MySQL日本語処理
現状と今後

松信 嘉範 (MATSUNOBU Yoshinori)
MySQL株式会社
シニアコンサルタント
ymatsunobu@mysql.com

今回の内容

- MySQL6.0の4バイトUTF-8文字対応
- 日本語文字列のソート
- Unicode、シフトJIS、日本語EUCの長所・短所の整理
- 日本語のテーブル名

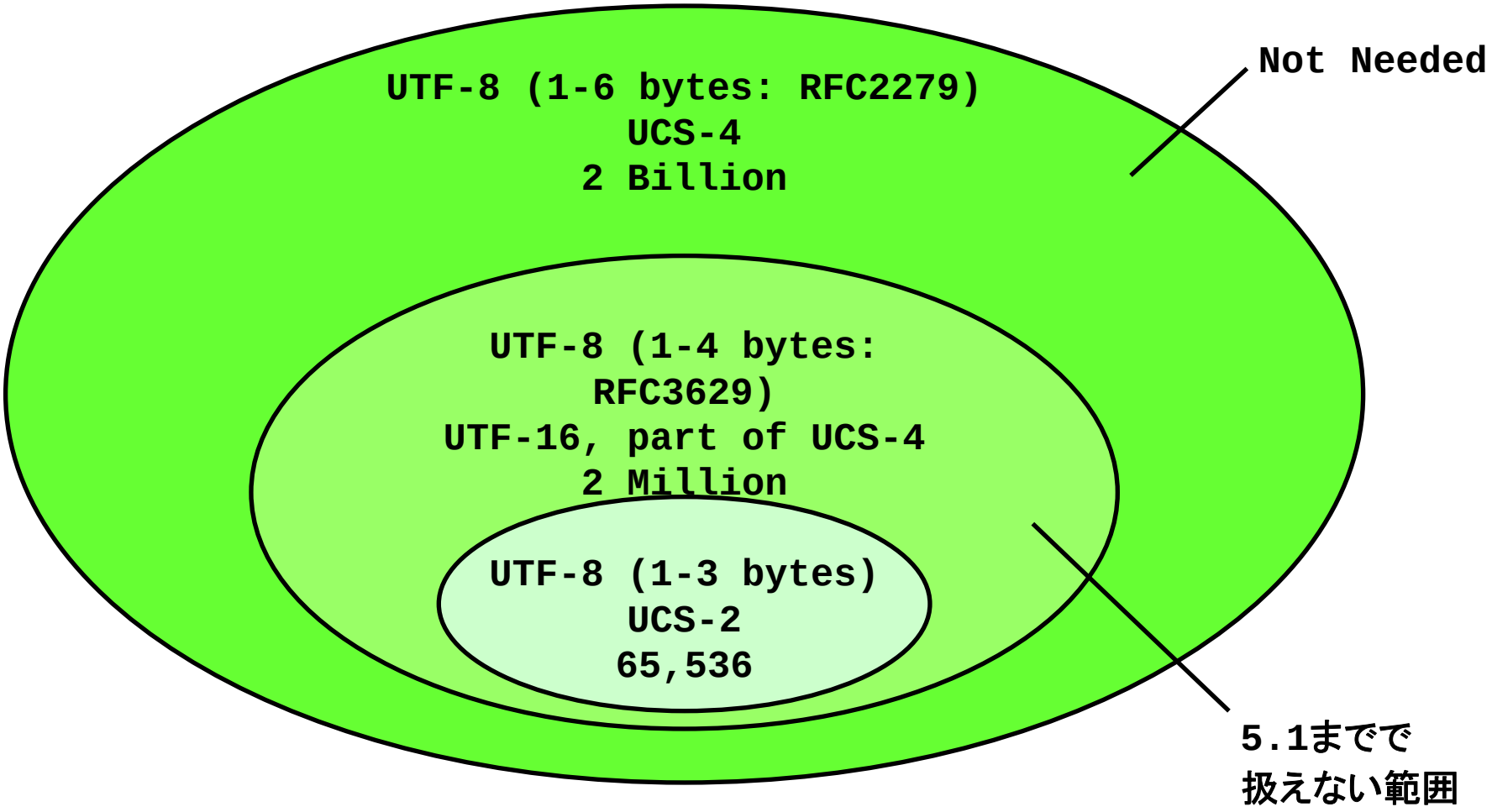
4バイトUTF-8文字とは

- UTF-8は1バイト～4バイトで表現される
  - 1バイト: ASCII
  - 2バイト: ラテン/ギリシャ/キリル/アラビア文字など
  - 3バイト: 日本語文字(全角/半角)、ハングル文字など
  - 4バイト: JIS第3・第4水準漢字の一部など
    (JIS X 0213の一部が該当)

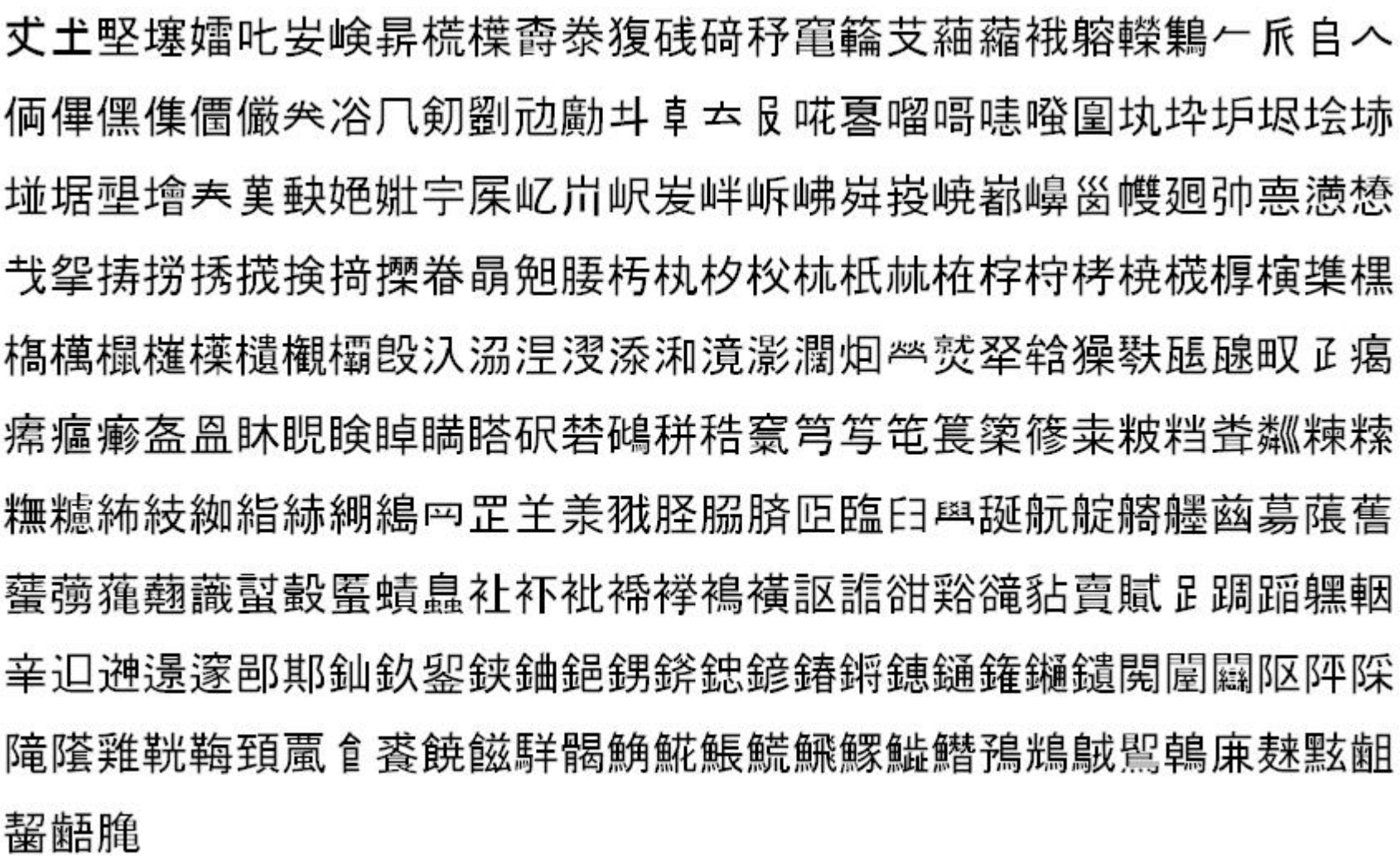
  ※5、6バイトも用意されているが使われる予定は無い。
  1～4バイトまでで約203万文字を扱える

- 4バイトのUTF-8文字を扱うには、「サロゲート領域」を考慮する必要がある
- MySQL5.1までは、サロゲート領域のサポートが無く、3バイトUTF-8までしか扱えない

Unicodeのカバー範囲



JIS X 0213の4バイトUTF-8文字一覧



5.1までの動作

```
mysql> CREATE TABLE t1 (c1 VARCHAR(30)) CHARSET utf8;
Query OK, 0 rows affected (0.17 sec)

# 0xF0909080 is a 4byte UTF-8 character
mysql> INSERT INTO t1 VALUES(0xF0909080);
Query OK, 1 row affected, 1 warning (0.08 sec)

# "abc" + 4byte UTF-8 + "def"
mysql> INSERT INTO t1 VALUES(0x616263F0909080646566);
Query OK, 1 row affected, 1 warning (0.03 sec)

# "def" is truncated
mysql> SELECT HEX(c1) FROM t1;
+---------+
| HEX(c1) |
+---------+
|         |
| 616263  |
+---------+
```

・格納自体ができない
・4バイトUTF-8文字だけでなく、その先が全部削られて格納される。
・sql_mode='STRICT_ALL_TABLES'の場合はWarningではなくErrorになる

6.0の動作

```
mysql> CREATE TABLE t1 (c1 VARCHAR(30)) CHARSET utf8;
Query OK, 0 rows affected (0.17 sec)

# 0xF0909080 is a 4byte UTF-8 character
mysql> INSERT INTO t1 VALUES(0xF0909080);
Query OK, 1 row affected (0.03 sec)

# "abc" + 4byte UTF-8 + "def"
mysql> INSERT INTO t1 VALUES(0x616263F0909080646566);
Query OK, 1 row affected (0.00 sec)

mysql> SELECT HEX(c1) FROM t1;
+----------------------+
| HEX(c1)              |
+----------------------+
| F0909080             |
| 616263F0909080646566 |
+----------------------+
2 rows in set (0.09 sec)
```

・そのまま格納できる
・もちろん取り出して読むこともできる

対応関係

	5.1まで	6.0以降
サロゲート領域 非対応	utf8 ucs2	utf8mb3 ucs2
サロゲート領域にも 対応	なし	utf8 utf16 utf32

- 5.1のutf8は、6.0のutfmb3に対応
- 6.0では、utf8(4バイト対応)に加え、utf16とutf32もサポート
- MySQL5.1以下でutf8作ったテーブルをそのまま(バイナリコピーで)6.0に持って行った場合、utf8mb3になる
- クライアントエンコーディングをutf8に設定すると(CharacterEncodingやSET NAMES等)
  - 5.1ではutf8 (3バイト)
  - 6.0ではutf8 (4バイト)

補足

- 4バイトUTF-8は、MEMORYストレージエンジンで副作用
  - VARCHAR (100) CHARSET utf8は400バイトを使う
  - GROUP BYなどで作られるIn Memoryテンポラリテーブルも同様
  - 可変長MEMORYストレージエンジンを準備中(eBay Patch)
- cp932やeucjpmsであれば、とりあえずJIS X 0213文字のINSERT/SELECTはできる
- シフトJIS(cp932)や日本語EUC(eucjpms)との変換は今のところ(MySQL内では)できない
  - PostgreSQLのSHIFT_JIS_2004のようなエンコーディングを作れば可能
  - 作ったところで、そもそもカバー範囲が違う(Unicodeの方が広い)ので、中国語とかは当然変換できない
  - 必要があれば、MySQLの外で変換して入れれば良い
  - 文字コード変換はCPUリソースを使うので、DBサーバよりもスケールアウトが簡単なところ(APサーバ等)で行なった方が良い
  - このニーズはどこまであるのか?

日本語文字列のソート

- 現状は、コードポイントの順にソート
  - シフトJIS、日本語EUC、Unicodeでソート順が微妙に違う
  - ひらがな、カタカナはだいたいうまくソートしてくれる
  - 第1水準漢字はシフトJIS、日本語EUCともになんとなくソートしてくれる
- ソートの標準仕様
  - JIS X 4061
  - 辞書(電話帳)検索にも対応
- MySQLは、日本語文字列のソートについて特に何か特別なことをしているわけではない
- 要望が強ければ、対処することも可能

ソート例1 (ひらがな･カタカナ)

電話帳順	MySQL cp932	MySQL eucjpms	MySQL utf8
ウィザード	ウィザード	ウィザード	ウィザード
グリフォン	クレリック	クレリック	クレリック
クレリック	グリフォン	グリフォン	グリフォン
ゴーゴン	ゴーゴン	ゴーゴン	ゴブリン
ゴブリン	ゴブリン	ゴブリン	ゴーゴン
ヴァルキリー	ホワイトドラゴン	ホワイトドラゴン	ホワイトドラゴン
ホワイトドラゴン	ヴァルキリー	ヴァルキリー	ヴァルキリー

- どれもそれなりに自然な形で並ぶ (微妙に正確性に欠ける)

- コードポイントで比較すると、濁音は清音よりも必ず後に来る
  - 電話帳では、次の音と比較した結果によっては前に来ることもある

- Unicodeのコードポイントでは、 ン < ヴ < ー の並び順

ソート例2 漢字

電話帳順	MySQL cp932	MySQL eucjpms	MySQL utf8
會川 (あいかわ: JIS第2水準)	阿部	阿部	加藤
阿部	安藤	安藤	宇野
安藤	宇野	宇野	安藤
宇野	加藤	加藤	會川
江藤	江藤	江藤	江藤
加藤	會川	會川	阿部

- シフトJISと日本語EUCは、
  - 第1水準漢字のコードポイント < 第2水準漢字のコードポイント
- 最も多用される第1水準漢字のコードポイントは「音読み順」なので、
  シフトJISと日本語EUCはそれなりに自然な形に並ぶ (正確なわけではない)
  ※第2水準漢字は部首順
- Unicodeは統制が無いに等しい

ソートの標準仕様 JIS X 4061

- **JIS X 4061**:1996（日本語文字列照合順番）
- 辞書順ソートを実現する
- エンコーディングに依存しない

- 実装上の課題
  - コードポイントではなく、標準仕様のソート順を数値化
  - その文字だけでなく、次の文字を考慮する必要がある
    - かき < がけ < かに
    - 重み付け

Unicodeで本当に幸せになれるのか?

- UTF-8
  - 日本語の文字は1文字あたり3バイト (一部4バイト)
  - シフトJISや日本語EUCに比べて大きく不利
- UTF-16
  - 日本語文字は2バイト(一部4バイト)だが、ASCII文字も2バイト
  - 現実的にはASCII文字を多用するので、シフトJISや日本語EUCに比べて大きく不利
- シフトJIS
  - 0x5C問題さえ無ければ…
- 日本語EUC
  - エンコーディングの統制が取れていれば…

3バイト文字と2バイト文字の違いは大きい

- Wikipedia日本語版で検証
  - UTF-8で作られている
  - 全コンテンツをXML形式でダウンロードできる
  - XMLを変換することで、MySQLにダンプする元となるテキストファイルを生成可能
- iconvでシフトJISや日本語EUCに変換させて、それぞれのサイズを比較
  - 一部変換できない文字があるが、サイズを比較する上では無視できる範囲

UTF-8 (オリジナル)	CP932	EUCJP-MS
2700.0MB	2013.6MB	2016.4MB

2700MB -> 2013.6MB (約25%の削減)
25%の省スペース化は、一般的なチューニングの観点からは「大きな成果」

日本語EUCのエンコーディング

- EUC-JP
  - MySQLのujis、JavaのEUC-JP
  - ベンダー定義文字やユーザー定義外字には非対応
- eucjpms
  - MySQLのeucjpms、PostgreSQLのEUC、PHPのeucJP-win
  - NEC/IBM文字、JIS X 0212(補助漢字)、ユーザー定義外字に対応
  - IBM拡張文字は3バイト
- cp51932 (Internet ExplorerのEUC-JP)
  - JIS X 0212、ユーザー定義外字には非対応
  - IBM拡張文字は2バイト
- FirefoxのEUC-JP
  - cp51932に、JIS X 0212を追加
- EUC_JP_Solaris
  - Javaのベンダー定義文字対応のEUC
  - ユーザー定義外字には非対応

MySQLをEUCにしたい場合によく推奨される設定

- クライアントサイド(ブラウザ)ではWindows-31J(またはUTF-8)
- Webサーバ/AP層でWindows-31J(またはUTF-8)からeucjpmsに変換
  - PHPならeucJP-win
  - PerlならEncode::EUCJPMS
  - Javaではeucjpmsに完全対応したエンコーディングは無い
- MySQLには変換なしでそのまま格納

- UTF-8ではなくEUCを使う理由
  - 省スペース化によるパフォーマンス向上とディスク領域削減
  - 多国籍言語対応が必要ならUTF-8しかない
- シフトJISではなくEUCを使う理由
  - 0x5C問題が嫌だから
- MySQL側で文字コード変換をしない理由
  - CPUリソースを過度に使わせないため
  - AP層の方がスケールアウトしやすい

日本語のテーブル名

- 5.0までは、日本語でテーブルを作ることは事実上できなかった
  - 「テーブル名.frm」というテーブル定義ファイルができる
  - ここに日本語名が入る
  - エンコーディング環境の異なるOS間では機能しない
  - クライアントエンコーディングが違ってもだめ
  - 「港(0x60, `)」など、特別な記号と一致するコードポイントを含むとだめ
- 5.1からは、日本語を含むマルチバイト文字に対応
  - 「@ + Unicodeのコードポイント」で表現

例

```
$ ls
@4ed5@5165@5148.frm
@53d7@6ce8.frm
@53d7@6ce8@660e@7d30.frm
@5546@54c1.frm
@5546@54c1@533a@5206.frm
@5f97@610f@5148.frm
@793e@54e1.frm
@904b@9001@4f1a@793e.frm
@90fd@9053@5e9c@770c.frm
```

仕入先、受注、受注明細、商品、商品区分、得意先、
社員、運送会社、都道府県

※Microsoft AccessのサンプルデータベースNorthwindからの移行

ありがとうございました

- ご質問/ご意見等は
  - ymatsunobu@mysql.com