**[MySQL字符集中文乱码终极解决方案和mysql查询中文问题解决方法[转贴]](http://www.oschina.net/question/565065_86411)**

[**9月19日成都OSC源创会火热报名 —— 送机械键盘和开源内裤**](http://city.oschina.net/chengdu/event/248043?channel=news_body)  http://www.oschina.net/img/hot3.png?t=1435877996000

**MySQL字符集终极解决方案 [转贴]**

开源数据库MySQL从来都是中小企业构建web应用的首选，特别是和PHP配合简直就是一对黄金搭档，深受web开发人员的喜爱。但自从4.1以来MySQL加入了多字符集的支持，很多MySQL使用者发现中文居然不能使用了，显示变成了一堆乱码！以致于很多人还在使用3.24.58的老版本，最近上MySQL网站，发现居然不提供3.24版本的下载了，MySQL已经彻底放弃3.24版本了。好在我还留有一份windows版的copy，就当作纪念吧。  
怎么会产生乱码现象的，怎么解决？只要翻下网上的解决方案，马上就可以得出答案：“在获得连接之后执行一句set names 'gb2312'”，但这样做的原因是什么呢？总结一下我的经验。  
  
**MySQL处理连接时，外部连接发送过来的SQL请求会根据以下顺序进行转换：**character\_set\_client           //客户连接所采用的字符集  
|  
character\_set\_connection  //MySQL连接字符集  
|  
character\_set\_database    //数据库所采用的字符集（表，列）  
|  
character\_set\_results        //客户机显示所采用的字符集  
  
  
一. 产生乱码的根本原因在于：  
1.客户机没有正确地设置client字符集，导致原先的SQL语句被转换成connection所指字符集，而这种转换，是会丢失信息的，如果client是utf8格式，那么如果转换成gb2312格式，这其中必定会丢失信息，反之则不会丢失。一定要保证connection的字符集大于client字符集才能保证转换不丢失信息。  
2. 数据库字体没有设置正确，如果数据库字体设置不正确，那么connection字符集转换成database字符集照样丢失编码，原因跟上面一样。  
  
二.为什么set names 'gb2312'就可以了呢  
set names 'gb2312'相当于这三条语句:  
set character\_set\_client = gb2312;  
set character\_set\_connection = gb2312;  
set character\_set\_results = gb2312;  
这样做的话，上述产生乱码的原因1就不存在了，因为编码格式都统一了，但是这样做并不是万金油。原因有：  
1.你的client不一定是用gb2312编码发送SQL的，如果编码不是gb2312那么转换成gb2312就会产生问题。  
2.你的数据库中的表不一定是gb2312格式，如果不是gb2312格式而是其他的比如说latin1，那么在存储字符集的时候就会产生信息丢失。  
  
**综上，终极解决方案如下:**  
1.首先要明确你的客户端时候何种编码格式，这是最重要的（IE6一般用utf8，命令行一般是gbk，一般程序是gb2312)  
2.确保你的数据库使用utf8格式，很简单，所有编码通吃。  
3.一定要保证connection字符集大于等于client字符集，不然就会信息丢失，比如latin1<gb2312<gbk<utf8  
若设置set character\_set\_client = gb2312，那么至少connection的字符集要大于等于gb2312，否则就会丢失信息  
4.以上三步做正确的话，那么所有中文都被正确地转换成utf8格式存储进了数据库，为了适应不同的浏览器，不同的客户端，你可以修改character\_set\_results来以不同的编码显示中文字体，由于utf8是大方向，因此web应用是我还是倾向于使用utf8格式显示中文的。  
  
  
**以上就是我的心得了。附上连接源码，现行设置，程序中就可以不考虑字符集问题了**  
include "conf/system.php";  
  
class Connection {  
private $conn;  
  
function \_\_construct() {  
global $mysql\_ipaddr, $mysql\_port, $mysql\_db, $mysql\_user, $mysql\_pass;  
  
try {  
$this->conn = new PDO("mysql:host=$mysql\_ipaddr;port=$mysql\_port;dbname=$mysql\_db", $mysql\_user, $mysql\_pass);  
} catch (PDOException $e) {  
print "MySQL服务器连接失败: " . $e->getMessage() . "<br>";  
die();  
}  
}  
  
public function getConnection() {  
if ($this->conn != null) {  
$this->conn->query("set character\_set\_client = gb2312");    //客户端使用gb2312格式  
$this->conn->query("set character\_set\_connection = utf8"); //连接字符集使用utf8格式  
$this->conn->query("set character\_set\_results = utf8");       //显示字符集使用utf8格式  
return $this->conn;  
}  
}  
  
public function closeConnection() {  
if ($this->conn != null) {  
$this->conn = null;  
}  
}  
}

**我现在在mysql上遇到一个问题,我们的字符集是gb2312.在中文模糊查找时,会有不相关的结果集.**

**从问题的根本原因分析，还有下面的问题。  
例：   
汉字“不”的第1、2字节ascii值分别为：178与187   
汉字“安”的第1、2字节ascii值分别为：176与178   
汉字“花”的第1、2字节ascii值分别为：187与168   
聪明的人已经看出来了：在字符串“安花”中模糊查找字符“不”字时，mysql系统也会认为两者匹配!**

出现这个问题的原因是：MySQL在查询字符串时是大小写不敏感的，在编绎MySQL时一般以ISO-8859字符集作为默认的字符集，因此在比较过程中中文编码字符大小写转换造成了这种现象。  
  
**方法一:**  
解决方法是对于包含中文的字段加上"binary"属性，使之作为二进制比较，例如将"name char(10)"改成"name char(10)binary"。   
  
**方法二:**  
如果你使用源码编译MySQL，可以编译MySQL时使用--with--charset=gbk 参数，这样MySQL就会直接支持中文查找和排序了。  
  
**方法三:**  
可以使用 Mysql 的 locate 函数来判断。以上述问题为例,使用方法为:  
SELECT \* FROM table WHERE locate(field,'李') > 0;  
本站使用的就是这种方法，感觉还不错。:P  
  
**方法四:**  
把您的Select语句改成这样,SELECT \* FROM TABLE WHERE FIELDS LIKE BINARY '%FIND%'即可!

升级的根本，如果想使用“正确”的字符集，还是先用mysqldump导出成文件，然后导入。

**转载声明：**本文转自<http://hi.baidu.com/xhero2008/blog/item/8d5b2f3fe2c617e955e7234c.html/cmtid/bd79d354f5c1da54d1090692>

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

**mysql查询中文问题解决方法[转贴]**

原文：

<http://blog.sina.com.cn/u/4909c13c010003va>   
  
  
**Q:**  
我在写一个查询条件时的问题如下：  
如我想写一个字段中包含“李”字的所有记录  
$str="李";  
select \* from table where field like '%$str%' ;  
显示的记录中除了包含”李”字的记录，还有不包含“李”字的记录。为什么？  
**A:**  
在MySQL中，进行中文排序和查找的时候，对汉字的排序和查找结果是错误的。这种情况在MySQL的很多版本中都存在。如果这个问题不解决，那么MySQL将无法实际处理中文。  
  
出现这个问题的原因是：MySQL在查询字符串时是大小写不敏感的，在编绎MySQL时一般以ISO-8859字符集作为默认的字符集，因此在比较过程中中文编码字符大小写转换造成了这种现象。  
  
方法一:  
解决方法是对于包含中文的字段加上"binary"属性，使之作为二进制比较，例如将"name char(10)"改成"name char(10)binary"。   
  
方法二:  
如果你使用源码编译MySQL，可以编译MySQL时使用--with--charset=gbk 参数，这样MySQL就会直接支持中文查找和排序了。  
  
方法三:  
可以使用 Mysql 的 locate 函数来判断。以上述问题为例,使用方法为:  
SELECT \* FROM table WHERE locate(field,'李') > 0;  
本站使用的就是这种方法，感觉还不错。:P  
  
方法四:   
把您的Select语句改成这样,SELECT \* FROM TABLE WHERE FIELDS LIKE BINARY '%FIND%'即可!

<http://blog.sina.com.cn/u/4909c13c010003va>  
  
  
**Q:**  
我在写一个查询条件时的问题如下：  
如我想写一个字段中包含“李”字的所有记录  
$str="李";  
select \* from table where field like '%$str%' ;  
显示的记录中除了包含”李”字的记录，还有不包含“李”字的记录。为什么？  
**A:**  
在MySQL中，进行中文排序和查找的时候，对汉字的排序和查找结果是错误的。这种情况在MySQL的很多版本中都存在。如果这个问题不解决，那么MySQL将无法实际处理中文。  
  
出现这个问题的原因是：MySQL在查询字符串时是大小写不敏感的，在编绎MySQL时一般以ISO-8859字符集作为默认的字符集，因此在比较过程中中文编码字符大小写转换造成了这种现象。  
  
方法一:  
解决方法是对于包含中文的字段加上"binary"属性，使之作为二进制比较，例如将"name char(10)"改成"name char(10)binary"。   
  
方法二:  
如果你使用源码编译MySQL，可以编译MySQL时使用--with--charset=gbk 参数，这样MySQL就会直接支持中文查找和排序了。  
  
方法三:  
可以使用 Mysql 的 locate 函数来判断。以上述问题为例,使用方法为:  
SELECT \* FROM table WHERE locate(field,'李') > 0;  
本站使用的就是这种方法，感觉还不错。:P  
  
方法四:   
把您的Select语句改成这样,SELECT \* FROM TABLE WHERE FIELDS LIKE BINARY '%FIND%'即可!

**转载声明：**本文转自[http://www.playhosts.com/bbs/read.php?tid=12357](http://hi.baidu.com/xhero2008/blog/item/8d5b2f3fe2c617e955e7234c.html/cmtid/bd79d354f5c1da54d1090692)

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

**Mysql 字符集转换及版本升级/降级的详细教程 [转贴]**

**原帖地址** [**http://club.muzone.cn/viewthread.php?tid=28605**](http://club.muzone.cn/viewthread.php?tid=28605)  
  
MySQL 4.1开始，对多语言的支持有了很大变化 (这导致了问题的出现)。尽管大部分的地方 (包括个人使用和主机提供商)，MySQL 3、4.0 仍然占主导地位；但 MySQL 4.1 乃至5.0是 MySQL 官方推荐的数据库，已经有主机提供商开始提供并将会越来越多；因为 latin1 在许多地方 (下边会详细描述具体是哪些地方) 作为默认的字符集，成功的蒙蔽了许多 PHP 程序的开发者和用户，掩盖了在中文等语言环境下会出现的问题。  
  
MySQL 4.1开始把多国语言字符集分的更加详细，所以导致数据库迁移，或则dz论坛升级到4.0后（dz4.0开始使用gbk或utf-8编码）出现乱码问题。  
  
MySQL 4.1的字符集支持(Character Set Support)有两个方面：字符集(Character set)和排序方式(Collation)。对于字符集的支持细化到四个层次: 服务器(server)，数据库(database)，数据表(table)和连接(connection)。  
  
查看系统的字符集和排序方式的设定可以通过下面的两条命令：

**引用:**

mysql> SHOW VARIABLES LIKE 'character\_set\_%';   
+--------------------------+----------------------------+   
| Variable\_name | Value |   
+--------------------------+----------------------------+   
| character\_set\_client | latin1 |   
| character\_set\_connection | latin1 |   
| character\_set\_database | latin1 |   
| character\_set\_results | latin1 |   
| character\_set\_server | latin1 |   
| character\_set\_system | utf8 |   
| character\_sets\_dir | /usr/share/mysql/charsets/ |   
+--------------------------+----------------------------+   
7 rows in set (0.00 sec)   
  
mysql> SHOW VARIABLES LIKE 'collation\_%';   
+----------------------+-------------------+   
| Variable\_name | Value |   
+----------------------+-------------------+   
| collation\_connection | latin1\_swedish\_ci |   
| collation\_database | latin1\_swedish\_ci |   
| collation\_server | latin1\_swedish\_ci |   
+----------------------+-------------------+   
3 rows in set (0.00 sec)

MySQL 4.1 对于字符集的指定可以细化到一台机器上安装的 MySQL，其中的一个数据库，其中的一张表，其中的一栏，应该用什么字符集。但是，传统的 Web 程序在创建数据库和数据表时并没有使用那么复杂的配置，它们用的是默认的配置，那么，默认的配置从何而来呢？  
  
编译 MySQL 时，指定了一个默认的字符集，这个字符集是 latin1；  
安装 MySQL 时，可以在配置文件 (my.ini) 中指定一个默认的的字符集，如果没指定，这个值继承自编译时指定的；  
启动 mysqld 时，可以在命令行参数中指定一个默认的的字符集，如果没指定，这个值继承自配置文件中的；  
此时 character\_set\_server 被设定为这个默认的字符集；  
当创建一个新的数据库时，除非明确指定，这个数据库的字符集被缺省设定为 character\_set\_server；  
当选定了一个数据库时，character\_set\_database 被设定为这个数据库默认的字符集；  
在这个数据库里创建一张表时，表默认的字符集被设定为 character\_set\_database，也就是这个数据库默认的字符集；  
当在表内设置一栏时，除非明确指定，否则此栏缺省的字符集就是表默认的字符集；  
这个字符集就是数据库中实际存储数据采用的字符集，mysqldump 出来的内容就是这个字符集下的；  
当我们按照原来的方式通过PHP存取MySQL数据库时，就算设置了表的默认字符集为utf8并且通过UTF-8编码发送查询，你会发现存入数据库的仍然是乱码。问题就出在这个connection连接层上。  
想要进行“正确”的存储和得到“正确”的结果，最方便的是在所有query开始之前执行一下：  
  
SET NAMES 'gbk';  
其中gbk是数据库字符集。  
  
它相当于下面的三句指令：  
SET character\_set\_client = gbk;  
SET character\_set\_results = gbk;  
SET character\_set\_connection = gbk;  
  
4.1和5.0默认使用的是latin1字符集(木头:妈的,老外真霸道,妄想让全世界都是使用瑞典字符集吗)  
如果我们只想使用gbk字符集存储和获取数据,  
我们在编译mysql 4.1和 5.0的时候，需要注意在my.ini或者my.cnf中添加两处参数

**CODE:**

[mysqld]   
default-character-set=utf8

[[复制到剪切板]](javascript:)

**CODE:**

#settings for clients (connection, results, clients)   
[mysql]   
default-character-set=utf8

[[复制到剪切板]](javascript:)

下面我们来说主题，如何转换数据库字符集  
两种方法，

**引用:**

第一种----更改存储字符集   
主要的思想就是把数据库的字符集有latin1改为gbk，big5,或者utf8; 以下操作必须拥有主机权限。假设当前操作的数据库名为：database   
  
导出   
首先需要把数据导为mysql4.0的格式，具体的命令如下：   
mysqldump -uroot -p --default-character-set=latin1 --set-charset=gbk --skip-opt databse > d4.sql   
  
--default-characte-set 以前数据库的字符集，这个一般情况下都是latin1的，   
--set-charset 导出的数据的字符集，这个可以设置为gbk，utf8,或者big5   
导入   
首先使用下面语句新建一个GBK字符集的数据库（test）   
  
CREATE DATABASE `d4` DEFAULT CHARACTER SET gbk COLLATE gbk\_chinese\_ci;   
然后把刚才导出的数据导入到当前的数据库中就ok了。   
  
mysql -uroot -p --default-character-set=gbk -f d4<d4.sql   
通过以上的导出和导入就把数据库的字符集改为正确的存储方式了。   
  
其中d4为新建库的名称，d4.sql为导出文件的名字   
  
但是这种方法，发现数据库数据存储量无端变大30%，真是郁闷

**引用:**

另外一种其实原理相同，但是需要手动操作，一般用于第一种方法失败后的选择   
不过这种方法如果数据库很大，估计很难做，因为光打开文件就能让你死机   
  
首先还是用phpmyadmin或者用mysql本身的dump导出 .sql文件   
  
然后用UltraEdit打开你备份的所有xxxx.sql文件，查找

**CODE:**

DEFAULT CHARSET=latin1

[[复制到剪切板]](javascript:)

latin1这里也许是别的，反正是你不想要的，要转成gbk或者big5的字符集   
把这个替换为“空”   
在查找

**CODE:**

CREATE TABLE cdb\_sessions (  
sid char(6) character set latin1 collate latin1\_bin NOT NULL default '',  
ip1 tinyint(3) unsigned NOT NULL default '0',  
ip2 tinyint(3) unsigned NOT NULL default '0',  
ip3 tinyint(3) unsigned NOT NULL default '0',  
ip4 tinyint(3) unsigned NOT NULL default '0',  
uid mediumint(8) unsigned NOT NULL default '0',  
username char(15) NOT NULL default '',  
groupid smallint(6) unsigned NOT NULL default '0',  
styleid smallint(6) unsigned NOT NULL default '0',  
invisible tinyint(1) NOT NULL default '0',  
`action` tinyint(1) unsigned NOT NULL default '0',  
lastactivity int(10) unsigned NOT NULL default '0',  
fid smallint(6) unsigned NOT NULL default '0',  
tid mediumint(8) unsigned NOT NULL default '0',  
nickname char(15) NOT NULL default '',  
UNIQUE KEY sid (sid)  
) ENGINE=HEAP MAX\_ROWS=1000;

[[复制到剪切板]](javascript:)

替换为

**CODE:**

CREATE TABLE `cdb\_sessions` (  
`sid` char(6) binary NOT NULL default '',  
`ip1` tinyint(3) unsigned NOT NULL default '0',  
`ip2` tinyint(3) unsigned NOT NULL default '0',  
`ip3` tinyint(3) unsigned NOT NULL default '0',  
`ip4` tinyint(3) unsigned NOT NULL default '0',  
`uid` mediumint(8) unsigned NOT NULL default '0',  
`username` char(15) NOT NULL default '',  
`groupid` smallint(6) unsigned NOT NULL default '0',  
`styleid` smallint(6) unsigned NOT NULL default '0',  
`invisible` tinyint(1) NOT NULL default '0',  
`action` tinyint(1) unsigned NOT NULL default '0',  
`lastactivity` int(10) unsigned NOT NULL default '0',  
`fid` smallint(6) unsigned NOT NULL default '0',  
`tid` mediumint(8) unsigned NOT NULL default '0',  
`nickname` char(15) NOT NULL default '',  
UNIQUE KEY `sid` (`sid`)  
) TYPE=HEAP MAX\_ROWS=2000;

[[复制到剪切板]](javascript:)

这一步更为简单的办法就是删除掉关于cdb\_sessions表的这一段，将来全新装一个d4，将这个表导出   
将其内容复制，粘贴到 sql文件的最后面[code][/code]   
  
保存后，再把这个sql文件导入到你的库中   
  
就OK了

用这两种方法就可以很方便的把4.1和5.0的mysql数据库降级到4.0  
简单的过程就是  
A导出4.1/5.0的库  
B进行处理，转换成gbk字符集  
C彻底卸载4.1或者5.0  
D安装4.0.26  
E然后导入处理完的库  
  
降级的时候导出库可以用这个方法  
mysqldump -uroot -p --default-character-set=latin1 --set-charset=gbk --skip-opt databse --compatible=mysql40 > d4.sql  
这样导出的就是4.0的库勒  
  
至于mysql版本的升级，  
如果数据文件中有中文信息，那么将MySQL 4.0的数据文件，直接拷贝到MySQL 4.1中就是不可以的，即便在my.ini中设置了default-character-set为正确的字符集。虽然貌似没有问题，但MySQL 4.1的字符集有一处非常恼人的地方，以gbk为例，原本MySQL 4.0数据中varchar,char等长度都会变为原来的一半，这样存储中文容量不变，而英文的存储容量就少了一半。这是直接拷贝数据文件带来的最大问题。  
  
所以，升级的根本，如果想使用“正确”的字符集，还是先用mysqldump导出成文件，然后导入。