# mysql使用utf8mb4经验吐血总结

发表于 2016-10-23 | 更新于 2016-10-23 | 分类于 MySQL | 阅读次数 23029



## 1. utf8 与 utf8mb4 异同

先看官方手册 https://devmysql.com/doc/refman/5.6/en/charset-unicode-utf8mb4.html 的说明：

```
The character set named utf8 uses a maximum of three bytes per character and contains only BMP characters. The utf8mb4 char

- For a BMP character, utf8 and utf8mb4 have identical storage characteristics: same code values, same encoding, same length
- For a supplementary character, utf8 cannot store the character at all, whereas utf8mb4 requires four bytes to store it. S
```

MySQL在 5.5.3 之后增加了 `utf8mb4` 字符编码，mb4即 most bytes 4。简单说 utf8mb4 是 utf8 的超集并完全兼容utf8，能够用四个字节存储更多的字符。

但提开数据库，标准的 UTF-8 字节集编码是可以用 1-4 个字节去编码21位字符，这几乎包含了最世界上所有能看见的语言了。然而在MySQL里实现的utf8最长使用3个字节，也就是只支持到了 Unicode 中的 基本多文本平面（U+0000至U+FFFF），包含了控制符、拉丁文、中、日、韩等绝大多数国际字符，但并不是所有，最常见的就算现在手机端常用的表情字符 emoji和一些不常用的汉字，如“墨”，这些需要四个字节才能编码出来。

注：QQ里面的内置的表情不算，它是通过特殊映射到的一个gif图片。一般输入法自带的就是。

也就是当你的数据库里要求能够存入这些表情或者汉字时，可以把字段定义为 utf8mb4，同时要注意连接字符集也要设置为utf8mb4，否则在严格模式下会出现 `Incorrect string value: '\xF0\x9F\x98\x82\xE5\xA5\x...' for column 'name'` 这样的错误，非严格模式下此后的数据会被截断。

提示：另外一种能够存储emoji的方式是，不关心数据库表字符集，只要连接字符集使用 latin1，但相信我，你绝对不想这个干，一是这种字符集简用管理极不规范，二是存储空间被放大（读者可以想下为什么）。

## 2. utf8mb4_unicode_ci 与 utf8mb4_general_ci 如何选择

字符除了需要存储，还需要排序或比较大小，涉及到与编码字符集对应的 排序字符集（collation），utf8mb4对应的排序字符集常用的有 `utf8mb4_unicode_ci`、`utf8mb4_general_ci`，到底采用哪个在 stackoverflow 上有个讨论，What's the difference between utf8_general_ci and utf8_unicode_ci

主要从排序准确性和性能两方面看：

- 准确性

  `utf8mb4_unicode_ci` 是基于标准的Unicode来排序和比较，能够在各种语言之间精确排序

  `utf8mb4_general_ci` 没有实现Unicode排序规则，在遇到某些特殊语言或字符是，排序结果可能不是所期望的。

  **但是在绝大多数情况下，这种特殊字符的顺序一定要那么精确吗**，比如Unicode把ß、æ当成ss和oe来看；而general会把它们当成s、e，再如 ÀÁÂÃÄÅ 各自都与 A 相等。

- 性能

  `utf8mb4_general_ci` 在比较和排序的时候更快

  `utf8mb4_unicode_ci` 在特殊情况下，Unicode排序规则为了能够处理特殊字符的情况，实现了略微复杂的排序算法。

  **但是在绝大多数情况下，不会发生此类复杂比较**，general和它上比Unicode可能快些，但相比现在的CPU来说，它远远不足以成为考虑性能的因素，索引除及、SQL设计才是。 我个人推荐是 `utf8mb4_unicode_ci`，将来 8.0 里也极有可能使用成为默认的规则，相比选择哪一种collation，使用者应该更关心字符集与排序规则在本里要统一就好。

这也从另一个角度告诉我们，不要可能产生乱码的字段作为主键或唯一索引，我遇到过一例，以 url 来作为唯一索引，但是总记录的有可能是乱码，导致后来想把它们修复就特别麻烦。

## 3. 怎么从utf8转换为utf8mb4

### 3.1 “伪”转换

如果你的表definer又和连接字符集都是utf8，那么直接在你的表上执行

```
ALTER TABLE tbl_name CONVERT TO CHARACTER SET utf8mb4;
```

则能够表表上所有的列的character类型变成 utf8mb4，表定义的默认字符集也会修改。连接的时候需要使用 `set names utf8mb4` 便可以插入四字节字符。（如果依然使用 utf8 连接，只要不出现四字节字符则完全没问题）。

上面的 convert 有两个问题，一是它不能ONLINE，也就是执行之后全表禁止修改，有关这方面的讨论见 mysql 5.6 原生Online DDL解析；二是，它可能会自动该表字段类型定义，如 VARCHAR 被转成 MEDIUMTEXT，可以通过 MODIFY 指定类型为原类型。

另外 `ALTER TABLE tbl_name DEFAULT CHARACTER SET utf8mb4` 这样的语句就不要随便执行了，特别是当表原本不是utf8时，除非表是空的或者你确认表里只有拉丁字符，否则正常和乱的就范在一起了。

**最重要的是**，你连接时使用的是latin1字符集写入了历史数据，表定义是latin1 或utf8，不要期望通过 `ALTER ... CONVERT ...` 能够让你这到用utf8读取历史中文数据的目的，说明用，老老实实做逻辑dump，所以既才叫它“伪”转换

### 3.2 character-set-server

一旦你决定使用utf8mb4，强烈建议你要修改服务端 `character-set-server=utf8mb4`，不同的语言对它的处理方法不一样，c++，php, python可以设置character-set，但java驱动依赖于 character-set-server 选项，后面有介绍。

同时还要谨慎一些特殊选项，如 遇到腾讯云CDB连接字符集设置一个坑。个人不建议设置全局 `init_connect`。

## 4. key 768 long 错误

字符集从utf8转到utf8mb4之后，最容易引起的就是索引键超长的问题。

对于表行格式是 `COMPACT` 或 `REDUNDANT`，InnoDB有单个索引最大字节数 768 的限制，而字段定义的是最存储的字符数，比如 `VARCHAR(200)` 代表能够存200个汉字，索引定义是字符集类型最大长度算的，即 utf8 maxbytes=3, utf8mb4 maxbytes=4，算下来utf8和utf8mb4两种情况的索引长度分别为600 bytes和800bytes，后者超过了768，导致出错：`Error 1071: Specified key was too long; max key length is 767 bytes`。

`COMPRESSED` 和 `DYNAMIC` 格式不受限制，但也依然不建议索引太长，尤表费空间和cpu搜索资源。

如果已有定义超过这个长度的，可加上前缀索引，如果暂不能加上前缀索引（像唯一索引），可把该字段的字符集改回utf8或latin1。

但是，（**敲黑板啦，很重要**），要防止出现 `Illegal mix of collations (utf8_general_ci,IMPLICIT) and (utf8mb4_general_ci,COERCIBLE) for operation '='` 错误：连接字符集使用utf8mb4，但 SELECT/UPDATE where条件有utf8类型的列，且条件右边存在不属于utf8字符，就会触发该异常。表示踩过这个坑。

再多加一个友好提示：EXPLAIN 结果里面的 key_len 指的是索索索引长度，单位是bytes，而且是以字符集支持的单字节最大字节数算的，这也是为什么 INDEX_LENGTH 跟账对害的一个原因。

## 5. C/C++ 内存空间分配问题

这是我们这边的开发遇到的一个棘手的问题，C或c++连接MySQL使用的是linux系统上的 libmysqlclient 动态库，程序获取到数据之后根据自定义的一个网络协议，按照mysql字段定义的固定字节数来传输数据，从utf8转utf8mb4 之后，c++里面针对character单字符内存空间分配，从3个增加到4个，引起异常。

这个问题其实是想说明，使用utf8mb4之后，官方建议尽量用 varchar 代替 char，这样可以减少固定存储空间浪费（关于char与varchar的选择，可参考 这里），但开发设计表时 varchar 的大小不能随意加大，它虽然是变长的，但客户端在定义变量来获取数据时，是以定义的为准，而非实际长度。按需分配，避免程序使用过多的内存。

## 6. java驱动使用

Java里面里面所实现的UTF-8编码就是支持4字节的，所以不需要配置 `mb4` 这样的字段，但如果从MySQL读写emoji，MySQL驱动版本要在 5.1.13 及以上版本，数据库连接依然是 `characterEncoding=UTF-8`。

但还没完，遇到一个大坑。官方手册 里还有这么一段话：

```
Connector/J did not support utf8mb4 for servers 5.5.2 and newer.

Connector/J now auto-detects servers configured with character_set_server=utf8mb4 or treats the Java encoding utf-8 passed
   using characterEncoding=... as utf8mb4 (A the SET NAMES= calls it makes when establishing the connection. (Bug #54179)
```

意思是，java驱动会自动检测服务端 `character_set_server` 的配置，如果为utf8mb4，驱动在建立连接的时候设置 `SET NAMES utf8mb4`，然而其他语言没有依赖于这样的特性。

## 7. 主从复制报错

这个问题没有遇到，只是看官方文档有提到，曾经也看到过类似的技术文章。

大概就是从库的版本比主库的版本低，导致有些字符集不支持；或者人工修改了从库上的表或字段的字符集定义，都有可能引起异常。

## 8. join 查询问题

这个问题是之前在美承务老师公众号看到的一篇文章MySQL表字段字符集不同导致的索引失效问题，自己也验证了一下，的确会有问题：

```
CREATE TABLE t1 (
  f_id varchar(20) NOT NULL,
  f_action char(25) NOT NULL DEFAULT '' COMMENT '',
  PRIMARY KEY (`f_id`),
) ENGINE=InnoDB DEFAULT CHARSET=utf8 ROW_FORMAT=DYNAMIC;

CREATE TABLE t1_copy_mb4 (
  f_id varchar(20) CHARACTER SET utf8mb4 NOT NULL,
  f_action char(25) NOT NULL DEFAULT '' COMMENT '',
  PRIMARY KEY (`f_id`),
) ENGINE=InnoDB DEFAULT CHARSET=utf8 ROW_FORMAT=DYNAMIC;

1.
EXPLAIN extended select * from t1 INNER JOIN t1_copy_mb4 t2 on t1.f_id=t2.f_id where t1.f_id='4210356';

2.
EXPLAIN extended select * from t1 INNER JOIN t1_copy_mb4 t2 on t1.f_id=t2.f_id where t2.f_id='4210356';
```

对应上面1,2 的截图：

其中 2的warnings 有 convert：

- (convert(t1.f_id using utf8mb4) = '4210356')

官网能找到这一点解释的还是开头那个地址：

```
Similarly, the following comparison in the WHERE clause works according to the collation of utf8mb4_col:

  SELECT * FROM utf8_tbl, utf8mb4_tbl
  WHERE utf8_tbl.utf8_col = utf8mb4_tbl.utf8mb4_col;
```

只是索引失效发生在utf8mb4列 在条件 后边。（关于MySQL的隐式类型转换，见 这里）。

## 9. 参考

- https://devmysql.com/doc/refman/8.0/en/charset-unicode-conversion.html
- http://forums.mysql.com/read.php?103,187048,188748#msg-188748
- Why are we using utf8mb4_general_ci and not utf8mb4_unicode_ci?
- How to support full Unicode in MySQL databases
- 10分钟学会理解和解决MySQL乱码问题

---

原文链接地址：http://seanlook.com/2016/10/23/mysql-utf8mb4/

---

`# mysql` `# 字符集`

‹ 遇到腾讯云CDB连接字符集设置一个坑

让mysqldump变成开发导出导入的魔法 ›