MySQL字符集调研

Questions

- Server处理字符的流程
- 所有字符集参数的作用

```
character_set_client
character_set_connection
character_set_database
character_set_filesystem
character_set_results
character_set_server
character_set_system
character_sets_dir
```

- set names做了什么
- set charset做了什么
- init_connect动态改变无效
- JDBC连接串characterEncoding的作用
- sqlmode traditional下插入异常
- 如何将一个表的字符集从utf8升级到utf8mb4

连接时的字符集

客户端连接到MySQL Server总共分四步 1. 客户端socket connect，等待服务器返回消息 2. 服务器accept connect，发送random string和服务器端的参数（服务器版本，字符集等）给客户端 3. 客户端利用random string加密用户名密码，并加入客户端的参数，如字符集等，发送给服务器 4. 服务器端验证用户名，密码，并根据客户端的其他参数进行THD属性设置。

可以看到这里涉及到两个字符集，服务器端的字符集，以及客户端传送的字符集。

服务器端默认字符集

```
#define MYSQL_DEFAULT_CHARSET_NAME "latin1"
#define MYSQL_DEFAULT_COLLATION_NAME "latin1_swedish_ci"
```

即MySQL默认的字符集是**latin1**。

如何改变这个值呢？上面是在预编译时就确定的值,cmake编译时可以通过设置下面的变量改变上面的默认值

```
DEFAULT_CHARSET The default server character set    latin1
DEFAULT_COLLATION   The default server collation    latin1_swedish_ci
```

如果配置文件设置了**character_set_server**，那么也会Server的改变默认字符集设置。

```
    if (!(default_charset_info=
          get_charset_by_csname(default_character_set_name,
                                MY_CS_PRIMARY, MYF(MY_WME))))
```

全局变量初始化

```
global_system_variables.collation_server=  default_charset_info;
global_system_variables.collation_database=  default_charset_info;
global_system_variables.collation_connection= default_charset_info;
global_system_variables.character_set_results= default_charset_info;
global_system_variables.character_set_client= default_charset_info;
global_system_variables.character_set_filesystem= character_set_filesystem;
```

mysql客户端连接时的字符集

可以通过指定参数default-character-set来指定字符集，如果不指定，则会使用系统的字符集，

```
➜  ~  locale
LANG=en_US.UTF-8
LANGUAGE=en_US:
LC_CTYPE="en_US.UTF-8"
LC_NUMERIC="en_US.UTF-8"
LC_TIME="en_US.UTF-8"
LC_COLLATE="en_US.UTF-8"
LC_MONETARY="en_US.UTF-8"
LC_MESSAGES="en_US.UTF-8"
LC_PAPER="en_US.UTF-8"
LC_NAME="en_US.UTF-8"
LC_ADDRESS="en_US.UTF-8"
LC_TELEPHONE="en_US.UTF-8"
LC_MEASUREMENT="en_US.UTF-8"
LC_IDENTIFICATION="en_US.UTF-8"
LC_ALL=en_US.UTF-8
```

会使用**LC_CTYPE**的字符集作为客户端默认的字符集。

当客户端连接到服务器时，会调用***thd_init_client_charset***进行设置客户端。

```
    thd->variables.character_set_results=
    thd->variables.collation_connection=
    thd->variables.character_set_client= cs;
```

这里的cs就是客户端指定的或这默认的cs，此时我们可以看下变量：

```
mysql> show variables like '%char%';
+--------------------------+----------------------------------------------
------------------------------------------+
| Variable_name            | Value
|
+--------------------------+----------------------------------------------
------------------------------------------+
| character_set_client     | utf8
|
| character_set_connection | utf8
|
| character_set_database   | latin1
|
| character_set_filesystem | binary
|
| character_set_results    | utf8
|
| character_set_server     | latin1
|
| UTF     | utf8
|
| character_sets_dir       | /home/louis/src/pxc-repo/Percona-XtraDB-
Cluster-5.6.20-68.0/debug/mysql/share/charsets/ |
+--------------------------+----------------------------------------------
------------------------------------------+
```

可以看到***character_set_client***，***character_set_connection***，***character_set_results***都改变成了utf8。

character_set_client用于在分词阶段选择编码，此时对于字符串常量的检查并不完整，只是当成字节流。
character_set_connection用于语法分析阶段的编码，会将字节流按照**character_set_client**的编码格式解析，然后转化成character_set_connection的编码格式。如果按照原始编码解析不了，则会转化成`?`，此时不会报异常。
当需要将字符串写入表中的时候，此时会讲字符串按照**character_set_connection**的编码格式解析，然后转化成对应字段设置的字符集的格式。***Item::save_str_value_in_field***，此时如果按照**character_set_connection**的编码格式解析解析不了，那么会直接报错。

```
mysql> show variables like '%char%';
+--------------------------+----------------------------------------------
------------------------------------------+
| Variable_name            | Value
|
+--------------------------+----------------------------------------------
------------------------------------------+
| character_set_client     | utf8
|
| character_set_connection | utf8mb4
|
| character_set_database   | latin1
|
| character_set_filesystem | binary
|
| character_set_results    | utf8mb4
|
| character_set_server     | latin1
|
| character_set_system     | utf8
|
| character_sets_dir       | /home/louis/src/pxc-repo/Percona-XtraDB-
Cluster-5.6.20-68.0/debug/mysql/share/charsets/ |
+--------------------------+----------------------------------------------
------------------------------------------+
8 rows in set (0.00 sec)

mysql> set character_set_connection = utf8;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into t1 values('\U+1F604');
ERROR 1366 (HY000): Incorrect string value: '\xF0\x9F\x98\x84' for column
'c1' at row 1
```

set names做了什么?

```
int set_var_collation_client::update(THD *thd)
{
  thd->variables.character_set_client= character_set_client;
  thd->variables.character_set_results= character_set_results;
  thd->variables.collation_connection= collation_connection;
  thd->update_charset();
  thd->protocol_text.init(thd);
  thd->protocol_binary.init(thd);
  return 0;
}
```

可以看到SET NAMES就是同时修改了**character_set_client**，**character_set_results**和**character_set_connection**

set charset做了什么？

SET CHARSET utf8mb4;

会修改character_set_client,character_set_results为utf8mb4,而character_set_connection设置为character_set_database。

init_connect为何不起作用？

init_connect用于指定新的连接初始化执行的一些命令。为了让指定连接的默认字符集，DBA可以通过设置init_connect达到目的。

```
set global init_connect='set names utf8mb4';
```

但是在设置之后，我再次登录发现字符集变量并没有改变，从代码中可以看出原因：

```
  if (opt_init_connect.length && !(sctx->master_access & SUPER_ACL))
  {
    execute_init_command(thd, &opt_init_connect, &LOCK_sys_init_connect);
```

可以看到，除了需要设置init_connect，还需要此用户没有SUPER权限，而DBA都有SUPER权限，所以我们设置之后，对我们自己没有作用，但是对于一般的用户来说是起作用的，因为一般用户基本都不具备SUPER权限。

init_connect是在连接创建之后执行的，也就是完成handshake之后设置的。所以会覆盖连接时设置的字符集属性。

JDBC连接串characterEncoding的作用

我们知道JDBC连接串里面可以指定字符集，但是无法指定utf8mb4，只能指定utf8，因为JDBC驱动端的编码只有UTF8这一种叫法。JDBC会通过server返回的版本号，来判断server是否支持utf8mb4(5.5.2之后才支持)。

类似mysql客户端，会在connection建立的时候，发送一个客户端的编码，这个就是由characterEncoding指定的，如果不指定的话，默认就是UTF8：

```
    private String getEncodingForHandshake() {
        String enc = this.connection.getEncoding();
        if (enc == null) {
            enc = "UTF-8";
        }
        return enc;
    }
```

JDBC连接MySQL分为几个步骤

1.第一个步骤和mysql客户端一样，建立连接的过程，这个过程会把characterEncoding传过去，效果类似set names.此时服务器端也会给客户端返回一个服务器的cs，

```
static bool send_server_handshake_packet(MPVIO_EXT *mpvio,
                                         const char *data, uint data_len)
{
...

  int2store(end, mpvio->client_capabilities);
  /* write server characteristics: up to 16 bytes allowed */
  end[2]= (char) default_charset_info->number;
```

与前面说所的一样，default_charset_info值在服务器启动的时候被赋予值，默认值是latin，如果设置了character_set_server的话，会改变这个值。

2.连接之后会show variables来获取自己的字符集属性。

3.调用**configureClientCharacterSet**来调整字符集。

- 如果设置了characterEncoding=utf8且服务器返回的cs=utf8mb4，则执行set names utf8mb4;
- 如果没有设置任何characterEncoding，会使用utf8作为默认的字符集，然后会比较服务器返回的cs，如果cs和cs_client,cs_connection不一致的话，会按照服务器返回的cs执行set names。

从JDBC执行set names的时机可以看出来，init_connect对于JDBC是不生效的，因为如果JDBC后面调整了字符集，那么就会覆盖了init_connect。