

第一篇：数据库基本操作

一、数据库系统

1、定义：就一个软件，是计算机系统中专门管理数据，存储数据的软件。

2、组成：

数据库

数据库管理系统

二、MySQL 的体系结构

1、说白了就是客户端/服务器端 的体系结构。使用的是两个程序，第一个程序就是运行并将数据存放在数据库服务器的 MySQL 服务器程序，指的是 mysqld 程序，第二个，就是 MySQL 客户端程序，它负责连接到数据库服务上，通过它来发送指令让服务器进行执行来操作数据库中表内容的。

```
[root@localhost ~]# ps -le | grep mysqld
4 S     0 1488      1  0  80    0 - 1568 -          ?          00:00:00 mysqld_safe
4 S  1012 1518 1488  0  80    0 - 31767 -          ?          00:00:00 mysqld
```

2、MySQL 好处：

>自动处理并发

>远程使用 MySQL 客户端进行登陆即可操作数据库中的数据

3、MySQL 与 mysql 的区别

MySQL 是一个完整的数据系统

mysql 是 MySQL 的客户端程序（黑窗口）

4、SQL 语句：标准化的语言（结构化查询语言）

数据定义语言（DDL）：定义和管理数据对象的，比如建立数据库，创建表

数据定义语言（DML）：操作语言

数据控制语言（DCL）：管理数据库的语言

数据查询语言（DQL）：用于查询语句

5、连接数据库：

连接之前设置系统变量



cmd-> 命令行-》mysql

-u 用户名

-p 密码

-h 主机名

-b 关闭风鸣器

-P 端口 3306

```
C:\Users\Administrator>mysql -hlocalhost -uroot -b -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
```

欢迎进入 MySQL 系统，你可以使用分号；或者\g 结束语句输入

6、MySQL 语法特点:

>分号结束

>提示继续输入符号->

>命令不区分大小写，系统字习惯用大写

```
mysql> show databases  
-> ;
```

```
mysql> SHOW DATABASES;
```

7、退出

>quit

>exit

\q

```
mysql> exit;  
Bye
```

9、常见的操作符

>\c 取消当前操作

>\g 代替分号

```
mysql> show databaseseeeee\c;  
ERROR:  
No query specified
```

```
mysql> show databases\g
```

三、MySQL 数据库基本操作

1、创建数据库

格式:

create database 数据库名

格式:

create database [if not exists] 数据库名

```
mysql> create database if not exists testdb;  
Query OK, 1 row affected (0.03 sec)
```

1>、每创建一个数据库，会在你的服务器的 data 目录下创建一个以数据名命名的文夹。

2>、数据库是唯一的



2、查看所有数据库

show databases

```
mysql> show databases;
```

3、进入某个数据库，进入到数据库下的指定的库你才能进行操作

格式: use 数据库名

```
mysql> use lamp;  
Database changed
```

4、删除数据库*****

格式: drop database 数据库名

格式: drop database [if exists] 数据库名

```
mysql> drop database lamp94;  
Query OK, 0 rows affected (0.00 sec)
```

4、查看当前使用的数据库

```
select database()
```

```
mysql> select database();
+-----+
| database |
+-----+
| lamp   |
+-----+
```

5、查看所有表

```
show tables
```

```
mysql> show tables;
```

6、创建数据表

```
create table 表名(列的信息)
```

```
mysql> CREATE TABLE username (
    -> id int<10> PRIMARY KEY NOT NULL AUTO_INCREMENT,
    -> name char<32>,
    -> idCode char<18> 最后没有逗号
    -> ) ENGINE=MyISAM CHARSET=utf8;
Query OK, 0 rows affected (0.20 sec)
```

7、查看表结构

```
desc 表名
```

```
mysql> desc username;
+-----+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key  | Default | Extra       |
+-----+-----+-----+-----+-----+-----+
| id   | int<10> | NO   | PRI  | NULL    | auto_increment |
| name | char<32> | YES  |      | NULL    |              |
| idCode | char<18> | YES  |      | NULL    |              |
+-----+-----+-----+-----+-----+-----+
```

8、查看建表语句

```
show create table 表名
```

\G 样式竖起来

```
mysql> show create table user\G;
```

9、删除表

格式: drop table 数据表名

```
drop table [if exists] 数据表名
```

```
mysql> drop table if exists username;
Query OK, 0 rows affected (0.02 sec)
```

10、查看数据库信息

```
mysql> \s;
```

11、查看帮助文档

```
mysql> help create table;
```

```
C:\Users\Administrator>mysqldump -uroot -p -d lamp94 admin > d:\admin.sql
Enter password:
```

12、导出MySQL数据库(请勿进入数据库备份,直接在外面操作)

1、导出MySQL数据库的表结构以及表数据:

```
mysqldump -u 用户名 -p 数据库的名 > 导出的文件的名
```

```
mysqldump -u 用户名 -p 数据库的名 要导出的表名 > 导出的文件名
```

2、只导出表结构。(命令行执行)

```
mysqldump -u 用户名 -p -d 数据库名 > 导出的文件名
```

```
mysqldump -u 用户名 -p -d 数据库名 数据表名 > 导出的文件名
```

13、MySQL数据库导入

注意:

1、你得先进入数据库并选择了数据库

2、source 数据库文件名路径

第二篇：数据表结构操作

四、修改表数据操作

1、插入数据

```
insert into 表名 values("","");
insert into 表名 (列名 1, 列名 2....) values (值 1, 值 2....);
```

```
mysql> insert into test values('3','lilili');
Query OK, 1 row affected (0.13 sec)

mysql> insert into test (id,name) values ('4','wwwww');
Query OK, 1 row affected (0.08 sec)
```

快速插入测试数据：

```
mysql> insert into test(name) select name from test;
Query OK, 4 rows affected (0.06 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

指定列要与查询处理的数据对应一致

2、修改数据*****

格式：update 表名 set 列名='值'
update 表名 set 列名='值' where 列名=值
 列名>值
 列名<值

注意：修改的时候一定要加上 where 条件。

```
mysql> select * from test;
```

```
mysql> update test set name='wangwu' where id=1;
Query OK, 1 row affected (0.11 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

3、删除数据*****

格式：delete from 表名
 delete from 表名 where 条件

```
mysql> delete from test where id>=7;
Query OK, 2 rows affected (0.08 sec)
```

备注：修改和删除操作时先查出来数据再进行操作即可。

4、导出 MySQL 数据库

1、导出 MySQL 数据库的表结构以及表数据：

mysqldump -u 用户名 -p 数据库的名 > 导出的文件的名

mysqldump -u 用户名 -p 数据库的名 要导出的表名 > 导出的文件名

具体：

```
C:\Users\Administrator>mysqldump -u root -p --default-character-set=utf8 -d lamp test > d:\\test.bak.sql
Enter password:
```

注意：》在命令行执行语句 》语句结尾不要加上分号

2、只导出表结构。(命令行执行)

mysqldump -u 用户名 -p -d 数据库名 > 导出的文件名

mysqldump -u 用户名 -p -d 数据库名 表名 > 导出的文件名

5、MySQL 数据库导入

注意：

- 1、你得先进入数据库并选择了数据库
- 2、source 数据库文件名 **文件使用绝对路径并且使用 /**
- 3、CMD 窗口支持拖拽补全路径

五、修改密码

命令行操作第一种方法

1、mysqladmin -u 用户名 -p password 新密码

2、mysql> 2、set password for '用户名'@'登录的主机地址' = password('密码');

六、添加用户

格式：

grant 权限 on 数据库.数据表 to 用户名@登录主机 identified by '密码';

权限：all 代表所有权限，其他权限见 mysql 手册

```
mysql> grant all on lamp.user to admin@localhost identified by 'username';
Query OK, 0 rows affected (0.03 sec)
```

数据库.数据表.*.* 表示所有数据库里面的所有表

登录主机：你可以指定 ip 也可以指定 IP 段，如果你想在哪里都能登录用%，有些版本的 mysql 指定了%之后在本地不能登录，所以你需要在添加一个相同的用户他的登录地址为 localhost。

更新权限：

```
flush privileges;
```

查看权限：

```
show grants for '用户名'@'登录主机'
```

删除用户并删除用户的权限：

```
drop user '用户名'@'主机地址';
```

```
flush privileges;
```

七、数值类的数据类型

1、数值类的数据列表类型

》在声明整型数据列时，我们可以为它指定要显示数据的宽度 M(1~255)，如 int (5) 指定显示的宽度为 5 个字符，如果没有给它指定要显示的宽度，默认使用 11。显示的宽度只是用于显示而已，并不能限制取值的范围和占用空间，如 int (3) 至少显示 3 个字符，占用空间是 4 个字节，显示宽度的取值范围还是 int 规定的。

》通常配合着 zerofill 来进行使用，代表的是前导 0 如果你的数值没有超过你指定的位数的话，缺多少用 0 填充。

```

mysql> create table test4(
-> id int<3> zerofill,
-> name char<12>
-> >engine=innodb charset=utf8;
Query OK, 0 rows affected (0.50 sec)

mysql> insert into test4 (id,name) values (12,'zhangsan');
Query OK, 1 row affected (0.06 sec)

mysql> insert into test4 (id,name) values (12345,'zhangsan');
Query OK, 1 row affected (0.06 sec)

mysql> select * from test4;
+----+-----+
| id | name |
+----+-----+
| 12 | zhangsan |
| 12345 | zhangsan |
+----+-----+
2 rows in set (0.00 sec)

```

》 decimal 和 double 类型， decimal 类型里面存储的是字符串，通常我们会用这种类型来存储和财务有关的东西， double 类型会进行四舍五入
浮点数类型

float(M,D)单精度 double(M,D)双精度

定点数类型

decimal(M,D) M 为整数位+小数位

decimal (10,2) 不进行四舍五入 doubel (10,2) 会进行四舍五入

2、unsigned(无符号),可选属性。保存非负数或需要较大上限制值时候使用。使用这个约束类型的时候应该紧放在类型的后面 无符号最小值为 0

```

mysql> create table test5(
-> id int<3> unsigned,
-> name char<6>
-> >engine=innodb charset=utf8;
Query OK, 0 rows affected (0.56 sec)

mysql> insert into test5 (id,name) values (-1,'zhangsan');
Query OK, 1 row affected, 2 warnings (0.06 sec)

mysql> select * from test5;
+----+-----+
| id | name |
+----+-----+
| 0 | zhangsan |
+----+-----+
1 row in set (0.02 sec)

```

3、auto_increment(自增),可选属性，值从 1 开始，每行加一，你需要定义一个 not null 和给这个列指定一个索引（primary key , unique）因为该值不允许重复，删除记录时不会删除对应的该值的。

```

mysql> create table test6(
-> id int<3> auto_increment not null primary key,
-> name char<6>
-> >engine=innodb charset=utf8;
Query OK, 0 rows affected (0.48 sec)

```

4、没有插入默认为 null。

如果列的类型不加 not null 默认为 null , 如果该列不填值的话, 那该列的值为 null
如果列的类型为 not null , 如果该列不填值的话, 拿该列的值为空。

```
mysql> create table test7<
-> id int(3) not null ,
-> pid int(3)
-> engine=innodb charset=utf8;
Query OK, 0 rows affected (1.11 sec)

mysql> insert into test7 values (1,11);
mysql> insert into test7 (id) values (1);
mysql> insert into test7 (pid) values (11);
mysql> insert into test7 values ();
mysql> select * from test7;
+---+---+
| id | pid |
+---+---+
| 1 | 11 |
| 1 | NULL |
| 0 | 11 |
| 0 | NULL |
+---+---+
4 rows in set (0.00 sec)
```

5、如果设置为 not null 的情况下, 我们需要给该列设置默认值 default

```
mysql> create table test9<
-> name char(4) default 'san',
-> id int(4) auto_increment primary key
-> engine=innodb charset=utf8;
Query OK, 0 rows affected (0.48 sec)

mysql> insert into test9 (name) values ('lisi');
Query OK, 1 row affected (0.50 sec)

mysql> insert into test9 values ();
Query OK, 1 row affected (0.08 sec)

mysql> select * from test9;
+---+---+
| name | id |
+---+---+
| lisi | 1 |
| san | 2 |
+---+---+
2 rows in set (0.01 sec)
```

八、字符串类型

1、char 的长度为声明时的长度, varchar 长度为不是声明时长度而是根据插入的值来的。

char 定长 varchar 变长

2、char 列去掉了尾部的空格, varchar 保留这些字符。

```

mysql> create table test11(
    -> passwd  varchar(6)
    -> ,name char(4)
    -> )engine=innodb charset=utf8;
Query OK, 0 rows affected (1.33 sec)

mysql> insert into test11 values('a aa','b bb');
Query OK, 1 row affected, 1 warning (0.08 sec)

mysql> select * from test11;
+-----+-----+
| passwd | name |
+-----+-----+
| a aa | b bb |
+-----+-----+

```

3、char 和 varchar, char 比 varchar 要快, 因为 char 是定长的, 他不用去看要存多少个字节。char 会比较浪费空间。

4、如果超出了指定的字符 char 和 varchar 将进行截取将后面的省略掉。

```

mysql> insert into test11 values('1234567','12345');
Query OK, 1 row affected, 2 warnings (0.12 sec)

mysql> select * from test11;
+-----+-----+
| passwd | name |
+-----+-----+
| a aa | b bb |
| 123456 | 1234 |
+-----+-----+
2 rows in set (0.00 sec)

```

5、text 用来存储一些比较大的文本。

6、enum 枚举 1~255 个成员使用 1 个字节来进行存储, 对于 255~65535 使用 2 个字节来进行存储

enum('男','女','保密')

enum 注意:

- 1、如果插入的值超出了枚举的范围那么将插入空字符串作为特殊的错误值
- 2、当你不给枚举类型的列插入值的时候他的值为 null, 为不让他 null 所以我们要给他默认值。 default

```

mysql> create table test12(
    -> name char(32),
    -> sex enum('男','女','保密'),
    -> )engine=MyISAM charset=utf8;
Query OK, 0 rows affected (0.14 sec)

```

```
mysql> insert into test12 values ('zhangsan','男');
```

```
mysql> insert into test12 values ('lishi','未知');
```

```

mysql> select * from test12;
+-----+-----+
| name | sex |
+-----+-----+
| zhangsan | 男 |
| lishi | 未知 |
+-----+-----+

```

2、每个枚举类型都有索引:

空字符串 索引为 0 第一个枚举元素 为 1 第二个枚举元素 为 2.....

```

mysql> insert into test12 values ('zhangsan',0);
Query OK, 1 row affected, 1 warning (0.00 sec)

mysql> insert into test12 values ('lishi',1);
Query OK, 1 row affected (0.00 sec)

mysql> insert into test12 values ('wangwu',2);
Query OK, 1 row affected (0.00 sec)

mysql> insert into test12 values ('zhaoliu',3);
Query OK, 1 row affected (0.00 sec)

mysql> insert into test12 values ('liuqi',4);
Query OK, 1 row affected, 1 warning (0.00 sec)

mysql> select * from test12;
+-----+-----+
| name | sex |
+-----+-----+
| zhangsan | 男 |
| lishi | 女 |
| wangwu | 保密 |
| zhaoliu | 保密 |
| liuqi | 保密 |
+-----+-----+

```

7、set 集合类型

1~8 成员占 1 个字节
 9~16 占 2 个字节
 17~24 占 3 个字节
 25~32 占 4 个字节
 33~64 占 8 个字节

特点：超出范围的值忽略掉，重复的值只取一个。

8、日期型

建议：用整数保存时间 time();

九、创建索引

索引优化好了，能解决大概 90% 左右的访问速度问题。

索引是在数据库中用来提高搜索性能的。通常在数据库优化的时候先做就是索引优化。

索引的分类：

1、常规索引，最基本的索引没有任何限制。

```

create index 索引的名称 on 表格 (列名)
create table 表名([.....],index 索引名(列名))

```

删除索引：

```
drop index 索引名 on 表名
```

2、唯一索引，和常规索引相似，但是索引的列的值必须唯一。

```
create table 表名([.....],unique 索引名(列名))
```

```
create unique index 索引名 on 表名(列名);
```

删除索引：

```
drop index 索引名 on 表名
```

```
mysql> create table user5(
```

```
    -> id int unsigned not null auto_increment,
```

1.PRIMARY KEY (主键索引)

```
    mysql>ALTER TABLE `table_name` ADD PRIMARY KEY ( `column` )
```

2.UNIQUE(唯一索引)

```
    mysql>ALTER TABLE `table_name` ADD UNIQUE (`column` )
```

3.INDEX(普通索引)

```
    mysql>ALTER TABLE `table_name` ADD INDEX index_name ( `column` )
```

4.FULLTEXT(全文索引)

```
    mysql>ALTER TABLE `table_name` ADD FULLTEXT ( `column` )
```

5.多列索引

```
    mysql>ALTER TABLE `table_name` ADD INDEX index_name ( `column1`, `column2`, `column3a` )
```

```
-> name char(32),  
-> primary key(id)  
-> )charset = utf8;
```

3、主键索引，主键索引和唯一索引差不多，但是唯一索引可以有空格，但是主键索引不能有空格

一个表里面只能有一列有主键索引。

```
mysql> create table user4(  
-> id int unsigned auto_increment primary key not null,  
-> name char(32)  
-> )charset=utf8;
```

删除主键索引比较麻烦，如果有 auto_increment 时候要先删除后才能删除主键

4、全文本索引

英文是以空格为分界的。 全文本索引会以空格为分界进行分别索引。

但是我们中文。有什么绝对的分隔符吗？使用---斯芬克斯。

5、创建索引的一些规则：

- 1、最适合创建错用的列通常是 where 条件中的列。
- 2、索引不同的列基数越大，创建的索引效果越好。
- 3、不要过度使用索引，每个额外的索引都要占用额外的磁盘空间，降低写操作的性能。

十、存储引擎：

MySQL 的存储引擎是一个很重要的特征，用户可以根据应用的需要选择如何存储和索引数据是否支持事务等等。

事务：实例 银行转账

（男厕所--表锁 与厕所单间--行锁的关系）

表锁：资源开销小，加锁快，锁定的粒度大，发生锁冲突的概率高。

行锁：资源开销大，加锁慢，锁定粒度小，发生锁冲突的概率低。

MyISAM 表：不支持事务，优势是访问速度快，对事务的完整性没有要求或者你的表以 select, insert 为主的通常我们会选择 MyISAM 表。

InnoDB 表：提供事务处理，InnoDB 的处理效果差一些，占用更多的磁盘空间。

十一、 数据表的存储位置：

MySQL 将数据以记录形式存在中，而表中则以文件的形式存在在磁盘的一个目录中，这个目录就是一个 数据库目录。而 MySQL 每种表再该目录有不同的文件格式，但是一个共同的特点，就是每种表至少有一个存放表结构定义的.frm 文件。一个

MyISAM 数据数据表中有三个文件，他们分别是：以.FRM 为后缀的结构定义文件，以.MYD 为后缀名的数据文件，和以.MYI 为后缀名的索引文件。而 INNODB 由于采用表空间的概念来管理数据表，它只用一个与数据表对应的并以.FRM 为后缀名的文件，同在一个目录下的其他文件表示为表空间，存储数据表的数据和索引。

十二、MySQL 默认字符集

计算机只能识别二进制码，给程序，处理的数据进行编码，一套文字符号和比较规则的集合。

十三、字符集

查看所有字符集：

```
mysql> show character set;
```

使用时数据库中的 utf8 就是 UTF-8；

[乱码怎么解决：](#)

- 1、你的编辑器 utf-8 无 bom 的格式。
- 2、你需要强行设定让浏览器以 utf-8 格式去解析
- 3、PHP 和 MySQL 链接的设置要设定为 UTF-8
- 4、你的表要设置为 UTF-8

MySQL 的字符集包括：

字符集： 是用来定义 MySQL 存储字符串的方式 36

校对规则：是对规则是定义了比较字符串的方式 70

一对多的关系： 1 个字集可以对应多个校对规则

utf8_general_ci 不区分大小写，这个你在注册用户名和邮箱的时候就要使用。

utf8_general_cs 区分大小写，如果用户名和邮箱用这个 就会照成不良后果

ci 是 case insensitive，即 "大小写不敏感"，

查看校对规则与字符集的关系：

```
mysql> show collation like 'gbk%';
+-----+-----+-----+-----+-----+
| Collation      | Charset | Id | Default | Compiled | Sortlen |
+-----+-----+-----+-----+-----+
| gbk_chinese_ci | gbk     | 28 | Yes     | Yes      | 1        |
| gbk_bin         | gbk     | 87 |          | Yes      | 1        |
+-----+-----+-----+-----+-----+
```

查看服务器校对规则：

```
mysql> show variables like 'collation_server';
+-----+-----+
| Variable_name      | Value           |
+-----+-----+
| collation_server  | latin1_swedish_ci |
+-----+-----+
```

查看服务器字符集:

```
mysql> show variables like 'character_server';
```

创建字数据库符集;

```
mysql> create database demo default character set utf8 collate utf8_general_ci;
```

十四、修改表:

语法:

```
alter table 表名 动作;
```

1、删除主键索引:

1、先移除 auto_increment

2、alter table 表名 drop primary key;

```
mysql> CREATE TABLE username (
    ->     id int<10> PRIMARY KEY NOT NULL AUTO_INCREMENT,
    ->     name char<32> DEFAULT NULL,
    ->     idCode char<18> DEFAULT NULL
    -> ) ENGINE=MyISAM CHARSET=utf8;
Query OK, 0 rows affected (0.11 sec)
```

```
mysql> alter table username change id id int;
Query OK, 0 rows affected (0.23 sec)
Records: 0  Duplicates: 0  Warnings: 0
第一个id是旧id,
第二个id是新id,
这样表示没有修改!
mysql> desc username;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| id   | int<11> | NO  | PRI | 0       |          |
| name | char<32> | YES |     | NULL    |          |
| idCode | char<18> | YES |     | NULL    |          |
+-----+-----+-----+-----+-----+
```

```
mysql> alter table username drop primary key;
Query OK, 0 rows affected (0.13 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> desc username;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| id   | int<11> | NO  |     | 0       |          |
| name | char<32> | YES |     | NULL    |          |
| idCode | char<18> | YES |     | NULL    |          |
+-----+-----+-----+-----+-----+
```

3、改变列名或者列的类型

```
alter table 表名 change 旧列名 新列名 列类型;
```

```
alter table 表名 modify 列名 列类型
```

注意: change 和 modify 区别,change 可以改变列名, modify 不行。

具体:

修改字段类型:

```
mysql> alter table t1 modify sex char(3);
```

更改字段名:

```
mysql> alter table t1 change 原来字段名 新的字段名 char(3);
```

```
mysql> alter table test2 change id id int not null auto_increment primary key;
Query OK, 0 rows affected (0.87 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

更改表名: alter table 旧表名 rename as 新表名

```
mysql> alter table test2 rename as users;
Query OK, 0 rows affected (0.34 sec)
```

```
mysql> alter table t1 rename as users;
```

4、添加列

alter table 表名 add 列名 类型;

具体:

1.》添加 age 字段

```
mysql> alter table t1 add age int unsigned not null default '0';
```

2》加到最前面

```
alter table 表名 add 列名 类型 first;
```

3》加到 XXOO 后面

```
alter table 表名 add 列名 类型 after XXOO 列名;
```

具体:

在指定的字段 name 后加:

```
mysql> alter table t1 add sex int unsigned not null default '0' after name;
```

添加第一个时直接用 FIRST

```
mysql> alter table users add sex int unsigned not null default '0' after name;
Query OK, 0 rows affected (0.61 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc users;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| id   | int(11) | NO | PRI | NULL | auto_increment |
| name | char(8) | NO |     | NULL |             |
| sex  | int(10) unsigned | NO |     | 0   |             |
+-----+-----+-----+-----+-----+
```

5、删除列

alter table 表明 drop 要删除的字段名

6、auto_increment 回归到指定位置使用: alter table 表名 auto_increment=你要的值

7、清除数据

```
mysql> truncate table test12;
Query OK, 0 rows affected (0.00 sec)
```

truncate table 表名 实质: 表以前的表删掉了, 然后再又重新建了一个。

总结: 使用 create 时需要带上 table 关键字, select , insert into 需要指定表名和字段, drop 带上 from 表名。

第三篇：表数据操作

十五、增

1、插入整行数据

insert into 表名 values (值 1, 值 2, 值 3....) 建议不要使用

```
mysql> desc users;
+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key  | Default | Extra       |
+-----+-----+-----+-----+-----+
| id   | int(11) | NO   | PRI  | NULL    | auto_increment |
| name | char(8)  | NO   |      | NULL    |             |
+-----+-----+-----+-----+-----+
2 rows in set (0.01 sec)

mysql> insert into users values (null,'zhangsan');
Query OK, 1 row affected (0.11 sec)
```

注意：

- 1、每个列都要提供一个值，如果是 auto_increment 就插入 null 来代替这个字段的值
- 2、使用这种方式来插入数据必须要保证插入的值的顺序和表字段的顺序相对应。
- 3、表结构和字段顺序出现变化会使这种方式的语句产生错误。

2、插入行的一部分 (常用)

insert into 表名(字段 1, 字段 2, 字段 3...) values(值 1, 值 2, 值 3...)

```
mysql> insert into users (name) values ('lisi');
Query OK, 1 row affected (0.06 sec)
```

注意：

- 1、在字段中没有插入的字段，如果是 not null 一定要设置 default

3、插入多行

insert into 表名 (字段 1, 字段 2, 字段 3...)
values(值 1, 值 2, 值 3...),(值 4, 值 5, 值 6),()

```
mysql> insert into users (name,sex) values
-> ('sansan',1),
-> ('lilili',0),
-> ('wuwuwu',1);
Query OK, 3 rows affected (0.07 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

4、插入某些查询的结果

```
mysql> insert into users (name,sex) select name,sex from users;
Query OK, 5 rows affected (0.12 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

注意：

查询语句不带括号，查询的值与插入的值一一对应
经常用来快速插入测试数据。

十六、查

1、单列查找

select 字段名 from 表名

```
mysql> select name from users;
```

2、多列查找:

select 字段名 1,字段名 2,字段名 3..... from 表名

```
mysql> select id,name,sex from users;
```

注意：在选择多列时一定要在列名之间加上逗号，最后一列不加

3、查找所有列

select * from 表名

注意：

- 1、* 是可以查询出所有的列。
- 2、如果表的结构发生变化的时候有可能会是*返回的列的顺序发生变化。
- 3、使用*检索不需要的列通常会降低搜索和应用程序的性能。

4、查找过滤重复值的行

select distinct 列名 from 表名

```
mysql> select * from users;
+----+-----+-----+
| id | name | sex |
+----+-----+-----+
| 1  | zhangsan | 0 |
| 2  | liming | 0 |
| 3  | liming | 1 |
| 4  | lilili | 0 |
| 5  | liming | 1 |
| 6  | zhangsan | 0 |
| 7  | liming | 0 |
| 8  | sansan | 1 |
| 9  | lilili | 0 |
| 10 | wuwuwu | 1 |
+----+-----+-----+
mysql> select distinct name from users;
+-----+
| name |
+-----+
| zhangsan |
| liming |
| lilili |
| sansan |
| wuwuwu |
+-----+
```

注意：

- 1、如果使用 distinct 关键字必须直接放在列名的最前面
- 2、当使用 distinct 属性修饰多个列时，表示同时过滤多个列组成的记录。

```
mysql> select distinct name,sex from users;
+-----+-----+
| name | sex |
+-----+-----+
| zhangsan | 0 |
| liming | 0 |
| liming | 1 |
| lilili | 0 |
| sansan | 1 |
| wuwuwu | 1 |
+-----+-----+
```

5、限制结果

select 字段名 from 表名 limit m m 为你要取的数目

```
mysql> select id,name from users limit 2;
+---+-----+
| id | name |
+---+-----+
| 1 | zhangsan |
| 2 | liming |
+---+-----+
```

select 字段名 from 表名 limit n,m 从第几条开始取(第一条为 0), m 为取 m 条数

```
mysql> select id,name from users limit 2,4;
+---+-----+
| id | name |
+---+-----+
| 3 | liming |
| 4 | lilili |
| 5 | liming |
| 6 | zhangsan |
+---+-----+
```

where条件后带上字段名，
limit直接跟表名，也是
显示的条数而已

注意：如果没有足够的行，MySQL 将只能返回能返回最多的数量的行。

```
mysql> select id,name from users limit 8,5;
+---+-----+
| id | name |
+---+-----+
| 9 | lilili |
| 10 | wuuuwu |
+---+-----+
```

分页：

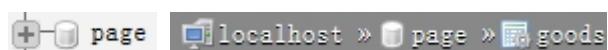
```
$everyPage = 10;//一页显示 10 条
$nowPage = 2 ;//当前页为 2
($nowPage-1)*$everyPage = 10,10 依次类推
```

6、完全限定的表名、库名（使用 . ）

select 库名.表名.字段名 from 库名.表名;

```
mysql> select database ();
+-----+
| database () |
+-----+
| lamp94 |
+-----+
1 row in set (0.00 sec)

mysql> select page.goods.price,page.goods.addtime from page.goods where id<=2;
+-----+
| price | addtime |
+-----+
| 45   | 2147483647 |
| 434  | 2147483647 |
+-----+
2 rows in set (0.00 sec)
```



7、数据排序

```
select 字段名 1, 字段名 2 from 表名 order by 字段名 [asc|desc]
```

注意：

1、order by 字段名 默认是 asc 升序 desc 是降序

2、排序的字段不一定是要显示的字段

```
select 字段名 1, 字段名 2 from 表名 order by 字段名 3 [asc,desc], 字段名 4 [asc,desc]
```

很多人认为是先对字段名 3 进行排序，然后对字段名 4 进行排序，**但实际是先对字段 3 进行排序，排序完成之后，如果字段 3 的结果有相同的值才按照字段 4 来进行排序。**

```
mysql> create table goods(
-> id int not null auto_increment primary key,
-> price decimal(10,2),
-> num int
-> )engine=innodb charset=utf8;

mysql> select * from goods;
+----+-----+---+
| id | price | num |
+----+-----+---+
| 1 | 88.00 | 100 |
| 2 | 50.00 | 60  |
| 3 | 50.00 | 100 |
| 4 | 88.00 | 60  |
| 5 | 50.00 | 100 |
+----+-----+---+

mysql> select * from goods order by price asc;
+----+-----+---+
| id | price | num |
+----+-----+---+
| 2 | 50.00 | 60 |
| 3 | 50.00 | 100 |
| 5 | 50.00 | 100 |
| 1 | 88.00 | 100 |
| 4 | 88.00 | 60 |
+----+-----+---+

mysql> select * from goods order by price asc,num desc;
+----+-----+---+
| id | price | num |
+----+-----+---+
| 3 | 50.00 | 100 |
| 5 | 50.00 | 100 |
| 2 | 50.00 | 60 |
| 1 | 88.00 | 100 |
| 4 | 88.00 | 60 |
+----+-----+---+
```

8、过滤数据

```
select 字段名.... from 表名 where 你要的条件
```

```
mysql> select * from users where id<=3;
+----+-----+---+
| id | name   | sex |
+----+-----+---+
| 1  | zhangsan | 0  |
| 2  | liming  | 0  |
| 3  | liming  | 1  |
+----+-----+---+
```

between 起始值 and 结束值

```
mysql> select * from users where id between 2 and 4;
+----+-----+---+
| id | name   | sex |
+----+-----+---+
| 2  | liming  | 0  |
| 3  | liming  | 1  |
| 4  | lilili  | 0  |
+----+-----+---+
```

如果值为字符串需要使用单引号或双引号来限定字符串

```
mysql> select * from users where name='liming';
+----+-----+---+
| id | name   | sex |
+----+-----+---+
| 2  | liming  | 0  |
| 3  | liming  | 1  |
| 5  | liming  | 1  |
| 7  | liming  | 0  |
+----+-----+---+
```

in 指定条件

```
mysql> select * from users where id in(1,2,5);
+----+-----+---+
| id | name   | sex |
+----+-----+---+
| 1  | zhangsan | 0  |
| 2  | liming  | 0  |
| 5  | liming  | 1  |
+----+-----+---+
```

not : 否定后面的操作

```
mysql> select * from users where id not between 3 and 9;
+----+-----+---+
| id | name   | sex |
+----+-----+---+
| 1  | zhangsan | 0  |
| 2  | liming  | 0  |
| 10 | wuwuwu  | 1  |
+----+-----+---+
```

is null 检查空值

注意：数据库在进行匹配条件的时不会将值为 null 的结果输出，因为数据库不知道是否匹配

is not null 检查非空值

9、组合过滤条件

》使用 and 和 or 来进行连接

注意：

- 1、组合 and 和 or 可以多次结合使用。
- 2、SQL 标准 and 的优先级高于 or 。
- 3、使用 or 可以代替 in，但是推荐使用 in，in 操作符比 or 操作符清单执行要快。

```
mysql> select * from users where sex=1 or name='liming' and id>3;
+----+-----+----+
| id | name | sex |
+----+-----+----+
| 3  | liming | 1  |
| 5  | liming | 1  |
| 7  | liming | 0  |
| 8  | sansan | 1  |
| 10 | wuwuwu | 1  |
+----+-----+----+
mysql> select * from users where (sex=1 or name='liming') and id>3;
+----+-----+----+
| id | name | sex |
+----+-----+----+
| 5  | liming | 1  |
| 7  | liming | 0  |
| 8  | sansan | 1  |
| 10 | wuwuwu | 1  |
+----+-----+----+
mysql> select * from users where id=1 or id=4;
+----+-----+----+
| id | name | sex |
+----+-----+----+
| 1  | zhangsan | 0  |
| 4  | lilili   | 0  |
+----+-----+----+
2 rows in set (0.00 sec)

mysql> select * from users where id in(1,4);
+----+-----+----+
| id | name | sex |
+----+-----+----+
| 1  | zhangsan | 0  |
| 4  | lilili   | 0  |
+----+-----+----+
```

10、通配符

LIKE 模糊查询

select * from 表名 where 字段名 like ";

_ (任意一个字符)

% (一个或多个任意字符)

```
mysql> select * from users where name like '%g';
```

同理

注意：

- 1、不要过度使用通配符，如果其他操作能够达到目的的，应该用其他的操作。
- 2、不要把通配符放在搜索的开始处。

11、拼接字段

Concat (字段 1, 字段 2)

```
mysql> select concat(id,name) from users where id<3;
+-----+
| concat(id,name) |
+-----+
| 1zhangsan      |
| 2liming        |
+-----+
```

12、别名

字段名 as 别名

```
mysql> select concat(id,name) as IdNames from users where id<3;
+-----+
| IdNames   |
+-----+
| 1zhangsan |
| 2liming   |
+-----+
```

13、select 的时候执行运算符操作

```
mysql> select num,price*0.8 as newPrice from goods where num=100;
+----+-----+
| num | newPrice |
+----+-----+
| 100 |    70.400 |
| 100 |    40.000 |
| 100 |    40.000 |
+----+-----+
```

14、数据汇总

count()确定行数

注意：

1、count(*)是对表中的所有的行进行汇总。

2、count(字段)是对字段中非空的行进行汇总。他会忽略 null

sum(字段)计算字段的和

min(字段) 取字段中最小值

max(字段) 取字段中最大值

avg(字段) 计算平均数

```
mysql> select count(*) as nums from goods;
+----+
| nums |
+----+
| 5   |
+----+
```

```
mysql> select avg(num) as numAvg from goods;
+-----+
| numAvg |
+-----+
| 84.0000 |
+-----+
```

15、对结果分组

group by 字段名

```
mysql> select count(*) as nums,sex from users group by sex;
```

nums	sex
6	0
4	1

```
mysql> select * from users;
```

id	name	sex
1	zhangsan	0
2	liming	0
3	liming	1

```
mysql> select * from users group by sex;
```

id	name	sex	默认查询最前的数据
1	zhangsan	0	
3	liming	1	

group by 字段名 having 条件

```
$sql = "select * from post where recyle=0 limit {$info['start']},{$limit}";
```

对分组后的数据进行过滤，通常我们说 where 表示的是前置过滤，如果有 group by 有 having 条件 表示的是再次过滤（对分组后的数据）。

```
mysql> select *,count(*) as nums from users group by sex having id>2;
```

id	name	sex	nums
3	liming	1	4

where

group by having

order by

limit

这些子句是有顺序的顺序是：

select 字段名 from 表名 where group by havingorder by....limit

注意：where条件

```
$sql = "select * from post where title like '%$search%' and recyle=0 limit {$info['start']},{$limit} ";
```

```
$sql = "select * from post where recyle=0 limit {$info['start']},{$limit}";
```

limit只是限制条数而已放在最后

16、多表查询（连接查询）

对多张表进行查询，可以通过连接运算符实现多个表的查询。

```
mysql> create table cats<
-> id int not null auto_increment primary key,
-> pid int,
-> name varchar(32),
-> desn text
-> >engine=innodb charset=utf8;
Query OK, 0 rows affected (0.51 sec)
```

```
mysql> create table products<
-> id int not null auto_increment primary key,
-> cid int,
-> name varchar(32),
-> price decimal(10,2),
-> desn text,
-> addtime int(10)
-> >engine=innodb charset=utf8;
Query OK, 0 rows affected (1.30 sec)
```

商品类别表：

添加数据

```
mysql> insert into cats(pid,name,desn) values (0,'soft','there are soft books');
Query OK, 1 row affected (0.09 sec)
```

```
mysql> select * from cats;
+----+----+----+-----+
| id | pid | name | desn
+----+----+----+-----+
| 1 | 0 | soft | there are soft books |
| 2 | 1 | java | this is java book |
| 3 | 1 | php | this is php book |
| 4 | 1 | C++ | this is C++ book |
| 5 | 2 | j2ee | this is j2ee book |
| 6 | 2 | j2me | this is j2me book |
| 7 | 3 | xsphp | this is xsphp book |
+----+----+----+-----+
```

商品表：

添加数据

```
mysql> insert into products(cid,name,price,desn,addtime) values (1,'java','89.45','this is xsphp book',now());
Query OK, 1 row affected, 1 warning (0.06 sec)
```

```
mysql> select * from products;
+----+----+----+----+-----+
| id | cid | name | price | desn
| addtime
+----+----+----+----+-----+
| 1 | 1 | java | 89.45 | this is xsphp book | 2147483647 |
| 2 | 1 | javaee | 89.45 | this is xsphp book | 2147483647 |
| 3 | 2 | javatwo | 89.45 | this is xsphp book | 2147483647 |
| 4 | 3 | xsphp | 89.45 | this is xsphp book | 2147483647 |
+----+----+----+----+-----+
```

思路：多表查询时只不过是再 from 后面指定多个表名，之间用逗号分隔（，）要显示的字段名，在多个表中有重复的字段名需要分别对表指定别名进而来指定字段名。如 cats 表中有 name 字段，products 表中也有，此时需要取别名进行指定或者直接用原来的表名指定。

相同字段名查询方法：

用表名

```
mysql> select pid,price,cats.name from cats,products;
```

取别名

```
mysql> select c.pid,p.price,c.name from cats as c,products as p;
```

形成笛卡尔乘积，以上的查询没有意义。----属于非等值查询

添加条件：（任意合理添加）----等值查询

```
mysql> select c.id,c.name,p.price,p.name from cats as c,products as p where c.id=p.cid and c.id=1;
```

17、多表查询（自查询）

使用同一张表利用别名原理，连接自身的查询操作。

```
mysql> desc cats;
+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra
+-----+-----+-----+-----+
| id   | int<11> | NO  | PRI | NULL    | auto_increment
| pid  | int<11> | YES |     | NULL    |
| name | varchar<32> | YES |     | NULL    |
| desn | text   | YES |     | NULL    |
+-----+-----+-----+-----+
4 rows in set <0.01 sec>

mysql> select * from cats;
+-----+-----+-----+-----+
| id | pid | name | desn
+-----+-----+-----+-----+
| 1 | 0 | soft | there are soft books
| 2 | 1 | java | this is java book
| 3 | 1 | php  | this is php book
| 4 | 1 | C++  | this is C++ book
| 5 | 2 | j2ee | this is j2ee book
| 6 | 2 | j2me | this is j2me book
| 7 | 3 | xsphp | this is xsphp book
+-----+-----+-----+-----+
```

查询条件

```
mysql> select a.id as aid,a.name as fname,b.id as bid,b.name as bname from cats as a,cats b where a.id=b.pid;
+-----+-----+-----+
| aid | fname | bid | bname
+-----+-----+-----+
| 1 | soft | 2 | java
| 1 | soft | 3 | php
| 1 | soft | 4 | C++
| 2 | java | 5 | j2ee
| 2 | java | 6 | j2me
| 3 | php  | 7 | xsphp
+-----+-----+-----+
```

别名 取别名可以省略as 连接条件

18、嵌套查询（子查询）

在一个 select 语句的 where 子句中，包含另一个 select 语句也称为子查询

1、子查询总是从内向外进行处理。

2、对于能够嵌套的数据没有限制，但是不要嵌套太多。

3、子查询一般和 in 操作符结合使用，也可以和= !=这些结合 使用

子查询的结果作为父查询的条件

查询出来的是笛卡尔积
两两乘积

```
mysql> select * from type;
+-----+-----+-----+-----+
| id | name | pid | status | path
+-----+-----+-----+-----+
| 1 | 电影点细腻 | 0 | 1 | 0
| 2 | 技术交流 | 0 | 1 | 0
| 3 | 日韩 | 1 | 1 | 0-1
| 4 | 欧美 | 1 | 1 | 0-1
| 5 | 讲座 | 2 | 1 | 0-2
| 6 | 视频交流 | 2 | 1 | 0-2
| 7 | 学习经验 | 2 | 1 | 0-2
| 8 | 连队趣事 | 0 | 1 | 0
| 23 | 种种 | 0 | 1 | 0
| 22 | 直播吧 | 0 | 1 | 0
| 20 | 好哦啊 | 8 | 1 | 0-8
| 19 | 信息你想 | 2 | 1 | 0-2
| 18 | 西欧西欧 | 1 | 1 | 0-1
+-----+-----+-----+-----+
```

后置:

```
mysql> select * from type as one,type as two where one.id = two.pid;
+----+----+----+----+----+----+----+----+
| id | name | pid | status | path | id | name | pid | status | path |
+----+----+----+----+----+----+----+----+
| 1  | 电影点细腻 | 0 | 1 | 0 | 3 | 日韩 | 1 | 1 | 1 | 0-1 |
| 1  | 电影点细腻 | 0 | 1 | 0 | 4 | 欧美 | 1 | 1 | 1 | 0-1 |
| 2  | 技术交流 | 0 | 1 | 0 | 5 | 讲座 | 2 | 1 | 1 | 0-2 |
| 2  | 技术交流 | 0 | 1 | 0 | 6 | 视频交流 | 2 | 1 | 1 | 0-2 |
| 2  | 技术交流 | 0 | 1 | 0 | 7 | 教程经验 | 2 | 1 | 1 | 0-2 |
| 8  | 连队趣事 | 0 | 1 | 0 | 20 | 好吐槽 | 8 | 1 | 1 | 0-8 |
| 2  | 技术交流 | 0 | 1 | 0 | 19 | 信息你想 | 2 | 1 | 1 | 0-2 |
| 1  | 电影点细腻 | 0 | 1 | 0 | 18 | 西欧西欧 | 1 | 1 | 1 | 0-1 |
+----+----+----+----+----+----+----+----+
8 rows in set (0.00 sec)

mysql> select * from products where cid in(select id from cats where name like 'j%');
+----+----+----+----+----+
| id | cid | name | price | desn | addtime |
+----+----+----+----+----+
| 3 | 2 | javatwo | 89.45 | this is xsphp book | 2147483647 |
+----+----+----+----+----+
1 row in set (0.02 sec)

mysql> select * from products;
+----+----+----+----+----+
| id | cid | name | price | desn | addtime |
+----+----+----+----+----+
| 1 | 1 | javajava | 89.45 | this is xsphp book | 2147483647 |
| 2 | 1 | javaee | 89.45 | this is xsphp book | 2147483647 |
| 3 | 2 | javatwo | 89.45 | this is xsphp book | 2147483647 |
| 4 | 3 | xsphp | 89.45 | this is xsphp book | 2147483647 |
+----+----+----+----+----+
4 rows in set (0.00 sec)

mysql> select id from cats where name like 'j%';
+----+
| id |
+----+
| 2 |
| 5 |
| 6 |
+----+
```

前置

```
mysql> select c.name,(select p.name from products as p where id=1) as pname from products p,cats as c;
+----+----+
| name | pname |
+----+----+
|      |      |

mysql> select p.name from products as p where id=1;
+----+
| name |
+----+
| javajava |
```

自查询

```
mysql> select * from classes;
+----+----+----+----+
| id | name | zhiwu | banji |
+----+----+----+----+
| 1 | yhj | js | 90 |
| 2 | wqg | bzs | 90 |
| 3 | gzy | xmjl | 90 |
| 4 | 1 | js | 90 |
| 5 | 2 | bzs | 90 |
| 6 | 3 | xmjl | 90 |
+----+----+----+----+
classes as a

mysql> select * from classes;
+----+----+----+----+
| id | name | zhiwu | banji |
+----+----+----+----+
| 1 | yhj | js | 90 |
| 2 | wqg | bzs | 90 |
| 3 | gzy | xmjl | 90 |
| 4 | 1 | js | 90 |
| 5 | 2 | bzs | 90 |
| 6 | 3 | xmjl | 90 |
+----+----+----+----+
classes as b

select * from classes where banji=90;
id bzs js xmjl banji
1 wqg yhj gzy 90
```

关联查询

```
mysql> select * from classss;
+----+-----+-----+-----+
| id | name | zhiwu | banji |
+----+-----+-----+-----+
| 1  | 1    | js    | 90   |
| 2  | 2    | bzs   | 90   |
| 3  | 3    | xmjl  | 90   |
+----+-----+-----+-----+
mysql> select * from names;
+----+-----+
| id | name |
+----+-----+
| 1  | yhj  |
| 2  | wqg  |
| 3  | gzy  |
+----+-----+
select a.id, b.name, a.zhiwu, a.banji from classss as a,names as b where a.name=b.id;
结果:
+----+-----+-----+-----+
| id | name | zhiwu | banji |
+----+-----+-----+-----+
| 1  | yhj  | js    | 90   |
| 2  | wqg  | bzs   | 90   |
| 3  | gzy  | xmjl  | 90   |
+----+-----+-----+-----+
```

十七、修

格式: update 表名 set 列名='值';

update 表名 set 列名='值' where 列名=值 (条件);

列名>值

列名<值 等等

注意: 修改的时候一定要加上 where 条件。

```
mysql> select * from test;
```

```
mysql> update test set name='wangwu' where id=1;
Query OK, 1 row affected (0.11 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

十八、删

格式: delete from 表名

delete from 表名 where 条件;

```
mysql> delete from test where id>=7;
Query OK, 2 rows affected (0.08 sec)
```

备注: 修改和删除操作时先查出来数据再进行操作即可。

总结: 一定要加上条件 (where)!! , 不然将修改所有的数据。

TRUNCATE TABLE 表名; 清空数据表

第四篇 MySQL 与 PHP 结合使用

十九、PHP 与数据库

- 1、mysql 最低的，最差的，适合过程化编程
- 2、mysqli 是 mysql 之后改进安全性的，但是只能连接 MySQL 数据库
- 3、PDO 性能，安全性与 mysqli 差不多，是面向对象的首选，扩展非常好，可进行跨平台使用 BD2,SQLServer,Oracle MySQL

二十、PHP 与数据使用步骤

1、连接数据库

```
header('content-type:text/html;charset=utf-8');
$link = mysql_connect('localhost', 'root', '') or die('连接数据库失败，请检测配置文件信息');
var_dump($link);
```

扩展：

```
$link = mysql_connect('example.com:3307', 'mysql_user', 'mysql_password');
if (!$link) {
    die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully';
mysql_close($link);
```

```
header('content-type:text/html;charset="utf-8"');
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password') or die('连接失败:错误号 '.mysql_errno().'--错误信息:'.mysql_error());
```

》返回 MySQL 系统的错误提示文本

```
string mysql_error ([ resource $link_identifier ] )
```

返回上一个 MySQL 函数的错误文本，如果没有出错则返回 ""（空字符串）。如果没有连接资源号，则使用上一个成功打开的连接从 MySQL 服务器提取错误信息。

```
mysql_connect("localhost", "root", "");
mysql_select_db("mysqldb");
echo mysql_errno() . ":" . mysql_error(). "<br>";
```

```
1049: Unknown database 'mysqldb'
```

》设置客户端字符集

```
bool mysql_set_charset ( string $charset [, resource $link_identifier ] )
```

```
//1_2、设置字符集
```

```
mysql_set_charset('utf8');
```

```
//设置字符集
```

```
mysql_query('set names utf8');
```

2、选择一个数据库

```
bool mysql_select_db ( string $database_name [, resource $link_identifier ] )
```

```
mysql_select_db('lamp94');
```

3、准备SQL语句

```
$sql = "select * from users";
```

一种返回结果集(select\show\desc)

一种没有返回结果集(update\insert\delete)

4、执行SQL语句

```
resource mysql_query ( string $query [, resource $link_identifier ] )
```

```
$sql = "select * from users";
$result = mysql_query($sql);
```

```
Resource id #4
```

5、处理结果集

有结果集的结果处理:

```
mysql_num_rows — 取得结果集中行的数目
```

```
int mysql_num_rows ( resource $result )
```

`mysql_num_rows()` 返回结果集中行的数目。此命令仅对 SELECT 语句有效。要取得被 INSERT , UPDATE 或者 DELETE 查询所影响到的行的数目，用 `mysql_affected_rows()`。

```
$sql = "select * from users";
$result = mysql_query($sql);
var_dump(mysql_num_rows($result));
```

```
int 14
```

```
mysql_fetch_row — 从结果集中取得一行作为
```

索引数组

array **mysql_fetch_row** (**resource \$result**)

返回根据所取得的行生成的数组，如果没有更多行则返回 FALSE。执行没有数据行时返回 FALSE。

mysql_fetch_row() 从和指定的结果标识关联的结果集中取得一行数据并作为数组返回。每个结果的列储存在一个数组的单元中，偏移量从 0 开始。

每次执行该函数时只能取出一行数据作为数组返回

下标

依次调用 **mysql_fetch_row()** 将返回结果集中的下一行，如果没有更多行则返回 FALSE。

Note: 此函数返回的字段名大小写敏感。

这样我们就可以直接使用条件型循环进行输出结果集

```
$sql = "select * from users";
$result = mysql_query($sql);
var_dump(mysql_fetch_row($result));
var_dump(mysql_fetch_row($result));
```

索引数组，多次调用，依次输出下一行数据

```
array (size=3)
  0 => string '1' (length=1)
  1 => string 'zhangsan' (length=8)
  2 => string '0' (length=1)

array (size=3)
  0 => string '2' (length=1)
  1 => string 'liming' (length=6)
  2 => string '0' (length=1)
```

mysql_fetch_assoc — 从结果集中取得一行作为关联数组

array **mysql_fetch_assoc** (**resource \$result**)

返回根据从结果集取得的行生成的关联数组，如果没有更多行则返回 FALSE。

Note: 此函数返回的字段名大小写敏感。

```
$sql = "select * from users";
$result = mysql_query($sql);
var_dump(mysql_fetch_assoc($result));
var_dump(mysql_fetch_assoc($result));
```

```

array (size=3)
  'id' => string '1' (length=1)
  'name' => string 'zhangsan' (length=8)
  'sex' => string '0' (length=1)

array (size=3)
  'id' => string '2' (length=1)
  'name' => string 'liming' (length=6)
  'sex' => string '0' (length=1)

```

`mysql_fetch_array` — 从结果集中取得一行作为关联数组，或数字数组，或二者兼有

```
array mysql_fetch_array ( resource $result [, int $result_type ] )
```

返回根据从结果集取得的行生成的数组，如果没有更多行则返回 `FALSE`。

`mysql_fetch_array()` 中可选的第二个参数 `result_type` 是一个常量，可以接受以下值： `MYSQL_ASSOC`，`MYSQL_NUM` 和 `MYSQL_BOTH`。本特性是 PHP 3.0.7 起新加的。本参数的默认值是 `MYSQL_BOTH`。

```

$sql = "select * from users";
$result = mysql_query($sql);
var_dump(mysql_fetch_array($result,MYSQL_ASSOC));
var_dump(mysql_fetch_array($result));

```

```

array (size=3)
  'id' => string '1' (length=1)
  'name' => string 'zhangsan' (length=8)
  'sex' => string '0' (length=1)

array (size=6)
  0 => string '2' (length=1)
  'id' => string '2' (length=1)
  1 => string 'liming' (length=6)
  'name' => string 'liming' (length=6)
  2 => string '0' (length=1)
  'sex' => string '0' (length=1)

```

完整操作

```

//设置浏览器解析编码
header('content-type:text/html;charset="utf-8"');
//连接数据库
$link = mysql_connect('localhost', 'root', '') or die('连接数据库失败，请检测配置文件信息');
//设置客户端字符集
mysql_set_charset('utf8');
//选择数据库
$resource = mysql_select_db('lamp94');
//准备SQL语句
$sql = "select * from users where id<=3";
//执行SQL语句
$result = mysql_query($sql);
//处理结果集
while($array = mysql_fetch_assoc($result)){
    echo $array['id'].'----'.$array['name'].'
}

```

没有结果集的结果处理:

mysql_affected_rows — 取得前一次 MySQL 操作所影响的记录行数

```
int mysql_affected_rows ([ resource $link_identifier ]) 
```

取得最近一次与 *Link_identifier* 关联的 **INSERT**, **UPDATE** 或 **DELETE** 查询所影响的记录行数。

```
$link = mysql_connect('localhost', 'root', '') or die('连接数据库失败, 请检测配置文件信息');
mysql_set_charset('utf8');
$resource = mysql_select_db('lamp94');
$sql = " update users set name='heheh' where id<=3";
$result = mysql_query($sql);
//echo mysql_affected_rows($result);---error
echo mysql_affected_rows($link).'  
';//最近的资源
echo mysql_affected_rows();//默认最近的资源
```

3

3

mysql_insert_id — 取得上一步 INSERT 操作产生的 ID

```
int mysql_insert_id ([ resource $link_identifier ]) 
```

mysql_insert_id() 返回给定的 *Link_identifier* 中上一步 **INSERT** 查询中产生的 **AUTO_INCREMENT** 的 ID 号, 如果没有指定 *Link_identifier*, 则使用上一个打开的连接。

如果上一查询没有产生 **AUTO_INCREMENT** 的值, 则 **mysql_insert_id()** 返回 0。如果需要保存该值以后使用, 要确保在产生了值的查询之后立即调用 **mysql_insert_id()**。

```
$sql = " insert into users (name) values ('ninini')";
$result = mysql_query($sql);
echo mysql_insert_id($link).'  
';
echo mysql_insert_id();

mysql> select id,name from users limit 14,10;
+----+-----+
| id | name  |
+----+-----+
| 15 | ninini |
+----+-----+
```

15	15
----	----

```
header('content-type:text/html;charset="utf-8"');
//连接数据库
$link = mysql_connect('localhost', 'root', '') or die('连接数据库失败, 请检测配置文件信息');
mysql_set_charset('utf8');
$resource = mysql_select_db('lamp94');
$sql = " insert into users (name) values ('wowowo')";
$result = mysql_query($sql);
if(mysql_affected_rows()>0){
    echo '插入成功,插入的最后ID号'.mysql_insert_id();
}else{
    echo '插入失败';
}
```

插入成功，插入的最后ID号17

```
mysql> select id,name from users limit 15,10;
+---+-----+
| id | name  |
+---+-----+
| 16 | wowow |
| 17 | wowow |
+---+-----+
```

6、关闭数据库

mysql_close — 关闭 MySQL 连接

```
bool mysql_close ([ resource $link_identifier ] )
```

mysql_close() 关闭指定的连接标识所关联的到 MySQL 服务器的非持久连接。

通常不需要使用 **mysql_close()**，因为已打开的非持久连接会在脚本执行完毕后自动关闭。

```
mysql_close($link);
```

二十二、排错思路

排错思路：

- 1、先注意提示。
- 2、断点调试。 echo 'dddd'; exit () ;
- 3、输出调试。 echo "aaa"; var_dump(); echo " " ;

从错误提示的行开始，先查看上一行和下一行再看该行是否错误，然后逐行向上利用断点，输出进行调试代码。建议写代码的时候写几行运行看看 **(var_dump())** 代码执行出来是不是你想要的代码。

二十三、补充知识点（在添加时间时候用）

date — 格式化一个本地时间 / 日期

```
string date ( string $format [, int $timestamp ] )
```

返回将整数 timestamp (时间戳) 按照给定的格式字串而产生的字符串。如果没有给出时间戳则使用本地当前时间 (time())。换句话说，timestamp 是可选的参数，默认值为 time()。

```
echo date("Y-m-d H:i:s",time()).'<br`>';
echo date("Y-m-d H:i:s");
```

2014-10-30 14:46:46

2014-10-30 14:46:46

time — 返回当前的 Unix 时间戳

返回自从 Unix 纪元 (格林威治时间 1970 年 1 月 1 日 00:00:00) 到当前时间的秒数。

```
| | | // 7 days; 24 hours; 60 mins; 60secs  
$nextWeek = time() + (7 * 24 * 60 * 60);| 时间戳是秒数  
echo 'Now:'. date('Y-m-d') ."  
echo 'Next Week: '. date('Y-m-d', $nextWeek);
```

Now:2014-10-30

Next Week: 2014-11-06

可以把 **date()** 和 [mktime\(\)](#) 结合使用来得到未来或过去的日期。

mktime — 取得一个日期的 Unix 时间戳

```
int mktime ([ int $hour [, int $minute [, int $second [, int $month [, int $day [, int $year  
echo date("M-d-Y", mktime(0, 0, 0, 12, 32, 1997));
```

Jan-01-1998

```
echo mktime(0, 0, 0, 12, 32, 1997);
```

883612800 时间戳

项目时间处理:

添加时间入库时使用本地时间 **time()**, 在输出显示使用 **date()** 进行转化再设置时区即可

添加页面:

 doAddArticle.php

```
$addtime = time(); //入库时间为time()的时间戳
```

显示页面:

 viewArticle.php

```
date_default_timezone_set("PRC");
```

```
"创建时间:".date("Y-m-d H:i:s",$addtime);
```

为什么在添加时间时不直接设置时区添加时区的对应时间戳而是直接添加**time()**?
因为时你在中国的项目突然要用到外国上线,这样原本数据库中在中国添加的时间
就要重新进行修改了,造成不必要的麻烦,所以我们就统一添加返回自从 Unix 纪元
(格林威治时间 1970 年 1 月 1 日 00:00:00) 到当前时间(**php**系统时间)的秒数。
所以我们将时间的显示方式(**date_default_timezone_set**)一般设置在项目配置文件
统一使用即可,修改也方便。

二十四、获取字符串函数

多字节使用函数（中文，一个中文占 3 个字节）

mb_substr — Get part of string

```
string mb_substr ( string $str , int $start [, int $length [, string $encoding ]] )
```

```
.mb_substr($cons['content'],0,200,'utf-8')
```

单字节使用函数（英文，数字）

substr — 返回字符串的子串

```
string substr ( string $string , int $start [, int $length ] )
```

返回字符串 *string* 由 *start* 和 *length* 参数指定的子字符串。

二十五、多个页面进行传值（a 链接，\$_SESSION（用户跟踪），数据库中取）隐藏域

A 链接：

进行传值的页面：

 index.php

```
$sql = "select id,title,ctime,mtime,content,rcount from article";
$result = mysql_query($sql);
$cons = mysql_fetch_assoc($result)

<a href='./doDeleteArticle.php?id={$cons['id']}>删除</a>
```

使用 a 链接鼠标经过的时候可以查看状态栏（浏览器左下角）

进行接受值再进一步处理的页面：

 doDeleteArticle.php

```
//使用$_GET接收值,a 链接也是通过get传值,可以查看地址栏中的参数和对应的值.
$articleId = $_GET['id'];
$sql = "delete from article where id={$articleId}";
mysql_query($sql);
```

数据库中：（在一个页面中）



viewArticle.php

```
$sql = "select * from article where id={$articleId}";  
|  
$result = mysql_query($sql);  
|  
$contentDetail = mysql_fetch_assoc($result);
```

打开文件表示已经阅读一次，直接显示+1 次：

```
阅读次数:'.($contentDetail['rCount']+1)
```

进行修改数据库中的数据，再查看已经修改数据：

```
//修改阅读次数  
$sql = "update article set rCount=rCount+1 where id={$articleId}";  
mysql_query($sql);  
mysql_close();
```