## mysql\_unbuffered\_query()

mysql\_query()函数查询的方式是将结果全部查询出来，缓存到内存当中，但当结果集非常庞大时，就会超内存。

mysql\_unbuffered\_query()不会缓存结果集，而是查询出来数据后立马对结果集进行操作，边查询边返回。不会有内存的限制；

但是，使用mysql\_unbufferde\_query()时不能使用mysql\_num\_rows(),且向MySQL发送一条新的SQL查询之前，必须提取掉所有未缓存的 SQL 查询所产生的结果行。

try{  
 $pdo = new PDO('mysql:host=localhost;dbname=test','root','root',array(PDO::*ATTR\_ERRMODE*=>PDO::*ERRMODE\_EXCEPTION*)) ;  
 $pdo->query('set names utf8');  
 $pdo->setAttribute(PDO::*MYSQL\_ATTR\_USE\_BUFFERED\_QUERY*,false);  
 $statement = $pdo->query("select goods\_name from goods limit 20");  
 echo '内存大小：'.*memory\_get\_usage*().'<br>';  
  
 while($row = $statement->fetch(PDO::*FETCH\_ASSOC*)){  
 *print\_r*($row);  
 echo "<br>";  
 echo '内存大小；'.*memory\_get\_usage*().'<br>';  
 }  
}catch(Exception $e){  
 echo $e->getMessage();  
}

//注意：dbname=后面不能有空格，否则读取不到数据库

//memory\_get\_usage():返回php脚本所占的内存



内存占用率很低。

## sql语句查询表的字段名

$sql = “desc users”;// desc 表名

$res = mysql\_query($sql);

$row = mysql\_fetch\_assoc($res)

echo $row[‘Field’];//Field必须大写，得到字段名

## 特殊的sql语句

IF：

例：update `group` set `group\_count`=if(`group\_count`>1,`group\_count`-1,0) where `id`=1;

解释：if($parm1,$parm2,$parm3)

$parm1是条件，若$parm1为真，则执行$parm2，假则执行$parm3.

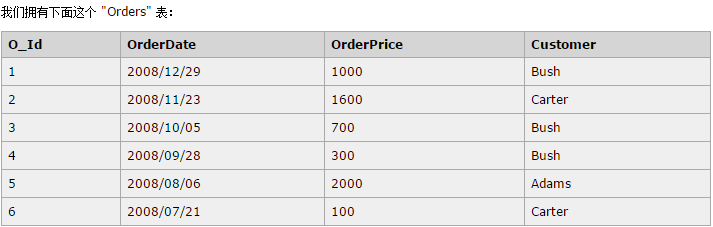
IFNULL(exp1,exp2)

若exp1为null，返回exp2的值；否则还是返回exp1。

Order by：order by 多个字段，每个字段后面都有排序方式，默认ASC

例如：select table a order by a.time1 ,a.time2 desc,a.time3 asc

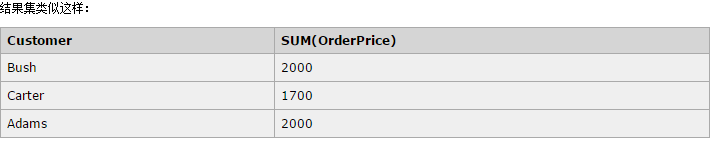
Group by:



现在，我们希望查找每个客户的总金额（总订单）。

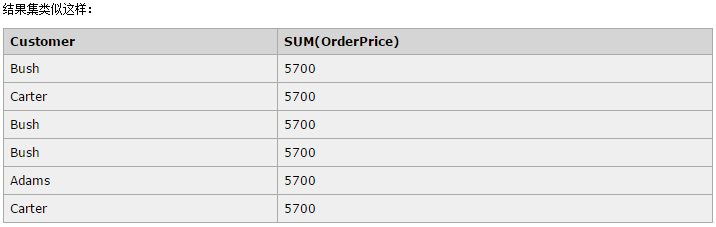
我们使用下列 SQL 语句：

SELECT Customer,SUM(OrderPrice) FROM Orders GROUP BY Customer



当省略 group by时：

SELECT Customer,SUM(OrderPrice) FROM Orders



show table status from 数据库 like 表名；

包含信息：

array (

'Name' => 'message',

'Engine' => 'InnoDB',

'Version' => '10',

'Row\_format' => 'Compact',

'Rows' => '6355',

'Avg\_row\_length' => '41',

'Data\_length' => '262144',

'Max\_data\_length' => '0',

'Index\_length' => '0',

'Data\_free' => '7340032',

'Auto\_increment' => '6384',

'Create\_time' => '2017-05-12 09:20:34',

'Update\_time' => NULL,

'Check\_time' => NULL,

'Collation' => 'utf8\_general\_ci',

'Checksum' => NULL,

'Create\_options' => '',

'Comment' => '',

)

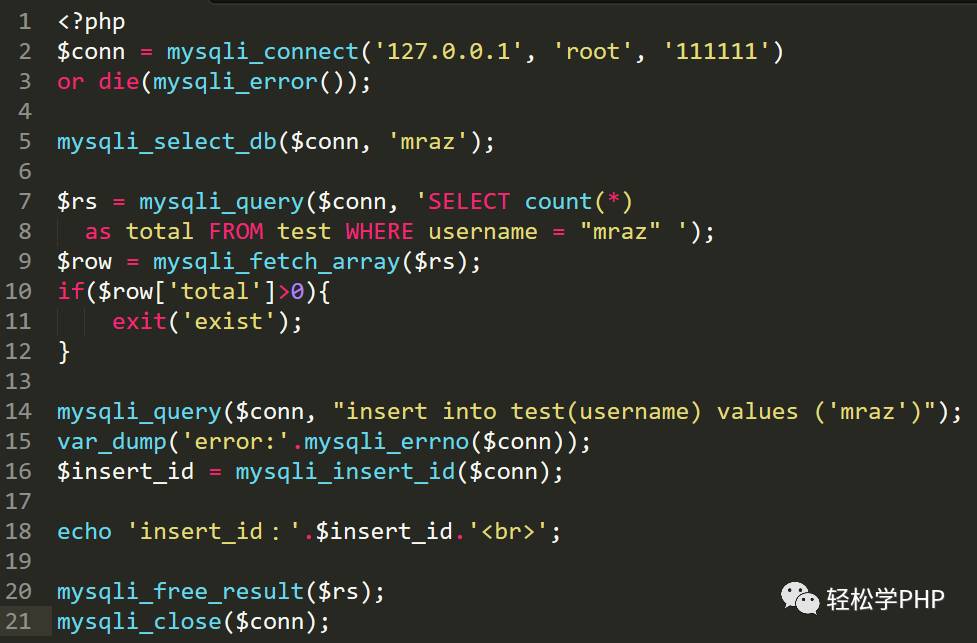
## SQL语句反引号``和单引号’’的作用

反引号`` ：**为了区分MYSQL的保留字与普通字符而引入的符号。**

举个例子：SELECT `select` FROM `test` WHERE select='字段值'

单引号’’： **字段值是字符或字符串，则要加引号**

## 高并发请求，解决插入多条重复记录



少量请求不会有问题，但是一旦高并发请求时，会插入多条username为mraz的记录；

解决：

利用Mysql的 FOR UPDATE语句和事务的隔离性，且FOR UPDATE仅适用于innoDB，必须在事物中才生效；



## Mysql锁机制

以MyISAM引擎为例：

有两种模式的表级锁：表共享读锁，表独占写锁。

对MyISAM表的读操作，不会阻塞其他用户对同一表的读请求，但会阻塞对同一表的写请求；

对MyISAM表的写操作，则会阻塞其他用户对同一表的读和写操作；

加锁：

MyISAM在执行查询语句（SELECT）前，会自动给涉及的所有表加读锁，在执行更新操作（UPDATE、DELETE、INSERT等）前，会自动给涉及的表加写锁，这个过程并不需要用户干预

行级锁以InnoDB引擎为例：

InnoDB的行级锁定同样分为两种类型，共享锁和排他锁，而在锁定机制的实现过程中为了让行级锁定和表级锁定共存，InnoDB也同样使用了意向锁（表级锁定）的概念，也就有了意向共享锁和意向排他锁这两种。



意向锁是InnoDB自动加的，不需用户干预。对于UPDATE、DELETE和INSERT语句，InnoDB会自动给涉及数据集加排他锁（X)；对于普通SELECT语句，InnoDB不会加任何锁；事务可以通过以下语句显示给记录集加共享锁或排他锁。

共享锁（S）：SELECT \* FROM table\_name WHERE ... LOCK IN SHARE MODE

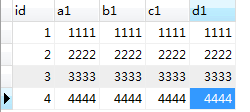
排他锁（X)：SELECT \* FROM table\_name WHERE ... FOR UPDATE

InnoDB行锁是通过给索引上的索引项加锁来实现的，只有通过索引条件检索数据，InnoDB才使用行级锁，否则，InnoDB将使用表锁

## Mysql存在物理外键的表如何清空

|  |
| --- |
| 1. **SET** FOREIGN\_KEY\_CHECKS = 0; 2. **TRUNCATE** **TABLE** TABLE\_E; 3. **SET** FOREIGN\_KEY\_CHECKS = 1; |

## 将一个数据表的数据导出到另个表，且字段名不相同

Test1: 

Test2: 

|  |
| --- |
| $sql = "insert into yd\_test2 (`a2`,`b2`,`c2`,`d2`) select `a1`,`b1`,`c1`,`d1` from yd\_test1"; |

将test1中的a1,b1,c1,d1字段，对应test2表中a2,b2,c2,d2分别导入，使用一下sql语句：