注意：操作系统管理员 root，密码 oracle
操作系统一般用户 oracle，密码 oracle

数据库管理员 sys，密码 oracle
数据库一般管理员 system，密码 oracle
数据库测试用户 scott，密码 tiger
数据库测试用户 hr，密码 hr

1.显示表的结构（其中 describe 可以缩写成 desc）

```
[oracle@rhel ~]$ sqlplus scott/tiger

SQL*Plus: Release 10.2.0.1.0 - Production on Tue May 17 23:34:32 2011

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options

SQL> show user
USER is "SCOTT"
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
DEPT                           TABLE
EMP                            TABLE
BONUS                          TABLE
SALGRADE                       TABLE

SQL> desc emp;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                     NOT NULL NUMBER(4)
 ENAME                                              VARCHAR2(10)
 JOB                                                VARCHAR2(9)
 MGR                                                NUMBER(4)
 HIREDATE                                           DATE
 SAL                                                NUMBER(7,2)
 COMM                                               NUMBER(7,2)
 DEPTNO                                             NUMBER(2)

SQL>
```

补充：【数据类型】

Number(p,s)：数字类型，p 表示数字的有效长度(从数字的左边第 1 位不为 0 的开始算起，直到最右边的长度；取值范围 0~38 位)，s 表示数字的精度(即小数点右边的位数，取值范围-84~127 位)；

Varchar2(s)：可变长的字符类型，s 表示字符串的长度，取值范围 1~4000 位；

Char(s)：定长的字符类型，s 表示字符串的长度，取值范围 1~2000 位；

Date：时间类型，表示时间的年月日，没有长度和精度，取值范围公元前 4713 年 12 月 31 日~公元后 9999 年 12 月 31 日；

2.在 SQL>命令行中修改 SQL 语句

2.1 a[ppend] text，表示在添加文本到当前行的末尾；

```
SQL> show user
USER is "SCOTT"
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
DEPT                           TABLE
EMP                            TABLE
BONUS                          TABLE
SALGRADE                       TABLE

SQL> a  where tname = 'DEPT';
  1* select * from tab where tname = 'DEPT'
SQL>
```

注意：在 a 后面需添加 2 个半角空格；

2.2 c[hange] /old/new，表示在当前行中将*旧*文本改成*新*文本；

```
SQL> select * from tab where tname = 'DEPT';

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
DEPT                           TABLE

SQL> c /DEPT/EMP
  1* select * from tab where tname = 'EMP'
SQL>
```

c[hange] /old/，表示从当前行中删除文本；

```
SQL> select * from tab where tname = 'EMP';

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
EMP                            TABLE

SQL> c /where tname = 'EMP'/
  1* select * from tab
SQL>
```

2.3 l[ist]，表示列举出缓冲区中所有的行；

```
SQL> select deptno,sum(sal),count(*),avg(sal)
  2  from emp
  3  group by deptno
  4  order by deptno;

    DEPTNO   SUM(SAL)   COUNT(*)   AVG(SAL)
---------- ---------- ---------- ----------
        10       8750          3 2916.66667
        20      10875          5       2175
        30       9400          6 1566.66667

SQL> l
  1  select deptno,sum(sal),count(*),avg(sal)
  2  from emp
  3  group by deptno
  4* order by deptno
SQL>
```

l[ist] n,表示列举出缓冲区中的第 n 行；

```
SQL> l
  1  select deptno,sum(sal),count(*),avg(sal)
  2  from emp
  3  group by deptno
  4* order by deptno
SQL> l 3
  3* group by deptno
SQL>
```

l[ist] m n,表示列举出缓冲区中的第 m 行到第 n 行；

```
SQL> l
  1  select deptno,sum(sal),count(*),avg(sal)
  2  from emp
  3  group by deptno
  4* order by deptno
SQL> l 1 3
  1  select deptno,sum(sal),count(*),avg(sal)
  2  from emp
  3* group by deptno
SQL>
```

注意：*标注的行，表示当前活动行，如果修改时不指定哪一行的话，则修改默认的指定行；

```
SQL> l
  1  select deptno,sum(sal),count(*),avg(sal)
  2  from emp
  3  group by deptno
  4* order by deptno
SQL> c /deptno/1
  4* order by 1
SQL> l
  1  select deptno,sum(sal),count(*),avg(sal)
  2  from emp
  3  group by deptno
  4* order by 1
SQL>
```

2.4 n，表示将指定行标记为活动行；

```
SQL> l
  1  select deptno,sum(sal),count(*),avg(sal)
  2  from emp
  3  group by deptno
  4* order by 1
SQL> 2
  2* from emp
SQL>
```

n text，表示用文本代替第 n 行；

```
SQL> l
  1  select deptno,sum(sal),count(*),avg(sal)
  2  from emp
  3  group by deptno
  4* order by 1
SQL> 2 haha
SQL> l
  1  select deptno,sum(sal),count(*),avg(sal)
  2  haha
  3  group by deptno
  4* order by 1
SQL>
```

0 text，表示在第 1 行之前添加 1 行文本

```
SQL> l
  1  select deptno,sum(sal),count(*),avg(sal)
  2  from emp
  3  group by deptno
  4* order by 1
SQL> 0 haha
SQL> l
  1  haha
  2  select deptno,sum(sal),count(*),avg(sal)
  3  from emp
  4  group by deptno
  5* order by 1
SQL>
```

2.5 del，表示删除当前活动行；

```
SQL> l
  1  select deptno,sum(sal),count(*),avg(sal)
  2  from emp
  3  group by deptno
  4* order by 1
SQL> del
SQL> l
  1  select deptno,sum(sal),count(*),avg(sal)
  2  from emp
  3* group by deptno
SQL>
```

del n，表示删除第 n 行；

```
SQL> l
  1  select deptno,sum(sal),count(*),avg(sal)
  2  from emp
  3  group by deptno
  4* order by 1
SQL> del 3
SQL> l
  1  select deptno,sum(sal),count(*),avg(sal)
  2  from emp
  3* order by 1
SQL>
```

del m n，表示删除第 m 行至第 n 行；

```
SQL> l
  1  select deptno,sum(sal),count(*),avg(sal)
  2  from emp
  3  group by deptno
  4* order by 1
SQL> del 3 4
SQL> l
  1  select deptno,sum(sal),count(*),avg(sal)
  2* from emp
SQL>
```

注意：如果想删除所有的行，那该怎么办？

```
SQL> l
  1  select deptno,sum(sal),count(*),avg(sal)
  2  from emp
  3  group by deptno
  4* order by 1
SQL> del 1 4
SQL> l
SP2-0223: No lines in SQL buffer.
SQL>
```

2.6 input，表示在指定的活动行后面添加若干行；（例：在第 2 行后添加 2 行，结尾处回车即可）

```
SQL> l
  1  select deptno,sum(sal),count(*),avg(sal)
  2  from emp
  3  group by deptno
  4* order by 1
SQL> 2
  2* from emp
SQL> i
  3i where deptno = 10
  4i and deptno <> 20
  5i
SQL> l
  1  select deptno,sum(sal),count(*),avg(sal)
  2  from emp
  3  where deptno = 10
  4  and deptno <> 20
  5  group by deptno
  6* order by 1
SQL>
```

input text，表示在指定的活动行后面添加 1 行文本；（例：在第 2 行后添加 1 行文本）

```
SQL> l
  1  select deptno,sum(sal),count(*),avg(sal)
  2  from emp
  3  group by deptno
  4* order by 1
SQL> 2
  2* from emp
SQL> i where deptno = 10 and deptno <> 20
SQL> l
  1  select deptno,sum(sal),count(*),avg(sal)
  2  from emp
  3  where deptno = 10 and deptno <> 20
  4  group by deptno
  5* order by 1
SQL>
```

2.7 cl[ear] buff[er]，表示从 SQL 缓冲区中删除所有行；

```
SQL> l
  1  select deptno,sum(sal),count(*),avg(sal)
  2  from emp
  3  group by deptno
  4* order by 1
SQL> cl buff
buffer cleared
SQL> l
SP2-0223: No lines in SQL buffer.
SQL>
```

2.8 r[un]，表示执行缓冲区中的 SQL 语句；

```
SQL> l
  1  select deptno,sum(sal),count(*),avg(sal)
  2  from emp
  3  group by deptno
  4* order by 1
SQL> run
  1  select deptno,sum(sal),count(*),avg(sal)
  2  from emp
  3  group by deptno
  4* order by 1

    DEPTNO   SUM(SAL)   COUNT(*)   AVG(SAL)
---------- ---------- ---------- ----------
        10       8750          3 2916.66667
        20      10875          5       2175
        30       9400          6 1566.66667

SQL>
```

注意：也可以用/代替 run，也表示执行缓冲区中的 SQL 语句；

```
SQL> l
  1  select deptno,sum(sal),count(*),avg(sal)
  2  from emp
  3  group by deptno
  4* order by 1
SQL> /

    DEPTNO   SUM(SAL)   COUNT(*)   AVG(SAL)
---------- ---------- ---------- ----------
        10       8750          3 2916.66667
        20      10875          5       2175
        30       9400          6 1566.66667

SQL>
```

2.9 !，表示临时退出 SQL>命令行，回到操作系统命令行；在操作系统命令行中输入 exit 后，将会退回到 SQL>命令行；

```
SQL> show user
USER is "SCOTT"
SQL> !
[oracle@rhel ~]$ exit
exit

SQL>
```

exit，表示永久退出 SQL>命令行，回到操作系统命令行；在操作系统中必须重新连接数据库中的用户，才能回到 SQL>命令行；

```
SQL> show user
USER is "SCOTT"
SQL> exit
Disconnected from Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
[oracle@rhel ~]$ sqlplus scott/tiger

SQL*Plus: Release 10.2.0.1.0 - Production on Wed May 18 00:35:17 2011

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options

SQL> show user
USER is "SCOTT"
SQL>
```

2.10 sav[e] *filename* [rep[lace] app[end]],表示将缓冲区中的 SQL 语句保存到某个文件(可指定路径)，其中 replace 表示新的内容将覆盖原来的内容；append 表示在原有内容的基础上向后追加；默认的文件扩展名.sql

注意：/u01，是操作系统中的主文件夹；

cd */filename*，表示打开某个路径下的文件夹；

ls，表示列举当前文件夹底下的子文件夹以及文件；

cd ..表示退回到上一级目录；

cd \ 回车，表示退回到根目录；

more filename，表示获取该文件中内容(须为文本文件)

```
[oracle@rhel ~]$ cd /u01
[oracle@rhel u01]$ ls
app  Disk1  oradata
[oracle@rhel u01]$
```

例：保存文件内容　　SQL>sav *filename*

```
SQL> show user
USER is "SCOTT"
SQL> l
SP2-0223: No lines in SQL buffer.
SQL> select * from dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

SQL> l
  1* select * from dept
SQL> sav /u01/haha.sql
Created file /u01/haha.sql
SQL> !
[oracle@rhel ~]$ cd /u01
[oracle@rhel u01]$ ls
app  Disk1  haha.sql  oradata
[oracle@rhel u01]$ more haha.sql
select * from dept
/
[oracle@rhel u01]$
```

例：覆盖内容　　SQL>sav *filename rep*

```
SQL> show user
USER is "SCOTT"
SQL> l
  1* select * from dept
SQL> select deptno,dname,loc from dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

SQL> l
  1* select deptno,dname,loc from dept
SQL> sav /u01/haha.sql rep
Wrote file /u01/haha.sql
SQL> !
[oracle@rhel ~]$ cd /u01
[oracle@rhel u01]$ ls
app  Disk1  haha.sql  oradata
[oracle@rhel u01]$ more haha.sql
select deptno,dname,loc from dept
/
[oracle@rhel u01]$
```

例：追加内容　　SQL>sav *filename app*

```
[oracle@rhel u01]$ more haha.sql
select deptno,dname,loc from dept
/
[oracle@rhel u01]$ exit
exit

SQL> select empno,ename,deptno,sal from emp
  2  where empno = 7788;

     EMPNO ENAME          DEPTNO        SAL
---------- ---------- ---------- ----------
      7788 SCOTT              20       3000

SQL> sav /u01/haha.sql app
Appended file to /u01/haha.sql
SQL> !
[oracle@rhel ~]$ cd /u01
[oracle@rhel u01]$ ls
app  Disk1  haha.sql  oradata
[oracle@rhel u01]$ more haha.sql
select deptno,dname,loc from dept
/
select empno,ename,deptno,sal from emp
where empno = 7788
/
[oracle@rhel u01]$
```

2.11 get *filename*，表示将操作系统下的文件内容读取到 SQL 命令行中；

```
[oracle@rhel u01]$ more haha.sql
select deptno,dname,loc from dept
/
select empno,ename,deptno,sal from emp
where empno = 7788
/
[oracle@rhel u01]$ exit
exit

SQL> get /u01/haha.sql
  1  select deptno,dname,loc from dept
  2  /
  3  select empno,ename,deptno,sal from emp
  4* where empno = 7788
SQL>
```

2.12 start *filename*，表示在 SQL 命令行中运行操作系统下的文件中的 SQL 命令；

```
SQL> start /u01/haha.sql

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON


     EMPNO ENAME          DEPTNO        SAL
---------- ---------- ---------- ----------
      7788 SCOTT              20       3000

SQL>
```

注意：也可以@*filename* 去执行操作系统下的文件中的 SQL 命令；

```
SQL> @/u01/haha.sql

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON


     EMPNO ENAME          DEPTNO        SAL
---------- ---------- ---------- ----------
      7788 SCOTT              20       3000

SQL>
```

2.13 spool on，表示将缓存打开，不然只能保存最近执行的 1 条命令；

spool *filename*，表示将缓存中出现的命令以及结果输出到某个文件中；

spool off，表示关闭缓存，同时文件会自动保存；

```
SQL> show user
USER is "SCOTT"
SQL> spool on
SQL> spool /u01/hehe.log
SQL> select empno,ename,sal from emp;

     EMPNO ENAME             SAL
---------- ---------- ----------
      7369 SMITH             800
      7499 ALLEN            1600
      7521 WARD             1250
      7566 JONES            2975
      7654 MARTIN           1250
      7698 BLAKE            2850
      7782 CLARK            2450
      7788 SCOTT            3000
      7839 KING             5000
      7844 TURNER           1500
      7876 ADAMS            1100

     EMPNO ENAME             SAL
---------- ---------- ----------
      7900 JAMES             950
      7902 FORD             3000
      7934 MILLER           1300

14 rows selected.

SQL> spool off
SQL>
```

查看文件的内容：

```
[oracle@rhel ~]$ cd /u01
[oracle@rhel u01]$ ls
app  Disk1  haha.sql  hehe.log  oradata
[oracle@rhel u01]$ more hehe.log
SQL> select empno,ename,sal from emp;

     EMPNO ENAME             SAL
---------- ---------- ----------
      7369 SMITH             800
      7499 ALLEN            1600
      7521 WARD             1250
      7566 JONES            2975
      7654 MARTIN           1250
      7698 BLAKE            2850
      7782 CLARK            2450
      7788 SCOTT            3000
      7839 KING             5000
      7844 TURNER           1500
      7876 ADAMS            1100

     EMPNO ENAME             SAL
---------- ---------- ----------
      7900 JAMES             950
      7902 FORD             3000
      7934 MILLER           1300

14 rows selected.

SQL> spool off
[oracle@rhel u01]$
```

2.14 设定会话中结果的显示格式：

set linesize 100，表示显示的行宽度，将结果显示时占的总宽度为 100 个字符宽宽的，数字可任意修改(正数)，不然可能显示结果过长而产生折行；

例：设定前

```
SQL> show user
USER is "SCOTT"
SQL> select * from emp where rownum <= 1;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM
---------- ---------- --------- ---------- --------- ---------- ----------
    DEPTNO
----------
      7369 SMITH      CLERK           7902 17-DEC-80        800
        20


SQL>
```

设定后

```
SQL> set linesize 150
SQL> /

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20

SQL>
```

set pagesize 30，表示每页记录数，即每页可以显示多少条数据，包括标题行，数字可任意修改(正数)，不然会产生翻页；

例：设定前

```
SQL> show user
USER is "SCOTT"
SQL> select * from emp;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10

14 rows selected.

SQL>
```

设定后

```
SQL> set pagesize 30
SQL> /

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10

14 rows selected.

SQL>
```

注意：以上 linesize，pagesize 的设定，是当前会话有效，如果会话重新登录，或者打开新的会话，则设定会还原成默认。

2.15 vi *filename*，表示新建一个文件，并对其中的内容进行编辑；

例：step 1：在操作系统的/u01 目录下新建一个文件 temp01.sql；

```
[oracle@rhel ~]$ cd /u01
[oracle@rhel u01]$ ls
app  Disk1  oradata
[oracle@rhel u01]$ vi temp01.sql

~
~
~
~
"temp01.sql" [New File]                                                    0,0
```

step 2：点击 i 按钮，转换文本为-insert-模式，此时可以在文本中书写内容；

```
[oracle@rhel ~]$ cd /u01
[oracle@rhel u01]$ ls
app  Disk1  oradata
[oracle@rhel u01]$ vi temp01.sql

~
~
~
~
-- INSERT --                                                               0,1
```

step 3：书写完毕后，点击 Esc 按钮，退出编辑模式；

```
select * from dept;

~
~
~
~
~
                                                                       2,0-1
```

step 4：输入:wq，然后回车，将会保存文件并退出，回到操作系统命令行；

注意：:q 表示退出不保存

:q! 表示强制退出不保存

```
select * from dept;

~
~
~
:wq
~
"temp01.sql" 2L, 21C written
[oracle@rhel u01]$ ls
app  Disk1  oradata  temp01.sql
[oracle@rhel u01]$
```

【第一章 编写基本的 SQL select 语句】

1.用户拥有的表相关信息的查询；

1.1 查询当前用户拥有对象的信息；

```
[oracle@rhel ~]$ sqlplus scott/tiger

SQL*Plus: Release 10.2.0.1.0 - Production on Wed May 18 02:06:01 2011

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options

SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
DEPT                           TABLE
EMP                            TABLE
BONUS                          TABLE
SALGRADE                       TABLE

SQL>
```

注意：tab 是数据字典，可以查询用户拥有对象的信息，具体表定义，可通过 desc 命令了解；

```
SQL> desc tab;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 TNAME                                     NOT NULL VARCHAR2(30)
 TABTYPE                                            VARCHAR2(7)
 CLUSTERID                                          NUMBER

SQL>
```

当然也可以通过 user_tables 进行查询；

```
SQL> select table_name,tablespace_name from user_tables;

TABLE_NAME                     TABLESPACE_NAME
------------------------------ ------------------------------
DEPT                           USERS
EMP                            USERS
BONUS                          USERS
SALGRADE                       USERS

SQL>
```

2. 查看表的具体信息，以及其中包含的数据；

2.1 查看表的结构；

```
SQL> desc emp
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                     NOT NULL NUMBER(4)
 ENAME                                              VARCHAR2(10)
 JOB                                                VARCHAR2(9)
 MGR                                                NUMBER(4)
 HIREDATE                                           DATE
 SAL                                                NUMBER(7,2)
 COMM                                               NUMBER(7,2)
 DEPTNO                                             NUMBER(2)

SQL> desc dept
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DEPTNO                                    NOT NULL NUMBER(2)
 DNAME                                              VARCHAR2(14)
 LOC                                                VARCHAR2(13)

SQL>
```

2.2 查看表中的数据；

注意：*表示显示一张表中所有列的定义信息；

```
SQL> set linesize 150
SQL> set pagesize 30
SQL> select * from emp;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10

14 rows selected.

SQL> select * from dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

SQL>
```

当然也可以显示一张表指定的某些列的信息；

```
SQL> select empno,ename,job,deptno,sal from emp;

     EMPNO ENAME      JOB           DEPTNO        SAL
---------- ---------- --------- ---------- ----------
      7369 SMITH      CLERK             20        800
      7499 ALLEN      SALESMAN          30       1600
      7521 WARD       SALESMAN          30       1250
      7566 JONES      MANAGER           20       2975
      7654 MARTIN     SALESMAN          30       1250
      7698 BLAKE      MANAGER           30       2850
      7782 CLARK      MANAGER           10       2450
      7788 SCOTT      ANALYST           20       3000
      7839 KING       PRESIDENT         10       5000
      7844 TURNER     SALESMAN          30       1500
      7876 ADAMS      CLERK             20       1100
      7900 JAMES      CLERK             30        950
      7902 FORD       ANALYST           20       3000
      7934 MILLER     CLERK             10       1300

14 rows selected.
```

3. 算术运算

注意：数字可以进行加减乘除运算，日期可以进行加减运算，字符串不能算术运算；

3.1 查询 emp 表中 KING 员工的工资信息；

```
SQL> show user
USER is "SCOTT"
SQL> select * from emp where ename = 'KING';

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
```

3.2 将 KING 的工资进行加、减、乘、除运算；

```
SQL> select ename,sal+100,sal-100,sal*2,sal/2 from emp
  2  where ename = 'KING';

ENAME      SAL+100    SAL-100      SAL*2      SAL/2
---------- ---------- ---------- ---------- ----------
KING             5100       4900      10000       2500

SQL>
```

3.3 查询数据库系统时间，常以服务器默认的格式进行显示(根据数据库的字符集而定)；

注意：dual 为数据库中的虚表，隶属于管理员 sys 用户，但所有的用户都可以访问；无实际意义，仅充

当 select 语句的结构(用 select 取系统信息、临时结果等时，以 dual 充当语句结构)；

```
SQL> show user
USER is "SCOTT"
SQL> select sysdate from dual;

SYSDATE
---------
18-MAY-11
```

3.4 修改系统时间的显示格式；(session 表明此修改限当前会话有效)

```
SQL> show user
USER is "SCOTT"
SQL> alter session set nls_date_format = 'yyyy-mm-dd hh24:mi:ss';

Session altered.

SQL> select sysdate from dual;

SYSDATE
-------------------
2011-05-19 20:06:29

SQL>
```

3.5 将系统时间进行加减运算；

3.5.1 加减一个数字，表示将所给的时间加减多少天；(例：加减 2 天)

```
SQL> select sysdate from dual;

SYSDATE
-------------------
2011-05-19 20:06:29

SQL> select sysdate + 2,sysdate -2 from dual;

SYSDATE+2           SYSDATE-2
------------------- -------------------
2011-05-21 20:08:07 2011-05-17 20:08:07

SQL>
```

3.5.2 如果想追加多少小时，多少分，多少秒，那该怎么办？(例：加减 2 小时，2 分钟，2 秒钟)

```
SQL> select sysdate from dual;

SYSDATE
-------------------
2011-05-19 20:08:48

SQL> select sysdate + 2/24,sysdate + 2/24/60,sysdate + 2/24/60/60 from dual;

SYSDATE+2/24        SYSDATE+2/24/60     SYSDATE+2/24/60/60
------------------- ------------------- -------------------
2011-05-19 22:09:12 2011-05-19 20:11:12 2011-05-19 20:09:14

SQL>
```

注意：时间换算时的进制问题；

注意：以上时间格式的修改，限当前会话有效；
得到的结果只是临时数据进行显示，并不会改变表中原有的数据值；

4.关于 null 的运算

注意：null 既不是 0，也不是空字符串，而是一个不确定的值

step 1：描述表中数据的信息(注意 comm 字段的值)

```
SQL> show user
USER is "SCOTT"
SQL> select * from emp;

     EMPNO ENAME      JOB              MGR HIREDATE                 SAL       COMM     DEPTNO
---------- ---------- --------- ---------- ------------------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 1980-12-17 00:00:00        800                    20
      7499 ALLEN      SALESMAN        7698 1981-02-20 00:00:00       1600        300         30
      7521 WARD       SALESMAN        7698 1981-02-22 00:00:00       1250        500         30
      7566 JONES      MANAGER         7839 1981-04-02 00:00:00       2975                    20
      7654 MARTIN     SALESMAN        7698 1981-09-28 00:00:00       1250       1400         30
      7698 BLAKE      MANAGER         7839 1981-05-01 00:00:00       2850                    30
      7782 CLARK      MANAGER         7839 1981-06-09 00:00:00       2450                    10
      7788 SCOTT      ANALYST         7566 1987-04-19 00:00:00       3000                    20
      7839 KING       PRESIDENT            1981-11-17 00:00:00       5000                    10
      7844 TURNER     SALESMAN        7698 1981-09-08 00:00:00       1500          0         30
      7876 ADAMS      CLERK           7788 1987-05-23 00:00:00       1100                    20
      7900 JAMES      CLERK           7698 1981-12-03 00:00:00        950                    30
      7902 FORD       ANALYST         7566 1981-12-03 00:00:00       3000                    20
      7934 MILLER     CLERK           7782 1982-01-23 00:00:00       1300                    10

14 rows selected.

SQL>
```

step 2：检索 comm 为 null 或者非 null 的记录

注意：where 条件中不能使用=或者<>(不等号的表示!=或者<>或者^=，工作中多使用<>)

```
SQL> select * from emp where comm is null;

     EMPNO ENAME      JOB              MGR HIREDATE                 SAL       COMM     DEPTNO
---------- ---------- --------- ---------- ------------------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 1980-12-17 00:00:00        800                    20
      7566 JONES      MANAGER         7839 1981-04-02 00:00:00       2975                    20
      7698 BLAKE      MANAGER         7839 1981-05-01 00:00:00       2850                    30
      7782 CLARK      MANAGER         7839 1981-06-09 00:00:00       2450                    10
      7788 SCOTT      ANALYST         7566 1987-04-19 00:00:00       3000                    20
      7839 KING       PRESIDENT            1981-11-17 00:00:00       5000                    10
      7876 ADAMS      CLERK           7788 1987-05-23 00:00:00       1100                    20
      7900 JAMES      CLERK           7698 1981-12-03 00:00:00        950                    30
      7902 FORD       ANALYST         7566 1981-12-03 00:00:00       3000                    20
      7934 MILLER     CLERK           7782 1982-01-23 00:00:00       1300                    10

10 rows selected.

SQL> select * from emp where comm is not null;

     EMPNO ENAME      JOB              MGR HIREDATE                 SAL       COMM     DEPTNO
---------- ---------- --------- ---------- ------------------- ---------- ---------- ----------
      7499 ALLEN      SALESMAN        7698 1981-02-20 00:00:00       1600        300         30
      7521 WARD       SALESMAN        7698 1981-02-22 00:00:00       1250        500         30
      7654 MARTIN     SALESMAN        7698 1981-09-28 00:00:00       1250       1400         30
      7844 TURNER     SALESMAN        7698 1981-09-08 00:00:00       1500          0         30

SQL>
```

step 3：null 值的计算

注意：任何值与 null 四则运算之后的值仍未 null；

```
SQL> select empno,ename,comm,comm+100 jia,comm-100 jian,comm*2 cheng,comm/2 chu
  2  from emp;

     EMPNO ENAME            COMM        JIA       JIAN      CHENG        CHU
---------- ---------- ---------- ---------- ---------- ---------- ----------
      7369 SMITH
      7499 ALLEN             300        400        200        600        150
      7521 WARD              500        600        400       1000        250
      7566 JONES
      7654 MARTIN           1400       1500       1300       2800        700
      7698 BLAKE
      7782 CLARK
      7788 SCOTT
      7839 KING
      7844 TURNER              0        100       -100          0          0
      7876 ADAMS
      7900 JAMES
      7902 FORD
      7934 MILLER

14 rows selected.
```

5. 列的别名

注意：列的别名中如果包含空格、特殊字符、关键词、或者大小写敏感的信息，需加双引号；

```
SQL> show user
USER is "SCOTT"
SQL> select empno "bian hao",ename "@#$%",sal "select",deptno "BuMen"
  2  from emp;

  bian hao @#$%           select      BuMen
---------- ---------- ---------- ----------
      7369 SMITH             800         20
      7499 ALLEN            1600         30
      7521 WARD             1250         30
      7566 JONES            2975         20
      7654 MARTIN           1250         30
      7698 BLAKE            2850         30
      7782 CLARK            2450         10
      7788 SCOTT            3000         20
      7839 KING             5000         10
      7844 TURNER           1500         30
      7876 ADAMS            1100         20
      7900 JAMES             950         30
      7902 FORD             3000         20
      7934 MILLER           1300         10

14 rows selected.
```

6.连字运算符和单引号的使用

注意：字符串信息需加单引号；

6.1 显示 【名字】 is a name of 【编号】

```
SQL> show user
USER is "SCOTT"
SQL> select ename||' is a name of '||empno info
  2  from emp;

INFO
----------------------------------------------------------------
SMITH is a name of 7369
ALLEN is a name of 7499
WARD is a name of 7521
JONES is a name of 7566
MARTIN is a name of 7654
BLAKE is a name of 7698
CLARK is a name of 7782
SCOTT is a name of 7788
KING is a name of 7839
TURNER is a name of 7844
ADAMS is a name of 7876
JAMES is a name of 7900
FORD is a name of 7902
MILLER is a name of 7934

14 rows selected.
```

6.2 显示 【编号】 ‘s name is 【名字】

```
SQL> show user
USER is "SCOTT"
SQL> select empno||'''s name is '||ename info
  2  from emp;

INFO
----------------------------------------------------------------
7369's name is SMITH
7499's name is ALLEN
7521's name is WARD
7566's name is JONES
7654's name is MARTIN
7698's name is BLAKE
7782's name is CLARK
7788's name is SCOTT
7839's name is KING
7844's name is TURNER
7876's name is ADAMS
7900's name is JAMES
7902's name is FORD
7934's name is MILLER

14 rows selected.
```

6.3 显示　　【编号】 is a number, and it’s name is 【名字】

注意：两个单引号‘‘表示转义，打印结果时显示一个单引号；

```
SQL> show user
USER is "SCOTT"
SQL> select empno||' is a number,and it''s name is '||ename info
  2  from emp;

INFO
--------------------------------------------------------------------------------
7369 is a number,and it's name is SMITH
7499 is a number,and it's name is ALLEN
7521 is a number,and it's name is WARD
7566 is a number,and it's name is JONES
7654 is a number,and it's name is MARTIN
7698 is a number,and it's name is BLAKE
7782 is a number,and it's name is CLARK
7788 is a number,and it's name is SCOTT
7839 is a number,and it's name is KING
7844 is a number,and it's name is TURNER
7876 is a number,and it's name is ADAMS
7900 is a number,and it's name is JAMES
7902 is a number,and it's name is FORD
7934 is a number,and it's name is MILLER

14 rows selected.
```

例：连接时间、字符串、数字

```
SQL> select 'Today is '||sysdate||',and Roger''s phone number is '||13802142274 userinfo
  2  from dual;

USERINFO
--------------------------------------------------------------------
Today is 2011-05-19 20:29:17,and Roger's phone number is 13802142274
```

7.显示不重复的信息

7.1 如果想知道 emp 表中总共有多少个部门，那怎么看？

例：若直接查看，则会存在重复信息；使用 distinct 消除重复行；

注意：distinct 写在字段最前面，它会影响之后的所有字段信息；

```
SQL> show user
USER is "SCOTT"
SQL> select deptno from emp;

    DEPTNO
----------
        20
        30
        30
        20
        30
        30
        10
        20
        10
        30
        20
        30
        20
        10

14 rows selected.

SQL> select distinct deptno from emp;

    DEPTNO
----------
        30
        20
        10
```

例：如果后面跟多个字段信息，将会把所有字段作为组合信息，与其他记录进行比较；

注意：temp info 对应的 1234 信息，只是作为临时数据进行显示，而 emp 表中并不包含此信息；

```
SQL> show user
USER is "SCOTT"
SQL> select job,1234 "temp info" from emp;

JOB       temp info
--------- ----------
CLERK           1234
SALESMAN        1234
SALESMAN        1234
MANAGER         1234
SALESMAN        1234
MANAGER         1234
MANAGER         1234
ANALYST         1234
PRESIDENT       1234
SALESMAN        1234
CLERK           1234
CLERK           1234
ANALYST         1234
CLERK           1234

14 rows selected.
```

将 job 与 temp info 作为组合信息，记录之间相互比较，留下不重复记录；

```
SQL> select distinct job,1234 "temp info" from emp;

JOB       temp info
--------- ----------
SALESMAN        1234
CLERK           1234
MANAGER         1234
ANALYST         1234
PRESIDENT       1234
```

【第二章 约束和排序数据】

1. 在 emp 表中选择工资介于 1500 到 2500 的员工的信息；

注意：使用 between 下边界 and 上边界时，条件包括边界值；

```
SQL> show user
USER is "SCOTT"
SQL> select * from emp
  2  where sal between 1500 and 2500;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7499 ALLEN      SALESMAN         7698 20-FEB-81       1600        300         30
      7782 CLARK      MANAGER          7839 09-JUN-81       2450                    10
      7844 TURNER     SALESMAN         7698 08-SEP-81       1500          0         30
```

2. 在 emp 表中选择位于 10,20 部门的员工的信息；

```
SQL> show user
USER is "SCOTT"
SQL> select * from emp
  2  where deptno in (10,20);

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK            7902 17-DEC-80        800                    20
      7566 JONES      MANAGER          7839 02-APR-81       2975                    20
      7782 CLARK      MANAGER          7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST          7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT             17-NOV-81       5000                    10
      7876 ADAMS      CLERK            7788 23-MAY-87       1100                    20
      7902 FORD       ANALYST          7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK            7782 23-JAN-82       1300                    10

8 rows selected.
```

3. 在 emp 表中选择位于员工名字中包含大写字符‘O’的员工的信息；

```
SQL> show user
USER is "SCOTT"
SQL> select * from emp
  2  where ename like '%O%';

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
```

注意：如果查询的名字中包含%或者_，而且查询的时候又要查询这样的信息，需要用到换位码；

4. 通配符%和_的使用，以及换位码的使用；

注意：通配符%，表示 0 或者多个字符一样；通配符_，表示 1 个字符一样；

4.1 参照 emp 表创建一张 t01 表；

```
SQL> show user
USER is "SCOTT"
SQL> create table t01
  2  as select * from emp where 1=2;

Table created.
```

注意：通过上述方式创建的表 t01，和 emp 表的结构一样，但是其中没有数据；

```
SQL> select * from t01;

no rows selected
```

4.2 添加包含通配符的测试用数据；

```
SQL> insert into t01(empno,ename,sal) values(1001,'haha%hehe',1000);

1 row created.

SQL> insert into t01(empno,ename,sal) values(1002,'%xixi',2000);

1 row created.

SQL> insert into t01(empno,ename,sal) values(1003,'haha_hehe',3000);

1 row created.

SQL> insert into t01(empno,ename,sal) values(1004,'_xixi',4000);

1 row created.

SQL> commit;

Commit complete.

SQL> select * from t01;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      1001 haha%hehe                                      1000
      1002 %xixi                                          2000
      1003 haha_hehe                                      3000
      1004 _xixi                                          4000
```

4.3 换位码的使用方法；（此处以\作为换位码，其实换位码也可指定其他字符）

例：检索包含%的记录信息；

```
SQL> show user
USER is "SCOTT"
SQL> select * from t01
  2  where ename like '%\%%' escape '\';

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      1001 haha%hehe                                      1000
      1002 %xixi                                          2000
```

例：检索以%开头的记录信息；

```
SQL> select * from t01
  2  where ename like '\%%' escape '\';

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      1002 %xixi                                          2000
```

例：检索包含_的记录信息；

```
SQL> select * from t01
  2  where ename like '%\_%' escape '\';

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      1003 haha_hehe                                      3000
      1004 _xixi                                          4000
```

例：检索以_开头的记录信息；

```
SQL> select * from t01
  2  where ename like '\_%' escape '\';

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      1004 _xixi                                          4000
```

5. 复合条件的使用(参照第二章 1,2 练习)

5.1 对于 and 条件复合(可以将 between...and...进行转换)

例：在 emp 表中选择工资介于 1500 到 2500 的员工的信息；

```
SQL> show user
USER is "SCOTT"
SQL> select * from emp
  2  where sal >= 1500 and sal <= 2500;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
```

5.2 对于 or 条件复合(可以将 in()进行转换)

例：在 emp 表中选择位于 10,20 部门的员工的信息；

```
SQL> select * from emp
  2  where deptno = 10 or deptno = 20;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10

8 rows selected.
```

6. 对于表中数据的排序

6.1 asc，表示按照所给字段进行升序排列(不指明时默认按所给字段升序排列)；

desc，表示按照所给字段进行降序排列；

例：emp 表中 10 部门员工的信息按照 sal 进行升序排序；

```
SQL> show user
USER is "SCOTT"
SQL> select * from emp
  2  where deptno = 10
  3  order by sal asc;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
```

例：emp 表中 20 部门员工的信息按照 sal 进行降序排序；

```
SQL> select * from emp
  2  where deptno = 20
  3  order by sal desc;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
```

6.2 如果 **order by** 后面跟多个字段，则将结果集先按照第 **1** 个字段进行排序，【条件 **1**】，再按照第 **2** 个字段进行排序；

注意：【条件 **1**】如果按照第 **1** 个字段分不开先后顺序的时候，才会按照第 **2** 个字段进行排序；

注意：**asc** 或者 **desc** 影响的字段，仅仅是它紧挨着的那个字段升降顺序；

例：**emp** 表中 **10** 部门员工的信息按照 **empno** 升序、**sal** 降序进行升序排序；

```
SQL> select * from emp
  2  where deptno = 10
  3  order by empno asc,sal desc;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10
```

例：**emp** 表中 **20** 部门员工的信息按照 **empno** 降序、**sal** 升序进行升序排序；

```
SQL> select * from emp
  2  where deptno = 20
  3  order by empno desc,sal asc;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
```

6.3 当然在排序的时候也可以使用字段在表中定义的先后位置进行排序；

注意：首先确认字段在表中定义的先后顺序；

```
SQL> show user
USER is "SCOTT"
SQL> desc emp
 Name                                                                    Null?    Type
 ----------------------------------------------------------------------- -------- ------------------------------------
 EMPNO                                                                   NOT NULL NUMBER(4)
 ENAME                                                                            VARCHAR2(10)
 JOB                                                                              VARCHAR2(9)
 MGR                                                                              NUMBER(4)
 HIREDATE                                                                         DATE
 SAL                                                                              NUMBER(7,2)
 COMM                                                                             NUMBER(7,2)
 DEPTNO                                                                           NUMBER(2)
```

例：**emp** 表中 **10** 部门员工的信息按照 **empno** 升序、**sal** 降序进行升序排序；

```
SQL> select * from emp
  2  where deptno = 10
  3  order by 1 asc,6 desc;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10
```

例：**emp** 表中 **20** 部门员工的信息按照 **empno** 降序、**sal** 升序进行升序排序；

```
SQL> select * from emp
  2  where deptno = 20
  3  order by 1 desc,sal asc;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
```

6.4 当然除了可以使用 number 类型的字段进行排序外，还可以使用字符串或者时间类型的字段进行排序；

注意：字符串排序：按照字符对应的 ASCII 码的先后进行排序；

日期排序：按照日期的先后进行排序，时间越往后越大；

例：emp 表中员工的信息按照 job 升序、ename 降序进行排列；

注意：结果集中如果 job 相同的情况下，会按照 ename 进行降序排列；

```
SQL> show user
USER is "SCOTT"
SQL> select * from emp
  2  order by job asc,ename desc;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30

14 rows selected.
```

例：emp 表中员工的信息按照 hiredate 升序，sal 降序进行排列；

注意：将日期类型的 hiredate 的显示格式进行设定；

注意：结果集中如果 hiredate 相同的情况下，会按照 sal 进行降序排列；

```
SQL> show user
USER is "SCOTT"
SQL> alter session set nls_date_format = 'yyyy-mm-dd hh24:mi:ss';

Session altered.

SQL> select * from emp
  2  order by hiredate asc,sal desc;

     EMPNO ENAME      JOB              MGR HIREDATE                 SAL       COMM     DEPTNO
---------- ---------- --------- ---------- ------------------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 1980-12-17 00:00:00        800                    20
      7499 ALLEN      SALESMAN        7698 1981-02-20 00:00:00       1600        300         30
      7521 WARD       SALESMAN        7698 1981-02-22 00:00:00       1250        500         30
      7566 JONES      MANAGER         7839 1981-04-02 00:00:00       2975                    20
      7698 BLAKE      MANAGER         7839 1981-05-01 00:00:00       2850                    30
      7782 CLARK      MANAGER         7839 1981-06-09 00:00:00       2450                    10
      7844 TURNER     SALESMAN        7698 1981-09-08 00:00:00       1500          0         30
      7654 MARTIN     SALESMAN        7698 1981-09-28 00:00:00       1250       1400         30
      7839 KING       PRESIDENT            1981-11-17 00:00:00       5000                    10
      7902 FORD       ANALYST         7566 1981-12-03 00:00:00       3000                    20
      7900 JAMES      CLERK           7698 1981-12-03 00:00:00        950                    30
      7934 MILLER     CLERK           7782 1982-01-23 00:00:00       1300                    10
      7788 SCOTT      ANALYST         7566 1987-04-19 00:00:00       3000                    20
      7876 ADAMS      CLERK           7788 1987-05-23 00:00:00       1100                    20

14 rows selected.
```

7. 使用结果集中列的别名进行排序；

例：按照 emp 表中员工对应的年薪(sal*12)进行排序；

```
SQL> select empno,ename,sal*12 year_sal
  2  from emp
  3  order by year_sal desc;

     EMPNO ENAME        YEAR_SAL
---------- ---------- ----------
      7839 KING            60000
      7902 FORD            36000
      7788 SCOTT           36000
      7566 JONES           35700
      7698 BLAKE           34200
      7782 CLARK           29400
      7499 ALLEN           19200
      7844 TURNER          18000
      7934 MILLER          15600
      7521 WARD            15000
      7654 MARTIN          15000
      7876 ADAMS           13200
      7900 JAMES           11400
      7369 SMITH            9600

14 rows selected.
```

注意：当然也可以按照字符串、日期对应的别名进行排序；

8.如果排序的字段中包含 null 值，那结果会怎样？

注意：在字段进行比较大小的时候，null 值比任何值都大；

例：emp 表中员工的信息按照 comm 升序进行排列；

```
SQL> show user
USER is "SCOTT"
SQL> select * from emp
  2  order by comm asc;

     EMPNO ENAME      JOB              MGR HIREDATE                 SAL       COMM     DEPTNO
---------- ---------- --------- ---------- ------------------- ---------- ---------- ----------
      7844 TURNER     SALESMAN        7698 1981-09-08 00:00:00       1500          0         30
      7499 ALLEN      SALESMAN        7698 1981-02-20 00:00:00       1600        300         30
      7521 WARD       SALESMAN        7698 1981-02-22 00:00:00       1250        500         30
      7654 MARTIN     SALESMAN        7698 1981-09-28 00:00:00       1250       1400         30
      7788 SCOTT      ANALYST         7566 1987-04-19 00:00:00       3000                    20
      7839 KING       PRESIDENT            1981-11-17 00:00:00       5000                    10
      7876 ADAMS      CLERK           7788 1987-05-23 00:00:00       1100                    20
      7900 JAMES      CLERK           7698 1981-12-03 00:00:00        950                    30
      7902 FORD       ANALYST         7566 1981-12-03 00:00:00       3000                    20
      7934 MILLER     CLERK           7782 1982-01-23 00:00:00       1300                    10
      7698 BLAKE      MANAGER         7839 1981-05-01 00:00:00       2850                    30
      7566 JONES      MANAGER         7839 1981-04-02 00:00:00       2975                    20
      7369 SMITH      CLERK           7902 1980-12-17 00:00:00        800                    20
      7782 CLARK      MANAGER         7839 1981-06-09 00:00:00       2450                    10

14 rows selected.
```

注意：结果集会按照 comm 升序排列(有值的进行排列，null 放在后面，但是 null 值之间怎么排列暂不考虑[因为 null 和 null 之间无法比较大小])；

如果，我想把黑色部分和白色部分位置上下条换，该怎么办？

```
SQL> select * from emp
  2  order by comm asc nulls first;

     EMPNO ENAME      JOB              MGR HIREDATE                 SAL       COMM     DEPTNO
---------- ---------- --------- ---------- ------------------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 1980-12-17 00:00:00        800                    20
      7782 CLARK      MANAGER         7839 1981-06-09 00:00:00       2450                    10
      7902 FORD       ANALYST         7566 1981-12-03 00:00:00       3000                    20
      7900 JAMES      CLERK           7698 1981-12-03 00:00:00        950                    30
      7876 ADAMS      CLERK           7788 1987-05-23 00:00:00       1100                    20
      7566 JONES      MANAGER         7839 1981-04-02 00:00:00       2975                    20
      7698 BLAKE      MANAGER         7839 1981-05-01 00:00:00       2850                    30
      7934 MILLER     CLERK           7782 1982-01-23 00:00:00       1300                    10
      7788 SCOTT      ANALYST         7566 1987-04-19 00:00:00       3000                    20
      7839 KING       PRESIDENT            1981-11-17 00:00:00       5000                    10
      7844 TURNER     SALESMAN        7698 1981-09-08 00:00:00       1500          0         30
      7499 ALLEN      SALESMAN        7698 1981-02-20 00:00:00       1600        300         30
      7521 WARD       SALESMAN        7698 1981-02-22 00:00:00       1250        500         30
      7654 MARTIN     SALESMAN        7698 1981-09-28 00:00:00       1250       1400         30

14 rows selected.
```

例：同上，emp 表中员工的信息按照 comm 降序进行排列；

```
SQL> select * from emp
  2   order by comm desc;

     EMPNO ENAME      JOB              MGR HIREDATE                 SAL       COMM     DEPTNO
---------- ---------- --------- ---------- ------------------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 1980-12-17 00:00:00        800                    20
      7782 CLARK      MANAGER         7839 1981-06-09 00:00:00       2450                    10
      7902 FORD       ANALYST         7566 1981-12-03 00:00:00       3000                    20
      7900 JAMES      CLERK           7698 1981-12-03 00:00:00        950                    30
      7876 ADAMS      CLERK           7788 1987-05-23 00:00:00       1100                    20
      7566 JONES      MANAGER         7839 1981-04-02 00:00:00       2975                    20
      7698 BLAKE      MANAGER         7839 1981-05-01 00:00:00       2850                    30
      7934 MILLER     CLERK           7782 1982-01-23 00:00:00       1300                    10
      7788 SCOTT      ANALYST         7566 1987-04-19 00:00:00       3000                    20
      7839 KING       PRESIDENT            1981-11-17 00:00:00       5000                    10
      7654 MARTIN     SALESMAN        7698 1981-09-28 00:00:00       1250       1400         30
      7521 WARD       SALESMAN        7698 1981-02-22 00:00:00       1250        500         30
      7499 ALLEN      SALESMAN        7698 1981-02-20 00:00:00       1600        300         30
      7844 TURNER     SALESMAN        7698 1981-09-08 00:00:00       1500          0         30

14 rows selected.
```

如果，我想把黑色部分和白色部分位置上下条换，该怎么办？

```
SQL> select * from emp
  2  order by comm desc nulls last;

     EMPNO ENAME      JOB              MGR HIREDATE                 SAL       COMM     DEPTNO
---------- ---------- --------- ---------- ------------------- ---------- ---------- ----------
      7654 MARTIN     SALESMAN        7698 1981-09-28 00:00:00       1250       1400         30
      7521 WARD       SALESMAN        7698 1981-02-22 00:00:00       1250        500         30
      7499 ALLEN      SALESMAN        7698 1981-02-20 00:00:00       1600        300         30
      7844 TURNER     SALESMAN        7698 1981-09-08 00:00:00       1500          0         30
      7788 SCOTT      ANALYST         7566 1987-04-19 00:00:00       3000                    20
      7839 KING       PRESIDENT            1981-11-17 00:00:00       5000                    10
      7876 ADAMS      CLERK           7788 1987-05-23 00:00:00       1100                    20
      7900 JAMES      CLERK           7698 1981-12-03 00:00:00        950                    30
      7902 FORD       ANALYST         7566 1981-12-03 00:00:00       3000                    20
      7934 MILLER     CLERK           7782 1982-01-23 00:00:00       1300                    10
      7698 BLAKE      MANAGER         7839 1981-05-01 00:00:00       2850                    30
      7566 JONES      MANAGER         7839 1981-04-02 00:00:00       2975                    20
      7369 SMITH      CLERK           7902 1980-12-17 00:00:00        800                    20
      7782 CLARK      MANAGER         7839 1981-06-09 00:00:00       2450                    10

14 rows selected.
```

【第三章 单行函数】

1. 字符处理函数(主要针对字符串的格式进行设定)

1.1 字符串大小写处理函数

注意：lower(s)，将所给的字符串 s 全部转换成小写；

upper(s)，将所给的字符串 s 全部转换成大写；

initcap(s)，将所给的字符串 s 中的每个单词转换成首字母大写；

```
SQL> show user
USER is "SCOTT"
SQL> select lower('HELLO Roger,WELCOME TO HEBUT!') result from dual;

RESULT
------------------------------
hello roger,welcome to hebut!

SQL> select upper('hello Roger,Welcome to HEBUT!') result from dual;

RESULT
------------------------------
HELLO ROGER,WELCOME TO HEBUT!

SQL> select initcap('HELLO Roger,welcome to HeBut!') result from dual;

RESULT
------------------------------
Hello Roger,Welcome To Hebut!
```

例：如何忽略大小写敏感性

step 1：创建一张表 t02，字段定义如下；

```
SQL> show user
USER is "SCOTT"
SQL> create table t02(no number(4),info varchar2(10),dt date);

Table created.

SQL> desc t02
 Name                                                                    Null?    Type
 ----------------------------------------------------------------------- -------- ----------------------------
 NO                                                                               NUMBER(4)
 INFO                                                                             VARCHAR2(10)
 DT                                                                               DATE
```

step 2：插入测试数据，并提交；

注意：测试数据要全面考虑，既有测试正常分支，又有防止异常干扰的；

```
SQL> insert into t02 values(1,'oracle',sysdate);

1 row created.

SQL> insert into t02 values(2,'ORACLE',sysdate);

1 row created.

SQL> insert into t02 values(3,'OrAcle',sysdate);

1 row created.

SQL> insert into t02 values(4,'elcaro',sysdate);

1 row created.

SQL> commit;

Commit complete.

SQL> select * from t02;

        NO INFO       DT
---------- ---------- -------------------
         1 oracle     2011-05-19 23:17:29
         2 ORACLE     2011-05-19 23:17:40
         3 OrAcle     2011-05-19 23:17:57
         4 elcaro     2011-05-19 23:18:33
```

step 3：查询关于 oracle 的信息(无论输入信息是 ORACLE，还是 oracle，还是 ORaclE，都可以查到所有关于 oracle 的信息，但是前提是这个单词 oracle 没有拼写错误，哈哈！)；

```
SQL> select * from t02
  2  where lower(info) = lower('oRaclE');

        NO INFO       DT
---------- ---------- -------------------
         1 oracle     2011-05-19 23:17:29
         2 ORACLE     2011-05-19 23:17:40
         3 OrAcle     2011-05-19 23:17:57
```

注意：检索的时候只要使用其中任意一个大小写处理函数，就可以把字符串的敏感性忽略掉；但是由此也会造成其他的影响，后续课程在探讨；

1.2 concat(s1，s2)：将字符串 s1 和字符串 s2 拼接起来；

注意：此函数只有 2 个参数，等同于将 s1 和 s2 用||拼接起来；

```
SQL> show user
USER is "SCOTT"
SQL> select concat(empno,ename) result
  2  from emp
  3  where rownum <= 3;

RESULT
--------------------------------------------------
7369SMITH
7499ALLEN
7521WARD
```

注意：rownum <= 3，表示只取得表中前 3 条记录即可；(但，运算符只能写<=或者<某个不小于 1 的整数)

注意：rownum 是表中数据的行号，是数据库自动添加的，但是在 select * from …查询时看不到，需要

显示的进行指定才可；同样还有 rowid，也是数据库自动添加的，64 进制的数；(此了解即可，一般编程用 rownum 时较多，做数据文件维护时两者会结合使用)

例：查询 dept 表中的所有记录，以及 rownum 和 rowid 的信息；

```
SQL> show user
USER is "SCOTT"
SQL> desc dept
 Name                                                                    Null?    Type
 ----------------------------------------------------------------------- -------- ------------------------------
 DEPTNO                                                                  NOT NULL NUMBER(2)
 DNAME                                                                            VARCHAR2(14)
 LOC                                                                              VARCHAR2(13)

SQL> select rownum,deptno,dname,loc,rowid
  2  from dept;

    ROWNUM     DEPTNO DNAME          LOC           ROWID
---------- ---------- -------------- ------------- ------------------
         1         10 ACCOUNTING     NEW YORK      AAAM0iAAEAAAAANAAA
         2         20 RESEARCH       DALLAS        AAAM0iAAEAAAAANAAB
         3         30 SALES          CHICAGO       AAAM0iAAEAAAAANAAC
         4         40 OPERATIONS     BOSTON        AAAM0iAAEAAAAANAAD
```

1.3 substr(s，m，[n])：表示从所给的字符串 s 中取得指定范围的子串；

注意：m 表示从字符串 s 的第几位开始，正整数表示从左到右数第几位开始，负整数表示从右至左数第几位开始；但是截取的方向都是向右；

n 表示截取子串的长度，如果不写，即从第 m 位开始向右直到结束；

例：字符串‘HowAreYou’

```
SQL> select substr('HowAreYou',4,3) result from dual;

RES
---
Are

SQL> select substr('HowAreYou',-6,3) result from dual;

RES
---
Are

SQL> select substr('HowAreYou',4) result from dual;

RESULT
------
AreYou

SQL> select substr('HowAreYou',-6) result from dual;

RESULT
------
AreYou
```

注意：如果字符串中含有空格，则空格算 1 个字符(前提是半角空格)

1.4 length(s)：表示返回所给字符串 s 的字符数，即长度(如果有半角空格，则算 1 个字符；当然全角空格算 2 个字符宽度)

例：字符串‘Hello Roger!’

```
SQL> select length('Hello Roger!') result from dual;

    RESULT
----------
        12
```

注意：区别一下 varchar2 和 char 类型的字符串

step 1：创建表 t03，字段定义如下；

```
SQL> show user
USER is "SCOTT"
SQL> create table t03(s1 varchar2(20), s2 char(20));

Table created.
```

step 2：插入测试数据，并提交；

```
SQL> insert into t03 values('abcdef','abcdef');

1 row created.

SQL> insert into t03 values('hello','roger');

1 row created.

SQL> commit;

Commit complete.
```

step 3：查询表中的数据；

```
SQL> select * from t03;

S1                   S2
-------------------- --------------------
abcdef               abcdef
hello                roger
```

step 4：区别一下；

```
SQL> desc t03
 Name                                                                    Null?    Type
 ----------------------------------------------------------------------- -------- ----------------------------
 S1                                                                               VARCHAR2(20)
 S2                                                                               CHAR(20)

SQL> select length(s1),length(s2) from t03;

LENGTH(S1) LENGTH(S2)
---------- ----------
         6         20
         5         20
```

原因：1)可参照第 2 页中对于 varchar2 和 char 的介绍；

2)由于 varchar2 是可变长字符类型，所以即使字段定义了 20 位，如果存放的数据达不到 20 位时，按实际长度进行存储；

但是，char 是定长字符类型，如果字段定义成 20 位，即使存放的数据达不到 20 位时，也会在原有的字符串右边补上半角空格，以总长 20 位的长度进行存储；

结论：1)在存储上，一般 varchar2 类型的数据所占空间比 char 类型的少；

2)在查询效率上，char 类型的比 varchar2 类型的快；（了解）

1.5 instr(s,s1,[m],[n])：表示子字符串 s1 在字符串 s 中，从第 m 位开始检索，第 n 次出现的时候，所在的位置是哪；

注意：如果 m 是负整数，表示从右第 m 开始向左检索；

如果 m，n 都省略，表示从左边第 1 位开始向右检索，在 s 中第 1 次出现 s1 的位置在哪；

例：字符串‘13802142274’，子字符串‘2’

```
SQL> select instr('13802142274','2') from dual;

INSTR('13802142274','2')
------------------------
                       5

SQL> select instr('13802142274','2',1,1) from dual;

INSTR('13802142274','2',1,1)
----------------------------
                           5


SQL> select instr('13802142274','2',4,2) from dual;

INSTR('13802142274','2',4,2)
----------------------------
                           8

SQL> select instr('13802142274','2',9,1) from dual;

INSTR('13802142274','2',9,1)
----------------------------
                           9
```

```
SQL> select instr('13802142274','2',-4,1) from dual;

INSTR('13802142274','2',-4,1)
-----------------------------
                            8

SQL> select instr('13802142274','2',-5,1) from dual;

INSTR('13802142274','2',-5,1)
-----------------------------
                            5

SQL> select instr('13802142274','2',-5,2) from dual;

INSTR('13802142274','2',-5,2)
-----------------------------
                            0
SQL> select instr('13802142274','2',1,4) from dual;

INSTR('13802142274','2',1,4)
----------------------------
                           0
```

注意：如果在 s 中找不到子串 s1，则显示结果为 0；

1.6 lpad(s，n，s1)：在字符串 s 左边补充子串 s1，直到长度为 n；

rpad(s，n，s1)：在字符串 s 右边补充子串 s1，直到长度为 n；

例：将 emp 表中的 ename 向左补充*到 10 位；

```
SQL> show user
USER is "SCOTT"
SQL> select ename,lpad(ename,10,'*') new_name from emp;

ENAME      NEW_NAME
---------- ----------
SMITH      *****SMITH
ALLEN      *****ALLEN
WARD       ******WARD
JONES      *****JONES
MARTIN     ****MARTIN
BLAKE      *****BLAKE
CLARK      *****CLARK
SCOTT      *****SCOTT
KING       ******KING
TURNER     ****TURNER
ADAMS      *****ADAMS
JAMES      *****JAMES
FORD       ******FORD
MILLER     ****MILLER

14 rows selected.
```

例：将 emp 表中的 ename 向左补充#到 15 位；

```
SQL> select ename,rpad(ename,15,'#') new_name from emp;

ENAME      NEW_NAME
---------- ---------------
SMITH      SMITH##########
ALLEN      ALLEN##########
WARD       WARD###########
JONES      JONES##########
MARTIN     MARTIN#########
BLAKE      BLAKE##########
CLARK      CLARK##########
SCOTT      SCOTT##########
KING       KING###########
TURNER     TURNER#########
ADAMS      ADAMS##########
JAMES      JAMES##########
FORD       FORD###########
MILLER     MILLER#########

14 rows selected.
```

1.7 trim(leading|trailing|both s1 from s)：表示将子字符串 s1 从 s 中去除；

注意：leading 表示将 s 左边的 s1 去除；trailing 表示将 s 右边的 s1 去除；both 表示将 s 两边的 s1 去除；（默认）

如果 s 中间包含 s1，则不作处理；

例 1：默认状况下，是将字符串两边的半角空格去除；

```
SQL> select trim('    abc    def   ') result from dual;

RESULT
----------
abc    def

SQL> select trim(both ' ' from '    abc    def   ') result from dual;

RESULT
----------
abc    def
```

如果只去除左边的半角空格？

注意：ltrim 在去除字符串中的半角空格时可以使用；

```
SQL> select trim(leading ' ' from '    abc    def   ') result from dual;

RESULT
-------------
abc    def

SQL> select ltrim('    abc    def   ') result from dual;

RESULT
-------------
abc    def
```

如果只去除右边的半角空格？

注意：rtrim 在去除字符串中的半角空格时可以使用

```
SQL> select trim(trailing ' ' from '    abc    def   ') result from dual;

RESULT
--------------
    abc    def

SQL> select rtrim('    abc    def   ') result from dual;

RESULT
--------------
    abc    def
```

例 2：指定去除特定的字符 s1；

step 1：创建表 t04，表定义如下；

```
SQL> show user
USER is "SCOTT"
SQL> create table t04(no number(2), info varchar2(20));

Table created.
```

step 2：添加测试数据，并提交；

```
SQL> insert into t04 values(10,'***abc***def***');

1 row created.

SQL> commit;

Commit complete.
```

step 3：查询表中的数据；

```
SQL> select * from t04;

        NO INFO
---------- --------------------
        10 ***abc***def***
```

step 4：如果去除两边的*

```
SQL> select trim('*' from info) result from t04;

RESULT
--------------------
abc***def

SQL> select trim(both '*' from info) result from t04;

RESULT
--------------------
abc***def
```

step 5：如果只去除左边的*

```
SQL> select trim(leading '*' from info) result from t04;

RESULT
--------------------
abc***def***
```

step 6：如果只去除右边的*

```
SQL> select trim(trailing '*' from info) result from t04;

RESULT
--------------------
***abc***def
```

注意：处理字符时，ltrim 和 rtrim 不适用；

1.8 replace(s,s1,s2)：表示将字符串 s 中的 s1 用 s2 替换掉；

例：1.7 中得到的表 t04 中的数据

```
SQL> show user
USER is "SCOTT"
SQL> select * from t04;

        NO INFO
---------- --------------------
        10 ***abc***def***
```

将*替换成# （默认将所有的*都用#替换）

```
SQL> select no,replace(info,'*','#') result from t04;

        NO RESULT
---------- --------------------
        10 ###abc###def###
```

或者部分*替换成#

```
SQL> select no,replace(info,'**','##') result from t04;

        NO RESULT
---------- ----------------------------------------
        10 ##*abc##*def##*
```

例：将‘13802142274’中的‘0214’替换成‘****’

```
SQL> select replace('13802142274','0214','****') result from dual;

RESULT
-----------
138****2274
```

注意：如此替换时，s1 和 s2 的字符长度须一致；

拓展：如果此电话号码存储在某张表中，该如何处理？

预想效果：138****2274

step 1：创建表 t05，字段定义如下；

```
SQL> show user
USER is "SCOTT"
SQL> create table t05(no number(2),tel varchar2(15));

Table created.
```

step 2：添加测试数据，并提交；

```
SQL> insert into t05 values(10,'13802142274');

1 row created.

SQL> commit;

Commit complete.
```

step 3：查询数据；

```
SQL> select * from t05;

        NO TEL
---------- ---------------
        10 13802142274
```

step 4：格式编辑；

```
SQL> select replace(tel,substr(tel,4,4),'****') result from t05;

RESULT
-------------------------------------------------------------
138****2274
```

2. 数字处理函数(主要针对数字的格式进行设定)

2.1 round(c，[n])：表示求数字 c 的四舍五入值；

注意：n 表示保留到小数点后面多少位；如果 n 省略，则保留成整数；如果 n 为负数，则保留到小数点左边多少位；

```
SQL> show user
USER is "SCOTT"
SQL> select round(123.457) d1,round(123.457,0) d2,
  2         round(123.457,2) d3,round(123.457,-2) d4,
  3         round(123.457,-3) d5
  4  from dual;

        D1         D2         D3         D4         D5
---------- ---------- ---------- ---------- ----------
       123        123     123.46        100          0
```

提示：如果 c=123.457，n=2，则需要看小数点右面第 3 位(n+1)，若>=5,则进 1，反之舍去；
如果 n=-2，则需要看小数点左边第 2 位(n)，若>=5,则进 1，反之舍去；

2.2 trunc(c，[n])：表示将数字 c 截断，保留到小数点后第 n 位；

注意：n 表示保保留到小数点后第 n 位，则第 n 位之后的数字，不考虑四舍五入，直接舍去；

如果 n 省略，则只保留成整数，小数部分直接舍去；如果 n 为负数，则保留到小数点左边第 n 位，之后的数字，不考虑四舍五入，直接舍去；

```
SQL> select trunc(123.457) d1,trunc(123.457,0) d2,
  2         trunc(123.457,2) d3,trunc(123.457,-2) d4,
  3         trunc(123.457,-3) d5
  4  from dual;

        D1         D2         D3         D4         D5
---------- ---------- ---------- ---------- ----------
       123        123     123.45        100          0
```

2.3 mod(m，n)：表示取得 m 除以 n 得到的余数；

注意：n=0 时，mod(m，n)=m；（特殊）

```
SQL> select mod(10,3) d1,mod(10,0) d2,mod(0,10) d3
  2  from dual;

        D1         D2         D3
---------- ---------- ----------
         1         10          0
```

2.4 ceil(c)：表示将数字 c 向上取整；

注意：ceil(c)取得的结果应该为大于或者等于 c 的最小整数；

```
SQL> select ceil(123.457) d1,ceil(123.789) d2,
  2         ceil(123.00000000) d3,ceil(123.00000001) d4
  3  from dual;

        D1         D2         D3         D4
---------- ---------- ---------- ----------
       124        124        123        124
```

floor(c)：表示将数字 c 向下取整；

注意：floor(c)取得的结果应该为小于或者等于 c 的最大整数；

```
SQL> select floor(123.457) d1,floor(123.789) d2,
  2         floor(123.00000000) d3,floor(123.00000001) d4
  3  from dual;

        D1         D2         D3         D4
---------- ---------- ---------- ----------
       123        123        123        123
```

3. 日期处理函数(主要针对日期的格式进行设定)

3.1 关于一个日期加减多少天、小时、分钟、秒，请参照第一章的 3.5 将系统时间进行加减运算；

3.2 两个日期想减，得到的是两个日期之间相差的天数；

注意：日期的比较中，越往后的时间越大；

例：查看 emp 表中，员工入职了多长时间？

step 1：修改日期的显示格式；

```
SQL> show user
USER is "SCOTT"
SQL> alter session set nls_date_format = 'yyyy-mm-dd hh24:mi:ss';

Session altered.
```

step 2：查看 emp 表中 hiredate 信息，以及服务器的系统时间 sysdate；

```
SQL> show user
USER is "SCOTT"
SQL> select empno,ename,hiredate,sysdate
  2  from emp;

     EMPNO ENAME      HIREDATE            SYSDATE
---------- ---------- ------------------- -------------------
      7369 SMITH      1980-12-17 00:00:00 2011-05-20 03:57:16
      7499 ALLEN      1981-02-20 00:00:00 2011-05-20 03:57:16
      7521 WARD       1981-02-22 00:00:00 2011-05-20 03:57:16
      7566 JONES      1981-04-02 00:00:00 2011-05-20 03:57:16
      7654 MARTIN     1981-09-28 00:00:00 2011-05-20 03:57:16
      7698 BLAKE      1981-05-01 00:00:00 2011-05-20 03:57:16
      7782 CLARK      1981-06-09 00:00:00 2011-05-20 03:57:16
      7788 SCOTT      1987-04-19 00:00:00 2011-05-20 03:57:16
      7839 KING       1981-11-17 00:00:00 2011-05-20 03:57:16
      7844 TURNER     1981-09-08 00:00:00 2011-05-20 03:57:16
      7876 ADAMS      1987-05-23 00:00:00 2011-05-20 03:57:16
      7900 JAMES      1981-12-03 00:00:00 2011-05-20 03:57:16
      7902 FORD       1981-12-03 00:00:00 2011-05-20 03:57:16
      7934 MILLER     1982-01-23 00:00:00 2011-05-20 03:57:16

14 rows selected.
```

step 3：取得员工的入职时间；

注意：结果得到的是关于天的一个相对精确的数值；

```
SQL> select empno,ename,hiredate,sysdate, sysdate - hiredate result
  2  from emp;

     EMPNO ENAME      HIREDATE            SYSDATE             RESULT
---------- ---------- ------------------- ------------------- ----------
      7369 SMITH      1980-12-17 00:00:00 2011-05-20 03:57:59 11111.1653
      7499 ALLEN      1981-02-20 00:00:00 2011-05-20 03:57:59 11046.1653
      7521 WARD       1981-02-22 00:00:00 2011-05-20 03:57:59 11044.1653
      7566 JONES      1981-04-02 00:00:00 2011-05-20 03:57:59 11005.1653
      7654 MARTIN     1981-09-28 00:00:00 2011-05-20 03:57:59 10826.1653
      7698 BLAKE      1981-05-01 00:00:00 2011-05-20 03:57:59 10976.1653
      7782 CLARK      1981-06-09 00:00:00 2011-05-20 03:57:59 10937.1653
      7788 SCOTT      1987-04-19 00:00:00 2011-05-20 03:57:59 8797.16527
      7839 KING       1981-11-17 00:00:00 2011-05-20 03:57:59 10776.1653
      7844 TURNER     1981-09-08 00:00:00 2011-05-20 03:57:59 10846.1653
      7876 ADAMS      1987-05-23 00:00:00 2011-05-20 03:57:59 8763.16527
      7900 JAMES      1981-12-03 00:00:00 2011-05-20 03:57:59 10760.1653
      7902 FORD       1981-12-03 00:00:00 2011-05-20 03:57:59 10760.1653
      7934 MILLER     1982-01-23 00:00:00 2011-05-20 03:57:59 10709.1653

14 rows selected.
```

如果想取得入职多少个星期，改怎么办？（对，将得到的天数/7，即可得到一个关于周的一个相对精确的数值；）

3.3 months_between(date1,date2) ：表示取得两个日期之间的月数(date1-date2)；（一个关于月的一个相对精确的数值）

```
SQL> show user
USER is "SCOTT"
SQL> select empno,ename,hiredate,sysdate, months_between(hiredate,sysdate) result
  2  from emp;

     EMPNO ENAME      HIREDATE            SYSDATE              RESULT
---------- ---------- ------------------- ------------------- ----------
      7369 SMITH      1980-12-17 00:00:00 2011-05-20 04:13:05 -365.10244
      7499 ALLEN      1981-02-20 00:00:00 2011-05-20 04:13:05       -363
      7521 WARD       1981-02-22 00:00:00 2011-05-20 04:13:05 -362.94115
      7566 JONES      1981-04-02 00:00:00 2011-05-20 04:13:05 -361.58631
      7654 MARTIN     1981-09-28 00:00:00 2011-05-20 04:13:05  -355.7476
      7698 BLAKE      1981-05-01 00:00:00 2011-05-20 04:13:05 -360.61857
      7782 CLARK      1981-06-09 00:00:00 2011-05-20 04:13:05 -359.36051
      7788 SCOTT      1987-04-19 00:00:00 2011-05-20 04:13:05 -289.03793
      7839 KING       1981-11-17 00:00:00 2011-05-20 04:13:05 -354.10244
      7844 TURNER     1981-09-08 00:00:00 2011-05-20 04:13:05 -356.39277
      7876 ADAMS      1987-05-23 00:00:00 2011-05-20 04:13:05  -287.9089
      7900 JAMES      1981-12-03 00:00:00 2011-05-20 04:13:05 -353.55406
      7902 FORD       1981-12-03 00:00:00 2011-05-20 04:13:05 -353.55406
      7934 MILLER     1982-01-23 00:00:00 2011-05-20 04:13:05  -351.9089

14 rows selected.
```

3.4 add_months(date,n) ：表示在所给时间的基础上加减多少个月；

注意：这里的月表示自然月，每月的天数由数据库自动计算取得，不用认为指定每月多少天；

n 的值可以为小数，但是运算时会将 n 向下取整；

```
SQL> select sysdate, add_months(sysdate, 3) result from dual;

SYSDATE             RESULT
------------------- -------------------
2011-05-20 04:17:11 2011-08-20 04:17:11

SQL> select sysdate, add_months(sysdate, 1.2) result from dual;

SYSDATE             RESULT
------------------- -------------------
2011-05-20 04:17:21 2011-06-20 04:17:21
```

3.5 next_day(date,’ char’) ：表示在所给时间的基础上，看一下下个星期几是多少号；

注意：char 不一定要写成英文的星期，取决于数据库服务器的字符集编码；

例：数据库服务器的字符集编码为英文时，‘friday’；

数据库服务器的字符集编码为中文时，‘星期五’；

数据库服务器的字符集编码为日文时，‘金曜日’；

```
SQL> !cal
      May 2011
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31


SQL> select sysdate, next_day(sysdate,'friday') result  from dual;

SYSDATE             RESULT
------------------- -------------------
2011-05-20 04:18:10 2011-05-27 04:18:10
```

3.6 last_day(date) ：表示取得所给时间所在月的最后一天；

```
SQL> select sysdate, last_day(sysdate) result  from dual;

SYSDATE             RESULT
------------------- -------------------
2011-05-20 04:21:52 2011-05-31 04:21:52
```

3.7 round(date[,' fmt']) ：表示将所给的时间四舍五入保留到年，月；

例：将当前系统时间保留到月；（保留到月的时候，如果对应的天数在 1-15 之间，则舍去，结果显示为当月的第 1 天；如果对应的天数在 16-*之间，则往上加 1 个月，结果显示为下个月的第 1 天）

```
SQL> alter session set nls_date_format = 'yyyy-mm-dd hh24:mi:ss';

Session altered.

SQL> select sysdate from dual;

SYSDATE
-------------------
2011-05-21 00:01:42

SQL> select round(sysdate,'month') result from dual;

RESULT
-------------------
2011-06-01 00:00:00
```

例：将当前系统时间保留到年；（保留到年的时候，如果对应的月数在 1-6 之间，则舍去，结果显示为当年的第 1 天；如果对应的天数在 7-12 之间，则往上加 1 个年，结果显示为下个年的第 1 天）

```
SQL> alter session set nls_date_format = 'yyyy-mm-dd hh24:mi:ss';

Session altered.

SQL> select sysdate from dual;

SYSDATE
-------------------
2011-05-21 00:02:58

SQL> select round(sysdate,'year') result from dual;

RESULT
-------------------
2011-01-01 00:00:00
```

3.8 trunc(date[,' fmt'])：表示将所给的时间截断到年，月；

例：将当前系统时间截断到月；（不必考虑天数问题，直接截断，结果显示为当前月的第 1 天）

```
SQL> alter session set nls_date_format = 'yyyy-mm-dd hh24:mi:ss';

Session altered.

SQL> select sysdate from dual;

SYSDATE
-------------------
2011-05-21 00:05:48

SQL> select trunc(sysdate,'month') result from dual;

RESULT
-------------------
2011-05-01 00:00:00
```

例：将当前系统时间截断到年；（不必考虑月数问题，直接截断，结果显示为当前年的第 1 天）

```
SQL> alter session set nls_date_format = 'yyyy-mm-dd hh24:mi:ss';

Session altered.

SQL> select sysdate from dual;

SYSDATE
-------------------
2011-05-21 00:06:19

SQL> select trunc(sysdate,'year') result from dual;

RESULT
-------------------
2011-01-01 00:00:00
```

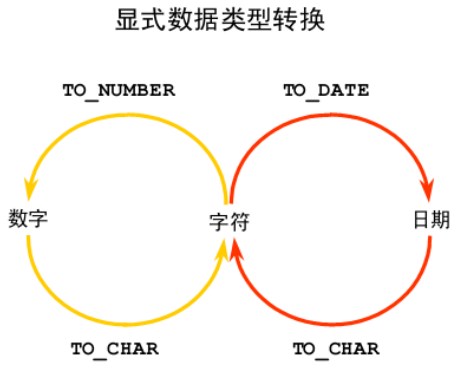
4. 隐式的数据类型转换（了解，工作中基本不用隐式转换的方式）

注意：隐式转换的时候，如果自动完成没有问题，则正常存取；如果自动转换完成不了，则报错！

对于直接赋值， Oracle 服务器能够自动地进行下面的转换：

从	到
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE
NUMBER	VARCHAR2
DATE	VARCHAR2

5. 显示的数据类型转换



5.1 to_char(number|date,’ fmt’)：表示将 number 或者 date 类型的数据按照 fmt 指定的格式转换成字符串；

例：将系统时间 sysdate 转换成字符串；

注意：默认的时间显示格式为日-月-年，可以使用 nls_date_format 进行修正，但此实验中暂时不用；

```
SQL> show user
USER is "SCOTT"
SQL> select sysdate from dual;

SYSDATE
---------
21-MAY-11

SQL> select to_char(sysdate,'yyyy/mm/dd hh24:mi:ss') result from dual;

RESULT
-------------------
2011/05/21 00:20:58
```

当然日期的年、月、日的显示格式还有很多，以下是常用的类型；

```
SQL> select to_char(sysdate,'yyyy') t1, to_char(sysdate,'year') t2,
  2         to_char(sysdate,'mm') t3, to_char(sysdate,'month') t4, to_char(sysdate,'mon') t5,
  3         to_char(sysdate,'dd') t6, to_char(sysdate,'day') t7, to_char(sysdate,'dy') t8
  4  from dual;

T1   T2                                        T3 T4        T5  T6 T7        T8
---- ----------------------------------------- -- --------- --- -- --------- ---
2011 twenty eleven                             05 may       may 21 saturday  sat
```

当然日期的小时、分钟、秒的显示格式还有很多，以下是常用的类型；

```
SQL> select to_char(sysdate,'hh24:mi:ss am') t1,
  2         to_char(sysdate,'dd "of" month') t2,
  3         to_char(sysdate,'ddspth') t3
  4  from dual;

T1          T2              T3
----------- --------------- --------------
00:28:46 am 21 of may       twenty-first
```

例：讲给定的数字 12345678.11223344 转换成字符串，并设定一定的格式；

```
SQL> select to_char(12345678.11223344,'$099,999,999.99999999') result from dual;

RESULT
----------------------
 $012,345,678.11223344
```

注意：0 或 9 都表示显示一位数字；但 0 也表示前导零，有 1 个即可；

如果格式的整数部分的长度无法涵盖给定数字的整数部分，则无法正常显示；

如果格式的小数部分的长度无法涵盖给定数字的小数部分，将会进行四舍五入进行保留；

```
SQL> select to_char(12345678.11223344,'$09,999.99999999') result from dual;

RESULT
-----------------
#################

SQL> select to_char(12345678.11223344,'$099,999,999.9999') result from dual;

RESULT
------------------
 $012,345,678.1122

SQL> select to_char(12345678.11225566,'$099,999,999.9999') result from dual;

RESULT
------------------
 $012,345,678.1123
```

如果，想以本地货币的形式进行显示，则用 L 开头；（例：本地货币为人民币 RMB）

```
SQL> conn / as sysdba
Connected.
SQL> show user
USER is "SYS"
SQL> alter session set nls_currency = 'RMB';

Session altered.

SQL> select to_char(12345678.11223344,'L099,999,999.99999999') result from dual;

RESULT
--------------------------------
        RMB012,345,678.11223344
```

5.2 to_number(char[, ‘fmt’])：表示将字符串按照某种格式转换成数字；

注意：如果字符串中包含数字之外的字符，则转换会报错！

转换完成后，即可对此结果进行加减乘除四则运算；（略）

```
SQL> select to_number('12345') result from dual;

    RESULT
----------
     12345

SQL> select to_number('0,012,345.6789','0,999,999.9999') result from dual;

    RESULT
----------
12345.6789

SQL> select to_number('0,012,345.6789') result from dual;
select to_number('0,012,345.6789') result from dual
                 *
ERROR at line 1:
ORA-01722: invalid number


SQL> select to_number('abc123') result from dual;
select to_number('abc123') result from dual
                 *
ERROR at line 1:
ORA-01722: invalid number
```

5.3 to_date(char[, ‘fmt’])：表示将字符串按照某种特定格式转换成日期；

注意：如果字符串中包含不可识别的字符，则转换会报错！

转换完成后，即可对此结果进行加减运算；（略）

```
SQL> select sysdate from dual;

SYSDATE
---------
21-MAY-11

SQL> select to_date('25-dec-11') result from dual;

RESULT
---------
25-DEC-11

SQL> select to_date('abcd') from dual;
select to_date('abcd') from dual
               *
ERROR at line 1:
ORA-01858: a non-numeric character was found where a numeric was expected
```

注意：以上给出的字符串 25-dec-11，是日期的默认显示格式，则可以直接进行转换；但是转换后的日期可能会有问题，之后的实验将会说明一切!!

step 1：提示：将给出的字符串转换成日期时，需要对其中的年份进行解析，分为 yy 解析和 rr 解析方式；比如上面给出的 25-dec-11 中的 11，转换成年份的时候，系统就不一定知道是 2011 还是 1911 或者其他；

step 2：原则：yy 解析会把 11 解析成本世纪(数据库服务器所处的世纪)的 AA11 年；

rr 解析会把 11 做判断，得出是上世纪的 BB11 年，或者本世纪的 AA11 年，或者下世纪的 CC11 年；

当前年	指定的日期	RR 格式	YY 格式
1995	27-OCT-95	1995	1995
1995	27-OCT-17	2017	1917
2001	27-OCT-17	2017	2017
2001	27-OCT-95	1995	2095

		如果指定的两位数字年是：	
		0–49	50–99
如果当前年是两位数字表示：	0–49	返回的日期是在当前世纪中	返回的日期是在当前世纪的上个世纪中
	50–99	返回的日期是在当前世纪的下个世纪中	返回的日期是在当前世纪中

step 3：证明：确认一下上面的原则是否正确？！

注意：实验过程中需要修改*数据库服务器的时间*，但是这些操作在工作中禁止！（非法）

step 3.1 ：如果数据库服务器时间为 1995 年，修改时间的显示格式；

注意：修改服务器时间的操作，只有 sys 用户有权限；

```
SQL> conn / as sysdba
Connected.
SQL> show user
USER is "SYS"
SQL> alter system set fixed_date = '1995-01-01';

System altered.

SQL> select sysdate from dual;

SYSDATE
---------
01-JAN-95

SQL> alter session set nls_date_format = 'yyyy-mm-dd';

Session altered.

SQL> select sysdate from dual;

SYSDATE
----------
1995-01-01
```

step 3.2 ：对给出的‘27-oct-95’，‘27-oct-17’进行 yy 和 rr 解析；

```
SQL> select to_date('27-oct-95','dd-mon-yy') t1, to_date('27-oct-17','dd-mon-yy') t2
  2  from dual;

T1         T2
---------- ----------
1995-10-27 1917-10-27

SQL> select to_date('27-oct-95','dd-mon-rr') t1, to_date('27-oct-17','dd-mon-rr') t2
  2  from dual;

T1         T2
---------- ----------
1995-10-27 2017-10-27
```

初步结论 1：yy 在解析 17 时，结果为 1917，感觉有些不太合理，因为这个时间离 1995 年差的太久；95 解析的还好点；

rr 在解析 17 和 95 时，得到的结果离当前的 1995 年差的不太多，感觉不错！

step 3.3 ：如果数据库服务器时间为 2001 年，修改时间的显示格式；

```
SQL> show user
USER is "SYS"
SQL> alter system set fixed_date = '2001-01-01';

System altered.

SQL> alter session set nls_date_format = 'yyyy-mm-dd';

Session altered.

SQL> select sysdate from dual;

SYSDATE
----------
2001-01-01
```

step 3.4 ：同样对给出的‘27-oct-95’，‘27-oct-17’进行 yy 和 rr 解析；

```
SQL> select to_date('27-oct-95','dd-mon-yy') t1, to_date('27-oct-17','dd-mon-yy') t2
  2  from dual;

T1         T2
---------- ----------
2095-10-27 2017-10-27

SQL> select to_date('27-oct-95','dd-mon-rr') t1, to_date('27-oct-17','dd-mon-rr') t2
  2  from dual;

T1         T2
---------- ----------
1995-10-27 2017-10-27
```

初步结论 2：yy 在解析 95 时，结果为 2095，感觉有些不太合理，因为这个时间离 2001 年差的太久；17 解析的还好点；

rr 在解析 95 和 17 时，得到的结果离当前的 2001 年差的不太多，感觉不错！

【最终结论】

1. 好像在通过 yy 解析的时候，可能会出现不太合理的结果；而 rr 解析得到的结果却一直挺好！
2. 在工作中，多数情况会使用 yy 解析；
3. 只有在跨世纪或者处理多年前的历史数据的时候，才会出现 yy 解析不正常的现象，才会用到 rr 解析；

提醒：喂！明白这个后，别忘了，你做了修改服务器时间的操作，记得改回来!!

```
SQL> show user
USER is "SYS"
SQL> alter system set fixed_date = none;

System altered.
```

这样的话，时间会恢复回去，但是会造成时间紊乱，影响数据库中的数据，因此 fixed_date 禁止修改！

6. 通用函数的使用(适用于任意数据类型，并且适用于 null 值)

6.1 nvl(e1,e2)：如果第 1 个参数 e1 的值为 null，则返回 e2 的值；

```
SQL> select nvl(null,'haha') s1, nvl(null,1234) n1, nvl(null,sysdate) t1
  2  from dual;

S1           N1 T1
---- ---------- -------------------
haha       1234 2011-05-21 01:41:32
```

注意：null 被标识为空，同样空串 ‘’ 也会被数据库标识为空；

```
SQL> select nvl('','haha') s1, nvl('',1234) n1, nvl('',sysdate) t1
  2  from dual;

S1   N1   T1
---- ---- -------------------
haha 1234 2011-05-21 01:43:16
```

6.2 nvl2(e1,e2,e3)：如果第 1 个参数 e1 的值为 null，则返回 e3 的值；

如果第 1 个参数 e1 的值不为 null，则返回 e2 的值；

```
SQL> select nvl2(null,'e2','e3') s1, nvl2('abc123','e2','e3') s2
  2  from dual;

S1 S2
-- --
e3 e2
```

6.3 nullif(e1,e2)：如果 e1=e2，则返回 null；如果 e1<>e2，则返回 e1；

注意：第 1 个参数 e1 不能为 null；但可以为空串 ‘’；

两个参数的类型需要一致；

```
SQL> select nullif('abc','abc') r1, nullif(123,456) r2,
  2         nullif('','') r3, nullif('','abc') r4
  3  from dual;

R1          R2 R R
--- ---------- - -
           123
```

6.4 coalesce(e1,e2,e3,…en)：返回参数中第 1 个非空表达式；

注意：null 属于空，空串 ‘’ 也属于空；

```
SQL> select coalesce(null,'','abc','def') r1  from dual;

R1
---
abc
```

但是，如果前 n-1 个表达式都为空，则返回第 n 个参数，而不论其是否为空；

```
SQL> select coalesce(null,null,'','',null) r1 from dual;

R
-
```

7. case 和 decode 的用法(实现了 if-then-else 的分支效果)

7.1 case…end ：一个完整的 case 语句，必须有 end；

step 1：准备测试数据；

```
SQL> show user
USER is "SCOTT"
SQL> set linesize 120
SQL> set pagesize 30
SQL> select * from emp;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10

14 rows selected.
```

step 2：假定条件：

如果 JOB 是 CLERK，则 SAL 增加 1；----------【分支 1】

如果 JOB 是 SALESMAN，则 SAL 增加 2；-------【分支 2】

如果 JOB 是 MANAGER，则 SAL 增加 3；--------【分支 3】

如果 JOB 是其他，则 SAL 不变；----------【分支 4】

step 3：

第 1 种写法：

注意：job 属于以上 3 种以外的，则 sal 不变；因为有 else 分支；

```
SQL> select empno,ename,job,sal old_sal,
  2         case job
  3           when 'CLERK' then sal+1
  4           when 'SALESMAN' then sal+2
  5           when 'MANAGER' then sal+3
  6           else sal
  7         end as new_sal
  8  from emp
  9  order by job;

     EMPNO ENAME      JOB          OLD_SAL    NEW_SAL
---------- ---------- --------- ---------- ----------
      7788 SCOTT      ANALYST          3000       3000
      7902 FORD       ANALYST          3000       3000
      7934 MILLER     CLERK            1300       1301
      7900 JAMES      CLERK             950        951
      7369 SMITH      CLERK             800        801
      7876 ADAMS      CLERK            1100       1101
      7698 BLAKE      MANAGER          2850       2853
      7566 JONES      MANAGER          2975       2978
      7782 CLARK      MANAGER          2450       2453
      7839 KING       PRESIDENT        5000       5000
      7844 TURNER     SALESMAN         1500       1502
      7654 MARTIN     SALESMAN         1250       1252
      7521 WARD       SALESMAN         1250       1252
      7499 ALLEN      SALESMAN         1600       1602

14 rows selected.
```

第 2 种写法：

```
SQL> select empno,ename,job,sal old_sal,
  2    case when job='CLERK' then sal+1
  3         when job='SALESMAN' then sal+2
  4         when job='MANAGER' then sal+3
  5         else sal
  6    end as new_sal
  7  from emp
  8  order by job;

     EMPNO ENAME      JOB          OLD_SAL    NEW_SAL
---------- ---------- --------- ---------- ----------
      7788 SCOTT      ANALYST          3000       3000
      7902 FORD       ANALYST          3000       3000
      7934 MILLER     CLERK            1300       1301
      7900 JAMES      CLERK             950        951
      7369 SMITH      CLERK             800        801
      7876 ADAMS      CLERK            1100       1101
      7698 BLAKE      MANAGER          2850       2853
      7566 JONES      MANAGER          2975       2978
      7782 CLARK      MANAGER          2450       2453
      7839 KING       PRESIDENT        5000       5000
      7844 TURNER     SALESMAN         1500       1502
      7654 MARTIN     SALESMAN         1250       1252
      7521 WARD       SALESMAN         1250       1252
      7499 ALLEN      SALESMAN         1600       1602

14 rows selected.
```

注意：第 2 种写法的好处就是可以在 when 条件中写复合条件；

注意：当分支 4，即 else 分支不存在的情况下， sal 将会返回 null 值(显示在 new_sal 中)。

```
SQL> del 5
SQL> l
  1  select empno,ename,job,sal old_sal,
  2    case when job='CLERK' then sal+1
  3         when job='SALESMAN' then sal+2
  4         when job='MANAGER' then sal+3
  5    end as new_sal
  6  from emp
  7* order by job
SQL> /

     EMPNO ENAME      JOB          OLD_SAL    NEW_SAL
---------- ---------- --------- ---------- ----------
      7788 SCOTT      ANALYST         3000
      7902 FORD       ANALYST         3000
      7934 MILLER     CLERK           1300       1301
      7900 JAMES      CLERK            950        951
      7369 SMITH      CLERK            800        801
      7876 ADAMS      CLERK           1100       1101
      7698 BLAKE      MANAGER         2850       2853
      7566 JONES      MANAGER         2975       2978
      7782 CLARK      MANAGER         2450       2453
      7839 KING       PRESIDENT       5000
      7844 TURNER     SALESMAN        1500       1502
      7654 MARTIN     SALESMAN        1250       1252
      7521 WARD       SALESMAN        1250       1252
      7499 ALLEN      SALESMAN        1600       1602

14 rows selected.
```

7.2 decode(col|expr, search1, result1[, search2, result2, …,][, default])

注意：decode 只是将 case 的书写方式进行了简化；

例：按照 7.1 的 step 2 的条件进行查询；

```
SQL> select empno,ename,job,sal old_sal,
  2    decode(job,'CLERK',   sal+1,
  3               'SALESMAN',sal+2,
  4               'MANAGER', sal+3,
  5                          sal)  as new_sal
  6  from emp
  7  order by job;

     EMPNO ENAME      JOB          OLD_SAL    NEW_SAL
---------- ---------- --------- ---------- ----------
      7788 SCOTT      ANALYST         3000       3000
      7902 FORD       ANALYST         3000       3000
      7934 MILLER     CLERK           1300       1301
      7900 JAMES      CLERK            950        951
      7369 SMITH      CLERK            800        801
      7876 ADAMS      CLERK           1100       1101
      7698 BLAKE      MANAGER         2850       2853
      7566 JONES      MANAGER         2975       2978
      7782 CLARK      MANAGER         2450       2453
      7839 KING       PRESIDENT       5000       5000
      7844 TURNER     SALESMAN        1500       1502
      7654 MARTIN     SALESMAN        1250       1252
      7521 WARD       SALESMAN        1250       1252
      7499 ALLEN      SALESMAN        1600       1602

14 rows selected.
```

注意：如果 decode 中的 default 值省略了，结果就像没有 else 分支一样；

```
SQL> select empno,ename,job,sal old_sal,
  2    decode(job,'CLERK',   sal+1,
  3               'SALESMAN',sal+2,
  4               'MANAGER', sal+3)  as new_sal
  5  from emp
  6  order by job;

     EMPNO ENAME      JOB          OLD_SAL    NEW_SAL
---------- ---------- --------- ---------- ----------
      7788 SCOTT      ANALYST         3000
      7902 FORD       ANALYST         3000
      7934 MILLER     CLERK           1300       1301
      7900 JAMES      CLERK            950        951
      7369 SMITH      CLERK            800        801
      7876 ADAMS      CLERK           1100       1101
      7698 BLAKE      MANAGER         2850       2853
      7566 JONES      MANAGER         2975       2978
      7782 CLARK      MANAGER         2450       2453
      7839 KING       PRESIDENT       5000
      7844 TURNER     SALESMAN        1500       1502
      7654 MARTIN     SALESMAN        1250       1252
      7521 WARD       SALESMAN        1250       1252
      7499 ALLEN      SALESMAN        1600       1602

14 rows selected.
```

【第四章　从多表中查询数据】

1.笛卡尔积：例，表 t06 和 t07 进行笛卡尔积关联时，t06 中的每一条记录都会和 t07 中每一条记录进行关联；

注意：此时得到的结果中，有些数据是不正常的，成为业务中的‘垃圾数据’（没有实际的意义）；但是，有时候需要短时间内生成大量的测试数据，便于进行压力测试，或者性能测试，此时笛卡尔积才有点意义；

例：说明一下笛卡尔积的弊端；

step 1：建立测试用表 t06 和 t07，注意应在测试用户 scott 下建立；

```
SQL> create table t06(id number(2),name varchar2(20),deptno number(2));

Table created.

SQL> create table t07(deptno number(2),dname varchar2(20),loc varchar2(20));

Table created.
```

step 2：插入测试数据，并提交；

```
SQL> insert into t06 values(1,'xiaozhang',10);

1 row created.

SQL> insert into t06 values(2,'xiaowang',20);

1 row created.

SQL> insert into t06 values(2,'xiaoli',30);

1 row created.

SQL> commit;

Commit complete.

SQL> insert into t07 values(10,'itaa','tj1');

1 row created.

SQL> insert into t07 values(20,'itbb','tj2');

1 row created.

SQL> insert into t07 values(30,'itcc','tj3');

1 row created.

SQL> commit;

Commit complete.
```

step 3：说明一下笛卡尔积生成数据的问题；

注意：本来应该是可以确定每个人在哪个部门，但是因为笛卡尔积的原因，出现了不正常数据；

```
SQL> select rownum,t06.name,t07.dname
  2  from t06,t07;

    ROWNUM NAME                 DNAME
---------- -------------------- --------------------
         1 xiaozhang            itaa
         2 xiaozhang            itbb
         3 xiaozhang            itcc
         4 xiaowang             itaa
         5 xiaowang             itbb
         6 xiaowang             itcc
         7 xiaoli               itaa
         8 xiaoli               itbb
         9 xiaoli               itcc

9 rows selected.
```

注意：第 1,5,9 条数据为正常数据；

2.等值连接(或者简单连接、内连接)

例：想知道 emp 表中，每位员工所在的部门的名称；

step 1：查询 emp、dept 表的内容；

```
SQL> select * from emp;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10

14 rows selected.

SQL> select * from dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

step 2：通过 deptno 将两张表进行关联；

方法一：使用简单连接(等值连接)的方式实现；

```
SQL> select e.empno, e.ename, e.deptno, d.dname
  2  from emp e,dept d
  3  where e.deptno = d.deptno;

     EMPNO ENAME          DEPTNO DNAME
---------- ---------- ---------- --------------
      7369 SMITH              20 RESEARCH
      7499 ALLEN              30 SALES
      7521 WARD               30 SALES
      7566 JONES              20 RESEARCH
      7654 MARTIN             30 SALES
      7698 BLAKE              30 SALES
      7782 CLARK              10 ACCOUNTING
      7788 SCOTT              20 RESEARCH
      7839 KING               10 ACCOUNTING
      7844 TURNER             30 SALES
      7876 ADAMS              20 RESEARCH
      7900 JAMES              30 SALES
      7902 FORD               20 RESEARCH
      7934 MILLER             10 ACCOUNTING

14 rows selected.
```

注意：使用表别名的原因：因为两张表中都含有 deptno 字段，为了避免混淆，所以要必须指定此字段来自于哪张表；

方法二：使用内连接的方式实现；

```
SQL> select e.empno, e.ename, e.deptno, d.dname
  2  from emp e
  3  inner join dept d
  4  on e.deptno = d.deptno;

     EMPNO ENAME          DEPTNO DNAME
---------- ---------- ---------- --------------
      7369 SMITH              20 RESEARCH
      7499 ALLEN              30 SALES
      7521 WARD               30 SALES
      7566 JONES              20 RESEARCH
      7654 MARTIN             30 SALES
      7698 BLAKE              30 SALES
      7782 CLARK              10 ACCOUNTING
      7788 SCOTT              20 RESEARCH
      7839 KING               10 ACCOUNTING
      7844 TURNER             30 SALES
      7876 ADAMS              20 RESEARCH
      7900 JAMES              30 SALES
      7902 FORD               20 RESEARCH
      7934 MILLER             10 ACCOUNTING

14 rows selected.
```

注意：等值连接(简单连接，内连接)出现的结果，是在多张表中都需要存在的数据，即他们能够匹配上，如果匹配不上，则这样的结果不会出现；

注意：在使用简单连接(等值连接)的时候，如果连接的表多于 2 个，则至少使用 n-1 个有效的连接条件(n：数据表的个数)；

3．非等值连接：(使用非等号进行条件连接，例> < != <= >=)

例：想知道 emp 表中每个员工的薪水所处的级别；

step 1：查询 emp 表和 salgrade 表的信息；

```
SQL> select * from emp;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10

14 rows selected.

SQL> select * from salgrade;

     GRADE      LOSAL      HISAL
---------- ---------- ----------
         1        700       1200
         2       1201       1400
         3       1401       2000
         4       2001       3000
         5       3001       9999
```

step 2：使用非等值连接查询员工的薪水级别；

```
SQL> select e.ename, e.sal, j.grade
  2  from emp e, salgrade j
  3  where e.sal between j.losal and j.hisal;

ENAME             SAL      GRADE
---------- ---------- ----------
SMITH             800          1
JAMES             950          1
ADAMS            1100          1
WARD             1250          2
MARTIN           1250          2
MILLER           1300          2
TURNER           1500          3
ALLEN            1600          3
CLARK            2450          4
BLAKE            2850          4
JONES            2975          4
SCOTT            3000          4
FORD             3000          4
KING             5000          5

14 rows selected.
```

注意：between…and…在真正编译的时候，会转换成>= and <= ；

4. 自连接(从表自身内部的关联中查询相关信息)

例：想知道 emp 表中，每位员工对应的上级主管的名字；

分析：员工对应的经理，其实他(经理)也是公司的员工；想一想，老板也是员工，哈哈！

step 1：查询 emp 表的信息；

```
SQL> select * from emp;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10

14 rows selected.
```

step 2：确定 mgr 和 empno 的关系；(其实员工对应的经理的编号，正是经理的员工编号)

```
SQL> select e.empno,e.ename,e.mgr, m.empno,m.ename
  2  from emp e, emp m
  3  where e.mgr = m.empno;

     EMPNO ENAME             MGR      EMPNO ENAME
---------- ---------- ---------- ---------- ----------
      7902 FORD             7566       7566 JONES
      7788 SCOTT            7566       7566 JONES
      7900 JAMES            7698       7698 BLAKE
      7844 TURNER           7698       7698 BLAKE
      7654 MARTIN           7698       7698 BLAKE
      7521 WARD             7698       7698 BLAKE
      7499 ALLEN            7698       7698 BLAKE
      7934 MILLER           7782       7782 CLARK
      7876 ADAMS            7788       7788 SCOTT
      7782 CLARK            7839       7839 KING
      7698 BLAKE            7839       7839 KING
      7566 JONES            7839       7839 KING
      7369 SMITH            7902       7902 FORD

13 rows selected.
```

注意：哎，少了一个员工的信息，谁？！ 对，KING，因为他是头儿，没有他的老板的信息!!

5．外连接(分为左外连接、右外连接、全外连接)

5.1 左外连接(left [outer] join)

例：查询一下 emp 表中 14 位员工对应的部门的名称？

注意：使用左连接的时候，以左边表(员工信息)的记录为主，若有右边表(部门)的信息与之对应，则右表的信息显示出来；如没有右表的信息与之对应，则右表相关的信息显示为空；

而相应的左表的信息全部都会显示，右表的信息，可能有，可能空；

step 1：参照 emp 创建员工表 t08，参照 dept 创建部门表 t09；

```
SQL> show user
USER is "SCOTT"
SQL> create table t08
  2  as select * from emp;

Table created.

SQL> create table t09
  2  as select * from dept;

Table created.
```

step 2：作测试数据，并提交；

```
SQL> update t08 set deptno = 99
  2  where empno in (7369,7499,7782);

3 rows updated.

SQL> commit;

Commit complete.
```

step 3：查看表中的测试数据；

```
SQL> set linesize 120
SQL> set pagesize 30
SQL> select * from t08;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    99
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         99
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    99
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10

14 rows selected.

SQL> select * from t09;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

注意：t08 中 deptno 为 99 的数据(注意点),因为 99 号部门在部门表 t09 中没有相关信息；

此处仅测试练习用，工作中要求(员工的信息中必须要有部门的信息，而此部门必须要在部门中有说明)；

step 4：左连接，测试一下，并查看结果；

```
SQL> select e.empno, e.ename, e.deptno, d.deptno, d.dname
  2  from t08 e
  3  left join t09 d
  4  on e.deptno = d.deptno;

     EMPNO ENAME          DEPTNO     DEPTNO DNAME
---------- ---------- ---------- ---------- --------------
      7934 MILLER             10         10 ACCOUNTING
      7839 KING               10         10 ACCOUNTING
      7902 FORD               20         20 RESEARCH
      7876 ADAMS              20         20 RESEARCH
      7788 SCOTT              20         20 RESEARCH
      7566 JONES              20         20 RESEARCH
      7900 JAMES              30         30 SALES
      7844 TURNER             30         30 SALES
      7698 BLAKE              30         30 SALES
      7654 MARTIN             30         30 SALES
      7521 WARD               30         30 SALES
      7782 CLARK              99
      7499 ALLEN              99
      7369 SMITH              99

14 rows selected.
```

注意：因为部门表 t09 中没有 99 的信息，所以结果中，编号为 7782、7499，7369 这 3 个人的部门信息为空；

注意：以上是 SQL-99 的语法标准，同样 db2，sql server，my sql 等都满足此标准；但是对于各自的产品，也会有一些特性；

例：在 oracle 中，左连接是这样写的：（如下）

```
SQL> select e.empno, e.ename, e.deptno, d.deptno, d.dname
  2  from t08 e, t09 d
  3  where e.deptno = d.deptno(+);

     EMPNO ENAME          DEPTNO     DEPTNO DNAME
---------- ---------- ---------- ---------- --------------
      7934 MILLER             10         10 ACCOUNTING
      7839 KING               10         10 ACCOUNTING
      7902 FORD               20         20 RESEARCH
      7876 ADAMS              20         20 RESEARCH
      7788 SCOTT              20         20 RESEARCH
      7566 JONES              20         20 RESEARCH
      7900 JAMES              30         30 SALES
      7844 TURNER             30         30 SALES
      7698 BLAKE              30         30 SALES
      7654 MARTIN             30         30 SALES
      7521 WARD               30         30 SALES
      7782 CLARK              99
      7499 ALLEN              99
      7369 SMITH              99

14 rows selected.
```

注意：写成这样的话，也是左连接：（如下）

```
SQL> select e.empno, e.ename, e.deptno, d.deptno, d.dname
  2  from t08 e, t09 d
  3   where d.deptno(+) = e.deptno;
```

结论：1.加号(+)在右表，就属于左连接，以左表数据为基准嘛！(注：左右表的区分要看在 from 中的位置)；　2. 由于(+)的位置容易混淆连接的方式，所以工作中在 oracle 的环境中多采用 sql-99 的书写方式(left [outer] join)，便于识别连接方式；

5.2 右外连接(right [outer] join)

例：如果想知道每个部门中都有哪些员工？

step 1：测试数据参照 5.1 中的 t08 和 t09；

```
SQL> show user
USER is "SCOTT"
SQL> select * from t08;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    99
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         99
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    99
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10

14 rows selected.

SQL> select * from t09;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

step 2：书写右连接；

注意：1.表的前后顺序； 2.(+)所在的位置； 3.注意以哪张表(t09)为基准表；

```
SQL> select e.empno, e.ename, e.deptno, d.deptno, d.dname
  2  from t08 e
  3  right join t09 d
  4  on e.deptno = d.deptno;

     EMPNO ENAME          DEPTNO     DEPTNO DNAME
---------- ---------- ---------- ---------- --------------
      7521 WARD               30         30 SALES
      7566 JONES              20         20 RESEARCH
      7654 MARTIN             30         30 SALES
      7698 BLAKE              30         30 SALES
      7788 SCOTT              20         20 RESEARCH
      7839 KING               10         10 ACCOUNTING
      7844 TURNER             30         30 SALES
      7876 ADAMS              20         20 RESEARCH
      7900 JAMES              30         30 SALES
      7902 FORD               20         20 RESEARCH
      7934 MILLER             10         10 ACCOUNTING
                                         40 OPERATIONS

12 rows selected.
```

注意：t09 中的 40 号部门，有部门信息，但是没有员工；

t08 中的 99 号部门，没有包含在 t09 中，所以不会出现在右边列(d.dname)中；

例：在 oracle 中，右连接是这样写的：（如下）

```
SQL> select e.empno, e.ename, e.deptno, d.deptno, d.dname
  2  from t08 e, t09 d
  3  where e.deptno(+) = d.deptno;

     EMPNO ENAME          DEPTNO     DEPTNO DNAME
---------- ---------- ---------- ---------- --------------
      7521 WARD               30         30 SALES
      7566 JONES              20         20 RESEARCH
      7654 MARTIN             30         30 SALES
      7698 BLAKE              30         30 SALES
      7788 SCOTT              20         20 RESEARCH
      7839 KING               10         10 ACCOUNTING
      7844 TURNER             30         30 SALES
      7876 ADAMS              20         20 RESEARCH
      7900 JAMES              30         30 SALES
      7902 FORD               20         20 RESEARCH
      7934 MILLER             10         10 ACCOUNTING
                                         40 OPERATIONS

12 rows selected.
```

注意：写成这样的话，也是右连接：（如下）

```
SQL> select e.empno, e.ename, e.deptno, d.deptno, d.dname
  2  from t08 e, t09 d
  3  where d.deptno = e.deptno(+);

     EMPNO ENAME          DEPTNO     DEPTNO DNAME
---------- ---------- ---------- ---------- --------------
      7521 WARD               30         30 SALES
      7566 JONES              20         20 RESEARCH
      7654 MARTIN             30         30 SALES
      7698 BLAKE              30         30 SALES
      7788 SCOTT              20         20 RESEARCH
      7839 KING               10         10 ACCOUNTING
      7844 TURNER             30         30 SALES
      7876 ADAMS              20         20 RESEARCH
      7900 JAMES              30         30 SALES
      7902 FORD               20         20 RESEARCH
      7934 MILLER             10         10 ACCOUNTING
                                         40 OPERATIONS

12 rows selected.
```

结论：1.右连接和左连接只是相对而言(表的顺序不一样，连接的方式也就不同了)；
2.其实右连接可以转换为左连接，工作中用左连接居多；
3.为避免混淆，还是用 sql-99 吧！

5.3 全外连接(full [outer] join)

注意：1.左连接和右连接的重合部分，为内连接的结果；
2.全连接 = 左连接 + 右连接 + [内连接]，然后重复的记录保留 1 份；

例：全连接：

```
SQL> set linesize 120
SQL> set pagesize 30
SQL> select e.empno, e.ename, e.deptno, d.deptno, d.dname
  2  from t08 e
  3  full join t09 d
  4  on e.deptno = d.deptno;

     EMPNO ENAME          DEPTNO     DEPTNO DNAME
---------- ---------- ---------- ---------- --------------
      7934 MILLER             10         10 ACCOUNTING
      7839 KING               10         10 ACCOUNTING
      7902 FORD               20         20 RESEARCH
      7876 ADAMS              20         20 RESEARCH
      7788 SCOTT              20         20 RESEARCH
      7566 JONES              20         20 RESEARCH
      7900 JAMES              30         30 SALES
      7844 TURNER             30         30 SALES
      7698 BLAKE              30         30 SALES
      7654 MARTIN             30         30 SALES
      7521 WARD               30         30 SALES
      7782 CLARK              99
      7499 ALLEN              99
      7369 SMITH              99
                                         40 OPERATIONS

15 rows selected.
```

注意：full join 之后的结果，不会进行排序；

----------*----------

注意：以上结果按照 empno 升序排列后与左连接 union 右连接进行比较；
提示：union 表示将两块数据集进行并集，然后默认按照第 1 列进行升序排列；

[full join 排序后的结果]

```
SQL> select e.empno, e.ename, e.deptno, d.deptno, d.dname
  2  from t08 e
  3  full join t09 d
  4  on e.deptno = d.deptno
  5  order by e.empno asc;

     EMPNO ENAME          DEPTNO     DEPTNO DNAME
---------- ---------- ---------- ---------- --------------
      7369 SMITH              99
      7499 ALLEN              99
      7521 WARD               30         30 SALES
      7566 JONES              20         20 RESEARCH
      7654 MARTIN             30         30 SALES
      7698 BLAKE              30         30 SALES
      7782 CLARK              99
      7788 SCOTT              20         20 RESEARCH
      7839 KING               10         10 ACCOUNTING
      7844 TURNER             30         30 SALES
      7876 ADAMS              20         20 RESEARCH
      7900 JAMES              30         30 SALES
      7902 FORD               20         20 RESEARCH
      7934 MILLER             10         10 ACCOUNTING
                                         40 OPERATIONS

15 rows selected.
```

[union 的结果]-----(即，将左连接和右连接取并集)

```
SQL> select e.empno, e.ename, e.deptno, d.deptno, d.dname
  2  from t08 e left join t09 d
  3  on e.deptno = d.deptno
  4  union
  5  select e.empno, e.ename, e.deptno, d.deptno, d.dname
  6  from t08 e right join t09 d
  7  on e.deptno = d.deptno;

     EMPNO ENAME          DEPTNO     DEPTNO DNAME
---------- ---------- ---------- ---------- --------------
      7369 SMITH              99
      7499 ALLEN              99
      7521 WARD               30         30 SALES
      7566 JONES              20         20 RESEARCH
      7654 MARTIN             30         30 SALES
      7698 BLAKE              30         30 SALES
      7782 CLARK              99
      7788 SCOTT              20         20 RESEARCH
      7839 KING               10         10 ACCOUNTING
      7844 TURNER             30         30 SALES
      7876 ADAMS              20         20 RESEARCH
      7900 JAMES              30         30 SALES
      7902 FORD               20         20 RESEARCH
      7934 MILLER             10         10 ACCOUNTING
                                         40 OPERATIONS

15 rows selected.
```

结论：full join 得到的结果按照第 1 列升序排列后 等于 左连接 union 右连接的结果；

【第五章 组函数的使用】

1. 常用组函数的使用；

1.1 函数说明：(distinct 表示去除重复值；all 表示所有的值，包括重复值(默认选项)；)

注意：使用组函数时，空值被自动忽略(count(*)例外)；但是可以使用 nvl、nvl2 等将空值进行转换；

avg([distinct|all] expr)：表示计算 expr 的平均值；

sum([distinct|all] expr)：表示计算 expr 的总计值；

max([distinct|all] expr)：表示计算 expr 的最大值；

min([distinct|all] expr)：表示计算 expr 的最小值；

count(*|[distinct|all] expr)：表示计算表中数据的总数；

1.2 实验证明：

step 1：查看 emp 表的信息；

```
SQL> show user
USER is "SCOTT"
SQL> set linesize 120
SQL> set pagesize 30
SQL> select * from emp;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10

14 rows selected.
```

step 2：使用相关组函数，并进行注意说明；

例：计算员工的平均工资、平均奖金；

```
SQL> select avg(sal), avg(comm) from emp;

  AVG(SAL)  AVG(COMM)
---------- ----------
2073.21429        550
```

注意：平均奖金 550 是几个人的？ 4 个人的是吧！ 如果是所有人的那可就是 null 了，understand?

例：计算员工的总工资、总奖金；

```
SQL> select sum(sal), sum(comm) from emp;

  SUM(SAL)  SUM(COMM)
---------- ----------
     29025       2200
```

注意：总奖金 2200 是几个人的？ 4 个人的是吧！ 如果是所有人的那可就是 null 了，understand?

例：计算员工的最高工资、最低工资，最高奖金、最低奖金；

```
SQL> select max(sal), min(sal), max(comm), min(comm) from emp;

  MAX(SAL)   MIN(SAL)  MAX(COMM)  MIN(COMM)
---------- ---------- ---------- ----------
      5000        800       1400          0
```

注意：看一下奖金的最大值，如果没有忽略 null 值，想一下最大值应该是谁？

例：计算一下总的部门数；

```
SQL> select count(distinct deptno) s1 from emp;

        S1
----------
         3
```

例：计算员工的总人数；

```
SQL> select count(ename), count(*) from emp;

COUNT(ENAME)   COUNT(*)
------------ ----------
          14         14
```

注意:计算总人数的时候，可以按照 ename 进行统计，当然也可以按照 empno 统计；但是，使用 count(*)的时候，也是统计表中的记录数，包括重复和含 null 值的行；

特例：猜一下下面的结果？

```
SQL> select count(empno) s1, count(*) s2, count(1234) s3, count('haha') s4,
  2         count(comm) s5, count(COMM) s6, count('comm') s7, count('COMM') s8
  3  from emp;

        S1         S2         S3         S4         S5         S6         S7         S8
---------- ---------- ---------- ---------- ---------- ---------- ---------- ----------
        14         14         14         14          4          4         14         14
```

注意：按照 s5 或者 s6 计算记录数时，因为它是表中单独的字段，所以会忽略其中的空值；但是，再看看 s7 和 s8 的结果，理解么？不理解的话，看看 s3 和 s4 的结果，是不是有点明白！

如果以上都理解的话，想想这句话的结果？

```
SQL> select count("comm") s9, count("COMM") s10 from emp;
```

好了，不多唠叨了，自己动手试验一下吧！

言归正传，若按照总人数计算平均奖金，呵呵，咋办？(也就是说，我想让 14 个员工的总奖金平均分配到 14 个员工)

```
SQL> select count(nvl(comm,0)) s1, sum(nvl(comm,0)) s2, avg(nvl(comm,0)) s3
  2  from emp;

        S1         S2         S3
---------- ---------- ----------
        14       2200 157.142857
```

注意：此处使用的是 nvl(comm，0)将 null 转换成了 0；(不一定转换成 0，可根据业务要求适当调整)

step 3：总结；

1.组函数使用时忽略 null 值，count(*)除外；

2.为防止 null 值影响业务，可用 nvl 等函数进行转换；

3.再想想上面的那个特例吧！

2.创建数据组(group by)

提示：上面讲的都是将数据作为一个数据集，求最大、最小值，总、平均值等；

那么，是否可以将一个数据集按照某些信息进行再分组，然后求各小组的统计值呢？ 呵呵，当然可以！

例：计算一下各部门的总工资、总人数、平均工资；

```
SQL> show user
USER is "SCOTT"
SQL> select deptno,sum(sal),count(*),avg(sal)
  2  from emp
  3  group by deptno;

    DEPTNO   SUM(SAL)   COUNT(*)   AVG(SAL)
---------- ---------- ---------- ----------
        30       9400          6 1566.66667
        20      10875          5       2175
        10       8750          3 2916.66667
```

注意：1.select 之后，from 之前出现的字段，要么是被用于组函数，要么是被用于分组(即用在 group by 后面，不过也可以在 select 后不出现)；

2.group by 子句用于 where 条件之后，即可以在分组之前将不合条件的数据过滤掉；

3.group by 在 where 之后，select 之前被编译，所以不能在 group by 中使用列的别名；

4.Oracle 9i 中，分组之后的结果(默认)会按照用于分组的字段进行升序排列；但是 Oracle 10g 中不会进行此排序，需要特别指定才可以；

2.1 也可以在 group by 之后使用多列进行分组；

例：计算一下各部门、各工作岗位的总工资、总人数、平均工资；

```
SQL> select deptno,job,sum(sal),count(*),avg(sal)
  2  from emp
  3  group by deptno,job
  4  order by deptno;

    DEPTNO JOB         SUM(SAL)   COUNT(*)   AVG(SAL)
---------- --------- ---------- ---------- ----------
        10 CLERK           1300          1       1300
        10 MANAGER         2450          1       2450
        10 PRESIDENT       5000          1       5000
        20 ANALYST         6000          2       3000
        20 CLERK           1900          2        950
        20 MANAGER         2975          1       2975
        30 CLERK            950          1        950
        30 MANAGER         2850          1       2850
        30 SALESMAN        5600          4       1400

9 rows selected.
```

注意：观察 20 部门的 ANALTST、CLERK 以及 30 部门的 SALESMAN；

其中，数据先按照 deptno 进行分组，然后再按照 job 进行分组，然后进行统计；

2.2 where 子句和 having 子句的使用；

提示：可以使用 where 子句对分组前的数据进行筛选；

可以使用 having 子句对分组后的数据进行筛选；

注意：组函数不能使用在 where 子句中，但可以使用在 having 子句中；

例：计算部门 10、20 员工的总工资、总人数、平均工资；（分组前用 where 子句筛选数据）

```
SQL> select deptno,sum(sal),count(*),avg(sal)
  2  from emp
  3  where deptno in (10,20)
  4  group by deptno
  5  order by deptno;

    DEPTNO   SUM(SAL)   COUNT(*)   AVG(SAL)
---------- ---------- ---------- ----------
        10       8750          3 2916.66667
        20      10875          5       2175
```

例：计算哪个部门的平均工资小于 2000，并查看总工资、总人数；（分组后用 having 子句筛选数据）

```
SQL> select deptno,sum(sal),count(*),avg(sal)
  2  from emp
  3  group by deptno
  4  having avg(sal) < 2500
  5  order by deptno;

    DEPTNO   SUM(SAL)   COUNT(*)   AVG(SAL)
---------- ---------- ---------- ----------
        20      10875          5       2175
        30       9400          6 1566.66667
```

注意：having 子句出现在 group by 之后；

若在 where 子句中进行条件限定，则报错！（因为 where 子句中不能使用组函数）

```
SQL> select deptno,sum(sal),count(*),avg(sal)
  2  from emp
  3  where avg(sal) < 2500
  4  having avg(sal) < 2500
  5  order by deptno;
where avg(sal) < 2500
      *
ERROR at line 3:
ORA-00934: group function is not allowed here
```

总结：SQL 语句的编译顺序；(from →where→group by→having→select→order by)

3. 函数嵌套

例：求平均工资最大的那个部门的名字是什么？（呵呵，想一下如何实现！）

4. 特例

注意：having 子句可以写在 group by 之后，也可以写在 group by 之前；（工作中一般写在 group by 之后，这样便于理解；）

先看下面的 SQL 语句的写法，对还是错？

```
SQL> select sum(sal),count(*),avg(sal)
  2  from emp
  3  having avg(sal) < 2500;
```

提示：先想一下，再看结果！想一下为什么结果会是这样！

```
SQL> select sum(sal),count(*),avg(sal)
  2  from emp
  3  having avg(sal) < 2500;

  SUM(SAL)   COUNT(*)   AVG(SAL)
---------- ---------- ----------
     29025         14 2073.21429
```

原因：在没有 group by 的情况下，select 后面只有被用于组函数的字段；
而没有 group by 的情况下，会将全部数据作为一组进行统计，然后通过 having 进行筛选；

如果把其中的 2500 改成 2000，想一下，结果会是怎样？

```
SQL> select sum(sal),count(*),avg(sal)
  2  from emp
  3  having avg(sal) < 2000;

no rows selected
```

【第六章 子查询】

1. select 子句进行逻辑嵌套；（简化了逻辑运算的步骤）

注意：子查询可以应用在 from、where、having 子句中；

例：在 emp 表中，谁的工资比 CLARK 的多？

【method 1】-----分步查询-----

step 1：查询 emp 表的信息；

```
SQL> select * from emp;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10

14 rows selected.
```

step 2：查询 CLARK 的工资；

```
SQL> select empno,ename,sal from emp
  2  where ename = 'CLARK';

     EMPNO ENAME             SAL
---------- ---------- ----------
      7782 CLARK            2450
```

step 3：查询谁的工资比这个工资多；

```
SQL> select * from emp
  2  where sal > 2450;

     EMPNO ENAME      JOB              MGR HIREDATE          SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
```

【method 2】------嵌套查询------

```
SQL> select * from emp
  2  where sal > (select sal from emp where ename = 'CLARK');

     EMPNO ENAME      JOB              MGR HIREDATE          SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
```

注意：在圆括号中的查询得到一个唯一的结果，并将它作为外层查询的一个参数；
对，圆括号中的那个查询就是子查询；

小结：以上子查询为单行子查询，即得到的内层结果只有唯一一个；

如下情况，也属于单行子查询：
例：查询和 CLARK 在同一部门、并且 sal 大于 CLARK 的员工的信息；
【method 1】-------分步查询------
step 1：查询 CLARK 的部门号和工资；

```
SQL> select empno,ename,job,sal,deptno
  2  from emp
  3  where ename = 'CLARK';

     EMPNO ENAME      JOB              SAL     DEPTNO
---------- ---------- --------- ---------- ----------
      7782 CLARK      MANAGER         2450         10
```

step 2：通过复合条件查询

```
SQL> select * from emp
  2  where deptno = 10 and sal > 2450;

     EMPNO ENAME      JOB              MGR HIREDATE          SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
```

【method 2】-------嵌套查询------
step 1：通过子查询实现；

```
SQL> select * from emp
  2  where deptno = (select deptno from emp where ename = 'CLARK')
  3  and sal > (select sal from emp where ename = 'CLARK');

     EMPNO ENAME      JOB              MGR HIREDATE          SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
```

注意：上面在取得 CLARK 的 deptno 和 sal 的时候，使用的为单行子查询；

当然，你也可以试一下：查询和 SCOTT 在同一部门、并且 sal 大于 CLARK 的员工的信息？

2.继续看一下单行子查询的例子；

2.1 在 emp 表中，那个员工的工资最少？

```
SQL> select * from emp
  2  where sal = (select min(sal) from emp);

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
```

注意：不用考虑并列最小的问题，因为最小值就一种，得到这个结果就行；

2.2 在 emp 表中谁和 haha 在同一部门？（真逗，木有人啊！）

```
SQL> select * from emp
  2  where deptno = (select deptno from emp where ename = 'haha');

no rows selected

SQL> select deptno from emp where ename = 'haha';

no rows selected
```

注意：当子查询没有得到结果时，其返回值为 null，外层的查询中的 where 条件 *deptno = null*，则 where 不成立，则没有数据返回；(如果 deptno 存在 null 值，那么在比较两个 null 值结果依然是 null，所以还是没有数据返回)

简单的说就是，内 0 外 0；

2.3 子查询用在 from 中；（子查询得到临时结果集）

例：查询位于 SALES 部门的员工的信息、个数；

【method 1】

```
SQL> select * from emp
  2  where deptno = (select deptno from dept where dname = 'SALES');

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30

6 rows selected.

SQL> select count(*) from emp
  2  where deptno = (select deptno from dept where dname = 'SALES');

  COUNT(*)
----------
         6
```

【method 2】

```
SQL> select e.empno, e.ename, d.dname
  2  from emp e, dept d
  3  where e.deptno = d.deptno;

     EMPNO ENAME      DNAME
---------- ---------- --------------
      7369 SMITH      RESEARCH
      7499 ALLEN      SALES
      7521 WARD       SALES
      7566 JONES      RESEARCH
      7654 MARTIN     SALES
      7698 BLAKE      SALES
      7782 CLARK      ACCOUNTING
      7788 SCOTT      RESEARCH
      7839 KING       ACCOUNTING
      7844 TURNER     SALES
      7876 ADAMS      RESEARCH
      7900 JAMES      SALES
      7902 FORD       RESEARCH
      7934 MILLER     ACCOUNTING

14 rows selected.

SQL> select count(*)
  2  from (select e.empno, e.ename, d.dname
  3        from emp e, dept d
  4        where e.deptno = d.deptno)
  5  where dname = 'SALES';

  COUNT(*)
----------
         6
```

当然，实现的方法有很多，自己也可以实现一把，呵呵！（以下供参考）

```
SQL> select e.empno, e.ename, d.dname
  2  from emp e, dept d
  3  where e.deptno = d.deptno
  4  and d.dname = 'SALES';

     EMPNO ENAME      DNAME
---------- ---------- --------------
      7499 ALLEN      SALES
      7521 WARD       SALES
      7654 MARTIN     SALES
      7698 BLAKE      SALES
      7844 TURNER     SALES
      7900 JAMES      SALES

6 rows selected.
```

2.4 子查询用在 having 子句中；

例：查询 emp 表中，哪个部门的最低工资比部门 10 的最低工资还低；

step 1：查询 emp 表中各部门的最低工资情况；

```
SQL> select deptno,min(sal)
  2  from emp
  3  group by deptno;

    DEPTNO   MIN(SAL)
---------- ----------
        30        950
        20        800
        10       1300
```

step 2：查询比部门 10 的最低工资还低的部门信息；

```
SQL> select deptno,min(sal)
  2  from emp
  3  group by deptno
  4  having min(sal) < (select min(sal) from emp where deptno = 10);

    DEPTNO   MIN(SAL)
---------- ----------
        30        950
        20        800
```

2.5 再练习：在 emp 表中哪个部门的平均工资最高？

注意：不能像下面这样写!!

```
SQL> select deptno,avg(sal)
  2  from emp
  3  group by deptno;

    DEPTNO   AVG(SAL)
---------- ----------
        30 1566.66667
        20       2175
        10 2916.66667
```

如果你回答：部门 10 的平均工资最高，为 2916.6667；但是，得到这个结果的时候，你参与计算了么？

所以，不能像上面那样写！

为什么不能？因为 emp 表中的数据量较少，可以看出结果；但是，如果数据量很大，分组较多的情况下，是根本不能看出来的…

还是，按部就班，循规蹈矩的来做吧…

首先，确认一下问题：在 emp 表中哪个部门的平均工资最高？

step 1：查询最高的平均工资是多少？

```
SQL> select max(avg(sal)) from emp
  2  group by deptno;

MAX(AVG(SAL))
-------------
   2916.66667
```

step 2：查询哪个部门的平均工资等于最高的平均工资；

```
SQL> select deptno, avg(sal)
  2  from emp
  3  group by deptno
  4  having avg(sal) = (select max(avg(sal)) from emp group by deptno);

    DEPTNO   AVG(SAL)
---------- ----------
        10 2916.66667
```

step 3：查询部门 10 所对应的名字；

```
SQL> select dname from dept
  2  where deptno = (select deptno from emp
  3                  group by deptno
  4                  having avg(sal) = (select max(avg(sal)) from emp group by deptno));

DNAME
--------------
ACCOUNTING
```

3.多行子查询(即子查询返回的值有多个)

试一下：

```
SQL> select deptno,ename
  2  from emp
  3  where sal = (select min(sal) from emp group by deptno);
where sal = (select min(sal) from emp group by deptno)
            *
ERROR at line 3:
ORA-01427: single-row subquery returns more than one row
```

注意：自查询中的返回值有多个，无法与外层条件一一匹配，所以报错！

解决：对于多行子查询，需应用多行运算符(in、any、all)

解释：in（多行子查询）:表示外层条件属于集合中的任意成员；

any（多行子查询）：表示外层条件只需满足子查询的任意一个值就行；

all（多行子查询）：表示外层条件必须需满足子查询的全部的值才行；

3.1 在 emp 表中，每个部门工资最少的那个人的 empno、ename1？-----in-----

step 1：先查询每个部门的最少工资是多少；

```
SQL> select deptno,min(sal) from emp
  2  group by deptno;

    DEPTNO   MIN(SAL)
---------- ----------
        30        950
        20        800
        10       1300
```

step 2：由此，查询每个部门工资最少的那个人的 deptno、empno、ename1、sal 等？

```
SQL> select deptno,empno,ename,sal
  2  from emp
  3  where sal in (select min(sal) from emp group by deptno);

    DEPTNO      EMPNO ENAME             SAL
---------- ---------- ---------- ----------
        20       7369 SMITH             800
        30       7900 JAMES             950
        10       7934 MILLER           1300
```

3.2 在 emp 表中，查询：比某一个部门的最高工资还多的员工的信息；-----any-----

分析：某一个部门：至于哪个部门，不确定，有就行

step 1：查询各部门的最高工资；

```
SQL> select deptno,max(sal) from emp
  2  group by deptno;

    DEPTNO   MAX(SAL)
---------- ----------
        30       2850
        20       3000
        10       5000
```

step 2：

```
SQL> select deptno,empno,ename,sal from emp
  2  where sal > any(select max(sal) from emp group by deptno);

    DEPTNO      EMPNO ENAME             SAL
---------- ---------- ---------- ----------
        20       7566 JONES            2975
        20       7788 SCOTT            3000
        10       7839 KING             5000
        20       7902 FORD             3000
```

注意：以上这些人的工资比某一部门的最高工资还要多；

重点：只要这些人的工资，比各部门最高工资中最少的那个多就行了；　所以，比 2850 多就行了！

3.3 在 emp 表中，查询：工资比 SALESMAN 的工资还少，并且工作岗位不是 SALESMAN 的员工的信息；

-----all-----

step 1：查看 SALESMAN 的工资；

```
SQL> select * from emp
  2  where job = 'SALESMAN';

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
```

step 2：查询相关结果；

```
SQL> select deptno,job,empno,ename,sal from emp
  2  where sal < all(select sal from emp where job = 'SALESMAN')
  3  and job <> 'SALESMAN';

    DEPTNO JOB            EMPNO ENAME             SAL
---------- --------- ---------- ---------- ----------
        20 CLERK           7369 SMITH             800
        20 CLERK           7876 ADAMS            1100
        30 CLERK           7900 JAMES             950
```

注意：sal < all()：表示 sal 要小于其中的最小值；
sal > all()：表示 sal 要大于其中的最大值；

【第七章 关于绑变量的使用】

1. 概述

例如，在执行以下语句的时候，默认情况下数据库会将其分别编译执行，因为这是 3 句不一样的命令，由此会浪费很多的资源；

```
select empno, ename, sal from emp where empno = 7369;
select empno, ename, sal from emp where empno = 7499;
select empno, ename, sal from emp where empno = 7521;
```

使用绑定变量，减少相同或相似语句的重复编译，减少不必要的 I/O 操作，就可以节省资源，提高效率；

2. 通过以下方法实现绑定变量的效果；

2.1 在 SQL 代码中使用&符号；

注意：对于例题中的 3 条语句，如下操作只编译一次即可；

```
SQL> show user
USER is "SCOTT"
SQL> select empno,ename,sal from emp where empno = &haha;
```

注意：每次执行这条命令时，都会提示：输入 haha 代表的变量值，即输入 7369 或 7499 或 7521 即可；
但是，当第 2 次再执行此命令时，不会再重新编译，因为前后两次的命令一样；
数据库直接取第 1 次编译后生成的批代码直接运行，取回相应结果即可；

```
SQL> select empno,ename,sal from emp where empno = &haha;
Enter value for haha: 7369
old   1: select empno,ename,sal from emp where empno = &haha
new   1: select empno,ename,sal from emp where empno = 7369

     EMPNO ENAME             SAL
---------- ---------- ----------
      7369 SMITH             800

SQL> /
Enter value for haha: 7499
old   1: select empno,ename,sal from emp where empno = &haha
new   1: select empno,ename,sal from emp where empno = 7499

     EMPNO ENAME             SAL
---------- ---------- ----------
      7499 ALLEN            1600

SQL> /
Enter value for haha: 7521
old   1: select empno,ename,sal from emp where empno = &haha
new   1: select empno,ename,sal from emp where empno = 7521

     EMPNO ENAME             SAL
---------- ---------- ----------
      7521 WARD             1250
```

2.2 使用 define 命令定义变量值；

step 1：查看数据库自带的变量定义；

```
SQL> show user
USER is "SCOTT"
SQL> define
DEFINE _DATE           = "27-MAY-11" (CHAR)
DEFINE _CONNECT_IDENTIFIER = "orcl" (CHAR)
DEFINE _USER           = "SCOTT" (CHAR)
DEFINE _PRIVILEGE      = "" (CHAR)
DEFINE _SQLPLUS_RELEASE = "1002000100" (CHAR)
DEFINE _EDITOR         = "ed" (CHAR)
DEFINE _O_VERSION      = "Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options" (CHAR)
DEFINE _O_RELEASE      = "1002000100" (CHAR)
SQL>
```

step 2：自己定义变量 haha，然后进行查看；

```
SQL> define haha = 2000;
SQL> define haha;
DEFINE HAHA            = "2000" (CHAR)
SQL> define
DEFINE _DATE           = "27-MAY-11" (CHAR)
DEFINE _CONNECT_IDENTIFIER = "orcl" (CHAR)
DEFINE _USER           = "SCOTT" (CHAR)
DEFINE _PRIVILEGE      = "" (CHAR)
DEFINE _SQLPLUS_RELEASE = "1002000100" (CHAR)
DEFINE _EDITOR         = "ed" (CHAR)
DEFINE _O_VERSION      = "Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options" (CHAR)
DEFINE _O_RELEASE      = "1002000100" (CHAR)
DEFINE HAHA            = "2000" (CHAR)
SQL>
```

step 3：使用自定义的变量 haha；（使用&符号引用自定义的变量）

```
SQL> select empno,ename,sal from emp where sal > &haha;
old   1: select empno,ename,sal from emp where sal > &haha
new   1: select empno,ename,sal from emp where sal > 2000

     EMPNO ENAME             SAL
---------- ---------- ----------
      7566 JONES            2975
      7698 BLAKE            2850
      7782 CLARK            2450
      7788 SCOTT            3000
      7839 KING             5000
      7902 FORD             3000
```

注意：使用&haha 时，它会自动用已定义的 *2000* 进行替换；（替换的效果也可以看到）

补充：

step 4：如果想修改自定义的变量值；（重新赋值即可）

```
SQL> define haha;
DEFINE HAHA            = "2000" (CHAR)
SQL> define haha = 1500;
SQL> define
DEFINE _DATE           = "27-MAY-11" (CHAR)
DEFINE _CONNECT_IDENTIFIER = "orcl" (CHAR)
DEFINE _USER           = "SCOTT" (CHAR)
DEFINE _PRIVILEGE      = "" (CHAR)
DEFINE _SQLPLUS_RELEASE = "1002000100" (CHAR)
DEFINE _EDITOR         = "ed" (CHAR)
DEFINE _O_VERSION      = "Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options" (CHAR)
DEFINE _O_RELEASE      = "1002000100" (CHAR)
DEFINE HAHA            = "1500" (CHAR)
```

step 5：如果想删除自定义的变量；（undefine 即可）

```
SQL> define haha;
DEFINE HAHA            = "1500" (CHAR)
SQL> undefine haha;
SQL> define
DEFINE _DATE           = "27-MAY-11" (CHAR)
DEFINE _CONNECT_IDENTIFIER = "orcl" (CHAR)
DEFINE _USER           = "SCOTT" (CHAR)
DEFINE _PRIVILEGE      = "" (CHAR)
DEFINE _SQLPLUS_RELEASE = "1002000100" (CHAR)
DEFINE _EDITOR         = "ed" (CHAR)
DEFINE _O_VERSION      = "Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options" (CHAR)
DEFINE _O_RELEASE      = "1002000100" (CHAR)
```

2.3 在 SQL 代码中使用&&符号；

step 1：查看数据库自带的变量定义；

```
SQL> define
DEFINE _DATE           = "27-MAY-11" (CHAR)
DEFINE _CONNECT_IDENTIFIER = "orcl" (CHAR)
DEFINE _USER           = "SCOTT" (CHAR)
DEFINE _PRIVILEGE      = "" (CHAR)
DEFINE _SQLPLUS_RELEASE = "1002000100" (CHAR)
DEFINE _EDITOR         = "ed" (CHAR)
DEFINE _O_VERSION      = "Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options" (CHAR)
DEFINE _O_RELEASE      = "1002000100" (CHAR)
```

step 2：执行以下 SQl 语句，提示输入时，将变量赋值为 sal；

```
SQL> select empno,ename,&&haha
  2  from emp
  3  where &haha > 2500
  4  order by &haha;
Enter value for haha: sal
old   1: select empno,ename,&&haha
new   1: select empno,ename,sal
old   3: where &haha > 2500
new   3: where sal > 2500
old   4: order by &haha
new   4: order by sal

     EMPNO ENAME              SAL
---------- ---------- ----------
      7698 BLAKE            2850
      7566 JONES            2975
      7902 FORD             3000
      7788 SCOTT            3000
      7839 KING             5000
```

step 3：执行完上述 SQL 命令后，再次查看系统中变量的定义情况；

```
SQL> define
DEFINE _DATE            = "27-MAY-11" (CHAR)
DEFINE _CONNECT_IDENTIFIER = "orcl" (CHAR)
DEFINE _USER            = "SCOTT" (CHAR)
DEFINE _PRIVILEGE       = "" (CHAR)
DEFINE _SQLPLUS_RELEASE = "1002000100" (CHAR)
DEFINE _EDITOR          = "ed" (CHAR)
DEFINE _O_VERSION       = "Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options" (CHAR)
DEFINE _O_RELEASE       = "1002000100" (CHAR)
DEFINE HAHA             = "sal" (CHAR)
```

注意：会发现自动多了一个关于 haha = 'sal'的定义；

原因：在使用&&haha 定义变量时，系统中会自动定义 haha 的值；则之后使用这些变量的时候，需要用&引用才可，例&haha；

step 4：如果想对此变量进行修改或者删除，请参照 define 与 undefine 命令；

2.4 如果在 SQL 代码或者前台代码中没有使用绑定变量，也可以在数据库层面上进行处理；

注意：修改数据库的系统参数，从而实现绑定变量的效果；

但是，这种修改，治标而不治本；（需修改 sql 代码或前台代码才可彻底实现）

step 1：查看此参数；（cursor_sharing）

提示：此参数有 3 种值，exact （SQL 命令完全一样时，不重新编译）；

similar （SQL 命令部分一样，数据库自动实现变量的绑定）；

force （SQL 命令不一样时，强制使用绑定变量，容易出问题）；

默认情况下为 exact，一般可修改为 similar，但不推荐修改为 force；

```
SQL> conn sys/oracle as sysdba
Connected.
SQL> show user
USER is "SYS"
SQL> show parameter cursor_sharing

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
cursor_sharing                       string      EXACT
```

注意：这是在无法修改源代码的情况下，不得已才使用此参数，但是效率的变化也不太明显；

step 2：修改此参数；（cursor_sharing 为动态参数，修改后默认即时生效）

```
SQL> show user
USER is "SYS"
SQL> alter session set cursor_sharing = similar;

Session altered.

SQL> alter system set cursor_sharing = similar;

System altered.
```

注意：alter session 修改后只会影响此会话的效果；

alter system 修改后会影响数据库中所有用户的效果，影响太大，一般不推荐；

【第八章 操纵数据】

1. insert 操作；

1.1 在一张表中添加 1 条新数据；

step 1：创建 1 张表 t10，结构参照 emp 表；

```
SQL> show user
USER is "SCOTT"
SQL> create table t10
  2  as select * from emp where 1=2;

Table created.
```

step 2：查看表结构以及表中的数据；

```
SQL> desc t10
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                              NUMBER(4)
 ENAME                                              VARCHAR2(10)
 JOB                                                VARCHAR2(9)
 MGR                                                NUMBER(4)
 HIREDATE                                           DATE
 SAL                                                NUMBER(7,2)
 COMM                                               NUMBER(7,2)
 DEPTNO                                             NUMBER(2)

SQL> select * from t10;

no rows selected
```

step 3：向表 t10 中插入 1 条新数据；

例 1：

```
SQL> show user
USER is "SCOTT"
SQL> insert into t10
  2  values(1001,'name1','stu',1010,sysdate,2000,100,11);

1 row created.

SQL> commit;

Commit complete.

SQL> set linesize 120
SQL> select * from t10;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      1001 name1      stu              1010 27-MAY-11       2000        100         11
```

注意：如果没有指明给 t10 表中的哪些字段赋值，则表示要给所有的字段依次赋值；
赋值的顺序按照表中字段的定义顺序；

例 2：

```
SQL> insert into t10(empno,ename,sal,deptno)
  2  values(1002,'name2',2200,12);

1 row created.

SQL> commit;

Commit complete.

SQL> select * from t10;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      1001 name1      stu              1010 27-MAY-11       2000        100         11
      1002 name2                                            2200                    12
```

注意：此处已指定给 t10 表中的某些字段进行赋值，那么赋值的顺序就按照给出的顺序依次赋值即可；
赋值过程中，时间和字符串要加单引号；注意类型匹配，不然会发生隐式类型转换；

(1)其他未赋值的字段，如果有 default 值，则显示 default 值；

step 1：修改 t10 的表结构；（修改 ename 的 default 值）

```
SQL> alter table t10 modify(ename default 'haha');

Table altered.
```

step 2：如果向 t10 中 insert 数据的时候，没有给 ename 赋值，则会以默认值’haha’进行赋值；

```
SQL> insert into t10(empno,job,sal,deptno)
  2  values(1111,'str',4000,99);

1 row created.

SQL> commit;

Commit complete.

SQL> select * from t10;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      1111 haha       str                                 4000                    99
```

或者显示以 default 值进行赋值时，也会以默认值’haha’进行赋值；

```
SQL> insert into t10(empno,ename,job,sal,deptno)
  2  values(2222,default,'str',4500,99);

1 row created.

SQL> commit;

Commit complete.

SQL> select * from t10;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      1111 haha       str                                 4000                    99
      2222 haha       str                                 4500                    99
```

(2)其他未赋值的字段，如果没有定义 default 值，那显示 null；当然也可以强制赋值为 null，覆盖其自身的 default 值；（例：以下）

```
SQL> insert into t10
  2  values(1003,'name3',null,null,null,2400,null,13);

1 row created.

SQL> commit;

Commit complete.

SQL> select * from t10;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      1001 name1      stu              1010 27-MAY-11       2000        100         11
      1002 name2                                            2200                    12
      1003 name3                                            2400                    13
```

注意：hiredate 中存放的 sysdate 值数据特殊值，或者通过 to_date(格式)转换后再进行赋值；

例 3：

```
SQL> insert into t10(empno,ename,sal)
  2  values(&haha,'&hehe',&xixi);
Enter value for haha: 1004
Enter value for hehe: name4
Enter value for xixi: 2600
old   2: values(&haha,'&hehe',&xixi)
new   2: values(1004,'name4',2600)

1 row created.

SQL> commit;

Commit complete.
```

注意：使用绑定变量进行赋值，时间和字符串记得用单引号；

1.2 在一张表中插入多条数据；

step 1：查询 t10 中的数据信息；

```
SQL> select * from t10;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      1001 name1      stu              1010 27-MAY-11       2000        100         11
      1002 name2                                            2200                    12
      1003 name3                                            2400                    13
      1004 name4                                            2600
```

step 2：从 emp 表中复制数据，放入 t10 表；

注意：insert 和 select 中，列的个数和数据类型要求一致；字段的名称，可以不一致；

例 1：

```
SQL> insert into t10(empno,ename,job,sal)
  2  select empno,ename,job,sal from emp
  3  where rownum <= 3;

3 rows created.

SQL> commit;

Commit complete.

SQL> select * from t10;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      1001 name1      stu              1010 27-MAY-11       2000        100         11
      1002 name2                                            2200                    12
      1003 name3                                            2400                    13
      1004 name4                                            2600
      7369 SMITH      CLERK                                  800
      7499 ALLEN      SALESMAN                              1600
      7521 WARD       SALESMAN                              1250
```

例 2：

```
SQL> insert into t10
  2  select * from emp;

14 rows created.
```

注意：因为 t10 和 emp 表的结构一样，所以上面的命令表示：将 emp 表中所有的数据插入 t10 表；

当然还有很多复杂的操作，在此不一一列举；

2. update 操作；

2.1 更新一张表中的 1 条数据；

step 1：查看 t10 的数据；

```
SQL> select * from t10;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      1001 name1      stu             1010 27-MAY-11       2000        100         11
      1002 name2                                           2200                    12
      1003 name3                                           2400                    13
      1004 name4                                           2600
```

step 2：更新 1002 的 comm 值为 200；

```
SQL> update t10 set comm = 200
  2  where empno = 1002;

1 row updated.

SQL> commit;

Commit complete.

SQL> select * from t10;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      1001 name1      stu             1010 27-MAY-11       2000        100         11
      1002 name2                                           2200        200         12
      1003 name3                                           2400                    13
      1004 name4                                           2600
```

注意：如果忘记写 where 条件，则 t10 表中所有数据的 comm 都会被更新为 200；

警告：在更新数据时，谨慎查看 where 条件；（通过控制 where 条件，限定更新数据的数量）

```
SQL> update t10 set comm = 200;

4 rows updated.

SQL> commit;

Commit complete.

SQL> select * from t10;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      1001 name1      stu             1010 27-MAY-11       2000        200         11
      1002 name2                                           2200        200         12
      1003 name3                                           2400        200         13
      1004 name4                                           2600        200
```

step 3：如果修改 1 条数据的多个字段值，那么…(字段之间以逗号隔开)

```
SQL> update t10 set job = 'tr', comm = 600
  2  where empno = 1003;

1 row updated.

SQL> commit;

Commit complete.
```

3. delete 操作；

3.1 通过控制 where 条件，限定删除数据的数量；

step 1：查看 t10 的数据信息；

```
SQL> select * from t10;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      1001 name1      stu             1010 27-MAY-11       2000        200         11
      1002 name2                                           2200        200         12
      1003 name3      tr                                   2400        600         13
      1004 name4                                           2600        200
```

step 2：删除编号为 1001 的记录；

例 1：

```
SQL> delete from t10
  2  where empno = 1001;

1 row deleted.

SQL> commit;

Commit complete.

SQL> select * from t10;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      1002 name2                                        2200        200         12
      1003 name3      tr                                2400        600         13
      1004 name4                                        2600        200
```

注意：from 可以省略；

如果没有写 where 条件，t10 中的所有数据将会被删除；(工作中基本不存在 delete 操作)

例 2：如果不小心删错了数据，想找回来，那该怎么办呢？-----不着急，后续的课程逐步介绍！

4.关于 Insert 语句的 with check option 操作；

概述：with check option 是指在 insert 数据的时候将会对数据进行审核，如果不满足条件，则报错；如果满足条件，则成功；

例： step 1：创建测试表 t11，参照 emp 表，但是不要数据；

```
SQL> show user
USER is "SCOTT"
SQL> create table t11
  2  as select * from emp where 1=2;

Table created.
```

step 2：向 t11 表中 insert 数据，insert 语句的写法与之前的稍有区别；

注意：如果添加的数据满足 check 条件，则 insert 没有问题；

```
SQL> insert into (select empno,ename,sal,deptno
  2               from t11
  3               where deptno = 99 with check option)
  4  values(1001,'oca1',2000,99);

1 row created.

SQL> commit;

Commit complete.

SQL> set linesize 120
SQL> select * from t11;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      1001 oca1                                         2000                    99
```

注意：如果添加的数据不满足 check 条件，则 insert 报错(数据不满足 check 条件)；

```
SQL> insert into (select empno,ename,sal,deptno
  2               from t11
  3               where deptno = 99 with check option)
  4  values(1001,'oca1',2000,88);
             from t11
                  *
ERROR at line 2:
ORA-01402: view WITH CHECK OPTION where-clause violation
```

小结：当然也可以将底下的 values 子句换成子查询；

```
SQL> insert into (select empno,ename,sal,deptno
  2               from t11
  3               where deptno = 99 with check option)
  4  select empno,ename,sal,99 from emp
  5  where rownum <= 2;

2 rows created.

SQL> commit;

Commit complete.

SQL> select * from t11;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      1001 oca1                                          2000                    99
      7369 SMITH                                          800                    99
      7499 ALLEN                                         1600                    99
```

注意：此处使显示标识为 99，是想说明：如果抽出来的数据中 deptno 为 99 的话，则 insert 成功；反之，如果是其他的部门号，则报错；

```
SQL> insert into (select empno,ename,sal,deptno
  2               from t11
  3               where deptno = 99 with check option)
  4  select empno,ename,sal,deptno from emp
  5  where deptno = 10;
             from t11
                  *
ERROR at line 2:
ORA-01402: view WITH CHECK OPTION where-clause violation
```

哎，稍等，如果……　如果把 with check option 去掉，想一下，deptno=10 的数据能不能 insert 成功呢？？

如下：

```
SQL> insert into (select empno,ename,sal,deptno
  2               from t11
  3               where deptno = 99)
  4  select empno,ename,sal,deptno from emp
  5  where deptno = 10;
```

到底能不能 insert 成功呢？自己测试一下吧！　呵呵，不过要告诉我出现该结果的原因！

结论：如果不写 with check option，则 where deptno = 99 就形同虚设，相当于不限定条件；

5. merge 操作；（相当于 update 和 insert 操作二合一）

例：总店和分店，将总店的商品信息合并到分店中去；

总店：

	总店商品ID	总店商品名称	总店商品价格
zd	zsp_id	zsp_name	zsp_price
	101	sp001	11.11
	102	sp002	11.12
	103	sp003	11.13
	105	sp005	11.15

分店：

	总店商品ID	总店商品名称	总店商品价格
fd	fsp_id	fsp_name	fsp_price
	101	sp001	15.11
	102	sp002	15.12
	103	sp003	15.13
	104	sp004	15.14

step 1：创建总店表 ad、分店表 fd，对应的字段参照定义；

```
SQL> show user
USER is "SCOTT"
SQL> create table zd(zsp_id number(3), zsp_name varchar2(20), zsp_price number(5,2));

Table created.

SQL> create table fd(fsp_id number(3), fsp_name varchar2(20), fsp_price number(5,2));

Table created.
```

step 2：添加测试数据，insert 过程(略)；

```
SQL> select * from zd;

    ZSP_ID ZSP_NAME              ZSP_PRICE
---------- -------------------- ----------
       101 sp001                     11.11
       102 sp002                     11.12
       103 sp003                     11.13
       105 sp005                     11.15

SQL> select * from fd;

    FSP_ID FSP_NAME              FSP_PRICE
---------- -------------------- ----------
       101 sp001                     15.11
       102 sp002                     15.12
       103 sp003                     15.13
       104 sp004                     15.14
```

step 3：书写 merge 语句，按照以下要求更新分店中的数据信息；

要求：1.总店和分店的商品信息通过商品的 ID 进行关联；

2.总店有，分店也有的商品，按照总店商品的价格去更新分店商品的价格；

总店有，分店没有的商品，将总店商品的信息添加到分店中去；

总店没有，分店有的商品，不作处理；

```
SQL> merge into fd f
  2   using zd z
  3   on (f.fsp_id = z.zsp_id)
  4  when matched then
  5   update set f.fsp_price = z.zsp_price
  6  when not matched then
  7   insert values(z.zsp_id,z.zsp_name,z.zsp_price);

4 rows merged.

SQL> commit;

Commit complete.
```

注意：想一下，这个 4 rows merged 中的 4 是怎么来的？

step 4：查看分店 merge 后的结果；

```
SQL> select * from fd;

    FSP_ID FSP_NAME              FSP_PRICE
---------- -------------------- ----------
       101 sp001                     11.11
       102 sp002                     11.12
       103 sp003                     11.13
       104 sp004                     15.14
       105 sp005                     11.15
```

注意：考虑一下 fd 中各条数据的变动情况是否符合要求；

【第九章 创建和管理表】

1.建表的过程；

概述：一个用户要想能够建表，则需要满足一系列的条件；

条件 1：有一个用户；(此处创建一个新的用户)[此内容第 13 章讲到]

条件 2：用户要有连接数据库的权限；[此内容第 13 章讲到]

条件 3：用户要有建表的权限；[此内容第 13 章讲到]

条件 4：用户要有一块存储区域，用于存储表中的数据；[此内容管理 I 的第 5 章讲到]

那么，咱现在提前学习一下，呵呵，一步一步来吧！

step 1：由管理员 sys 创建一个用户 user001，密码 user001；

```
SQL> conn sys/oracle as sysdba
Connected.
SQL> show user
USER is "SYS"
SQL> create user user001 identified by user001;

User created.
```

注意：此时 user001 连接不上数据库，因为没有连接数据库的权限；

```
SQL> conn user001/user001
ERROR:
ORA-01045: user USER001 lacks CREATE SESSION privilege; logon denied


Warning: You are no longer connected to ORACLE.
```

step 2：由管理员 sys 授予用户 user001 连接数据库的权限(create session)；

```
SQL> conn / as sysdba
Connected.
SQL> show user
USER is "SYS"
SQL> grant create session to user001;

Grant succeeded.

SQL> conn user001/user001
Connected.
SQL> show user
USER is "USER001"
```

注意：或者，授予用户 user001 角色 connect 也行，因为此角色中也包含 create session 权限；

注意：用户 user001 此时还无法建表，因为没有建表(create table)的权限；

```
SQL> show user
USER is "USER001"
SQL> create table t12(id number(2), name varchar2(20));
create table t12(id number(2), name varchar2(20))
*
ERROR at line 1:
ORA-01031: insufficient privileges
```

step 3：由管理员 sys 授予用户 user001 建表的权限(create table)；

```
SQL> conn / as sysdba
Connected.
SQL> show user
USER is "SYS"
SQL> grant create table to user001;

Grant succeeded.
```

注意：由于没有存储区域，虽然能建表，但是无处存放；

```
SQL> conn user001/user001
Connected.
SQL> show user
USER is "USER001"
SQL> create table t12(id number(2), name varchar2(20));
create table t12(id number(2), name varchar2(20))
*
ERROR at line 1:
ORA-01950: no privileges on tablespace 'USERS'
```

step 4：由管理员 sys 分配一块存储区域；(指定在表空间 users 上可以使用 5MB 的空间)

```
SQL> conn / as sysdba
Connected.
SQL> show user
USER is "SYS"
SQL> alter user user001 quota 5m on users;

User altered.
```

step 5：此时，用户 user001 可以建表；

```
SQL> conn user001/user001
Connected.
SQL> show user
USER is "USER001"
SQL> create table t12(id number(2), name varchar2(20));

Table created.

SQL> desc t12
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------
 ID                                                             NUMBER(2)
 NAME                                                           VARCHAR2(20)
```

注意：更多权限相关的知识，以及表空间的知识，后续的课程中继续讲解；

2.访问其他用户的表；(此内容第 13 章讲到)

概述：如果当前用户访问自己的表，没有问题；

但是，如果用想访问其他用户的表，需要有 select 权限；

并且在访问其他用户的表时，需要加上那个用户的名字；(表示表的所有者)

例：用户 user001 想访问用户 scott 下的 dept 表；

step 1：用户 user001 直接访问 scott 下的 dept 表，肯定不行，报错！

```
SQL> show user
USER is "USER001"
SQL> select * from scott.dept;
select * from scott.dept
                    *
ERROR at line 1:
ORA-00942: table or view does not exist
```

step 2：用户 scott 授权给用户 user001 访问 dept 表的权限；

```
SQL> conn scott/tiger
Connected.
SQL> grant select on dept to user001;

Grant succeeded.
```

step 3：用户 user001 即可以进行访问；

```
SQL> conn user001/user001
Connected.
SQL> show user
USER is "USER001"
SQL> select * from scott.dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

注意：当然管理员 sys 也可以将 scott 用户下的 dept 表的访问权限给 user001，但是一般不这么做！

注意：关于授权和收权，第 13 章详细讲解；

3.时间类型 date 和 timestamp(第 16 章详细讲解)

概述：

Date：时间类型，表示时间的年、月、日，没有长度和精度，取值范围公元前 4713 年 12 月 31 日~公元后 9999 年 12 月 31 日；

Timestamp：时间戳类型，表示时间的年、月、日、小时、分钟、秒、毫秒[上午下午、时区]，没有长度，但有精度；而精度确定的是毫秒，长度为 0~9 之间，默认为 6，可以不标识；

例：

step 1：创建表 t13，字段类型如下；

```
SQL> show user
USER is "SCOTT"
SQL> create table t13(t1 date, t2 timestamp);

Table created.

SQL> desc t13
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 T1                                                 DATE
 T2                                                 TIMESTAMP(6)
```

step 2：查询系统时间、系统详细时间；

```
SQL> select sysdate, systimestamp from dual;

SYSDATE   SYSTIMESTAMP
--------- ---------------------------------------------------------------------------
28-MAY-11 28-MAY-11 01.18.38.272183 AM +08:00
```

step 3：修改时间的显示格式；

```
SQL> alter session set nls_date_format = 'yyyy-mm-dd';

Session altered.

SQL> alter session set nls_timestamp_format = 'yyyy-mm-dd hh24:mi:ss';

Session altered.

SQL> select sysdate, systimestamp from dual;

SYSDATE    SYSTIMESTAMP
---------- ---------------------------------------------------------------------------
2011-05-28 2011-05-28 01:24:35
```

注意：用 alter session 命令；

step 3：将上述时间插入 t13 中相应字段，并查询；

```
SQL> insert into t13
  2  values(sysdate, systimestamp);

1 row created.

SQL> commit;

Commit complete.

SQL> select * from t13;

T1         T2
---------- ---------------------------------------------------------------------------
2011-05-28 2011-05-28 01:25:26
```

4.用子查询的方式建表；

例：创建一张表 t14，定义参照 dept 表；

```
SQL> show user
USER is "SCOTT"
SQL> create table t14
  2  as select * from dept where 1=2;

Table created.

SQL> desc t14
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------
 DEPTNO                                                         NUMBER(2)
 DNAME                                                          VARCHAR2(14)
 LOC                                                            VARCHAR2(13)

SQL> select * from t14;

no rows selected
```

注意：建表时的 where 1=2 表示：新表 t14 中不包含 dept 表中的数据；　　若不写 where 条件，表示在参照 dept 表定义的时候，数据一并拷贝过去；

新表 t14 的字段名称、类型、长度，和 dept 表完全相同；

例：创建表 t15，定义参照 dept，但是 t15 的字段名由自己定义；(数据暂时不要)

【method 1】

```
SQL> show user
USER is "SCOTT"
SQL> create table t15(haha, hehe, xixi)
  2  as select * from dept where 1=2;

Table created.

SQL> desc t15
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------
 HAHA                                                           NUMBER(2)
 HEHE                                                           VARCHAR2(14)
 XIXI                                                           VARCHAR2(13)
```

注意：t15 字段名由自己定义，但是类型和长度依然参照 dept 表；

【method 2】

```
SQL> show user
USER is "SCOTT"
SQL> drop table t15;

Table dropped.

SQL> create table t15
  2  as select deptno haha, dname hehe, loc xixi
  3     from dept where 1=2;

Table created.

SQL> desc t15
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------
 HAHA                                                           NUMBER(2)
 HEHE                                                           VARCHAR2(14)
 XIXI                                                           VARCHAR2(13)
```

注意：在子查询中定义列的别名，也可以改变 t15 的字段名；

5.将用户 scott 进行初始化；

概述：即，将 scott 用户的信息恢复到初始的状态；

step 1：由管理员 sys 删除 scott 用户下所有与其相关的东西；

```
SQL> show user
USER is "SYS"
SQL> drop user scott cascade;
drop user scott cascade
*
ERROR at line 1:
ORA-01940: cannot drop a user that is currently connected
```

注意：如果提示无法删除处于连接状态的用户 scott，按照以下步骤继续；

step 1.1：在 sys 用户下确认 scott 的会话信息；

```
SQL> show user
USER is "SYS"
SQL> select sid, serial#, username from v$session
  2  where username = 'SCOTT';

       SID    SERIAL# USERNAME
---------- ---------- ------------------------------
       147        199 SCOTT
       150        154 SCOTT
       159        594 SCOTT
```

step 1.2：将用户 scott 的会话进行关闭；

```
SQL> alter system kill session '147,199' immediate;

System altered.

SQL> alter system kill session '150,154' immediate;

System altered.

SQL> alter system kill session '159,594' immediate;

System altered.
```

注意：可能有点延迟，需要等待会…

注意：确认一下用户 scott 的会话被终止完毕；

```
SQL> select sid, serial#, username
  2  from v$session
  3  where username = 'SCOTT';

no rows selected
```

step 1.3：删除用户 scott 的所有信息；

```
SQL> show user
USER is "SYS"
SQL> drop user scott cascade;

User dropped.
```

step 2：在操作系统下查找用于初始化用户 scott 的脚本；

路径：

```
[oracle@rhel ~]$ cd $ORACLE_HOME/rdbms/admin
[oracle@rhel admin]$
```

注意：大小写敏感；

注意：用 ls 查看目录下的文件，脚本名称：utlsampl.sql

step 3：在 sys 用户下运行此脚本；

```
SQL> show user
USER is "SYS"
SQL> @$ORACLE_HOME/rdbms/admin/utlsampl.sql
Disconnected from Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
[oracle@rhel ~]$
```

注意：别忘了@符号；

step 4：切换到用户 scott，查看其初始信息；

```
[oracle@rhel ~]$ sqlplus scott/tiger

SQL*Plus: Release 10.2.0.1.0 - Production on Sat May 28 02:03:09 2011

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options

SQL> show user
USER is "SCOTT"
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
DEPT                           TABLE
EMP                            TABLE
BONUS                          TABLE
SALGRADE                       TABLE
```

6.修改表结构；

例：创建测试用表 temp，定义参照 emp 表，数据不要；

```
SQL> show user
USER is "SCOTT"
SQL> create table temp
  2  as select * from emp where 1=2;

Table created.

SQL> desc temp
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                              NUMBER(4)
 ENAME                                              VARCHAR2(10)
 JOB                                                VARCHAR2(9)
 MGR                                                NUMBER(4)
 HIREDATE                                           DATE
 SAL                                                NUMBER(7,2)
 COMM                                               NUMBER(7,2)
 DEPTNO                                             NUMBER(2)
```

6.1 添加新的一列；

```
SQL> alter table temp add(haha number(10));

Table altered.

SQL> desc temp;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                              NUMBER(4)
 ENAME                                              VARCHAR2(10)
 JOB                                                VARCHAR2(9)
 MGR                                                NUMBER(4)
 HIREDATE                                           DATE
 SAL                                                NUMBER(7,2)
 COMM                                               NUMBER(7,2)
 DEPTNO                                             NUMBER(2)
 HAHA                                               NUMBER(10)
```

如果一次增加多列呢？

```
SQL> alter table temp add(hehe varchar2(10), xixi date);

Table altered.

SQL> desc temp;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                              NUMBER(4)
 ENAME                                              VARCHAR2(10)
 JOB                                                VARCHAR2(9)
 MGR                                                NUMBER(4)
 HIREDATE                                           DATE
 SAL                                                NUMBER(7,2)
 COMM                                               NUMBER(7,2)
 DEPTNO                                             NUMBER(2)
 HAHA                                               NUMBER(10)
 HEHE                                               VARCHAR2(10)
 XIXI                                               DATE
```

注意：新增加的列只会出现在表的末尾，不能在指定的某些字段前后增加新列；

6.2 修改已存在的列；

```
SQL> alter table temp modify(haha varchar2(10));

Table altered.

SQL> desc temp;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                              NUMBER(4)
 ENAME                                              VARCHAR2(10)
 JOB                                                VARCHAR2(9)
 MGR                                                NUMBER(4)
 HIREDATE                                           DATE
 SAL                                                NUMBER(7,2)
 COMM                                               NUMBER(7,2)
 DEPTNO                                             NUMBER(2)
 HAHA                                               VARCHAR2(10)
 HEHE                                               VARCHAR2(10)
 XIXI                                               DATE
```

如果一次修改多个列呢？

```
SQL> alter table temp modify(hehe number(10), xixi varchar2(10));

Table altered.

SQL> desc temp;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                              NUMBER(4)
 ENAME                                              VARCHAR2(10)
 JOB                                                VARCHAR2(9)
 MGR                                                NUMBER(4)
 HIREDATE                                           DATE
 SAL                                                NUMBER(7,2)
 COMM                                               NUMBER(7,2)
 DEPTNO                                             NUMBER(2)
 HAHA                                               VARCHAR2(10)
 HEHE                                               NUMBER(10)
 XIXI                                               VARCHAR2(10)
```

注意：在修改列的类型和长度的时候，表中应没有数据，否则，可能报错(类型无法转换，或不能缩短等)；

或者，表中确实已经存在数据，而又想改变数据的类型或者长度，那怎么办？

提示：如果类型可以转换，可能没有问题；否则，报错！

如果数据长度没有满位，可能可以缩短，但不保证一定；

例：

temp 1：新建测试表 td，定义参照 dept，数据保留；

```
SQL> create table td as select * from dept;

Table created.

SQL> desc td;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DEPTNO                                             NUMBER(2)
 DNAME                                              VARCHAR2(14)
 LOC                                                VARCHAR2(13)

SQL> select * from td;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

temp 2：修改 deptno 的长度或者类型；

```
SQL> alter table td modify(deptno number(4));

Table altered.

SQL> desc td;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DEPTNO                                             NUMBER(4)
 DNAME                                              VARCHAR2(14)
 LOC                                                VARCHAR2(13)

SQL> alter table td modify(deptno varchar2(4));
alter table td modify(deptno varchar2(4))
                      *
ERROR at line 1:
ORA-01439: column to be modified must be empty to change datatype
```

temp 3：类型修改为 varchar2 时报错，但一定要修改，怎么办？

提示：可以新加一列，类型为 varchar2，长度为 4；

```
SQL> alter table td add (haha varchar2(4));

Table altered.

SQL> desc td;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DEPTNO                                             NUMBER(4)
 DNAME                                              VARCHAR2(14)
 LOC                                                VARCHAR2(13)
 HAHA                                               VARCHAR2(4)
```

temp 4：将 deptno 中的数据关联更新到 haha 列中；

```
SQL> update td set haha = to_char(deptno);

4 rows updated.

SQL> commit;

Commit complete.

SQL> select * from td;

    DEPTNO DNAME          LOC           HAHA
---------- -------------- ------------- ----
        10 ACCOUNTING     NEW YORK      10
        20 RESEARCH       DALLAS        20
        30 SALES          CHICAGO       30
        40 OPERATIONS     BOSTON        40
```

temp 5：将 deptno 列中的数据清空；

```
SQL> update td set deptno = null;

4 rows updated.

SQL> commit;

Commit complete.

SQL> select * from td;

    DEPTNO DNAME          LOC           HAHA
---------- -------------- ------------- ----
           ACCOUNTING     NEW YORK      10
           RESEARCH       DALLAS        20
           SALES          CHICAGO       30
           OPERATIONS     BOSTON        40
```

temp 6：修改 deptno 的类型或者长度；

```
SQL> alter table td modify(deptno varchar2(4));

Table altered.

SQL> desc td;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DEPTNO                                             VARCHAR2(4)
 DNAME                                              VARCHAR2(14)
 LOC                                                VARCHAR2(13)
 HAHA                                               VARCHAR2(4)
```

temp 7：将 deptno 中的数据关联更新回来；

```
SQL> update td set deptno = haha;

4 rows updated.

SQL> commit;

Commit complete.

SQL> select * from td;

DEPT DNAME          LOC           HAHA
---- -------------- ------------- ----
10   ACCOUNTING     NEW YORK      10
20   RESEARCH       DALLAS        20
30   SALES          CHICAGO       30
40   OPERATIONS     BOSTON        40
```

temp 8：删除 haha 列；

```
SQL> alter table td drop(haha);

Table altered.

SQL> desc td;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DEPTNO                                             VARCHAR2(4)
 DNAME                                              VARCHAR2(14)
 LOC                                                VARCHAR2(13)

SQL> select * from td;

DEPT DNAME          LOC
---- -------------- -------------
10   ACCOUNTING     NEW YORK
20   RESEARCH       DALLAS
30   SALES          CHICAGO
40   OPERATIONS     BOSTON
```

注意：在维护表结构的时候，注意选择业务不忙的时候，否则影响较大；

6.3 删除已存在的列；

例：

step 1：创建测试表 te，定义参照 emp，数据保留；

```
SQL> show user
USER is "SCOTT"
SQL> create table te as select * from emp;

Table created.

SQL> desc te;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                              NUMBER(4)
 ENAME                                              VARCHAR2(10)
 JOB                                                VARCHAR2(9)
 MGR                                                NUMBER(4)
 HIREDATE                                           DATE
 SAL                                                NUMBER(7,2)
 COMM                                               NUMBER(7,2)
 DEPTNO                                             NUMBER(2)
```

step 2：删除一列，查看修改后的信息；

```
SQL> alter table te drop(deptno);

Table altered.

SQL> alter table te drop column comm;

Table altered.

SQL> desc te;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                              NUMBER(4)
 ENAME                                              VARCHAR2(10)
 JOB                                                VARCHAR2(9)
 MGR                                                NUMBER(4)
 HIREDATE                                           DATE
 SAL                                                NUMBER(7,2)
```

注意：以上两个 alter 命令都可以一次删除一列；如果想一次删除多列，需用前者；

step 3：删除多列，查看修改后的信息；

```
SQL> alter table te drop(mgr,hiredate,sal);

Table altered.

SQL> desc te;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                              NUMBER(4)
 ENAME                                              VARCHAR2(10)
 JOB                                                VARCHAR2(9)
```

注意：如果表中的数据库很大，删除列的操作慎用，否则等待时间过长，影响业务；

注意：不能将表中的所有列都删除，至少保留一列；

```
SQL> desc te;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                              NUMBER(4)
 ENAME                                              VARCHAR2(10)
 JOB                                                VARCHAR2(9)

SQL> alter table te drop(EMPNO,ENAME,JOB);
alter table te drop(EMPNO,ENAME,JOB)
*
ERROR at line 1:
ORA-12983: cannot drop all columns in a table
```

注意：删除列的时候，表中可以包含数据；但是列被删除后，数据将被清空，空间会被释放；如果后悔删除列了，不好意思，无法后退；所以工作中慎用此命令；

特例：

前提：1.有一张表，其中的字段数较多，数据量很大；2.业务较忙；

需求：删除其中一列，并立即添加一个新列，且与删除的列同名；

step 1：创建测试表 big，定义参照 sys 用户下的 dba_objects，数据保留；

```
SQL> conn / as sysdba
Connected.
SQL> show user
USER is "SYS"
SQL> create table scott.big
  2  as select * from dba_objects;

Table created.
```

step 2：向 big 表中添加大量数据，模拟实验环境；

```
SQL> conn scott/tiger
Connected.
SQL> show user
USER is "SCOTT"
SQL> insert into big
  2  select * from big;

50318 rows created.

SQL> /

100636 rows created.

SQL> /

201272 rows created.

SQL> /

402544 rows created.

SQL> commit;

Commit complete.

SQL> select count(*) from big;

  COUNT(*)
----------
    805088
```

step 3：开启时间跟踪，测试删除一列，并添加新列，且与删除的列同名；注意查看消耗的时间；

step 3.1：查看 big 表的列信息；

```
SQL> desc big;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 OWNER                                              VARCHAR2(30)
 OBJECT_NAME                                        VARCHAR2(128)
 SUBOBJECT_NAME                                     VARCHAR2(30)
 OBJECT_ID                                          NUMBER
 DATA_OBJECT_ID                                     NUMBER
 OBJECT_TYPE                                        VARCHAR2(19)
 CREATED                                            DATE
 LAST_DDL_TIME                                      DATE
 TIMESTAMP                                          VARCHAR2(19)
 STATUS                                             VARCHAR2(7)
 TEMPORARY                                          VARCHAR2(1)
 GENERATED                                          VARCHAR2(1)
 SECONDARY                                          VARCHAR2(1)
```

step 3.2：开启时间跟踪；

```
SQL> set timing on;
```

注意：以上设定是为了计算某个命令执行的时间长度；

如果想关闭时间跟踪的话，使用 set timing off 即可；

step 3.3：测试删除一列 status，注意查看使用的时间；

```
SQL> alter table big drop(status);

Table altered.

Elapsed: 00:00:29.96
```

step 3.4：添加新列 status，类型和长度与原来相同，注意查看使用的时间；

```
SQL> alter table big add(status varchar2(7));

Table altered.

Elapsed: 00:00:00.25
```

小结：1.删除一列消耗的时长：29.96 秒；　　2.添加一列消耗的时长：0.25 秒；
为了追加一列而等待了那么长时间，有些不必要，能不能缩短这个时间呢？　　呵呵，能！

优化 step 3 的操作；(采用其它方式实现此需求)
step 4：查看 big 表中列的定义信息；

```
SQL> desc big;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 OWNER                                              VARCHAR2(30)
 OBJECT_NAME                                        VARCHAR2(128)
 SUBOBJECT_NAME                                     VARCHAR2(30)
 OBJECT_ID                                          NUMBER
 DATA_OBJECT_ID                                     NUMBER
 OBJECT_TYPE                                        VARCHAR2(19)
 CREATED                                            DATE
 LAST_DDL_TIME                                      DATE
 TIMESTAMP                                          VARCHAR2(19)
 TEMPORARY                                          VARCHAR2(1)
 GENERATED                                          VARCHAR2(1)
 SECONDARY                                          VARCHAR2(1)
 STATUS                                             VARCHAR2(7)
```

step 5：将表中的某列(object_type)禁用，再查看 big 表中列的定义信息；；

```
SQL> alter table big set unused(object_type);

Table altered.

Elapsed: 00:00:00.22
SQL> desc big;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 OWNER                                              VARCHAR2(30)
 OBJECT_NAME                                        VARCHAR2(128)
 SUBOBJECT_NAME                                     VARCHAR2(30)
 OBJECT_ID                                          NUMBER
 DATA_OBJECT_ID                                     NUMBER
 CREATED                                            DATE
 LAST_DDL_TIME                                      DATE
 TIMESTAMP                                          VARCHAR2(19)
 TEMPORARY                                          VARCHAR2(1)
 GENERATED                                          VARCHAR2(1)
 SECONDARY                                          VARCHAR2(1)
 STATUS                                             VARCHAR2(7)
```

注意：禁用某列之后，效果或者现象与删除一列类似，但是禁用列中存放的数据并未被清除；
列被禁用之后，desc 查看表定义，已经没有了；select 查看数据时，也没有了；
此列被禁用后，不能被重新启用，此命令慎用；

step 6：添加新列，类型和长度与被删除的列同名；

```
SQL> alter table big add(object_type varchar2(19));

Table altered.

Elapsed: 00:00:00.14
```

小结：1.禁用一列消耗的时长：0.22 秒；　　2.添加一列消耗的时长：0.14 秒；

呵呵，好像时间节省了不少啊！但是，事情还没有结果，继续……

step 7：清空被禁用列中的数据，以便于释放空间；

```
SQL> alter table big drop unused columns;

Table altered.

Elapsed: 00:00:27.02
```

注意：清空被禁用列中的数据的操作，要选择业务不忙的时间进行；（清空数据，释放空间比较耗时）

结论：1.被删除或者禁用的列，无法再恢复；

2.如果表中有多列被禁用时，采用清空被禁用列中的数据的操作时，所有禁用列的数据将都被清空；（不能指定个别被禁用的列释放空间）

3.禁用多列的命令参照如下：（列名之间用逗号隔开即可）

```
SQL> alter table big set unused(created,timestamp);

Table altered.

Elapsed: 00:00:00.47
```

7.修改表的名字；

step 1：查看用户的对象；

```
SQL> set timing off;
SQL> show user
USER is "SCOTT"
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
DEPT                           TABLE
EMP                            TABLE
BONUS                          TABLE
SALGRADE                       TABLE
TE                             TABLE
TEMP                           TABLE
TD                             TABLE
BIG                            TABLE

8 rows selected.
```

step 2：修改某表的名字；

```
SQL> rename big to small;

Table renamed.
```

step 3：再次查看用户的对象；

```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
DEPT                           TABLE
EMP                            TABLE
BONUS                          TABLE
SALGRADE                       TABLE
TE                             TABLE
TEMP                           TABLE
TD                             TABLE
SMALL                          TABLE

8 rows selected.
```

8.修改列的名字；

step 1：查看表的定义；

```
SQL> desc small;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 OWNER                                              VARCHAR2(30)
 OBJECT_NAME                                        VARCHAR2(128)
 SUBOBJECT_NAME                                     VARCHAR2(30)
 OBJECT_ID                                          NUMBER
 DATA_OBJECT_ID                                     NUMBER
 LAST_DDL_TIME                                      DATE
 TEMPORARY                                          VARCHAR2(1)
 GENERATED                                          VARCHAR2(1)
 SECONDARY                                          VARCHAR2(1)
 STATUS                                             VARCHAR2(7)
 OBJECT_TYPE                                        VARCHAR2(19)
```

step 2：修改列的名字；

```
SQL> alter table small rename column object_id to haha;

Table altered.
```

step 3：再次查看表的定义；

```
SQL> desc small;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 OWNER                                              VARCHAR2(30)
 OBJECT_NAME                                        VARCHAR2(128)
 SUBOBJECT_NAME                                     VARCHAR2(30)
 HAHA                                               NUMBER
 DATA_OBJECT_ID                                     NUMBER
 LAST_DDL_TIME                                      DATE
 TEMPORARY                                          VARCHAR2(1)
 GENERATED                                          VARCHAR2(1)
 SECONDARY                                          VARCHAR2(1)
 STATUS                                             VARCHAR2(7)
 OBJECT_TYPE                                        VARCHAR2(19)
```

9.删除表(drop)

例：关于数据库的回收站(recyclebin)；

step 1：创建测试表 t21，定义参照 dept，数据保留；

```
SQL> show user
USER is "SCOTT"
SQL> create table t21
  2  as select * from dept;

Table created.
```

step 2：查看 scott 用户下对象的信息；

```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
DEPT                           TABLE
EMP                            TABLE
BONUS                          TABLE
SALGRADE                       TABLE
T21                            TABLE
```

step 3：查看回收站的定义和内容；

```
SQL> desc user_recyclebin
 Name                                              Null?    Type
 ------------------------------------------------- -------- ------------------------------------
 OBJECT_NAME                                       NOT NULL VARCHAR2(30)
 ORIGINAL_NAME                                              VARCHAR2(32)
 OPERATION                                                  VARCHAR2(9)
 TYPE                                                       VARCHAR2(25)
 TS_NAME                                                    VARCHAR2(30)
 CREATETIME                                                 VARCHAR2(19)
 DROPTIME                                                   VARCHAR2(19)
 DROPSCN                                                    NUMBER
 PARTITION_NAME                                             VARCHAR2(32)
 CAN_UNDROP                                                 VARCHAR2(3)
 CAN_PURGE                                                  VARCHAR2(3)
 RELATED                                           NOT NULL NUMBER
 BASE_OBJECT                                       NOT NULL NUMBER
 PURGE_OBJECT                                      NOT NULL NUMBER
 SPACE                                                      NUMBER

SQL> select object_name,original_name,type from user_recyclebin;

no rows selected
```

注意：也可以使用 show recyclebin 的命令进行查看，效果类似；

step 4：然后删除表 t21；

```
SQL> drop table t21;

Table dropped.
```

step 5：查看回收站；

```
SQL> show recyclebin;
ORIGINAL NAME    RECYCLEBIN NAME                OBJECT TYPE  DROP TIME
---------------- ------------------------------ ------------ -------------------
T21              BIN$oyLyK3Y8rZ/gQAB/AQAexA==$0 TABLE        2011-05-13:15:06:33
```

或者：

```
SQL> select object_name,original_name,type from user_recyclebin;

OBJECT_NAME                    ORIGINAL_NAME                    TYPE
------------------------------ -------------------------------- --------------------------
BIN$oyLyK3Y8rZ/gQAB/AQAexA==$0 T21                              TABLE
```

step 6：如果后悔删除 t21 那张表，想恢复回来，那怎么办？

前提：回收站中还残留此对象；

```
SQL> flashback table t21 to before drop;

Flashback complete.

SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
DEPT                           TABLE
EMP                            TABLE
BONUS                          TABLE
SALGRADE                       TABLE
T21                            TABLE
```

或者：flashback table "BIN$oyLyK3Y8rZ/gQAB/AQAexA==$0" to before drop;

注意：如果存留在回收站时间过长，空间可能被系统自动回收，对象不能再找回；

如果 drop 的表虽然放进了回收站，但是回收站的空间被手动释放了，此时文件也无法找回；

```
SQL> show recyclebin;
ORIGINAL NAME    RECYCLEBIN NAME                OBJECT TYPE  DROP TIME
---------------- ------------------------------ ------------ -------------------
T21              BIN$oyR5+bJqeAngQAB/AQBOsA==$0 TABLE        2011-05-13:15:44:32
SQL> purge recyclebin;

Recyclebin purged.
```

但是,如果在 drop 表的时候没有放进回收站，而是直接删除(像 windows 下用 shift+delete 删除文件)，那么文件不能再找回来；

```
SQL> create table t22 as select * from dept;

Table created.

SQL> drop table t22 purge;

Table dropped.

SQL> show recyclebin;
SQL>
```

特例：如果想闪回回收站中特定的某张表，那该怎么办？

step 1：创建测试表 t23，定义参照 dept，数据保留；

```
SQL> show user
USER is "SCOTT"
SQL> create table t23
  2  as select * from dept;

Table created.
```

step 2：查看一下 t23 的数据，然后 drop 此表；

```
SQL> select * from t23;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

SQL> drop table t23;

Table dropped.
```

step 3：查看回收站；

```
SQL> show recyclebin;
ORIGINAL NAME    RECYCLEBIN NAME                OBJECT TYPE  DROP TIME
---------------- ------------------------------ ------------ -------------------
T23              BIN$oyO2v92BQmXgQAB/AQAkgw==$0 TABLE        2011-05-13:16:01:32
```

step 4：再重建 t23，定义参照 dept，数据不要；

```
SQL> show user
USER is "SCOTT"
SQL> create table t23
  2  as select * from dept where 1=2;

Table created.

SQL> select * from t23;

no rows selected
```

step 5：添加一条测试数据，并查看，然后 drop 此表；

```
SQL> insert into t23 values(99,'hebut','haha');

1 row created.

SQL> commit;

Commit complete.

SQL> select * from t23;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        99 hebut          haha

SQL> drop table t23;

Table dropped.
```

step 6：查看回收站；

```
SQL> show recyclebin;
ORIGINAL NAME    RECYCLEBIN NAME                OBJECT TYPE  DROP TIME
---------------- ------------------------------ ------------ -------------------
T23              BIN$oyO2v92CQmXgQAB/AQAkgw==$0 TABLE        2011-05-13:16:03:48
T23              BIN$oyO2v92BQmXgQAB/AQAkgw==$0 TABLE        2011-05-13:16:01:32
```

注意：表 t23 在不同的时间创建，又在不同的时间被删除；

step 7：再重建 t23，定义参照 emp，数据保留 3 条；

```
SQL> show user
USER is "SCOTT"
SQL> create table t23
  2  as select * from emp where rownum <= 3;

Table created.
```

step 8：查看一下 t23 的数据，然后 drop 此表；

```
SQL> set linesize 100
SQL> select * from t23;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30

SQL> drop table t23;

Table dropped.
```

step 9：查看回收站；

```
SQL> show recyclebin;
ORIGINAL NAME    RECYCLEBIN NAME                OBJECT TYPE  DROP TIME
---------------- ------------------------------ ------------ -------------------
T23              BIN$oyO2v92DQmXgQAB/AQAkgw==$0 TABLE        2011-05-13:16:06:35
T23              BIN$oyO2v92CQmXgQAB/AQAkgw==$0 TABLE        2011-05-13:16:03:48
T23              BIN$oyO2v92BQmXgQAB/AQAkgw==$0 TABLE        2011-05-13:16:01:32
SQL>
```

注意：此时回收站中有 3 张表的原名都叫 t23，如果想恢复包含 haha 数据的那张表，该怎么办？

注意：如果使用默认闪回命令(flashback table t23 to before drop;)，那么恢复的是最新 drop 的那张表，但是那张表里没有测试数据 haha；

step 10：首先，确认要恢复的是哪张表，即哪张表里包含 haha；（依次查看）

提示：按照删除时间的先后顺序查找想恢复的对象；

```
SQL> select * from "BIN$oyO2v92BQmXgQAB/AQAkgw==$0";

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

SQL> select * from "BIN$oyO2v92CQmXgQAB/AQAkgw==$0";

    DEPTNO DNAME          LOC
---------- -------------- -------------
        99 hebut          haha

SQL> select * from "BIN$oyO2v92DQmXgQAB/AQAkgw==$0";

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
```

注意：表被 drop 的时候，实际上是将表进行了改名，原表里的数据并未被清除，只是被标记为可回收；
表明被改成了乱七八糟的名字，所以在查看表内数据的时候要加双引号；

小结：由此可确定，表“BIN$oyO2v92CQmXgQAB/AQAkgw==$0”内含有 haha 数据；
由此可将此表进行恢复；

step 11：其次，恢复已确认的对象；

```
SQL> flashback table "BIN$oyO2v92CQmXgQAB/AQAkgw==$0" to before drop;

Flashback complete.
```

step 12：最后，查看对象是否恢复成功，数据是否正确；

```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
DEPT                           TABLE
EMP                            TABLE
BONUS                          TABLE
SALGRADE                       TABLE
BIN$oyO2v92BQmXgQAB/AQAkgw==$0 TABLE
BIN$oyO2v92DQmXgQAB/AQAkgw==$0 TABLE
T23                            TABLE

7 rows selected.

SQL> select * from t23;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        99 hebut          haha
```

拓展：如果想释放回收站中特定的某些对象占用的空间，那该怎么办？

例：释放和 emp 表关联的那个 t23 表的空间；

step 1：查看回收站的内容；

```
SQL> show recyclebin;
ORIGINAL NAME    RECYCLEBIN NAME                OBJECT TYPE  DROP TIME
---------------- ------------------------------ ------------ -------------------
T23              BIN$oyO2v92DQmXgQAB/AQAkgw==$0 TABLE        2011-05-13:16:06:35
T23              BIN$oyO2v92BQmXgQAB/AQAkgw==$0 TABLE        2011-05-13:16:01:32
```

step 2：查看哪个表和 emp 表的数据关联；

```
SQL> select * from "BIN$oyO2v92BQmXgQAB/AQAkgw==$0";

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

SQL> select * from "BIN$oyO2v92DQmXgQAB/AQAkgw==$0";

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
```

注意：由此确定表“BIN$oyO2v92DQmXgQAB/AQAkgw==$0”为查找的对象；

step 3：将上述确定的对象进行空间释放；

```
SQL> purge table "BIN$oyO2v92DQmXgQAB/AQAkgw==$0";

Table purged.
```

step 4：确认一下回收站中的对象被释放；

```
SQL> show recyclebin;
ORIGINAL NAME    RECYCLEBIN NAME                OBJECT TYPE  DROP TIME
---------------- ------------------------------ ------------ -------------------
T23              BIN$oyO2v92BQmXgQAB/AQAkgw==$0 TABLE        2011-05-13:16:01:32
SQL> select * from "BIN$oyO2v92BQmXgQAB/AQAkgw==$0";

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

10.截断表(truncate)

提示：truncate 命令与 delete 命令相似；

区别：delete：属于 DML 操作，在没有 commit 之前可以进行 rollback；被 delete 的数据可以进行闪回查询、闪回恢复等；被 delete 的数据所占的空间暂时不被系统回收；

truncate:属于 DDL 操作,会进行隐式 commit,所以被 truncate 的数据不能进行恢复;被 truncate 的数据所占的空间被释放；(工作中此命令慎用)

例：

step 1：创建测试表 t24，定义参照 dept，数据保留；

```
SQL> show user
USER is "SCOTT"
SQL> create table t24
  2  as select * from dept;

Table created.
```

step 2：truncate 表 t24；

```
SQL> truncate table t24;

Table truncated.
```

step 3：查询 t24 的数据；

```
SQL> select * from t24;

no rows selected
```

11.添加注释(comment)；

例：新建一张表 t25，字段定义如下；

```
SQL> show user
USER is "SCOTT"
SQL> create table t25(id number(6),name varchar2(20),loc varchar2(20),dt date);

Table created.

SQL> desc t25
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------------------
 ID                                                 NUMBER(6)
 NAME                                               VARCHAR2(20)
 LOC                                                VARCHAR2(20)
 DT                                                 DATE
```

11.1 给表名 t25 添加注释；

```
SQL> show user
USER is "SCOTT"
SQL> comment on table t25 is 'This is a student info table.';

Comment created.
```

11.2 查看关于表名 t25 的注释；

```
SQL> col comments for a30
SQL> select * from user_tab_comments
  2  where table_name = 'T25';

TABLE_NAME                     TABLE_TYPE  COMMENTS
------------------------------ ----------- ------------------------------
T25                            TABLE       This is a student info table.
```

11.3 给表 t25 中的字段添加注释；

```
SQL> comment on column t25.name is 'This is a name of student.';

Comment created.
```

11.4 查看关于表 t25 中字段的注释；

```
SQL> set pagesize 30
SQL> col comments for a30
SQL> select * from user_col_comments
  2  where table_name = 'T25';

TABLE_NAME                     COLUMN_NAME                    COMMENTS
------------------------------ ------------------------------ ------------------------------
T25                            ID
T25                            NAME                           This is a name of student.
T25                            LOC
T25                            DT
```

【第十章 内置约束】

1.约束的类型

提示：

1） not null：表示指定的列不能包含空值；

2） unique：表示指定的列的值或者列的组合的值对于表中的行数据而言必须是唯一的；

3） primary key：表示表中每条数据的唯一性标识；(满足非空唯一性)

4） foreign key：表示在列与引用表的列之间建立的一种强制外键关系；

5） check：表示指定一个必须为真的条件；

注意：1.对于约束的命名，可以有自己来定义，或者默认按照 Oracle 指定的格式命名(SYS_Cn)；

2.对于 not null 约束，只能定义在列级上；(只能紧跟在列定义之后写)

对于其他的约束，可以定义在列级或者表级；(可以紧跟着列定义写，也可以在列定义完了之后在写)；

3.对于同一用户下的约束，约束名不能重复；

4.对于约束的定义信息，可以查 user_constraints；

2.各种约束的使用

2.1 not null 约束；

例：

step 1：创建测试表 t51，并添加 not null 约束；

```
SQL> show user
USER is "SCOTT"
SQL> create table t51(id number(4) not null,
  2                   name varchar2(20),info varchar2(100));

Table created.
```

注意：不能将 not null 约束定义在表级；

```
SQL> create table t51(id number(4),name varchar2(20),info varchar2(100),
  2                   constraint id_haha not null);
                constraint id_haha not null)
                                   *
ERROR at line 2:
ORA-00904: : invalid identifier
```

或者，可以在创建完表之后再追加；

step 2：创建测试表 t52；

```
SQL> create table t52(id number(4),name varchar2(20),info varchar2(100));

Table created.

SQL> desc t52
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------------
 ID                                                             NUMBER(4)
 NAME                                                           VARCHAR2(20)
 INFO                                                           VARCHAR2(100)
```

step 3：添加约束；

```
SQL> alter table t52 add(id number(4) not null);
alter table t52 add(id number(4) not null)
                    *
ERROR at line 1:
ORA-01430: column being added already exists in table
```

注意：添加列级 not null 约束时，不能使用 add 命令；需要用 modify 命令；

```
SQL> alter table t52 modify(id number(4) not null);

Table altered.

SQL> desc t52
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER(4)
 NAME                                               VARCHAR2(20)
 INFO                                               VARCHAR2(100)
```

注意：表中如包含不满足约束的数据时，则约束添加报错！

step 4：查看约束的定义；

```
SQL> select constraint_name,constraint_type,table_name from user_constraints
  2  where table_name in ('T51','T52');

CONSTRAINT_NAME                C TABLE_NAME
------------------------------ - ------------------------------
SYS_C005396                    C T51
SYS_C005397                    C T52
```

注意：Oracle 自定义的约束名管理起来不方便，自定义一下；

step 5：创建表 t53，添加 not null 约束；

```
SQL> create table t53(id number(4) constraint id_haha not null,
  2                   name varchar2(20),info varchar2(100));

Table created.

SQL> desc t53
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER(4)
 NAME                                               VARCHAR2(20)
 INFO                                               VARCHAR2(100)
```

step 6：查看约束的定义；

```
SQL> select constraint_name,constraint_type,table_name from user_constraints
  2  where table_name = 'T53';

CONSTRAINT_NAME                C TABLE_NAME
------------------------------ - ------------------------------
ID_HAHA                        C T53
```

step 7：测试约束是否有效；

```
SQL> insert into t53
  2  values(null,'itaa','It is a company.');
values(null,'itaa','It is a company.')
       *
ERROR at line 2:
ORA-01400: cannot insert NULL into ("SCOTT"."T53"."ID")


SQL> insert into t53
  2  values(1234,'itaa','It is a company.');

1 row created.
```

2.2 unique 约束；

例：

step 1：创建测试表 t54，添加列级 unique 约束；

```
SQL> create table t54(id number(4) constraint id_un_haha unique,
  2                   name varchar2(20),info varchar2(100));

Table created.

SQL> desc t54
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------------
 ID                                                             NUMBER(4)
 NAME                                                           VARCHAR2(20)
 INFO                                                           VARCHAR2(100)
```

如果是表级 unique 约束，应该这样来写；

```
SQL> create table t55(id number(4),name varchar2(20),info varchar2(100),
  2                   constraint id_un_hehe unique(id));
```

注意：info 后面有个逗号；约束定义在表级时，需指定为哪列(id)添加约束；

step 2：查看约束的信息；

```
SQL> select constraint_name,constraint_type,table_name from user_constraints
  2  where table_name in ('T54','T55');

CONSTRAINT_NAME                C TABLE_NAME
------------------------------ - ------------------------------
ID_UN_HAHA                     U T54
ID_UN_HEHE                     U T55
```

step 3：测试约束是否有效；

```
SQL> insert into t54(id,name) values(1,'itaa');

1 row created.

SQL> insert into t54(id,name) values(2,'itbb');

1 row created.

SQL> insert into t54(id,name) values(1,'itab');
insert into t54(id,name) values(1,'itab')
*
ERROR at line 1:
ORA-00001: unique constraint (SCOTT.ID_UN_HAHA) violated
```

特例：

step 4：向 t55 中添加如下数据，成功！为什么？

```
SQL> insert into t55 values(null,'itaa','first row');

1 row created.

SQL> insert into t55 values(null,'itbb','second row');

1 row created.

SQL> insert into t55 values(null,'itcc','third row');

1 row created.

SQL> commit;

Commit complete.
```

step 5：查看 t55 的数据；

```
SQL> col info for a30
SQL> select * from t55;

        ID NAME                 INFO
---------- -------------------- ------------------------------
           itaa                 first row
           itbb                 second row
           itcc                 third row
```

原因：因为 null 之间无法比较大小，所以 Oracle 认为 null 值之间是不相等的，即满足 unique 的约束；

证明：

```
SQL> select count(*) from t55 where id = null;

  COUNT(*)
----------
         0
```

注意：如果此结果小于等于 1，则满足 unique 约束；哈哈，这是 Oracle 的算法问题，毕竟它不等于 4；

2.3 primary key 约束；

例：

step 1：创建测试表 t56，添加列级主键约束；

```
SQL> create table t56(id number(4) constraint id_pk_haha primary key,
  2                   name varchar2(20),info varchar2(100));

Table created.

SQL> desc t56
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------------
 ID                                                    NOT NULL NUMBER(4)
 NAME                                                           VARCHAR2(20)
 INFO                                                           VARCHAR2(100)
```

如果是表级 unique 约束，应该这样来写；

```
SQL> create table t57(id number(4),name varchar2(20),info varchar2(100),
  2                   constraint id_pk_hehe primary key(id));

Table created.

SQL> desc t57
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------------
 ID                                                    NOT NULL NUMBER(4)
 NAME                                                           VARCHAR2(20)
 INFO                                                           VARCHAR2(100)
```

step 2：查看约束的信息；

```
SQL> select constraint_name,constraint_type,table_name from user_constraints
  2  where table_name in ('T56','T57');

CONSTRAINT_NAME              C TABLE_NAME
---------------------------- - ------------------------------
ID_PK_HAHA                   P T56
ID_PK_HEHE                   P T57
```

step 3：测试约束是否有效；

提示：主键满足非空，唯一性；（因此可以唯一的标识表中某一行数据）

测试 1：输入非空的值；

```
SQL> insert into t56 values(1,'itaa','tian jin');

1 row created.

SQL> insert into t56 values(2,'itbb','bei jing');

1 row created.

SQL> insert into t56 values(1,'itab','tian bei');
insert into t56 values(1,'itab','tian bei')
*
ERROR at line 1:
ORA-00001: unique constraint (SCOTT.ID_PK_HAHA) violated
```

测试 2：输入 null 值；

```
SQL> insert into t56 values(null,'itaa','tian jin');
insert into t56 values(null,'itaa','tian jin')
                       *
ERROR at line 1:
ORA-01400: cannot insert NULL into ("SCOTT"."T56"."ID")
```

小结：之前写的 unique、primary key 是添加在一列上，当然也可以添加在多列上，作为一个复合约束存在；

step 4：创建测试表 t58，添加多列的 unique、primary key 约束；

```
SQL> create table t58(empno number(10),ename varchar2(20),deptno number(4),job varchar2(10),sal number(6,2),
  2                   constraint e12_pk_haha primary key(empno,ename),
  3                   constraint e34_un_hehe unique(deptno,job));

Table created.

SQL> desc t58
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------------
 EMPNO                                                 NOT NULL NUMBER(10)
 ENAME                                                 NOT NULL VARCHAR2(20)
 DEPTNO                                                         NUMBER(4)
 JOB                                                            VARCHAR2(10)
 SAL                                                            NUMBER(6,2)
```

step 5：查看 t58 上的约束；

```
SQL> select constraint_name,constraint_type,table_name from user_constraints
  2  where table_name = 'T58';

CONSTRAINT_NAME                C TABLE_NAME
------------------------------ - ------------------------------
E12_PK_HAHA                    P T58
E34_UN_HEHE                    U T58
```

注意：以上多列的 unique、primary key 约束，只能定义在表级上；

2.4 foreign key 约束；

提示：外键是基于父表中的主键存在的，是逻辑的约束关系；

不能创建不存在主键的外键；(报错)

step 1：创建父表 t60，定义参照 dept，数据保留；

```
SQL> show user
USER is "SCOTT"
SQL> create table t60 as select * from dept;

Table created.

SQL> select * from t60;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

注意：在使用子查询方式建表时，只会参照定义，不会继承约束关系；

step 2：在父表 t60 的 deptno 上添加主键；

```
SQL> alter table t60 add(constraint t60_pk_haha primary key(deptno));

Table altered.
```

step 3：创建子表 t61，定义参照 emp，数据不要；

```
SQL> create table t61 as select * from emp where 1=2;

Table created.

SQL> select * from t61;

no rows selected
```

step 4：在子表 t61 的 deptno 上添加外键；

```
SQL> alter table t61 add(constraint t61_fk_haha foreign key(deptno)
  2                      references t60(deptno));

Table altered.
```

step 5：查看 t60 和 t61 中约束的定义；

```
SQL> select constraint_name,constraint_type,table_name from user_constraints
  2  where table_name in ('T60','T61');

CONSTRAINT_NAME                C TABLE_NAME
------------------------------ - ------------------------------
T60_PK_HAHA                    P T60
T61_FK_HAHA                    R T61
```

或者，可查看这些约束关联的列的信息；(user_cons_columns)

```
SQL> set linesize 120
SQL> col column_name for a30
SQL> select * from user_cons_columns where table_name in ('T60','T61');

OWNER                          CONSTRAINT_NAME                TABLE_NAME
------------------------------ ------------------------------ ------------------------------
COLUMN_NAME                      POSITION
------------------------------ ----------
SCOTT                          T61_FK_HAHA                    T61
DEPTNO                                  1

SCOTT                          T60_PK_HAHA                    T60
DEPTNO                                  1
```

step 6：测试主外键约束是否有效；

测试 1：向子表 t61 添加数据；

如果，子表 t61 的 deptno 的值在父表 t60 的 deptno 的值查询不到，则插入子表数据时报错；

```
SQL> select * from t61;

no rows selected

SQL> insert into t61(empno,ename,sal,deptno) values(1112,'a',1000,10);

1 row created.

SQL> commit;

Commit complete.

SQL> insert into t61(empno,ename,sal,deptno) values(1113,'b',1200,99);
insert into t61(empno,ename,sal,deptno) values(1113,'b',1200,99)
*
ERROR at line 1:
ORA-02291: integrity constraint (SCOTT.T61_FK_HAHA) violated - parent key not found
```

测试 2：删除父表 t60 中的数据；

如果，父表 t60 的 deptno 的值在子表 t61 的 deptno 的值正在使用的话，则删除父表数据时报错；

```
SQL> select * from t61;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      1112 a                                           1000                    10

SQL> delete t60 where deptno = 10;
delete t60 where deptno = 10
*
ERROR at line 1:
ORA-02292: integrity constraint (SCOTT.T61_FK_HAHA) violated - child record found
```

简单而言，不允许没有爸爸的孩子存在；

但是，父表毕竟是基础表，如果你想删除爸爸记录的时候，顺便把孩子删掉！God，传说中的‘株连九族’；

实现方式 1：使用 on delete cascade 关键词

step 1：准备子表 t61 的测试数据；

```
SQL> insert into t61
  2  select * from emp where empno in (7369,7499);

2 rows created.

SQL> commit;

Commit complete.

SQL> select * from t61;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      1112 a                                           1000                    10
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
```

step 2：修改子表 t61 的外键定义；

```
SQL> alter table t61 drop constraint t61_fk_haha;

Table altered.

SQL> alter table t61 add(constraint t61_fk_haha_1 foreign key(deptno)
  2                      references t60(deptno) on delete cascade);

Table altered.
```

step 3：删除父表 t60 的数据；

```
SQL> select * from t60;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

SQL> delete t60 where deptno = 10;

1 row deleted.

SQL> commit;

Commit complete.
```

step 4：查看 t60 和 t61 的相关数据；

```
SQL> select * from t60;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

SQL> select * from t61;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
```

实现方式 2：使用 on delete set null 关键词

step 1：准备子表 t61 的测试数据；

```
SQL> insert into t60
  2  select * from dept where deptno = 10;

1 row created.

SQL> insert into t61
  2  select * from emp where empno = 7782;

1 row created.

SQL> commit;

Commit complete.
```

注意：向表中插入数据时，先父表 t60，后子表 t61；否则，报错！

step 2：查看 t60 和 t61 的测试数据；

```
SQL> select * from t60;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
        10 ACCOUNTING     NEW YORK

SQL> select * from t61;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
```

step 3：修改子表 t61 的外键定义；

```
SQL> alter table t61 drop constraint t61_fk_haha_1;

Table altered.

SQL> alter table t61 add(constraint t61_fk_haha_2 foreign key(deptno)
  2                      references t60(deptno) on delete set null);

Table altered.
```

step 4：删除父表 t60 的数据；

```
SQL> delete t60 where deptno = 20;

1 row deleted.

SQL> commit;

Commit complete.
```

step 5：查看 t60 和 t61 的相关数据；

```
SQL> select * from t60;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
        10 ACCOUNTING     NEW YORK

SQL> select * from t61;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
```

哎，方式 2 毕竟还好点，留下了孤儿；

注意：工作中如果存在垃圾数据的话，要及时进行处理；

2.5 check 约束；

例：向表中插入数据时，检查数据是否满足条件，与 with check option 类似；

step 1：创建测试表 t62，将 check 约束添加在列级；

```
SQL> create table t62(no number(4),name varchar2(20),
  2                   sal number(6,2) constraint t62_chk_haha check(sal > 2000));

Table created.
```

如果添加在表级，应该这样写：

```
SQL> create table t63(no number(4),name varchar2(20),sal number(6,2),
  2                   constraint t62_chk_hehe check(sal > 2000));

Table created.
```

当然也可以在表定义完之后，再进行添加，命令如下：

```
SQL> create table t64(no number(4),name varchar2(20),sal number(6,2));

Table created.

SQL> alter table t64 add constraint t62_chk_xixi check(sal > 2000);

Table altered.
```

step 2：向 t62 中添加数据进行测试，查看约束是否有效；

```
SQL> insert into t62 values(1001,'a',2500);

1 row created.

SQL> insert into t62 values(1002,'b',1500);
insert into t62 values(1002,'b',1500)
*
ERROR at line 1:
ORA-02290: check constraint (SCOTT.T62_CHK_HAHA) violated
```

3.稍微总结一下：

1）not null 约束只能在列级添加，并且只能通过 modify 命令进行滞后添加；（不能用 add 命令）

2）除 not null 之外的约束，可以在列级或者表级进行定义；

3）C 表示 check，P 表示 primary key，R 表示 foreign key，U 表示 unique；（not null 属于 check 约束）

4. 特殊处理

例：删除主键时，若有外键关联，须注意；

4.1 回想父表 t60 和子表 t61 的关系；

1） 若主键独立存在时，删除命令参照如下：

alter table 表名 drop constraint 主键约束的名字；

注意：要是 Oracle 自动定义的名字，你得先查出来，然后再删；

或者参照以下命令：

alter table 表名 drop constraint primary key；

因为主键可以唯一定位数据，是表中数据的唯一标识，所以如上写的时候没有问题；

2） 若主键存在外键关联，则单纯删除主键会报错，应该关联删除，命令参照如下：

alter table 表名 drop constraint primary key cascade；

3） 当然在 drop 表的时候，也可以如下操作：(表示跟表关联的约束一并删除)

drop table 表名 cascade constraints；

5.约束的禁用和启用

5.1 关于约束的禁用；

step 1：创建测试表 t65，定义参照 emp，数据不要；

```
SQL> show user
USER is "SCOTT"
SQL> create table t65
  2  as select * from emp where 1=2;

Table created.

SQL> desc t65
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                              NUMBER(4)
 ENAME                                              VARCHAR2(10)
 JOB                                                VARCHAR2(9)
 MGR                                                NUMBER(4)
 HIREDATE                                           DATE
 SAL                                                NUMBER(7,2)
 COMM                                               NUMBER(7,2)
 DEPTNO                                             NUMBER(2)
```

step 2：向 t65 添加 not null、unique、primary key 约束；

```
SQL> alter table t65 add constraint pk_haha primary key(empno,deptno);

Table altered.

SQL> alter table t65 add constraint un_haha unique(ename,job,mgr);

Table altered.

SQL> alter table t65 add constraint not_haha check(sal > 2000);

Table altered.
```

注意：not null 约束的添加使用 modify 命令；

```
SQL> alter table t65 modify(ename varchar2(10) constraint not_hehe not null);

Table altered.
```

step 3：查看 t65 中约束的信息；

```
SQL> col owner for a20;
SQL> col constraint_name for a20;
SQL> col table_name for a20;
SQL> col column_name for a20;
SQL> select owner,constraint_name,table_name,column_name
  2  from user_cons_columns where table_name = 'T65';

OWNER                CONSTRAINT_NAME      TABLE_NAME           COLUMN_NAME
-------------------- -------------------- -------------------- --------------------
SCOTT                NOT_HEHE             T65                  ENAME
SCOTT                NOT_HAHA             T65                  SAL
SCOTT                PK_HAHA              T65                  EMPNO
SCOTT                PK_HAHA              T65                  DEPTNO
SCOTT                UN_HAHA              T65                  ENAME
SCOTT                UN_HAHA              T65                  JOB
SCOTT                UN_HAHA              T65                  MGR

7 rows selected.
```

注意：col 列名 for a 数字，表示设定列的显示格式；

例：col owner for a20 这句话等同于 column owner format a20；将 owner 以 20 字符宽度进行显示；

step 4：添加测试数据，测试约束是否有效；

测试 1： 非空约束；

```
SQL> insert into t65(empno,ename,job,sal,deptno)
  2  values(1,null,'a',1000,10);
values(1,null,'a',1000,10)
         *
ERROR at line 2:
ORA-01400: cannot insert NULL into ("SCOTT"."T65"."ENAME")
```

测试 2： check 约束；

```
SQL> insert into t65(empno,ename,job,sal,deptno)
  2  values(2,'name2','b',3000,20);

1 row created.

SQL> insert into t65(empno,ename,job,sal,deptno)
  2  values(1,'name01','a',1000,10);
insert into t65(empno,ename,job,sal,deptno)
*
ERROR at line 1:
ORA-02290: check constraint (SCOTT.NOT_HAHA) violated


SQL> commit;

Commit complete.
```

测试 3： 主键约束；

```
SQL> select * from t65;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
         2 name2      b                                   3000                    20

SQL> insert into t65(empno,ename,job,sal,deptno)
  2  values(2,'name22','c',4000,20);
insert into t65(empno,ename,job,sal,deptno)
*
ERROR at line 1:
ORA-00001: unique constraint (SCOTT.PK_HAHA) violated
```

测试 4： 唯一性约束；

```
SQL> insert into t65(empno,ename,job,mgr,sal,deptno)
  2  values(3,'name3','c',1,3600,30);

1 row created.

SQL> insert into t65(empno,ename,job,mgr,sal,deptno)
  2  values(4,'name3','c',1,3700,40);
insert into t65(empno,ename,job,mgr,sal,deptno)
*
ERROR at line 1:
ORA-00001: unique constraint (SCOTT.UN_HAHA) violated
```

step 5：禁用相关约束；

提示：查看 step 3 中各约束的相关信息；(禁用了 not null 和主键约束)

```
SQL> alter table t65 disable constraint not_hehe;

Table altered.

SQL> alter table t65 disable constraint pk_haha;

Table altered.
```

step 6：再查看 t65 中约束的信息；(disable 失效)

```
SQL> select constraint_name,constraint_type,table_name,status
  2  from user_constraints where table_name = 'T65';

CONSTRAINT_NAME      C TABLE_NAME           STATUS
-------------------- - -------------------- --------
PK_HAHA              P T65                  DISABLED
UN_HAHA              U T65                  ENABLED
NOT_HAHA             C T65                  ENABLED
NOT_HEHE             C T65                  DISABLED

SQL> select owner,constraint_name,table_name,column_name
  2  from user_cons_columns where table_name = 'T65';

OWNER                CONSTRAINT_NAME      TABLE_NAME           COLUMN_NAME
-------------------- -------------------- -------------------- --------------------
SCOTT                NOT_HEHE             T65                  ENAME
SCOTT                NOT_HAHA             T65                  SAL
SCOTT                PK_HAHA              T65                  EMPNO
SCOTT                PK_HAHA              T65                  DEPTNO
SCOTT                UN_HAHA              T65                  ENAME
SCOTT                UN_HAHA              T65                  JOB
SCOTT                UN_HAHA              T65                  MGR
```

step 7：添加测试数据，测试约束是否失效；

前提：查看 t65 中的数据；

```
SQL> select * from t65;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
         2 name2      b                                   3000                    20
         3 name3      c                 1                 3600                    30
```

测试 1： 非空约束；(禁用 not null 约束后，ename 中即可放入 null 值)

```
SQL> insert into t65(empno,ename,job,sal,deptno)
  2  values(5,'name5','d',5000,50);

1 row created.

SQL> insert into t65(empno,ename,job,sal,deptno)
  2  values(6,null,'e',5500,60);

1 row created.

SQL> commit;

Commit complete.
```

测试 2： 主键约束；

1）查看 t65 中的数据；

```
SQL> select * from t65;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
         2 name2      b                                   3000                    20
         3 name3      c                 1                 3600                    30
         5 name5      d                                   5000                    50
         6            e                                   5500                    60
```

2） 插入测试数据；(其中(empno,deptno)=(2,20)为不正常数据，因为 t65 中已经存在；)

```
SQL> insert into t65(empno,ename,job,mgr,sal,deptno)
  2  values(7,'name7','f',1,7000,70);

1 row created.

SQL> insert into t65(empno,ename,job,mgr,sal,deptno)
  2  values(2,'name8','g',1,8000,20);

1 row created.

SQL> commit;

Commit complete.
```

小结：有上述实验可断定 not null 和主键约束已经失效；

但是，为了下面的实验，顺便也将非法数据进行了 commit；

5.2 关于约束的启用；

提示：查看 t65 中的数据；

```
SQL> select * from t65;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
         2 name2      b                             3000                    20
         3 name3      c                  1          3600                    30
         5 name5      d                             5000                    50
         6            e                             5500                    60
         7 name7      f                  1          7000                    70
         2 name8      g                  1          8000                    20

6 rows selected.
```

step 1：在 user_constraints 和 user_cons_columnst65 中查询关于 t65 中约束的信息；(disable 失效)

```
SQL> select constraint_name,constraint_type,table_name,status
  2  from user_constraints where table_name = 'T65';

CONSTRAINT_NAME      C TABLE_NAME           STATUS
-------------------- - -------------------- --------
PK_HAHA              P T65                  DISABLED
UN_HAHA              U T65                  ENABLED
NOT_HAHA             C T65                  ENABLED
NOT_HEHE             C T65                  DISABLED

SQL> select owner,constraint_name,table_name,column_name
  2  from user_cons_columns where table_name = 'T65';

OWNER                CONSTRAINT_NAME      TABLE_NAME           COLUMN_NAME
-------------------- -------------------- -------------------- --------------------
SCOTT                NOT_HEHE             T65                  ENAME
SCOTT                NOT_HAHA             T65                  SAL
SCOTT                PK_HAHA              T65                  EMPNO
SCOTT                PK_HAHA              T65                  DEPTNO
SCOTT                UN_HAHA              T65                  ENAME
SCOTT                UN_HAHA              T65                  JOB
SCOTT                UN_HAHA              T65                  MGR

7 rows selected.
```

step 2：启用被禁用的约束；

提示：由 step 1 中可知，关于 ename 的非空约束 not_hehe、关于 empno、deptno 的联合主键约束 pk_haha 目前处于禁用的状态；

```
SQL> alter table t65 enable constraint not_hehe;
alter table t65 enable constraint not_hehe
                                  *
ERROR at line 1:
ORA-02293: cannot validate (SCOTT.NOT_HEHE) - check constraint violated


SQL> alter table t65 enable constraint pk_haha;
alter table t65 enable constraint pk_haha
*
ERROR at line 1:
ORA-02437: cannot validate (SCOTT.PK_HAHA) - primary key violated
```

注意：因为 t65 中存在不满足约束的数据，所以约束启用时，会报错！

step 3：查询 t65 中不满足约束的数据，然后进行处理；

step 3.1：查询；

```
SQL> select * from t65 where ename is null;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
         6            e                                   5500                    60

SQL> select empno,deptno,count(*)
  2  from t65
  3  group by empno,deptno;

     EMPNO     DEPTNO   COUNT(*)
---------- ---------- ----------
         5         50          1
         6         60          1
         7         70          1
         2         20          2
         3         30          1
```

注意：count(*) = 2，表示存在主键重复的数据；

step 3.2：处理；（可删除不需要的数据，或者做其他处理，由自己定义；）

```
SQL> update t65 set ename = 'name6' where empno = 6 and deptno = 60;

1 row updated.

SQL> update t65 set empno = 8, deptno = 80 where ename = 'name8';

1 row updated.

SQL> commit;

Commit complete.

SQL> select * from t65;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
         2 name2      b                                   3000                    20
         3 name3      c                   1                3600                    30
         5 name5      d                                   5000                    50
         6 name6      e                                   5500                    60
         7 name7      f                   1                7000                    70
         8 name8      g                   1                8000                    80

6 rows selected.
```

step 4：由此处理之后，再将相关约束启用；

```
SQL> alter table t65 enable constraint not_hehe;

Table altered.

SQL> alter table t65 enable constraint pk_haha;

Table altered.
```

step 5：至于启用之后约束是否生效，可参照 5.1 的 step 4；

【第十一章 创建视图】

1.回忆

例：如果想知道 emp 表中每位员工所在的部门名称，以及个人的工资等级情况，该怎么办？

提示：emp 包含员工的信息，dept 包含部门的信息，salgrade 包含工资的等级信息；

实现方式：通过多表关联查询；

```
SQL> show user
USER is "SCOTT"
SQL> set linesize 100
SQL> set pagesize 30
SQL> select e.ename,d.dname,s.grade
  2  from emp e,dept d,salgrade s
  3  where e.deptno = d.deptno
  4  and e.sal between s.losal and s.hisal;

ENAME      DNAME               GRADE
---------- -------------- ----------
SMITH      RESEARCH                1
JAMES      SALES                   1
ADAMS      RESEARCH                1
WARD       SALES                   2
MARTIN     SALES                   2
MILLER     ACCOUNTING              2
TURNER     SALES                   3
ALLEN      SALES                   3
CLARK      ACCOUNTING              4
BLAKE      SALES                   4
JONES      RESEARCH                4
SCOTT      RESEARCH                4
FORD       RESEARCH                4
KING       ACCOUNTING              5

14 rows selected.
```

如果 LD 来视察工作，想看以上的信息，那你该怎么办呢？

让 LD 直接看 emp,dept,salgrade 表？如果 LD 发现 emp 中的 sal 很有意思，或者 LD 要求看所有的信息，那你该怎么办？给不给看呢？　那你写个关联查询给 LD，好像不太好！

参照如下方式：

哈哈，那就写个 select * 让 LD 放心…

方式一：

step 1：创建表 e01，存放关联查询得到的数据；

```
SQL> create table e01
  2  as select e.ename,d.dname,s.grade
  3     from emp e,dept d,salgrade s
  4     where e.deptno = d.deptno
  5     and e.sal between s.losal and s.hisal;

Table created.
```

step 2：查询 e01 的全部数据；

```
SQL> select * from e01;

ENAME      DNAME               GRADE
---------- -------------- ----------
SMITH      RESEARCH                1
JAMES      SALES                   1
ADAMS      RESEARCH                1
WARD       SALES                   2
MARTIN     SALES                   2
MILLER     ACCOUNTING              2
TURNER     SALES                   3
ALLEN      SALES                   3
CLARK      ACCOUNTING              4
BLAKE      SALES                   4
JONES      RESEARCH                4
SCOTT      RESEARCH                4
FORD       RESEARCH                4
KING       ACCOUNTING              5

14 rows selected.
```

注意：此 e01 中的数据不能自动随着基表(emp、dept、salgrade)的数据的改变而改变；

方式二：

step 1：创建视图 v01，定义参照关联查询；

```
SQL> create view v01
  2  as select e.ename,d.dname,s.grade
  3     from emp e,dept d,salgrade s
  4     where e.deptno = d.deptno
  5     and e.sal between s.losal and s.hisal;

View created.

SQL> desc v01
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------------
 ENAME                                                          VARCHAR2(10)
 DNAME                                                          VARCHAR2(14)
 GRADE                                                          NUMBER
```

step 2：查询 v01 中的全部数据；

```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
DEPT                           TABLE
EMP                            TABLE
BONUS                          TABLE
SALGRADE                       TABLE
E01                            TABLE
V01                            VIEW

SQL> select * from v01;

ENAME      DNAME               GRADE
---------- -------------- ----------
SMITH      RESEARCH                1
JAMES      SALES                   1
ADAMS      RESEARCH                1
WARD       SALES                   2
MARTIN     SALES                   2
MILLER     ACCOUNTING              2
TURNER     SALES                   3
ALLEN      SALES                   3
CLARK      ACCOUNTING              4
BLAKE      SALES                   4
JONES      RESEARCH                4
SCOTT      RESEARCH                4
FORD       RESEARCH                4
KING       ACCOUNTING              5

14 rows selected.
```

注意：视图只是观看基表数据的一个窗口，里面不包含真正的数据，真正的数据依然在基表中；
如果基表的数据发生变化，那么通过视图查看的时候也会看到修改后的结果；

2.前提条件

提示：如果用户想创建视图，要有 create view 的权限；
视图的创建过程与表的创建过程相似；

2.1 授权(create view)，并创建视图；

step 1：由 scott 用户创建视图 v02，定义参照 emp，数据可查看；

```
SQL> create view v02
  2  as select * from emp;
create view v02
            *
ERROR at line 1:
ORA-01031: insufficient privileges
```

step 2：由 sys 用户授予 scott 用户 create view 的权限；

```
SQL> conn / as sysdba
Connected.
SQL> show user
USER is "SYS"
SQL> grant create view to scott;

Grant succeeded.
```

step 3：scott 用户重新执行创建视图的命令；

```
SQL> conn scott/tiger
Connected.
SQL> create view v02
  2  as select * from emp;

View created.
```

step 4：查询 v02 中的数据；

```
SQL> select * from v02;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10

14 rows selected.
```

注意：可以通过视图对基表的数据进行 DML 操作，但是视图中必须有基表的关键字段(例：empno)

```
SQL> update v02 set sal = 0 where deptno = 10;

3 rows updated.

SQL> rollback;

Rollback complete.
```

如果 v02 的定义换成以下这样，则对 v02 的 DML 操作就有问题；

例 1：

step 1：修改 v02 视图的定义；

提示：对于视图定义的修改，不能通过 alter view 命令；使用 create or replace 命令；

```
SQL> create or replace view v02
  2  as select * from emp
  3  with read only;

View created.
```

step 2：测试对 v02 的 DML 操作；

提示：由于视图被定义为只读，所以 DML 操作受到了限制；

```
SQL> insert into v02(empno,ename,sal,deptno)
  2  values(1234,'haha',1000,10);
insert into v02(empno,ename,sal,deptno)
                *
ERROR at line 1:
ORA-01733: virtual column not allowed here


SQL> update v02 set sal = 2000 where empno = 7788;
update v02 set sal = 2000 where empno = 7788
               *
ERROR at line 1:
ORA-01733: virtual column not allowed here


SQL> delete v02 where ename = 'SCOTT';
delete v02 where ename = 'SCOTT'
       *
ERROR at line 1:
ORA-01752: cannot delete from view without exactly one key-preserved table
```

例 2：

step 1：修改 v02 视图的定义；

```
SQL> create or replace view v02
  2  as select ename,sal,deptno from emp;

View created.

SQL> desc v02
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------------
 ENAME                                                          VARCHAR2(10)
 SAL                                                            NUMBER(7,2)
 DEPTNO                                                         NUMBER(2)
```

step 2：测试对 v02 的 DML 操作；

```
SQL> insert into v02 values('haha',1000,10);
insert into v02 values('haha',1000,10)
*
ERROR at line 1:
ORA-01400: cannot insert NULL into ("SCOTT"."EMP"."EMPNO")


SQL> update v02 set ename = 'hehe' where ename = 'SCOTT';

1 row updated.

SQL> delete v02 where ename = 'CLARK';

1 row deleted.

SQL> rollback;

Rollback complete.
```

注意：empno 是基表中的主键所定义的字段，而视图中却没有定义，所以在通过视图对基表进行 insert 的时候，就无法对 empno 字段进行赋值，因此会报错！

3.一些小测试

提示：如果视图中包含了以下 a)、b)、c)时，不能添加、更新、删除基表的数据；包含 d)时，不能添加、更新基表的数据；包含 e)时，不能添加基表的数据；

a) 组函数

step 1：创建视图；

```
SQL> create view v03
  2  as select sum(sal) s,avg(sal) a from emp;

View created.

SQL> desc v03
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------------
 S                                                              NUMBER
 A                                                              NUMBER

SQL> select * from v03;

         S          A
---------- ----------
     29025 2073.21429
```

注意：在组函数之后要定义列的别名，否则报错！

```
SQL> create view v03
  2  as select sum(sal),avg(sal) from emp;
as select sum(sal),avg(sal) from emp
          *
ERROR at line 2:
ORA-00998: must name this expression with a column alias
```

step 2：测试；

```
SQL> delete v03 where s < 30000;
delete v03 where s < 30000
       *
ERROR at line 1:
ORA-01732: data manipulation operation not legal on this view
```

b) group by 子句

step 1：创建视图；

```
SQL> create view v04
  2  as select deptno,sum(sal) s,count(*) c,avg(sal) a
  3     from emp
  4     group by deptno
  5     order by deptno;

View created.

SQL> desc v04
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------------
 DEPTNO                                                         NUMBER(2)
 S                                                              NUMBER
 C                                                              NUMBER
 A                                                              NUMBER
```

step 2：测试；

```
SQL> select * from v04;

    DEPTNO          S          C          A
---------- ---------- ---------- ----------
        10       8750          3 2916.66667
        20      10875          5       2175
        30       9400          6 1566.66667

SQL> delete v04 where deptno = 10;
delete v04 where deptno = 10
       *
ERROR at line 1:
ORA-01732: data manipulation operation not legal on this view
```

c) distinct 关键字

step 1：创建视图；

```
SQL> create view v05
  2  as select distinct deptno from emp;

View created.

SQL> desc v05
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------------
 DEPTNO                                                         NUMBER(2)
```

step 2：测试；

```
SQL> select * from v05;

    DEPTNO
----------
        30
        20
        10

SQL> delete v05 where deptno = 10;
delete v05 where deptno = 10
       *
ERROR at line 1:
ORA-01732: data manipulation operation not legal on this view
```

d)视图中包含通过基表中的字段表达式的来的信息，基表的数据不能更新或者添加；

step 1：创建视图；

```
SQL> create or replace view v06
  2  as select empno,ename,sal * 12 + nvl(comm,0) as haha from emp;

View created.

SQL> desc v06
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------------
 EMPNO                                                 NOT NULL NUMBER(4)
 ENAME                                                          VARCHAR2(10)
 HAHA                                                           NUMBER
```

step 2：测试；

```
SQL> update v06 set haha = 12345 where empno = 7788;
update v06 set haha = 12345 where empno = 7788
                *
ERROR at line 1:
ORA-01733: virtual column not allowed here
```

e)视图中不包含基表中被定义为 not null 的列；

step 1：创建测试表；

```
SQL> create table t71(id number(2),name varchar2(20) not null,info varchar2(20));

Table created.

SQL> desc t71
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------------
 ID                                                             NUMBER(2)
 NAME                                                  NOT NULL VARCHAR2(20)
 INFO                                                           VARCHAR2(20)
```

step 2：创建视图；

```
SQL> create view v07
  2  as select id,info from t71;

View created.

SQL> desc v07
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------------
 ID                                                             NUMBER(2)
 INFO                                                           VARCHAR2(20)
```

step 3：测试；

```
SQL> insert into v07 values(1,'info 01');
insert into v07 values(1,'info 01')
*
ERROR at line 1:
ORA-01400: cannot insert NULL into ("SCOTT"."T71"."NAME")
```

4.with check option 子句

提示：满足条件的数据才能通过视图看到，其他的看不到；

类型一：

step 1：创建视图时在子查询中添加 with check option 子句；

```
SQL> create or replace view v08
  2  as select * from emp
  3     where deptno = 10 with check option;

View created.
```

step 2：查看视图中能看到的数据；

```
SQL> select * from v08;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10
```

step 3：测试；

step 3.1：不能通过视图修改使用 with check option 限定的字段的值；

```
SQL> update v08 set deptno = 20;
update v08 set deptno = 20
       *
ERROR at line 1:
ORA-01402: view WITH CHECK OPTION where-clause violation
```

step 3.2：不能通过视图修改不属于 10 部门的数据；

```
SQL> update v08 set ename = 'itaa' where deptno = 20;

0 rows updated.

SQL> update v08 set ename = 'itaa' where empno = 7788;

0 rows updated.
```

step 3.3：不能通过视图向基表中插入 10 部门以外的数据；

```
SQL> insert into v08(empno,ename,deptno) values(1001,'itaa',10);

1 row created.

SQL> insert into v08(empno,ename,deptno) values(1001,'itaa',15);
insert into v08(empno,ename,deptno) values(1001,'itaa',15)
            *
ERROR at line 1:
ORA-01402: view WITH CHECK OPTION where-clause violation
```

5.删除视图

step 1：查询 scott 用户下的对象；

```
SQL> select * from tab
  2  order by tabtype;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
EMP                            TABLE
T71                            TABLE
E01                            TABLE
DEPT                           TABLE
SALGRADE                       TABLE
BONUS                          TABLE
V07                            VIEW
V06                            VIEW
V05                            VIEW
V04                            VIEW
V03                            VIEW
V08                            VIEW
V02                            VIEW
V01                            VIEW

14 rows selected.
```

step 2：生成删除视图的脚本；

```
SQL> select 'drop view '||tname||';'
  2  from tab where tname like 'V%';

'DROPVIEW'||TNAME||';'
----------------------------------------
drop view V02;
drop view V01;
drop view V03;
drop view V04;
drop view V05;
drop view V06;
drop view V07;
drop view V08;

8 rows selected.
```

step 3：删除视图；(根据上述得到的信息生成脚本；)

注意：在 drop 视图时，实际上是将视图的定义删掉，因为视图中没有数据，所以不存在空间的释放问题；

6.Top-N 分析

提示：先查看 emp 表中的员工的信息；

```
SQL> select * from emp;

     EMPNO ENAME      JOB              MGR HIREDATE          SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7788 hehe       ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10

14 rows selected.
```

查询 emp 中工资最高的前 5 名员工的信息；

```
SQL> select * from (
  2      select * from emp order by sal desc)
  3  where rownum <= 5;

     EMPNO ENAME      JOB              MGR HIREDATE          SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7788 hehe       ANALYST         7566 19-APR-87       3000                    20
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
```

如果想查询工资最少的后 5 名员工的信息呢？

```
SQL> select * from (
  2      select * from emp order by sal)
  3  where rownum <= 5;

     EMPNO ENAME      JOB              MGR HIREDATE          SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
```

重点：如果想查询工资(按工资降序排列)位于第 6 名~第 10 名员工的信息呢？

测试：如果你上面的查询写成这样，那就错了！(注意 rownum 只能使用<或者<=)

```
SQL> select * from (
  2      select * from emp order by sal desc)
  3  where rownum between 6 and 10;

no rows selected
```

注意：要记得 between…and…解析成什么！

例：

step 1：创建测试表 t72，定义参照 emp，数据保留；

```
SQL> show user
USER is "SCOTT"
SQL> create table t72
  2  as select * from emp;

Table created.
```

step 2：添加测试数据，查看结果；

```
SQL> insert into t72
  2  select empno+1,ename||'x',job,mgr+1,hiredate,sal+1,nvl(comm,0)+1,deptno
  3  from emp;

14 rows created.

SQL> commit;

Commit complete.
```

```
SQL> set linesize 100
SQL> set pagesize 50
SQL> select * from t72;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	hehe	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10
7370	SMITHx	CLERK	7903	17-DEC-80	801	1	20
7500	ALLENx	SALESMAN	7699	20-FEB-81	1601	301	30
7522	WARDx	SALESMAN	7699	22-FEB-81	1251	501	30
7567	JONESx	MANAGER	7840	02-APR-81	2976	1	20
7655	MARTINx	SALESMAN	7699	28-SEP-81	1251	1401	30
7699	BLAKEx	MANAGER	7840	01-MAY-81	2851	1	30
7783	CLARKx	MANAGER	7840	09-JUN-81	2451	1	10
7789	hehex	ANALYST	7567	19-APR-87	3001	1	20
7840	KINGx	PRESIDENT		17-NOV-81	5001	1	10
7845	TURNERx	SALESMAN	7699	08-SEP-81	1501	1	30
7877	ADAMSx	CLERK	7789	23-MAY-87	1101	1	20
7901	JAMESx	CLERK	7699	03-DEC-81	951	1	30
7903	FORDx	ANALYST	7567	03-DEC-81	3001	1	20
7935	MILLERx	CLERK	7783	23-JAN-82	1301	1	10

step 3：实现分页的效果；

提示：定义分页相关变量(页码：pagenum，每页显示记录数：pagerecord)；

函数：row_number()，表示根据提供的字段信息进行自定义行号生成；

以下是查看第 1 页，即第 1~5 条信息；

```
SQL> define pagenum = 1;
SQL> define pagerecord = 5;
SQL> select * from
  2   (select empno,ename,sal,row_number() over (order by sal desc) as sortid
  3    from t72)
  4  where sortid between &pagerecord * &pagenum - (&pagerecord - 1)
  5              and     &pagerecord * &pagenum;
old   4: where sortid between &pagerecord * &pagenum - (&pagerecord - 1)
new   4: where sortid between 5 * 1 - (5 - 1)
old   5:              and     &pagerecord * &pagenum
new   5:              and     5 * 1

     EMPNO ENAME             SAL     SORTID
---------- ---------- ---------- ----------
      7840 KINGx            5001          1
      7839 KING             5000          2
      7789 hehex            3001          3
      7903 FORDx            3001          4
      7788 hehe             3000          5
```

如果想查看第 3 页，即第 11~15 条信息；

```
SQL> define pagenum = 3;
SQL> /
old   4: where sortid between &pagerecord * &pagenum - (&pagerecord - 1)
new   4: where sortid between 5 * 3 - (5 - 1)
old   5:              and     &pagerecord * &pagenum
new   5:              and     5 * 3

     EMPNO ENAME             SAL     SORTID
---------- ---------- ---------- ----------
      7783 CLARKx           2451         11
      7782 CLARK            2450         12
      7500 ALLENx           1601         13
      7499 ALLEN            1600         14
      7845 TURNERx          1501         15
```

如果想查看总共有多少页？

```
SQL> select ceil(count(*)/&pagerecord) pagesum from t72;
old   1: select ceil(count(*)/&pagerecord) pagesum from t72
new   1: select ceil(count(*)/5) pagesum from t72

   PAGESUM
----------
         6
```

当然，也可以把上述 SQL 语句保存为脚本，执行的时候也会稍微便利一点；

实际工作中，如果想实现更加智能化的交互，会使用 PL/SQL 程序实现，此处不作介绍；

step 4：如果不使用 row_number()函数，使用默认的 rownum，该如何实现；

```
SQL> show user
USER is "SCOTT"
SQL> select * from (
  2    select t.*, rownum b from (
  3      select empno, ename, sal, rownum a from t72 order by sal desc) t )
  4  where b between &pagerecord * &pagenum - (&pagerecord - 1)
  5          and     &pagerecord * &pagenum;
```

注意：Oracle 自动生成的 rownum，如果通过 3 层子查询进行检索时，默认的属性会被覆盖；

小结：row_number()生成的行号与 rownum 之间的区别；

1.row_number()可以按照指定的字段，指定顺序，生成相应的排序号；

2.而 rownum 是按照数据进入表中的时间顺序，由 Orace 自定义的顺序号，可能会因为数据的更新、删除而自动变化；

```
SQL> select empno,ename,sal,rownum,row_number() over (order by sal desc) new_rownum
  2  from emp;

     EMPNO ENAME              SAL     ROWNUM NEW_ROWNUM
---------- ---------- ---------- ---------- ----------
      7839 KING             5000          9          1
      7902 FORD             3000         13          2
      7788 hehe             3000          8          3
      7566 JONES            2975          4          4
      7698 BLAKE            2850          6          5
      7782 CLARK            2450          7          6
      7499 ALLEN            1600          2          7
      7844 TURNER           1500         10          8
      7934 MILLER           1300         14          9
      7521 WARD             1250          3         10
      7654 MARTIN           1250          5         11
      7876 ADAMS            1100         11         12
      7900 JAMES             950         12         13
      7369 SMITH             800          1         14

14 rows selected.
```

【第十二章 其它的数据库对象】

1.序列(sequence)

1.1 定义；

```
CREATE SEQUENCE sequence
       [INCREMENT BY n]
       [START WITH n]
       [{MAXVALUE n | NOMAXVALUE}]
       [{MINVALUE n | NOMINVALUE}]
       [{CYCLE | NOCYCLE}]
       [{CACHE n | NOCACHE}];
```

指定序列的名称；
序列的增长值；可以为正数，可以为负数，只是表示的增长方向不同；默认1；
序列的开始值；默认为1；
序列的最大值；默认情况下无最大值(其实最大值为10^27)，最小为1；
序列的最小值；默认情况下无最小值(其实最小值为-10^26)，最大为-1；
序列产生的值是否循环；默认不循环；
序列是否预先产生一些值放入缓存，以提高效率；默认产生20个；

1.2 创建序列；

提示：需要有 create sequence 的权限；

step 1：创建 sq01，参数如下；

```
SQL> show user
USER is "SCOTT"
SQL> create sequence sq01
  2    increment by 10
  3    start with 2
  4    maxvalue 100
  5    nocache
  6    nocycle;

Sequence created.
```

step 2：序列有效值的取得；

提示：nextval，表示从指定序列中取得下一个有效值；

currval，表示从指定序列中取得当前有效值；

```
SQL> select sq01.nextval from dual;

   NEXTVAL
----------
         2

SQL> /

   NEXTVAL
----------
        12
```

```
SQL> select sq01.currval from dual;

   CURRVAL
----------
        12
```

但是，如果会话刚开始，或者在其他会话中初次使用此序列，currval 取得时会报错，提示需激活；

```
SQL> conn scott/tiger
Connected.
SQL> show user
USER is "SCOTT"
SQL> select sq01.currval from dual;
select sq01.currval from dual
                     *
ERROR at line 1:
ORA-08002: sequence SQ01.CURRVAL is not yet defined in this session
```

那该如何激活呢？很简单，先取一下 nextval，再取 currval 即可；

```
SQL> select sq01.nextval from dual;

   NEXTVAL
----------
        22

SQL> select sq01.currval from dual;

   CURRVAL
----------
        22
```

step 3：有效序列值的使用；

用途：作为列的主键值；作为列的指定填充值

step 3.1：创建测试表；

```
SQL> show user
USER is "SCOTT"
SQL> create table tb01(sid number(3),sname varchar2(20),dt date);

Table created.
```

step 3.2：设定主键；

```
SQL> alter table tb01 add constraint pk_tb0101 primary key(sid);

Table altered.
```

step 3.3：因为序列产生的值满足主键的前提条件(非空、唯一)，所以可以用来充当主键值；

```
SQL> insert into tb01
  2  values(sq01.nextval,'name'||sq01.nextval,sysdate);

1 row created.

SQL> /

1 row created.

SQL> /

1 row created.

SQL> commit;

Commit complete.
```

step 3.4：查看生成的数据；

```
SQL> select * from tb01;

       SID SNAME                DT
---------- -------------------- ---------
        32 name32               03-JUN-11
        42 name42               03-JUN-11
        52 name52               03-JUN-11
```

或者，作为更新值出现；

step 3.5：解释略；

```
SQL> update tb01 set sname = 'haha'||sq01.nextval;

3 rows updated.

SQL> commit;

Commit complete.
```

step 3.6：查看结果；

```
SQL> select * from tb01;

       SID SNAME                DT
---------- -------------------- ---------
        32 haha62               03-JUN-11
        42 haha72               03-JUN-11
        52 haha82               03-JUN-11
```

注意：序列自动产生的值不会重复，除非特殊设定(如 cycle)；

step 4：有效序列值的查看；(user_sequences)

```
SQL> select sq01.currval from dual;

   CURRVAL
----------
        82

SQL> set linesize 100
SQL> select * from user_sequences;

SEQUENCE_NAME                  MIN_VALUE  MAX_VALUE INCREMENT_BY C O CACHE_SIZE LAST_NUMBER
------------------------------ ---------- ---------- ------------ - - ---------- -----------
SQ01                                    1        100           10 N N          0          92
```

注意：其中 last_number 为 92，表示序列当前值为 82 的情况下，下一个有效的值为 92；

提示：上述 sq01 未设置 cache，如果多个用户使用此序列时，需要排队等待，效率会受到影响；因此，可根据业务的需求，适当设定 cache 的大小；

1.3 序列的修改；

例：

step 1：创建序列，cache 设定为 10；

```
SQL> create sequence sq02
  2    increment by 1
  3    start with 1
  4    nocycle
  5    cache 10;

Sequence created.
```

优点：预留 10 个序列值在 cache 中，提高用户取值的效率；但是在没有用完这 10 个序列值的前提下，系统不会预留下一组序列值；

缺点：系统异常中断时，cache 的数据会丢失，造成序列值丢失，致使取到的序列值不连续；

step 2：查看下一组有效序列值的第 1 个值是什么？

```
SQL> select sq02.nextval from dual;

   NEXTVAL
----------
         1

SQL> select * from user_sequences where sequence_name = 'SQ02';

SEQUENCE_NAME                  MIN_VALUE  MAX_VALUE INCREMENT_BY C O CACHE_SIZE LAST_NUMBER
------------------------------ ---------- ---------- ------------ - - ---------- -----------
SQ02                                    1 1.0000E+27            1 N N         10          11
```

注意：start with n，此值 n 只会在初次使用序列时有效，以后不会再用到此值；
如果想重新使用 start with 的值，需删除后重建才行，修改的话会报错；

```
SQL> alter sequence sq02
  2    increment by 1
  3    start with 20
  4    nocycle
  5    cache 5;
  start with 20
  *
ERROR at line 3:
ORA-02283: cannot alter starting sequence number
```

但是，其他的参数可以进行修改，修改之后的序列只会影响以后产生的序列值；

```
SQL> alter sequence sq02
  2    increment by 2
  3    nocycle
  4    cache 5;

Sequence altered.

SQL> select sq02.nextval from dual;

   NEXTVAL
----------
        12

SQL> select * from user_sequences where sequence_name = 'SQ02';

SEQUENCE_NAME                  MIN_VALUE  MAX_VALUE INCREMENT_BY C O CACHE_SIZE LAST_NUMBER
------------------------------ ---------- ---------- ------------ - - ---------- -----------
SQ02                                    1 1.0000E+27            2 N N          5          22
```

注意：1.为什么 nextval 为 12？因为序列修改后，之前的值废弃，从下一组有效值开始；
2.为什么 last_number 为 22？因为增量值为 2，且缓存中的 5 个值未用完，下一组有效值从 22 开始；

注意：如果新的 maxvalue 小于当前的序列值，则修改时报错；

```
SQL> alter sequence sq02
  2    maxvalue 10;
alter sequence sq02
*
ERROR at line 1:
ORA-04009: MAXVALUE cannot be made to be less than the current value
```

1.4 删除序列；

```
SQL> drop sequence sq01;

Sequence dropped.

SQL> drop sequence sq02;

Sequence dropped.
```

2.索引(index)

提示：1.在表中定义主键约束或者唯一性约束的时候，Oracle 会自动创建该字段上的索引；

2.用户也可自定义某些字段的索引；

3.索引创建完成有数据库自动使用，不需要特别指定；

2.1 创建索引；

step 1：创建测试表，添加主键(empno)；

```
SQL> show user
USER is "SCOTT"
SQL> create table tb02
  2  as select * from emp;

Table created.

SQL> alter table tb02 add constraint pk_tb0201 primary key(empno);

Table altered.
```

step 2：在 ename 上添加索引；

```
SQL> create index ind_haha on tb02(ename);

Index created.
```

step 3：查看索引的信息；

```
SQL> select INDEX_NAME,INDEX_TYPE,TABLE_NAME,UNIQUENESS from user_indexes
  2  where TABLE_NAME = 'TB02';

INDEX_NAME                     INDEX_TYPE                TABLE_NAME                     UNIQUENES
------------------------------ ------------------------- ------------------------------ ---------
PK_TB0201                      NORMAL                    TB02                           UNIQUE
IND_HAHA                       NORMAL                    TB02                           NONUNIQUE
```

step 4：查看表的大小和索引的大小；

```
SQL> col SEGMENT_NAME for a30;
SQL> col BYTES for a30;
SQL> select SEGMENT_NAME,BYTES/1024/1024||'m' from user_segments
  2  where SEGMENT_NAME in ('TB02','PK_TB0201','IND_HAHA');

SEGMENT_NAME                   BYTES/1024/1024||'M'
------------------------------ ----------------------------------------
TB02                           .0625m
PK_TB0201                      .0625m
IND_HAHA                       .0625m
```

小结：

优点：通过索引可以提高检索数据的效率；(就像通过书的目录查找相关信息；)

确点：索引也需要占用数据库的空间，且需要维护；(书的目录也属于书的一部分，书越厚，对应的目录也会越厚；书的内容经常修改，则目录也需要对应修改；)

2.2 修改索引；(一般情况下将索引直接删除重建即可)

step 1：删除索引；

```
SQL> drop index ind_haha;

Index dropped.
```

step 2：重建索引；

```
SQL> create index ind_hehe on tb02(ename,job,mgr);

Index created.
```

2.3 查询索引的相关信息；(user_indexes 和 user_ind_columns)

```
SQL> col COLUMN_NAME for a20;
SQL> select c.index_name,c.column_name,c.column_position,x.uniqueness
  2  from user_indexes x,user_ind_columns c
  3  where x.index_name = c.index_name
  4  and c.table_name = 'TB02';

INDEX_NAME                     COLUMN_NAME          COLUMN_POSITION UNIQUENES
------------------------------ -------------------- --------------- ---------
PK_TB0201                      EMPNO                              1 UNIQUE
IND_HEHE                       ENAME                              1 NONUNIQUE
IND_HEHE                       JOB                                2 NONUNIQUE
IND_HEHE                       MGR                                3 NONUNIQUE
```

注意：uniqueness 表示唯一性索引(主键自动生成)，还是非唯一性索引(手动生成，未添加唯一性约束)；

2.4 导致索引失效的原因；

1) 需要当心的 where 子句；

某些 SELECT 语句中的 WHERE 子句不使用索引。 这里有一些例子。

在下面的例子里，‘!=’ 将不使用索引。 记住，索引只能告诉你什么存在于表中，而不能告诉你什么不存在于表中。

不使用索引:

```
SELECT ACCOUNT_NAME
FROM TRANSACTION
WHERE AMOUNT !=0;
```

使用索引:

```
SELECT ACCOUNT_NAME
FROM TRANSACTION
WHERE AMOUNT >0;
```

下面的例子中，‘||’是字符连接函数。 就象其他函数那样，停用了索引。

不使用索引:

```
SELECT ACCOUNT_NAME,AMOUNT
FROM TRANSACTION
WHERE ACCOUNT_NAME||ACCOUNT_TYPE=’AMEXA’;
```

2) 避免在索引列上使用 not；

我们要避免在索引列上使用 NOT，NOT 会产生在和在索引列上使用函数相同的影响。 当 ORACLE”遇到”NOT,他就会停止使用索引转而执行全表扫描。

3) 避免在索引列上使用计算；

WHERE 子句中，如果索引列是函数的一部分。优化器将不使用索引而使用全表扫描。

低效：

```
SELECT … FROM   DEPT   WHERE SAL * 12 > 25000;
```

高效:

```
SELECT … FROM DEPT WHERE SAL > 25000/12;
```

4) 注意复合索引的使用顺序；

TABLE(KEY1, KEY2,KEY3,……)

使用索引

```
WHERE KEY1=****;
WHERE KEY1=*** AND KEY2=&&&&;
WHERE KEY1=*** AND KEY3=&&&&;
```

索引失效

```
WHERE KEY2=*****;
WHERE KEY2=**** AND KEY3=&&&&
```

5) 避免改变索引的类型；

当比较不同数据类型的数据时，ORACLE 自动对列进行简单的类型转换。

假设 EMPNO 是一个数值类型的索引列。

```
SELECT …
FROM EMP
WHERE EMPNO = ‘123’
```

实际上,经过 ORACLE 类型转换，语句转化为:

```
SELECT …
FROM EMP
WHERE EMPNO = TO_NUMBER( ‘123’ )
```

幸运的是,类型转换没有发生在索引列上,索引的用途没有被改变。

现在,假设 EMP_TYPE 是一个字符类型的索引列。

```
SELECT …
FROM EMP
WHERE EMP_TYPE = 123
```

这个语句被 ORACLE 转换为:

```
SELECT …
FROM EMP
WHERE TO_NUMBER(EMP_TYPE)=123
```

因为内部发生的类型转换，这个索引将不会被用到!

为了避免 ORACLE 对你的 SQL 进行隐式的类型转换，最好把类型转换用显式表现出来。

注意：索引的使用规范；

2.5 基于函数的索引；

step 1：创建测试表 te01，定义参照 emp，数据保留；

```
SQL> create table te01 as select * from emp;

Table created.

SQL> set linesize 120;
SQL> set pagesize 30
SQL> select * from te01;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10

14 rows selected.
```

step 2：如果不知道 scott 的名字大小写的情况下，查询 scott 相关信息；

提示：存在隐患!!

```
SQL> select * from te01 where lower(ename) = 'scott';

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
```

step 3：查看执行计划；

3.1：例 1

step 3.1.1：在 ename 上创建索引；

```
SQL> create index ind_te0102 on te01(ename);

Index created.
```

step 3.1.2：原始测试方法；

```
SQL> select * from te01 where ename = 'SCOTT';

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
```

step 3.1.3：生成执行计划；

```
SQL> explain plan
  2     set statement_id = 'info 2011001' into plan_table
  3     for select * from te01 where ename = 'SCOTT';

Explained.
```

step 3.1.4：查看执行计划；

```
SQL> col OPERATION for a20;
SQL> col OPTIONS for a20;
SQL> col OBJECT_NAME for a10;

SQL> SELECT LPAD(' ',2*(LEVEL-1))||operation operation, options, object_name, position
  2  FROM plan_table
  3  START WITH id = 0 AND statement_id = 'info 2011001'
  4  CONNECT BY PRIOR id = parent_id AND statement_id = 'info 2011001';

OPERATION            OPTIONS              OBJECT_NAM   POSITION
-------------------- -------------------- ---------- ----------
SELECT STATEMENT                                             2
  TABLE ACCESS       BY INDEX ROWID       TE01                1
    INDEX            RANGE SCAN           IND_TE0102          1
```

注意：此时，由 options 可知使用了索引进行查询；

如果上述的 sql 变换了写法，使用字符函数忽略了一下大小写敏感性；

step 4：重新生成执行计划；

```
SQL> explain plan
  2     set statement_id = 'info 2011002' into plan_table
  3     for select * from te01 where lower(ename) = 'scott';

Explained.
```

step 5：查看执行计划；

```
SQL> SELECT LPAD(' ',2*(LEVEL-1))||operation operation, options, object_name, position
  2  FROM plan_table
  3  START WITH id = 0 AND statement_id = 'info 2011002'
  4  CONNECT BY PRIOR id = parent_id AND statement_id = 'info 2011002';

OPERATION            OPTIONS              OBJECT_NAM   POSITION
-------------------- -------------------- ---------- ----------
SELECT STATEMENT                                             3
  TABLE ACCESS       FULL                 TE01                1
```

注意：此时 options 显示为 full，表示未使用索引，而是进行了全表扫描；

原因：使用函数编辑了存在索引的字段，导致索引失效；

那么，如何才能既忽略大小写敏感性，又使用索引呢？？　　----使用基于函数的索引----

step 6：重新创建索引；

```
SQL> drop index ind_te0102;

Index dropped.

SQL> create index ind_te0102 on te01(lower(ename));

Index created.
```

step 7：生成执行计划；

```
SQL> explain plan
  2     set statement_id = 'info 2011003' into plan_table
  3     for select * from te01 where lower(ename) = 'scott';

Explained.
```

step 8：查看执行计划；

```
SQL> SELECT LPAD(' ',2*(LEVEL-1))||operation operation, options, object_name, position
  2  FROM plan_table
  3  START WITH id = 0 AND statement_id = 'info 2011003'
  4  CONNECT BY PRIOR id = parent_id AND statement_id = 'info 2011003';

OPERATION            OPTIONS              OBJECT_NAM   POSITION
-------------------- -------------------- ---------- ----------
SELECT STATEMENT                                             2
  TABLE ACCESS       BY INDEX ROWID       TE01                1
    INDEX            RANGE SCAN           IND_TE0102          1
```

注意：由此，实现了既忽略大小写敏感性，又使用索引的效果；

小结：执行计划部分不属于此课程，了解即可；

2.6 删除索引；

```
SQL> drop index ind_te0102;

Index dropped.
```

注意：如果是因为主键或者唯一性约束而产生的索引，再删除相应约束或者删除表的时候，索引也会自动删除；

3.同义词(synonym)

提示：1.创建同义词需要有 create synonym 的权限；

2.一般用户创建的同义词为私有同义词，仅归自己使用，除非它授权给别人使用；

管理员 sys 可以给一般用户创建私有同义词，但一般不这么做；管理员 sys 一般会创建 public 同义词，进而可以让数据库中所有的用户进行使用；(但依然存在一些权限的问题)

3.对于 public 同义词，只能由管理员 sys 创建和删除，除非它授权给别人；

4.关于权限的问题，第 13 章详细介绍；

3.1 一般用户创建[私有]同义词；

step 1：创建测试表；

```
SQL> show user
USER is "SCOTT"
SQL> create table hjadfeffdhfjdjddr
  2  as select * from dept;

Table created.
```

step 2：使用测试表；

```
SQL> select * from hjadfeffdhfjdjddr;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

感慨：哎，表名写的乱七八糟，挺难记，稍不留神就错了！

step 3：创建同义词；

```
SQL> create synonym haha for hjadfeffdhfjdjddr;
create synonym haha for hjadfeffdhfjdjddr
*
ERROR at line 1:
ORA-01031: insufficient privileges


SQL> conn / as sysdba
Connected.
SQL> show user
USER is "SYS"
SQL> grant create synonym to scott;

Grant succeeded.

SQL> conn scott/tiger
Connected.
SQL> show user
USER is "SCOTT"
SQL> create synonym haha for hjadfeffdhfjdjddr;

Synonym created.
```

step 4：使用同义词；

```
SQL> select * from haha;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

激动：终于不用抄那么长的名字了，呵呵！感觉就像给*乱七八糟*的名字起了个好记的代号；

注意：其他一般用户不能使用此同义词，除非经过所有者或管理员授权；

```
SQL> conn hr/hr
ERROR:
ORA-28000: the account is locked


Warning: You are no longer connected to ORACLE.
SQL> conn / as sysdba
Connected.
SQL> alter user hr account unlock identified by hr;

User altered.

SQL> conn hr/hr
Connected.
SQL> select * from haha;
select * from haha
              *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL> select * from scott.haha;
select * from scott.haha
                    *
ERROR at line 1:
ORA-00942: table or view does not exist
```

注意：即使 hr 用户在访问 haha 的时候加上了 scott，依然不能查询；

原因：1. hr 没有访问同义词 haha 的权限；

2. 或者，hr 没有访问*乱七八糟*那张表的权限；

解决：可授予 hr 用户访问同义词 haha 或者表*乱七八糟*的权限；

```
SQL> conn scott/tiger
Connected.
SQL> grant select on haha to hr;

Grant succeeded.

SQL> conn hr/hr
Connected.
SQL> select * from scott.haha;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

step 5：查询 synonym 相关的信息；(user_synonyms)

```
SQL> show user
USER is "SCOTT"
SQL> select SYNONYM_NAME,TABLE_OWNER,TABLE_NAME from user_synonyms;

SYNONYM_NAME                   TABLE_OWNER                    TABLE_NAME
------------------------------ ------------------------------ ------------------------------
HAHA                           SCOTT                          HJADFEFFDHFJDJDDR
```

3.2 管理员 sys 创建[公有]同义词；

step 1：创建测试表；

```
SQL> show user
USER is "SYS"
SQL> create table scott.luanqibazao2
  2  as select * from scott.salgrade;

Table created.
```

step 2：使用测试表；

```
SQL> select * from scott.luanqibazao2;

     GRADE      LOSAL      HISAL
---------- ---------- ----------
         1        700       1200
         2       1201       1400
         3       1401       2000
         4       2001       3000
         5       3001       9999
```

step 3：创建同义词；

```
SQL> show user
USER is "SYS"
SQL> create public synonym hehe
  2  for scott.luanqibazao2;

Synonym created.
```

step 4：使用同义词；

```
SQL> conn scott/tiger
Connected.
SQL> select * from hehe;

     GRADE      LOSAL      HISAL
---------- ---------- ----------
         1        700       1200
         2       1201       1400
         3       1401       2000
         4       2001       3000
         5       3001       9999
```

注意：scott 用户可以访问，理所当然！

看看 hr 用户能不能呢？

```
SQL> conn hr/hr
Connected.
SQL> select * from hehe;
select * from hehe
              *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL> select * from scott.hehe;
select * from scott.hehe
                    *
ERROR at line 1:
ORA-00942: table or view does not exist
```

奇怪，管理员 sys 已经创建了公有的同义词了，为什么还不能访问呢？

原因：hr 用户没有访问 scott 用户下 salgrade 表的权限；

解决：由 scott 或者 sys 授权给 hr；（即把 hehe 或者 salgrade 的访问权限给 hr 即可）

```
SQL> show user
USER is "SCOTT"
SQL> grant select on salgrade to hr;

Grant succeeded.

SQL> conn hr/hr
Connected.
SQL> select * from hehe;

     GRADE      LOSAL      HISAL
---------- ---------- ----------
         1        700       1200
         2       1201       1400
         3       1401       2000
         4       2001       3000
         5       3001       9999
```

注意：同义词 hehe 前不用加用户名，因为 hehe 是公有的；

step 5：查询同义词相关的信息；（dba_synonyms）

```
SQL> show user
USER is "SYS"
SQL> col OWNER for a15;
SQL> col SYNONYM_NAME for a15;
SQL> col TABLE_OWNER for a15;
SQL> col TABLE_NAME for a15;
SQL> select OWNER,SYNONYM_NAME,TABLE_OWNER,TABLE_NAME
  2  from dba_synonyms
  3  where table_owner = 'SCOTT'
  4  and table_name = 'LUANQIBAZAO2';

OWNER           SYNONYM_NAME    TABLE_OWNER     TABLE_NAME
--------------- --------------- --------------- ---------------
PUBLIC          HEHE            SCOTT           LUANQIBAZAO2
```

3.3 一般用户删除属于自己的[私有]同义词；

```
SQL> conn scott/tiger
Connected.
SQL> drop synonym haha;

Synonym dropped.
```

3.4 管理员 sys 删除[共有]同义词；

```
SQL> conn scott/tiger
Connected.
SQL> drop public synonym hehe;
drop public synonym hehe
                    *
ERROR at line 1:
ORA-01031: insufficient privileges


SQL> conn / as sysdba
Connected.
SQL> drop public synonym hehe;

Synonym dropped.
```

【第十三章 控制用户访问】

1. 概念了解

提示：1. 权限：执行特殊 SQL 语句的权利；

2. 系统权限：用来访问数据库，执行改变数据库结构的一些操作权利；

3. 对象权限：用来操作数据对象内容的权利；

1.1 系统权限；

提示：数据库中有 100 多种系统权限，数据库管理员具有授予用户这些权限的能力；

注意：要根据用户的需求，分配适当的权限；

ADMINISTER ANY SQL TUNING SET	ALTER ANY TYPE	CREATE ANY INDEX	CREATE INDEXTYPE
ADMINISTER DATABASE TRIGGER	ALTER DATABASE	CREATE ANY INDEXTYPE	CREATE JOB
ADMINISTER RESOURCE MANAGER	ALTER PROFILE	CREATE ANY JOB	CREATE LIBRARY
ADMINISTER SQL TUNING SET	ALTER RESOURCE COST	CREATE ANY LIBRARY	CREATE MATERIALIZED VIEW
ADVISOR	ALTER ROLLBACK SEGMENT	CREATE ANY MATERIALIZED VIEW	CREATE OPERATOR
ALTER ANY CLUSTER	ALTER SESSION	CREATE ANY OPERATOR	CREATE PROCEDURE
ALTER ANY DIMENSION	ALTER SYSTEM	CREATE ANY OUTLINE	CREATE PROFILE
ALTER ANY EVALUATION CONTEXT	ALTER TABLESPACE	CREATE ANY PROCEDURE	CREATE PUBLIC DATABASE LINK
ALTER ANY INDEX	ALTER USER	CREATE ANY RULE	CREATE PUBLIC SYNONYM
ALTER ANY INDEXTYPE	ANALYZE ANY	CREATE ANY RULE SET	CREATE ROLE
ALTER ANY LIBRARY	ANALYZE ANY DICTIONARY	CREATE ANY SEQUENCE	CREATE ROLLBACK SEGMENT
ALTER ANY MATERIALIZED VIEW	AUDIT ANY	CREATE ANY SQL PROFILE	CREATE RULE
ALTER ANY OPERATOR	AUDIT SYSTEM	CREATE ANY SYNONYM	CREATE RULE SET
ALTER ANY OUTLINE	BACKUP ANY TABLE	CREATE ANY TABLE	CREATE SEQUENCE
ALTER ANY PROCEDURE	BECOME USER	CREATE ANY TRIGGER	CREATE SESSION
ALTER ANY ROLE	CHANGE NOTIFICATION	CREATE ANY TYPE	CREATE SYNONYM
ALTER ANY RULE	COMMENT ANY TABLE	CREATE ANY VIEW	CREATE TABLE
ALTER ANY RULE SET	CREATE ANY CLUSTER	CREATE CLUSTER	CREATE TABLESPACE
ALTER ANY SEQUENCE	CREATE ANY CONTEXT	CREATE DATABASE LINK	CREATE TRIGGER
ALTER ANY SQL PROFILE	CREATE ANY DIMENSION	CREATE DIMENSION	CREATE TYPE
ALTER ANY TABLE	CREATE ANY DIRECTORY	CREATE EVALUATION CONTEXT	CREATE USER
ALTER ANY TRIGGER	CREATE ANY EVALUATION CONTEXT	CREATE EXTERNAL JOB	CREATE VIEW

DEBUG ANY PROCEDURE	DROP ANY TABLE	EXEMPT IDENTITY POLICY	RESUMABLE
DEBUG CONNECT SESSION	DROP ANY TRIGGER	EXPORT FULL DATABASE	SELECT ANY DICTIONARY
DELETE ANY TABLE	DROP ANY TYPE	FLASHBACK ANY TABLE	SELECT ANY SEQUENCE
DEQUEUE ANY QUEUE	DROP ANY VIEW	FORCE ANY TRANSACTION	SELECT ANY TABLE
DROP ANY CLUSTER	DROP PROFILE	FORCE TRANSACTION	SELECT ANY TRANSACTION
DROP ANY CONTEXT	DROP PUBLIC DATABASE LINK	GLOBAL QUERY REWRITE	SYSDBA
DROP ANY DIMENSION	DROP PUBLIC SYNONYM	GRANT ANY OBJECT PRIVILEGE	SYSOPER
DROP ANY DIRECTORY	DROP ROLLBACK SEGMENT	GRANT ANY PRIVILEGE	UNDER ANY TABLE
DROP ANY EVALUATION CONTEXT	DROP TABLESPACE	GRANT ANY ROLE	UNDER ANY TYPE
DROP ANY INDEX	DROP USER	IMPORT FULL DATABASE	UNDER ANY VIEW
DROP ANY INDEXTYPE	ENQUEUE ANY QUEUE	INSERT ANY TABLE	UNLIMITED TABLESPACE
DROP ANY LIBRARY	EXECUTE ANY CLASS	LOCK ANY TABLE	UPDATE ANY TABLE
DROP ANY MATERIALIZED VIEW	EXECUTE ANY EVALUATION CONTEXT	MANAGE ANY FILE GROUP	
DROP ANY OPERATOR	EXECUTE ANY INDEXTYPE	MANAGE ANY QUEUE	
DROP ANY OUTLINE	EXECUTE ANY LIBRARY	MANAGE FILE GROUP	
DROP ANY PROCEDURE	EXECUTE ANY OPERATOR	MANAGE SCHEDULER	
DROP ANY ROLE	EXECUTE ANY PROCEDURE	MANAGE TABLESPACE	
DROP ANY RULE	EXECUTE ANY PROGRAM	MERGE ANY VIEW	
DROP ANY RULE SET	EXECUTE ANY RULE	ON COMMIT REFRESH	
DROP ANY SEQUENCE	EXECUTE ANY RULE SET	QUERY REWRITE	
DROP ANY SQL PROFILE	EXECUTE ANY TYPE	READ ANY FILE GROUP	
DROP ANY SYNONYM	EXEMPT ACCESS POLICY	RESTRICTED SESSION	

1.2 对象权限；

提示：数据库中有 10 多种对象权限，针对不同的对象，其对象权限也不相同；数据库管理员具有授予用户这些权限的能力(不过，对象权限一般由对象的所有者授予)；

权限	SELECT	INSERT	UPDATE	DELETE	ALTER	INDEX	EXECUTE	READ	REFERENCE
Directory	no	no	no	no	no	no	no	yes	no
function	no	no	no	no	no	no	yes	no	no
procedure	no	no	no	no	no	no	yes	no	no
package	no	no	no	no	no	no	yes	no	no
DB Object	no	no	no	no	no	no	yes	no	no
Library	no	no	no	no	no	no	yes	no	no
Operation	no	no	no	no	no	no	yes	no	no
Sequence	no	no	no	no	yes	no	no	no	no
Table	yes	yes	yes	yes	yes	yes	no	no	yes
Type	no	no	no	no	no	no	yes	no	no
View	yes	yes	yes	yes	no	no	no	no	no

2.创建和管理用户

提示：需有 create user 权限，此操作一般由管理员 sys 完成，或者管理员 sys 授权给其他管理员完成；

step 1：创建新用户 us01，密码 us01；

```
SQL> show user
USER is "SYS"
SQL> create user us01 identified by us01;

User created.
```

注意：新用户创建完毕后，没有任何权限；需根据用户的需求进行授予；

step 2：修改用户的密码；

```
SQL> alter user us01 identified by us00;

User altered.
```

注意：如果管理员的密码忘了，那怎么办？

提示：1.secureCRT 软件特殊，可不经过客户端，直接连接数据库服务器；(了解)

2.以下操作，仅供参考，工作中慎用；(命令环境：secureCRT)

```
SQL> conn sys/oracle as sysdba
Connected.
SQL> conn / as sysdba
Connected.
```

注意：1.实验环境中设定的 sys 用户的密码为 oracle，在 secureCRT 下也可以不写用户名、密码；

2.管理员 sys 登陆的时候，必须使用 sysdba 的身份；

如果 sys 用户的密码忘记，重置即可；

```
SQL> alter user sys identified by haha;

User altered.

SQL> conn sys/haha as sysdba
Connected.
SQL> conn / as sysdba
Connected.
```

注意：用户的密码被修改之后，下次登陆时才会生效；

疑问：以上发出修改 sys 用户密码命令的执行者是谁呢？自己？　　对，可以是自己，也可以是其他的管理员，或者……

继续，往下看，开始迷茫了吧！

a）匿名登陆；结果是 sys；

```
SQL> conn / as sysdba
Connected.
SQL> show user
USER is "SYS"
```

b）由 scott 用户登陆；结果也是 sys；

```
SQL> conn scott/tiger as sysdba
Connected.
SQL> show user
USER is "SYS"
```

c）由*乱七八糟*用户登陆；结果还是 sys；

```
SQL> conn hasyefjehuhjd/dyfejbdhgggdj as sysdba
Connected.
SQL> show user
USER is "SYS"
```

小结：如果用户通过 secureCRT 的命令行登陆数据库服务器，身份选择 sysdba 的话，那数据库的操作者就是 sys 用户；　但是，如果不通过 secureCRT 软件，而是由客户端进行连接访问，以上操作都是禁止的。

注意：工作中一般采取一些方式限定，因为这样不安全；

（但是 Oracle 不承认这样不安全，因为拉里埃里森[Oracle 的 CEO]解释：Oracle 作为操作系统上的一个软件，职责有限，如果操作系统无法保证用户的安全，随便让用户登陆，那这些用户进来之后，那就随便随便了……）

多说几句，控制用户访问的方式有很多，例：用户名密码认证、操作系统认证、外部中间件认证等等。

放心，工作中会有很多办法解决此问题的！

step 3：将用户进行锁定，解锁；

```
SQL> show user
USER is "SYS"
SQL> alter user scott account lock;

User altered.

SQL> conn scott/tiger
ERROR:
ORA-28000: the account is locked


Warning: You are no longer connected to ORACLE.
```

注意：解锁的时候，用户的密码会失效，需要重置密码；

```
SQL> conn / as sysdba
Connected.
SQL> show user
USER is "SYS"
SQL> alter user scott account unlock identified by lion;

User altered.

SQL> conn scott/lion
Connected.
SQL> show user
USER is "SCOTT"
```

step 4：删除用户；

```
SQL> conn / as sysdba
Connected.
SQL> show user
USER is "SYS"
SQL> drop user us01 cascade;

User dropped.
```

注意：1.删除用户时，需有 drop user 的权限；
2.cascade，表示将用户相关的对象关联清除；

3.权限的授予(grant)和回收(revoke)

提示：1.由管理员授予用户系统权限，由对象的所有者授予对象权限；
2.由管理员回收用户系统权限，由对象的所有者回收对象权限；

例：

step 1：创建新用户 us02，密码 us02，并制定使用的表空间，以及限额；

```
SQL> show user
USER is "SYS"
SQL> create user us02 identified by us02
  2  default tablespace users
  3  quota 20m on users;

User created.
```

step 2：由管理员授予 us02 用户 *create session*、*create table*、*create view* 的系统权限；

```
SQL> grant create session, create table, create view to us02;

Grant succeeded.
```

step 3：测试用户 us02 拥有的系统权限是否生效；

```
SQL> conn us02/us02
Connected.
SQL> create table t01(id number(4),name varchar2(20),dt date);

Table created.

SQL> create view v01
  2  as select * from t01;

View created.
```

step 4：由对象所有者 scott 授予 us02 用户 *访问*和*更新* dept 表的权限；

```
SQL> show user
USER is "SCOTT"
SQL> grant select, update on dept to us02;

Grant succeeded.
```

step 5：测试用户 us02 拥有的对象权限是否生效；

```
SQL> show user
USER is "US02"
SQL> select * from scott.dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

SQL> update scott.dept set loc = 'tj itaa';

4 rows updated.

SQL> rollback;

Rollback complete.
```

step 6：由管理员回收 us02 用户 *create view* 的系统权限

```
SQL> show user
USER is "SYS"
SQL> revoke create view from us02;

Revoke succeeded.
```

step 7：测试用户 us02 是否失去此系统权限；

```
SQL> show user
USER is "US02"
SQL> create view v02
  2  as select * from t01;
create view v02
            *
ERROR at line 1:
ORA-01031: insufficient privileges
```

step 8：由对象所有者 scott 回收 us02 用户*更新* dept 表的权限；

```
SQL> show user
USER is "SCOTT"
SQL> revoke update on dept from us02;

Revoke succeeded.
```

step 9：测试用户 us02 是否失去此对象权限；

```
SQL> show user
USER is "US02"
SQL> update scott.dept set loc = 'tj itaa';
update scott.dept set loc = 'tj itaa'
            *
ERROR at line 1:
ORA-01031: insufficient privileges
```

4.角色的创建和管理

提示：1.单独授予用户相应的权限操作比较繁琐，而且容易出错；

2.可将用户进行分组，不同角色的人拥有的数据库的角色也不同，而不同的角色拥有的权限也不同；

例：

step 1：创建用户 pg01，密码 pg01；用户 ld01，密码 ld01；并指定使用表空间以及限额；

```
SQL> conn / as sysdba
Connected.
SQL> show user
USER is "SYS"
SQL> create user pg01 identified by pg01
  2  default tablespace users
  3  quota 10m on users;

User created.

SQL> create user ld01 identified by ld01
  2  default tablespace users
  3  quota 10m on users;

User created.
```

step 2：由管理员创建角色 programer，授予 create session、create table、select 用户 scott 下 dept 的权限；

由管理员创建角色 leader，授予 create session、create table、create view、select 和 update 用户 scott 下 dept 的权限；

```
SQL> create role programer;

Role created.

SQL> create role leader;

Role created.
```

```
SQL> grant create session, create table to programer,leader;

Grant succeeded.

SQL> grant create view to leader;

Grant succeeded.

SQL> grant select on scott.dept to programer,leader;

Grant succeeded.

SQL> grant update on scott.dept to leader;

Grant succeeded.
```

step 3：将角色 programer 授予用户 pg01，将角色 leader 授予用户 ld01；

```
SQL> grant programer to pg01;

Grant succeeded.

SQL> grant leader to ld01;

Grant succeeded.
```

step 4：测试用户 pg01、ld01 是否具有对应的权限；

```
SQL> conn pg01/pg01
Connected.
SQL> create table tb01(id number(2),dt date);

Table created.

SQL> select * from scott.dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

SQL> conn ld01/ld01
Connected.
SQL> create table tc01(id number(2),info varchar2(20));

Table created.

SQL> create view tv01 as select * from tc01;

View created.

SQL> select * from scott.dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

SQL> update scott.dept set loc='tj';

4 rows updated.

SQL> rollback;

Rollback complete.
```

注意：以上在授予用户权限时，将权限进行了封装，将角色授予用户，比较方便；假如又有新的用户进来，只要将角色授予即可，不必将单个的权限来回授予；

如果，再有下面这样的操作，那可就麻烦了；(工作中要谨慎细心)

step 5：将 create table 权限单独授予用户 pg01 和 ld01；

```
SQL> show user
USER is "SYS"
SQL> grant create table to pg01,ld01;

Grant succeeded.
```

注意：用户 pg01 和 ld01 已经具有了相应的权限，再次授予会造成重复；重复的权限并不完全合并；

step 6：即使由管理员回收了用户 pg01 和 ld01 的 create table 权限，你会发现什么？

提示：你可以多回收几次，但是没有用，会报错(没有对应权限可回收)！但是，create table 的权限依然残留!!

```
SQL> show user
USER is "SYS"
SQL> revoke create table from pg01,ld01;

Revoke succeeded.

SQL> revoke create table from pg01,ld01;
revoke create table from pg01,ld01
*
ERROR at line 1:
ORA-01952: system privileges not granted to 'PG01'
```

虽然提示用户都没有了 create table 的权限，但是测试一下用户能否建表！？

```
SQL> show user
USER is "PG01"
SQL> create table tb02(id date);

Table created.

SQL> show user
USER is "LD01"
SQL> create table tc02(id date);

Table created.
```

那该怎么彻底回收掉用户 pg01 和 ld01 中 create table 的权限呢？

step 7：将用户 pg01 和 ld01 的角色回收；(这一步可以省略)

```
SQL> show user
USER is "SYS"
SQL> revoke programer from pg01;

Revoke succeeded.

SQL> revoke leader from ld01;

Revoke succeeded.
```

step 8：从角色中回收 create table 权限；

```
SQL> show user
USER is "SYS"
SQL> revoke create table from programer,leader;

Revoke succeeded.
```

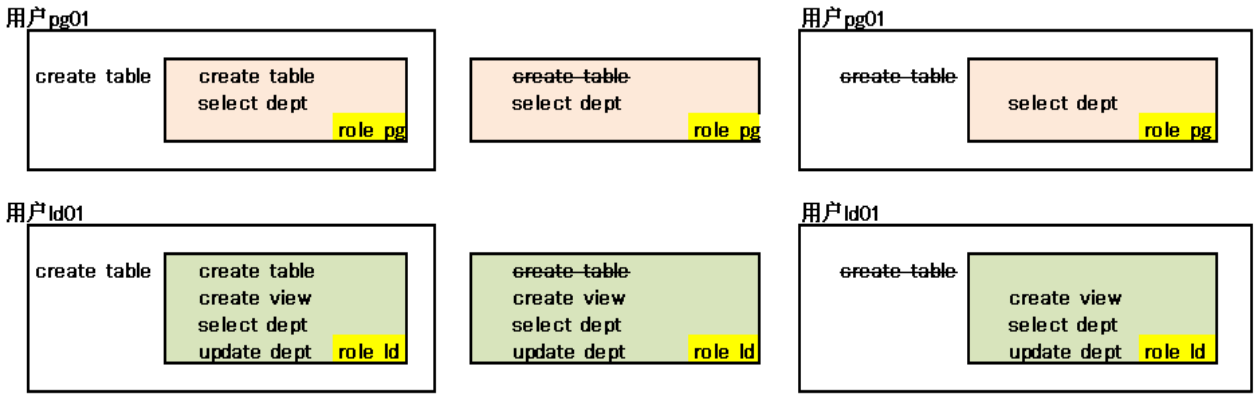
step 9：再将角色对应授予用户 pg01 和 ld01；(与 step 7 关联，也可以省略)

```
SQL> show user
USER is "SYS"
SQL> grant programer to pg01;

Grant succeeded.

SQL> grant leader to ld01;

Grant succeeded.
```

step 10：测试一下用户 pg01 和 ld01 是否还具有 create table 的权限？ （应该是没有了吧！）

```
SQL> show user
USER is "PG01"
SQL> create table tb03(id date);
create table tb03(id date)
*
ERROR at line 1:
ORA-01031: insufficient privileges
SQL> show user
USER is "LD01"
SQL> create table tc03(id date);
create table tc03(id date)
*
ERROR at line 1:
ORA-01031: insufficient privileges
```

小结：



1. 在授予用户角色，单个权限时，之间重复的部分并不合并；
2. 在回收用户的单个权限时，只是将外层的权限回收回来，封装在角色中权限无法直接从用户回收；
3. 在角色中封装的权限，需先从角色中回收；
4. 授予用户角色后，用户需重新连接数据库才能生效；

5. 查询用户的权限信息

例：dba_role_privs：可查询授予用户的角色信息；
dba_sys_privs：可查询授予用户的系统权限信息；
dba_tab_privs_made：可查询授予用户的对象权限信息；

step 1：查询用户 pg01 和 ld01 拥有的角色；

```
SQL> select * from dba_role_privs
  2  where grantee in ('PG01','LD01');

GRANTEE                        GRANTED_ROLE                   ADM DEF
------------------------------ ------------------------------ --- ---
PG01                           PROGRAMER                      NO  YES
LD01                           LEADER                         NO  YES
```

step 2：查询用户 pg01 和 ld01 拥有的系统权限；

```
SQL> show user
USER is "SYS"
SQL> select * from dba_sys_privs
  2  where grantee in ('PROGRAMER','LEADER');

GRANTEE                        PRIVILEGE                                ADM
------------------------------ ---------------------------------------- ---
PROGRAMER                      CREATE SESSION                           NO
LEADER                         CREATE VIEW                              NO
LEADER                         CREATE SESSION                           NO
```

注意：查询的时候 grantee 要写角色的名字；

step 3：查询用户 pg01 和 ld01 拥有的对象权限；

```
SQL> show user
USER is "SCOTT"
SQL> select * from user_tab_privs_made
  2  where grantee in ('PROGRAMER','LEADER');

GRANTEE                        TABLE_NAME
------------------------------ ------------------------------
GRANTOR                        PRIVILEGE                                GRA HIE
------------------------------ ---------------------------------------- --- ---
LEADER                         DEPT
SCOTT                          UPDATE                                   NO  NO

LEADER                         DEPT
SCOTT                          SELECT                                   NO  NO

PROGRAMER                      DEPT
SCOTT                          SELECT                                   NO  NO
```

注意：在 scott 用户下查询；

【第十四章 复习】

【第十五章 集合运算】

1.联合运算(union、union all)

1.1 union：从多表中返回所有行，但是除去任何重复的行；

step 1：创建测试表 a01，a02，定义参照 emp；

```
SQL> show user
USER is "SCOTT"
SQL> create table a01
  2  as select empno,ename,job,sal,deptno from emp where 1=2;

Table created.

SQL> create table a02(no,name,work,kyuro,bumen)
  2  as select empno,ename,job,sal,deptno from emp where 1=2;

Table created.
```

step 2：添加测试数据；

```
SQL> insert into a01
  2  select empno,ename,job,sal,deptno from emp where rownum <= 5;

5 rows created.

SQL> insert into a01 values(null,null,null,null,null);

1 row created.

SQL> commit;

Commit complete.

SQL> insert into a02
  2  select * from a01 where rownum <= 2;

2 rows created.

SQL> insert into a02
  2  select empno+2,ename||'k',job,sal+1000,deptno from a01
  3  where empno is not null and rownum <= 3;

3 rows created.

SQL> insert into a02 values(null,null,null,null,null);

1 row created.

SQL> /

1 row created.

SQL> commit;

Commit complete.
```

step 3：查看各表的数据；

```
SQL> select * from a01;

     EMPNO ENAME      JOB              SAL     DEPTNO
---------- ---------- --------- ---------- ----------
      7369 SMITH      CLERK             800         20
      7499 ALLEN      SALESMAN         1600         30
      7521 WARD       SALESMAN         1250         30
      7566 JONES      MANAGER          2975         20
      7654 MARTIN     SALESMAN         1250         30


6 rows selected.

SQL> select * from a02;

        NO NAME       WORK            KYURO      BUMEN
---------- ---------- --------- ---------- ----------
      7369 SMITH      CLERK             800         20
      7499 ALLEN      SALESMAN         1600         30
      7371 SMITHk     CLERK            1800         20
      7501 ALLENk     SALESMAN         2600         30
      7523 WARDk      SALESMAN         2250         30



7 rows selected.
```

step 4：使用联合(union)运算；

```
SQL> select empno,ename,sal from a01
  2  union
  3  select no,name,kyuro from a02;

     EMPNO ENAME             SAL
---------- ---------- ----------
      7369 SMITH             800
      7371 SMITHk           1800
      7499 ALLEN            1600
      7501 ALLENk           2600
      7521 WARD             1250
      7523 WARDk            2250
      7566 JONES            2975
      7654 MARTIN           1250


9 rows selected.
```

小结：1.union 运算过程中不忽略 null 记录；
2.两块数据集取并集，重复记录保留 1 份；
3.按照第 1 个 select 的字段进行显示；
4.结果集按照第 1 个 select 的第 1 列进行升序排列(默认)；
5.两个查询中，字段数要相同，对应字段的类型最好相同，字段的名字不必要相同；

1.2 union all：从多表中返回所有行，保留重复行；

提示：1.union all 运算过程中不忽略 null 记录；
2.两块数据集取并集，重复记录全部保留；
3.按照第 1 个 select 的字段进行显示；
4.结果集不进行排列(默认)；
5.两个查询中，字段数要相同，对应字段的类型最好相同，字段的名字不必要相同；
6.可将结果集进行 distinct，以便排除重复记录；
7.工作中业务数据一般情况下不会重复，多采用 union all 计算；

step 5：使用全联合(union all)运算

```
SQL> set pagesize 30
SQL> select empno,ename,sal from a01
  2  union all
  3  select no,name,kyuro from a02;

     EMPNO ENAME             SAL
---------- ---------- ----------
      7369 SMITH             800
      7499 ALLEN            1600
      7521 WARD             1250
      7566 JONES            2975
      7654 MARTIN           1250

      7369 SMITH             800
      7499 ALLEN            1600
      7371 SMITHk           1800
      7501 ALLENk           2600
      7523 WARDk            2250


13 rows selected.
```

2.交集运算(intersect)----[返回多个查询中重复的行]

提示：1. intersect 运算过程中不忽略 null 记录；

2. 按照第 1 个 select 的字段进行显示；

3. 两个查询中，字段数要相同，对应字段的类型最好相同，字段的名字不必要相同；

4. 两个查询颠倒位置不改变结果集；

step 1：继续使用 a01 和 a02 的数据进行测试；

```
SQL> select empno,ename,sal from a01
  2  intersect
  3  select no,name,kyuro from a02;

     EMPNO ENAME             SAL
---------- ---------- ----------
      7369 SMITH             800
      7499 ALLEN            1600


SQL> select count(*) from
  2  (select empno,ename,sal from a01
  3   intersect
  4   select no,name,kyuro from a02);

  COUNT(*)
----------
         3
```

3.差集(minus)----[返回在第 1 个查询中而不在第 2 个查询中的行]

step 1：继续使用 a01 和 a02 的数据进行测试；

```
SQL> select empno,ename,sal from a01
  2  minus
  3  select no,name,kyuro from a02;

     EMPNO ENAME             SAL
---------- ---------- ----------
      7521 WARD             1250
      7566 JONES            2975
      7654 MARTIN           1250
```

注意：结果只有 3 行；

step 2：继续使用 a01 和 a02 的数据进行测试；

```
SQL> select no,name,kyuro from a02
  2  minus
  3  select empno,ename,sal from a01;

        NO NAME            KYURO
---------- ---------- ----------
      7371 SMITHk           1800
      7501 ALLENk           2600
      7523 WARDk            2250
```

注意：结果只有 3 行；

4. 匹配 select 语句

特例：如果想显示 emp 表中员工的部门号、部门地址、入职日期，该怎么做？　应该涉及到 dept 表；

提示：emp 和 dept 表的结构无法对应匹配，如何处理？　对，类型转换，如下：

step 1：测试一下，估计得解释一下；

```
SQL> show user
USER is "SCOTT"
SQL> select deptno,to_char(null) location,hiredate
  2  from emp
  3  union
  4  select deptno,loc,to_date(null)
  5  from dept;

    DEPTNO LOCATION      HIREDATE
---------- ------------- ---------
        10 NEW YORK
        10               09-JUN-81
        10               17-NOV-81
        10               23-JAN-82
        20 DALLAS
        20               17-DEC-80
        20               02-APR-81
        20               03-DEC-81
        20               19-APR-87
        20               23-MAY-87
        30 CHICAGO
        30               20-FEB-81
        30               22-FEB-81
        30               01-MAY-81
        30               08-SEP-81
        30               28-SEP-81
        30               03-DEC-81
        40 BOSTON

18 rows selected.
```

分析：1. 在 emp 表中没有 loc 的信息，可以类型转换后与 dept 的 loc 对应；同理 hiredate；

2. 为什么用 null？也不一定，一般用 null，用其他的也没关系，不过你得了解结果中值的意思；

3. 结果为什么是 18 行？呵呵，emp 表中 14 行，dept 表中 4 行；

4. 注意：结果中第 1 列值相同的情况下，按照第 2 列升序排列；依次类推；

5. 总结一下

1. 两个查询中，字段数要相同，对应字段的类型最好相同，字段的名字不必要相同；
2. order by 子句只能出现在整个语句的最后，对结果集起作用；
   order by 子句中用于排序的列来自于：第 1 个查询中的列名、别名、字段的位置；
3. 除了 union all，其他集合运算的结果集按照第 1 个查询的第 1 个字段升序排列，必要的话再按照第 2 列、第 3 列…

【第十六章 日期函数】

1.date，timestamp 不再介绍，自己回忆一下吧！

```
SQL> col systimestamp for a50;
SQL> select sysdate,systimestamp from dual;

SYSDATE             SYSTIMESTAMP
------------------- --------------------------------------------------
2011-06-05 00:14:28 05-JUN-11 12.14.28.487990 AM +08:00
```

注意：对于会话所在的时区不敏感，不用考虑时区的问题；

拓展：

1.1 timestamp with time zone：带有时区的时间戳；

1.2 timestamp with local time zone：带有本地时区的时间戳，本地时区不显示；

例：

step 1：创建测试表；

```
SQL> create table tt01(d1 timestamp with time zone,d2 timestamp with local time zone);

Table created.

SQL> desc tt01
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 D1                                                 TIMESTAMP(6) WITH TIME ZONE
 D2                                                 TIMESTAMP(6) WITH LOCAL TIME
                                                     ZONE
```

step 2：放入测试数据；

```
SQL> insert into tt01 values(systimestamp,systimestamp);

1 row created.

SQL> commit;

Commit complete.
```

step 3：查询；

```
SQL> set linesize 100;
SQL> col d1 for a40;
SQL> col d2 for a40;
SQL> select * from tt01;

D1                                       D2
---------------------------------------- ----------------------------------------
05-JUN-11 12.24.14.283708 AM +08:00      05-JUN-11 12.24.14.283708 AM
```

注意：timestamp with local time zone，在存储时间时，将会话时间存为数据库服务器所在时区的时间，当在其他时区进行查询时，数据库将其转化为会话所在时区的时间进行显示；

2.tz_offset：此语句执行的时候所在的地区，相对于标准的格林尼治时间偏移了多少时区；

```
SQL> show user
USER is "SCOTT"
SQL> select tz_offset('US/Eastern') from dual;

TZ_OFFS
-------
-04:00

SQL> select tz_offset('PRC') from dual;

TZ_OFFS
-------
+08:00
```

注意：上述地区不是随便写的哦，需要在数据库中可查询的到；　　呵呵，中国是 PRC；

如果想知道时区名的列表，查询 v$timezone_names 即可；

```
SQL> col TZNAME for a20;
SQL> col TZABBREV for a20;
SQL> select * from v$timezone_names;

TZNAME               TZABBREV
-------------------- --------------------
Africa/Algiers       LMT
Africa/Algiers       PMT
Africa/Algiers       WET
Africa/Algiers       WEST
……
W-SU                 EET
W-SU                 EEST
WET                  LMT
WET                  WEST
WET                  WET

1393 rows selected.
```

3. current_date：显示当前会话所在时区的日期和时间；

```
SQL> show user
USER is "SCOTT"
SQL> alter session set nls_date_format = 'yyyy-mm-dd hh24:mi:ss';

Session altered.

SQL> select current_date from dual;

CURRENT_DATE
-------------------
2011-06-05 00:11:05
```

注意：对于会话所在的时区敏感，需考虑时区的问题；

4. current_timestamp：显示当前会话所在时区的日期和时间，包括小时分钟秒；

```
SQL> select current_timestamp from dual;

CURRENT_TIMESTAMP
---------------------------------------------------------------------------
05-JUN-11 12.16.52.121408 AM +08:00
```

注意：对于会话所在的时区敏感，需考虑时区的问题；
返回值是 timestamp with time zone 的类型；

5. localtimestamp：显示当前会话所在时区的日期和时间，包括小时分钟秒；

```
SQL> col CURRENT_TIMESTAMP for a40;
SQL> col LOCALTIMESTAMP for a40;
SQL> select current_timestamp,localtimestamp from dual;

CURRENT_TIMESTAMP                        LOCALTIMESTAMP
---------------------------------------- ----------------------------------------
05-JUN-11 12.31.41.808589 AM +08:00      05-JUN-11 12.31.41.808589 AM
```

注意：与 current_timestamp 区别一下，localtimestamp 不带时区，为 timestamp 类型；

6. dbtimezone：数据库服务器所在时区的值；
sessiontimezone：会话所在时区的值；

```
SQL> select dbtimezone,sessiontimezone from dual;

DBTIME SESSIONTIMEZONE
------ ---------------------------------------------------------------------------
+00:00 +08:00
```

7. extract：从时间中抽取相应的信息；

```
SQL> select extract(year from sysdate) d1,extract(month from sysdate) d2,
  2         extract(day from sysdate) d3,extract(hour from systimestamp) d4,
  3         extract(minute from systimestamp) d5,extract(second from systimestamp) d6
  4  from dual;

        D1         D2         D3         D4         D5         D6
---------- ---------- ---------- ---------- ---------- ----------
      2011          6          5         16         45  47.920007
```

注意：回忆一下第 3 章讲到的 to_char(date 类型)中时间的格式；

8. from_tz：将 timestamp 转换为 timestamp with time zone；

```
SQL> select from_tz(timestamp '2011-06-05 00:49:56','+8:00') from dual;

FROM_TZ(TIMESTAMP'2011-06-0500:49:56','+8:00')
---------------------------------------------------------------------------
05-JUN-11 12.49.56.000000000 AM +08:00
```

注意：时区可以换成时区名；

```
SQL> select from_tz(timestamp '2011-06-05 00:49:56','PRC') from dual;

FROM_TZ(TIMESTAMP'2011-06-0500:49:56','PRC')
---------------------------------------------------------------------------
05-JUN-11 12.49.56.000000000 AM PRC
```

9. to_timestamp：将字符串转换为 timestamp；
   to_timestamp_tz:将字符串转换为 timestamp with time zone；

```
SQL> select to_timestamp('2011-06-05 00:49:56','yyyy-mm-dd hh24:mi:ss') t1,
  2         to_timestamp_tz('2011-06-05 00:49:56 +08:00','yyyy-mm-dd hh24:mi:ss tzh:tzm') t2
  3  from dual;

T1
---------------------------------------------------------------------------
T2
---------------------------------------------------------------------------
05-JUN-11 12.49.56.000000000 AM
05-JUN-11 12.49.56.000000000 AM +08:00
```

10. interval year to month：存储经过多少年多少月的一个时间段；
    interval day to second：存储经过多少天多少小时、分钟、秒的一个时间段；
    提示：year：长度受 BC4712.1.1~AD9999.12.31 限制；month 类似；
          day：长度 0~9，默认为 2；
          second：毫秒的长度 0~9，默认 6；
    10.1 ：定义测试表 tt001，tt002；

```
SQL> create table tt001(t1 interval year(3) to month);

Table created.

SQL> desc tt001
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 T1                                                 INTERVAL YEAR(3) TO MONTH

SQL> create table tt002(t2 interval day(3) to second);

Table created.

SQL> desc tt002
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 T2                                                 INTERVAL DAY(3) TO SECOND(6)
```

10.2 ：放入测试数据；

```
SQL> insert into tt001 values('1-3');

1 row created.

SQL> insert into tt001 values(interval '123-8' year(3) to month);

1 row created.

SQL> insert into tt001 values(interval '123' year(3));

1 row created.

SQL> insert into tt001 values(interval '345' month(3));

1 row created.

SQL> insert into tt001 values(interval '0-9' year to month);

1 row created.

SQL> commit;

Commit complete.

SQL> insert into tt002 values(interval '2 5:15:25.456' day to second(3));

1 row created.

SQL> insert into tt002 values(interval '2 5:15' day to minute);

1 row created.

SQL> insert into tt002 values(interval '2 15' day to hour);

1 row created.

SQL> insert into tt002 values(interval '123' day(3));

1 row created.

SQL> insert into tt002 values(interval '0 15:25:35.123456' day to second(6));

1 row created.

SQL> commit;

Commit complete.
```

10.3 ：查看测试数据；

```
SQL> select * from tt001;

T1
---------------------------------------------------------------------------
+001-03
+123-08
+123-00
+028-09
+000-09

SQL> select * from tt002;

T2
---------------------------------------------------------------------------
+002 05:15:25.456000
+002 05:15:00.000000
+002 15:00:00.000000
+123 00:00:00.000000
+000 15:25:35.123456
```

10.4 ：使用方式；

```
SQL> select sysdate s0,sysdate + t1 s1 from tt001;

S0                  S1
------------------- -------------------
2011-06-05 01:32:12 2012-09-05 01:32:12
2011-06-05 01:32:12 2135-02-05 01:32:12
2011-06-05 01:32:12 2134-06-05 01:32:12
2011-06-05 01:32:12 2040-03-05 01:32:12
2011-06-05 01:32:12 2012-03-05 01:32:12

SQL> select sysdate,sysdate + t2 s2 from tt002;

SYSDATE             S2
------------------- -------------------
2011-06-05 01:32:18 2011-06-07 06:47:43
2011-06-05 01:32:18 2011-06-07 06:47:18
2011-06-05 01:32:18 2011-06-07 16:32:18
2011-06-05 01:32:18 2011-10-06 01:32:18
2011-06-05 01:32:18 2011-06-05 16:57:53
```

注意：可以对 sysdate 直接进行加减运算；

11. to_yminterval：将字符串转换为 interval year to menth 的类型；

例：系统时间经过 3 年 6 个月之后是什么时间？

```
SQL> select sysdate, sysdate + to_yminterval('3-6') s3 from dual;

SYSDATE             S3
------------------- -------------------
2011-06-05 01:36:28 2014-12-05 01:36:28
```

注意：年份默认长度 2 位，月份不要超多 12；

那要是，系统时间退后 3 年 6 个月之后是什么时间？

```
SQL> select sysdate, sysdate + to_yminterval('-3-6') s3 from dual;

SYSDATE             S3
------------------- -------------------
2011-06-05 01:38:52 2007-12-05 01:38:52
```

重点：to_timestamp

【第十七章 高级分组】

1. 回忆：第五章的组函数和分组；

例：将 emp 表中员工的信息按照部门、岗位进行分组，统计总工资，总人数，平均工资；

```
SQL> show user
USER is "SCOTT"
SQL> select deptno,job,sum(sal),count(*),avg(sal)
  2  from emp
  3  group by deptno,job
  4  order by deptno;

    DEPTNO JOB         SUM(SAL)   COUNT(*)   AVG(SAL)
---------- --------- ---------- ---------- ----------
        10 CLERK           1300          1       1300
        10 MANAGER         2450          1       2450
        10 PRESIDENT       5000          1       5000
        20 ANALYST         6000          2       3000
        20 CLERK           1900          2        950
        20 MANAGER         2975          1       2975
        30 CLERK            950          1        950
        30 MANAGER         2850          1       2850
        30 SALESMAN        5600          4       1400

9 rows selected.
```

2. group by 的扩展

2.1 rollup：对 group by 的分组结果进行小计和总计；

```
SQL> set pagesize 50;
SQL> select deptno,job,sum(sal),count(*),avg(sal)
  2  from emp
  3  group by rollup(deptno,job)
  4  order by deptno;

    DEPTNO JOB         SUM(SAL)   COUNT(*)   AVG(SAL)
---------- --------- ---------- ---------- ----------
        10 CLERK           1300          1       1300
        10 MANAGER         2450          1       2450
        10 PRESIDENT       5000          1       5000
        10                 8750          3 2916.66667
        20 ANALYST         6000          2       3000
        20 CLERK           1900          2        950
        20 MANAGER         2975          1       2975
        20                10875          5       2175
        30 CLERK            950          1        950
        30 MANAGER         2850          1       2850
        30 SALESMAN        5600          4       1400
        30                 9400          6 1566.66667
                          29025         14 2073.21429

13 rows selected.
```

注意：与最开始的结果相比，多出来 4 行记录；

分析：按 rollup 函数中的第 1 个字段(deptno)进行小计；最后再形成总计；

2.2 cube：对 group by 的分组结果进行小计和总计；

```
SQL> select deptno,job,sum(sal),count(*),avg(sal)
  2  from emp
  3  group by cube(deptno,job)
  4  order by deptno;

    DEPTNO JOB         SUM(SAL)   COUNT(*)   AVG(SAL)
---------- --------- ---------- ---------- ----------
        10 CLERK           1300          1       1300
        10 MANAGER         2450          1       2450
        10 PRESIDENT       5000          1       5000
        10                 8750          3 2916.66667
        20 ANALYST         6000          2       3000
        20 CLERK           1900          2        950
        20 MANAGER         2975          1       2975
        20                10875          5       2175
        30 CLERK            950          1        950
        30 MANAGER         2850          1       2850
        30 SALESMAN        5600          4       1400
        30                 9400          6 1566.66667
           ANALYST         6000          2       3000
           CLERK           4150          4     1037.5
           MANAGER         8275          3 2758.33333
           PRESIDENT       5000          1       5000
           SALESMAN        5600          4       1400
                          29025         14 2073.21429

18 rows selected.
```

分析：1. 按 cube 函数中的每个字段进行小计；最后再形成总计；

2. 比 rollup 多出来的数据为黑色边框中的数据，即按照第 2 个字段进行小计的结果；

小结：

1. rollup(1…n)会生成 n+1 个子集进行 union all；
2. cube(1…n)会生成 2 的 n 次方个子集进行 union all；

例：

```
group by rollup (a,b,c,d)
等价于
group by a,b,c,d
union all
group by a,b,c
union all
group by a,b
union all
group by a
union all
group by null
```

```
group by cube (a,b,c)
等价于
group by a,b,c
union all
group by a,b
union all
group by a,c
union all
group by b,c
union all
group by a
union all
group by b
union all
group by c
union all
group by null
```

3. grouping()：解释小计值出现对应结果的原因；

例：用 cube 进行分组统计时，使用 grouping()函数对分组字段进行分析；

```
SQL> select deptno,job,sum(sal),count(*),avg(sal),grouping(deptno) g1,grouping(job) g2
  2  from emp
  3  group by cube(deptno,job)
  4  order by deptno;

    DEPTNO JOB         SUM(SAL)   COUNT(*)   AVG(SAL)         G1         G2
---------- --------- ---------- ---------- ---------- ---------- ----------
        10 CLERK           1300          1       1300          0          0
        10 MANAGER         2450          1       2450          0          0
        10 PRESIDENT       5000          1       5000          0          0
        10                 8750          3 2916.66667          0          1
        20 ANALYST         6000          2       3000          0          0
        20 CLERK           1900          2        950          0          0
        20 MANAGER         2975          1       2975          0          0
        20                10875          5       2175          0          1
        30 CLERK            950          1        950          0          0
        30 MANAGER         2850          1       2850          0          0
        30 SALESMAN        5600          4       1400          0          0
        30                 9400          6 1566.66667          0          1
           ANALYST         6000          2       3000          1          0
           CLERK           4150          4     1037.5          1          0
           MANAGER         8275          3 2758.33333          1          0
           PRESIDENT       5000          1       5000          1          0
           SALESMAN        5600          4       1400          1          0
                          29025         14 2073.21429          1          1

18 rows selected.
```

注意：1. 此处使用 rollup 进行分组统计也可以；

2. 用于 grouping()函数的字段，为分组中使用到得字段；

3. 对于 grouping()函数对分组字段的分析结果，只有 0 和 1 两种值；

详解：1. grouping()统计结果为 0，表示：在取得组函数结果的时候，该字段参与了条件运算；

2. grouping()统计结果为 1，表示：在取得组函数结果的时候，该字段未参与条件运算；

例：

```
    DEPTNO JOB         SUM(SAL)   COUNT(*)   AVG(SAL)         G1         G2
---------- --------- ---------- ---------- ---------- ---------- ----------
        10                 8750          3 2916.66667          0          1
           ANALYST         6000          2       3000          1          0
                          29025         14 2073.21429          1          1
```

第 1 种情况

第 2 种情况

第 3 种情况

注意：

第 1 种情况：G1=0，G2=1 （解释：10 号部门的总工资、总人数、平均工资是多少，不限定岗位；）

第 2 种情况：G1=1，G2=0 （解释：岗位是 ANALYST 的总工资、总人数、平均工资是多少，不限定部门；）

第 3 种情况：G1=1，G2=1 （解释：所有员工的总工资、总人数、平均工资是多少，不限定部门和岗位；）

注意：

第 1 种情况：G1=0，G2=1 的情况下，表示在取得组函数结果时，job 没有参与条件运算，结果中 job 对应的结果为 null，表示此 null 值是由于 rollup 或者 cube 分组而造成的； 同理解释第 2 种情况和第 3 种情况；

第 1 种情况：G1=0，G2=1 的情况下，表示在取得组函数结果时，deptno 参与了条件运算，但是，如果 deptno 对应的结果为 null，表示此 null 值是由于存储而导致的空值；（即该记录中 deptno 原本就是空值）

小结：以上 grouping()结果中 0、1 要知道原因； 关于计算空值和存储空值，了解。

4. grouping sets()：分组集

例：在 emp 中，先按照 deptno、job 进行分组，计算员工的工资情况；再按照 job、mgr 进行分组，再计算员工的工资情况； 并且两部分的统计结果要一起显示；

```
SQL> select deptno,job,to_number(null) mgr,sum(sal),count(*),avg(sal)
  2  from emp
  3  group by deptno,job
  4  union all
  5  select to_number(null) deptno,job,mgr,sum(sal),count(*),avg(sal)
  6  from emp
  7  group by job,mgr
  8  order by 1,2,3;

    DEPTNO JOB              MGR   SUM(SAL)   COUNT(*)   AVG(SAL)
---------- --------- ---------- ---------- ---------- ----------
        10 CLERK                      1300          1       1300
        10 MANAGER                    2450          1       2450
        10 PRESIDENT                  5000          1       5000
        20 ANALYST                    6000          2       3000
        20 CLERK                      1900          2        950
        20 MANAGER                    2975          1       2975
        30 CLERK                       950          1        950
        30 MANAGER                    2850          1       2850
        30 SALESMAN                   5600          4       1400
           ANALYST          7566       6000          2       3000
           CLERK            7698        950          1        950
           CLERK            7782       1300          1       1300
           CLERK            7788       1100          1       1100
           CLERK            7902        800          1        800
           MANAGER          7839       8275          3 2758.33333
           PRESIDENT                   5000          1       5000
           SALESMAN         7698       5600          4       1400

17 rows selected.
```

注意：1. 如果集合运算的字段数匹配不上，应该用 null 进行充当；

2. 注意：两个分组之间以 union all 的形式进行合并；

3. order by 子句是对整个结果集按照第 1 个查询的 3 个字段进行排序；

如果想简化以上的算法，应该使用分组集；

```
SQL> select deptno,job,mgr,sum(sal),count(*),avg(sal)
  2  from emp
  3  group by grouping sets((deptno,job),(job,mgr))
  4  order by deptno,job,mgr;
```

注意：1. 此写法与前面写法生成的结果是相同的，但是效率会有所提升；

2. 排序如果不是必须的，最好省略，以提高检索效率；

以下知识点了解；

5.用于分组的复合列

例：rollup(a,(b,c),d)的解释；

```
group by rollup(a,(b,c),d)
等价于
group by a,b,c,d
union all
group by a,b,c
union all
group by a
union all
group by null
```

注意：1. 此处有 3 个列，应产生 4 个子集；b,c 属于符合列，看做一个整体；
2. 想一下 rollup(a,x,d)等价于什么？然后把 x 用 b,c 替换即可；

例：cube(a,(b,c),d)的解释；

```
group by cube (a,(b,c),d)
等价于
group by a, b,c,d
union all
group by a, b,c
union all
group by a,d
union all
group by b,c,d
union all
group by a
union all
group by b,c
union all
group by d
union all
group by null
```

注意：1. 此处有 3 个列，应产生 8 个子集；b,c 属于符合列，看做一个整体；
2. 想一下 cube(a,x,d)等价于什么？然后把 x 用 b,c 替换即可；

6.连接分组

例：

```
group by grouping sets(a,b), grouping sets(c,d)
等价于
group by a,c
union all
group by a,d
union all
group by b,c
```

union all

group by b,d

注意：是根据分组集形成笛卡尔积，并用于 group by 分组；

【第十八章 高级子查询】

1.高级子查询

提示：使用子查询的时候注意单行运算符和多行运算符的使用；

1.1 成对比较子查询

例：查询一些员工的信息，这些员工和 7499 或 7788 在同一个部门，被同一个经理管理，但是员工编号不等于 7499、7788 的员工；

step 1：查询 emp 表中员工的信息；

```
SQL> show user
USER is "SCOTT"
SQL> set linesize 100;
SQL> select * from emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

```
14 rows selected.
```

step 2：查询一下 7499 或 7788 的信息；

```
SQL> select deptno,mgr,empno,ename,job,hiredate,sal,comm
  2  from emp
  3  where empno in (7499,7788)
  4  order by deptno,mgr;
```

DEPTNO	MGR	EMPNO	ENAME	JOB	HIREDATE	SAL	COMM
20	7566	7788	SCOTT	ANALYST	19-APR-87	3000	
30	7698	7499	ALLEN	SALESMAN	20-FEB-81	1600	300

step 3：想一下，此结果该如何理解；

```
SQL> select deptno,mgr,empno,ename,job,hiredate,sal,comm
  2  from emp
  3  where (deptno,mgr) in (select deptno,mgr from emp
  4                         where empno in (7499,7788))
  5  and empno not in (7499,7788)
  6  order by deptno,mgr,empno;
```

DEPTNO	MGR	EMPNO	ENAME	JOB	HIREDATE	SAL	COMM
20	7566	7902	FORD	ANALYST	03-DEC-81	3000	
30	7698	7521	WARD	SALESMAN	22-FEB-81	1250	500
30	7698	7654	MARTIN	SALESMAN	28-SEP-81	1250	1400
30	7698	7844	TURNER	SALESMAN	08-SEP-81	1500	0
30	7698	7900	JAMES	CLERK	03-DEC-81	950	

注意：每个部门最后都会被 7839 管理，谁啊？ 呵呵，KING 嘛！

1.2 非成对比较子查询

例：查询一些员工的信息，这些员工和 7499 或 7788 在同一个部门，被同一个经理管理，但是员工编号不等于 7499、7788 的员工；

提示：哈哈，和上面的问题一样，只是实现的方式不一样；

step 1：想一下，此结果应该可以理解了吧；

```
SQL> select deptno,mgr,empno,ename,job,hiredate,sal,comm
  2  from emp
  3  where mgr in (select mgr from emp where empno in (7499,7788))
  4  and deptno in (select deptno from emp where empno in (7499,7788))
  5  and empno not in (7499,7788)
  6  order by deptno,mgr,empno;

    DEPTNO        MGR      EMPNO ENAME      JOB       HIREDATE          SAL       COMM
---------- ---------- ---------- ---------- --------- --------- ---------- ----------
        20       7566       7902 FORD       ANALYST   03-DEC-81       3000
        30       7698       7521 WARD       SALESMAN  22-FEB-81       1250        500
        30       7698       7654 MARTIN     SALESMAN  28-SEP-81       1250       1400
        30       7698       7844 TURNER     SALESMAN  08-SEP-81       1500          0
        30       7698       7900 JAMES      CLERK     03-DEC-81        950
```

注意：两种方式实现的结果虽然相同，但是工作中会采用成对比较子查询，因为写法简便，效率高；

提问：为啥成对比较子查询的效率高呢？　　呵呵，原因很简单，减少了 1 次子查询；

2.在 from 子句中使用子查询

例：查询一些员工的信息，这些员工的工资比所在部门的平均工资高；

```
SQL> select y.empno, y.ename, y.deptno, y.sal, b.avgsal
  2  from emp y, (select deptno, avg(sal) avgsal from emp
  3               group by deptno) b
  4  where y.deptno = b.deptno
  5  and y.sal > b.avgsal;

     EMPNO ENAME          DEPTNO        SAL     AVGSAL
---------- ---------- ---------- ---------- ----------
      7698 BLAKE              30       2850 1566.66667
      7499 ALLEN              30       1600 1566.66667
      7902 FORD               20       3000       2175
      7788 SCOTT              20       3000       2175
      7566 JONES              20       2975       2175
      7839 KING               10       5000 2916.66667

6 rows selected.
```

注意：其实 from 子句中的子查询得到的结果只是作为临时的数据集存在；

3.在 case 语句中使用子查询

例：哈哈，这个效果太逗了！

```
SQL> select empno,ename,sal,
  2    case
  3      when sal <= (select min(sal) from emp) then 'Too low,come on!'
  4      when sal > (select min(sal) from emp)
  5       and sal <= (select avg(sal) from emp) then 'Not yet,come on,go...'
  6      when sal > (select avg(sal) + 2000 from emp) then 'Have a rest,hehe! then continual!'
  7      else 'GanBaRe!'
  8      end as info
  9  from emp;
```

提示：GanBaRe 是啥啊？　頑張れ！　　In chinese means　‘加油吧!’。

4.关联更新

例：

step 1：创建测试表，定义参照 emp，数据保留 5 条；

```
SQL> show user
USER is "SCOTT"
SQL> create table tu01(no,name)
  2  as select empno,ename from emp where rownum <= 5;

Table created.

SQL> desc tu01
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------
 NO                                                             NUMBER(4)
 NAME                                                           VARCHAR2(10)
```

step 2：添加新列 info，查询表中的数据；

```
SQL> alter table tu01 add(info varchar2(50));

Table altered.

SQL> select * from tu01;

        NO NAME       INFO
---------- ---------- --------------------------------------------------
      7369 SMITH
      7499 ALLEN
      7521 WARD
      7566 JONES
      7654 MARTIN
```

step 3：进行关联更新；(将 emp 表中员工的部门，岗位，工资转存到 info 中去)

```
SQL> update tu01 t set info = (
  2      select deptno||','||job||','||sal from emp e
  3      where t.no = e.empno);

5 rows updated.

SQL> commit;

Commit complete.
```

step 4：查询更新之后的信息；

```
SQL> select * from tu01;

        NO NAME       INFO
---------- ---------- --------------------------------------------------
      7369 SMITH      20,CLERK,800
      7499 ALLEN      30,SALESMAN,1600
      7521 WARD       30,SALESMAN,1250
      7566 JONES      20,MANAGER,2975
      7654 MARTIN     30,SALESMAN,1250
```

5.关联删除

例：删除与 7788 在同一部门的员工信息；

step 1：创建测试表，定义参照 emp，数据保留；

```
SQL> show user
USER is "SCOTT"
SQL> create table tu02
  2  as select * from emp;

Table created.
```

step 2：关联删除；

```
SQL> delete from tu02
  2  where deptno = (select deptno from emp where empno = 7788);

5 rows deleted.

SQL> commit;
```

step 3：查看结果；

```
SQL> select * from tu02;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7499 ALLEN      SALESMAN         7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN         7698 22-FEB-81       1250        500         30
      7654 MARTIN     SALESMAN         7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER          7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER          7839 09-JUN-81       2450                    10
      7839 KING       PRESIDENT             17-NOV-81       5000                    10
      7844 TURNER     SALESMAN         7698 08-SEP-81       1500          0         30
      7900 JAMES      CLERK            7698 03-DEC-81        950                    30
      7934 MILLER     CLERK            7782 23-JAN-82       1300                    10

9 rows selected.
```

注意：7788 这个员工的信息也被删除了，因为他也满足条件；

6. exists 和 not exists

提示：exists 操作对子查询的结果存在与否进行检验：

如果子查询中没有结果，则条件被标记为 false，搜索继续；

如果子查询中存在结果，则条件被标记为 true，搜索终止；

例：查询 emp 中那些经理至少有 1 个下属？

提示：只要有就行，是谁？有几个？不是关注的对象。

```
SQL> select * from emp outer
  2  where exists (select 'X' from emp inner
  3                where inner.mgr = outer.empno);

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7902 FORD       ANALYST          7566 03-DEC-81       3000                    20
      7698 BLAKE      MANAGER          7839 01-MAY-81       2850                    30
      7839 KING       PRESIDENT             17-NOV-81       5000                    10
      7566 JONES      MANAGER          7839 02-APR-81       2975                    20
      7788 SCOTT      ANALYST          7566 19-APR-87       3000                    20
      7782 CLARK      MANAGER          7839 09-JUN-81       2450                    10

6 rows selected.
```

反之，那些人没有下属呢？

```
SQL> select * from emp outer
  2   where not exists (select 'X' from emp inner
  3                     where inner.mgr = outer.empno);

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7844 TURNER     SALESMAN         7698 08-SEP-81       1500          0         30
      7521 WARD       SALESMAN         7698 22-FEB-81       1250        500         30
      7654 MARTIN     SALESMAN         7698 28-SEP-81       1250       1400         30
      7499 ALLEN      SALESMAN         7698 20-FEB-81       1600        300         30
      7934 MILLER     CLERK            7782 23-JAN-82       1300                    10
      7369 SMITH      CLERK            7902 17-DEC-80        800                    20
      7876 ADAMS      CLERK            7788 23-MAY-87       1100                    20
      7900 JAMES      CLERK            7698 03-DEC-81        950                    30

8 rows selected.
```

7. with 语句

提示：如果临时数据集在一次查询中经常被用到，那么可以提前定义；

优点：降低查询的复杂度，提高检索效率；

例：在 emp 表中查询哪个部门(名字)的总工资大于所有部门的平均工资；

方法一：

```
SQL> select d.dname,sum(e.sal) as zsal
  2  from emp e,dept d
  3  where e.deptno = d.deptno
  4  group by d.dname
  5  having sum(e.sal) > (select sum(sal)/count(distinct deptno) from emp);

DNAME                ZSAL
-------------- ----------
RESEARCH            10875
```

此处是以子查询的方式实现；

方法二：

```
SQL> with dept_cost
  2  as (select d.dname,sum(e.sal) dsal
  3      from emp e,dept d
  4      where e.deptno = d.deptno
  5      group by d.dname),
  6  zd_sal as (select sum(dsal)/count(*) davg from dept_cost)
  7  select * from dept_cost
  8  where dsal > (select davg from zd_sal)
  9  order by 1;

DNAME                DSAL
-------------- ----------
RESEARCH            10875
```

注意：1.with 子句的目的：定义临时表，其中 dept_cost、zd_sal 属于临时表；
2.临时表只在此 SQL 语句中有效；
3.下面的查询用到了上面定义的临时表；

一点灵感：看下面的一个小问题？
例：如果 河河+工工+大大=河工大 ，其中河、工、大属于 0-9 之间的数字；
问：河、工、大分别是什么数？
step 1：创建一张测试表；

```
SQL> create table tu03(no number(1));

Table created.

SQL> desc tu03
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------
 NO                                                             NUMBER(1)
```

step 2：存入 0-9 这 10 个数字；

```
SQL> insert into tu03 values(0);

1 row created.

SQL> insert into tu03
  2  select max(no) + 1 from tu03;

1 row created.
```

… 中间省略 …

```
SQL> commit;

Commit complete.

SQL> select * from tu03;

        NO
----------
         0
```

```
         1
         2
         3
         4
         5
         6
         7
         8
         9

10 rows selected.
```

step 3：写个简单的 SQL 语句吧！

```
SQL> select x.no h, y.no g, z.no d
  2  from tu03 x, tu03 y, tu03 z
  3  where (10 * x.no + x.no) + (10 * y.no + y.no) + (10 * z.no + z.no)
  4        = (100 * x.no + 10 * y.no + z.no);

         H          G          D
---------- ---------- ----------
         0          0          0
         1          9          8
```

小结：以上只是采用简单算法实现，等学完后面的知识，想想有没有更好的解决办法！

【第十九章 分级取数据】

1.树形化结构

例：利用 emp 做相关练习；

step 1：查询 emp 表的信息；

```
SQL> select * from emp;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
```

中间数据省略

```
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10

14 rows selected.
```

step 2：(树形结构)显示 emp 表中员工之间的层级关系；

```
SQL> col result for a40;
SQL> select level,lpad(ename,length(ename)+(level*4-2),'-') result
  2  from emp
  3  start with ename = 'KING'
  4  connect by prior empno = mgr;

     LEVEL RESULT
---------- ----------------------------------------
         1 --KING
         2 ------JONES
         3 ----------SCOTT
         4 --------------ADAMS
         3 ----------FORD
         4 --------------SMITH
         2 ------BLAKE
         3 ----------ALLEN
         3 ----------WARD
         3 ----------MARTIN
         3 ----------TURNER
         3 ----------JAMES
         2 ------CLARK
         3 ----------MILLER

14 rows selected.
```

注意：start with 表示开始遍历的节点信息；
connect by 表示遍历的方向；
上述例子中的意思：从 KING 节点开始遍历；看一下 KING 的 empno 是谁的 mgr，由此可知方向由上及下；

step 3：查询 JONES 下属的信息；

```
SQL> select level,lpad(ename,length(ename)+(level*4-2),'-') result
  2  from emp
  3  start with ename = 'JONES'
  4  connect by prior empno = mgr;

     LEVEL RESULT
---------- ----------------------------------------
         1 --JONES
         2 ------SCOTT
         3 ----------ADAMS
         2 ------FORD
         3 ----------SMITH
```

注意：1.节点从哪开始，哪个节点的 level 就是 1，越向下 level 越低；
2.看一下 JONES 的 empno 是谁的 mgr，由此可知方向由上及下；

step 4：查询 SMITH 上级的信息；

```
SQL> select level,lpad(ename,length(ename)+(level*4-2),'-') result
  2  from emp
  3  start with ename = 'SMITH'
  4  connect by prior mgr = empno;

     LEVEL RESULT
---------- ----------------------------------------
         1 --SMITH
         2 ------FORD
         3 ----------JONES
         4 --------------KING
```

注意：1.节点从哪开始，哪个节点的 level 就是 1，越向下 level 越低；
2.看一下 SMITH 的 mgr 是谁的 empno，由此可知方向由下及上；

step 5：查询与 BLAKE 同级别的员工的信息；

```
SQL> with info
  2  as (select level le,ename,lpad(ename,length(ename)+(level*4-2),'-') result
  3      from emp
  4      start with ename = 'KING'
  5      connect by prior empno = mgr)
  6  select * from info
  7  where le = (select le from info where ename = 'BLAKE');

        LE ENAME      RESULT
---------- ---------- ----------------------------------------
         2 JONES      ------JONES
         2 BLAKE      ------BLAKE
         2 CLARK      ------CLARK
```

2.修剪分支

2.1 使用 where 子句修剪节点信息；

提示：在查询层级信息的时候，使用 where 子句控制那些节点信息不被查询；

step 1：ename 为 BLAKE 的节点信息不被查询；（影响单个节点）

```
SQL> select level,lpad(ename,length(ename)+(level*4-2),'-') result
  2  from emp
  3  where ename <> 'BLAKE'
  4  start with ename = 'KING'
  5  connect by prior empno = mgr;

     LEVEL RESULT
---------- ----------------------------------------
         1 --KING
         2 ------JONES
         3 ----------SCOTT
         4 --------------ADAMS
         3 ----------FORD
         4 --------------SMITH
         3 ----------ALLEN
         3 ----------WARD
         3 ----------MARTIN
         3 ----------TURNER
         3 ----------JAMES
         2 ------CLARK
         3 ----------MILLER

13 rows selected.
```

2.2 使用 connect by 子句修剪分支信息；

提示：在查询层级信息的时候，使用 connect by 子句控制那些分支信息不被查询；

step 1：ename 为 BLAKE 的分支信息不被查询；（影响此节点，以及节点之下所有的节点）

```
SQL> select level,lpad(ename,length(ename)+(level*4-2),'-') result
  2  from emp
  3  start with ename = 'KING'
  4  connect by prior empno = mgr
  5  and ename <> 'BLAKE';

     LEVEL RESULT
---------- ----------------------------------------
         1 --KING
         2 ------JONES
         3 ----------SCOTT
         4 --------------ADAMS
         3 ----------FORD
         4 --------------SMITH
         2 ------CLARK
         3 ----------MILLER

8 rows selected.
```

2 ------BLAKE
3 ----------ALLEN
3 ----------WARD
3 ----------MARTIN
3 ----------TURNER
3 ----------JAMES

【第二十章 DML 扩展】

前提：创建测试表，定义参照 emp，数据不要；

```
SQL> create table f01(eno,name)      as select empno,ename from emp where 1=2;

Table created.

SQL> create table f02(eno,work)      as select empno,job from emp where 1=2;

Table created.

SQL> create table f03(eno,dt,dno)    as select empno,hiredate,deptno from emp where 1=2;

Table created.

SQL> create table f04(eno,sal,comm)  as select empno,sal,comm from emp where 1=2;

Table created.
```

1. 无条件多表 insert

step 1：操作；

```
SQL> insert all
  2    into f01 values(empno,ename)
  3    into f02 values(empno,job)
  4    into f03 values(empno,hiredate,deptno)
  5    into f04 values(empno,sal,comm)
  6  select empno,ename,job,hiredate,deptno,sal,comm
  7  from emp
  8  where rownum <= 1;

4 rows created.

SQL> commit;

Commit complete.
```

注意：1. 将子查询得到的数据分块，无条件 insert 到多张表里；

2. 多张表中的字段值要和子查询中的字段值对应，名字和顺序可以不一样，但一定要有对应字段；

3. 上述例子中将检索到得 1 条数据，分组 insert 到以上 4 张表里；

4. 子查询中的每 1 条数据都会被分成 4 份，分别放入 4 张表中，直到所有数据操作完毕；

step 2：查询多张表内的信息；

```
SQL> select * from f01;

       ENO NAME
---------- ----------
      7369 SMITH

SQL> select * from f02;

       ENO WORK
---------- ---------
      7369 CLERK

SQL> select * from f03;

       ENO DT                DNO
---------- --------- ----------
      7369 17-DEC-80         20

SQL> select * from f04;

       ENO        SAL       COMM
---------- ---------- ----------
      7369        800
```

2. 有条件 all insert

前提：将 f01 到 f04 表中的信息删除，便于之后的实验；

```
SQL> truncate table f01;

Table truncated.

SQL> truncate table f02;

Table truncated.

SQL> truncate table f03;

Table truncated.

SQL> truncate table f04;

Table truncated.
```

step 1：操作；

```
SQL> insert all
  2    when ename = 'SCOTT' then
  3      into f01 values(empno,ename)
  4    when job = 'CLERK' then
  5      into f02 values(empno,job)
  6    when deptno = 10 then
  7      into f03 values(empno,hiredate,deptno)
  8    when sal > 2500 then
  9      into f04 values(empno,sal,comm)
 10  select empno,ename,job,hiredate,deptno,sal,comm
 11  from emp;

13 rows created.

SQL> commit;

Commit complete.
```

step 2：查询多张表内的信息；

```
SQL> select * from f01;

       ENO NAME
---------- ----------
      7788 SCOTT

SQL> select * from f02;

       ENO WORK
---------- ---------
      7369 CLERK
      7876 CLERK
      7900 CLERK
      7934 CLERK

SQL> select * from f03;

       ENO DT                DNO
---------- --------- ----------
      7782 09-JUN-81         10
      7839 17-NOV-81         10
      7934 23-JAN-82         10

SQL> select * from f04;

       ENO        SAL       COMM
---------- ---------- ----------
      7566       2975
      7698       2850
      7788       3000
      7839       5000
      7902       3000
```

注意：1.从子查询中得到的所有数据，依次经历各个条件，有满足条件的数据，就对应放入指定的表；

2.子查询中的每 1 条数据都会走遍所有条件分支，如果满足条件，则对应信息放入对应的表，直到所有数据操作完毕；

3.有条件 first insert

前提：将 f01 到 f04 表中的信息删除，便于之后的实验；（省略）

step 1：操作；

```
SQL> insert first
  2    when ename = 'SCOTT' then
  3      into f01 values(empno,ename)
  4    when job = 'CLERK' then
  5      into f02 values(empno,job)
  6    when deptno = 10 then
  7      into f03 values(empno,hiredate,deptno)
  8    when sal > 2500 then
  9      into f04 values(empno,sal,comm)
 10  select empno,ename,job,hiredate,deptno,sal,comm
 11  from emp;

10 rows created.

SQL> commit;

Commit complete.
```

step 2：查询多张表内的信息；

```
SQL> select * from f01;

       ENO NAME
---------- ----------
      7788 SCOTT

SQL> select * from f02;

       ENO WORK
---------- ---------
      7369 CLERK
      7876 CLERK
      7900 CLERK
      7934 CLERK

SQL> select * from f03;

       ENO DT                DNO
---------- --------- ----------
      7782 09-JUN-81         10
      7839 17-NOV-81         10

SQL> select * from f04;

       ENO        SAL       COMM
---------- ---------- ----------
      7566       2975
      7698       2850
      7902       3000
```

注意：1.子查询中的每 1 条数据只查找第一个为 true 条件分支，将对应信息放入对应的表之后，后续条件即使满足也不再判断；然后进行下一条数据，直到所有数据操作完毕；

4.Pivoting insert(枢轴式 insert)

例：以下是员工每周每天的销售情况；

如果想统计一下每周的销售情况，应该怎么做？

销售表A	sale_hd							
empno	weekid	sal_mon	sal_tue	sal_wed	sal_thur	sal_fri	sal_liu	sal_sun
1001	1	10	11	12	13	14	15	16
1001	2	11	12	13	14	15	16	17
1002	1	12	13	14	15	16	17	18
1002	2	13	14	15	16	17	18	19
1002	3	14	15	16	17	18	19	20
1003	1	15	16	17	18	19	20	21

原始方法：

step 1：创建测试表 sale_hd，填充测试数据(略)；

```
SQL> desc sale_hd
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------
 EMPNO                                                          NUMBER(4)
 WEEKID                                                         NUMBER(2)
 SAL_MON                                                        NUMBER(5)
 SAL_TUE                                                        NUMBER(5)
 SAL_WED                                                        NUMBER(5)
 SAL_THUR                                                       NUMBER(5)
 SAL_FRI                                                        NUMBER(5)
 SAL_LIU                                                        NUMBER(5)
 SAL_SUN                                                        NUMBER(5)
```

体力活：按照上述的定义去做一张表，数据自己按照题目要求填充；

step 2：查询 sale_hd 的信息；

```
SQL> select * from sale_hd;

     EMPNO     WEEKID    SAL_MON    SAL_TUE    SAL_WED   SAL_THUR    SAL_FRI    SAL_LIU    SAL_SUN
---------- ---------- ---------- ---------- ---------- ---------- ---------- ---------- ----------
      1001          1         10         11         12         13         14         15         16
      1001          2         11         12         13         14         15         16         17
      1002          1         12         13         14         15         16         17         18
      1002          2         13         14         15         16         17         18         19
      1002          3         14         15         16         17         18         19         20
      1003          1         15         16         17         18         19         20         21

6 rows selected.
```

step 3：查询每周的销售情况；

```
SQL> select weekid,sum(SAL_MON+SAL_TUE+SAL_WED+SAL_THUR+SAL_FRI+SAL_LIU+SAL_SUN) result
  2  from sale_hd
  3  group by weekid
  4  order by weekid;

    WEEKID     RESULT
---------- ----------
         1        322
         2        210
         3        119
```

是不是感觉比较简单！？

重点还没开始呢？

如果，如果将上述表中的数据存成这样，计算会变简单么？

销售明细表A	sale_ms	
empno	weekid	sal_day
1001	1	10
1001	1	11
1001	1	12
1001	1	13
1001	1	14
1001	1	15
1001	1	16
1001	2	11
1001	2	12
1001	2	13
1001	2	14
1001	2	15
1001	2	16
1001	2	17
1002	1	12
1002	1	13

.....未完待续！！

这样的话，只需要写一句 select weekid，sum(sal_day) from sale_ms
group by weekid;　　就 OK 了！

那如何实现呢？

step 1：先创建一张这样的表；

```
SQL> desc sale_ms
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                              NUMBER(4)
 WEEKID                                             NUMBER(2)
 SAL_DAY                                            NUMBER(5)
```

step 2：将 sale_hd 的数据存储进来；

疑惑：怎么存啊，挨个 insert 进来，NO…

```
insert all
  into sale_ms values(empno,weekid,sal_mon)
  into sale_ms values(empno,weekid,sal_tue)
  into sale_ms values(empno,weekid,sal_wed)
  into sale_ms values(empno,weekid,sal_thur)
  into sale_ms values(empno,weekid,sal_fri)
  into sale_ms values(empno,weekid,sal_liu)
  into sale_ms values(empno,weekid,sal_sun)
select empno,weekid,sal_mon,sal_tue,sal_wed,sal_thur,sal_fri,sal_liu,sal_sun
from sale_hd;
```

----------哈哈，上面的命令可以写在脚本里，免得写错！

```
-rw-r--r-- 1 oracle oinstall  413 Jun  5 21:08 pv2.sql
[oracle@rhel ~]$ exit
exit

SQL> show user
USER is "SCOTT"
SQL> @pv2.sql

42 rows created.
```

step 3：检索一下数据；

```
SQL> select * from sale_ms;

     EMPNO     WEEKID    SAL_DAY
---------- ---------- ----------
      1001          1         10
```

----------中间部分省略

```
      1003          1         21

42 rows selected.
```

step 4：终于到最后了！

```
SQL> select weekid,sum(sal_day) result
  2  from sale_ms
  3  group by weekid;

    WEEKID     RESULT
---------- ----------
         1        322
         2        210
         3        119
```

注意：以上数据的转存过程，就是枢轴式 insert；

【知识点补充】

1.关于组函数使用时对 null 值的使用；　　【第五章 组函数的使用】

例：以 emp 表为例，我想查询一下所有员工的工资和奖金的总和；

```
SQL> select * from emp;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10

14 rows selected.
```

想法一：单独实现，分别计算工资的总和，再计算奖金的总和，然后再相加；

```
SQL> select sum(sal) + sum(comm)  result from emp;

    RESULT
----------
     31225
```

注意：大家考虑一下，这个结果对么？

想法二：绑定实现，先计算每个人的工资和奖金的和，然后 14 人的再相加；

```
SQL> select sum(sal + comm) result from emp;

    RESULT
----------
      7800
```

注意：大家再考虑一下，这个结果对么？

奇怪了，同样都是查询 emp 表中所有员工的工资和奖金，为啥结果不一样呢？

原因:大家有没有看到 comm 中有 null 值,如果 sal 加上一个为 null 的 comm 值,结果会是什么？！ null 对吧，所以想法二就有问题了！　明白了么？

解决：那怎么将想法二进行改进而得到正确的结果呢？　　对，可以使用 nvl、nvl2 等方法将 null 值进行转换；

```
SQL> select sum(sal + nvl(comm,0)) result from emp;

    RESULT
----------
     31225
```

结论：1.组函数在计算过程中忽略 null 值，count(*)除外；

2.对于存在 null 值的字段，如果需要参加计算，需根据业务要求进行转换；