Oracle 10g

Install里的setup.exe安装

属性|兼容性模式|管理员身份运行

全局数据库名:orcl SID:orcl

所有帐户口令：orcl

口令管理

普通用户:scott 密码：tiger

普通管理员：system 密码：manager

超级管理员：sys 密码：change\_on\_install

设置Sqlplus 属性兼容性和管理员身份运行

设置Sqlplusw 属性兼容性和管理员身份运行

环境变量设置：

Set linesize 长度 ； ：设置一行的长度 如 set linesize 300 ；

Set pagesize ； ：设置一页的行数 如set pagesize 30 ；

Ed 文件名 ：创建命令记事本 如ed a

@ 文件名 ：读取文件中的命令并运行 如 @ a

@ 路径 文件名 ： 读取文件中的命令并运行 如 @ d:\demo.txt

Conn 用户名/密码 as sysdba 如切换到普通管理员：conn system/manager as sysdba

执行指定用户中表数据 如查找scott表： select \* from scott.emp ；

Show user ；显示当前用户

切换连接用户：

Select \* from tab； 获取当前用户所有表名称

desc 表名 ； 查看该表的结构 如 desc emp ；

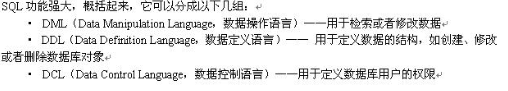
/ : 执行上一条指令

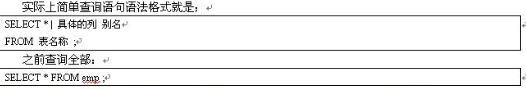
认识scott下的表：



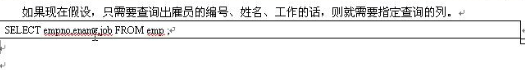


Sql语言：





如：select \* from emp ;



如： select empno,ename,job from emp ;

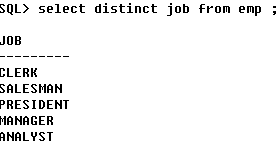
***改变列名***



如：select empno 编号 , ename 姓名 , job 工作 from emp ;





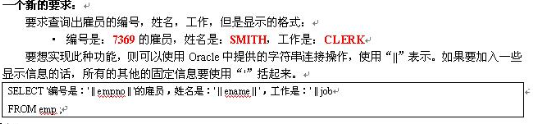






如查询有哪些种类的工作： select distinct job from emp ;

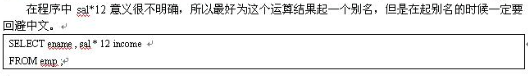
***添加注释信息***

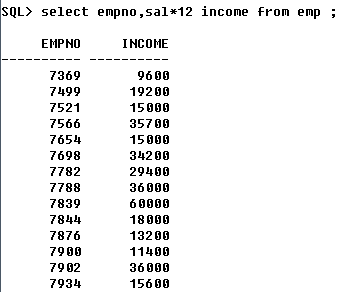




如： select '编号是'||empno||',姓名是'||ename||',工作是'||job from emp ;

可以进行四则运算

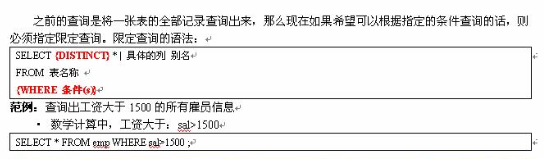


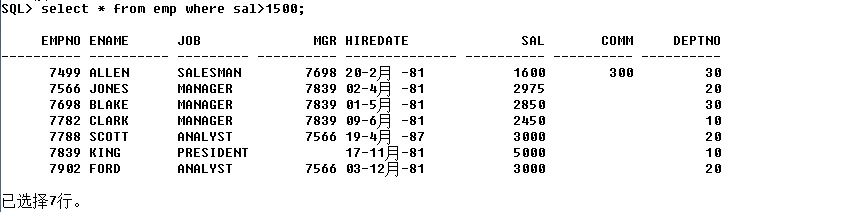


如查询雇员年薪：select empno,sal\*12 income from emp ;

**限定查询：**

**WHERE**

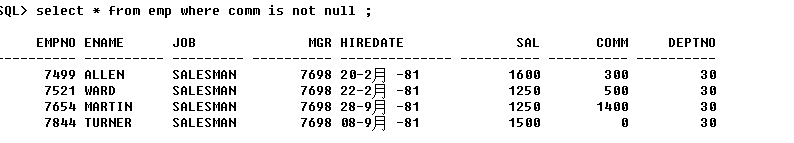




如查询工资大于1500的雇员信息 ：select \* from emp where sal>1500;

***NULL和NOT NULL***

不为空：is not null



如查询奖金不为空的雇员信息：select \* from emp where comm is not null;

为空： is null

***AND 和OR***

同时满足:and

满足其一：or

查询月薪大于1500，（并且）有奖金的雇员信息：select \* from emp where sal>1500 and comm is not null ;

查询月新大于1500或者有奖金的雇员信息：select \* from emp where sal>1500 or comm is not null;

Not 有取反的意思：

如查询月薪不大于1500或者没有奖金的雇员信息：select \* from emp where not(sal>1500 and comm is not null) ;

即： “大于....and有...” 取反为“不大于...or 没有...”

即： “大于...and 有...” 相当于“NOT (不大于...or 没有...）”

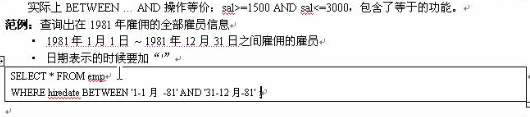
查询月薪大于1500小于3000的雇员信息：select \* from emp where sal>1500 and sal<3000 ;

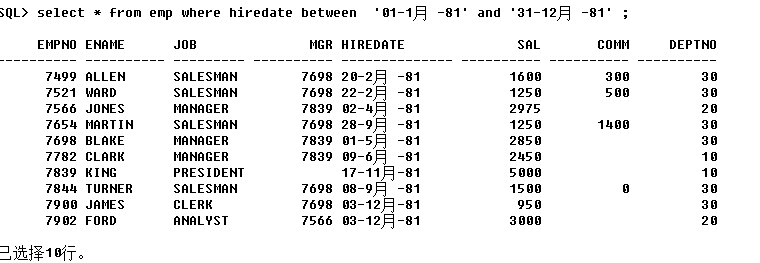
***Between and***

***（数值和日期的）***

select \* from emp where sal between 1500 and 3000 ;

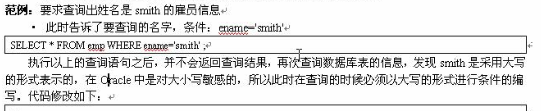
相当于 ： =<sal<=





如select \* from emp where hiredate between '01-1月 -81' and '31-12月 -81' ;

Between and 支持数字和日期的查询

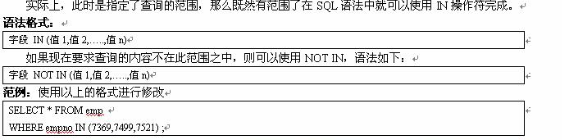


字符、日期等需要用’ ‘ 括起来

查询时条件的编写注意区分大小写

查询姓名为SMITH的雇员信息：select \* from emp where ename='SMITH' ;

***IN操作符和NOT IN（重点）***



查询empno为7369、7499、7521的雇员信息：

select \* from emp where empno in (7369,7499,7521) ;

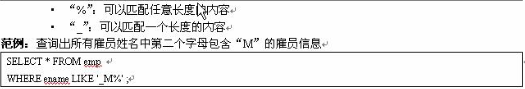
IN操作符不仅可以用在数字，还可用在字符串上，只是有无’’的区别

查询ename为SMITH.ALLEN.WARD的雇员信息：

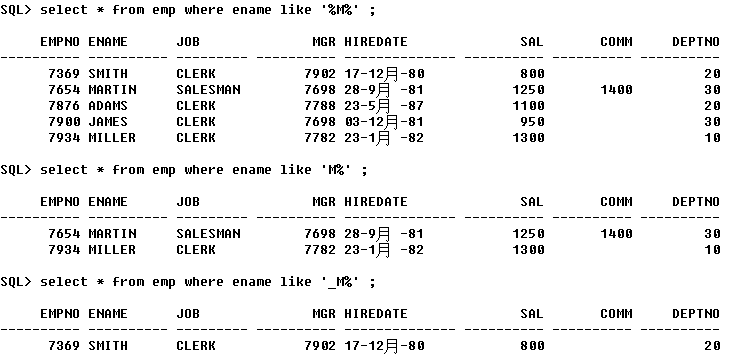
select \* from emp where ename in ('SMITH','ALLEN','WARD') ;

***LIKE(重点)***





% ：表示任意长度字符（即长度可以是0）



查询ename以M开头的雇员信息： select \* from emp where ename like 'M%' ;

查询ename第二个字符为M的雇员信息：select \* from emp where ename like '\_M%' ;

查询ename中有M的雇员信息： select \* from emp where ename like '%M%' ;



查询雇佣日期在81年的雇员信息：

1. Select \* from emp where hiredate between '01-1月 -81' and '31-12月 -81' ;

2. Select \* from emp where hiredate like ‘%81’;

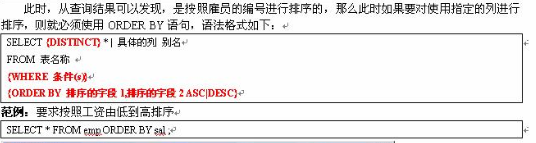
查询月薪中带5的雇员信息：select \* from emp where sal like '%5%' ;

***注：like语句后一般有 ’’ 数字形式也不例外加上’ ’***

> < = >= <=

不等于：<> 或 !=

***Order by***



***Asc：升序 （默认）***

***Desc：降序***

***排序语句在最后执行***

按工资降序排列，工资相同用按雇佣日期从早到晚排序：

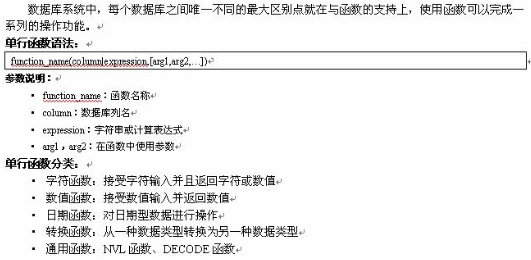
select \*

from emp

order by sal desc ,hiredate;

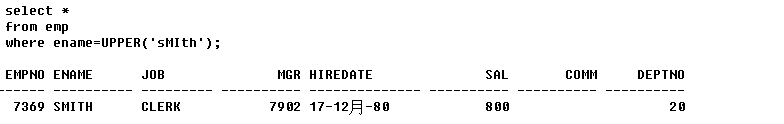
***单行函数（重点）***

***（其中DECODE()函数最重要）***



1. 字符函数：大小写转换、统计字符数等

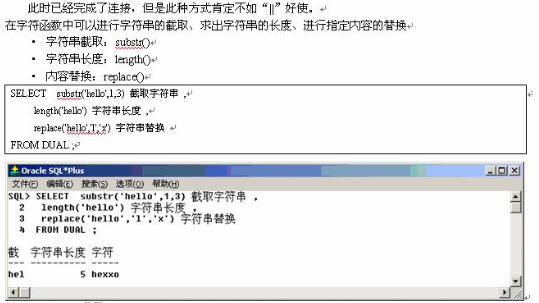
如不确定smith的每个字母大小情况下查询他的信息



UPPER:转换为大写

LOWER:转换为小写

INITCAP:第一个字母大写





***Substr（）函数截取从0和从1开始都是一样的效果（0,3）（1,3）***

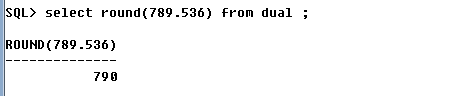
***Substr(‘hello’,1,3):从第一个开始截取三个***

***Substr(‘hello’-3,3):截取倒数三个字母***

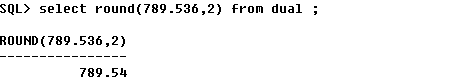
1. 数值函数：四舍五入、截断小数点、取余

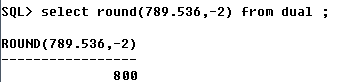
四舍五入：round（）

789.536



789.536保留两位小数四舍五入

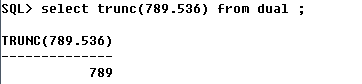




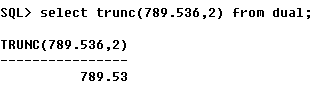
截断小数点：trunc（）



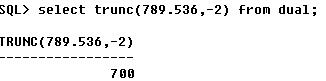
789.536不保留小数



789.536保留两位小数

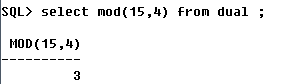


789.536

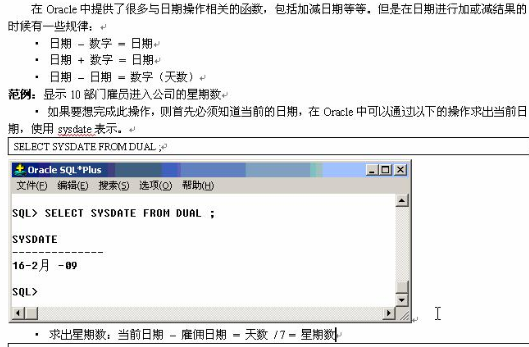


取余：mod（）

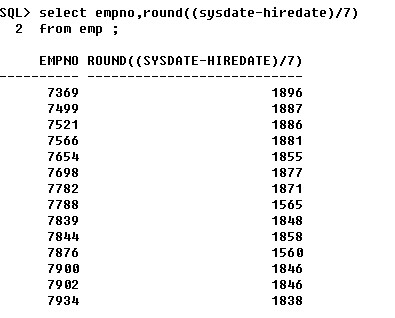
15/3的余数



1. 日期函数（重点）

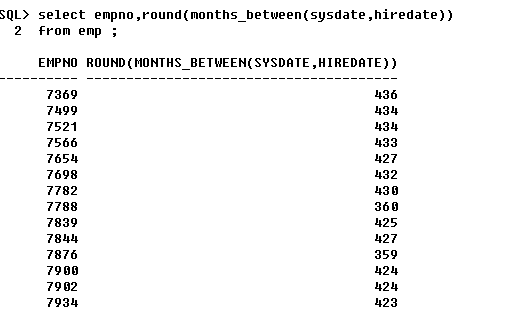


进入公司的星期数（四舍五入）





查询雇佣日期到当前日期的月数

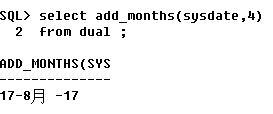


Months\_between(大日期，小日期)

一般程序中有小数点是用round函数进行四舍五入



当前日期加上四个月

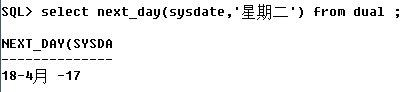




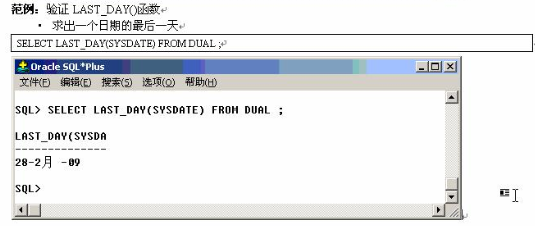
如：下一个星期一是什么时候（今天是2017.4.17.星期一）



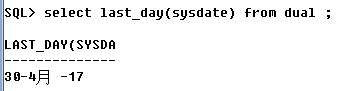
然而下一个星期二（今天是2017.4.17.星期一）则是4.18







本月最后一天：select last\_day(sysdate) from dual;



1. ***转换函数（最重要）***

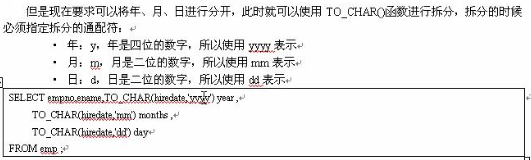
To\_char()

To\_number()

To\_date()

To\_char()

[日期的拆分和数字的格式化]



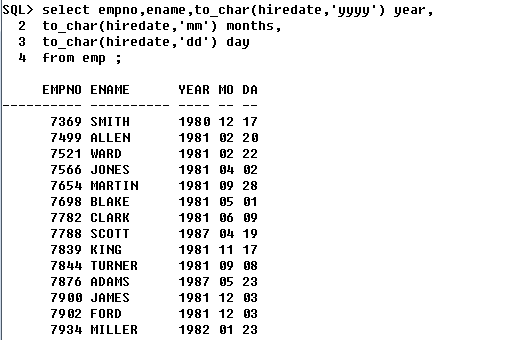
将hiredate按照yyyy mm dd形式查找出来：

select empno,ename,to\_char(hiredate,'yyyy') year,

to\_char(hiredate,'mm') months,

to\_char(hiredate,'dd') day

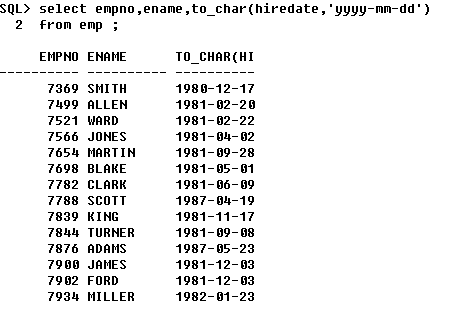
from emp ;



将hiredate按照yyyy-mm-dd形式查找出来：

select empno,ename,to\_char(hiredate,'yyyy-mm-dd')

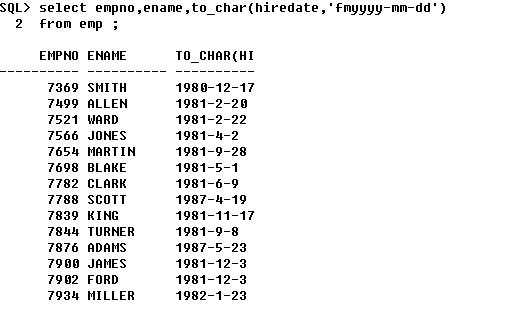
from emp ;





如：select empno,ename,to\_char(hiredate,'fmyyyy-mm-dd')

from emp ;



除了以上日期可以用到to\_char()函数

还有数字的格式化需要用到to\_cha()函数如：

查询雇员编号、姓名、工资（其中工资每三位一“，”并且加上单位）

其中：用“9”表示一位数

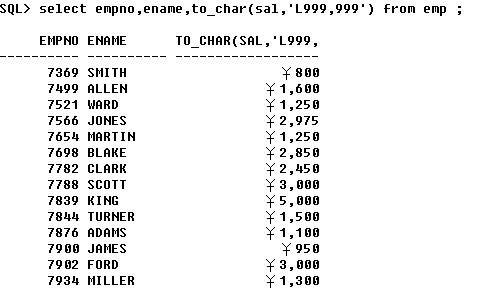
$:表示美元

L:表示当地单位

如：select empno,ename,to\_char(sal,'$999,999') from emp ;



又如：select empno,ename,to\_char(sal,'L999,999') from emp ;

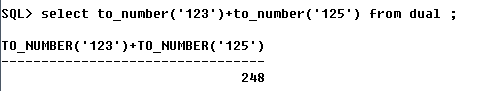


To\_number()

（字符换成数字）

(字符型一般有‘’【改变列名除外】；如‘123’表示的是字符串)

select to\_number('123')+to\_number('125') from dual ;

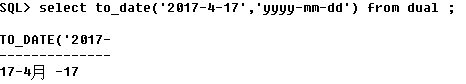


To\_date()

（字符串变为date型）

如：把字符串‘2017-4-17’变为date类型

select to\_date('2017-4-17','yyyy-mm-dd') from dual ;

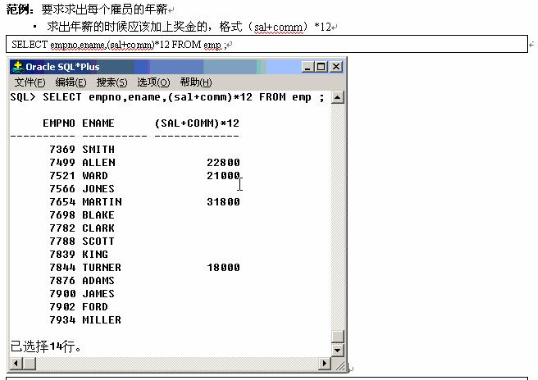


这样会把字符串形式变为Oracle默认的日期（date）类型形式：17-4月 -17

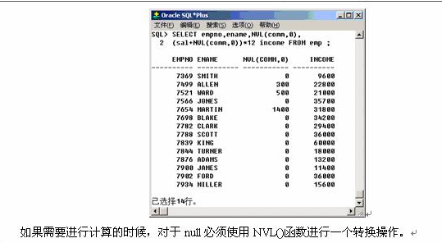
1. 通用函数

{NVL()函数、}

NVL()函数

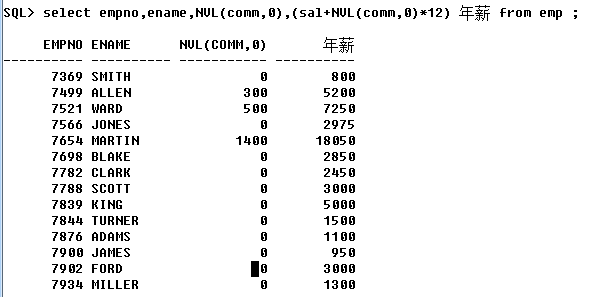




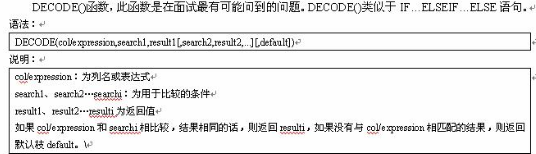


计算年薪（包括奖金comm）：

select empno,ename,NVL(comm,0),(sal+NVL(comm,0)\*12) 年薪 from emp ;



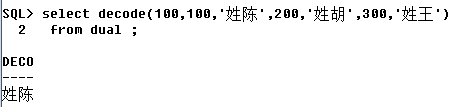
DECODE()函数【重点】



预先确定输入的是100；如输入是100时输出姓陈，如输入200姓胡，如输入300姓王

select decode(100,100,'姓陈',200,'姓胡',300,'姓王')

from dual ;



JOB

---------

CLERK：业务员

SALESMAN：销售员

PRESIDENT：总裁

MANAGER：经理

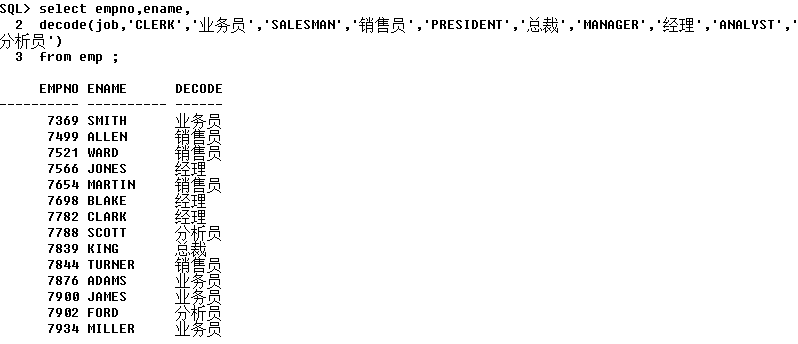
ANALYST：分析员

如要求：查询出雇员编号、姓名、工作（以上面的形式给出）

select empno,ename

decode(job,'CLERK','业务员','SALESMAN','销售员','PRESIDENT','总裁','MANAGER','经理','ANALYST','分析员')

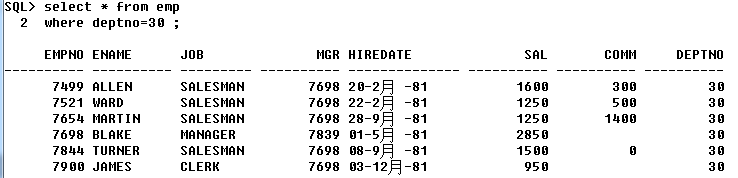
from emp ;



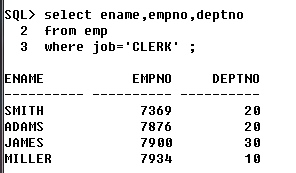
注：CLERK、SALESMAN、PRESIDENT、MANAGER、ANALYST都要有 ’’ 符号

练习

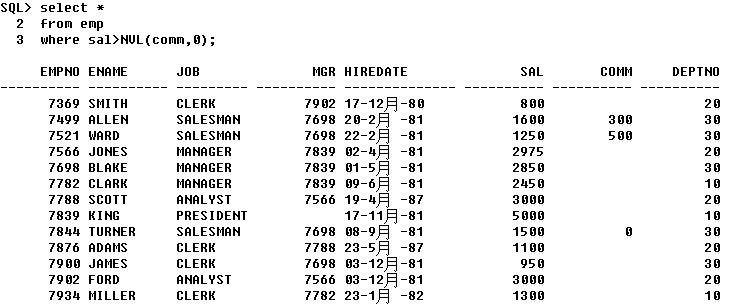




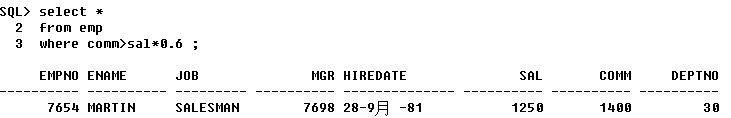




\*\*\*\*\*3.查询工资sal大于奖金comm的员工信息





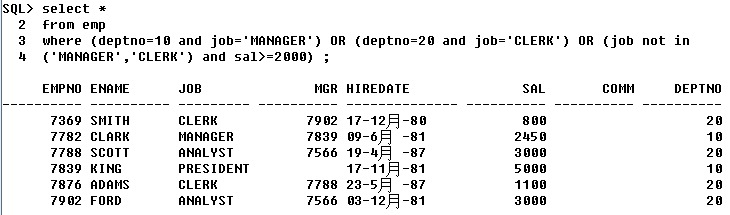


\*\*\*\*\*\*\*\*\*

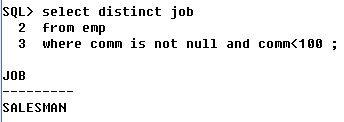




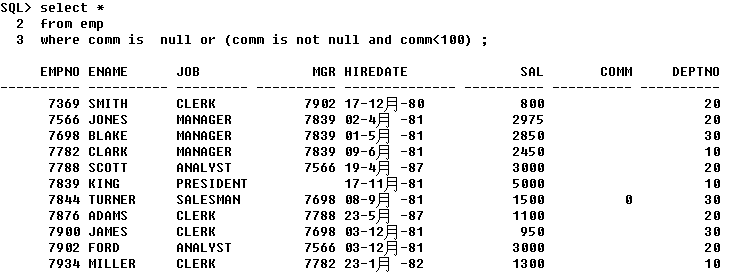
\*\*\*\*\*\*\*\*\*







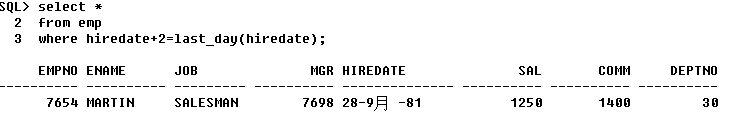




\*\*\*\*\*\*\*\*

（last\_day()）

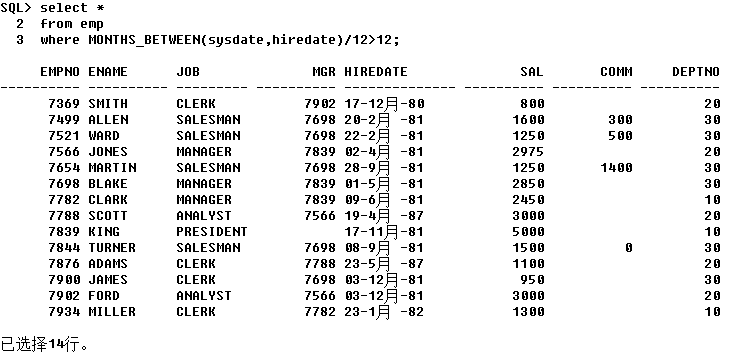
\*\*\*\*\*\*\*\*\*\*



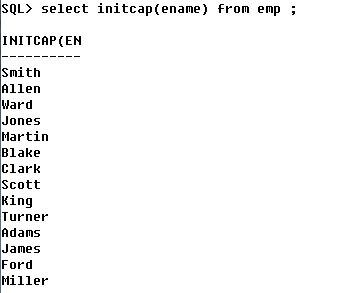
\*\*\*\*\*



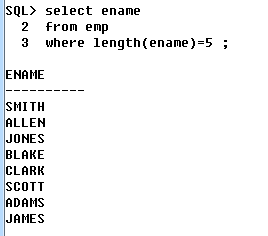
\*\*\*\*\*



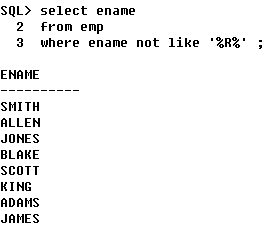




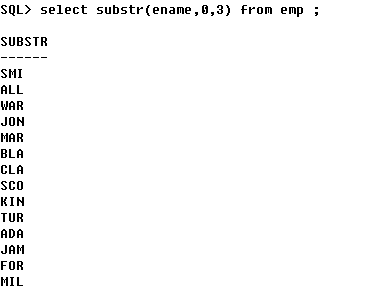




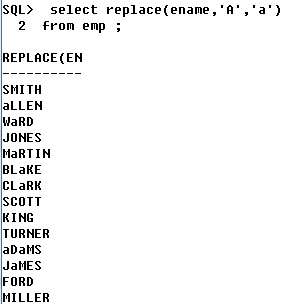








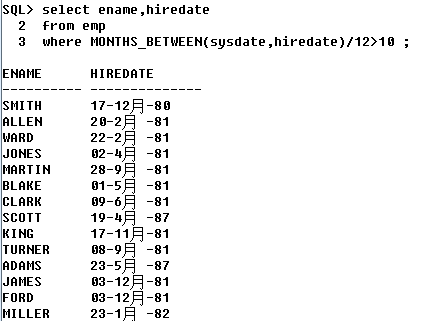




\*\*\*\*\*\*



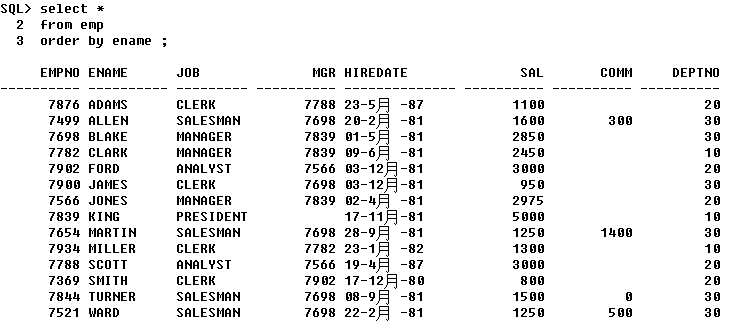
\*\*\*\*\*\*



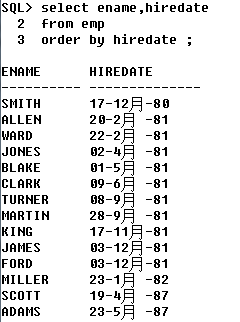
\*\*\*\*\*\*\*



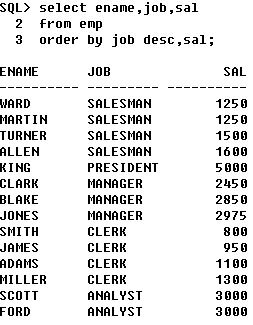
\*\*\*\*\*\*\*







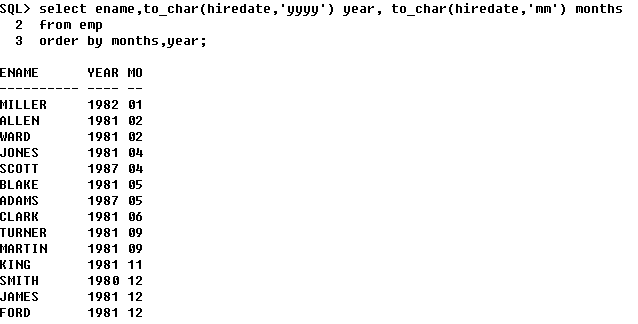




\*\*\*\*\*\*



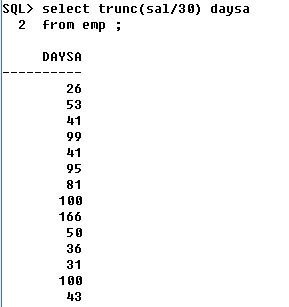
\*\*\*\*\*\*



\*\*\*\*\*



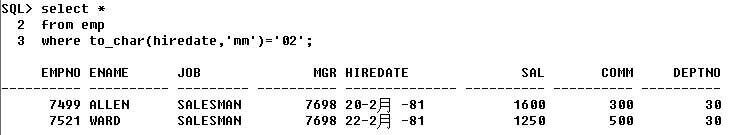
\*\*\*\*\*\*



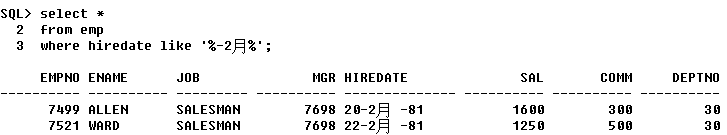
\*\*\*\*\*\*\*\*



\*\*\*\*\*\*\*\*



或

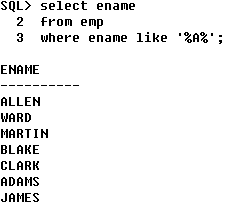


\*\*\*\*\*\*

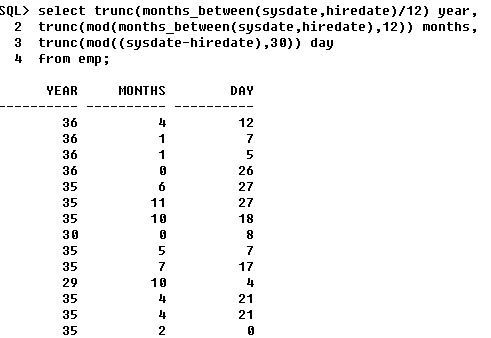




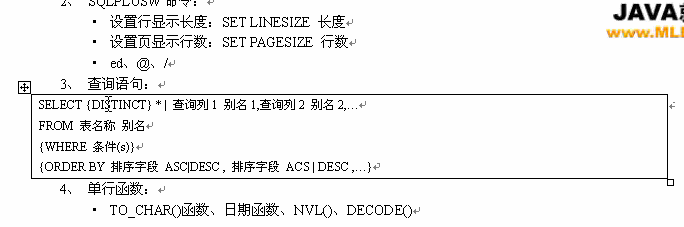




[mod():取余函数]



**回顾重点**



回顾结束

《多表查询》





如查询emp表和dept表：

select \*

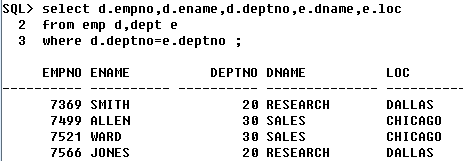
from emp,dept

where emp.deptno=dept.deptno ;







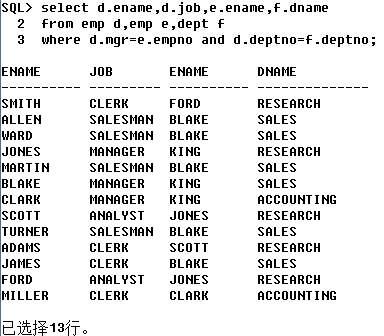








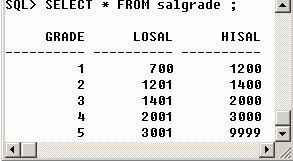




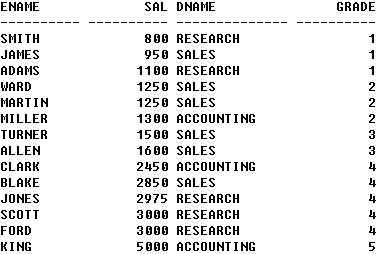
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



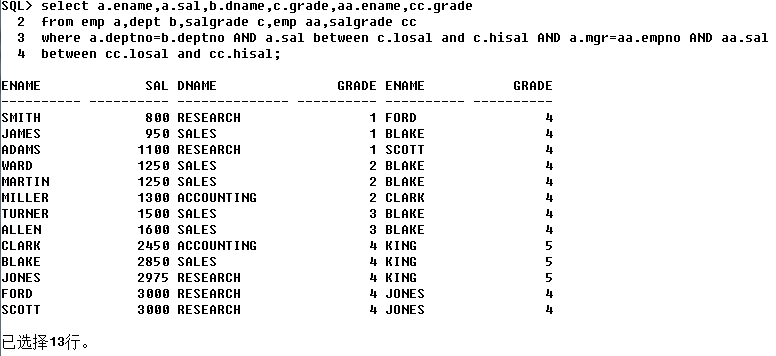


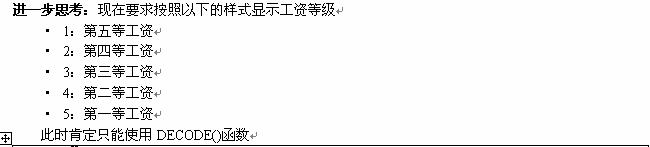


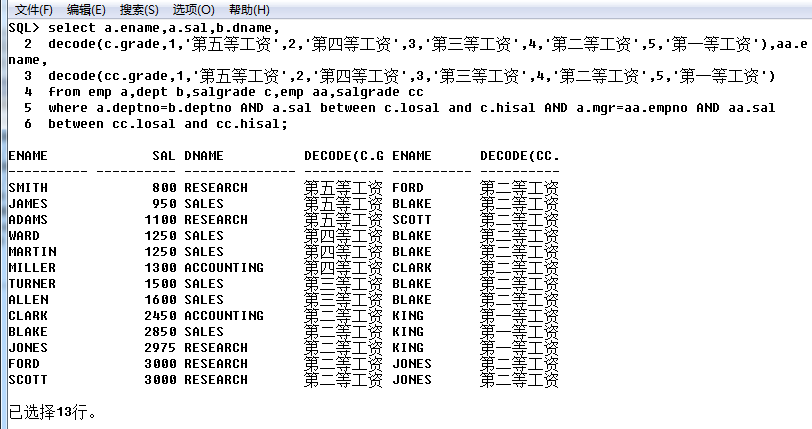










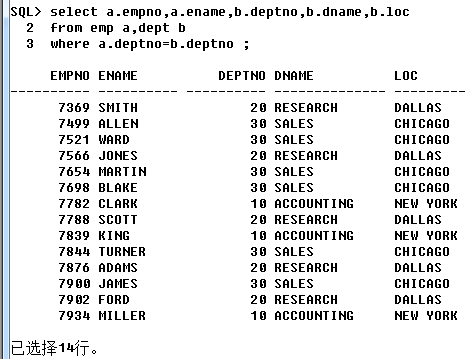


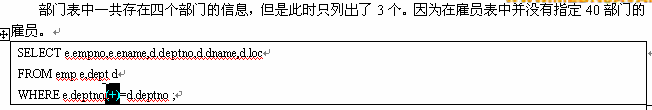
左右（外）连接

用左连接,就是以左边那个表为标准，左边那表的所有记录必须得全部出现

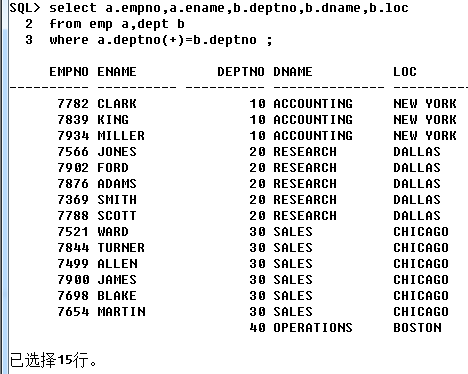
用右连接,就是以右边那个表为标准，右边那表的所有记录必须得全部出现

左连接：左表是主表  
右连接：右表是主表  
内连接：普通的，两表平等



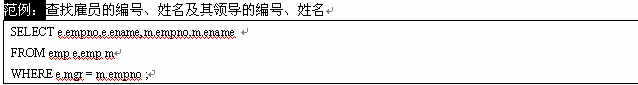


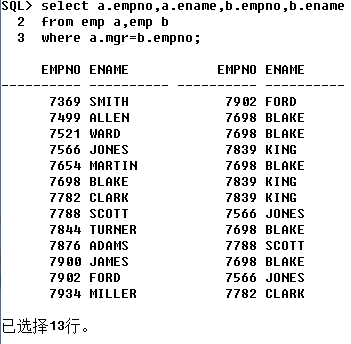
上面用了右连接，以右表b为准，b表里的deptno所有就必须有





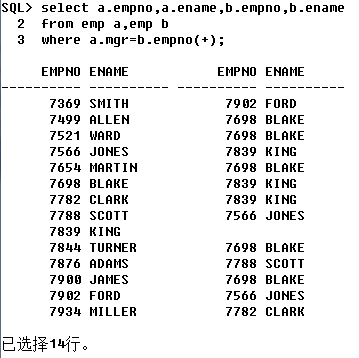








下面的用了左（外）连接，以左边的表a为准，a表里的mgr（存在空的mgr）所有记录必须有





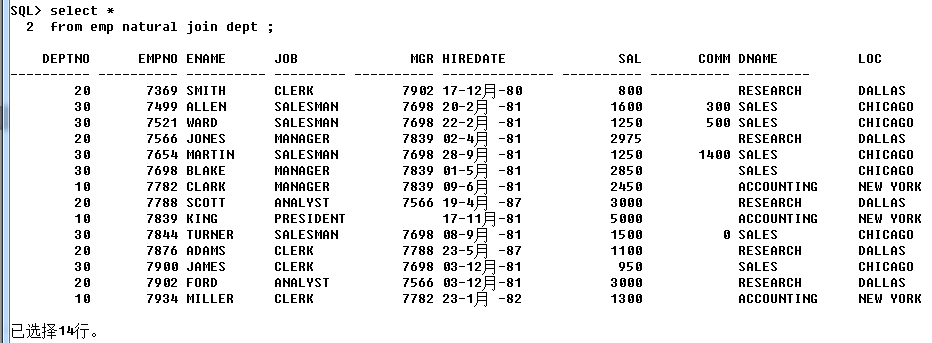
（了解就行）





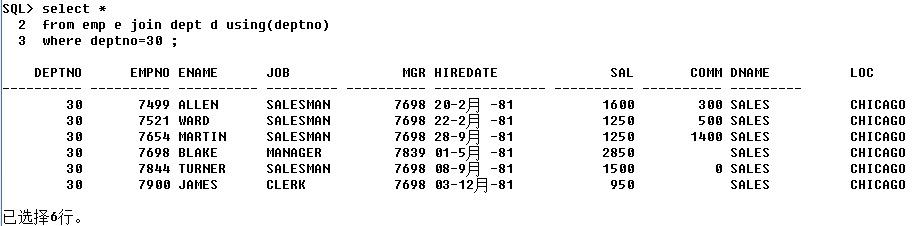
产生56种结果，即：14\*4（笛卡尔积）

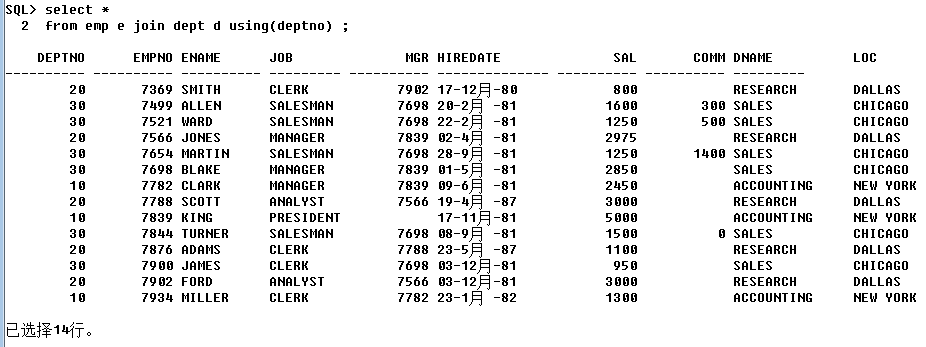




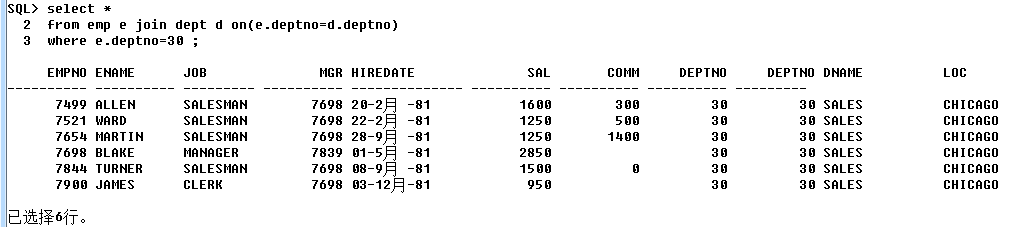




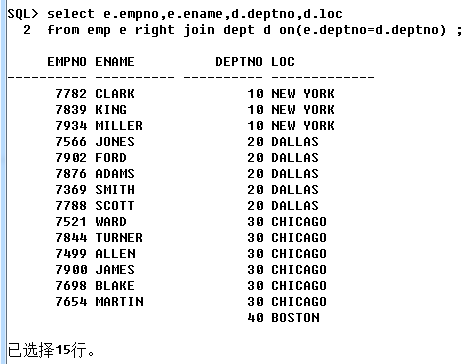
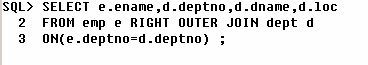




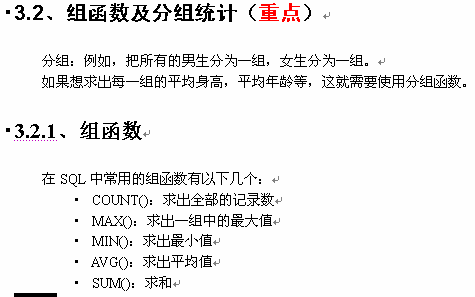




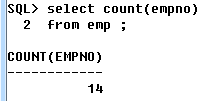




组函数和分组统计（**重点**）

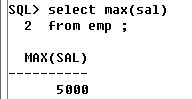




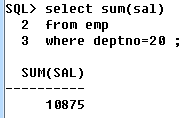


Count(\*)、count(empno)等同

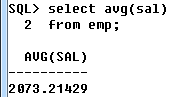






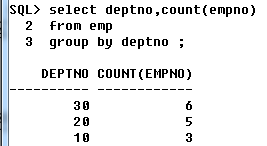




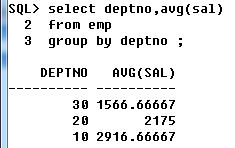


分组统计

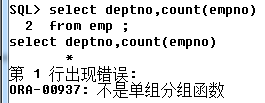


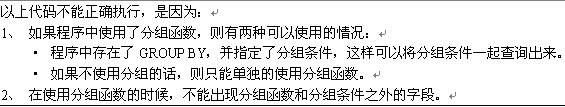












即deptno不符合“2”：是用分组函数count（）出现了分组函数和分组条件之外的字段“deptno”

**再如：**



“empno”字段是分组函数本身和分组条件之外的字段，而”deptno”是分组条件的字段

SQL> select empno,count(empno)

2 from emp;

select empno,count(empno)

\*

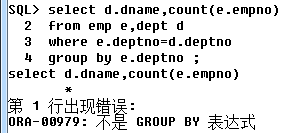
第 1 行出现错误:

ORA-00937: 不是单组分组函数

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



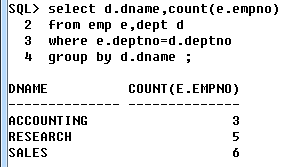
**错误示范：**



因为d.name是分组函数和分组条件之外的字段

所以可以把分组条件改为d.dname

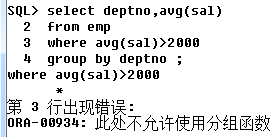
**即：**



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

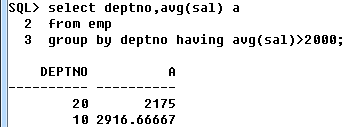


**错误示范：**



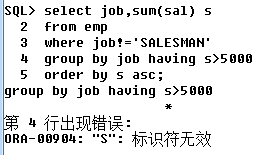


**是Asc不是acs**



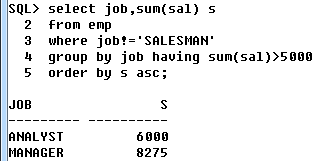


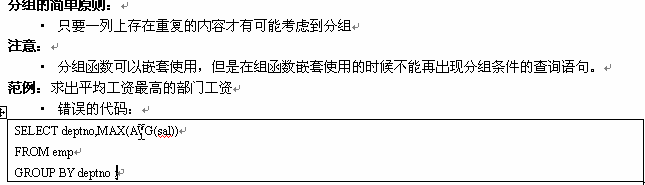
**错误示范：**



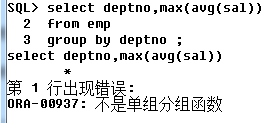
**Having中必须用分组函数函数表示而不能用别名**

**正确解答：**

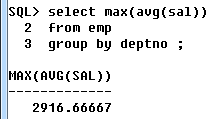




错误示范：



正确解答：

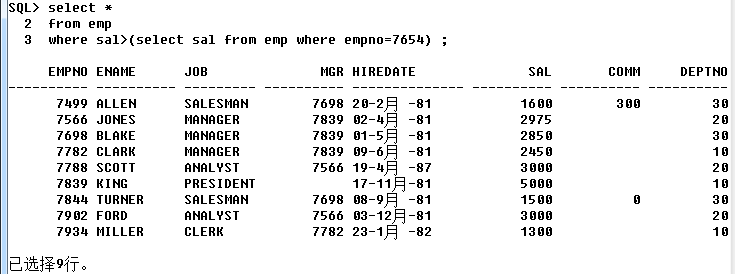


**子查询**

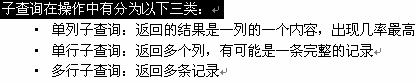


From语句后、where条件语句后可加子查询

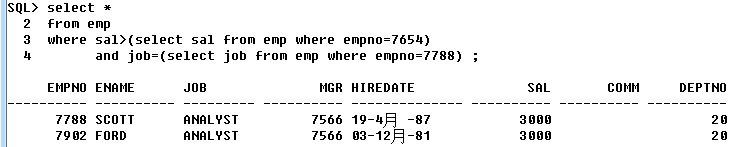




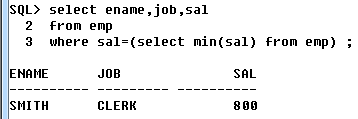






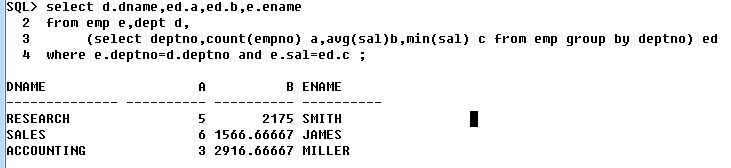




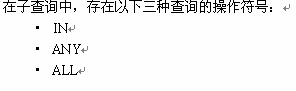


\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*









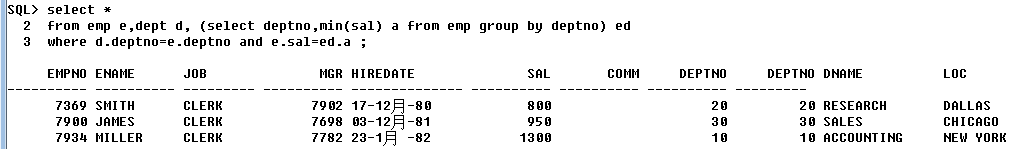


一解：

select \*

from emp e,dept d, (select deptno,min(sal) a from emp group by deptno) ed

where d.deptno=e.deptno and e.sal=ed.a ;

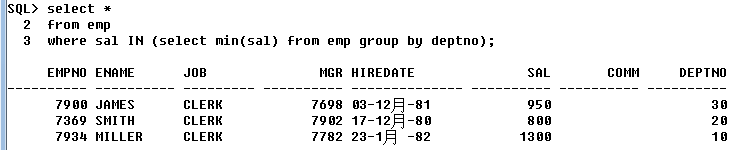


二解：

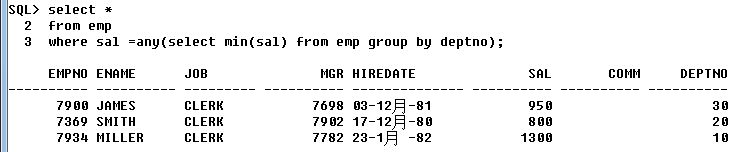
select \*

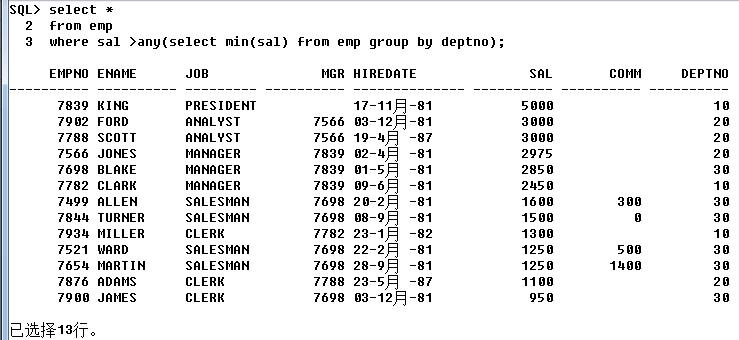
from emp

where sal IN (select min(sal) from emp group by deptno);

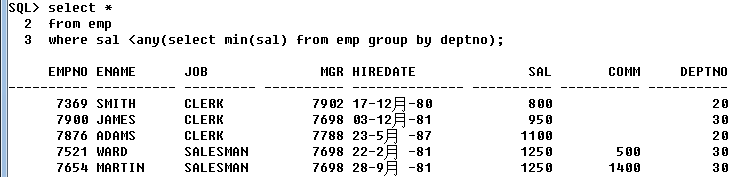




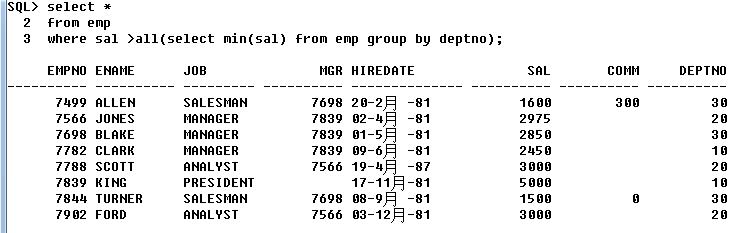
 （即大于其中一个就行）



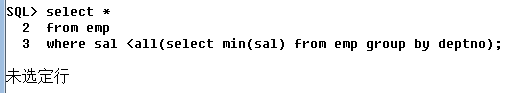
·<ANY ：（即小于其中一个就行）



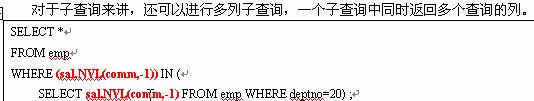
（大于全部才行）



（即小于全部才行）



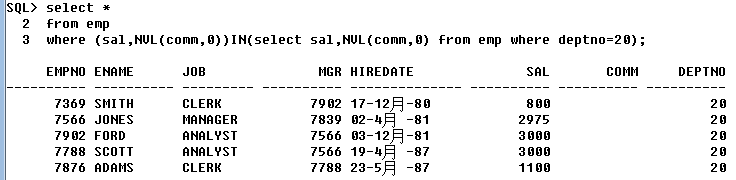
以下只做了解：（读懂就好）



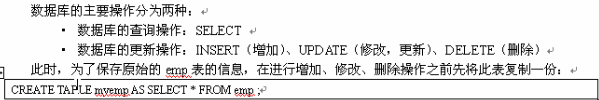
select \*

from emp

where (sal,NVL(comm,0))IN(select sal,NVL(comm,0) from emp where deptno=20);

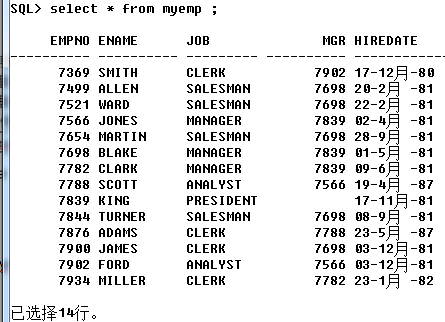


**数据库更新操作**

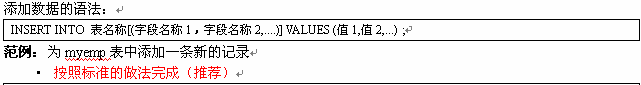




select \* from myemp ;

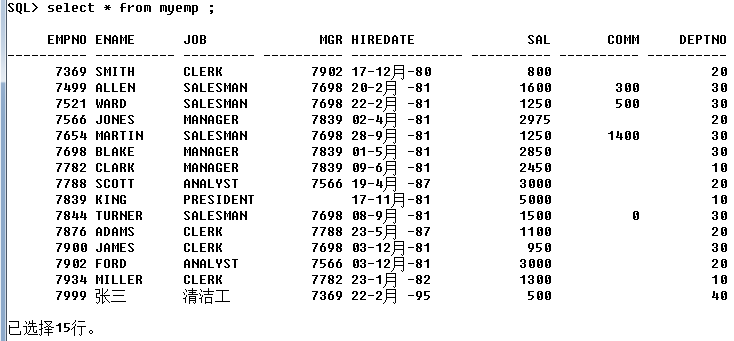


添加（insert）



insert into myemp(empno,ename,job,mgr,hiredate,sal,comm,deptno)

values(7999,'张三','清洁工',7369,'22-2月 -95',500,null,40) ;







即：

insert into myemp

values(7999,'张三','清洁工',7369,'22-2月 -95',500,null,40) ;



insert into myemp(empno,ename,job,hiredate,sal,deptno)

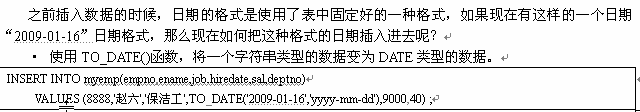
values(7999,'张三','清洁工','22-2月 -95',500,40) ;



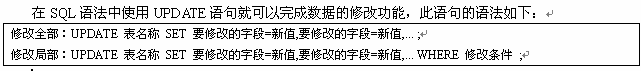


insert into myemp(empno,ename,job,mgr,hiredate,sal,comm,deptno)

values(7999,'张三','清洁工',null,'22-2月 -95',500,null,40) ;

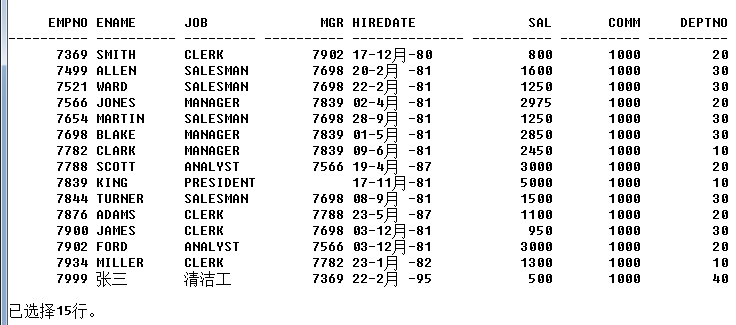


修改（更新）update





update myemp set comm=1000 ;



**范例：**将7999号员工的工资改为10000

update myemp set sal=10000

where empno=7999 ;



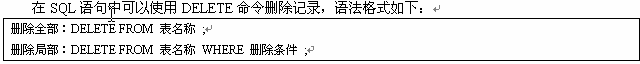
范例：在7999号员工工资基础上加上10000

update myemp set sal=sal+10000

where empno=7999 ;

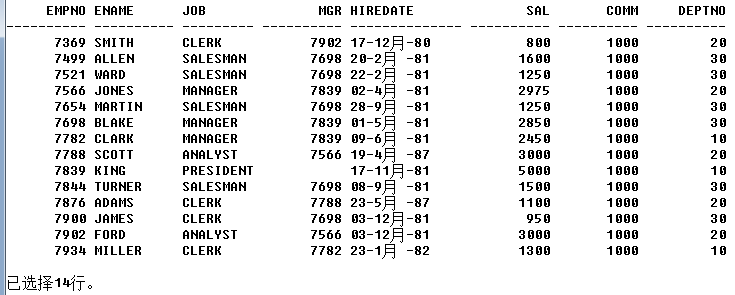


删除（delete）





delete from myemp where empno=7999;





delete from myemp where empno in(8899,7889,8888);

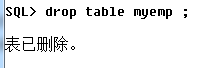


delete from myemp ;

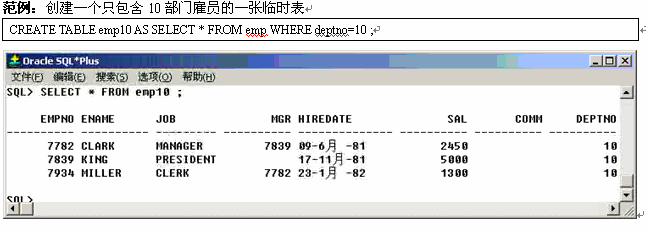




**删除整个表**

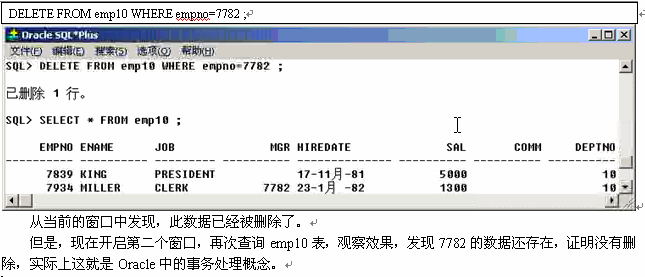


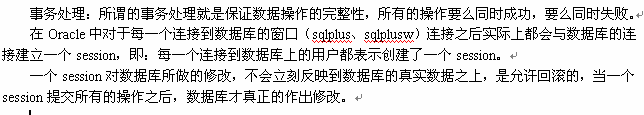
**事务处理**



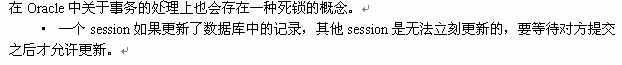


Delete from emp10 where empno=10 ;





一次数据库更新之后如果已经提交了（commit）那么无法回退



  事务（Transaction）是并发控制的基本单位。所谓的事务，它是一个操作序列，这些操作要么都执行，要么都不执行，它是一个不可分割的工作单位。例如，银行转账工作：从一个账号扣款并使另一个账号增款，这两个操作要么都执行，要么都不执行。所以，应该把它们看成一个事务。事务是**[数据库](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "http://blog.csdn.net/zdwzzu2006/article/details/_blank)**维护数据一致性的单位，在每个事务结束时，都能保持数据一致性。

       针对上面的描述可以看出，事务的提出主要是为了解决并发情况下保持数据一致性的问题。

       事务具有以下4个基本特征(ACID)。

●   Atomic（原子性）：事务中包含的操作被看做一个逻辑单元，这个逻辑单元中的操作要么全部成功，要么全部失败。

●   Consistency（一致性）：只有合法的数据可以被写入数据库，否则事务应该将其回滚到最初状态。

●   Isolation（隔离性）：事务允许多个用户对同一个数据进行并发访问，而不破坏数据的正确性和完整性。同时，并行事务的修改必须与其他并行事务的修改相互独立。

●   Durability（持久性）：事务结束后，事务处理的结果必须能够得到固化。

**2.事务的语句**　开始事物：BEGIN TRANSACTION  
　提交事物：COMMIT TRANSACTION  
　回滚事务：ROLLBACK TRANSACTION  
**3.事务的4个属性**     ①原子性(Atomicity)：事务中的所有元素作为一个整体提交或回滚，事务的个元素是不可分的，事务是一个完整操作。  
  ②一致性(Consistemcy)：事物完成时，数据必须是一致的，也就是说，和事物开始之前，数据存储中的数据处于一致状态。保证数据的无损。  
  ③隔离性(Isolation)：对数据进行修改的多个事务是彼此隔离的。这表明事务必须是独立的，不应该以任何方式以来于或影响其他事务。  
  ④持久性(Durability)：事务完成之后，它对于系统的影响是永久的，该修改即使出现系统故障也将一直保留，真实的修改了数据库  
**4.事务的保存点**     SAVE TRANSACTION 保存点名称 --自定义保存点的名称和位置  
     ROLLBACK TRANSACTION 保存点名称 --回滚到自定义的保存点

 其他高手的一些补充：

 事务的标准定义： 指作为单个逻辑工作单元执行的一系列操作，而这些逻辑工作单元需要具有原子性，  一致性，隔离性和持久性四个属性，统称为ACID特性。

所谓事务是用户定义的一个数据库操作序列,这些操作要么全做要么全不做,是一个不可分割的工作单位。例如,在关系数据库中,一个事务可以是一条SQL语句、一组SQL语句或整个程序。   
事务和程序是两个概念。一般地讲,一个程序中包含多个事务。  
事务的开始与结束可以由用户显式控制。如果用户没有显式地定义事务,则由DBMS按缺省规定自动划分事   
务。在SQL语言中,定义事务的语句有三条：   
BEGIN TRANSACTION   
COMMIT   
ROLLBACK

同生共死。。  
显示事务被用begin transaction 与 end transaction 标识起来，其中的 update 与 delete 语句或者全部执行或者全部不执行。。 如：  
begin transaction T1  
update student  
set name='Tank'  
where id=2006010  
delete from student  
where id=2006011  
commit  
简单地说，事务是一种机制，用以维护数据库的完整性。

其实现形式就是将普通的SQL语句嵌入到Begin Tran...Commit Tran 中（或完整形式 Begin Transaction...Commit Transaction)，当然，必要时还可以使用RollBack Tran 回滚事务，即撤销操作。

利用事务机制，对数据库的操作要么全部执行，要么全部不执行，保证数据库的一致性。需要使用事务的SQL语句通常是更新和删除操作等。

end transaction T1

关于savepoint

用户在事务（transaction）内可以声明（declare）被称为保存点（savepoint）  
的标记。保存点将一个大事务划分为较小的片断。

用户可以使用保存点（savepoint）在事务（transaction）内的任意位置作标  
记。之后用户在对事务进行回滚操作（rolling back）时，就可以选择从当前  
执行位置回滚到事务内的任意一个保存点。例如用户可以在一系列复杂的更  
新（update）操作之间插入保存点，如果执行过程中一个语句出现错误，用  
户 可以回滚到错误之前的某个保存点，而不必重新提交所有的语句。

在开发应用程序时也同样可以使用保存点（savepoint）。如果一个过程  
（procedure）内包含多个函数（function），用户可以在每个函数的开始位置  
创建一个保存点。当一个函数失败时， 就很容易将数据恢复到函数执行之前  
的状态，回滚（roll back）后可以修改参数重新调用函数，或执行相关的错误  
处理。

当事务（transaction）被回滚（rollback）到某个保存点（savepoint）后，  
**[Oracle](http://lib.csdn.net/base/oracle" \o "Oracle知识库" \t "http://blog.csdn.net/zdwzzu2006/article/details/_blank)**将释放由被回滚语句使用的锁。其他等待被锁资源的事务就可以继续  
执行。需要更新（update）被锁数据行的事务也可以继续执行。

将事务（transaction）回滚（roll back）到某个保存点（savepoint）的过程如  
下：  
1. **[oracle](http://lib.csdn.net/base/oracle" \o "Oracle知识库" \t "http://blog.csdn.net/zdwzzu2006/article/details/_blank)** 回滚指定保存点之后的语句  
2. Oracle 保留指定的保存点，但其后创建的保存点都将被清除  
3. Oracle 释放此保存点后获得的表级锁（table lock）与行级锁（row  
lock），但之前的数据锁依然保留。

被部分回滚的事务（transaction）依然处于活动状态，可以继续执行。

一个事务（transaction）在等待其他事务的过程中，进行回滚（roll back）到  
某个保存点（savepoint）的操作不会释放行级锁（row lock）。为了避免事务  
因为不能获得锁而被挂起，应在执行 UPDATE 或 DELETE 操作前使用 FOR  
UPDATE ... NOWAIT 语句。（以上内容讲述的是回滚保存点之前所获得的  
锁。而在保存点之后获得的行级锁是会被释放的，同时保存点之后执行的  
[SQL](http://blog.csdn.net/zdwzzu2006/article/details/javascript:;" \t "http://blog.csdn.net/zdwzzu2006/article/details/_self) 语句也会被完全回滚）。

1.没有提交(commit)的数据 删除后无法rollback  
2.提交(commit)了的数据 删除后可以使用rollback恢复  
3.删除数据后 提交(commit)则无法使用rollback恢复  
  
就是说，你提交了的数据是永久性的，rollback是在这个基础上进行备份的。



1. .列出至少有一个员工的部门号

select deptno

from emp

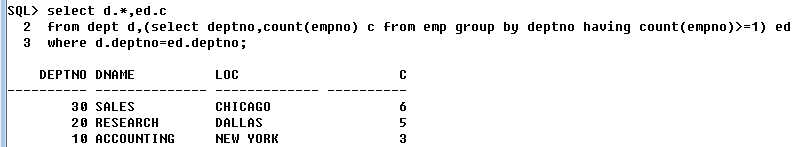
group by deptno having count(empno)>=1 ;

1. .列出部门信息

select d.\*,ed.c

from dept d,(select deptno,count(empno) c from emp group by deptno having count(empno)>=1) ed

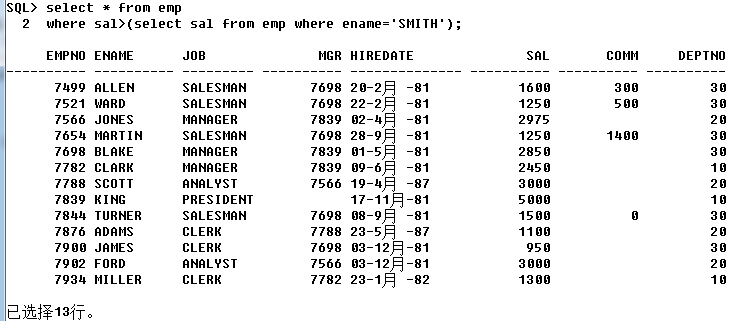
where d.deptno=ed.deptno;





select \* from emp

where sal>(select sal from emp where ename='SMITH');

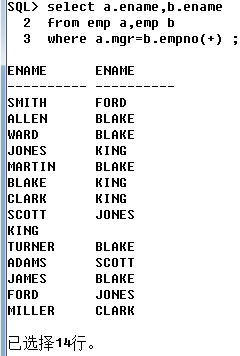




select a.ename,b.ename

from emp a,emp b

where a.mgr=b.empno(+) ;

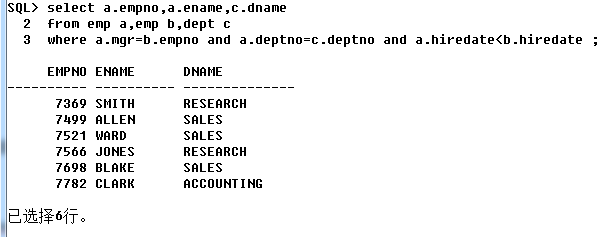




select a.empno,a.ename,c.dname

from emp a,emp b,dept c

where a.mgr=b.empno and a.deptno=c.deptno and a.hiredate<b.hiredate ;

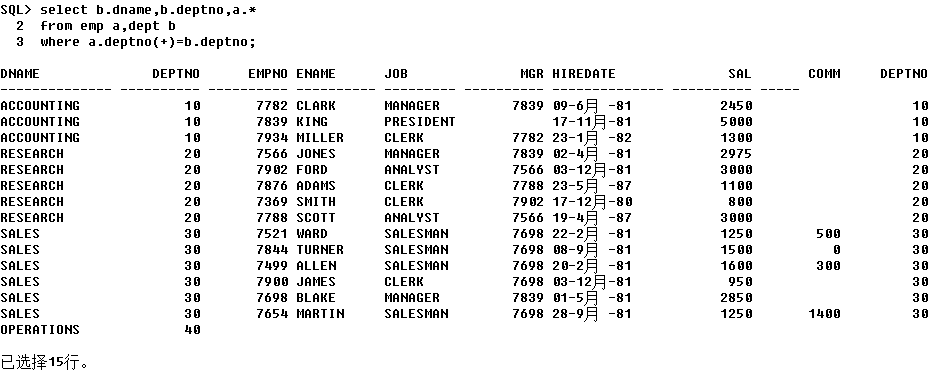




select b.dname,b.deptno,a.\*

from emp a,dept b

where a.deptno(+)=b.deptno;

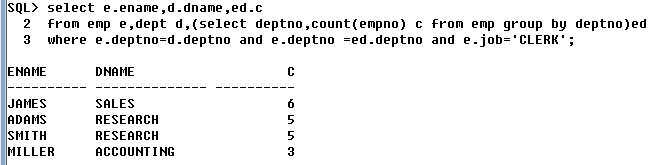




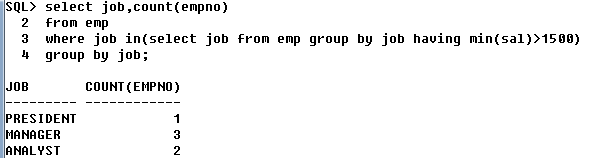
select e.ename,d.dname,ed.c

from emp e,dept d,(select deptno,count(empno) c from emp group by deptno)ed

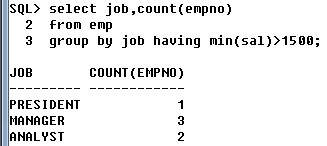
where e.deptno=d.deptno and e.deptno =ed.deptno and e.job='CLERK';







或

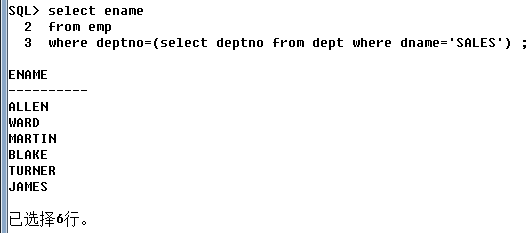




select ename

from emp

where deptno=(select deptno from dept where dname='SALES') ;



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



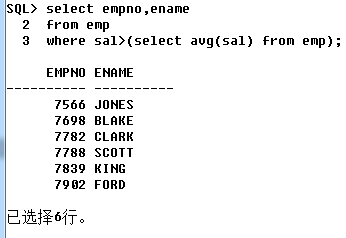
分步：

1. 列出高于平均薪金的员工

select empno,ename

from emp

where sal>(select avg(sal) from emp);

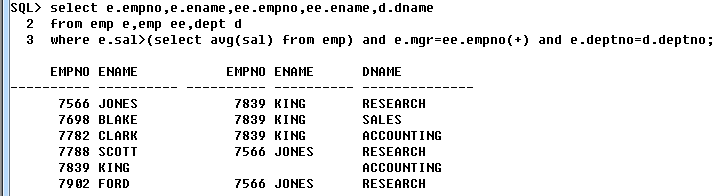


1. 列出高于平均薪金的员工、上级领导、部门

select e.empno,e.ename,ee.empno,ee.ename,d.dname

from emp e,emp ee,dept d

where e.sal>(select avg(sal) from emp) and e.mgr=ee.empno(+) and e.deptno=d.deptno;

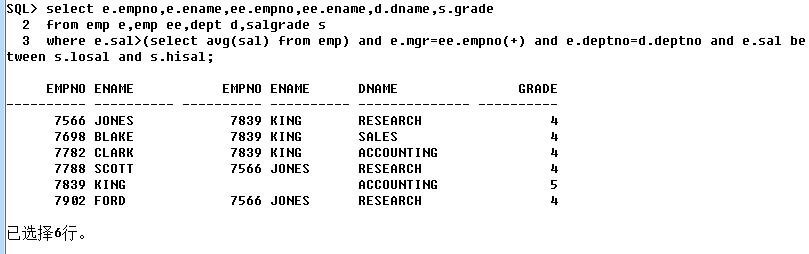


1. 列出高于平均薪金的员工、上级领导、部门、薪金等级

select e.empno,e.ename,ee.empno,ee.ename,d.dname,s.grade

from emp e,emp ee,dept d,salgrade s

where e.sal>(select avg(sal) from emp) and e.mgr=ee.empno(+) and e.deptno=d.deptno and e.sal between s.losal and s.hisal;

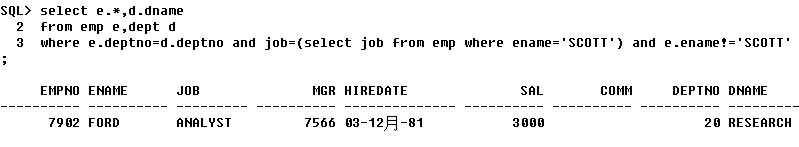




select e.\*,d.dname

from emp e,dept d

where e.deptno=d.deptno and job=(select job from emp where ename='SCOTT') and e.ename!='SCOTT' ;

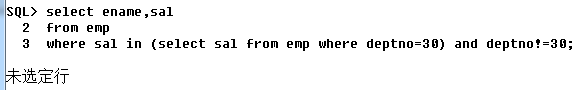




select ename,sal

from emp

where sal in (select sal from emp where deptno=30) and deptno!=30;



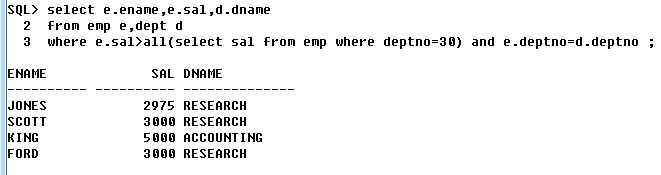
（无符合条件数据行）



select e.ename,e.sal,d.dname

from emp e,dept d

where e.sal>all(select sal from emp where deptno=30) and e.deptno=d.deptno ;



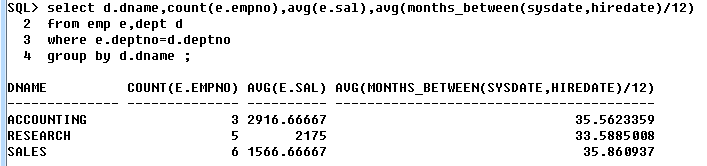


select d.dname,count(e.empno),avg(e.sal),avg(months\_between(sysdate,hiredate)/12)

from emp e,dept d

where e.deptno=d.deptno

group by d.dname ;

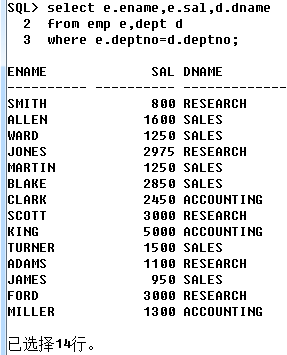




select e.ename,e.sal,d.dname

from emp e,dept d

where e.deptno=d.deptno;

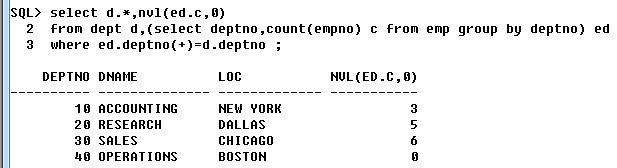




select d.\*,nvl(ed.c,0)

from dept d,(select deptno,count(empno) c from emp group by deptno) ed

where ed.deptno(+)=d.deptno ;



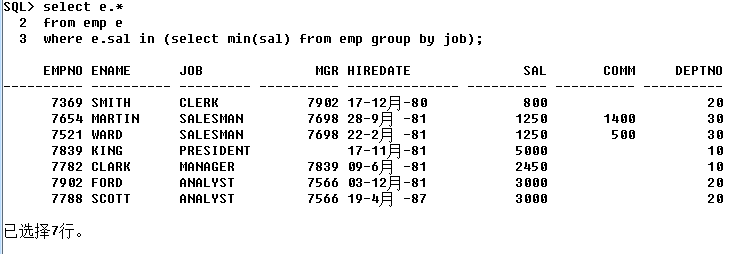


1. 列出各工作的最低工资和最低工资员工信息

select e.\*

from emp e

where e.sal in (select min(sal) from emp group by job);



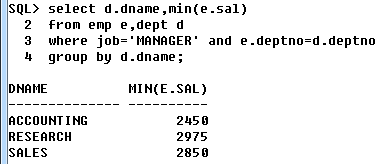


select d.dname,min(e.sal)

from emp e,dept d

where job='MANAGER' and e.deptno=d.deptno

group by d.dname;

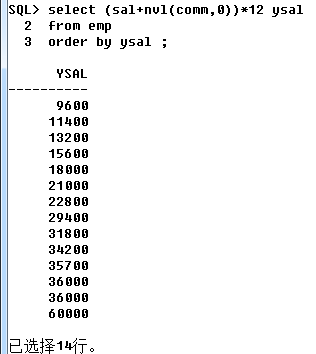




select (sal+nvl(comm,0))\*12 ysal

from emp

order by ysal ;

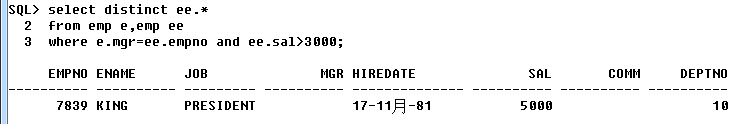




select distinct ee.\*

from emp e,emp ee

where e.mgr=ee.empno and ee.sal>3000;



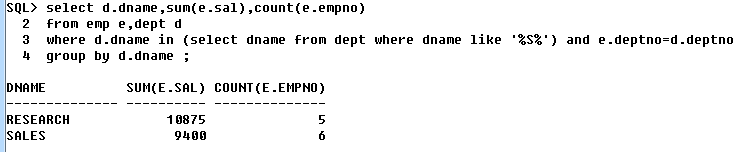


select d.dname,sum(e.sal),count(e.empno)

from emp e,dept d

where d.dname in (select dname from dept where dname like '%S%') and e.deptno=d.deptno

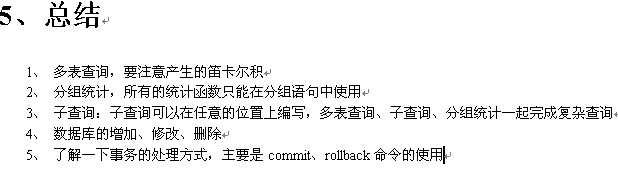
group by d.dname ;



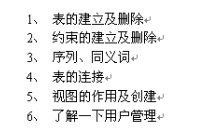


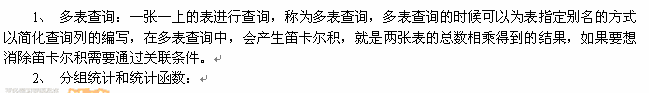
update emp set sal=sal\*1.1

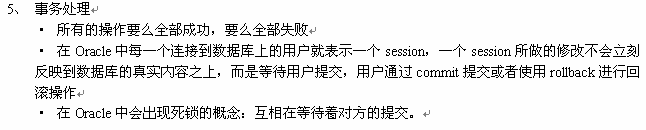
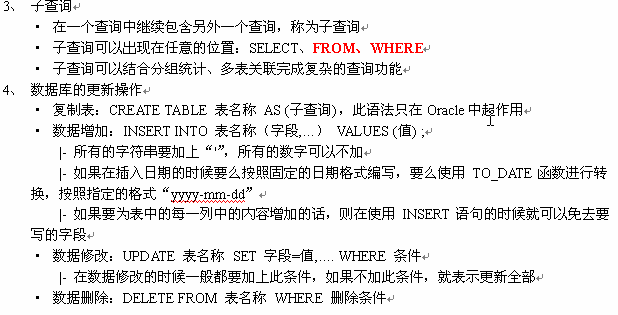
where (months\_between(sysdate,hiredate)/12)>10 ;



预习

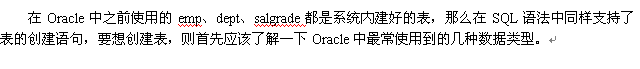




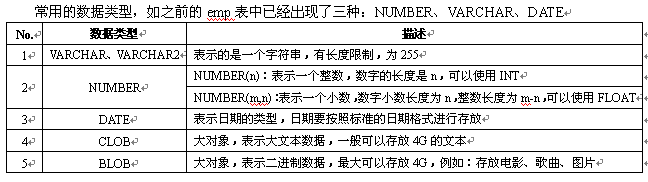


**表的管理**





数据类型



**表的建立**



create table 表名称（

字段 数据类型 {default 默认值}，

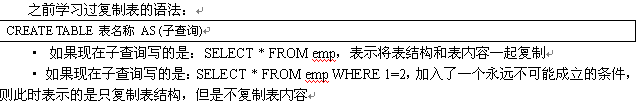
字段 数据类型 {default 默认值}，

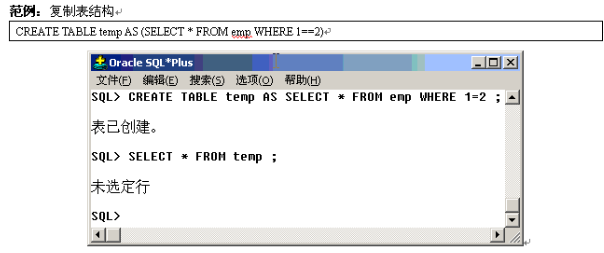
字段 数据类型 {default 默认值}，

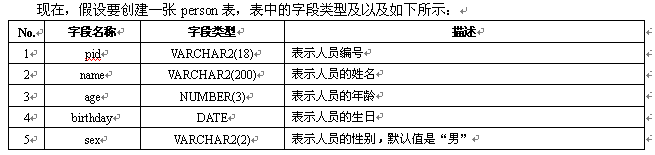
........

字段 数据类型 {default 默认值}

）;







create table person(

pid varchar2(18) ,

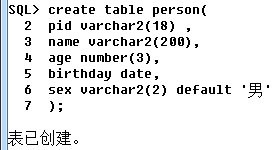
name varchar2(200),

age number(3),

birthday date,

sex varchar2(2) default '男'

);

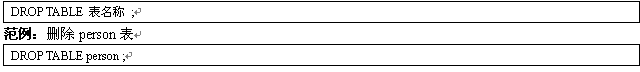




insert into person (pid,name,age,birthday,sex) values ('100','张三',18,to\_date('1995-09-12','yyyy-mm-dd'),'女');

注：varchar要‘’ number不用‘’ 日期要转换（用to\_date()函数）

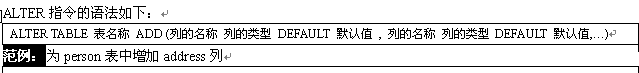
**表的删除**



drop table person ;

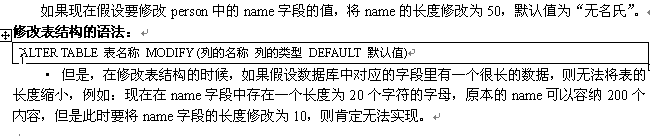


**表的修改**

alter table person add (address varchar2(20) default '略');



修改列的数据类型的方法:  
通常可以写成 alter table 表名 modify column 列名 新的列的类型  
例如:student表中列sname的类型是char(20),现在要修改为varchar(20),[SQL语句](https://www.baidu.com/s?wd=SQL%E8%AF%AD%E5%8F%A5&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YzPj64Pv7WnWD1nycsP1b0IAYqnWm3PW64rj0d0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtkrHcYP1bsn10knHDznjmvnWcL" \t "https://zhidao.baidu.com/question/_blank)如下  
 alter table student modify column sname varchar(20);  
   
同时修改列名和列的数据类型的方法:  
通常可以写成 alter table 表名 change column 旧列名 新列名 新的列类型  
例如:student表中列sname的类型是char(20),要修改为stuname varchar(20),[SQL语句](https://www.baidu.com/s?wd=SQL%E8%AF%AD%E5%8F%A5&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YzPj64Pv7WnWD1nycsP1b0IAYqnWm3PW64rj0d0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtkrHcYP1bsn10knHDznjmvnWcL" \t "https://zhidao.baidu.com/question/_blank)如下:  
 alter table student change column sname stuname varchar(20);

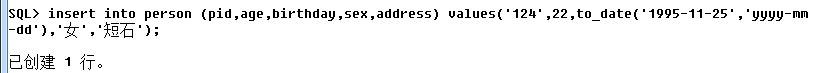


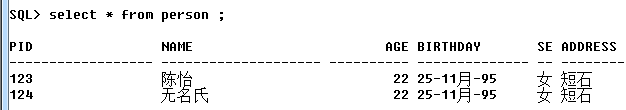






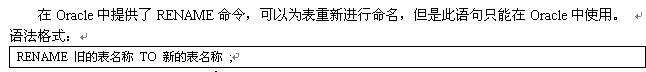
insert into person (pid,age,birthday,sex,address) values('124',22,to\_date('1995-11-25','yyyy-mm-dd'),'女','短石');





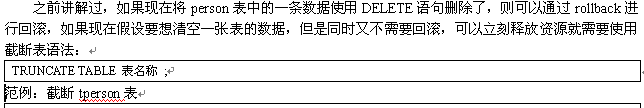
以上可看出不设置name就是默认值“无名氏”

**重命名**

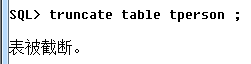




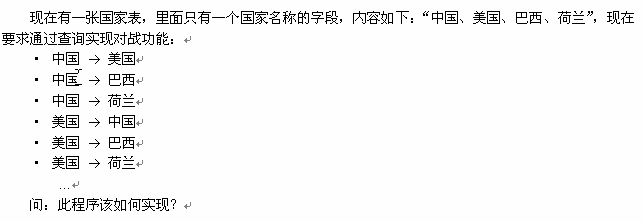
**截断表（了解）**



truncate table tperson ;



**ALTER** **TABLE**SCOTT.TEST **RENAME TO**TEST1--修改表名  
**ALTER TABLE**SCOTT.TEST **RENAME COLUMN**NAME **TO**NAME1 --修改表列名  
**ALTER TABLE**SCOTT.TEST **MODIFY**NAME1 **NUMBER**(20) --修改字段类型  
**ALTER TABLE**SCOTT.TEST **ADD**ADDRESS **VARCHAR2**(40) --添加表列  
**ALTER TABLE**SCOTT.TEST **DROP**NAME **CASCADECONSTRAINTS --删除表列**





先建表，然后加数据

create table national(

name varchar2(15)

);

--增加数据

insert into national(name) values ('中国');

insert into national(name) values ('美国');

insert into national(name) values ('巴西');

insert into national(name) values ('荷兰');



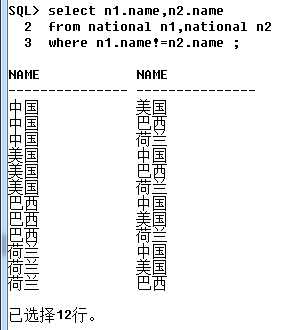
然后关联（但注意：不能中国vs中国-------）

所以：

select n1.name,n2.name

from national n1,national n2

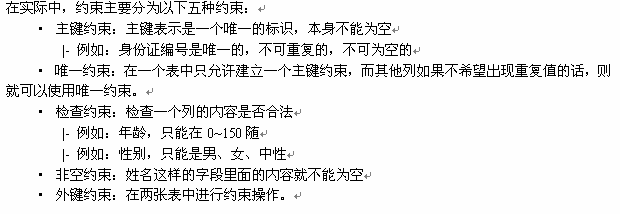
where n1.name!=n2.name ;



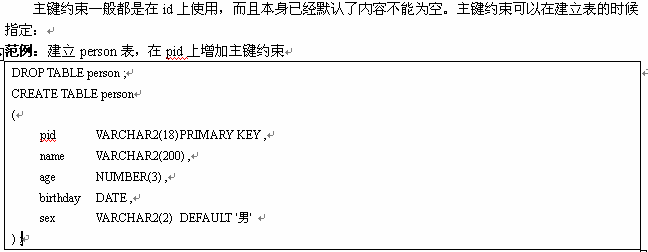
**约束**



约束的分类：



**主键约束**



create table person (

pid varchar2(15) primary key,

name varchar2(18),

age number(3),

birthday date,

sex varchar2(10) default '男',

address varchar2(18)

);

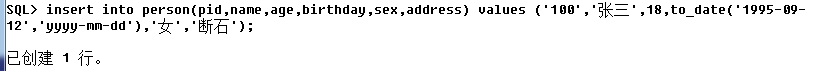


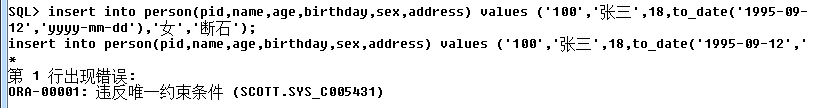


insert into person(pid,name,age,birthday,sex,address) values ('100','张三',18,to\_date('1995-09-12','yyyy-mm-dd'),'女','断石');

insert into person(pid,name,age,birthday,sex,address) values ('100','张三',18,to\_date('1995-09-12','yyyy-mm-dd'),'女','断石');

insert into person(name,age,birthday,sex,address) values ('张三',18,to\_date('1995-09-12','yyyy-mm-dd'),'女','断石');

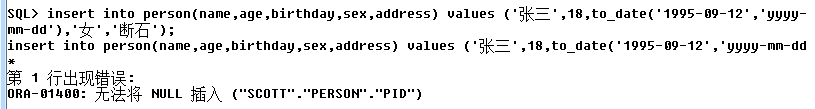


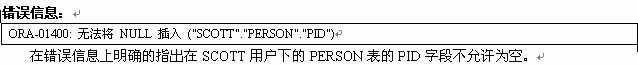


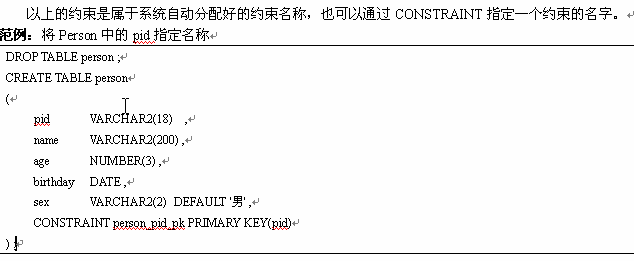








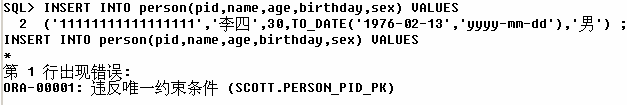




“CONSTRAINT person\_pid\_pk PRIMARY KEY(pid)”中person\_pid\_pk 代表改掉 错误提示信息的编号名称

如再







**非空约束(NOT NULL)**



create table person (

pid varchar2(15) primary key,

name varchar2(18) not null,

age number(3) not null,

birthday date,

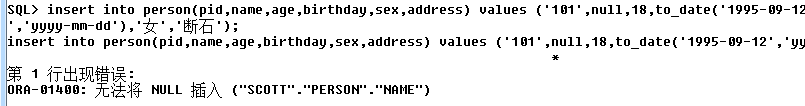
sex varchar2(10) default '男',

address varchar2(18)

);

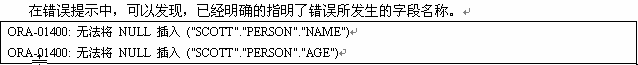


insert into person(pid,name,age,birthday,sex,address) values ('101',null,18,to\_date('1995-09-12','yyyy-mm-dd'),'女','断石');



insert into person(pid,name,age,birthday,sex,address) values ('102','张三',null,to\_date('1995-09-12','yyyy-mm-dd'),'女','断石');





**唯一约束（UNIQUE）**



create table person (

pid varchar2(15) primary key,

name varchar2(18) unique not null,

age number(3) not null,

birthday date,

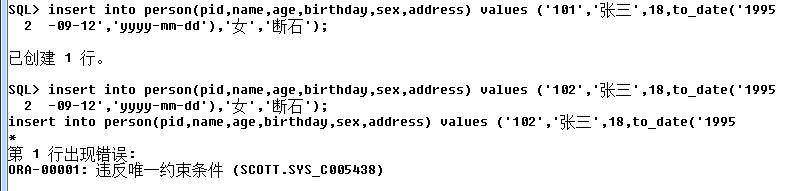
sex varchar2(10) default '男',

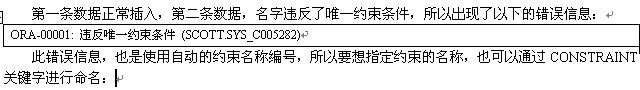
address varchar2(18)

);



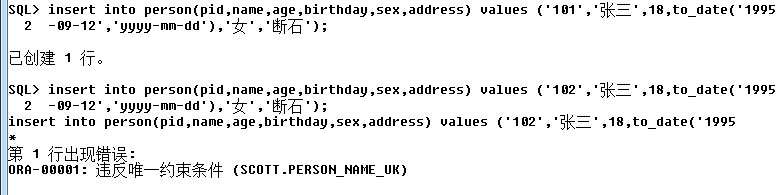






如下：





可以看到用constraint之后错误信息中变更了信息

**检查约束（check）**



如：

Age 在0~150

Sex 男、女、中

create table person (

pid varchar2(15) ,

name varchar2(18) not null,

age number(3) not null check(age between 0 and 150),

birthday date,

sex varchar2(10) default '男' check(sex in('男','女','中')),

address varchar2(18),

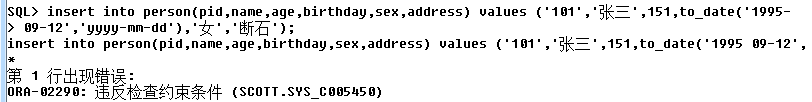
constraint person\_pid\_pk primary key(pid),

constraint person\_name\_uk unique(name)

);

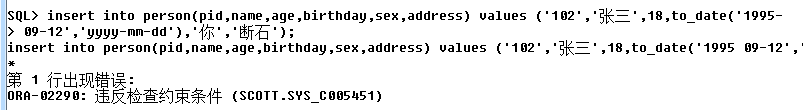


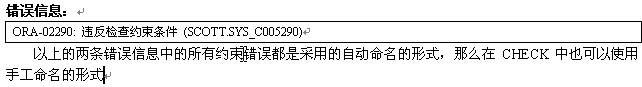
insert into person(pid,name,age,birthday,sex,address) values ('101','张三',151,to\_date('1995-09-12','yyyy-mm-dd'),'女','断石');



insert into person(pid,name,age,birthday,sex,address) values ('102','张三',18,to\_date('1995-

09-12','yyyy-mm-dd'),'你','断石');





create table person (

pid varchar2(15) ,

name varchar2(18) not null,

age number(3) not null ,

birthday date,

sex varchar2(10) default '男' ,

address varchar2(18),

constraint person\_pid\_pk primary key(pid),

constraint person\_name\_uk unique(name),

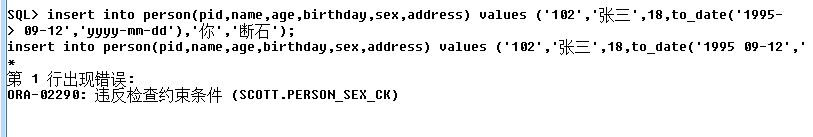
constraint person\_age\_ck check(age between 0 and 150),

constraint person\_sex\_ck check(sex in('男','女','中'))

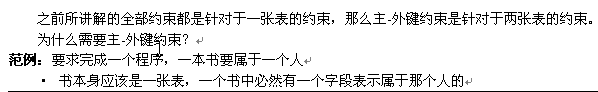
);







**外键约束（foreign key）**



建表person、book

**Person表建立：**

create table person (

pid varchar2(15) ,

name varchar2(18) not null,

age number(3) not null ,

birthday date,

sex varchar2(10) default '男' ,

address varchar2(18),

constraint person\_pid\_pk primary key(pid),

constraint person\_name\_uk unique(name),

constraint person\_age\_ck check(age between 0 and 150),

constraint person\_sex\_ck check(sex in('男','女','中'))

);



**Book表建立：**

create table book(

bid number(10) ,

bname varchar(15),

bprice number(5,2),

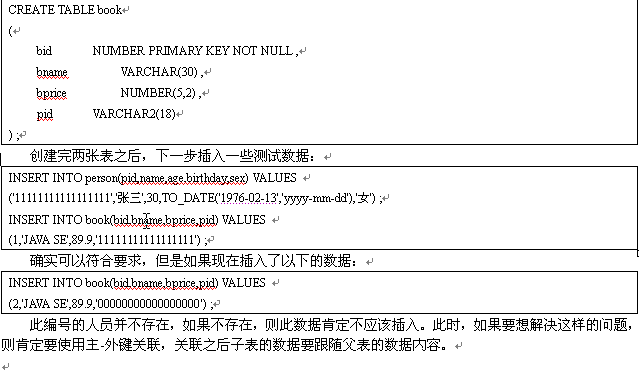
constraint book\_pid\_pk primary key(bid)

);



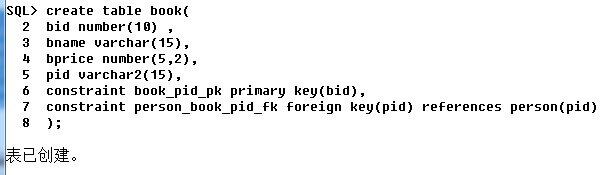
现在要在book表加新的列pid：属于谁的书

**推敲过程：**



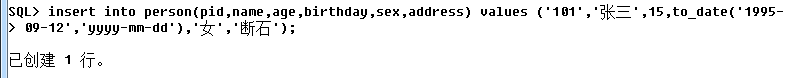
以上可知：推出person表为父表、book表为字表

**所以应该这样建立表book：**



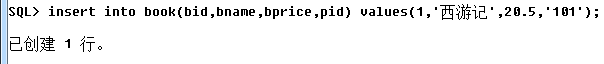
然后插入person表数据：

insert into person(pid,name,age,birthday,sex,address) values ('101','张三',15,to\_date('1995-09-12','yyyy-mm-dd'),'女','断石');



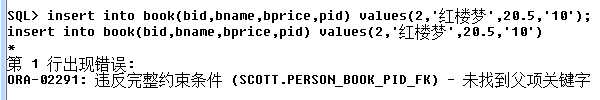
再插入book数据（pid与person表的pid对应）：

insert into book(bid,bname,bprice,pid) values(1,'西游记',20.5,'101');

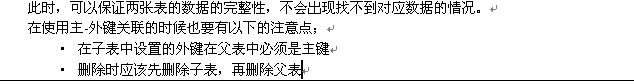


再插入book数据（pid与person表的pid不对应）：

insert into book(bid,bname,bprice,pid) values(2,'红楼梦',20.5,'10');

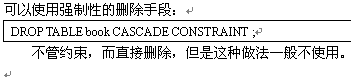


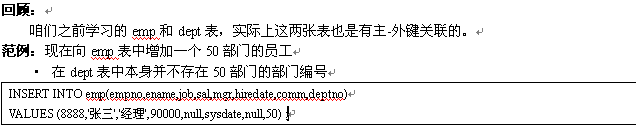
从错误信息：ORA-02291: 违反完整约束条件 (SCOTT.PERSON\_BOOK\_PID\_FK) - 未找到父项关键字 **可以看出创建的person\_book外键pid关联成功。**

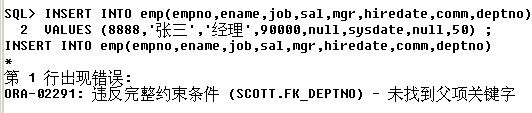


如果不按顺序删除则出现：

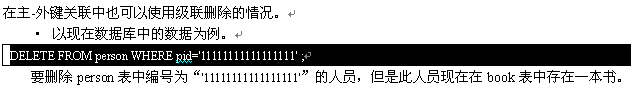


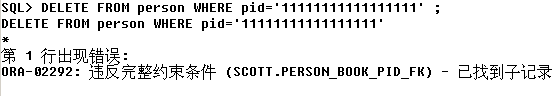


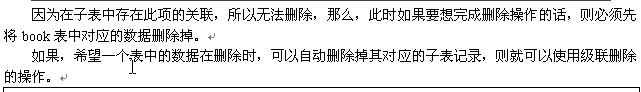












**级联删除需要在建立外键是就设置成级联模式：**

create table person (

pid varchar2(15) ,

name varchar2(18) not null,

age number(3) not null ,

birthday date,

sex varchar2(10) default '男' ,

address varchar2(18),

constraint person\_pid\_pk primary key(pid),

constraint person\_name\_uk unique(name),

constraint person\_age\_ck check(age between 0 and 150),

constraint person\_sex\_ck check(sex in('男','女','中'))

);

create table book(

bid number(10) ,

bname varchar(15),

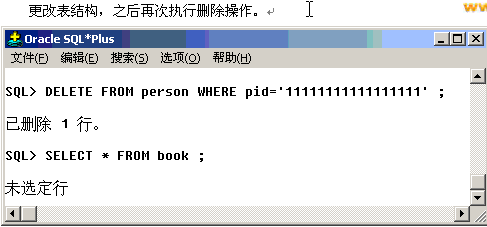
bprice number(5,2),

pid varchar2(15),

constraint book\_pid\_pk primary key(bid),

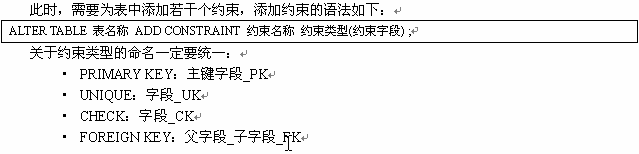
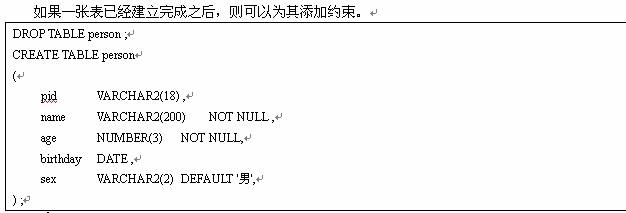
constraint person\_book\_pid\_fk foreign key(pid) references person(pid) **on delete cascade**

);



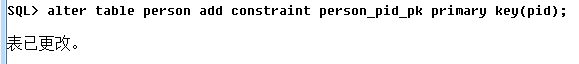
**修改约束、ROWNUM**

**修改约束（了解就好）**

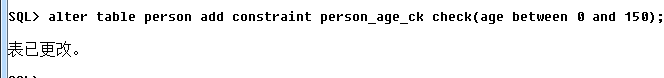




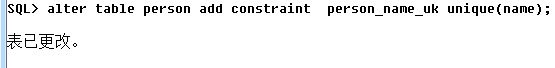
alter table person add constraint person\_pid\_pk primary key(pid);



alter table person add constraint person\_age\_ck check(age between 0 and 150);



alter table person add constraint person\_name\_uk unique(name);



为book表增加主键、外键约束

create table book(

bid number(10) ,

bname varchar(15),

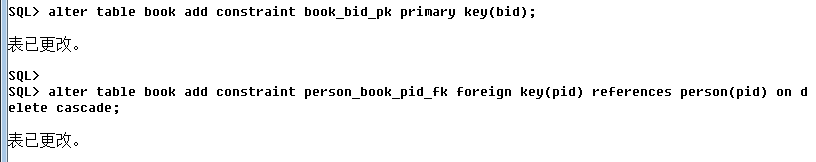
bprice number(5,2),

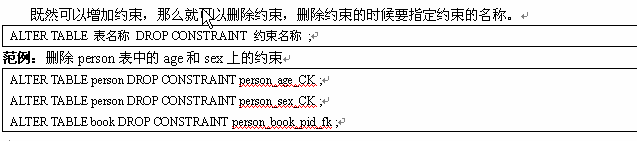
pid varchar2(15)

);

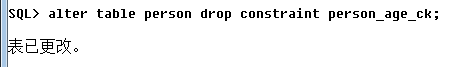
alter table book add constraint book\_bid\_pk primary key(bid);

alter table book add constraint person\_book\_pid\_fk foreign key(pid) references person(pid) on delete cascade;





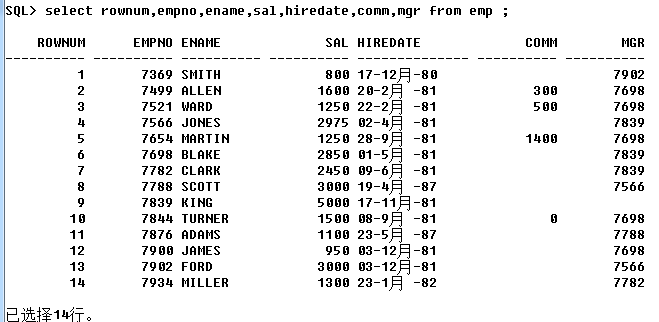
alter table person drop constraint person\_age\_ck;



**ROWNUM(重点)**



select rownum,empno,ename,sal,hiredate,comm,mgr from emp ;

即：给行编号

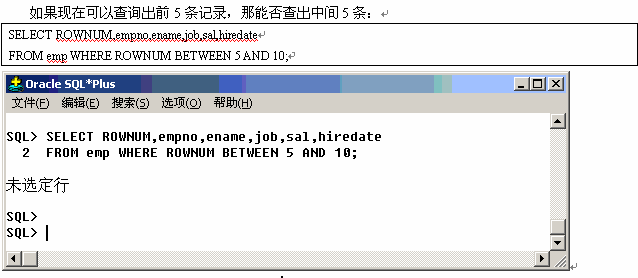


select rownum,empno,ename,sal,hiredate,comm,mgr

from emp

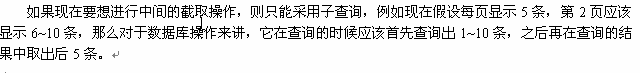
where rownum<=5 ;

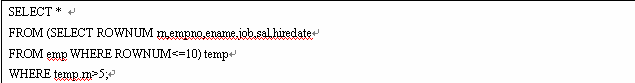


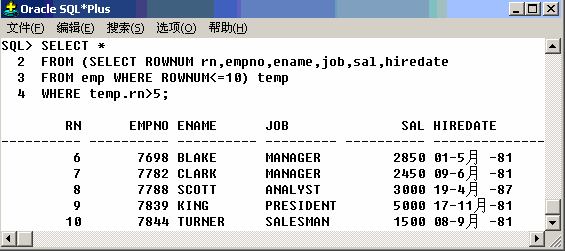


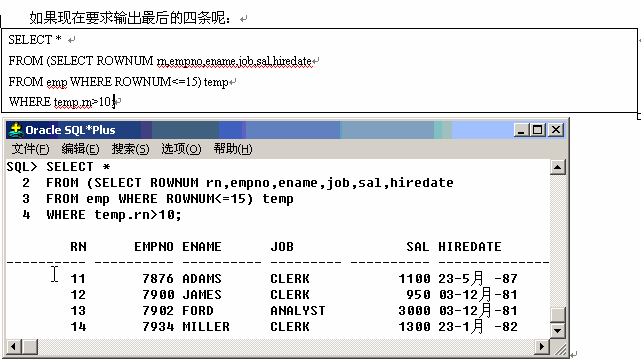
所以不行

**再用以下方法：**

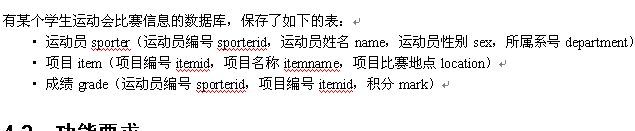


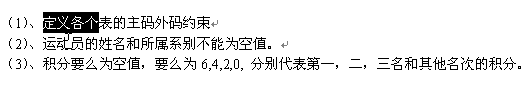


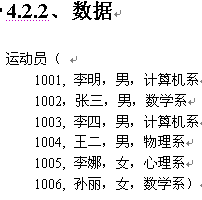
1

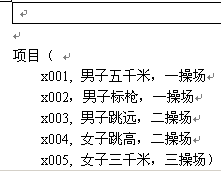


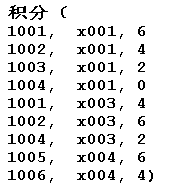
**建立、约束、查询全流程练习**











建表

**Sporter表：**

create table sporter (

sporterid number(4),

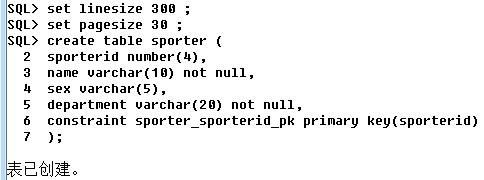
name varchar(10) not null,

sex varchar(5),

department varchar(20) not null,

constraint sporter\_sporterid\_pk primary key(sporterid)

);



**Item表：**

create table item (

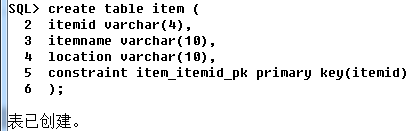
itemid varchar(4),

itemname varchar(10),

location varchar(10),

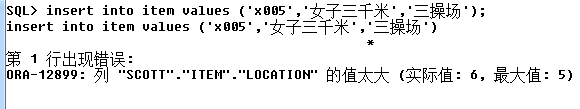
constraint item\_itemid\_pk primary key(itemid)

);



Location一开始设置大小‘5’

出现错误:



所以可以看出一个汉字：至少varchar(2) 3个汉字至少varchar(6)

**Grade表：**

create table grade(

sporterid number(4),

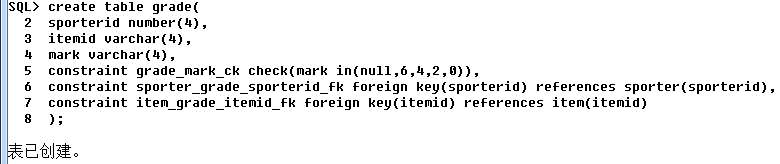
itemid varchar(4),

mark varchar(4),

constraint grade\_mark\_ck check(mark in(null,6,4,2,0)),

constraint sporter\_grade\_sporterid\_fk foreign key(sporterid) references sporter(sporterid),

constraint item\_grade\_itemid\_fk foreign key(itemid) references item(itemid)

);

**加入数据**



insert into sporter values(1001,'李明','男','计算机系');

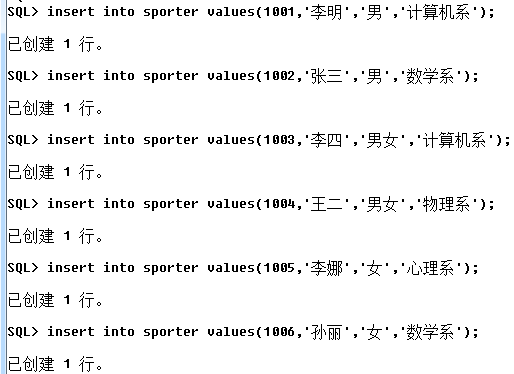
insert into sporter values(1002,'张三','男','数学系');

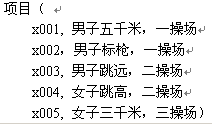
insert into sporter values(1003,'李四','男女','计算机系');

insert into sporter values(1004,'王二','男女','物理系');

insert into sporter values(1005,'李娜','女','心理系');

insert into sporter values(1006,'孙丽','女','数学系');





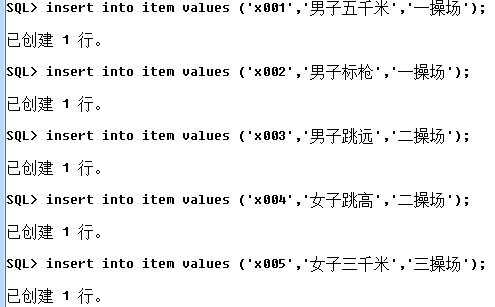
insert into item values ('x001','男子五千米','一操场');

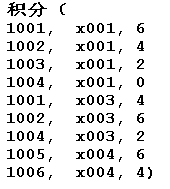
insert into item values ('x002','男子标枪','一操场');

insert into item values ('x003','男子跳远','二操场');

insert into item values ('x004','女子跳高','二操场');

insert into item values ('x005','女子三千米','三操场');





insert into grade values(1001,'x001','6');

insert into grade values(1002,'x001','4');

insert into grade values(1003,'x001','2');

insert into grade values(1004,'x001','0');

insert into grade values(1001,'x003','4');

insert into grade values(1002,'x003','6');

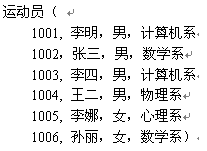
insert into grade values(1004,'x003','2');

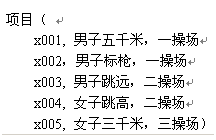
insert into grade values(1005,'x004','6');

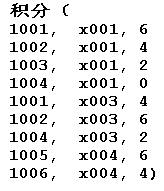
insert into grade values(1006,'x004','4');



**查询**







**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

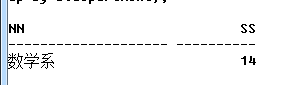


**方法一：**

select ed.nn,ed.ss

from (select s.department nn,sum(mark) ss from sporter s,grade g where s.sporterid=g.sporterid group by s.department) ed

where ed.ss=(select max(sum(mark)) ss from sporter s,grade g where s.sporterid=g.sporterid group by s.department);



**方法二：（用rownum）**

select \*

from(

select s.department nn,sum(mark) ss

from sporter s,grade g

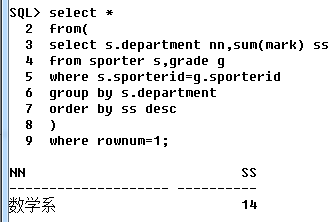
where s.sporterid=g.sporterid

group by s.department

order by ss desc

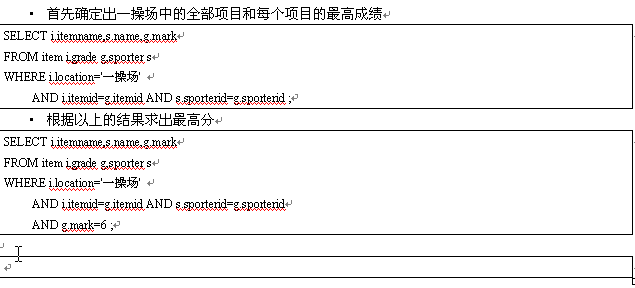
)

where rownum=1;





方法一：



方法二：

select b.itemname,d.name

from

(select itemid, max(mark) mm

from grade

group by itemid)a

,

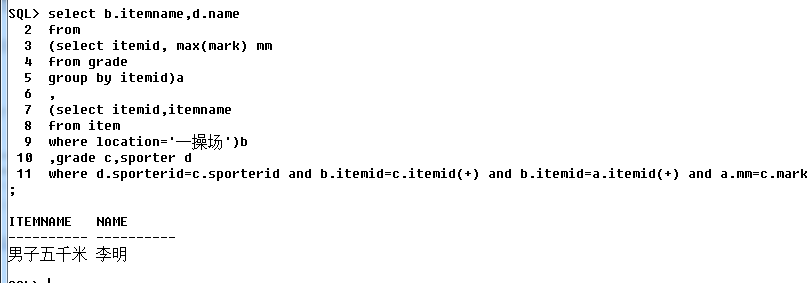
(select itemid,itemname

from item

where location='一操场')b

,grade c,sporter d

where d.sporterid=c.sporterid and b.itemid=c.itemid(+) and b.itemid=a.itemid(+) and a.mm=c.mark;

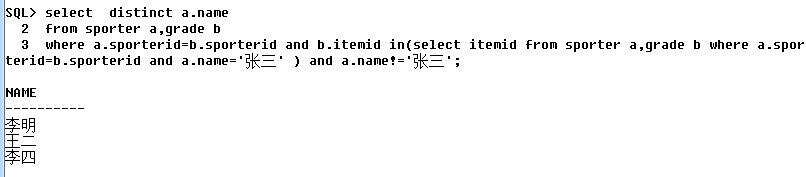




select distinct a.name

from sporter a,grade b

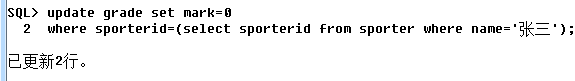
where a.sporterid=b.sporterid and b.itemid in(select itemid from sporter a,grade b where a.sporterid=b.sporterid and a.name='张三' ) and a.name!='张三';





update grade set mark=0

where sporterid=(select sporterid from sporter where name='张三');



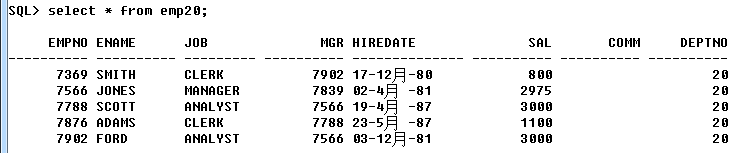


delete from item where itemname='女子跳高';





create table emp20 as select \* from emp where deptno=20;

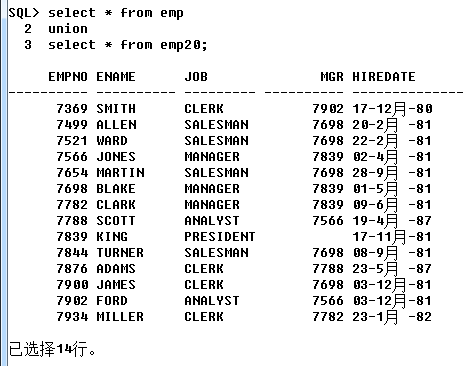


Union：

select \* from emp

union

select \* from emp20;

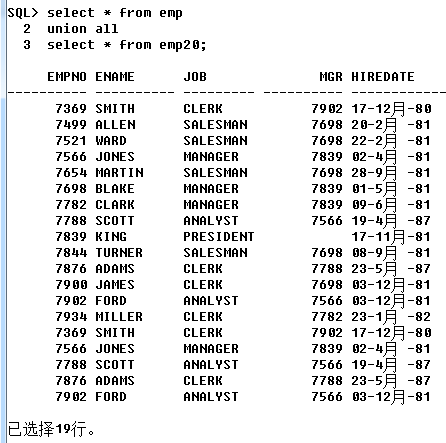


Union all：

select \* from emp

union all

select \* from emp20;

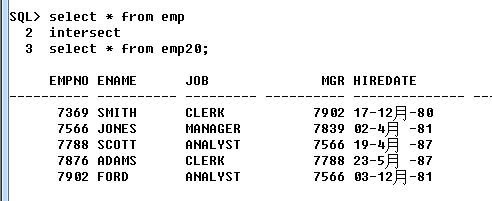


Intersect：两张表的交集

select \* from emp

intersect

select \* from emp20;

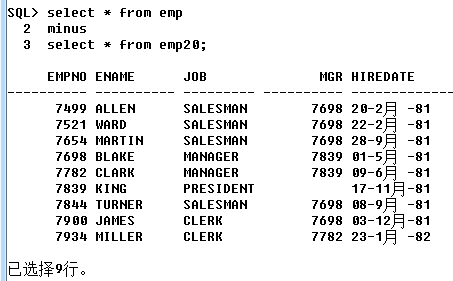


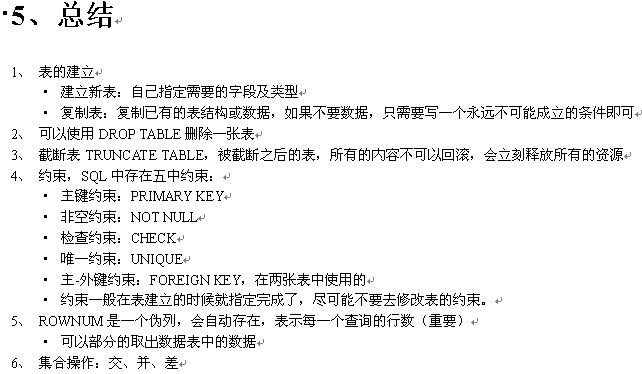
Minus：两张表的差集

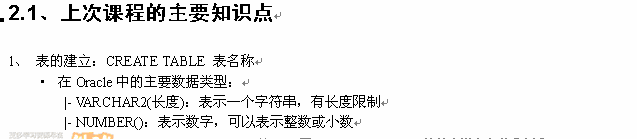
select \* from emp

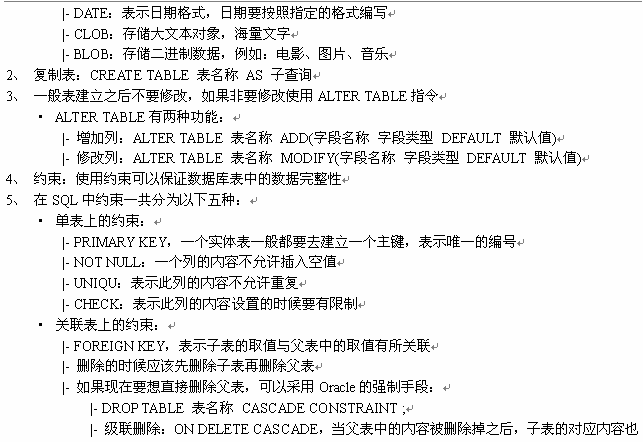
minus

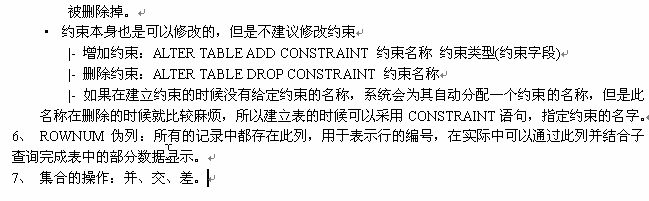
select \* from emp20;

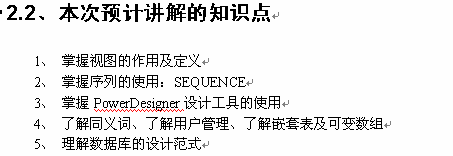




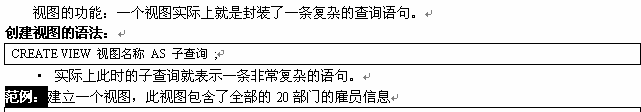








**视图**

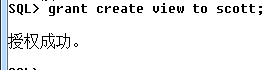


看你当前用户如果没有权限创建视图 用sys登录后 grant create view to 当前用户 再创建视图.

conn sys/change\_on\_install as sysdba;



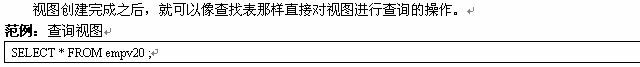
grant create view to scott;

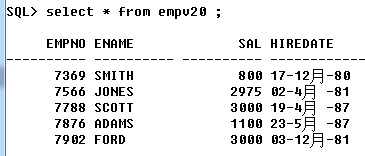


conn scott/tiger;

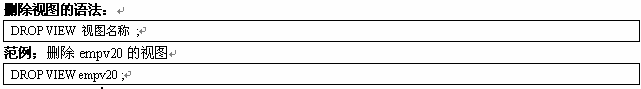
create view empv20 as select empno,ename,sal,hiredate from emp where deptno=20;



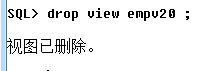




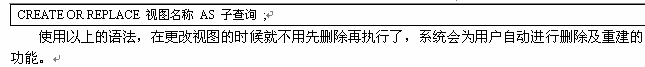




drop view empv20 ;



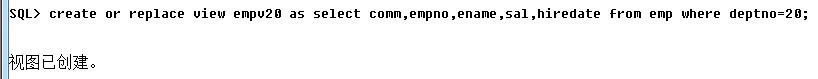


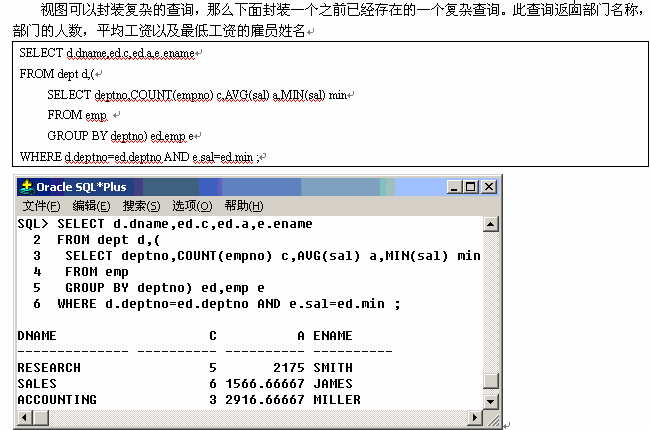


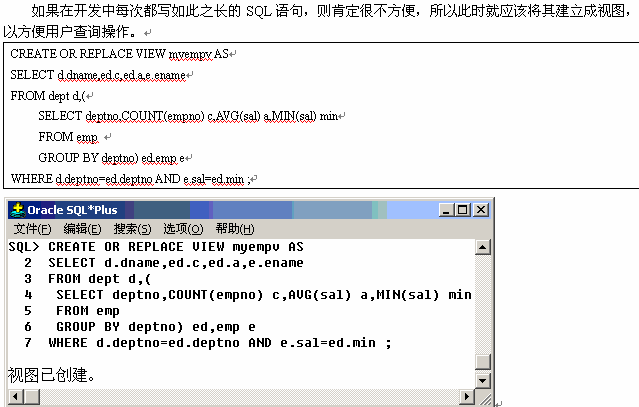
create view empv20 as select empno,ename,sal,hiredate from emp where deptno=20;

然后加一列comm（就要用create or replace）

create or replace view empv20 as select comm,empno,ename,sal,hiredate from emp where deptno=20;

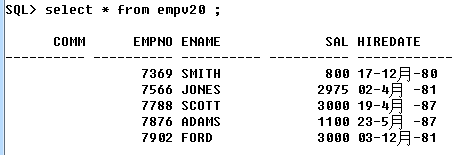




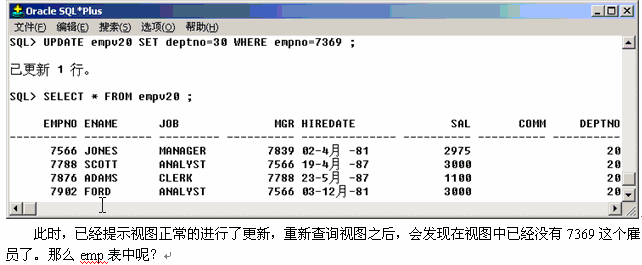


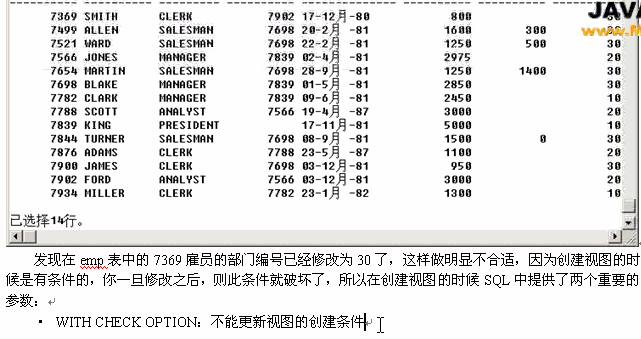


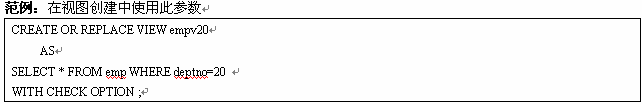


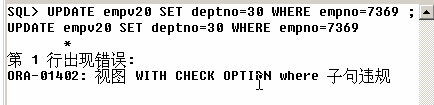












能改

**范例：创建只读视图**

**create or replace view empv20**

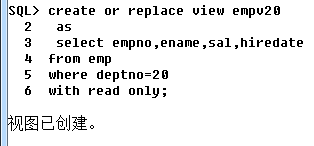
**as**

**select empno,ename,sal,hiredate**

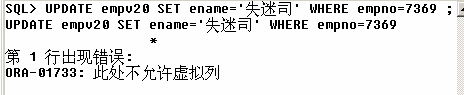
**from emp**

**where deptno=20**

**with read only;**

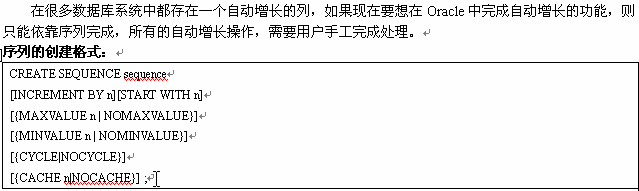


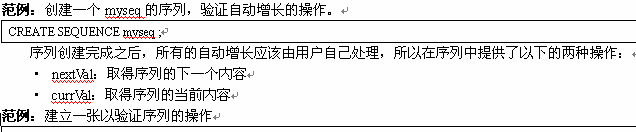
**然后修改试试：**



提示只读视图

**序列**





create sequence myseq ;

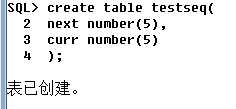


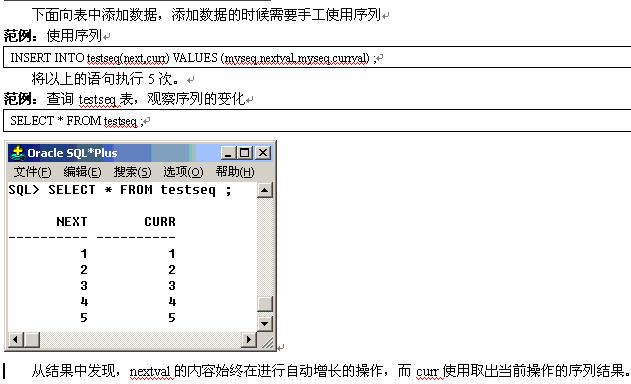
create table testseq(

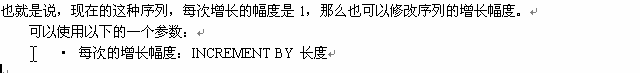
next number(5),

curr number(5)

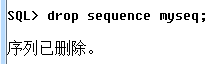
);







drop sequence myseq;

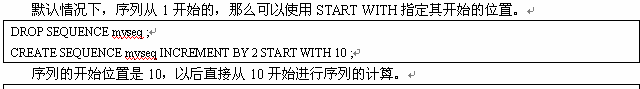


create sequence myseq

increment by 2;



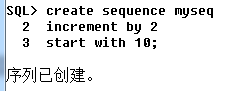




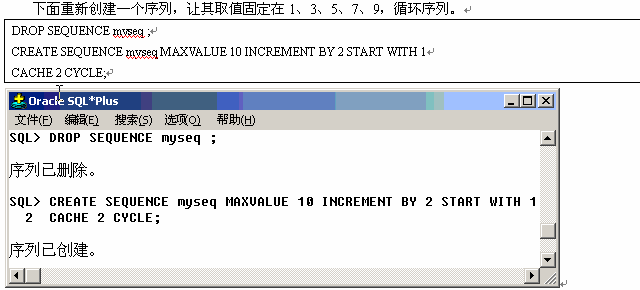
create sequence myseq

increment by 2

start with 10;



从10开始，跨度为2



最大值10 1开始 跨度2 循环模式 缓存为2

**一个工具：PL/SQL Developer**

**同义词**

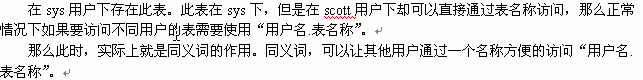




通过查询语句：select \* from tab where TNAME=’DUAL’;

在scott和system用户下没找到dual表

在sys用户下找到了dual表





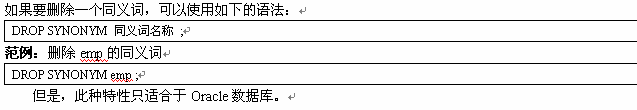
切换到sys用户

conn sys/change\_on\_insall as sysdba ;

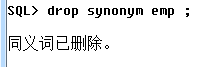
创建同义词emp

create synonym emp for scott.emp;

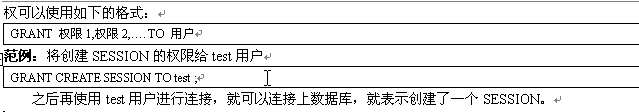
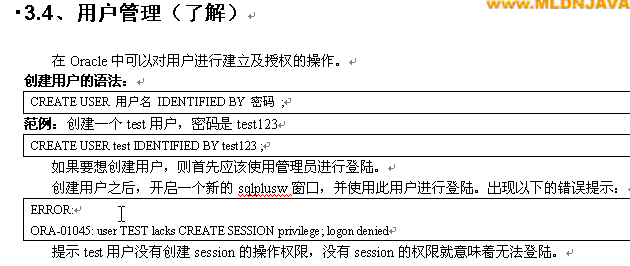


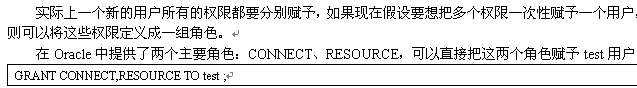


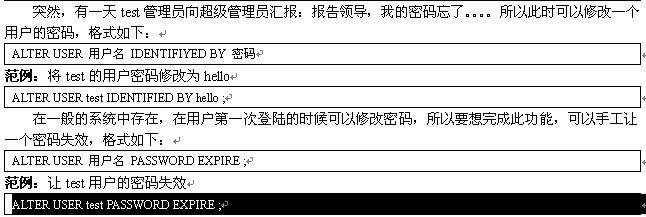
drop synonym emp ;



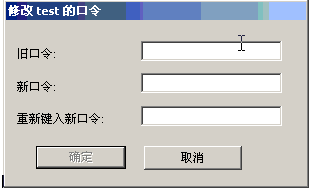
**用户管理、数据库的备份与恢复**

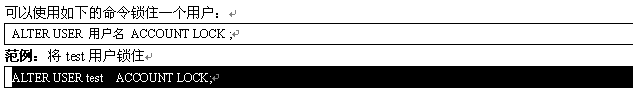


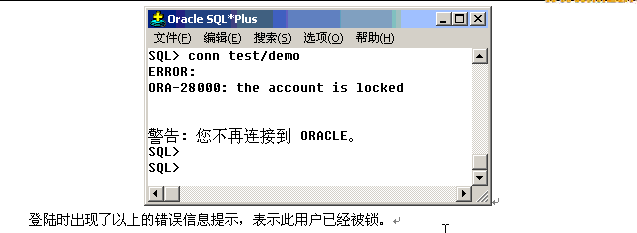


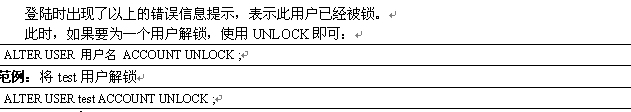


**出现提示框：**

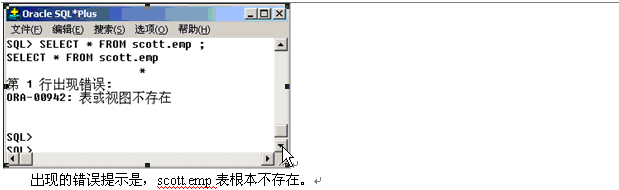


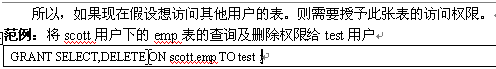




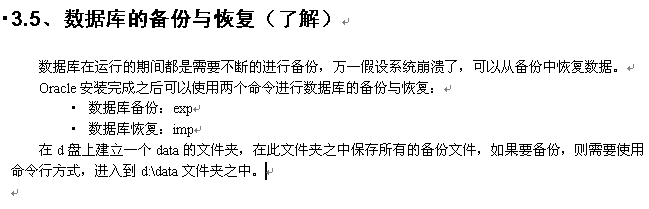












备份数据库

先在d盘下建立一个文件夹data

Win+r

Cmd

D:

Cd data

Exp

倒出来源的用户的用户名（如：scott）

倒出来源的用户的密码（如：tiger）

然后一直默认

恢复数据库

假设把scott用户下的表全部删除了；然后想恢复数据库数据

Win+r

Cmd

D:

Cd data

Imp

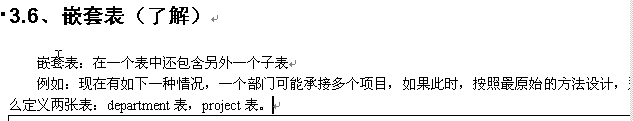
用户名（如：scott）

密码（如：tiger）

然后一直默认

**嵌套表、可变数组（都做了解）**

**嵌套表（oracle中有，作了解）**



create table department(

did number(2) ,

dname varchar(20),

constraint department\_did\_pk primary key (did)

);

create table project(

proid number(4),

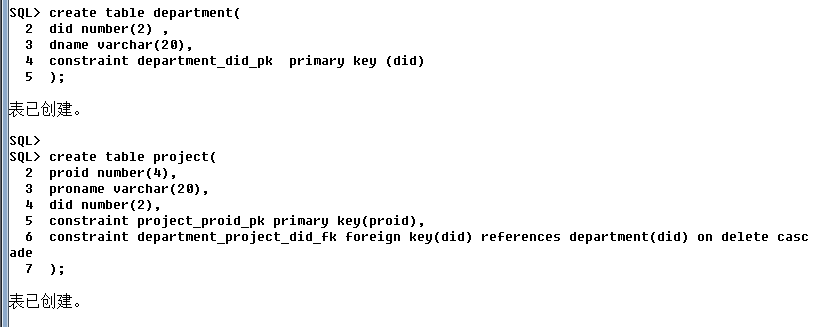
proname varchar(20),

did number(2),

constraint project\_proid\_pk primary key(proid),

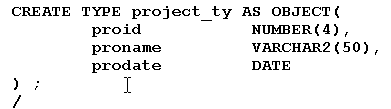
constraint department\_project\_did\_fk foreign key(did) references department(did) on delete cascade

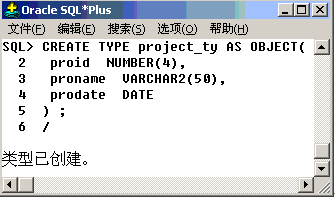
);

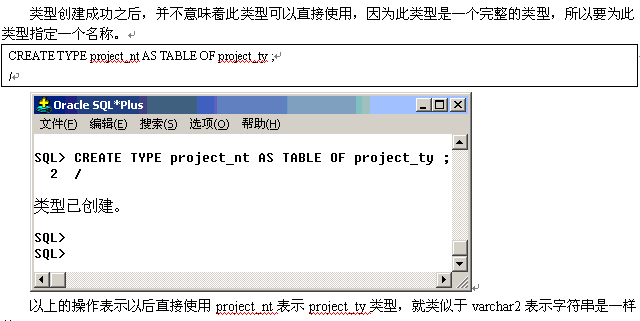




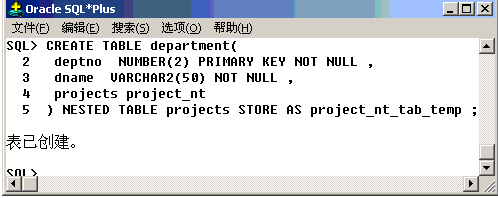




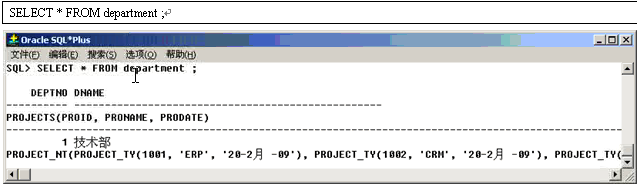
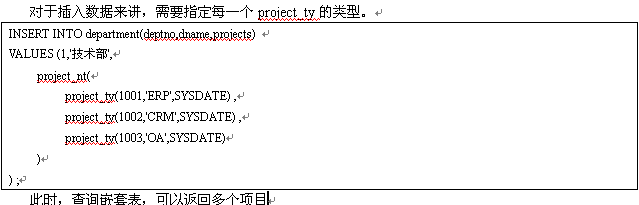




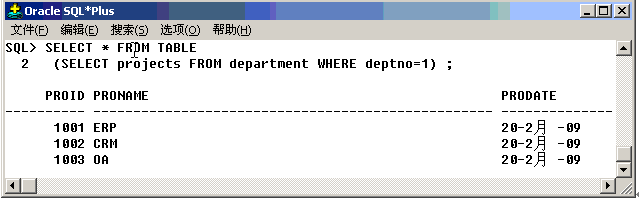
以下是完成嵌套表



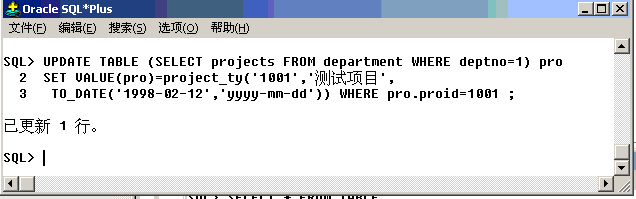


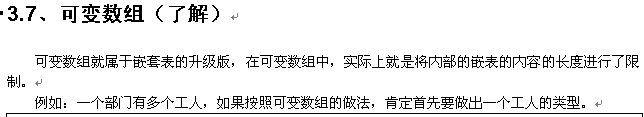


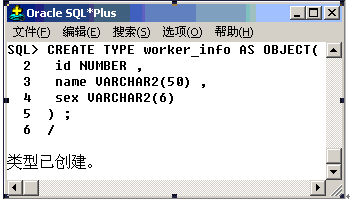




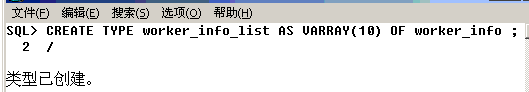




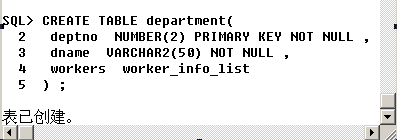








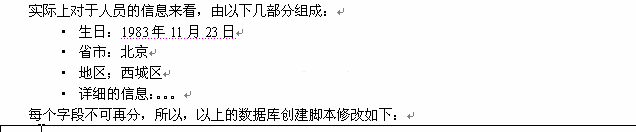
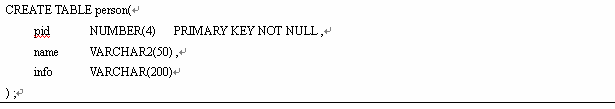
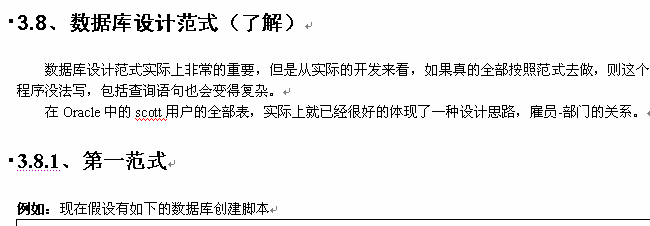




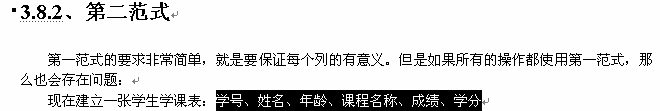


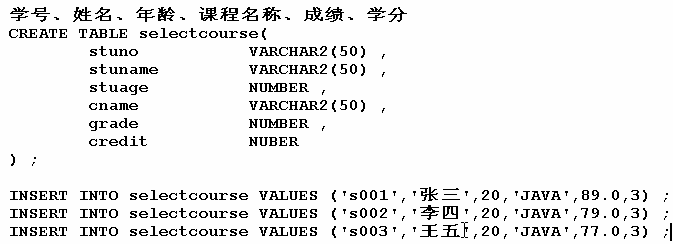


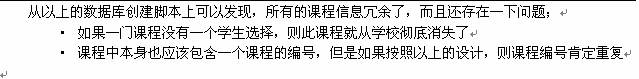
**数据库设计范式 PowerDesigner工具**

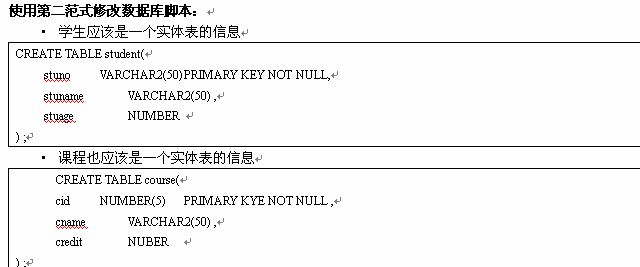


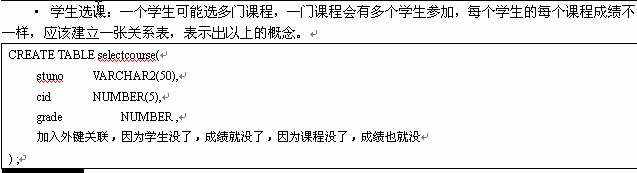


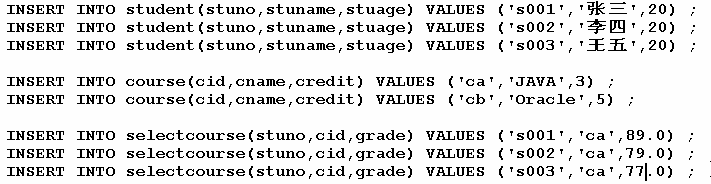


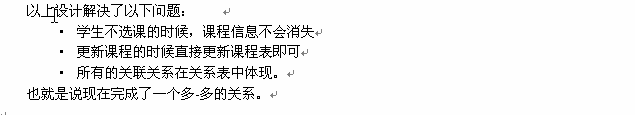


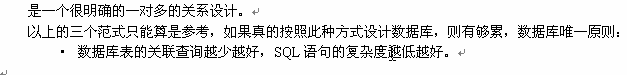


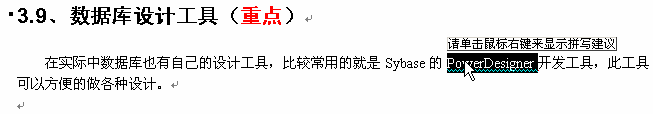




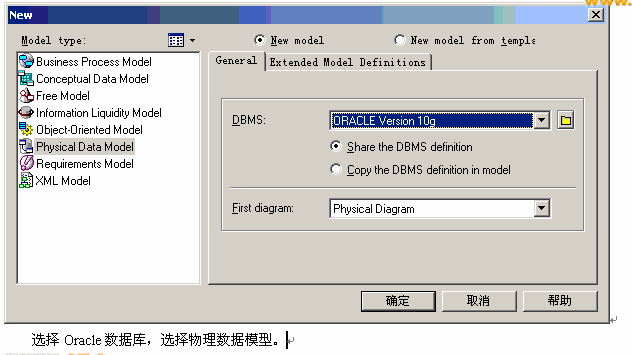


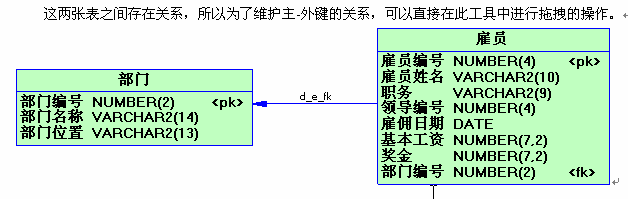
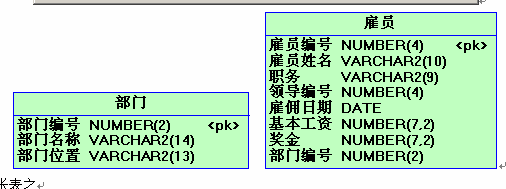




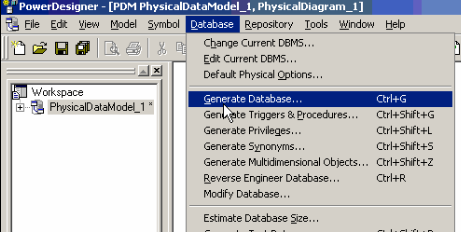








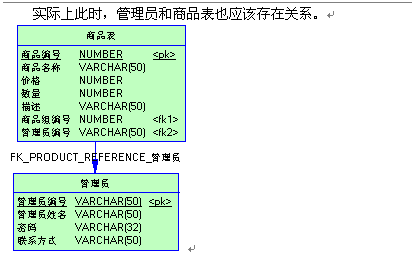
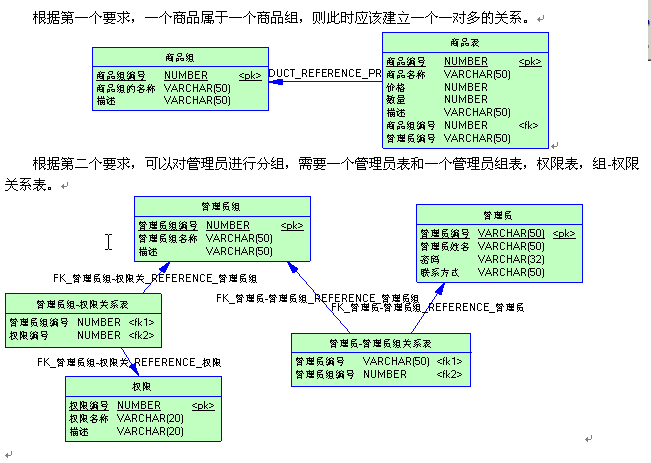
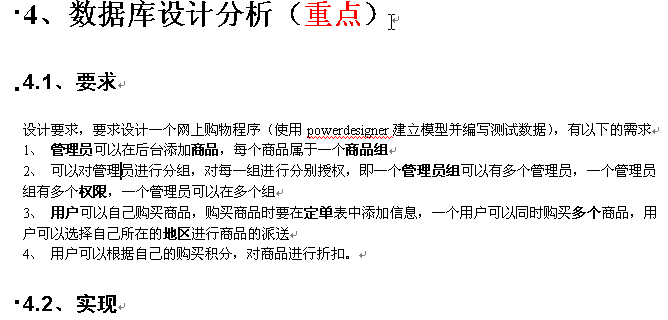
生成建表语句：



生成插入语句：



数据库设计分析



总结：比如：一个管理员属于一个管理员组；一个管理员组可以有多个管理员

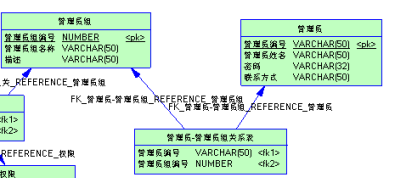
这样形成一对多的的情况

需要：三个表1.管理员表

2.管理员组表

3.组-员关系表

来表示他们一对多的关系



**总结**

