目录

4.1mysql插入中文数据乱码1

4.3.1mysql数据插入中文数据乱码案例1

4.3.2mysql命令插入中文数据不乱码实战3

4.3.3执行SQL文件插入中文数据不乱码实战4

4.3.4通过mysql命令加字符集参数导入数据解决乱码………………………………………..……..…5

4.3.5 mysql插入中文不乱码种方法小结……………………………………………..………..……5

4.4mysql数据库字符集知识6

4.4.1mysql数据库字符集介绍6

4.4.2mysql数据库常见字符集介绍7

4.4.3如何选择合适的字符集7

4.4.4查看当前mysql系统支持的字符集8

4.5mysql插入中文不乱码深度剖析10

4.5.1mysql数据库默认设置字符集10

4.5.2执行 set names latinl到底做什么？11

4.5.3,mysql命令参数—default-character-set=latinl在做什么11

4.6确保mysql数据库插入数据不乱码解决方案13

4.6.1更改mysql端字符集13

4.6.2统一mysql数据库及服务端字符集总结13

4.6.3彻底解决MySQL数据库插入中文乱码方案13

4.7如何更改生产mysql数据库表的字符集15

# 4.3mysql插入中文数据乱码问题

## 4.3.1 mysql数据库插入中文数据乱码案例

1.mysql中添加中文数据乱码演示

a.创建一个oldboy测试数据库并查看建表语句

mysql> create database oldboy;

Query OK, 1 row affected (0.01 sec)

mysql> show create database oldboy\G

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Database: oldboy

Create Database: CREATE DATABASE `oldboy` /\*!40100 DEFAULT CHARACTER SET utf8 \*/

1 row in set (0.00 sec)

提示：默认情况如果在安装mysql时没有指定字符集，则mysql默认字符集是latinl

b.在oldboy库下创建一个student表，并查看表结构和建表语句

mysql> use oldboy

Database changed

mysql> create table student(id int(4) NOT NULL AUTO\_INCREMENT,name char(20) NOT NULL,PRIMARY KEY (id));

Query OK, 0 rows affected (0.13 sec)

c.批量插入数据到oldboy库是student表

mysql> insert into student values(1,'oldboy'),(2,'oldgirl'),(3,'inca'),(4,'zuma'),(5,'kaka');

Query OK, 5 rows affected (0.00 sec)

Records: 5 Duplicates: 0 Warnings: 0

mysql> select \* from student;

+----+---------+

| id | name |

+----+---------+

| 1 | oldboy |

| 2 | oldgirl |

| 3 | inca |

| 4 | zuma |

| 5 | kaka |

+----+---------+

5 rows in set (0.00 sec)

提示：数字和英文的数据都是正常的，不会有乱码问题。

d.插入一条中文数据看看

mysql> insert into student values(6,'老男孩');

Query OK, 1 row affected (0.00 sec)

mysql> select \* from student;

+----+-----------+

| id | name |

+----+-----------+

| 1 | oldboy |

| 2 | oldgirl |

| 3 | inca |

| 4 | zuma |

| 5 | kaka |

| 6 | ？？？ | 问题出现中文内容乱码

+----+-----------+

6 rows in set (0.00 sec)

## 4.3.2mysql命令行插入中文不乱码实战

1.查看建表语句，要特别注意表语句结尾的latinl

mysql> show create table student\G

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: student

Create Table: CREATE TABLE `student` (

`id` int(4) NOT NULL AUTO\_INCREMENT,

`name` char(20) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=7 DEFAULT CHARSET=utf8

1 row in set (0.00 sec)

2.设置mysql客户端的字符集和建表字符集latinl一致

mysql> set names latin1; #设置字符集插入数据表的字符集然后插入中文语句测试

1 row in set (0.00 sec)

3.继续插入一条中文数据

mysql> insert into student values(7,'老男孩'); Linux命令行字符集也要调整

Query OK, 1 row affected (0.00 sec)

mysql> select \* from student;

+----+-----------+

| id | name |

+----+-----------+

| 1 | oldboy |

| 2 | oldgirl |

| 3 | inca |

| 4 | zuma |

| 5 | kaka |

| 6 | ??? |

| 7 | 老男孩 |

+----+-----------+

7 rows in set (0.00 sec)

这个就是mysql命令行插入数据确保不乱码的方法，如果要是更新数据很多，就需要执行SQL文件更新数据。

## 4.3.3执行SQL文件插入中文数据不乱码实战

1.将要更新的多个SQL语句放到文本文件中如test.sql所示

[root@db02 ~]# cat test.sql

set names latin1;

insert into student values(8,'老女孩');

mysql> system cat test.sql

set names latin1;

insert into student values(8,'老女孩');

mysql>

提示：务必加入set name latin1；确保插入数据不乱码

2.在mysql命令行通过source调用test.sql文件插入数据

mysql> source test.sql

Query OK, 0 rows affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

mysql> select \* from student;

+----+-----------+

| id | name |

+----+-----------+

| 1 | oldboy |

| 2 | oldgirl |

| 3 | inca |

| 4 | zuma |

| 5 | kaka |

| 6 | ??? |

| 7 | 老男孩 |

| 8 | 老女孩 |依然不乱码也是set names latin1在起作用;

+----+-----------+

8 rows in set (0.00 sec)

不乱码小结：

1. DQL,DML语句之前“set name系统及库表字符集“！
2. “set names 系统及库表的字符集“就是在改mysql客户端字符集

## 4.3.4通过mysql命令加字符集参数导入数据解决乱码

1.把要更新的多个SQL语句放入文本文件中，足矣，此次语句不带set names latin1

#set names latin1;

insert into student values(9,'张三');

1. 通过mysql命令加上字符集参数指定latin1字符集导入test.sql语句

[root@db02 ~]# mysql -uroot -p123456 -S /data/3306/mysql.sock --default-character-set=latin1 oldboy <test.sql

通过-e参数在mysql库查看结果

[root@db02 ~]# mysql -uroot -p123456 -S /data/3306/mysql.sock -e "set names latin1;select \* from oldboy.student;"

+----+-----------+

| id | name |

+----+-----------+

| 1 | oldboy |

| 2 | oldgirl |

| 3 | inca |

| 4 | zuma |

| 5 | kaka |

| 6 | ??? |

| 7 | 老男孩 |

| 8 | 老女孩 |

| 9 | 张三 |依然不是乱码也是--default-character-set=latin1起作用

+----+-----------+

## 4.3.5 mysql插入中文不乱码的5中方法小结

方法（1）：登录mysql，先做“set names latin1“然后执行更新语句执行语句文件

mysql> set names latin1;

Query OK, 0 rows affected (0.00 sec)

mysql> source test.sql

Query OK, 0 rows affected (0.00 sec)确保test.sql文件格式正确

方法（2）在SQL文件中指定set names latin1，然后登录执行

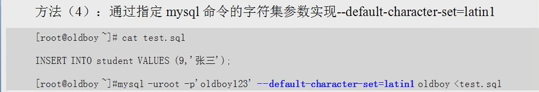
mysql> source test.sql

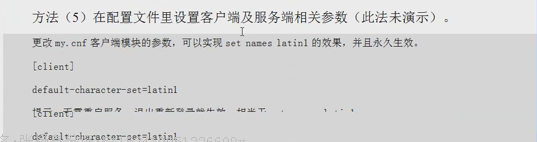
Query OK, 0 rows affected (0.00 sec)

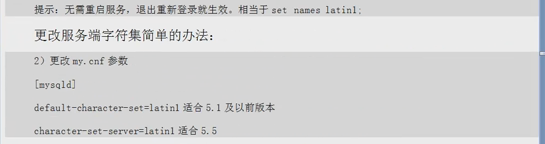
方法（3）：在SQL文件中指定set names latin1然后通过mysql命令导入

[root@db02 ~]# mysql -uroot -p123456 -S /data/3306/mysql.sock <test.sql

[root@db02 ~]# mysql -uroot -p123456 -S /data/3306/mysql.sock -e "set names latin1;select \* from oldboy.student;"

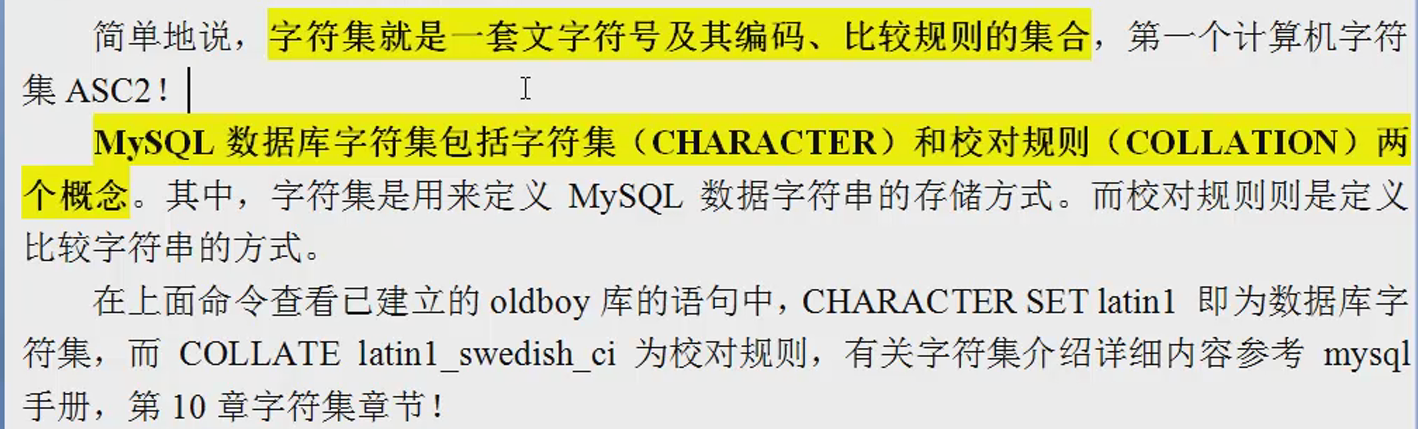


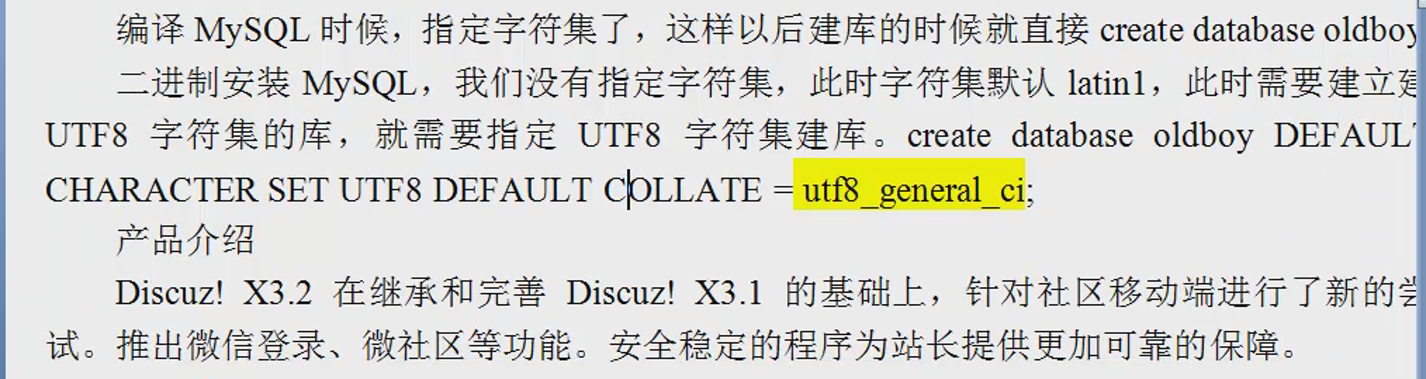


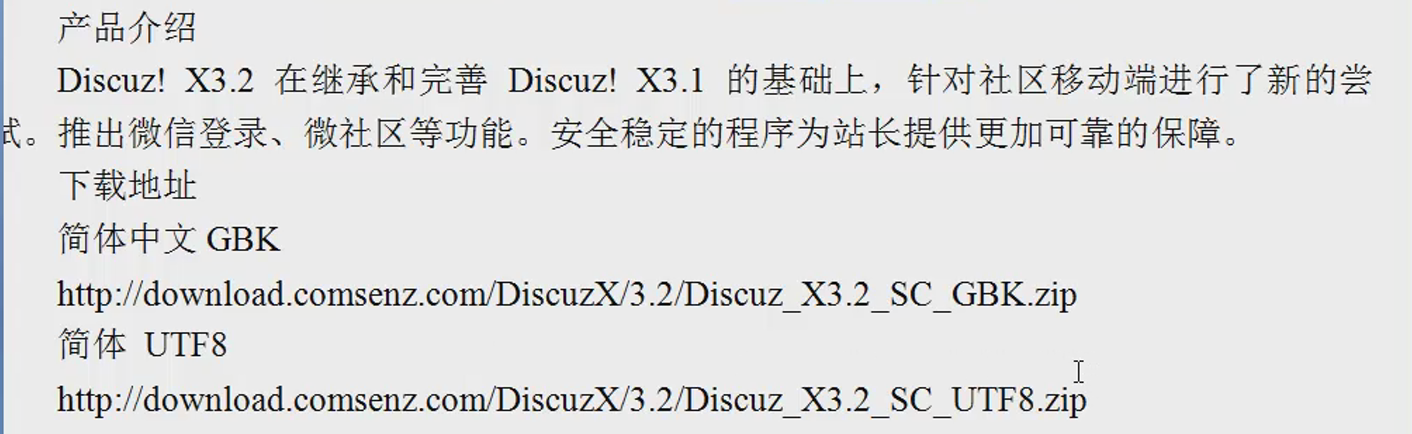


# 4.4mysql数据库字符集知识

## 4.4.1 mysql数据库字符集介绍

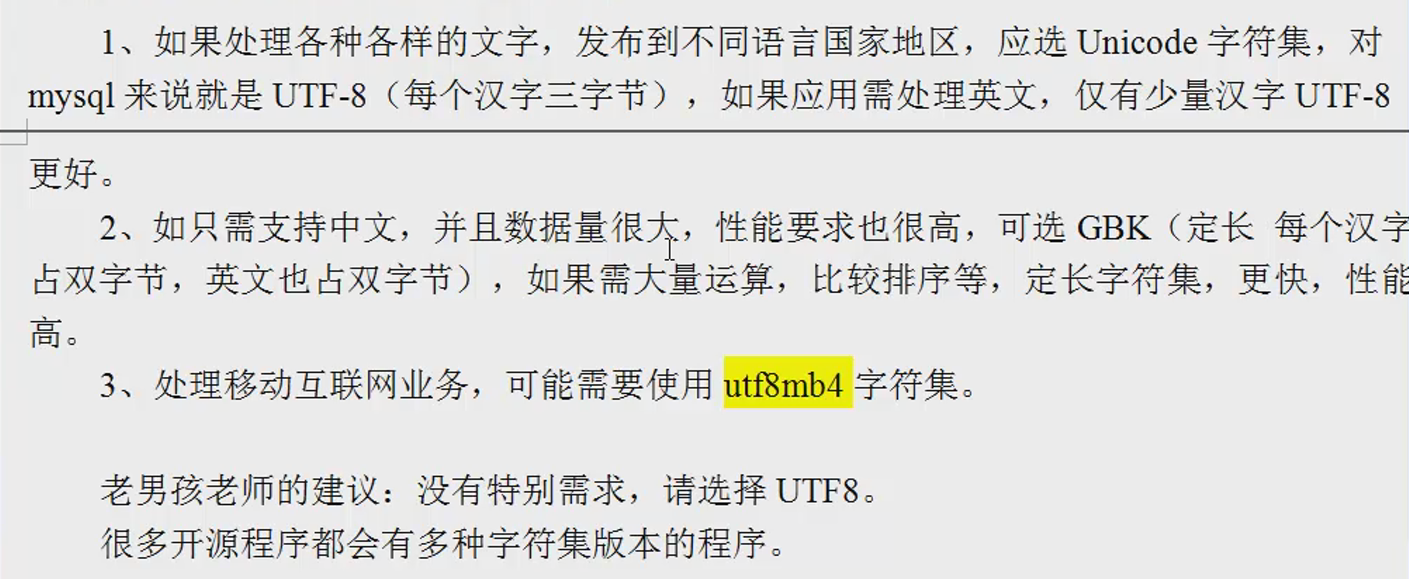






## 4.4.2mysql数据库常见字符集介绍

## 4.4.3mysql如何处理合适字符集



## 4.4.4 查看当前mysql系统支持的字符集

[root@db02 ~]# mysql -uroot -p123456 -S /data/3306/mysql.sock -e "show characterset;" set;"

+----------+-----------------------------+---------------------+--------+

| Charset | Description | Default collation | Maxlen |

+----------+-----------------------------+---------------------+--------+

| big5 | Big5 Traditional Chinese | big5\_chinese\_ci | 2 |

| dec8 | DEC West European | dec8\_swedish\_ci | 1 |

| cp850 | DOS West European | cp850\_general\_ci | 1 |

| hp8 | HP West European | hp8\_english\_ci | 1 |

| koi8r | KOI8-R Relcom Russian | koi8r\_general\_ci | 1 |

| latin1 | cp1252 West European | latin1\_swedish\_ci | 1 |

| latin2 | ISO 8859-2 Central European | latin2\_general\_ci | 1 |

| swe7 | 7bit Swedish | swe7\_swedish\_ci | 1 |

| ascii | US ASCII | ascii\_general\_ci | 1 |

| ujis | EUC-JP Japanese | ujis\_japanese\_ci | 3 |

| sjis | Shift-JIS Japanese | sjis\_japanese\_ci | 2 |

| hebrew | ISO 8859-8 Hebrew | hebrew\_general\_ci | 1 |

| tis620 | TIS620 Thai | tis620\_thai\_ci | 1 |

| euckr | EUC-KR Korean | euckr\_korean\_ci | 2 |

| koi8u | KOI8-U Ukrainian | koi8u\_general\_ci | 1 |

| gb2312 | GB2312 Simplified Chinese | gb2312\_chinese\_ci | 2 |

| greek | ISO 8859-7 Greek | greek\_general\_ci | 1 |

| cp1250 | Windows Central European | cp1250\_general\_ci | 1 |

| gbk | GBK Simplified Chinese | gbk\_chinese\_ci | 2 |

| latin5 | ISO 8859-9 Turkish | latin5\_turkish\_ci | 1 |

| armscii8 | ARMSCII-8 Armenian | armscii8\_general\_ci | 1 |

| utf8 | UTF-8 Unicode | utf8\_general\_ci | 3 |

| ucs2 | UCS-2 Unicode | ucs2\_general\_ci | 2 |

| cp866 | DOS Russian | cp866\_general\_ci | 1 |

| keybcs2 | DOS Kamenicky Czech-Slovak | keybcs2\_general\_ci | 1 |

| macce | Mac Central European | macce\_general\_ci | 1 |

| macroman | Mac West European | macroman\_general\_ci | 1 |

| cp852 | DOS Central European | cp852\_general\_ci | 1 |

| latin7 | ISO 8859-13 Baltic | latin7\_general\_ci | 1 |

| utf8mb4 | UTF-8 Unicode | utf8mb4\_general\_ci | 4 |

| cp1251 | Windows Cyrillic | cp1251\_general\_ci | 1 |

| utf16 | UTF-16 Unicode | utf16\_general\_ci | 4 |

| cp1256 | Windows Arabic | cp1256\_general\_ci | 1 |

| cp1257 | Windows Baltic | cp1257\_general\_ci | 1 |

| utf32 | UTF-32 Unicode | utf32\_general\_ci | 4 |

| binary | Binary pseudo charset | binary | 1 |

| geostd8 | GEOSTD8 Georgian | geostd8\_general\_ci | 1 |

| cp932 | SJIS for Windows Japanese | cp932\_japanese\_ci | 2 |

| eucjpms | UJIS for Windows Japanese | eucjpms\_japanese\_ci | 3 |

+----------+-----------------------------+---------------------+--------+

mysql> show variables like "character\_set%";

+--------------------------+-------------------------------------------+

| Variable\_name | Value |

+--------------------------+-------------------------------------------+

| character\_set\_client | utf8 |

| character\_set\_connection | utf8 |

| character\_set\_database | utf8 |

| character\_set\_filesystem | binary |

| character\_set\_results | utf8 |

| character\_set\_server | utf8 |

| character\_set\_system | utf8 |

| character\_sets\_dir | /application/mysql-5.5.32/share/charsets/ |

+--------------------------+-------------------------------------------+

8 rows in set (0.00 sec)

# 4.5mysql插入中文数据不乱码深度解剖

## 4.5.1mysql数据库默认设置字符集是什么

1.首先看下默认情况的字符集

mysql> show variables like "character\_set%";

+--------------------------+-------------------------------------------+

| Variable\_name | Value |

+--------------------------+-------------------------------------------+

| character\_set\_client | utf8 |

| character\_set\_connection | utf8 |

| character\_set\_database | utf8 |

| character\_set\_filesystem | binary |

| character\_set\_results | utf8 |

| character\_set\_server | utf8 |

| character\_set\_system | utf8 |

| character\_sets\_dir | /application/mysql-5.5.32/share/charsets/ |

+--------------------------+-------------------------------------------+

更改Linux系统字符集变量后，查看mysql中文字符集的变化

[root@db02 ~]# echo $LANG

en

[root@db02 ~]# mysql -uroot -p123456 -S /data/3306/mysql.sock -e "show variables like 'character\_set%';"

+--------------------------+-------------------------------------------+

| Variable\_name | Value |

+--------------------------+-------------------------------------------+

| character\_set\_client | utf8 # 客户端字符集 |

| character\_set\_connection | utf8 # 连接字符集 |

| character\_set\_database | utf8 #数据库字符集 |

| character\_set\_filesystem | binary |

| character\_set\_results | utf8 #返回结果字符集 |

| character\_set\_server | utf8 #服务器字符集 |

| character\_set\_system | utf8 |

| character\_sets\_dir | /application/mysql-5.5.32/share/charsets/ |

[root@db02 ~]#

mysql> show variables like "character\_set%";

+--------------------------+-------------------------------------------+

| Variable\_name | Value |

+--------------------------+-------------------------------------------+

| character\_set\_client | latin1 set names 字符集（客户）; |

| character\_set\_connection | latin1 set names 字符集（客户）; |

| character\_set\_database | utf8 |

| character\_set\_filesystem | binary |

| character\_set\_results | latin1 set names 字符集（客户）; |

| character\_set\_server | utf8 |

| character\_set\_system | utf8 |

| character\_sets\_dir | /application/mysql-5.5.32/share/charsets/ |

+--------------------------+-------------------------------------------+

8 rows in set (0.00 sec)

## 4.5.2数据库不乱码（utf8）：

1. Linux客户端字符集； （UTF8）
2. Linux服务端字符集；LANG="zh\_CN.UTF-8"
3. 数据库 客户端字符集；（UTF8）

set names gbk 相当于下面三条命令

方法一：set names 字符集：临时生效

SET character\_set\_client = gbk;

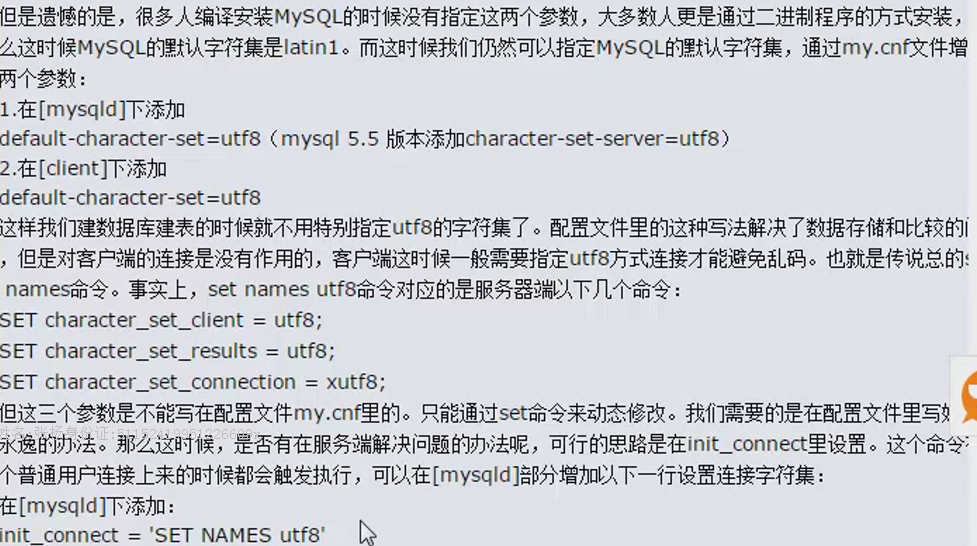
SET character\_set\_results = gbk;

SET character\_set\_connection = gbk;

方法二：

mysql -uroot -p123456 -S /data/3306/mysql.sock --default-character-set=gbk 修改客户端字符集

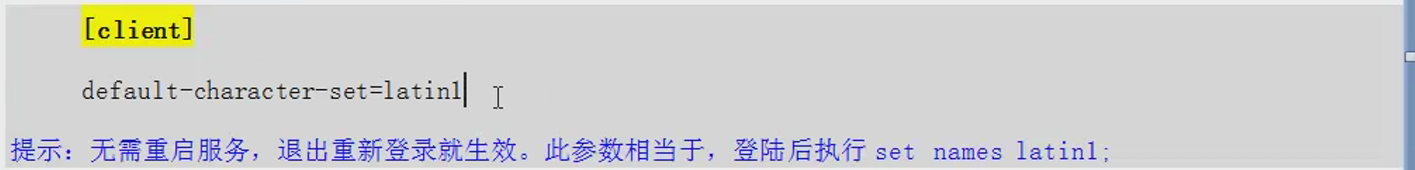
方法三：



方法四：编辑配置文件/etc/my.cnf（多实例）

[client]

--default-character-set=gbk



1. 数据库 服务端字符集；

[root@MySQL ~]# grep -server -B 1 /data/3306/my.cnf

[mysqld]

character-set-server=gbk

=============================

character\_set\_database | gbk |

character\_set\_server | gbk

其他编译的时候指定的字符集：

-DDEFAULT\_CHARSET=utf8\

-DDEFAULT\_COLLATION=utf8\

-DEXIRA\_CHARASETS=gbk，gbk2312,utf8,ascii\

1. 编译的时候指定服务端字符集：具体库的字符集；
2. 表的字符集；

mysql> show create table student；

+---------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------+

| Table | Create Table |

+---------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------+

| student | CREATE TABLE `student` (

`id` int(4) NOT NULL AUTO\_INCREMENT,

`name` char(20) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=10 DEFAULT CHARSET=utf8 |

+---------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------+

1 row in set (0.00 sec)

mysql> alter table student character set = gbk collate gbk\_chinese\_ci;

Query OK, 9 rows affected (0.21 sec)

Records: 9 Duplicates: 0 Warnings: 0

mysql> show create table student;

+---------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+

| Table | Create Table |

+---------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+

| student | CREATE TABLE `student` (

`id` int(4) NOT NULL AUTO\_INCREMENT,

`name` char(20) CHARACTER SET utf8 NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=10 DEFAULT CHARSET=gbk |

+---------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+

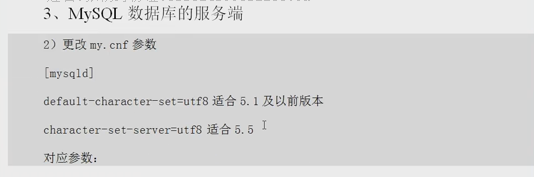
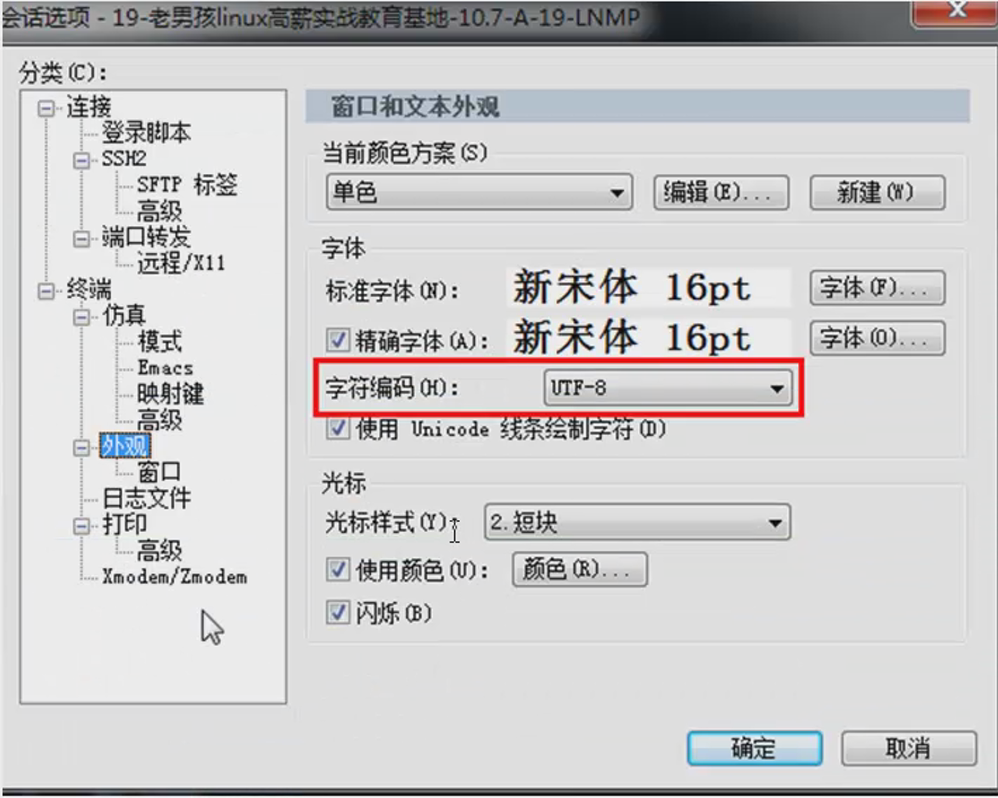
1 row in set (0.00 sec)

mysql>

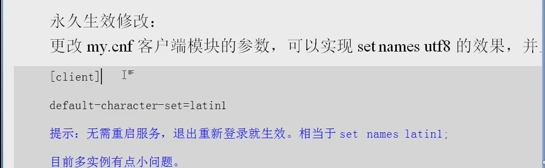
1. PHP/Java程序字符集；

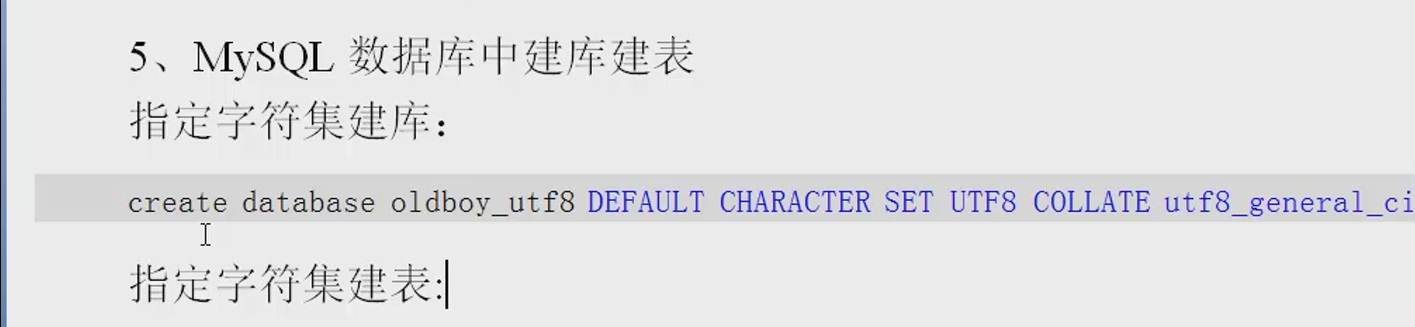
# 4.6确保mysql数据库插入数据不乱码解决方案

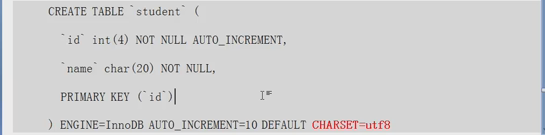








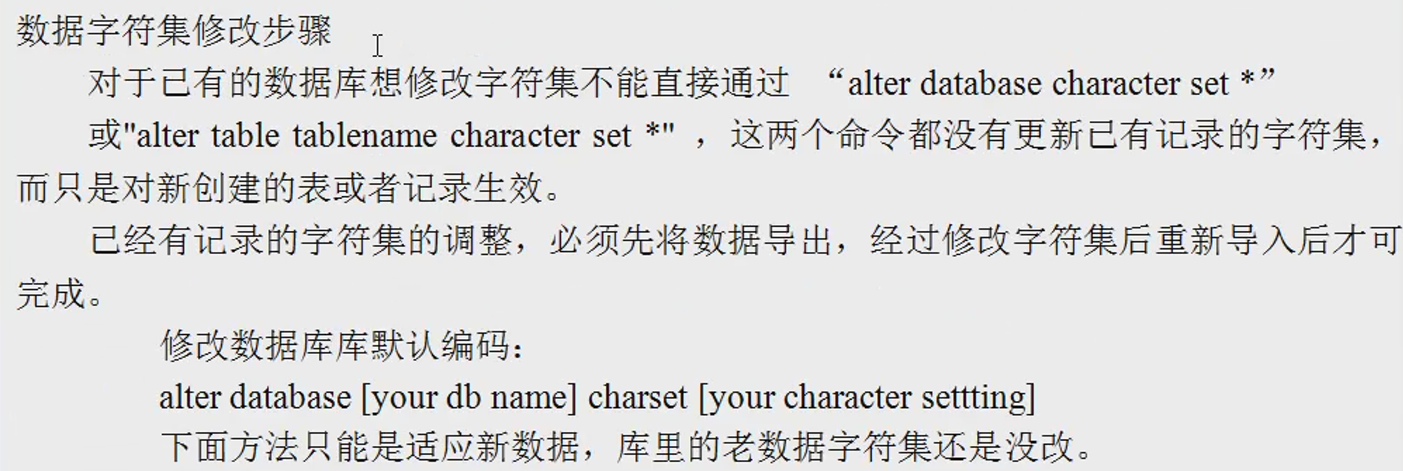






http://blog.sina.com.cn/s/blog\_7c35df9b010122ir.html

# 4.7 如何更改生产场景数据库表的字符集



1.更改oldboy字符集

mysql> alter database oldboy character set latin1 collate = latin1\_swedish\_ci;

Query OK, 1 row affected (0.00 sec)

mysql> show create database oldboy\G

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Database: oldboy

Create Database: CREATE DATABASE `oldboy` /\*!40100 DEFAULT CHARACTER SET latin1 \*/

1 row in set (0.00 sec)

2.更改student表字符集

mysql> alter table student character set latin1;

Query OK, 9 rows affected (0.21 sec)

Records: 9 Duplicates: 0 Warnings: 0

mysql> show create table student;

+---------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+

| Table | Create Table |

+---------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+

| student | CREATE TABLE `student` (

`id` int(4) NOT NULL AUTO\_INCREMENT,

`name` char(20) CHARACTER SET utf8 NOT NULL,

PRIMARY KEY (`id`)

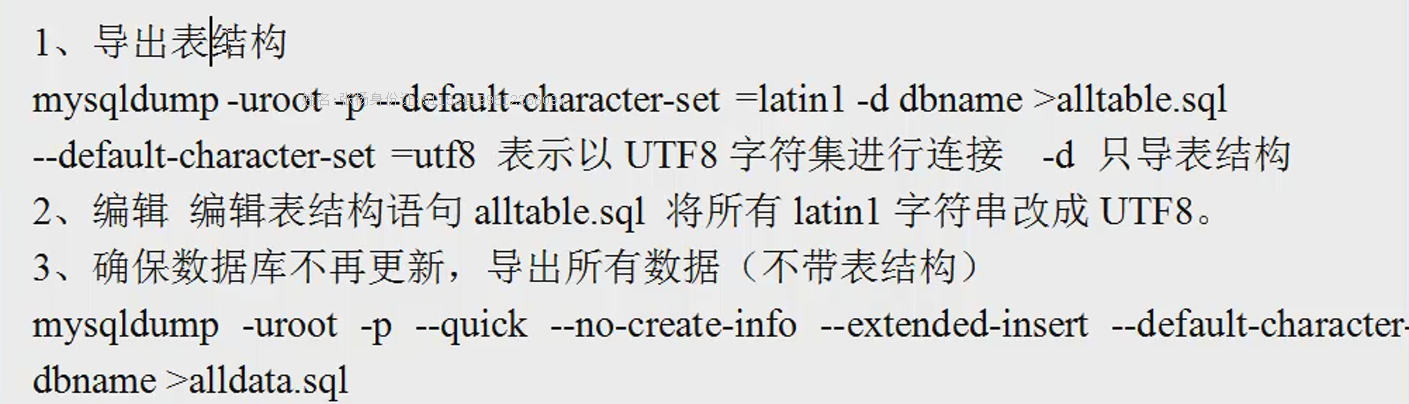
) ENGINE=InnoDB AUTO\_INCREMENT=10 DEFAULT CHARSET latin|

+---------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+

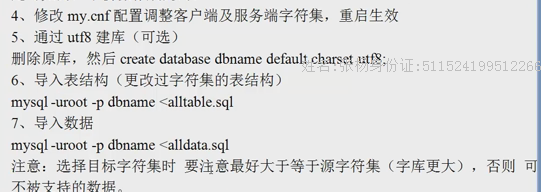
1 row in set (0.00 sec)

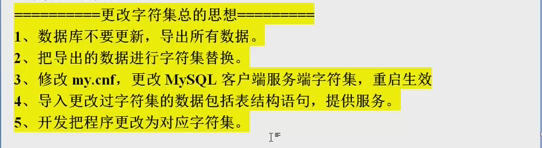
alter table student character set = gbk collate gbk\_chinese\_ci;

下面模拟将latin1字符集数据库修改成utf8字符集的实际过程









字符集修改问题：

对新数据生效，对老数据不行：

alter table test character set = gbk collate gbk\_chinese\_ci;

alter database oldboy CHARACTER SET latin1 COLLATE = latin1\_swedish\_ci;

生产修改方法：

1、mysqldump备份（数据量大先导出表结构，sed替换），在导出数据备份。

2、在表结构中把字符集改了。例如：sed替换

3、修改系统配置调整字符集生效

4、mysql把表结构还原，再把数据还原。

5、开发人员把程序的字符集调整好

MySQL日志种类：

错误日志(error log)

普通查询日志(general query log)

慢查询日志(slow query log)

二进制日志(binary log)

binlog:

row-level:

delete from test; 5条记录：

生成5条语句：

mysql> SET GLOBAL binlog\_format = 'ROW';

Query OK, 0 rows affected (0.00 sec)

mysql> show global variables like "binlog\_format%";

+---------------+-------+

| Variable\_name | Value |

+---------------+-------+

| binlog\_format | ROW |

+---------------+-------+

1 row in set (0.00 sec)

[root@MySQL 3306]# mysqlbinlog --base64-output="decode-rows" --verbose mysql-bin.000001

16:17:49

老男孩教育 2015/8/4 16:17:49

INSERT INTO `test` VALUES (1,'oldboy'),(2,'oldgirl'),(3,'inca'),(4,'zuma'),(5,'kaka');

老男孩教育 2015/8/4 16:18:17

### UPDATE `oldboy`.`test`

### WHERE

### @1=1

### @2='oldboy'

### SET

### @1=1

### @2='test'

### UPDATE `oldboy`.`test`

### WHERE

### @1=2

### @2='oldgirl'

### SET

### @1=2

### @2='test'

### UPDATE `oldboy`.`test`

### WHERE

### @1=3

### @2='inca'

老男孩教育 2015/8/4 16:18:30

ROW模式binlog记录的格式

set global wait\_timeout = 60;

set global interactive\_timeout = 60;

企业案例：mysql sleep线程过多的问题案例。

配置文件里修改：

[mysqld]

interactive\_timeout = 120 此参数设置后wait\_timeout自动生效。

wait\_timeout = 120

mysqladmin的相关命令：

mysqladmin -uroot -poldboy123 status

mysqladmin -uroot -poldboy123 -S /data/3306/mysql.sock -i 1 status

mysqladmin -uroot -poldboy123 flush-logs

mysqladmin -uroot -poldboy123 processlist

mysqladmin -uroot -poldboy123 extended-status

mysqladmin -uroot -poldboy123 processlist -i 1 实时跟踪。

watch mysqladmin -uroot -poldboy123 -S /data/3306/mysql.sock processlist乱码：

1. 环境本身不对，插入了错误数据，很难解决
2. 环境对的时候插入正确数据，客户端环境破坏，

数据库不乱码（utf8）：

1. Linux客户端字符集；LANG="en\_US.UTF-8"
2. Linux服务端字符集；
3. 数据库 客户端字符集；
4. 数据库 服务端字符集；
5. 具体库的字符集；
6. 表的字符集；
7. PHP/Java程序字符集；