MySQL知识点总结

. 数据操作：检索、排序、过滤、分组、汇总、计算、联接、子查询与组合查询

. 表操作：表的创建、修改、删除和重命名、表数据的插入、更新和删除

. 索引（含主、外键） 、视图

. 难度编程：存储过程与存储函数、触发器与事件、 PHP

. 数据库管理：事务处理、用户与权限、备份与还原、数据库维护

1. 检索数据： select , from ,

   Select [distinct] prod_id,prod_name from products [limit 4,5];

2. 检索排序： order by ,

   Select * from products order by prod_id [asc|desc],prod_name [asc|desc];

3. 过滤数据： where 字句

   = <> != >>= <<= between

（1）普通 where 字句

Select prod_id,prod_name from products where prod_name= ’liyang ’;

Select prod_id,prod_name from products where prod_id between 10 and 50;

Select prod_id,prod_name from products where prod_name is [not] null;

（2）组合 where 字句：使用 AND和 OR连接多个条件表达式，且 AND次序优于 OR;

（3） IN 与 NOT操作符

Select prod_id,prod_name from products where prod_id [not] in(1,2,3) |prod_name in(’zhangsan’ , ’lisi ’ , ’wangwu’);

（4） LIKE 操作符与通配符： “ %” 与“ _”

Select prod_id,prod_name from products where prod_name like ’%liu% ’;

Select prod_id,prod_name from products where prod_name like ’_u%’； 找出 u 位于名字的第二个位置的 prod_id 和 prod_name。

（5）正则表达式

4. 计算字段

（1）拼接字段： concat(, , ,)

Select concat(姓氏 , 名字) as 姓名 from orders;

Select concat(vend_name, ’(’,vend_country, ’) ’) from vendors;

（2）算术运算

Select prod_name,prod_price,prod_num,prod_price*prod_num as prod_money from products;

4. 使用数据处理函数： 文本处理函数、日期和时间处理函数、数值处理函数。

5. 汇总数据： 聚集函数 SUM() AVG() COUNT() MAX() MIN()

Select avg(distinct prod_price) from products;

Select avg(prod_price) 均价 ,max(prod_price) 最高价 from products;

6. 分组数据： group by , 创建分组、过滤分组、分组排序

Select count(prod_id),prod_id from products where prod_id>1000 group by prod_id having count(prod_id)>2 order by prod_id; 求出 prod_id 大于 1000 且产品数量大于 2 的产品数量，并按 prod_id 排序，注意分组语句中对象要前后一致，如下划线部分。

7. 使用子查询： 进行过滤 select , where，in(select , where，in(select ,)) 、作为计算字段使用子查询。

8. 联接： join ， on,

（1）普通联接

Select oi.order_num,oi.prod_id,p.prod_name,p.vend_id,v.vend_name from orderitems oi join products p on oi.prod_id=p.prod_id join vendors v on p.vend_id=v.vend_id where vend_name= ’liyang ’；可同时联接多个表且可同时用于数据过滤，这种类型的联接一般为内部联接。

（2）自联接：一个表自己与自己联接，注意判断好各字段与前后两个表的关系。

（3）自然联接：基本上简历的内部联接都是自然联接。

（4）外部联接： 在关系表中没有关联的信息的行也能显示出来的联接， 根据表在 join 字句的左边还是右边可分为左联接与右联接。

（5）带聚集函数的联接

Select c.cust_id,count(o.order_num) num_ord from customers c join orders o on c.cust_id=o.cust_id order by c.cust_id; 找出客户对应的订单数。

9. 组合查询： 连接多个（至少两个）查询语句，满足其中一个查询语句条件的结果都会显示出来 union （不重复显示） /union all （可重复显示即全部显示）

Select vend_id,prod_id,prod_price from products where prod_price<=5

Union [all]

Select vend_id,prod_id,prod_price from products where vend_id in(1001,1002)

order by prod_id;

注意每个查询必须包含相同的列、表达式或者聚集函数，列的数据类型必须兼容，排序语句只能放在最后面，该排序语句对组合查询语句中的所有 select 语句都适用。

10. 全文本搜索： 只支持引擎为 MyISAM的表，不支持引擎为 InnoDB 的表，可对搜索结果进行智能排序后输出，具有较高等级的行先返回。

Match(全文本搜索字段) against （’全文本搜索内容’ [with query expansion] ）其中下划线部分为拓展语句， 使用该语句， 除了可以返回符合所设置的 “ 全文本搜索内容 ” 的数据结果，还可返回与 “ 全文本搜索内容 ” 有较高相似度的数据结果。

（1）启用全文本搜索支持

Create table fs(id int not null primary key,c text,c1 text,fulltext(c,c1)) engine=MyISAM;

（2）进行全文本搜索

Select note_text from productnotes where match(note_text) against(’liyang ’ with query expansion);

11. 插入数据： insert into ， {values|select} ，

Insert into products(prod_id,prod_name,prod_price) values(1, ’豆浆’ ,2),(3, ’鸡蛋’ ,1); 可同时插入多行数据。

Insert into products(prod_id,prod_name,prod_price) select vend_id,vend_name, vend_price from vendors where vend_id<=10;

12. 更新数据： update [ignore] ， set ， ,一般情况下，若更新的数据中有部分数据出错，则全部数据返回到原来的数据，而 ignore 的作用在于即使更新的数据中出现错误，只对出现错误的数据返回到原来数据，而未出现错误的数据返回更新后的结果实现更新。

update products set prod_name=' 馒头',prod_price=1 where prod_id=1;

update customers set cust_city=concat(cust_city, ’市’)| cust_city =replace(cust_city, ’市’ ,’city ’) where cust_id>1000;

13. 删除数据： delete from ，

Delete from products where prod_id between 10 an 50;

14. 表的相关操作

（1）创建表：对表结构进行设置 create table ，

Create table products(prod_id int null auto_increment primary key,prod_name varchar(50),prod_price int,prod_city varchar(50) default ’广州’) engine= InnoDB; 每个字段名后需要设置数据类型， default 为指定默认值， 只支持常量不支持函数，且只在插入数据时起作用而在更新数据时不起作用， InnoDB 是一个可靠的事务处理引擎，但不支持全文本搜索。

（2）更新表：对表结构进行修改 alter table {add|drop} ，

Alter table products add prod_city varchar(50) ；

Alter table products drop prod_price;

（3）删除表：一旦删除，无法撤销 drop table ，

Drop table products;

（4）重命名表： rename table ， to ，

Rename table products to new_products;

15. 索引的相关操作

（1）创建索引：常用于数据的快速检索， MySQL中，常用索引在物理可分为： BTREE、HASH 索引两类； 在具体用途上可分为： INDEX、UNIQUE、PRIMARYKEY、FOREIGNKEY、FULLTEXT、SPATIAL 等。

①使用 create index 语句创建索引，对已存在的表创建索引

Create [unique|fulltext|spatial] index index_name [using BTREE|HASH] on tbl_name(index_col_name[,index_col_name ，]);

Create unique index index_products on products(prod_name(2) desc,prod_price);

②使用 create table 语句创建索引，创建表的同时创建索引

Create table seller(seller_id int not null auto_increment,seller_name char(50),seller_adress char(50),seller_contact char(50),product_type int,sales int,primary key(seller_id,product_type),[unique|fulltext|spatial] index index_seller(sales));

③使用 alter table 语句创建索引，修改表的同时添加索引

Alter table tbl_name add {[unique|fulltext|spatial] index index_tbl_name(字段名)|primary key （字段名） |foreign key （字段名） references elsetbl_name （相同字段名） };

（2）查看索引 :Show index from tbl_name [where expr];

（3）删除索引： drop index index_nameon tbl_name 语句或 alter table 语句

Drop index index_name on tbl_name;

Alter table tbl_name drop {[unique|fulltext|spatial] index index_tbl_name(字段名)|primary key （字段名） |foreign key （字段名） references elsetbl_name （相同字段名 }; （下划线部分不确定）

16. 视图的相关操作

视图：虚拟的表，视图本身不包含表中的列和数据，它包含只是一个 SQL查询 , 常用于检索数据。 * 视图的作用与规则。

（1）创建视图： Create view view_name as select , [where ,];

Create view view_products as select prod_id,prod_name,prod_price,prod_num, prod_price*prod_num as prod_money from products where prod_id<=10 [with check option];-- 下划线部分表示今后对该视图数据的修改都必须符合 prod_id<=10

（2）查看视图（用法同表） : select * from view_name;

（3）删除视图： drop view view_name;

17. 完整性： 实体完整性（主键与候选键） 、参照完整性（主键与外键） 、用户定义的完整性（非空约束与 check 约束）。

18. 创建主键约束： create table 语句或 alter table 语句

Create table products(prod_id int not null auto_increment primary key,c int);

作为列的主键约束；

Create table products(prod_id int not null auto_increment,c int,c1 int,primary key(prod_id)); 作为表的主键约束，且复合主键职能用这种形式创建

Alter table products add primary key(prod_id);

备注：实体完整性通过主键约束与候选键约束来实现，候选键约束的创建类似主键约束的创建，实质上同索引。

19. 设置表外键： create table 语句或 alter table 语句 , 外键中列的数目和数据类型必须与被参照表的主键中列的数目和对应数据类型一致。

alter table tbl_name add [constraint fk_name] foreign key(,) references,

Create table products(prod_id int not null auto_increment,c int,c1 int,foreign key(prod_id) references customers(prod_id));

alter table products add constraint fk_products_cust foreign key(cust_id) references cust(cust_id);

20. 存储过程： 为了以后的使用而保存的一条或多条 SQL语句的集合

-- 建立存储过程：建立一个可通过输入 item_id ，输出对应订单总金额的存储过程

->Delimiter // -- 改变分割符为 //

->create procedure ordertotal(in o_id int,out o_total decimal(10,2))

过程名字输入参数及类型输出参数及类型

->begin

->select sum(item_price*item_num) from orderitems where item_id=o_id into o_total;

->if o_total is null then

->select ’不存在该订单号’ ;

->end if;

->end;

->//

-- 执行存储过程：当 item_id=200005 时，得出对应订单总金额

->delimiter ; -- 将分割符改回分号

->call ordertotal(200005,@total); -- 由于不存在输出参数， 故定义一个输出变量，变量必须用 @开头

->select @total;

返回结果为 149.87

备注：书本第十一章后的编程题，使用 update 语句，两个参数类型都需要为 in 。

-- 显示存储过程

->Show create procedure ordertotal;

-- 删除存储过程

->Drop procedure ordertotal;

21. 存储函数

存储函数与存储过程的区别：

. 存储函数不能拥有输出参数；

. 存储函数可直接调用， 且不需使用 call 语句，而存储过程的调用必须使用 call 语句；

. 存储函数中必须包含一条 return 语句，而这条特殊的 SQL语句不允许包含于存储过程。

-- 建立存储函数：根据给定的 cust_id 返回客户所在的州名（缩写） ，若库中无给定的 cust_id ，则返回“不存在该客户” 。

->delimiter //

->create function fn_search(c_id int)

->returns varchar(50) -- 定义返回的数据类型，与函数部分中的数据类型需统一，如函数中的“不存在该客户”为 6 个字符，如果这里设置为 char(5) ，则无法输出该结果

->deterministic – 表示对于相同的输入值，返回值也相同

->begin

->declare state char(2); -- 声明一个变量 state ，作为输出的州变量

->select cust_state from customers where cust_id=c_id into state;

->if state is null then

->return(select ’不存在该客户’); -- 注意这里 return 不用加 s

->else

->return(select state);

->end if;

->end;

->//

-- 执行存储函数

->select fn_search(10001);

-- 删除存储函数

->drop function fn_search; -- 删除前要确定该函数无依赖关系，即不存在其他存储过程或存储函数调用过该存储函数。

22. 触发器： MySQL响应 insert 、 delete 、 update 语句时自动执行的一条 MySQL语句，创建触发器时需要给出的 4 条信息： 唯一的触发器名、 触发器相关的表、 触发器应该响应的活动（ insert delete 、 update ）、触发器何时执行（处理前或处理后） 。

（1） insert 触发器：当对表插入数据时起作用，含有一个虚拟表 New，可访问增加的行，只能用 after

-- 建立一个 insert 触发器，用于记录 insert 语句操作时的系统时间和插入的 order_num

->delimiter //

->create trigger trg_order_insert after insert on orders for each row

触发器 触发器名 执行时间 相关表

->begin

->insert into order_log(o_date,order_num) values(now(),new.order_num); -- order_log 是事先建立好的表， 用于记录 insert 语句操作时的系统时间和插入的 order_num

->end;

->//

-- 执行 insert 触发器

->delimiter ;

->insert into orders(order_date,cust_id) values(’2010-9-15 ’,10001);-- 由于 order_num 是自动递增的，故在这里不作为插入对象

（2）delete 触发器：当对表删除数据时起作用， 含有一个虚拟表 Old，可访问被删除的行，只能用 after ，创建方法与 insert 类似，区别在于 delete 和 old

-- 建立一个 delete 触发器，用于记录 delete 语句操作时的系统时间和删除的 order_num

->delimiter //

->create trigger trg_order_delete after delete on orders for each row

触发器 触发器名 执行时间 相关表

->begin

->insert into order_log(o_date,order_num) values(now(),old.order_num); -- order_log 是事先建立好的表， 用于记录 delete 语句操作时的系统时间和删除的 order_num

->end;

->//

-- 执行 delete 触发器

->delimiter ;

->delete from orders where order_num=20010;

（3）update 触发器：当对表修改数据时起作用， 同时含有 new 和 old 两个虚拟表。结合 New 可访问更新行的记录；结合 old 可访问更新前行的记录，可用 after ，也可用 before 。

①用 after

-- 建立一个 update 触发器，用于记录 update 语句操作时的系统时间和更新数据的 order_num

->delimiter //

->create trigger trg_order_update after update on orders for each row

触发器 触发器名 执行时间 相关表

->begin

->insert into order_log(o_date,order_num) values(now(),old.order_num);

->end;

->//

-- 执行 update 触发器

->delimiter ;

->update orders set order_date= ’2015-9-18 ’ where cust_id=10001;

②用 before

-- 建立一个 update 触发器，如果更新后的 prod_price 大于原来的 1.2 倍，则用原来的 1.2 倍作为当前价格

->delimiter //

->create trigger trg_order_update before update on orders for each row

触发器 触发器名 执行时间 相关表

->begin

->if new.prod_price>old.prod_price*1.2 then

->set new.prod_price=old.prod_price*1.2;

->end if;

->end;

->//

（4）删除触发器： drop trigger trg_name;

23. 事件： 临时触发器，要使用事件调度器，必须开启“ event_scheduler ”

. 查看： show variables like ’event_scheduler ’;

. 开启： set global event_scheduler=1;

（1）创建事件

CREATE EVENT EVENT_NAME ON SCHEDULE schedule

DO

event_body;

其中 schedule 的语法格式为

AT timestamp [+INTERVAL interval] ， |every interval -- 指定事件执行的时间，可以为某时刻点即 timestamp ，或某时刻点开始的 interval 时间后，或者为每隔 interval 时间执行一次

[starts timestamp [+INTERVAL interval]] -- 设置事件开始执行的时间

[ends timestamp [+INTERVAL interval]] -- 设置事件终止执行的时间

-- 建立一个事件，用于每个月向 customers 表中插入一条数据 “ liyang 、广州 ”，该事件从下个月开始并于 2015-12-31 结束

->delimiter //

->create event event_insert on schedule every 1 month

->starts curdate()+interval 1 month

->ends ’2015-12-31 ’

->do

->begin

->if year(curdate())<2015 then

->insert into customers(cust_name,cust_adress) values(’liyang ’, ’广州’);

->end if;

->end;

->//

（2）修改事件，用于修改时间的状态 :alter event event_name{enable|disable};

（3）删除事件 :drop event event_name;

24. 管理实务处理： start transaction ，

实务处理的术语：

（1）实务（ transaction ）：一组 SQL语句；

（2）回退 (rollback): 撤销指定 SQL语句的过程 ；

（3）提交 (commit) ：指定未存储的 SQL语句结果写入到数据库表里，提交后无法回退 ；

（4）保留点 (savepoint) ：实务处理中设置的临时占位符。

25. 安全管理 （用户创建修改与删除以及用户权限的查看设置与撤销）

（1）创建用户账号： create user ben identified by ’ben’;

（2）修改用户账号： update mysql.user set user= ’new_ben’ where user= ’ben’;

-- 从 mysql 数据库中的用户表 user 进行修改

（3）查看访问权限： show grants for new_ben;

（4）设置访问权限 :grant ， to ,

.grant {all|select,update,delete,insert}on {*.*|crashcourse.*|crashcourse.cus
tomers} to new_ben;
.grant select (cust_id,cust_name) on crashcourse.customers to new_ben;
-- 可针对{整个服务器 |整个数据库 |数据库中某个表 |数据库中某个表的某些字段 }, 对用户同时设置全部或一种或多种权限
（5）撤销访问权限： revoke， from，，用法与 grant， to, 类似
（6）更改口令（密码）
Set password for new_ben=password(’new_ben’);
（7）删除用户： drop user new_ben;
26. 数据库备份与还原
. 使用 SQL语句
backup table tbl_name to ， /restore table tbl_name from ， （只用于 MyISAM表）
select ， intooutfile ， /load data ， infile ， into table tlb_name
. 使用命令行实用程序 :mysqlhotcopy （只用于 MyISAM表）或 mysqldump/mysql
（1）使用 select ， intooutfile ， /load data ， infile ， into table tlb_name
. 备份数据：
Select * from mysql.products into outfile ’d:\products.txt ’
[Fields terminated by ’，’
optionally enclosed by ’”’
lines terminated by ’\n\r ’; -- 定义字段间的分割符、字符型数据的存放形式、行与行之间的分割符
. 恢复数据
Load data infile ’d:\products.txt ’into table customers.copy
[Fields terminated by ’，’
optionally enclosed by ’”’
lines terminated by ’\n\r ’; -- 必须与备份时一致
（2）使用命令行实用程序 mysqldump/mysql （文本形式）
进入 cmd运行界面 (mysqldump —help 可用于获取 mysqldump 的选项表及更多帮助信息)
. 备份整个数据库服务器、或整个数据库或数据库中某个表
Mysqldump －u root －proot －P 3306 －h localhost {all-databases|mysql_test [products]}>d:\data.sql
. 恢复数据
Mysql －u root －proot －P 3306 －h localhost {all-databases|mysql_test [products]}<d:\data.sql
27. 数据库维护语句 （可同时对一个或多个表进行操作）
（1） analyze table tbl_name; 更新表的索引散列程度，检查表键是否正确
（2） check table tbl_name; 检查一个或多个表是否有错误
（3） checksum table tbl_name; 对数据库中的表进行校验，保证数据的一致性
（4） optimize table tbl_name; 利用表中未使用的空间并整理数据文件碎片，保证数据读取效率
（5） repair table tbl_name; 修复一个或多个可能被损害的 MyISAM表
28. 二进制日志文件的使用： mysqlbinlog

29. 使用 PHP进行 MySQL数据库编程

编程步骤：

. 首先建立与 MySQL数据库服务器的连接；

. 然后选择要对其进行操作的数据库；

. 再执行相应的数据库操作，包括对数据的添加、删除、修改和查询等；

. 最后关闭与 MySQL数据库服务器的连接。

（1）数据库服务器连接、选择数据库

. 使用 mysql_connect （） 建立非持久连接

```
<? Php
    $con=mysql_connect( “ localhost:3306 ”, “ root ”, “ 123456 ”);
    if(!$con)
    {
    echo “ 数据库服务器连接失败！ <br> ”;
    die();
    }
    echo “ 数据库服务器连接成功！ <br> ”;
?> //  将 connect.php 部署在已开启的 WAMP平台环境中，并在浏览器地址中输入
“ http://localhost/connect.php ”
```

. 使用 mysql_pconnect （） 建立持久连接

```
<?php
$con=mysql_pconnect( “ localhost:3306 ”, “ root ”, “ 123456 ”);
 if(!$con)
 {
 die( “ 数据库服务器连接失败！ ” .mysql_error());// 终止程序运行，并返回错误信息
 }
echo “ MySQL服务器： localhost:3306<br> ”;
 echo “ 用户名： root<br> ”;
 echo “ 使用函数 mysql_pconnect （）永久连接数据库。 <br> ”;
?>
```

. 使用 mysql_select_db(databases[,connection]) 选择数据库

```
<?php
 $con=mysql_connect( “ localhost:3306 ”, “ root ”, “ 123456 ”);
 if(!$con)// 或者为 if(mysql_errno())
 {
  echo “ 数据库服务器连接失败！ <br> ”;
 die();
 }
mysql_select_db( “ mysql_test ”,$con);
 if( mysql_errno() )
 {
 echo “ 数据库选择失败！ <br> ”;
```

```
 die();
 }
 echo  “ 数据库选择成功！  <br> ”
?>
```

（2）数据的添加、更新和删除操作，mysql_query(SQL 语句 [,connection])，insert 、update 、delete 语句可置于函数 mysql_query （）中从而实现数据的添加、更新和删除操作

. 数据的添加

```
/* 向数据库 mysql_test 中的表 customers 添加一个名为 “ 李中华 ” 的客户的全部信息 */
<?php
$con=mysql_connect( “ localhost:3306 ”, “ root ”, “ 123456 ”) or die( “ 数据库服务器连接失败！ <br> ”);
Mysql_select_db( “ mysql_test ”,$con) or die( “ 数据库选择失败！ <br> ”);
Mysql_query( “ set names ’ gbk ’ ”); // 设置中文字符集
$sql= “ insert into customers( ’ cust_id ’, ’ cust_name ’, ’ cust_sex ’) ”;
$sql=$sql. “ values(null, ’ 李中华 ’ , ’ M’) ”;
if( mysql_query($sql,$con) )
echo “ 客户信息添加成功！ <br> ”;
else
echo “ 客户信息添加失败！ <br> ”;
?>
```

. 数据的更新

```
/* 将数据库 mysql_test 的表 customers 中的一个名为 “ 李中华 ” 的客户的地址修改为 “ 广州市 ” */
<?php
$con=mysql_connect( “ localhost:3306 ”, “ root ”, “ 123456 ”) or die( “ 数据库服务器连接失败！ <br> ”);
Mysql_select_db( “ mysql_test ”,$con) or die( “ 数据库选择失败！ <br> ”);
Mysql_query( “ set names ’ gbk ’ ”);
$sql= “ update customers set cust_address= ’ 广州市 ’ ” ;
$sql=$sql. “ where cust_name= ’ 李中华 ’ ” ;
if(mysql_query($sql,$con))
echo “ 客户地址修改成功！ <br> ”;
else
echo “ 客户地址修改失败 !<br> ”;
?>
```

. 数据的删除

```
/* 将数据库 mysql_test 的表 customers 中一个名为 “ 李中华 ” 的客户信息删除 */
<?php
$con=mysql_connect( “ localhost:3306 ”, “ root ”, “ 123456 ”) or die( “ 数据库服务器连接失败 <br> ”);
Mysql_select_db( “ mysql_test ”,$con) or die( “ 数据库选择失败！ <br> ”);
Mysql_query( “ set names ’ gbk ’ ”);
```

```
$sql= “ delete from customers ”;
$sql=$sql. “ where cust_name= ’李中华’ ” ;
if(mysql_query($sql,$con))
echo(“ 客户信息删除成功！ <br> ”);
else
echo(“ 客户信息删除失败！ <br> ”);
?>
```

（3）数据库的查询

. 使用 mysql_fetch_array(data[,array_type]) 读取结果集中的记录

```
/* 在数据库 mysql_test 的表 customers 中查询 cust_id 为 916 的客户的姓名 */
<?php
$con=mysql_connect( “ localhost:3306 ”, “ root ”, “ 123456 ”) or die( “ 数据库服务器连接失败！ <br> ”);
Mysql_select_db( “ mysql_test ”,$con) or die( “ 数据库选择失败！ <br> ”);
Mysql_query( “ set names ’gbk’ ”);
$sql= “ select cust_name from customers ”;
$sql=$sql. “ where cust_id=916 ”;
$result=mysql_query($sql,$con);
if($result)
{
echo “ 客户查询成功！ <br> ”;
$array=mysql_fetch_array($result,MYSQL_NUM);
if($array)
{
echo “ 读取到客户信息！ <br> ”;
echo “ 所要查询客户的姓名为： ” .$array[0]. “ <br> ”;
}
else
echo “ 未读取到客户信息！ <br> ”;
}
else
echo “ 客户查询失败！ <br> ”;
?>
```

. 使用 mysql_num_rows(data) 读取结果集中的记录数

```
/* 在数据库 mysql_test 的表 customers 中查询女性客户的人数 */
<?php
$con=mysql_connect( “ localhost:3306 ”, “ root ”, “ 123456 ”) or die( “ 数据库服务器连接失败！ <br> ”);
Mysql_select_db( “ mysql_test ”,$con) or die( “ 数据库选择失败！ <br> ”);
Mysql_query( “ set names ’gbk’ ”);
$sql= “ select * from customers ”;
$sql=$sql. “ where cust_sex= ’F’ ”;
```

```
$result=mysql_query($sql,$con);
if($result)
{
echo “查询成功！ <br>”;
$num=mysql_num_rows($result);//      如果结果为空，则为    0 行
echo “所要查询的女性客户人数为：      ”.$num.“位 <br>”;
}
else
echo “查询失败！ <br>”;
?>
```