# 1、修改数据库密码（不同数据库不一样）

## 1、postgresql

### ALTER USER postgres with password '123456';

## 2、MySQL

### set password for root@localhost=password('147094');

### 3、oracle

### alter usr healerjean identified by 147094

# 2、having

### 数据库中where与having区别

having 和where 都是用来筛选用的   
having 是筛选组 而where是筛选记录  
1》当分组筛选的时候 用having  
2》其它情况用where  
-----------------------------------------------------  
用having就一定要和group by连用，  
用group by不一有having （它只是一个筛选条件用的）

<http://blog.csdn.net/yangyi22/article/details/7483859>

### 举例

**一、给定一张学生成绩表tb\_grade：学号，姓名，科目号，科目名，分数。（如下表）**

（select \* from tb\_grade order by Sno,Cno;）  
+------+-------+-----+-------+-------+  
| Sno  | Sname | Cno | Cname | score |  
+------+-------+-----+-------+-------+  
| 1001 | 李菲  |   1 | 语文  |    86 |  
| 1001 | 李菲  |   2 | 数学  |    56 |  
| 1001 | 李菲  |   3 | 物理  |    48 |  
| 1001 | 李菲  |   4 | 化学  |    90 |  
| 1001 | 李菲  |   5 | 英语  |    57 |  
| 1002 | 王琪  |   1 | 语文  |    90 |  
| 1004 | 和树  |   3 | 物理  |    67 |  
| 1004 | 和树  |   4 | 化学  |    30 |  
| 1004 | 和树  |   5 | 英语  |    68 |  
+------+-------+-----+-------+-------+

 1、查询不及格科目数大于等于2的学生学号和不及格科目数量：

[**MySQL**](http://lib.csdn.net/base/mysql)> select Sno,count(score) as '不及格科目数' from tb\_grade where score<60 group by Sno having count(score)>=2;

+------+--------------+

# 3、distinct

如何用distinct消除重复记录的同时又能选取多个字段值？

需求是：我要消除name字段值重复的记录，同时又要得到id字段的值，其中id是自增字段。   
select distinct name from t1 能消除重复记录，但只能取一个字段，现在要同时取id,name这2个字段的值。   
select distinct id,name from t1 可以取多个字段，但只能消除这2个字段值全部相同的记录   
最后解决方法：   
SELECT *id*,name FROM t1 WHERE id IN(SELECT MAX(id) FROM t1 GROUP BY name) order by *id* desc

注意开头的 *id* 的 一定要，后面的order by 里有的字段一定要加进select 结果，要不然排序无效

# 4. SELECT嵌套

嵌套SELECT语句也叫子查询，一个 SELECT 语句的查询结果能够作为另一个语句的输入值。子查询不但能够出现在Where子句中，也能够出现在from子句中，作为一个临时表使用，也能够出现在select list中，作为一个字段值来返回。

## 1、单行子查询 ：单行子查询是指子查询的返回结果只有一行数据。当主查询语句的条件语句中引用子查询结果时可用单行比较符号（＝, >, <, >=, <=, <>）来进行比较。

例：  
select ename,deptno,sal  
from emp  
where deptno=(select deptno from dept where loc='NEW YORK')；  
  
2、多行子查询：多行子查询即是子查询的返回结果是多行数据。当主查询语句的条件语句中引用子查询结果时必须用多行比较符号（IN，ALL,ANY）来进行比较。其中，IN的含义是匹配子查询结果中的任一个值即可（"IN" 操作符，能够测试某个值是否在一个列表中），ALL则必须要符合子查询的所有值才可，ANY要符合子查询结果的任何一个值即可。而且须注意ALL 和ANY 操作符不能单独使用，而只能与单行比较符（=、>、< 、>= 、<= 、<>）结合使用。

例：

## 2、.多行子查询使用IN操作符号例子：查询选修了老师名叫Rona(假设唯一)的学生名字

sql> select stName

from Student

where stId in(selectdistinct stId from score where teId=(select teId from teacher where teName='Rona'));

   查询所有部门编号为A的资料：

SELECT ename,job,sal

FROM EMP

WHERE deptno in ( SELECT deptno FROM dept WHERE dname LIKE 'A%')；  
2）.多行子查询使用ALL操作符号例子：查询有一门以上的成绩高于Kaka的最高成绩的学生的名字:

sql> select stName

from Student

where stId in(select distinct stId from score where score >all(select score from score where stId=(select stId from Student where stName= 'Kaka') ));

## 4、FROM (SELECT内联视图子查询，里面的表相当于外面的

例：  
(1)SELECT ename,job,sal,rownum  
FROM (SELECT ename,job,sal FROM EMP ORDER BY sal)；  
(2)SELECT ename,job,sal,rownum  
FROM ( SELECT ename,job,sal FROM EMP ORDER BY sal)  
WHERE rownum<=5；  
SELECT \* FROM  ( SELECT \* FROM  id\_type  );    
SELECT \* FROM id\_type t;   相同结果

第一个查了两次 第二个一次  一个有子查询

## 5、在HAVING子句中使用子查询

例：  
SELECT deptno,job,AVG(sal) FROM EMP GROUP BY deptno,job HAVING AVG(sal)>(SELECT sal FROM EMP WHERE ename='MARTIN')；

让我们再看看一些具体的实例，

# 5、直接根据字段导入另一张表

INSERT INTO "public"."tb\_student"(id,created\_on,email,mobile,pwd,real\_name,source,updated\_on) SELECT tn\_account.uuid\_accid,create\_date,email

,mobile,pwd,real\_name,source,modify\_date from tn\_account;

# 6、[select into和 inseret into select 两种表复制语句](http://www.cnblogs.com/freshman0216/archive/2008/08/15/1268316.html)

<http://www.cnblogs.com/freshman0216/archive/2008/08/15/1268316.html>

## 1、Insert into

语句形式为：Insert into Table2(field1,field2,...) select value1,value2,... from Table1

      要求目标表Table2必须存在，由于目标表Table2已经存在，所以我们除了插入源表Table1的字段外，还可以插入常量。示例如下：

### 举例：Insert into Table2(a, c, d) select a,c,5 from Table1

## 2、select into

SELECT vale1, value2 into Table2 from Table1

      要求目标表Table2不存在，因为在插入时会自动创建表Table2，并将Table1中指定字段数据复制到Table2中。示例如下：

举例select a,c INTO Table2 from Table1

select ename,sal,job into pname,psal,pjob from emp where empno = eno;

# 7. declare 定义数据类型 （MySQL有oracle没有）

declare v\_number number(10);  
 v\_char varchar2(20);

## 解释：这个一般是用在函数和存储过程中的.

# 8. loop 循环

create or replace procedure pro\_test\_loop is  
i number;  
begin  
i:=0;  
loop  
  i:=i+1;  
  dbms\_output.put\_line(i);  
  if i>5 then  
    exit;  
  end if;  
end loop;  
end pro\_test\_loop;

# 9、函数

## 解释:用法

### 1、in表示传入，out表示输出

### 2、变量带取值范围，后面接分号RESULT\_VALUE:=INFO;

### 3、下面as和is区别

在存储过程(PROCEDURE)和函数(FUNCTION)中没有区别；http://blog.itpub.net/kindeditor/plugins/emoticons/images/20.gif  
在视图(VIEW)中只能用AS不能用IS；http://blog.itpub.net/kindeditor/plugins/emoticons/images/13.gif  
在游标(CURSOR)中只能用IS不能用AS。http://blog.itpub.net/kindeditor/plugins/emoticons/images/12.gif

## 1、编写大概的函数

|  |
| --- |
| create or replace function FNC\_IDNO\_CONVERT(INFO IN VARCHAR2) RETURN VARCHAR2 IS  BEGIN  //这里将来要写数据  return RESULT\_VALUE;  END FNC\_IDNO\_CONVERT; |

## 2、定义要用到的变量

|  |
| --- |
| create or replace function FNC\_IDNO\_CONVERT(INFO IN VARCHAR2) RETURN VARCHAR2 IS  RESULT\_VALUE varchar2(18);  i number;  nosum number;  temp number;  oneNum number;  indexno number;  fin number;  idLength number;  BEGIN  /\*这里将来要写数据\*/  return RESULT\_VALUE;  END FNC\_IDNO\_CONVERT; |

## 3、取得传入的参数进行赋值

|  |
| --- |
| create or replace function FNC\_IDNO\_CONVERT(INFO IN VARCHAR2) RETURN VARCHAR2 IS  RESULT\_VALUE varchar2(18);  i number;  nosum number;  temp number;  oneNum number;  indexno number;  fin number;  idLength number;  BEGIN  idLength:=length(INFO); //取得上面传入的的参数值  RESULT\_VALUE:=INFO;  /\*这里将来要写数据\*/  return RESULT\_VALUE;  END FNC\_IDNO\_CONVERT; |

## 4、使用各种条件判断和循环

### 1、if then end if

|  |
| --- |
| BEGIN  idLength:=length(INFO); //取得上面传入的的参数值  RESULT\_VALUE:=INFO;  /\*这里将来要写数据\*/  if idLength=18 then  RESULT\_VALUE:=substr(INFO, 0, 6) || substr(INFO, 9, 9);  end if;  if fin<10 then  begin  RESULT\_VALUE:=RESULT\_VALUE || to\_char(fin);  end;  end if;    2. if intip<0 **then** 3. **temp**:=intip-min\_int; 4. flag:=0; 5. **else** 6. **temp**:=intip; 7. flag:=1; 8. **end** if;   return RESULT\_VALUE; |

### 2、while loop begin 内容end; end loop;

|  |
| --- |
| i:=1;  nosum:=0;  while i<18 loop  begin  oneNum:=to\_number(substr(RESULT\_VALUE,i,1));  temp:=power(2,18-i);  temp:=mod(temp,11);  nosum:=nosum+oneNum\*temp;  i:=i+1;  end;  end loop; |

### 3、for循环

#### 1、数值型

|  |
| --- |
| --数值型For循环  procedure loop\_num\_for  (  lowest in number,  highest in number  )  is  begin  FOR even\_number in lowest .. highest --升序  // for i in (select one from table) loop  类似于这样的  loop  --处理非平滑增长的索引  if mod(even\_number,**2**)=**0**  then  dbms\_output.put\_line('now number:' || even\_number);  end if;  end loop;  end loop\_num\_for; |

#### 2、游标型

|  |
| --- |
| procedure loop\_cursor\_for  is  begin  declare cursor userinfo\_cur is select \* from userinfo\_table;  begin  FOR userinfo\_rec in userinfo\_cur  loop  dbms\_output.put\_line('username is:' || userinfo\_rec.user\_name);  end loop;  end;  end loop\_cursor\_for; |

# 10、存储过程

## 1、查询员工信息

|  |
| --- |
| create or replace procedure queryempinfo  (  eno in number,  pname out varchar2,  psal out number,  pjob out varchar2  )  as  begin  select ename,sal,job into pname,psal,pjob from emp where empno = eno;  end; |

# 11、执行一次刷新一次,产生随机数

## 1、|| L为拼接

## 2、lpad left左面不足填 0，一共6个

### select to\_char(sysdate,'yyyymmdd')||lpad(Ref\_NO\_SEQUENCE.Nextval,6,'0') from dual



# 12、dblink, oracle跨用户进行查找(个人理解，跨库查询)

## 1、创建dblink方法

### 1.sql创建直接执行（没有测试过，不能保证成功）

|  |
| --- |
| create database link TRUSTEE\_LINK  connect to PENSION\_OCPT  using '(DESCRIPTION =  (ADDRESS\_LIST =  (ADDRESS = (PROTOCOL = TCP)(HOST = 10.3.231.166)(PORT = 1521))  )  (CONNECT\_DATA =  (SERVER = DEDICATED)  (SERVICE\_NAME = orcl)  )  )'; |

## 2、databaselink 右键->新建



## 2.2查看sql即可产生下面的



## 3.使用

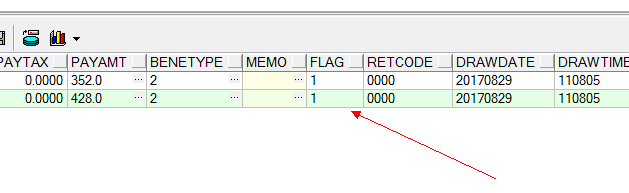
### 1、跨库查表

#### select \* from user\_bTablename@ TRUSTEE\_LINK;  ;

# 13.when case

|  |
| --- |
| "SELECT cp.ssid AS ssid," +  " cp.accname AS accname," +  " cp.PAYSUMUNTAX AS PAYSUMUNTAX," +  " cp.PAYTAX AS PAYTAX," +  " cp.payaftersum AS payAmt," +  " cp.benetype AS benetype," +  " cp.memo AS memo," +  " case cp.ssid when 'aa' then '0' else'1' end as flag, "+  " case cp.ssid when 'aa' then '3001' else '0000' end as retCode,"+  " to\_char(sysdate,'yyyyMMdd') as drawDate,"+  " to\_char(sysdate,'HH24mmss') as drawTime"+  " FROM CSIP\_OA\_PAYLISTINFO CPI" +  " JOIN CSIP\_OA\_TASKFILE CT" +  " ON CPI.PK\_PAYLISTINFO = CT.PK\_PAYLISTINFO" +  " JOIN CSIP\_OA\_PAYLIST CP" +  " ON CP.PK\_TASKFILE = CT.PK\_TASKFILE" +  " WHERE CPI.INNER\_LISTNO = '"+inworkflowno+"'" ; |

## 查出的数据



# 14、[ORACLE中%TYPE和%ROWTYPE的使用](http://blog.csdn.net/liangweiwei130/article/details/38117847)

## 1 %TYPE说明（相当于定义一个字段的变量）

为了使一个变量的数据类型与另一个已经定义了的变量（尤其是表的某一列）的数据类型相一致，[**Oracle**](http://lib.csdn.net/base/oracle)提供了%TYPE定义方式。当被参照的那个变量的数据类型改变了之后，这个新定义的变量的数据类型会自动跟随其改变，容易保持一致，也不用修改PL/SQL程序了。当不能确切地知道被参照的那个变量的数据类型时，就只能采用这种方法定义变量的数据类型。

### 1、 数据准备

--组织机构结构表

CREATE TABLE SF\_ORG

(

ORG\_ID INT NOT NULL, --组织机构主键ID

ORG\_NAME VARCHAR2(50),--组织机构名称

PARENT\_ID INT--组织机构的父级

)

--一级组织机构

INSERT INTO SF\_ORG(ORG\_ID, ORG\_NAME, PARENT\_ID) VALUES(1, '一级部门1',0);

--二级部门

INSERT INTO SF\_ORG(ORG\_ID, ORG\_NAME, PARENT\_ID) VALUES(2, '二级部门2',1);

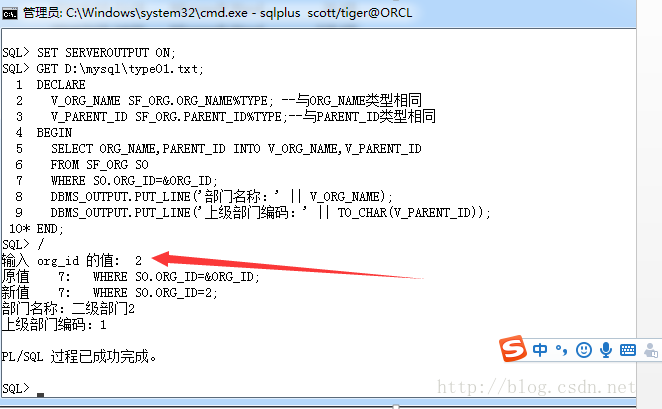
INSERT INTO SF\_ORG(ORG\_ID, ORG\_NAME, PARENT\_ID) VALUES(3, '二级部门3',1);

INSERT INTO SF\_ORG(ORG\_ID, ORG\_NAME, PARENT\_ID) VALUES(4, '二级部门4',1);

### 2、 %TYPE使用

声明两个与SF\_ORG表中的PARENT\_ID和ORG\_NAME列的数据类型相同的变量，然后用替换变量&ORG\_ID接受输入的组织机构编码，查询并显示该组织机构的名称、上级部门ID。注意，在使用由%TYPE定义的变量时要用“.”运算符指定表名限定词。

 执行结果如下图所示：



type01.txt里的SQL语句如下：

DECLARE

V\_ORG\_NAME SF\_ORG.ORG\_NAME%TYPE; --与ORG\_NAME类型相同

V\_PARENT\_ID SF\_ORG.PARENT\_ID%TYPE;--与PARENT\_ID类型相同

BEGIN

SELECT ORG\_NAME,PARENT\_ID INTO V\_ORG\_NAME,V\_PARENT\_ID

FROM SF\_ORG SO

WHERE SO.ORG\_ID=&ORG\_ID;

DBMS\_OUTPUT.PUT\_LINE('部门名称：' || V\_ORG\_NAME);

DBMS\_OUTPUT.PUT\_LINE('上级部门编码：' || TO\_CHAR(V\_PARENT\_ID));

END;

## 2 %ROWTYPE说明（相当于定义一个表的变量）

如果一个表有较多的列，使用%ROWTYPE来定义一个表示表中一行记录的变量，比分别使用%TYPE来定义表示表中各个列的变量要简洁得多，并且不容易遗漏、出错。这样会增加程序的可维护性。

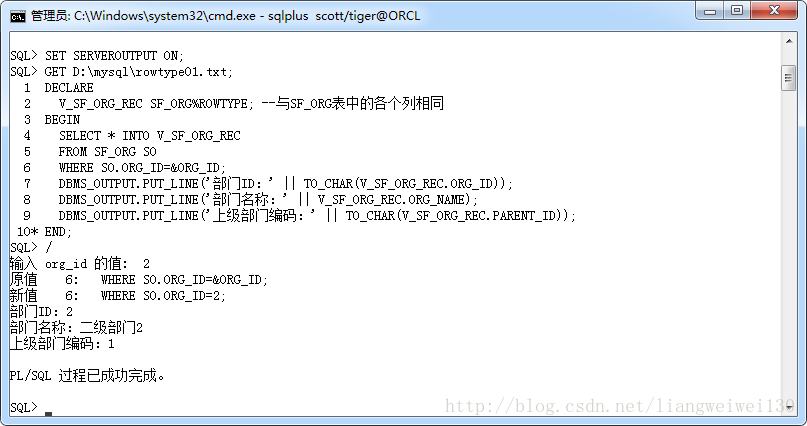
   为了使一个变量的数据类型与一个表中记录的各个列的数据类型相对应、一致，[**oracle**](http://lib.csdn.net/base/oracle)提供%ROWTYPE定义方式。当表的某些列的数据类型改变了之后，这个新定义的变量的数据类型会自动跟随其改变，容易保持一致，也不用修改PL/SQL程序了。当不能确切地知道被参照的那个表的结构及其数据类型时，就只能采用这种方法定义变量的数据类型。

   一行记录可以保存从一个表或游标中查询到的整个数据行的各列数据。一行记录的各个列与表中一行的各个列有相同的名称和数据类型。

#### 1、%ROWTYPE使用（行记录）

声明一个与SF\_ORG表中各个列的列名、数据类型都相同的行记录变量V\_SF\_ORG\_REC，然后用替换变量&ORG\_ID接受输入的组织机构编码，查询并显示该组织机构的名称、上级部门ID。注意，在使用由%ROWTYPE定义的变量时要用“.”运算符指定记录变量名限定词。

 执行结果如下图所示：



rowtype01.txt里的SQL语句如下：

DECLARE

V\_SF\_ORG\_REC SF\_ORG%ROWTYPE; --与SF\_ORG表中的各个列相同

BEGIN

SELECT \* INTO V\_SF\_ORG\_REC

FROM SF\_ORG SO

WHERE SO.ORG\_ID=&ORG\_ID;

DBMS\_OUTPUT.PUT\_LINE('部门ID：' || TO\_CHAR(V\_SF\_ORG\_REC.ORG\_ID));

DBMS\_OUTPUT.PUT\_LINE('部门名称：' || V\_SF\_ORG\_REC.ORG\_NAME);

DBMS\_OUTPUT.PUT\_LINE('上级部门编码：' || TO\_CHAR(V\_SF\_ORG\_REC.PARENT\_ID));

END;

# 15、cursor 游标的使用

rs\_cursor out SYS\_REFCURSOR)

OPEN rs\_cursor for select \* from emp;

declare cursor userinfo\_cur is select \* from userinfo\_table;

## 1、遍历游标 for while   --Fetch

<http://375940084.blog.51cto.com/2581965/1060822/>

### 1、解释：

#### 1、C\_EMP 为游标

#### 2、C\_ROW 为游标所对应的变量

|  |
| --- |
| 1. **CREATE** OR REPLACE **PROCEDURE** PRC\_LJ **IS** 2. **CURSOR** C\_EMP **IS** --声明显式游标 3. **SELECT** EMPNO,ENAME **FROM** EMP; 4. C\_ROW C\_EMP%ROWTYPE; --定义游标变量，该变量的类型为基于游标C\_EMP的记录 5. **BEGIN** 6. --For 循环 7. **FOR** C\_ROW IN C\_EMP LOOP 8. DBMS\_OUTPUT.PUT\_LINE(C\_ROW.EMPNO || '--' || C\_ROW.ENAME); 9. **END** LOOP; 11. --Fetch 循环 12. **OPEN** C\_EMP;--必须要明确的打开和关闭游标 13. LOOP 14. **FETCH** C\_EMP  //让游标C\_EMP把值赋予变量C\_ROW 15. **INTO** C\_ROW; 16. EXIT **WHEN** C\_EMP%NOTFOUND;  //条件就是当游标没有发现的时候退出 17. DBMS\_OUTPUT.PUT\_LINE(C\_ROW.EMPNO || '++' || C\_ROW.ENAME); 18. **END** LOOP; 19. **CLOSE** C\_EMP; 21. --While 循环 22. **OPEN** C\_EMP;--必须要明确的打开和关闭游标 23. **FETCH** C\_EMP **INTO** C\_ROW; 24. WHILE C\_EMP%FOUND LOOP 25. DBMS\_OUTPUT.PUT\_LINE(C\_ROW.EMPNO || '\*\*' || C\_ROW.ENAME); 26. **FETCH** C\_EMP **INTO** C\_ROW; 27. **END** LOOP; 28. **CLOSE** C\_EMP; 29. **END** PRC\_LJ; |

# 16、同义词synonym

## 1、目前的需求，sino，现在有两个数据库，一个是测试用的，一个是真实的数据库，但是现在测试的数据库和代码结合的不太好，但是还不想重新做

### 问题归纳解决：

### 1、中科软基本信息，需要先导入到临时表，然后手动的导入到正式表才能够完成测试，

### 2、现在为了解决这个问题。建立同义词（也就是建立同义词，让它和测试表的名字一样，然后同义词中使用dblink连接另外一张表，也就是正式表）

|  |
| --- |
| *-- Create the synonym*  create or replace synonym ST\_TDTRUSTEE  for TDTRUSTEE@TRUSTEE\_LINK; |

### 3、当我们测试的时候，使用测试的数据表，但是当我们不测试了，则将我们的同义词所代表的数据表删除。保留同义词，这样就保证了，不用修改代码即可完成现在的开发）) 即使数据库中标已经被删掉了，但是同义词就相当于是一张表了