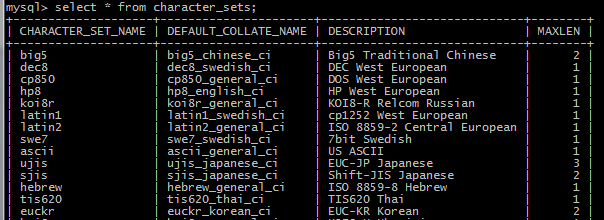
# Server(5.6.21)

## 一、支持字符集

通过查询information\_schema中的character\_sets系统表了解mysql支持那些字符集



CHARACTER\_SET\_NAME：字符集名，set names使用该名字进行字符集的更改

DEFAULT\_COLLATE\_NAME：字符比较方式名

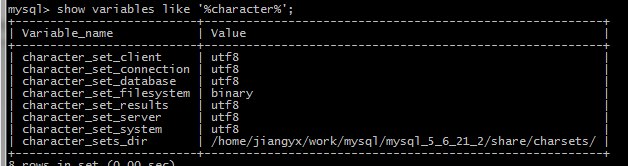
MAXLEN：该字符集下，单个字符最大的字节长度，如big5编码的一个汉字字符需要用2个字节进行存储，所有字符的存储都不会使用2字节以上的空间

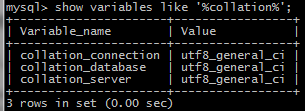
特殊说明

字符集中的字符可以有多种比较方式，比如以存储的二进制比较（也就是一个字节一个字节比较），或者以字符大小比较（如字符'a'和字符'中'的比较）

## 二、字符集相关变量

如下是mysql字符集相关的变量





### 变量说明

character\_set\_client：客户端的字符集

character\_set\_connection：连接的字符集，通过连接传入字符串所使用的字符集

character\_set\_server：服务器字符集，在创建database时所使用。create database为db指定的默认字符集是根据session级别character\_set\_server的值

character\_set\_database：在5.7中不推荐使用，该变量继承自character\_set\_server。每次使用use改变默认db，都会将session级别的变量调整为global级别一致的值。

character\_set\_results：生成结果集使用的字符串

character\_sets\_dir：mysql支持的字符集xml文档

character\_set\_system：操作系统当前环境的字符集

character\_set\_filesystem：文件系统的字符集

### 补充说明

1、character\_set\_connection/collation\_connection，character\_set\_database/collation\_database，character\_set\_server/collation\_server是成对的。修改对应的charset则collation也发生修改，并且collation是不可修改的

2、set names命令同时设置character\_set\_client/character\_set\_results/character\_set\_connection三个变量的session级别的字符集

3、create table为table指定的默认字符集是根据当前数据库的默认字符集

# C# connector：（6.9.6）

### connection对象的字符集确定

1、C#客户端通过连接串或者从服务器获取的server charset信息，创建driver上的encoding对象。首先从连接串中获取字符集，如果没有在连接串中指定字符集，则使用从服务器获取的server charset（character\_set\_server）字符集。

2、如果从服务器获取的character\_set\_client和character\_set\_connection两个属性和1中确定的字符集不同，则通过set names将这两个变量做session级别的修改

### 代码分析

在构造函数中，默认的charset就是utf-8

public Driver(MySqlConnectionStringBuilder settings)

{

encoding = Encoding.GetEncoding("UTF-8");

if (encoding == null)

throw new MySqlException(Resources.DefaultEncodingNotFound);

connectionString = settings;

serverCharSet = "utf-8";

serverCharSetIndex = -1;

maxPacketSize = 1024;

handler = new NativeDriver(this);

}

在连接过程中进行配置

1、是否在连接串中指定字符集，如果指定了，则使用连接串中指定的字符集。

2、没有指定字符集，则使用服务器的server charset，携程服务器的char set设置的也是utf-8的字符集，所以此处一定是utf-8的编码

3、如果在1和2都无法确定，使用utf-8的字符集

public virtual void Configure(MySqlConnection connection)

{

bool firstConfigure = false;

// if we have not already configured our server variables

// then do so now

if (serverProps == null)

{

firstConfigure = true;

// if we are in a pool and the user has said it's ok to cache the

// properties, then grab it from the pool

try

{

if (Pool != null && Settings.CacheServerProperties)

{

if (Pool.ServerProperties == null)

Pool.ServerProperties = LoadServerProperties(connection);

serverProps = Pool.ServerProperties;

}

else

serverProps = LoadServerProperties(connection);

LoadCharacterSets(connection);

}

catch (MySqlException ex)

{

// expired password capability

if (ex.Number == 1820)

{

IsPasswordExpired = true;

return;

}

throw;

}

}

#if AUTHENTICATED

string licenseType = serverProps["license"];

if (licenseType == null || licenseType.Length == 0 ||

licenseType != "commercial")

throw new MySqlException( "This client library licensed only for use with commercially-licensed MySQL servers." );

#endif

// if the user has indicated that we are not to reset

// the connection and this is not our first time through,

// then we are done.

if (!Settings.ConnectionReset && !firstConfigure) return;

//1、2

string charSet = connectionString.CharacterSet;

if (charSet == null || charSet.Length == 0)

{

if (serverCharSetIndex >= 0 && charSets.ContainsKey(serverCharSetIndex))

charSet = (string)charSets[serverCharSetIndex];

else

charSet = serverCharSet;

}

if (serverProps.ContainsKey("max\_allowed\_packet"))

maxPacketSize = Convert.ToInt64(serverProps["max\_allowed\_packet"]);

// now tell the server which character set we will send queries in and which charset we

// want results in

MySqlCommand charSetCmd = new MySqlCommand("SET character\_set\_results=NULL",

connection);

charSetCmd.InternallyCreated = true;

string clientCharSet;

serverProps.TryGetValue("character\_set\_client", out clientCharSet);

string connCharSet;

serverProps.TryGetValue("character\_set\_connection", out connCharSet);

if ((clientCharSet != null && clientCharSet.ToString() != charSet) ||

(connCharSet != null && connCharSet.ToString() != charSet))

{

MySqlCommand setNamesCmd = new MySqlCommand("SET NAMES " + charSet, connection);

setNamesCmd.InternallyCreated = true;

setNamesCmd.ExecuteNonQuery();

}

charSetCmd.ExecuteNonQuery();

//3

if (charSet != null)

Encoding = CharSetMap.GetEncoding(Version, charSet);

else

Encoding = CharSetMap.GetEncoding(Version, "utf-8");

handler.Configure();

}

### 补充说明

一旦确定字符集之后，connection上的字符集将不会发生改变。即使执行set names的sql 语句，进行编码转换的依然是通过1和2确定后的字符集；此时服务器认为，传过来的数据是通过set names指定的编码字符，将这些数据按照set names指定的字符集转换为目标字符集。

C# connector

server processor

连接指定编码字符串

server processor

目标编码字符串