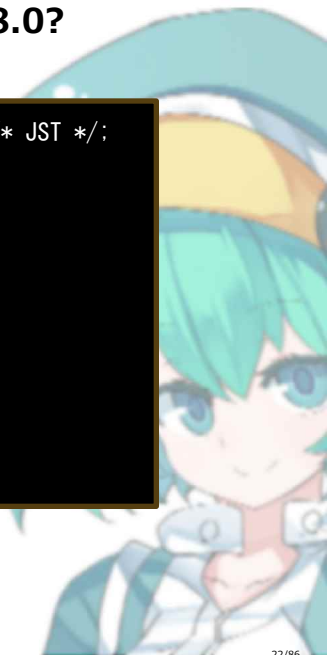
CURDATE()
2019-05-17

```
1 row in set (0.00 sec)
```

```
mysql80 101493> SELECT @@version;
```

@@version
8.0.16

```
1 row in set (0.00 sec)
```



## ここ は 変だよMySQL

- ナッツシエル ( What is New in MySQL 8.0 の章) が何故か  
どんどん増える

```
$ curl -L -s https://dev.mysql.com/doc/refman/8.0/en/mysql-nutshell.html | perl -nIE 'if ($_ =~ /(8\.\d+\.\d+)/) { say $1 }' | sort  
-t "." -k 3 -n | uniq -c  
1 8.0.0  
3 8.0.2  
3 8.0.3  
2 8.0.4  
2 8.0.12  
6 8.0.13  
6 8.0.14  
8 8.0.16
```

GA #とはな  
んだったのか



**時間ないん  
で手短に**



# MySQL

## 8.0.15、本番

## 導入しました！



**ド新規でいきたかつ  
たんですがバージョ  
ンアップです！**



# インスタンスの特徴

- 2011年にMySQL 5.5でサービス開始
  - 文字コードは (3バイト) UTF-8
  - 接続元アプリケーションはJava
- 2016年、MySQL 5.6バージョンアップ
- 2018年、アプリケーションのフルスクラッチリファクタリング (?) が決定
  - 接続元アプリケーションはPHP(Laravel)に変更
  - MySQLも8.0にバージョンアップすることが決定



# インスタンスの特徴

- 2011年にMySQL 5.5でサービス開始
  - 文字コードは (3バイト) UTF-8 ←救いだった
  - 接続元アプリケーションはJava
- 2016年、MySQL 5.6バージョンアップ
- 2018年、アプリケーションのフルスクラッチリファクタリング (?) が決定
  - 接続元アプリケーションはPHP(Laravel)に変更
  - MySQLも8.0にバージョンアップすることが決定



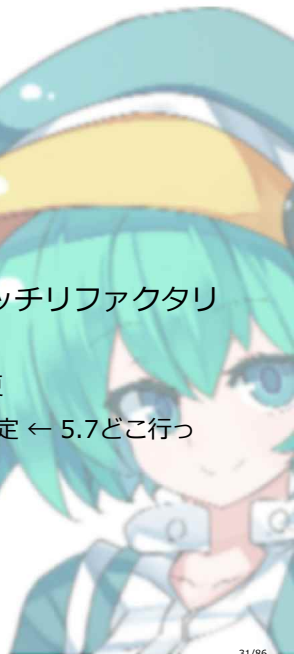
# インスタンスの特徴

- 2011年にMySQL 5.5でサービス開始
  - 文字コードは (3バイト) UTF-8 ←救いだった
  - 接続元アプリケーションはJava
- 2016年、MySQL 5.6バージョンアップ
- 2018年、アプリケーションのフルスクラッチリファクタリング (?) が決定
  - 接続元アプリケーションはPHP(Laravel)に変更
  - MySQLも8.0にバージョンアップすることが決定 ← **5.7どこ行きた？**



# インスタンスの特徴

- 2011年にMySQL 5.5でサービス開始
  - 文字コードは (3バイト) UTF-8 ←救いだっただ
  - 接続元アプリケーションはJava
- 2016年、MySQL 5.6バージョンアップ
- 2018年、アプリケーションのフルスクラッチリファクタリング (?) が決定
  - 接続元アプリケーションはPHP(Laravel)に変更
  - MySQLも8.0にバージョンアップすることが決定 ← 5.7どこ行っただ？
    - データ作り直しだからいいんじゃないね？



# インスタンスの特徴

- 2018年、アプリケーションのフルスクラッチリファクタリング（？）が決定
  - 接続元アプリケーションはPHP(Laravel)に変更
  - MySQLも8.0にバージョンアップすることが決定 ← 5.7どこ行った？
    - データ作り直しだからいいんじゃない？
- **2018年秋、既存のデータもアプリも捨てられないことが確定**



# インスタンスの特徴

- 2018年、アプリケーションのフルスクラッチリファクタリング（？）が決定
  - 接続元アプリケーションはPHP(Laravel)に変更
  - MySQLも8.0にバージョンアップすることが決定 ← 5.7どこ行った？
    - データ作り直したからいいんじゃないね？
- 2018年秋、既存のデータもアプリも捨てられないことが確定
  - **MySQL 5.6から8.0へのレプリケーションでのデータ移行**



# インスタンスの特徴

- 2018年、アプリケーションのフルスクラッチリファクタリング（？）が決定
  - 接続元アプリケーションはPHP(Laravel)に変更
  - MySQLも8.0にバージョンアップすることが決定 ← 5.7どこ行った？
    - データ作り直しだからいいんじゃない？
- 2018年秋、既存のデータもアプリも捨てられないことが確定
  - MySQL 5.6から8.0へのレプリケーションでのデータ移行
  - **しかも2011年から継ぎ足され続けた秘伝のJavaがつなぎに来る**

## インスタンスの特徴

- 2018年秋、既存のデータもアプリも捨てられないことが確定
  - MySQL 5.6から8.0へのレプリケーションでのデータ移行
  - しかも2011年から継ぎ足され続けた秘伝のJavaがつなぎに来る
- **2018年冬、気が付く**



# インスタンスの特徴

- 2018年秋、既存のデータもアプリも捨てられないことが確定
  - MySQL 5.6から8.0へのレプリケーションでのデータ移行
  - しかも2011年から継ぎ足され続けた秘伝のJavaがつなぎに来る
- 2018年冬、気が付く
  - **5.6から8.0へのレプリケーション、死ぬわ…**



# インスタンスの特徴

- 2018年秋、既存のデータもアプリも捨てられないことが確定
  - MySQL 5.6から8.0へのレプリケーションでのデータ移行
  - しかも2011年から継ぎ足され続けた秘伝のJavaがつなぎに来る
- 2018年冬、気が付く
  - 5.6から8.0へのレプリケーション、死ぬわ…
    - 主にSQLスレッドと俺の心が



# インスタンスの特徴

- 2018年秋、既存のデータもアプリも捨てられないことが確定
  - MySQL 5.6から8.0へのレプリケーションでのデータ移行
  - しかも2011年から継ぎ足され続けた秘伝のJavaがつなぎに来る
- 2018年冬、気が付く
  - 5.6から8.0へのレプリケーション、死ぬわ…
    - 主にSQLスレッドと俺の心が
    - サポートされない構成だしな



# インスタンスの特徴

- 2018年冬、気が付く
  - 5.6から8.0へのレプリケーション、死ぬわ…
    - 主にSQLスレッドと俺の心が
    - サポートされない構成だしな
  - 「せめて今のマスターを5.7にバージョンアップせねば…」



# インスタンスの特徴

- 2018年冬、気が付く
  - 5.6から8.0へのレプリケーション、死ぬわ…
    - 主にSQLスレッドと俺の心が
    - サポートされない構成だしな
  - 「せめて今のマスターを5.7にバージョンアップせねば…」
  - **「調整の結果、新規の開発とは別件としてバージョンアップを進めることになりました」**



# インスタンスの特徴

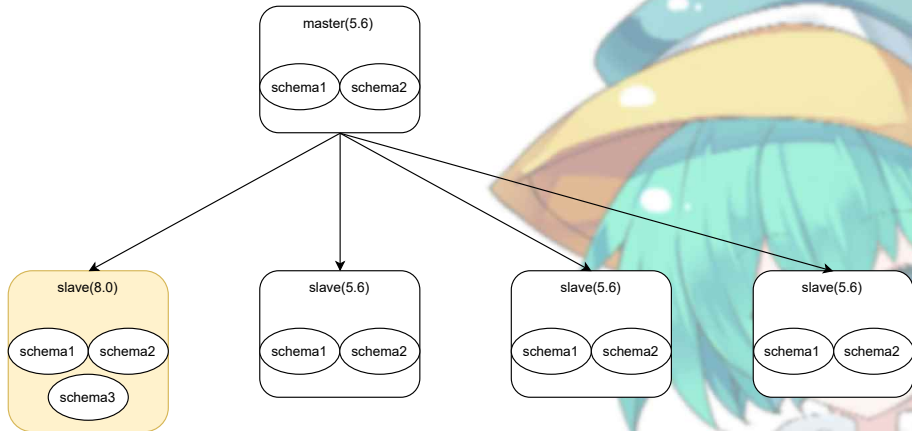
- 2018年冬、気が付く
  - 5.6から8.0へのレプリケーション、死ぬわ…
    - 主にSQLスレッドと俺の心が
    - サポートされない構成だしな
  - 「せめて今のマスターを5.7にバージョンアップせねば…」
  - 「調整の結果、新規の開発とは別件としてバージョンアップを進めることになりました」
  - 「メンテナンス日程は新規開発のリリースメンテと合わせてやることになりました」



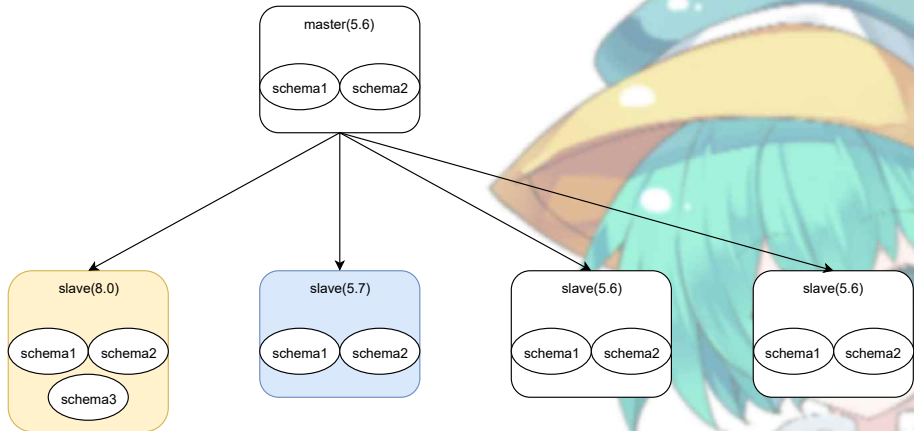
という訳  
で



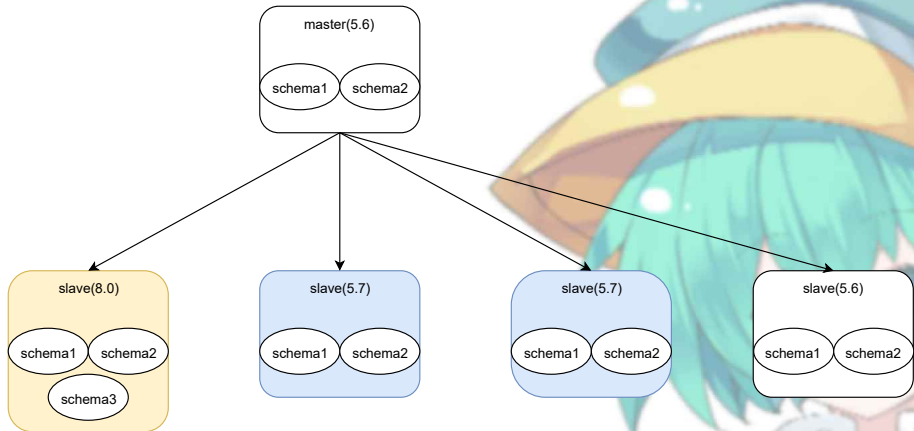
# 開発開始時



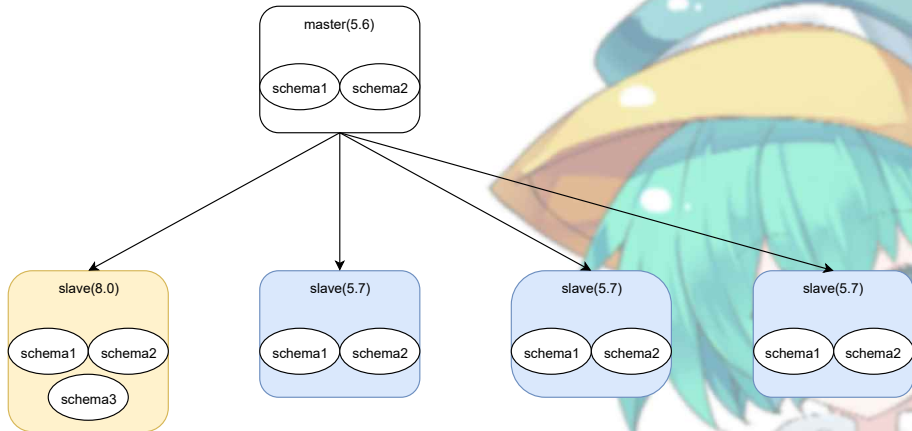
# スレーブから先にバージョンを上げて



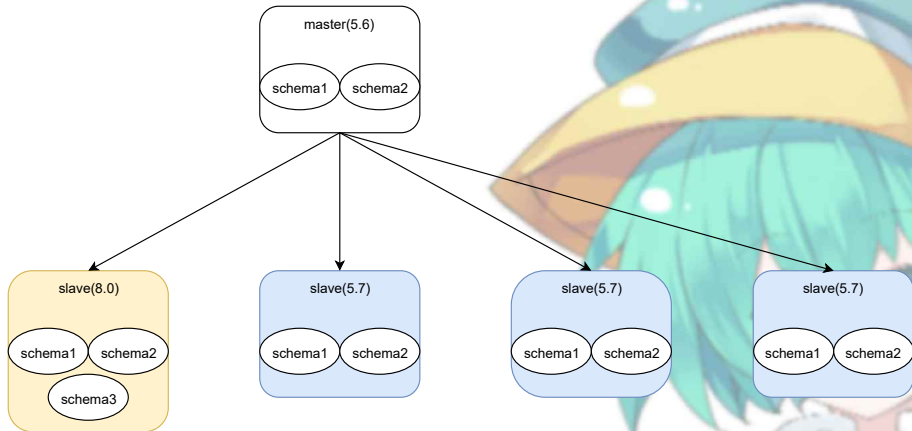
# 上げて



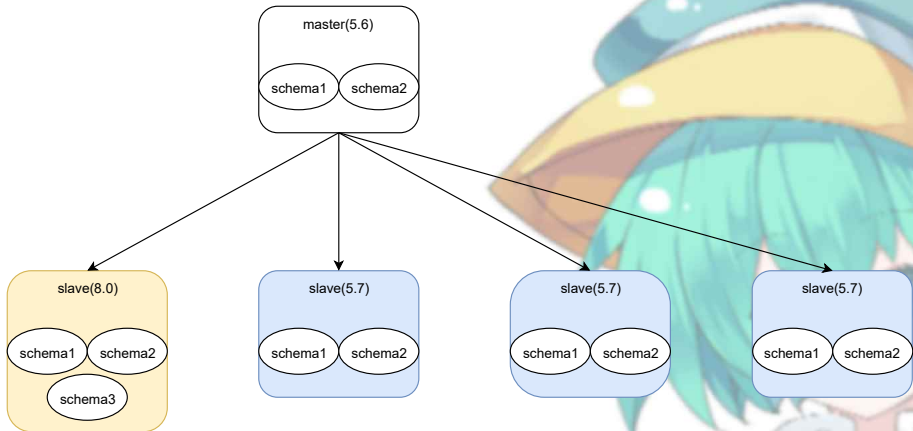
# 上げて



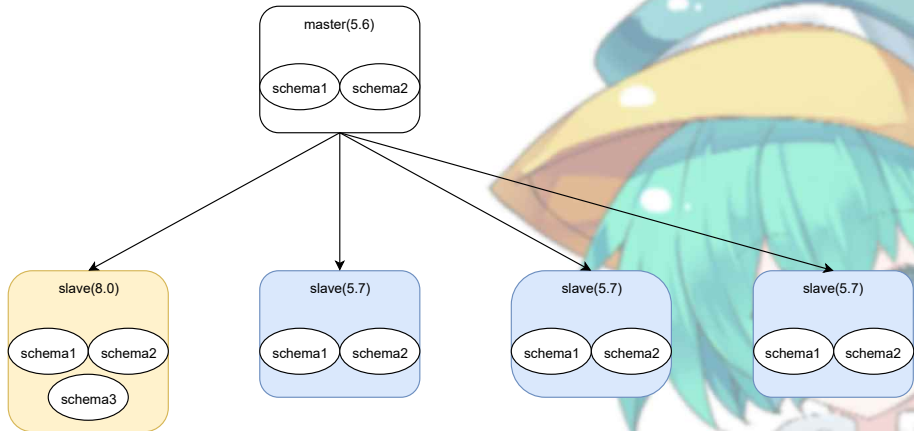
# 開発が終わり



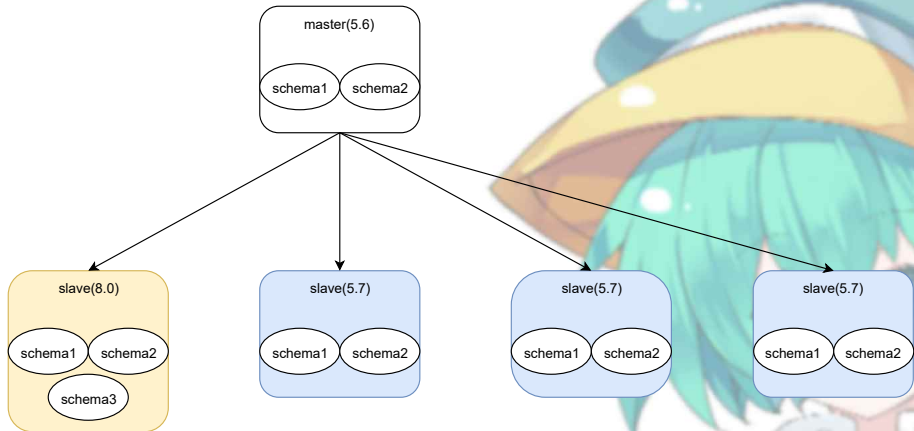
# データコンバートのリハーサルがされ



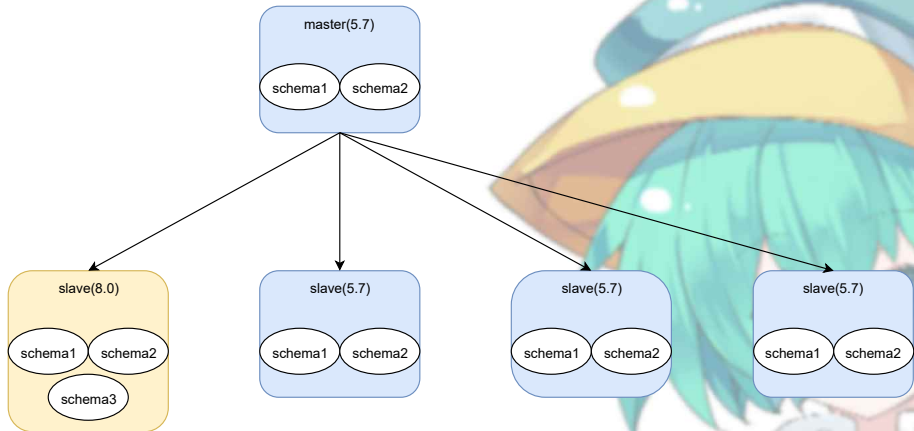
# カッターバー当日



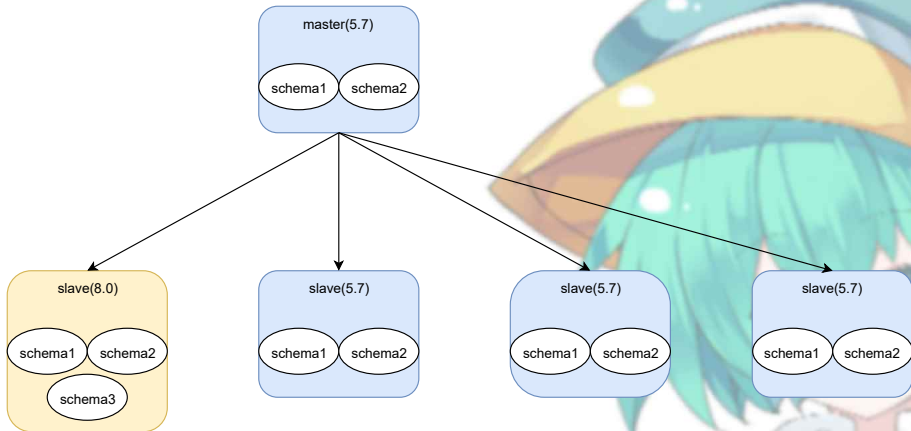
# データのコンバートが終わってから



# ガシャツ



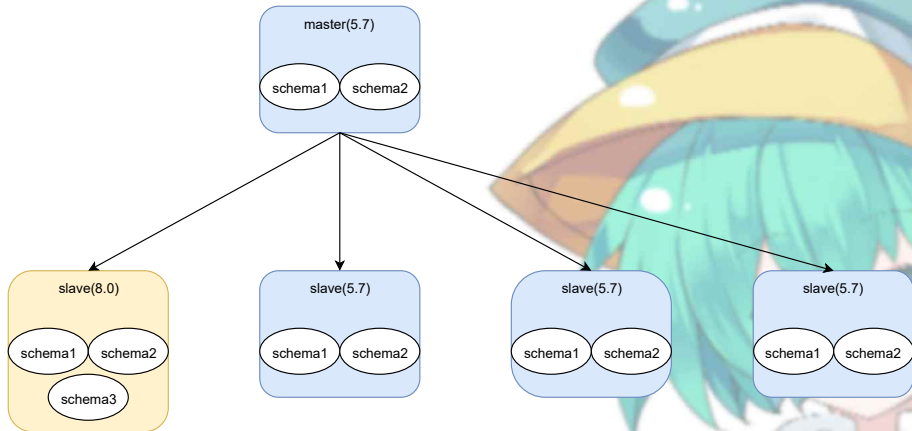
# やった5.7から8.0ならレプリケーションが上手く…



:(;`°'ω°'): な  
んか俺が考えて  
たのと違う



# 今もこのまま動いています



**「ぼくらが8.0にいた  
るまでのみちのり  
(シヨート)」**



## mysqldumpを8.0にリストアする

- おれ「まずはMySQL 5.6から取ったダンプを8.0にインポートしよう」
- ハッテンゼロ「 'utf8' is currently an alias for the character set UTF8MB3, but will be an alias for UTF8MB4 in a future release. Please consider using UTF8MB4 in order to be unambiguous. 」
- おれ「びっくりするくらいコンソールが流れた…」



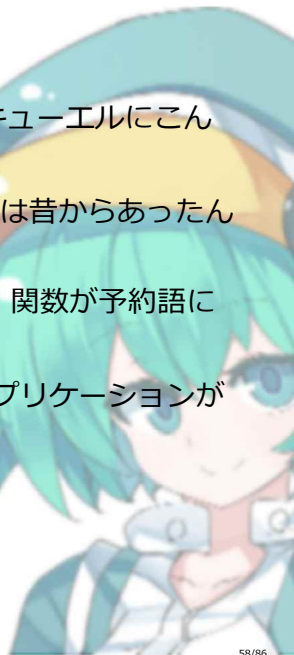
## 新しい構文を使う(1)

- おれ「再帰でないCTE(WITH句)便利～」
- ハッテンゼロ「ただし速くなるとは誰も言ってない」
- おれ「やはり CREATE TEMPORARY TABLE ... Engine = MyISAM が最強だったか…」



## 新しい構文を使う(2)

- かい はつ「Window関数便利！ マイエスキューエルにこんな便利なものがあったなんて！」
- おれ「(´-` ).oO (ポスグレやオラクルには昔からあったんやで…
- ハッテンゼロ「ただし rank カラムは RANK 関数が予約語になっているのでクオートしなければ殺す」
- おれ「あっ binlog\_format <> ROW だとレプリケーションが死ぬ」

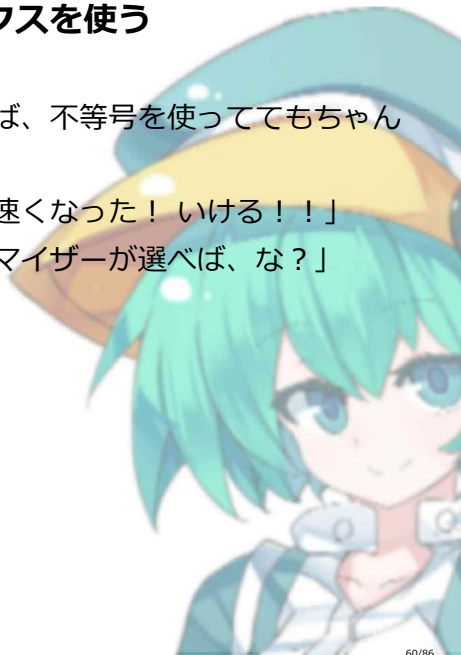


## 5.6からのレプリケーションはアカウント周りがつらい

- おれ「5.6なマスターにアカウントを追加するお。ハッテンゼロは `GRANT USAGE ON ..` でアカウントを作ろうとするとエラーになるから `CREATE USER` だお！」
- ゴーテンロク「生パスワードがbinlogに書かれないように、クエリーを変形させて書くよん」
- ゴーテンロク「`CREATE USER 'yoku0825'@'%' IDENTIFIED BY PASSWORD '*4266488C892EA7950486FEC0A1CFFC1BD9543F7B'`」
- ハッテンゼロ「You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'PASSWORD '\*4266488C892EA7950486FEC0A1CFFC1BD9543F7B'」
- おれ「またレプリケーションが死んだ…」

## 式インデックスを使う

- おれ「式インデックスがあれば、不等号を使っててもちゃんとインデックスが使える！」
- おれ「確かにテスト環境では速くなった！ いける！！」
- 本番ハッテンゼロ「オプティマイザーが選べば、な？」
- おれ「うっそ…」



# information\_schemaにアクセスするスクリプトが転ける

- おれ「あれ yt-healthcheck がなんか変」
- おれ「今までは SELECT \* で取ればカラム名が大文字だったけど、小文字でカラム名指定すれば返ってきていたのに…？」



# information\_schemaにアクセスするスクリプトが転ける

```
mysql57 6> SELECT table_schema, table_name, data_length, index_length, data_free FROM information_schema.tables WHERE (table_schema, table_name)= ('d1', 't1');
```

table_schema	table_name	data_length	index_length	data_free
d1	t1	16384	0	0

1 row in set (0.03 sec)

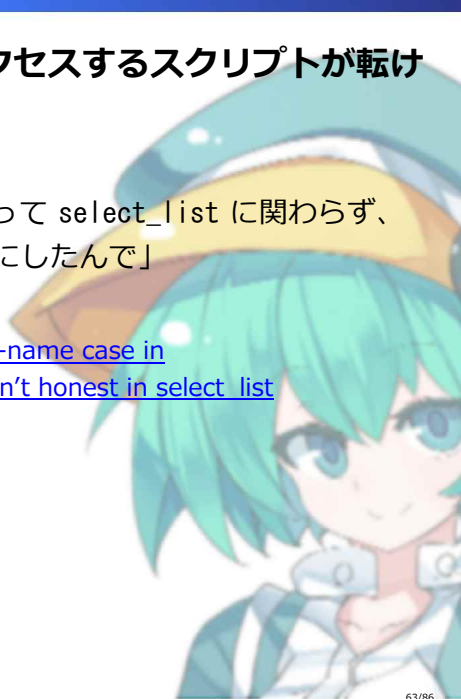
```
mysql80 11> SELECT table_schema, table_name, data_length, index_length, data_free FROM information_schema.tables WHERE (table_schema, table_name)= ('d1', 't1');
```

TABLE_SCHEMA	TABLE_NAME	DATA_LENGTH	INDEX_LENGTH	DATA_FREE
d1	t1	16384	0	0

1 row in set (0.05 sec)

# information\_schemaにアクセスするスクリプトが転ける

- ハッテンゼロ「SQL標準に則って select\_list に関わらず、  
カラム名は大文字で返すことにしたんで」
- MySQL Bugs「Won't fix」
  - [MySQL Bugs: #90852: field¥-name case in informaiton schema.tables isn't honest in select list](#)
- おれ「成歩堂」



# SHOW TABLE STATUS の結果が変わらない

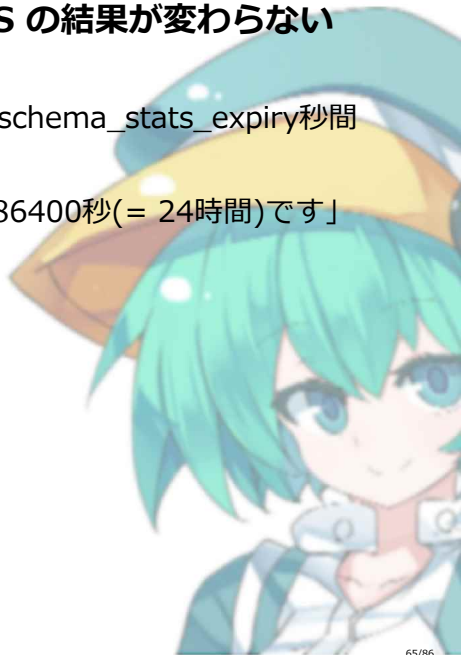
- おれ「INSERTしまくってもdata\_lengthもauto\_incrementも増えない…？」

```
mysql> SELECT COUNT(*) FROM t1;
+-----+
| COUNT(*) |
+-----+
|    10000 |
+-----+
1 row in set (0.00 sec)

mysql> SHOW TABLE STATUS\G
***** 1. row *****
      Name: t1
      Engine: InnoDB
      Version: 10
      Row_format: Dynamic
        Rows: 0
    Avg_row_length: 0
      Data_length: 16384
    Max_data_length: 0
      Index_length: 0
        Data_free: 0
    Auto_increment: NULL
      Create_time: 2019-05-16 15:41:52
      Update_time: NULL
      Check_time: NULL
      Collation: utf8mb4_0900_ai_ci
      Checksum: NULL
    Create_options:
      Comment:
1 row in set (0.00 sec)
```

## SHOW TABLE STATUS の結果が変わらない

- ハッテンゼロ「information\_schema\_stats\_expiry秒間  
キャッシュしてます」
- ハッテンゼロ「デフォルトは86400秒(= 24時間)です」
- おれ「誰得」



## sql\_require\_primary\_key(1)

- おれ「PRIMARY KEYのないテーブルを作ろうとするとエラーにできる sql\_require\_primary\_key が追加された！」
- かいはつ「試しにちょっとテーブル作ってみよう」
- ハッテンゼロ「Last\_SQL\_Error: Unable to create or change a table without a primary key, when the system variable 'sql\_require\_primary\_key' is set. 」
- おれ「まだ死ぬのかスレーブ…」

## sql\_require\_primary\_key(2)

- おれ「まさか本当にPRIMARY KEYがないテーブルを作るとは… SET GLOBAL sql\_require\_primary\_key = OFF っ  
と」
- ハッテンゼロ「Last\_SQL\_Error: Unable to create or change a table without a primary key, when the system variable 'sql\_require\_primary\_key' is set. 」
- おれ「!?!?!?!」
- ハッテンゼロ「ごめんな、SQLスレッドは起動した時のグローバル値をセッション値にコピーしてそのまま使うんだ」
- おれ「なるほどそう来たか」
  - ばぐれほしい…

# アトミックDDL

- ディービーエー「 DROP TABLE t1, t12 あっtypo」
- 5.7マスター「t12はUnknown Tableだけどt1消しちゃったからerror\_number= 1051でバイナリログに記録」
- 5.7スレーブ「t12はUnknown Tableだけどバイナリログにも1051って書いてあるからおk」



# アトミックDDL

- ハッテンゼロ「A commit for an atomic DDL statement was unsuccessful on the master and the slave. The slave supports atomic DDL statements but the master does not, so the action taken by the slave and master might differ. Check that their states have not diverged before proceeding. 」
- おれ「なるほどこれはMySQLっぽい」
- ディービーエー「m j s k」



## ロールとアカウントの区別 #とは

- おれ「mysql.user テーブル上はロールとアカウントの区別ってなさそう。DROP ROLE をアカウントに打ち込んでみたら消えたりしてwww」
- ハッテンゼロ「Query OK, 0 rows affected (0.00 sec)」
- おれ「DROP ROLE root@localhost でrootアカウントが消えた…」
- Umesh「Thank you for the report ad feedback.»
  - [MySQL Bugs: #93263: DROP ROLE username should be rejected](#)
  - **このバグは直ってます**

## パーティショニングの移行はたいへん(1)

- おれ「5.6の物理バックアップを8.0にリストアできるかしらん？」
- ハッテンゼロ「InnoDBネイティブじゃないパーティションとかデ々村」
- おれ「ですよ」
- おれ「でも8.0にも ALTER TABLE .. UPGRADE PARTITIONING 構文あるじゃん？ 起動してアップグレードできないのん？」
- MySQL Bugs「Thank you for your bug report. This issue has been addressed in the documentation. The updated documentation will appear on our website shortly.」
- おれ「なるほどドキュメントの方がバグ扱い」

## パーティショニングの移行はたいへん(2)

- おれ「仕方ない、5.6から5.7にインプレースアップグレードしてからmysqldump取って8.0に突っ込も。これならInnoDBネイティブパーティショニングだNE☆」
- ハッテンゼロ「core dumped」
- おれ「えっ嘘」
- おれ「パーティショニング全部引っぺがしたらクラッシュしなくなった…」
- ハッテンゼロ「たぶん正解です」
  - 再現させてレポートしようと思いつつまだできてない

## 飛び石で新Data Dictionaryの話でも

- おれ「mysqldumpが何故かダメだから5.7から8.0にインプ  
レースアップグレード！」
- ハッテンゼロ「Failed to update tables dictionary  
object. Error in Creating DD entry for d1.t1」
- おれ「えっ」
- おれ「1週間かけて調べた結果、変な（たぶんcp932でsjis  
じゃない）文字がテーブルコメントに含まれていたのがイク  
ナイ」
- ハッテンゼロ「たぶん正解です」
  - 再現させてレポートしようと思いつつまだできてない

## ふたたびパーティショニングの移行はたいへん(3)

- おれ「テーブルコメントの件も解決したので改めて検証進めるぞ」
- ハッテンゼロ「[ERROR] [MY-010520] [Server] Invalid (old?) table or database name '...' トバァー」
- おれ「えっ、ibdファイルもあるしなんならSELECTもできるんだけどエラーログが止まらない」
- Umesh「Thank you for the report.」
- おれ「やっぱりエラーログの方が間違ってるやがった」
  - [MySQL Bugs: #94519: Wrong messages in error¥-log when using partition with lower¥-case¥-table¥-names=1](#)
- おれ「絶対このへんなんか埋まってるし、パーティショニングは外しちまおう…」

## mysqld-auto.cnfと戦うDBA(1)

- おれ「SET GLOBAL collation\_server = 255;」
- ハッテンゼロ「Query OK, 0 rows affected (0.00 sec)」
- おれ「SET PERSIST\_ONLY collation\_server = 255;」
- ハッテンゼロ「Query OK, 0 rows affected (0.01 sec)」
- おれ「RESTART;」
- ハッテンゼロ「[ERROR] [MY-011268] [Server]  
Configuring persisted options failed: "Unknown  
collation: '255'"」
- MySQL Bugs「Not a bug」
  - [MySQL Bugs: #94573: SET PERSIST\\_ONLY can set incorrect value to innodb ft aux table](#)

## mysqld-auto.cnfと戦うDBA(2)

- おれ「ねえねえ、 SET PERSIST\_ONLY していい？」
- ふかまち「それダメだってブログに書いてありましたよ」
- おれ「矧…じゃあ SET PERSIST ならいい？」
- ふかまち「いいですけど」
- おれ「 SET PERSIST max\_connections= 300 …これたぶん、次に誰か “my.cnfでmax\_connections = 500にしたのに 300のままだ！” とかなりそうだよね」
- ふかまち「なりそうですね。でも performance\_schema.variables\_info でどこで設定されてるかは設定できるので」
- おれ「遅くなったな」

## mysqld-auto.cnfと戦うDBA(3)

- おれ「しかしやっぱ混乱するから SET PERSIST は封印しよう。mysqld-auto.cnf からも消しとく。ヴィムウ マイエス キューエルディー オート コン符」
  - ハッテンゼロ「 { "Version" : 1 , "mysql\_server" : { "collation\_server" : { "Value" : "255" , "Metadata" : { "Timestamp" : 1558002088561780 , "User" : "root" , "Host" : "localhost" } } } } 」
  - おれ「dd :wq mysqld再起動つと…しない」
  - おれ「おい mysqld-auto.cnf がおかしいってことくらいログに吐けよ」
- 
- ※ ファイルごと消せば大丈夫です
  - そのうちバグレポします

ハッテンゼロ  
楽しいです  
ね！



感想



## 感想

- yoku0825の独断と偏見によるものであり、所属する組織、所属しない組織またはNULLの意見を一切代表する訳がありません
- ド新規でutf8mb4統一、レプリケーションは8.0同士、でならそこそこイケる
- なんだかんだ言ってますけど新機能はそれなりに美味しくいただけます
- 5.7のdefault\_password\_lifetimeみたいな本当に「致命傷!」「罨!!」「トラウマになる!!!」みたいなのは  
**ない**
  - 「今のところ、観測範囲には」

## 感想

- むかーし（感覚だと5.7以前はつらい）に作られたバージョンアップにはなかなかクンフーが要ると思う
  - 「まず、立ち上がらない」みたいなのをトラブルシュートしたのは初めてな気がする
    - ごめん5.7でもあったわ
  - Data Dictionaryには「新機能は2バージョンくらい使わずに置いておいてから」とかいうの一切通用しない
    - 強くイキ



## ここ も 変だよMySQL

- それでも熱狂的なファンがいたりいなかったりする
  - データベースは枯れててナンボだとは思うんだけど、つい、ね…
- 持論「機能は待ってるだけじゃ勝手には枯れない。地雷原に除草剤を散布しながら歩いていく人間が必要」
  - MySQL-Fabricいかなぁー



# だけど涙が出ちゃう。だってDBAだもん

- あなたに合う地雷友達が きっと見つかる
  - [日本MySQLユーザ会](#)
  - [MySQL Casual](#)



あなたと  
マイエスキューエル  
今すぐ  
アップグレー  
ド



**Oracle  
ACE(地雷)か  
らは以上です**



**Any Questions  
and/or  
Suggestions?**



# MySQLとActive Recordの話

Ryuta Kamizono (@kamipo)


講演資料は下記のリンクからご覧いただけます。

<https://kamipo.github.io/talks/20190517-codetokyo19/#/title>



# Oracle ACEが語る！ ここがヘンだよMySQL & ここがスゴイよMySQL

2019/05/17 Oracle Code 2019 Tokyo  
Satoshi Mitani



データベースの  
一番、重要な要素は何だろう？



私の考え：

Durability / 耐久性

消えないこと・壊れないこと



ここがヘンだったよ  
&  
ここがスゴくなったよ



# 1. ヘンだった データファイル

# 壊れるデータファイル (ver <= 5.1)

- MyISAM
  - バージョン <= 5.1 までのデフォルト
  - crash-safe ではないストレージエンジン
  - 障害時にテーブルが壊れることがある
  - repair に結構時間がかかる . . .

# 壊れるデータファイル (ver <= 5.1)

- 5.5
  - InnoDBがデフォルトに
- 5.6
  - InnoDB Full Text Searchがサポート
  - MyISAM/InnoDBの差がほぼなくなる

## 2. ヘンだった スレーブ





# クラッシュセーフでないスレーブ (ver <= 5.5)

- relay-log.info ファイル
  - スレーブ(SQLスレッド)がどのポジションまで進んだかを記録
- ただのフラットファイル
  - ストレージエンジンとの一貫性が保てない

```
$ cat relay-log.info
/var/lib/mysql/mysqld-relay-bin.005719
145630
mysqld-bin.001393
889185442
66
```

- 障害→復旧時、2重適用があり得た



# クラッシュセーフでないスレーブ (ver <= 5.5)

- 5.6 のクラッシュ セーフスレーブで解決
  - リポジトリをフラットファイルからInnoDBに変更

```
relay_log_info_repository = TABLE  
relay_log_recovery = ON
```

- 8.0
  - クラッシュセーフスレーブがデフォルトに



### 3. ヘンだった レプリケーション

# 非同期レプリケーション (ver <= 5.6)

- 非同期しか選択肢なかった
- スレーブのマスター昇格時にデータロストのリスクあり
  - 当時、みなさん、どうしてたんだろ？



# 非同期レプリケーション (ver <= 5.6)

- 5.6
  - 準同期レプリケーション
  - (ざっくり) マスターを更新し、スレーブに伝えて、クライアントにACK
  - ごく最小限のロス
    - ACKがまだ返っていないがマスターに適用済みのデータがロスする
- 5.7
  - ロスレス準同期レプリケーション
  - (ざっくり) スレーブに伝えて、マスターを更新し、クライアントにACK
  - 完全解決



## 4. ヘンだった DDL



# クラッシュセーフでないDDL (ver <= 5.7)

- DDL 実行中に落ちると不整合に

```
mysql> SHOW TABLES;
+-----+
| Tables_in_d1 |
+-----+
| tbl          |

mysql> DROP TABLE tbl;
ERROR 1051 (42S02): Unknown table 'tbl'
```



# クラッシュセーフでないDDL (ver <= 5.7)

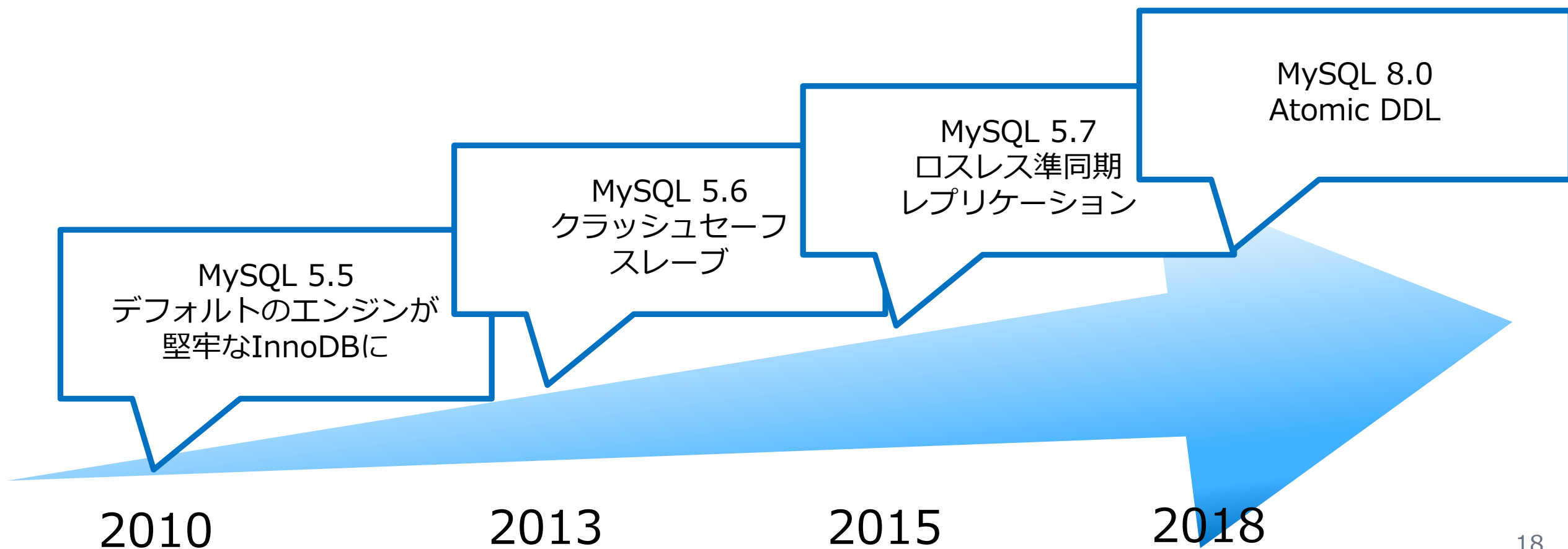
- MySQL 8.0のAtomic DDLで解決





# ここがヘンだったよ & ここがスゴくなったよ

- MySQLのDurabilityはバージョンアップを重ねるごとにスゴくなってきた





# MySQLと令和

とみたまさひろ

2019-05-17

Oracle Code Tokyo 2019

# 自己紹介

- とみたまさひろ
- @tmtms
- 日本MySQLユーザ会
- 富士通クラウドテクノロジーズ
- Oracle ACE Associate
- <http://tmtm.github.io/mysql-params/>
- 得意技: Ruby, 文字化け



祝令和元年 🎉

**今日は「令和」の話をします**

# その1

## 「令和」と言えば…

似てるけど違う文字  
「令」と「令」

# 同じに見えるけど別の文字

- 「令」 U+4EE4 CJK UNIFIED IDEOGRAPH
- 「令」 U+F9A8 CJK COMPATIBILITY IDEOGRAPH

困る！

# MySQLで

```
mysql> set @a='令和', @b='令和';
mysql> select @a, @b, hex(@a), hex(@b);
```

@a	@b	hex(@a)	hex(@b)
令和	令和	E4BBA4E5928C	EFA6A8E5928C

```
mysql> select @a=@b;
```

@a=@b
1

← 一致



# その2

## 「令和」と言えば…

# 異体字

「令」と「令」

# 違う字形だけど同じ文字

異体字セレクタ

- 「令」 U+4EE4
- 「令」 U+4EE4 **U+E0101**
- 「令」 U+4EE4 **U+E0102**

困る！

# MySQLで

```
mysql> set @a='令和', @b='令和', @c='令和';
mysql> select hex(@a), hex(@b), hex(@c)\G
***** 1. row *****
hex(@a): E4BBA4E5928C
hex(@b): E4BBA4F3A08481E5928C
hex(@c): E4BBA4F3A08482E5928C
mysql> select @a=@b, @b=@c;
+-----+-----+
| @a=@b | @b=@c |
+-----+-----+
|      1 |      1 | ← 一致
+-----+-----+
```

※都合により同じ字体に見えてます



# その3

## 「令和」と言えば…

元号

**元号と言えは…**

# 合字

- 明治: 𠄎 U+337E
- 大正: 𠄏 U+337D
- 昭和: 𠄐 U+337C
- 平成: 𠄑 U+337B
- 令和: 令和 U+32FF

# MySQLで

```
mysql> select '明治'='𐄂', '大正'='𐄃', '昭和'='𐄄',  
-> '平成'='𐄅', '令和'='𐄆'\G  
***** 1. row *****  
'明治'='𐄂': 1    ← 一致  
'大正'='𐄃': 1    ← 一致  
'昭和'='𐄄': 1    ← 一致  
'平成'='𐄅': 1    ← 一致  
'令和'='𐄆': 0    ← 不一致
```



# 本日のテーマ

「ここがヘンだよMySQL」

**なにが起きてるのか？**

# Unicodeの照合順序

Unicode Collation Algorithm (UCA)

<https://unicode.org/reports/tr10/tr10-34.html>

Default Unicode Collation Element Table (DUCET)

<https://www.unicode.org/Public/UCA/9.0.0/allkeys.txt>

文字毎にWeightという値が定義されている  
Weightが等しいなら等しい文字

その1

似てるけど違う文字

「令」と「令」

- 「令」 U+4EE4 CJK UNIFIED IDEOGRAPH  
DUCETには無いけど計算で求まる

```
[.FB40.0020.0002][.(CP | 0x8000).0000.0000]  
→ [.FB40.0020.0002][.CEE4.0000.0000]
```

- 「令」 U+F9A8 CJK COMPATIBILITY IDEOGRAPH  
DUCETにある

```
F9A8 ; [.FB40.0020.0002][.CEE4.0000.0000]
```

Weightが一致するから等しい

# その2

## 異体字

### 「令」と「令」

## 異体字セレクタ

- 「令」 U+4EE4
- 「令」 U+4EE4 **U+E0101**
- 「令」 U+4EE4 **U+E0102**

異体字セレクタはDUCETにある

```
E0101 ; [.0000.0000.0000] # VARIATION SELECTOR-18  
E0102 ; [.0000.0000.0000] # VARIATION SELECTOR-19
```

UCAではすべてゼロの文字は無視

# その3

## 合字

明治 / 大正 / 昭和 / 平成 / 令和

# 平成=𡇗

- 平(U+5E73)成(U+6210): 𡇗(U+337B)

平成	[.FB40.0020.0002]	[.DE73.0000.0000]	[.FB40.0020.0002]	[.E210.0000.0000]
𡇗	[.FB40.0020.001C]	[.DE73.0000.0000]	[.FB40.0020.001C]	[.E210.0000.0000]

ちょっと違う… 🤔

# utf8mb4\_0900\_ai\_ci

MySQLのデフォルトのCollation

要素	意味
utf8mb4	4バイトUTF-8
0900	Unicode 9.0.0
ai	アクセントの違いを無視
ci	大文字小文字の違いを無視

ciの場合はWeightの3番目を無視

ciの場合はWeightの3番目を無視

平成	[.FB40.0020.0002]	[.DE73.0000.0000]	[.FB40.0020.0002]	[.E210.0000.0000]
弐	[.FB40.0020.001C]	[.DE73.0000.0000]	[.FB40.0020.001C]	[.E210.0000.0000]

3番目を無視すると

平成	[.FB40.0020.]	[.DE73.0000.]	[.FB40.0020.]	[.E210.0000.]
弐	[.FB40.0020.]	[.DE73.0000.]	[.FB40.0020.]	[.E210.0000.]

一致

令和≠令和

「令和」がDUCETに無い！

「令和」はUnicode 9.0.0 に無い！

「令和」はUnicode 12.1.0 で追加

<http://unicode.org/versions/Unicode12.1.0/>

12.1 は「令和」のためだけに 5/7 にリリース

*Unicode 12.1 adds exactly one character, for a total of 137,929 characters.*

*The new character added to Version 12.1 is:*

*U+32FF SQUARE ERA NAME REIWA*

*Version 12.1 adds that single character to enable software to be rapidly updated to support the new Japanese era name in calendrical systems and date formatting. The new Japanese era name was officially announced on April 1, 2019, and is effective as of May 1, 2019.*

**MySQL独自の変な挙動じゃなくて  
Unicodeの規則だった！**

**Unicode規則にちゃんと従ってる  
MySQLえらい！**

# 本日のテーマ

**「ここがスゴイよMySQL」**