# 关于数据库的知识，网站www.w3school.com.cn

# 基本操作

# 启动：net start mysql

停止：net stop mysql

链接：mysql -uroot -p 密码：

查看数据表：show databases;

创建数据表：create database db\_name;

例如：create database student;

切换到某个数据库：use db\_name;

在数据库下定义表：create table name<属性>

例如：create table product <id int, name varchar(50), squence tinyint>;

删除数据库：drop datebase db\_name

删除表：drop table table\_name

查看表结构desc table\_name

查看表：select \* from 表名

重命名：rename table old\_name to new\_name

# 数据操作

# 增加数据，insert into

1.标准方式

Insert into 表名 （字段的列表）values（数据列表）

2.使用set;

Insert into 表名 set 字段 1=值 1，......

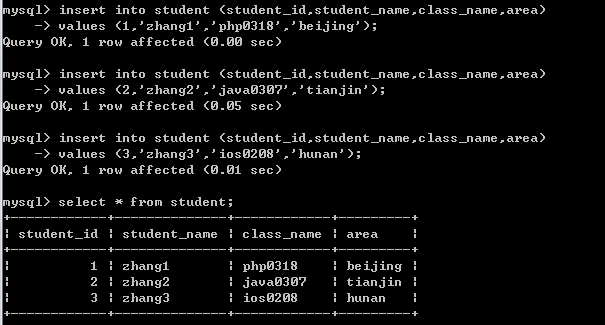
3.从其他数据表取数据插入：

Insert into 表名 （字段列表）select

## 标准方式

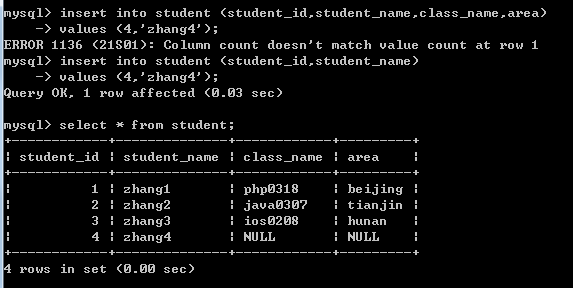
如果写了字段列表，则数据要与其一一对应，而且顺序也必须一致！另外数据类型也要对应，字符串型，日期型都要加引号。

指定所有字段

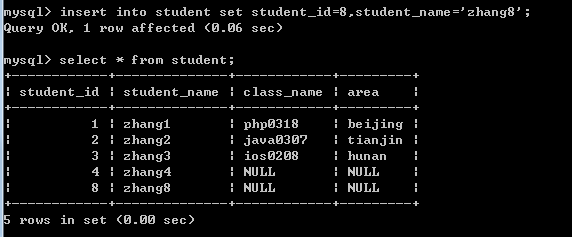


我们可以只插入部分字段给值，其余字段数据库会用默认值，前提是定义数据表时，此字段有默认值。

指定部分字段



## 使用set语法

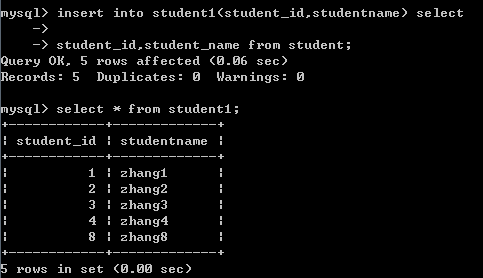


## 使用select从其他表调出数据

如果其他表结构或者与之对应的字段类型完全一致，这是可以写成

Insert into 表名 select \* from 其他表 where ..order by ..limit..

如果其他表字段并不一致，可以取部分字段的内容插入，但要求类型要对应。



# 替换数据，replace into

和insert into一样也是分三种，标准写法，set语法，从其他表取数据

1.标准方式

replace into 表名 （字段的列表）values（数据列表）

2.使用set;

replace into 表名 set 字段 1=值 1，......

3.从其他数据表取数据插入：

replace into 表名 （字段列表）select

和insert into的区别：

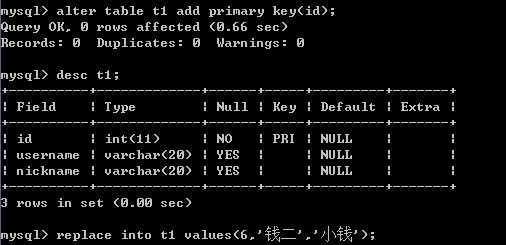
Insert into是直接插入，replace into是先判断表中的主键或唯一索引字段是否和插入数据相应字段一样的数据，它就会变成修改！（它会先删除旧数据，然后再写入一条新数据）

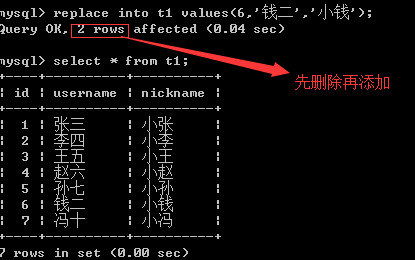
如果没有一样的数据，它就会插入一条新数据。

如果准备插入数据的表没有任何主键或唯一索引，则replace into会直接插入新数据，它就不判断了！

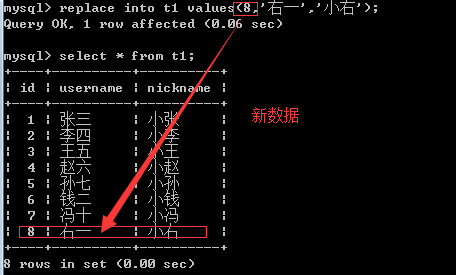
替换的例子







插入新数据的例子



影响多条数据的例子

当和主键以及唯一索引重复，例如 id 为主键，username为唯一索引，若输入:

replace into t1 values(6,'右一','小中');

则会替换第六条和第八条数据。

发现有两个重复记录，一个和id为6的记录，还有一个是username为新插入值的那条id为8的数据，它会先删除重复的两条数据，再插入一条新数据。因此影响到了3条行记录。

注：Replace into在数据有重复的时候，它不等同于update！

# 查询数据，select

格式：select \*或字段列表 from 表名 where 条件

查询是千变万化的。

查询所有数据

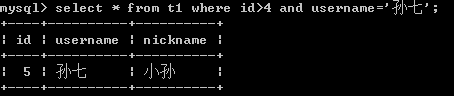
Select \* from t1;

查询部分数据，加条件

Select \* from t1 where id>4;

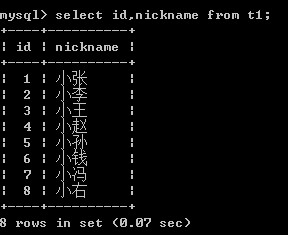


select \* from t1 where id>4 and username='孙七';



只查询部分字段

select id,nickname from t1;



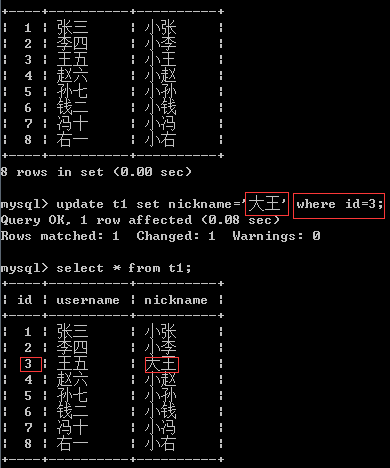
# 修改数据，update

格式：update 表名 set 字段名=新值，......where 条件 order by..limit

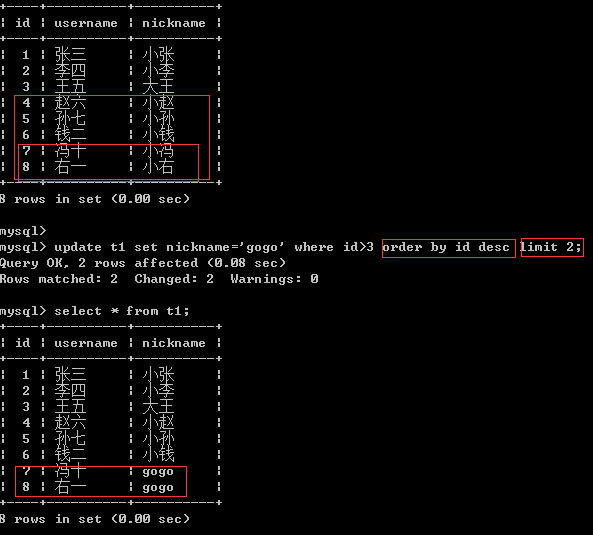
思考一下：如果不加where条件会出现什么结果呢？

结果是，把数据表里所有的记录都进行了更新！

update t1 set nickname='大王' where id=3;



update t1 set nickname='gogo' where id>3 order by id desc limit 2;



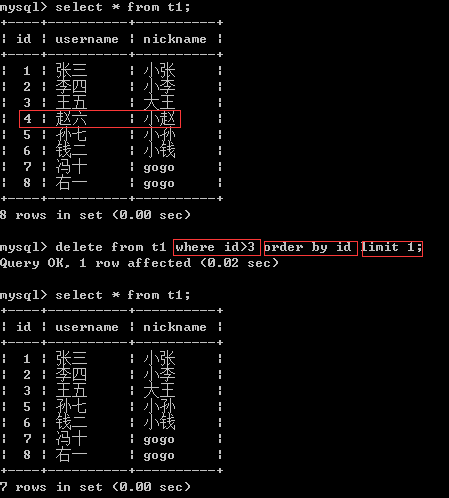
可以加where order by limit

# 删除数据，delete

格式：delete from 表名 where..order by..limit..

如果你不是想把表中所有的数据都删除，一定要记得写where条件！

可以加更多条件



# 快速删除所有数据，truncate

格式：truncate 表名

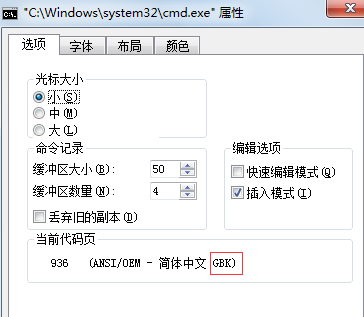
等同于，delete from 表名；

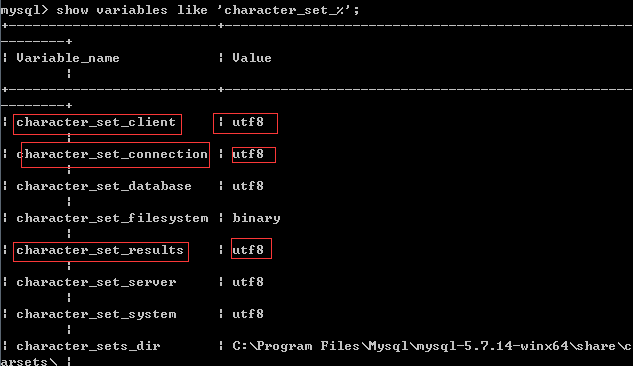
区别：1，truncate是直接把表删除，然后再重建表结构。优点是速度快。比一行一行删除快。

2，不会返回受影响的行数。

3，auto increment自动增量的字段恢复初值。

# 客户端的字符集





Character\_set\_client 客户端使用的编码

Character\_set\_connection 数据库连接使用的编码

Character\_set\_results 返回结果使用的编码

这三个变量的设置影响到数据是否是乱码

数据库的数据显示是乱码，怎么做？

方法是：设置character\_set\_results为客户端正确的编码

Set character\_set\_results=gbk;

我们进行数据的插入，会不会有问题呢？



原因是什么？

就是客户端明明是gbk 的编码，现在却告诉数据库编码是utf8

Character\_set\_clicent是utf8！

解决方法：设置客户端编码变量为正确的值

Set character\_set\_client=gbk;

结论：因此如果客户端的这两个变量设置正确，那么无论插入数据还是读取数据，都不会乱码！

Set names gbk; set names utf8; 可以改变上述三个变量。

create table t2

(

id int unsigned auto\_increment primary key,

username varchar(20) not null unique key,

nickname varchar(20) character set utf8

) default character set gbk;

客户端不会出现乱码情况，原因是数据库会根据客户端的编码进行正确的转换。

总结：

1. 首先确认我们客户端是什么编码。命令行客户端看属性。PHP文件保存编码。
2. 根据客户端编码，用set names gbk或utf8设置编码

就不会有乱码发生了！

|  |  |
| --- | --- |
| 操作符 | 描述 |
| = | 等于 |
| <> | 不等于 |
| > | 大于 |
| < | 小于 |
| >= | 大于等于 |
| <= | 小于等于 |
| BETWEEN | 在某个范围内 |
| LIKE | 搜索某种模式 |

注释：在某些版本的 SQL 中，操作符 <> 可以写为 !=。

**引号的使用**

请注意，我们在例子中的条件值周围使用的是单引号。

SQL 使用单引号来环绕文本值（大部分数据库系统也接受双引号）。如果是数值，请不要使用引号。

文本值：

这是正确的：

SELECT \* FROM Persons WHERE FirstName='Bush'

这是错误的：

SELECT \* FROM Persons WHERE FirstName=Bush

数值：

这是正确的：

SELECT \* FROM Persons WHERE Year>1965

这是错误的：

SELECT \* FROM Persons WHERE Year>'1965'

ORDER BY语句用于对结果集进行排序。

ORDER BY 语句

ORDER BY 语句用于根据指定的列对结果集进行排序。

ORDER BY 语句默认按照升序对记录进行排序。

如果您希望按照降序对记录进行排序，可以使用 DESC 关键字。

原始的表 (用在例子中的)：

Orders 表:

|  |  |
| --- | --- |
| Company | OrderNumber |
| IBM | 3532 |
| W3School | 2356 |
| Apple | 4698 |
| W3School | 6953 |

实例 1

以字母顺序显示公司名称：

SELECT Company, OrderNumber FROM Orders ORDER BY Company

结果：

|  |  |
| --- | --- |
| Company | OrderNumber |
| Apple | 4698 |
| IBM | 3532 |
| W3School | 6953 |
| W3School | 2356 |

实例 2

以字母顺序显示公司名称（Company），并以数字顺序显示顺序号（OrderNumber）：

SELECT Company, OrderNumber FROM Orders ORDER BY Company, OrderNumber

结果：

|  |  |
| --- | --- |
| Company | OrderNumber |
| Apple | 4698 |
| IBM | 3532 |
| W3School | 2356 |
| W3School | 6953 |

实例 3

以逆字母顺序显示公司名称：

SELECT Company, OrderNumber FROM Orders ORDER BY Company DESC

结果：

|  |  |
| --- | --- |
| Company | OrderNumber |
| W3School | 6953 |
| W3School | 2356 |
| IBM | 3532 |
| Apple | 4698 |

实例 4

以逆字母顺序显示公司名称，并以数字顺序显示顺序号：

SELECT Company, OrderNumber FROM Orders ORDER BY Company DESC, OrderNumber ASC

结果：

|  |  |
| --- | --- |
| Company | OrderNumber |
| W3School | 2356 |
| W3School | 6953 |
| IBM | 3532 |
| Apple | 4698 |

注意：在以上的结果中有两个相等的公司名称 (W3School)。只有这一次，在第一列中有相同的值时，第二列是以升序排列的。如果第一列中有些值为 nulls 时，情况也是这样的。

## INSERT INTO 语句

INSERT INTO 语句用于向表格中插入新的行。

语法

INSERT INTO 表名称 VALUES (值1, 值2,....)

我们也可以指定所要插入数据的列：

INSERT INTO table\_name (列1, 列2,...) VALUES (值1, 值2,....)

"Persons" 表：

|  |  |  |  |
| --- | --- | --- | --- |
| LastName | FirstName | Address | City |
| Carter | Thomas | Changan Street | Beijing |

INSERT INTO Persons VALUES ('Gates', 'Bill', 'Xuanwumen 10', 'Beijing')

结果：

|  |  |  |  |
| --- | --- | --- | --- |
| LastName | FirstName | Address | City |
| Carter | Thomas | Changan Street | Beijing |
| Gates | Bill | Xuanwumen 10 | Beijing |

在指定的列中插入数据

"Persons" 表：

|  |  |  |  |
| --- | --- | --- | --- |
| LastName | FirstName | Address | City |
| Carter | Thomas | Changan Street | Beijing |
| Gates | Bill | Xuanwumen 10 | Beijing |

INSERT INTO Persons (LastName, Address) VALUES ('Wilson', 'Champs-Elysees')

结果：

|  |  |  |  |
| --- | --- | --- | --- |
| LastName | FirstName | Address | City |
| Carter | Thomas | Changan Street | Beijing |
| Gates | Bill | Xuanwumen 10 | Beijing |
| Wilson |  | Champs-Elysees |  |

## Update 语句

Update 语句用于修改表中的数据。

语法：

UPDATE 表名称 SET 列名称 = 新值 WHERE 列名称 = 某值

Person:

|  |  |  |  |
| --- | --- | --- | --- |
| LastName | FirstName | Address | City |
| Gates | Bill | Xuanwumen 10 | Beijing |
| Wilson |  | Champs-Elysees |  |

更新某一行中的一个列

我们为 lastname 是 "Wilson" 的人添加 firstname：

UPDATE Person SET FirstName = 'Fred' WHERE LastName = 'Wilson'

结果：

|  |  |  |  |
| --- | --- | --- | --- |
| LastName | FirstName | Address | City |
| Gates | Bill | Xuanwumen 10 | Beijing |
| Wilson | Fred | Champs-Elysees |  |

更新某一行中的若干列

我们会修改地址（address），并添加城市名称（city）：

UPDATE Person SET Address = 'Zhongshan 23', City = 'Nanjing'

WHERE LastName = 'Wilson'

结果：

|  |  |  |  |
| --- | --- | --- | --- |
| LastName | FirstName | Address | City |
| Gates | Bill | Xuanwumen 10 | Beijing |
| Wilson | Fred | Zhongshan 23 | Nanjing |

## DELETE 语句

DELETE 语句用于删除表中的行。

语法

DELETE FROM 表名称 WHERE 列名称 = 值

Person:

|  |  |  |  |
| --- | --- | --- | --- |
| LastName | FirstName | Address | City |
| Gates | Bill | Xuanwumen 10 | Beijing |
| Wilson | Fred | Zhongshan 23 | Nanjing |

删除某行

"Fred Wilson" 会被删除：

DELETE FROM Person WHERE LastName = 'Wilson'

结果:

|  |  |  |  |
| --- | --- | --- | --- |
| LastName | FirstName | Address | City |
| Gates | Bill | Xuanwumen 10 | Beijing |

删除所有行

可以在不删除表的情况下删除所有的行。这意味着表的结构、属性和索引都是完整的：

DELETE FROM table\_name

或者：

DELETE \* FROM table\_name