Oracle SQL 和 PL/SQL 学习笔记

小可

原书：《精通 Oracle 10g SQL 和 PL/SQL》
说明：这是今年 4 月和 5 月自己学习 Oracle 的笔记，都是实例，一个知识点一个例子，我觉得这种学习方式比纯学理论的要好，整理出来供大家参考，希望对学习 oracle 和 SQL 的人有所帮助。我也还是一个初学者，希望大家多多交流，共同进步！
笔记里用到的 emp 表，dept 表都是完整版才有的 Samples。
附录里有 Oracle 完整版的下载地址，不是 Oracle 产品页面 600M 的精简版，是包含全部例子程序和管理工具的完全版，不需要注册，建议大家下载。
学习时间：2008 年 4 月~5 月
我的Blog：http://hi.baidu.com/mcxiaoke
联系方式：
E-mail:zxk198597@126.com
QQ:457897575

mcxiaoke
2008-5-28

目录

第一章 基本查询语句

更改会话为简体中文
alter session set nls_date_language='SIMPLIFIED CHINESE';

更改会话为美国英语
alter session set nls_date_language='AMERICAN';

以特定格式显示日期
alter session set nls_date_format='YYYY"年"MM"月"DD"日"';

使用 TO_CHAR 函数定制日期显示格式
select ename,to_char(hiredate, 'YYYY-MM-DD') from emp;

排除重复行
select 默认包含重复行，用 distinct 排除重复行
select distinct deptno,job from emp;

使用算术运算符
select ename,sal,sal*12 from emp;

NULL 处理

默认不处理 NULL：
select ename,sal,comm,sal+comm from emp;

ENAME	SAL	COMM	SAL+COMM
SMITH	800		
ALLEN	1600	300	1900
WARD	1250	500	1750
JONES	2975		
MARTIN	1250	1400	2650
BLAKE	2850		
CLARK	2450		
SCOTT	3000		
KING	5000		
TURNER	1500	0	1500
ADAMS	1100		
JAMES	950		
FORD	3000		
MILLER	1300		

用 NVL 函数处理 NULL 的情况：
select ename,sal,comm,sal+nvl(comm,0) from emp;

（如果 comm 存在就返回原值，不存在【为 NULL】就返回 0）

ENAME	SAL	COMM	SAL+NVL(COMM,0)
SMITH	800		800
ALLEN	1600	300	1900
WARD	1250	500	1750
JONES	2975		2975
MARTIN	1250	1400	2650
BLAKE	2850		2850
CLARK	2450		2450
SCOTT	3000		3000
KING	5000		5000
TURNER	1500	0	1500
ADAMS	1100		1100
JAMES	950		950
FORD	3000		3000
MILLER	1300		1300

用 NVL2 函数处理的情况：

```
select ename,sal,comm, nvl2(comm,sal+comm,sal) from emp;
```

（如果 comm 不是 NULL 返回 sal+comm，否则返回 sal 的值）

ENAME	SAL	COMM	NVL2(COMM,SAL+COMM,SAL)
SMITH	800		800
ALLEN	1600	300	1900
WARD	1250	500	1750
JONES	2975		2975
MARTIN	1250	1400	2650
BLAKE	2850		2850
CLARK	2450		2450
SCOTT	3000		3000
KING	5000		5000
TURNER	1500	0	1500
ADAMS	1100		1100
JAMES	950		950
FORD	3000		3000
MILLER	1300		1300

使用||连接字符串

```
select ename || "" || 's job is ' || job from emp;
----------------------------
SMITH's job is CLERK
ALLEN's job is SALESMAN
WARD's job is SALESMAN
JONES's job is MANAGER
```

MARTIN's job is SALESMAN
BLAKE's job is MANAGER
CLARK's job is MANAGER
SCOTT's job is ANALYST
KING's job is PRESIDENT
TURNER's job is SALESMAN
ADAMS's job is CLERK
JAMES's job is CLERK
FORD's job is ANALYST
MILLER's job is CLERK

使用函数 CONCAT 连接字符串
select concat(concat(ename, '''s salary is '),sal) from emp;
CONCAT(CONCAT(ENAME,'''SSALARYIS'),SAL)

SMITH's salary is 800
ALLEN's salary is 1600
WARD's salary is 1250
JONES's salary is 2975
MARTIN's salary is 1250
BLAKE's salary is 2850
CLARK's salary is 2450
SCOTT's salary is 3000
KING's salary is 5000
TURNER's salary is 1500
ADAMS's salary is 1100
JAMES's salary is 950
FORD's salary is 3000
MILLER's salary is 1300

不使用列别名
select ename,sal*12 from emp;
使用列别名：
select ename "Name",sal*12 "Annual Salary" from emp;
Name Annual Salary
---------- ---
SMITH 9600
ALLEN 19200
WARD 15000
JONES 35700
MARTIN 15000
BLAKE 34200
CLARK 29400
SCOTT 36000
KING 60000

```
TURNER                 18000
ADAMS                  13200
JAMES                  11400
FORD                   36000
MILLER                 15600
```

where 子句中使用数值

```
select ename,job,sal from emp where sal > 2000;
ENAME      JOB                       SAL
---------- --------- ----------------------------------------
JONES      MANAGER                    2975
BLAKE      MANAGER                    2850
CLARK      MANAGER                    2450
SCOTT      ANALYST                    3000
KING       PRESIDENT                  5000
FORD       ANALYST                    3000
```

where 子句中使用字符值

```
select ename,job,sal from emp where job='MANAGER';
ENAME      JOB                       SAL
---------- --------- ----------------------------------------
JONES      MANAGER                    2975
BLAKE      MANAGER                    2850
CLARK      MANAGER                    2450
```

where 子句中使用日期

```
select ename,sal,hiredate from emp where hiredate>'01-1 月-82';
ENAME                  SAL HIREDATE
---------- ---------- ------------------------------
SCOTT                  3000 19-4 月  -87
ADAMS                  1100 23-5 月  -87
MILLER                 1300 23-1 月  -82
```

where 子句中使用 between...and 操作符

```
select ename,sal,job,deptno from emp where sal between 2000 and 3000;
ENAME                  SAL JOB                 DEPTNO
---------- ---------- --------- ------------------------------------
JONES                  2975 MANAGER                 20
BLAKE                  2850 MANAGER                 30
CLARK                  2450 MANAGER                 10
SCOTT                  3000 ANALYST                 20
FORD                   3000 ANALYST                 20
```

where 子句中使用 like 操作符

```
select ename,sal from emp where ename like 'S%';
```

通配符%表示 0 个或多个字符，_表示单个字符

```
ENAME                SAL
---------- ------------------------------
SMITH                 800
SCOTT                3000
```

```
select ename,sal from emp where ename like '__O%';
```

(注意：两个下划线和一个字母 O，查询第名字第三个字母为 O 的雇员)

```
ENAME                SAL
---------- ---------------------------------
SCOTT                3000
```

where 中使用 IN 操作符

```
select ename,sal,job from emp where job IN ('CLERK','MANAGER');
ENAME                SAL JOB
---------- ---------- -------------------------
SMITH                 800 CLERK
JONES                2975 MANAGER
BLAKE                2850 MANAGER
CLARK                2450 MANAGER
ADAMS                 1100 CLERK
JAMES                 950 CLERK
MILLER                1300 CLERK
```

where 中使用 IS NULL 操作符

与 NULL 值比较是，不要使用=和<>操作符

```
select ename from emp where mgr IS NULL;
ENAME
----------
KING
```

逻辑操作符 AND OR NOT

```
select ename,sal,job,deptno from emp where deptno=20 AND job='CLERK';
ENAME                SAL JOB              DEPTNO
---------- ---------- --------- ---------------------------------------
SMITH                 800 CLERK                20
ADAMS                 1100 CLERK               20
```

```
select ename,sal,job,deptno from emp where sal>2500 OR job='MANAGER';
ENAME                SAL JOB              DEPTNO
---------- ---------- --------- ---------------------------------------
JONES                2975 MANAGER              20
BLAKE                2850 MANAGER              30
CLARK                2450 MANAGER              10
SCOTT                 3000 ANALYST             20
KING                 5000 PRESIDENT            10
```

FORD 3000 ANALYST 20

select ename,sal,comm,deptno from emp where comm IS NOT NULL;

ENAME	SAL	COMM	DEPTNO
ALLEN	1600	300	30
WARD	1250	500	30
MARTIN	1250	1400	30
TURNER	1500	0	30

优先级：NOT 最高，AND 其次，OR 最低

select ename,sal,job from emp
where (job='CLERK' OR job='MANAGER') AND sal BETWEEN 1000 AND 3000;

ENAME	SAL	JOB
JONES	2975	MANAGER
BLAKE	2850	MANAGER
CLARK	2450	MANAGER
ADAMS	1100	CLERK
MILLER	1300	CLERK

练习：

查询部门编号为 100 的雇员的名字和薪水

select first_name,department_id,salary from employees where department_id=100;

FIRST_NAME	DEPARTMENT_ID	SALARY
Nancy	100	12000
Daniel	100	9000
John	100	8200
Ismael	100	7700
Jose Manuel	100	7800
Luis	100	6900

显示 JOB_ID 包含 CLERK 的所有雇员的名字和岗位

select first_name,job_id,department_id from employees where job_id LIKE '%CLERK%';

部分查询结果如下：

FIRST_NAME	JOB_ID	DEPARTMENT_ID
Winston	SH_CLERK	50
Jean	SH_CLERK	50
Nandita	SH_CLERK	50
Sigal	PU_CLERK	30
Guy	PU_CLERK	30
Karen	PU_CLERK	30
Julia	ST_CLERK	50

显示 COMMISSION_PCT 非空的雇员的姓和 COMMISSION_PCT
select last_name,commission_pct from employees where commission_pct IS NOT NULL;
部分结果如下：

Doran	.3
Sewall	.25
Vishney	.25
Greene	.15
Marvins	.1
Lee	.1
Ande	.1
Banda	.1
Ozer	.25
Bloom	.2
Fox	.2

显示部门 80 和 90 中工资高于 9000 的所有雇员的姓名，工资和部门
select first_name,last_name,department_id,salary from employees
where (department_id=80 OR department_id=80) AND salary>9000;

Eleni	Zlotkey	80	10500
Peter	Tucker	80	10000
David	Bernstein	80	9500
Janette	King	80	10000
Patrick	Sully	80	9500
Clara	Vishney	80	10500
Danielle	Greene	80	9500
Lisa	Ozer	80	11500
Harrison	Bloom	80	10000
Tayler	Fox	80	9600
Ellen	Abel	80	11000
John	Russell	80	14000
Karen	Partners	80	13500
Alberto	Errazuriz	80	12000
Gerald	Cambrault	80	11000

第二章 排序数据

单列排序
当 select 同时包含多条语句时，order by 必须是最后一条语句

单列升序排序 ASC（NULL 行在最后）
select ename,sal from emp order by sal ASC;

ENAME	SAL
---	---

SMITH 800
JAMES 950
ADAMS 1100
WARD 1250
MARTIN 1250
MILLER 1300
TURNER 1500
ALLEN 1600
CLARK 2450
BLAKE 2850
JONES 2975

单列降序排序 DESC（NULL 行在最前）
select ename,sal from emp order by sal desc;
KING 5000
SCOTT 3000
FORD 3000
JONES 2975
BLAKE 2850
CLARK 2450
ALLEN 1600
TURNER 1500
MILLER 1300
WARD 1250
MARTIN 1250

列别名排序
select ename,sal*12 年收入 from emp order by 年收入 desc;
KING 60000
SCOTT 36000
FORD 36000
JONES 35700
BLAKE 34200
CLARK 29400
ALLEN 19200
TURNER 18000
MILLER 15600
WARD 15000
MARTIN 15000

非选择列表列排序
select ename from emp order by sal desc;
ENAME

KING
SCOTT

FORD
JONES
BLAKE
CLARK
ALLEN
TURNER
MILLER
WARD
MARTIN

多列排序
select ename,deptno,sal from emp order by deptno ASC, sal DESC;

ENAME	DEPTNO	SAL
KING	10	5000
CLARK	10	2450
MILLER	10	1300
SCOTT	20	3000
FORD	20	3000
JONES	20	2975
ADAMS	20	1100
SMITH	20	800
BLAKE	30	2850
ALLEN	30	1600
TURNER	30	1500
WARD	30	1250
MARTIN	30	1250
JAMES	30	950

第三章 SQL单行函数

数学函数：ROUND , TRUNC , MOD
字符函数: UPPER , LOWER , INITCAP , SUNSTR , CONCAT , INSTR , LPAD
日期函数： MONTHS_BETWEEN , NEXT_DAY LAST_DAY , ROUND
TRUNC , ADD_MONTHS
转换函数： TO_CHAR , TO_DATE , TO_NUMBER
正则表达式，对象函数和集合函数
函数： NVL , NVL2 , NULLIF , DECODE

数字函数
ROUND—四舍五入
select sum(sal),round(avg(sal)) from emp where deptno=10;
SUM(SAL) ROUND(AVG(SAL))

---------- --
8750 2917

TRUNC----保留小数的位数
select sum(sal),trunc(avg(sal),1) from emp where deptno=10;
SUM(SAL) TRUNC(AVG(SAL),1)
---------- --
8750 2916.6

MOD----取模，余数
select ename,sal,mod(sal,1000) from emp where deptno=10;
ENAME SAL MOD(SAL,1000)
---------- ---------- --
CLARK 2450 450
KING 5000 0
MILLER 1300 300

其它数学函数：
绝对值：abs(n) , e 的 N 次幂：exp(n) ， 小于等于 n 的最大整数：floor(n)
M 的 n 次幂：power(m,n) ， 检测正负：SIGN(n)

字符函数
UPPER/LOWER----转换大小写
select empno,sal,job from emp where upper(ename)='SMITH';
EMPNO SAL JOB
---------- ---------- ----------
7369 800 CLERK

CONCAT----字符串连接
select concat(concat(ename,';'),sal) from emp where deptno=10;
CONCAT(CONCAT(ENAME,';'),SAL)

CLARK;2450
KING;5000
MILLER;1300

INSTR---确定特定字符串在源字符串中的位置（索引，index）
select instr('Great Wall In China','China') from dual;
INSTR('GREATWALLINCHINA','CHINA')

15

SUBSTR----提取子字符串
select substr('Great Wall In China',7,30) from dual;
SUBSTR('GREAT

Wall In China

LPAD/RPAD----在字符串左端/右端添加字符串
select lpad('In China',19,'Great Wall ') from dual;
LPAD('INCHINA',19,'

Great Wall In China

其它字符函数：
ASCII(char) ：返回 ascii 码 ， CHR(n)：返回 ascii 码值对应的字符
LENGTH(char)：返回字符串长度

日期时间函数
MONTHS_BETWEEN----月份差
select ename,trunc(months_between(sysdate,hiredate)/12) work_year from emp where deptno=10;
ENAME WORK_YEAR

CLARK 26
KING 26
MILLER 26

ADD_MONTHS----特定日期之前或之后月份对应的日期
select ename,add_months(hiredate,20*12) "20 周年" from emp where deptno=20;
ENAME 20 周年

SMITH 17-12 月-00
JONES 02-4 月 -01
SCOTT 19-4 月 -07
ADAMS 23-5 月 -07
FORD 03-12 月-01

NEXT_DAY----特定日期之后对应日期
select sysdate,next_day(sysdate,'星期五') Friday from dual;
SYSDATE FRIDAY

22-5 月 -08 23-5 月 -08

EXTRACT----从日期时间值中抽取信息（YEAR,MONTH,DAY,HOUR）
select extract(month from sysdate) current_month from dual;
CURRENT_MONTH

5

转换函数

TO_CHAR----转换数字数据
select ename,to_char(sal,'L99999.99') salary from emp where deptno=20;
ENAME SALARY
---------- ----------
SMITH RMB800.00
JONES RMB2975.00
SCOTT RMB3000.00
ADAMS RMB1100.00
FORD RMB3000.00

TO_DATE
select ename,hiredate from emp where hiredate>to_date('1981-12-31','YYYY-MM-DD');
ENAME HIREDATE
---------- ----------
SCOTT 19-4 月 -87
ADAMS 23-5 月 -87
MILLER 23-1 月 -82

TO_NUMBER
select ename,sal from emp where sal>to_number('RMB3000.00','L99999.99');
ENAME SAL
---------- ----------
KING 5000

第四章 操纵数据

增加数据----INSERT
增加单行数据：
1. 不使用列列表，那么按列顺序依次插入
insert into dept values(50,'TRAIN','BOSTON');
select deptno,dname,loc from dept;
DEPTNO DNAME LOC
------- -------------- --------------------------------
10 ACCOUNTING NEW YORK
20 RESEARCH DALLAS
30 SALES CHICAGO
40 OPERATIONS BOSTON
50 TRAIN BOSTON

2. 使用列列表
必须包含主键列和 NOT NULL 列，增加指定列
insert into emp (empno,ename,job,hiredate) values(1234,'JOHN','CLERK','01-3 月-86');

3. 使用特定格式插入日期

```
insert into emp (empno,ename,job,hiredate)
  values(1356,'MARY','CLERK',
  to_date('1983-10-20','YYYY-MM-DD'));
```

4. 使用 default 提供数据

```
insert into dept values(60,'MARKET',DEFAULT);
select * from dept where deptno=60;
DEPTNO DNAME                LOC
------- -------------- ----------------------
        60 MARKET
```

5. 使用变量插入数据

脚本 loademp.sql：

```
accept no prompt '请输入雇员号：'
accept name prompt '请输入雇员名：'
accept title prompt '请输入雇员岗位：'
accept d_no prompt '请输入部门号：'
insert into emp(empno,ename,job,hiredate,deptno)
values(&no,'&name','&title',SYSDATE,&d_no);
```

运行：SQL> @c:\loademp

使用 insert 插入数据时，不仅要为 NOT NULL 列和主键列提供数据，其它数据也必须满足约束条件

使用子查询

```
insert into (select empno,ename,sal,deptno from emp where deptno>20)
values(1112,'MARY',2000,30);
```

多表插入

创建表 SAL_H,MGR_H

```
create table sal_h as select ename,hiredate,sal from emp where 1=0;
create table mgr_h as select ename,mgr,sal from emp where 1=0;
```

无条件 INSERT ALL

```
insert all
into sal_h values(ename,hiredate,sal)
into mgr_h values(ename,mgr,sal)
select ename,hiredate,mgr,sal from emp;
```

已创建 30 行

有条件 INSERT ALL

```
insert all
when sal>1000 then
into sal_h values(ename,hiredate,sal)
when mgr>7700 then
into mgr_h values(ename,mgr,sal)
```

select ename,hiredate,mgr,sal from emp;
已创建 19 行

有条件 INSERT FIRST
将子查询满足的首个条件插入表
insert first
when sal>3000 then
into sal_h values(ename,hiredate,sal)
when sal>1000 then
into mgr_h values(ename,mgr,sal)
select ename,hiredate,mgr,sal from emp;
已创建 13 行

更新数据----UPDATE
更新数字列时，可以直接提供数字值，也可以用单引号引住
更新字符和日期列时，必须用单引号引住
更新数据时，必须满足约束规则
更新数据时，数据必须与列的数据类型匹配

表达式更新
update emp set sal=2460 where ename='SCOTT';
update emp set sal=sal*1.1,comm=sal*0.1 where deptno=20;
update emp set hiredate=to_date('1984/01/01','YYYY/MM/DD') where empno=7788;
update emp set job=default where ename='SCOTT';

使用子查询更新数据
使用值更新关联数据：
update emp set (job,sal,comm)=(
select job,sal,comm from emp where ename='SMITH')
where ename='SCOTT';

复制表数据：
update employee set deptno=
update employee set deptno=
(select deptno from emp where empno=7788)
where job=(select job from emp where empno=7788);

使用 merge 语句
用于根据条件确定是执行 update 还是 insert，
如果行存在，执行 update；
如果行不存在，执行 insert
create table new as select * from emp where 1=0;

insert into new (empno,ename)
select empno,ename from emp where deptno=10;

```
merge into new n using emp e
on (n.empno=e.empno)
when matched then update set n.sal=e.sal
when not matched then
insert (n.empno,n.ename,n.sal,n.comm)
values (e.empno,e.ename,e.sal,e.comm);
```

删除数据---DELETE

删除满足条件的数据：

```
delete from emp where ename='SMITH';
```

删除表的所有数据：

```
delete from new;
```

截断表----truncate table

```
truncate table employee;
```

注：delete 可以回退，但 truncate table 不能回退，它不仅删除数据，而且释放表空间

使用子查询删除数据

```
create table new as select * from emp;
delete from new where deptno=
(select deptno from dept where dname='SALES');
```

事务控制---DML 语句

当执行事务操作语句时，oracle 会给表加上锁，防止其它用户改动表结构，和 JAVA 线程同步类似

在提交事务之前，其它用户读取的还是原来的数据

提交事务----commit

```
update emp set sal=2000 where ename='SCOTT';
commit;
```

回退事务

设置保存点----SAVEPOINT

取消部分事务----rollback to point；

取消全部事务----rollback；

设置只读事务----set transaction read only；（必须是第一条语句）

设置顺序事务----set transaction isolation level serializable；（必须是第一条语句）

第五章 连接查询

相等连接
显示所有雇员的名称，工资和部门名称
Dept 表的 deptno 为主键，emp 的 deptno 为外键
select emp.ename,emp.sal,dept.dname from emp,dept
where emp.deptno=dept.deptno;
结果如下：

```
ENAME                 SAL DNAME
---------- ---------- --------------------------
ALLEN                1600 SALES
WARD                 1250 SALES
JONES               3272.5 RESEARCH
MARTIN               1250 SALES
BLAKE                2850 SALES
CLARK                2450 ACCOUNTING
SCOTT                2000 RESEARCH
KING                 5000 ACCOUNTING
TURNER               1500 SALES
ADAMS                1210 RESEARCH
JAMES                950 SALES
FORD                3300 RESEARCH
MILLER              1300 ACCOUNTING
MARY                 2000 SALES
```

连接查询中使用 AND 指定其它条件
select emp.ename,emp.sal,dept.dname from emp,dept
where emp.deptno=dept.deptno AND dept.deptno=10;

```
ENAME                 SAL DNAME
---------- ---------- ---------------------------------
CLARK                2450 ACCOUNTING
KING                5000 ACCOUNTING
MILLER               1300 ACCOUNTING
```

在连接查询中使用表别名
select e.ename,d.loc from emp e,dept d
where e.deptno=d.deptno AND e.deptno=30;
结果如下：

```
ENAME          LOC
---------- ------------------------
ALLEN          CHICAGO
WARD           CHICAGO
MARTIN         CHICAGO
BLAKE          CHICAGO
TURNER         CHICAGO
```

JAMES CHICAGO
MARY CHICAGO

不等连接

```
select e.ename,e.sal,s.grade from emp e,salgrade s
where e.sal between s.losal and s.hisal;
```

结果如下：

```
ENAME                SAL      GRADE
---------- ---------- ----------
JAMES                950          1
WARD                1250          2
MARTIN              1250          2
ADAMS               1210          2
MILLER              1300          2
ALLEN               1600          3
SCOTT               2000          3
TURNER              1500          3
MARY                2000          3
BLAKE               2850          4
CLARK               2450          4
JONES             3272.5          5
KING                5000          5
FORD                3300          5
```

自连接

同一张表的连接查询(必须定义表别名)

```
select worker.ename||'s manager is '||manager.ename
from emp worker,emp manager
where worker.mgr=manager.empno;
WORKER.ENAME||'SMANAGERIS'||MANA
------------------------------------------------------------
ALLEN's manager is BLAKE
WARD's manager is BLAKE
JONES's manager is KING
MARTIN's manager is BLAKE
BLAKE's manager is KING
CLARK's manager is KING
SCOTT's manager is JONES
TURNER's manager is BLAKE
ADAMS's manager is SCOTT
JAMES's manager is BLAKE
FORD's manager is JONES
MILLER's manager is CLARK
```

外连接
外连接使用(+)操作符，只适用于列
select d.dname,e.ename from dept d,emp e
where d.deptno(+)=e.deptno AND d.deptno(+)=10;
结果如下：

```
DNAME          ENAME
-------------- ----------------------------
               ALLEN
               WARD
               JONES
               MARTIN
               BLAKE
ACCOUNTING     CLARK
               SCOTT
ACCOUNTING     KING
                TURNER
                ADAMS
                JAMES
                FORD
ACCOUNTING     MILLER
                MARY
```

SQL-99 连接
1. CROSS JOIN 连接，生成两张表的笛卡尔积
select d.dname,e.ename from dept d CROSS JOIN emp e;
结果很长，

```
DNAME          ENAME
-------------- ----------------------------
ACCOUNTING      ALLEN
RESEARCH        ALLEN
SALES          ALLEN
OPERATIONS      ALLEN
ACCOUNTING      WARD
RESEARCH        WARD
SALES          WARD
OPERATIONS      WARD
ACCOUNTING      JONES
RESEARCH        JONES
SALES          JONES
……
```

已选择 56 行

2. 建立 NATURAL JOIN 连接，基于同名列执行相等连接
select e.ename,e.sal,d.dname from dept d NATURAL JOIN emp e;

```
ENAME            SAL DNAME
```

---------- ---------- ----------
ALLEN 1600 SALES
WARD 1250 SALES
JONES 3272.5 RESEARCH
MARTIN 1250 SALES
BLAKE 2850 SALES
CLARK 2450 ACCOUNTING
SCOTT 2000 RESEARCH
KING 5000 ACCOUNTING
TURNER 1500 SALES
ADAMS 1210 RESEARCH
JAMES 950 SALES
FORD 3300 RESEARCH
MILLER 1300 ACCOUNTING
MARY 2000 SALES
已选择 14 行

3. 使用 using 子句建立相等连接

```
select e.ename,d.dname from dept d JOIN emp e USING(deptno);
```

ENAME DNAME
---------- ----------
ALLEN SALES
WARD SALES
JONES RESEARCH
MARTIN SALES
BLAKE SALES
CLARK ACCOUNTING
SCOTT RESEARCH
KING ACCOUNTING
TURNER SALES
ADAMS RESEARCH
JAMES SALES
FORD RESEARCH
MILLER ACCOUNTING
MARY SALES

4. 使用 ON 子句建立连接，连接列名称不同时

```
select e.ename,e.sal,d.dname from emp e JOIN dept d
ON e.deptno=d.deptno AND e.deptno=10;
```

ENAME SAL DNAME
---------- ---------- ----------
CLARK 2450 ACCOUNTING
KING 5000 ACCOUNTING
MILLER 1300 ACCOUNTING

左连接
----返回满足条件的数据和不满足条件的左边表的数据
select a.dname,b.ename from dept a LEFT JOIN emp b
ON a.deptno=b.deptno AND a.deptno=10;

DNAME	ENAME
ACCOUNTING	CLARK
ACCOUNTING	KING
ACCOUNTING	MILLER
RESEARCH	
SALES	
OPERATIONS	

右连接，与左连接类似
select a.dname,b.ename from dept a RIGHT JOIN emp b
ON a.deptno=b.deptno AND a.deptno=10;

DNAME	ENAME
ACCOUNTING	CLARK
ACCOUNTING	KING
ACCOUNTING	MILLER
	JONES
	FORD
	ADAMS
	SCOTT
	ALLEN
	JAMES
	TURNER
	MARTIN
	MARY
	BLAKE
	WARD

完全连接
返回满足条件的数据，和不满足条件的左边和右边的数据
select a.dname,b.ename from dept a FULL JOIN emp b
ON a.deptno=b.deptno AND a.deptno=10;

DNAME	ENAME
ACCOUNTING	CLARK
ACCOUNTING	KING
ACCOUNTING	MILLER
RESEARCH	
SALES	
OPERATIONS	
	ALLEN

WARD
JONES
MARTIN
BLAKE
SCOTT
TURNER
ADAMS
JAMES
FORD
MARY

注意：左连接，右连接和完全连接对照看

第六章 数据分组

MAX 和 MIN
取列或表达式的最大最小值
select max(sal) 最高工资,min(sal) 最低工资 from emp;
最高工资 最低工资
-------- --------------------------
5000 950

AVG 和 SUM
select avg(sal) 平均工资,sum(sal) 总计工资 from emp;
平均工资 总计工资
---------- ------------------------
2138.3571 29932.5

COUNT，取得总计行数
select count(*) 雇员总数 from emp;
雇员总数

14

COUNT 等分组函数会忽略 NULL 行
select count(comm) 补助非空的雇员数 from emp;
补助非空的雇员数

8

VARIANCE，取得列或表达式的方差
STDDEV，取得列或表达式的标准差
select variance(sal) 方差,stddev(sal) 标准差 from emp;

方差 标准差

---------- ------------------------

1286084.79 1134.05678

GROUP BY

Where 子句指定条件，group by 用于指定分组，having 限制分组结果显示，order by 用于排序数据

单列分组：

select deptno 部门代码,avg(sal) 部门平均工资 from emp group by deptno;

部门代码 部门平均工资

-------- -------------------------

10 2916.66667

20 2445.625

30 1628.57143

多列分组：

select deptno,job,avg(sal),max(sal) from emp group by deptno,job;

DEPTNO JOB AVG(SAL) MAX(SAL)

------ --------- ---------- --------------------------------------

10 CLERK 1300 1300

10 MANAGER 2450 2450

10 PRESIDENT 5000 5000

20 CLERK 1605 2000

20 ANALYST 3300 3300

20 MANAGER 3272.5 3272.5

30 2000 2000

30 CLERK 950 950

30 MANAGER 2850 2850

30 SALESMAN 1400 1600

改变分组排序结果：

select deptno,sum(sal) from emp group by deptno order by sum(sal) DESC;

DEPTNO SUM(SAL)

------ -------------------------

30 11400

20 9782.5

10 8750

使用 HAVING 限制分组结果：

select deptno,avg(sal),max(sal) from emp group by deptno having avg(sal)<2500;

DEPTNO AVG(SAL) MAX(SAL)

------ ---------- ---------------------------------

20 2445.625 3300

30 1628.57143 2850

分组函数注意事项：

1. 只能出现在选择列表，ORDER BY 子句和 HAVING 子句中
2. 分组函数忽略 NULL 行
3. 可以指定 ALL 或 DISTINCT 指定是否统计重复的行
4. 如果同时包含，ORDER BY 子句必须放在最后
5. 选择列表中出现的列表，表达式和分组函数都必须出现在 GROUP BY 子句中
6. 限制分组结果只能使用 HAVING 子句

ROLLUP
额外生成横向小记和总计
select deptno,job,avg(sal) from emp group by rollup(deptno,job);

DEPTNO	JOB	AVG(SAL)
10	CLERK	1300
10	MANAGER	2450
10	PRESIDENT	5000
10		2916.66667
20	CLERK	1605
20	ANALYST	3300
20	MANAGER	3272.5
20		2445.625
30		2000
30	CLERK	950
30	MANAGER	2850
30	SALESMAN	1400
30		1628.57143
		2138.03571

CUBE
额外生成横向统计，纵向统计和总计统计
select deptno,job,avg(sal) from emp group by cube(deptno,job);

DEPTNO	JOB	AVG(SAL)
		2000
		2138.03571
	CLERK	1365
	ANALYST	3300
	MANAGER	2857.5
	SALESMAN	1400
	PRESIDENT	5000
10		2916.66667
10	CLERK	1300
10	MANAGER	2450
10	PRESIDENT	5000
20		2445.625
20	CLERK	1605
20	ANALYST	3300

```
20 MANAGER          3272.5
30                   2000
30                1628.57143
30 CLERK             950
30 MANAGER           2850
30 SALESMAN          1400
```

GROUPING 函数

确定统计结果是否使用了特定的列

```
select deptno,job,sum(sal),grouping(deptno),grouping(job) from emp
group by rollup(deptno,job);
DEPTNO JOB              SUM(SAL) GROUPING(DEPTNO) GROUPING(JOB)
------- --------- ---------- --------------- ---------------------------------------------------------------
    10 CLERK              1300               0                 0
    10 MANAGER             2450               0                 0
    10 PRESIDENT           5000               0                 0
    10                     8750               0                 1
    20 CLERK               3210               0                 0
    20 ANALYST             3300               0                 0
    20 MANAGER             3272.5             0                 0
    20                     9782.5             0                 1
    30                     2000               0                 0
    30 CLERK               950                0                 0
    30 MANAGER              2850               0                 0
    30 SALESMAN            5600               0                 0
    30                     11400              0                 1
                           29932.5              1                 1
```

ROLLUP 只用复合列

例如 group by rollup(a,b,c)的操作结果等同于 group by(a,b,c),group by(a,b),group by(a)和 group by()的并集

```
select deptno,job,sum(sal) from emp group by rollup((deptno,job));
DEPTNO JOB              SUM(SAL)
------ --------- ---------------------------------
10 CLERK               1300
10 MANAGER              2450
10 PRESIDENT           5000
20 CLERK               3210
20 ANALYST             3300
20 MANAGER             3272.5
30                     2000
30 CLERK               950
30 MANAGER              2850
30 SALESMAN            5600
                       29932.5
```

CUBE 中使用复合列

```
select deptno,job,avg(sal) from emp group by cube((deptno,job));
DEPTNO JOB              AVG(SAL)
------ --------- ------------------------------
    10 CLERK                1300
    10 MANAGER              2450
    10 PRESIDENT            5000
    20 CLERK                1605
    20 ANALYST              3300
    20 MANAGER              3272.5
    30                      2000
    30 CLERK                 950
    30 MANAGER              2850
    30 SALESMAN             1400
                            2138.036
```

使用 GROUPING SETS 操作符合并多个分组的统计结果

显示部门平均工资：

```
select deptno,avg(sal) from emp group by deptno;
DEPTNO     AVG(SAL)
------ --------------------------
    10 2916.66667
    20     2445.625
    30 1628.57143
```

显示岗位平均工资：

```
select job,avg(sal) from emp group by job;
JOB              AVG(SAL)
--------- ------------------------
ANALYST              3300
CLERK                1365
MANAGER            2857.5
PRESIDENT            5000
SALESMAN             1400
                     2000
```

显示部门和岗位平均工资：

```
select deptno,job,avg(sal)    from emp group by grouping sets(deptno,job);
DEPTNO JOB              AVG(SAL)
------ --------- ------------------------------
    10                  2916.66667
    20                    2445.625
    30                  1628.57143
       ANALYST              3300
       CLERK                1365
       MANAGER            2857.5
       PRESIDENT            5000
```

SALESMAN 1400
2000

连接分组

连接多个 grouping sets
select deptno,job,mgr,sum(sal) from emp group by grouping sets(deptno),grouping sets(job,mgr);

DEPTNO	JOB	MGR	SUM(SAL)
10	CLERK		1300
10	MANAGER		2450
10	PRESIDENT		5000
20	CLERK		3210
20	ANALYST		3300
20	MANAGER		3272.5
30			2000
30	CLERK		950
30	MANAGER		2850
30	SALESMAN		5600
10			5000
10		7782	1300
10		7839	2450
20		7566	5300
20		7788	1210
20		7839	3272.5
30			2000
30		7698	6550
30		7839	2850

连接 rollup
select deptno,job,mgr,sum(sal) from emp group by deptno,rollup(job,mgr);

DEPTNO	JOB	MGR	SUM(SAL)
10	CLERK	7782	1300
10	CLERK		1300
10	MANAGER	7839	2450
10	MANAGER		2450
10	PRESIDENT		5000
10	PRESIDENT		5000
10			8750
20	CLERK	7566	2000
20	CLERK	7788	1210
20	CLERK		3210
20	ANALYST	7566	3300
20	ANALYST		3300
20	MANAGER	7839	3272.5

20	MANAGER		3272.5
20			9782.5
30			2000
30			2000
30	CLERK	7698	950
30	CLERK		950
30	MANAGER	7839	2850
30	MANAGER		2850
30	SALESMAN	7698	5600
30	SALESMAN		5600
30			11400

连接 CUBE

```
select deptno,job,mgr,sum(sal) from emp group by deptno,cube(job,mgr);
```

DEPTNO	JOB	MGR	SUM(SAL)
10			5000
10			8750
10		7782	1300
10		7839	2450
10	CLERK		1300
10	CLERK	7782	1300
10	MANAGER		2450
10	MANAGER	7839	2450
10	PRESIDENT		5000
10	PRESIDENT		5000
20			9782.5
20		7566	5300
20		7788	1210
20		7839	3272.5
20	CLERK		3210
20	CLERK	7566	2000
20	CLERK	7788	1210
20	ANALYST		3300
20	ANALYST	7566	3300
20	MANAGER		3272.5
20	MANAGER	7839	3272.5
30			2000
30			2000
30			2000
30			11400
30		7698	6550
30		7839	2850
30	CLERK		950
30	CLERK	7698	950
30	MANAGER		2850

```
30 MANAGER          7839          2850
30 SALESMAN                       5600
30 SALESMAN         7698          5600
```

第七章 子查询

单行子查询：只返回一行数据的子查询语句

```
select ename,sal,deptno from emp where deptno=
(select deptno from emp where ename='SCOTT')
AND ename<>'SCOTT';
ENAME                SAL       DEPTNO
---------- ---------- ----------------------------------
JONES                3272.5            20
ADAMS                1210              20
FORD                 3300              20
```

多行子查询，IN,ALL,ANY

IN----处理匹配子查询任一个值的行

```
select ename,job,sal,deptno from emp where job IN
(select distinct job from emp where deptno=10);
ENAME          JOB                SAL       DEPTNO
---------- --------- ---------- ------------------------------------
SCOTT          CLERK              2000              20
ADAMS          CLERK              1210              20
JAMES          CLERK               950              30
MILLER         CLERK              1300              10
JONES          MANAGER            3272.5            20
BLAKE          MANAGER            2850              30
CLARK          MANAGER            2450              10
KING           PRESIDENT          5000              10
```

ALL----必须匹配所有子查询结果，不能单独使用

```
select ename,sal,deptno from emp where sal>all
(select sal from emp where deptno=30);
ENAME                SAL       DEPTNO
---------- ---------- ----------------------------------
JONES                3272.5            20
KING                 5000              10
FORD                 3300              20
```

ANY----匹配子查询任一结果，不能单独使用，和 IN 类似

```
select ename,sal,deptno from emp where sal>any
(select sal from emp where deptno=30);
ENAME                SAL       DEPTNO
```

---------- ---------- ----------

ALLEN	1600	30
WARD	1250	30
JONES	3272.5	20
MARTIN	1250	30
BLAKE	2850	30
CLARK	2450	10
SCOTT	2000	20
KING	5000	10
TURNER	1500	30
ADAMS	1210	20
FORD	3300	20
MILLER	1300	10
MARY	2000	30

多列子查询

多列子查询示例：

```
select ename,job,sal,deptno from emp where(deptno,job)=
(select deptno,job from emp where ename='WARD');
```

ENAME	JOB	SAL	DEPTNO
ALLEN	SALESMAN	1600	30
WARD	SALESMAN	1250	30
MARTIN	SALESMAN	1250	30
TURNER	SALESMAN	1500	30

成对比较：要求多个列的数据同时匹配

例子，显示工资和补助与部门 30 的工资补助完全匹配的所有雇员：

```
update emp set sal=1500,comm=300 where ename='CLARK';
update emp set sal=1600,comm=300 where ename='SCOTT';
select ename,sal,comm,deptno from emp where(sal,nvl(comm,-1))
IN (select sal,nvl(comm,-1) from emp where deptno=30);
```

非成对比较：不要求多列数据同时匹配

例子，显示工资匹配于部门 30 的工资列表，补助匹配于部门 30 补助列表的所有雇员：

```
select ename,sal,comm,deptno from emp
where sal IN (select sal from emp where deptno=30)
AND    nvl(comm,-1) IN
(select nvl(comm,-1) from emp where deptno=30);
```

ENAME	SAL	COMM	DEPTNO
JAMES	950		30
MARY	2000		30
BLAKE	2850		30
TURNER	1500	0	30
CLARK	1500	300	10

ALLEN 1600 300 30
SCOTT 1600 300 20
WARD 1250 500 30
MARTIN 1250 1400 30

相关子查询

select 中使用相关子查询：
显示工资高于部门平均工资的所有雇员名，工资和部门号
select ename,sal,deptno from emp outer where sal>
(select avg(sal) from emp where deptno=outer.deptno);

ENAME	SAL	DEPTNO
JONES	3272.5	20
BLAKE	2850	30
KING	5000	10
FORD	3300	20
MARY	2000	30

update 中使用相关子查询
alter table emp ADD dept_name varchar2(20);
update emp e set dept_name=
(select dname from dept d where d.deptno=e.deptno);
已更新 14 行

在 delete 中使用相关查询
delete from emp e where empno=
(select empno from emp_copy where empno=e.empno);
注：emp_copy 是 emp 表的拷贝

使用 exists 操作符
显示在 NEW YORK 工作的所有雇员及其工资和部门号：
select ename,sal,deptno from emp e where exists
(select 1 from dept where deptno=e.deptno AND loc='NEW YORK');

ENAME	SAL	DEPTNO
CLARK	1500	10
KING	5000	10
MILLER	1300	10

使用 not exists 操作符
显示不在 NEW YORK 工作的所有雇员及工资和部门号：
select ename,sal,deptno from emp e where not exists
(select 1 from dept where deptno=e.deptno AND loc='NEW YORK');

ENAME	SAL	DEPTNO

```
ALLEN                1600            30
WARD                 1250            30
JONES              3272.5            20
MARTIN               1250            30
BLAKE                2850            30
SCOTT                1600            20
TURNER               1500            30
ADAMS                1210            20
JAMES                950             30
FORD                 3300            20
MARY                 2000            30
```

DDL 语句中使用子查询

```
create table new_emp(id,name,sal,job,deptno) AS select empno,ename,sal,job,deptno from emp;
```

FROM 子句中使用子查询，必须给子查询指定别名

```
select ename,job,sal from emp,
(select deptno,avg(sal) avgsal from emp group by deptno)
dept where emp.deptno=dept.deptno AND sal>dept.avgsal;
ENAME          JOB                       SAL
---------- --------- ---------------------------------
KING           PRESIDENT                 5000
JONES          MANAGER                 3272.5
FORD           ANALYST                   3300
MARY                                     2000
BLAKE          MANAGER                   2850
```

标量子查询表达式

是指只返回一个列值的子查询，适用于：

1）decode 和 case 的条件部分和表达式部分

2）除 group by 子句之外的所有 select 其它子句

3）在 update 语句的 set 和 where 子句比较操作符的左边

示例--以部门名升序显示雇员名，工资和部门号：

```
select ename,sal,deptno from emp e order by
(select dname from dept d where e.deptno=d.deptno);
ENAME                 SAL         DEPTNO
---------- ---------- ---------------------------------
CLARK                 1500            10
KING                  5000            10
MILLER                1300            10
JONES               3272.5            20
FORD                  3300            20
ADAMS                 1210            20
SCOTT                 1600            20
ALLEN                 1600            30
JAMES                  950            30
```

```
TURNER          1500          30
MARTIN          1250          30
MARY            2000          30
BLAKE           2850          30
WARD            1250          30
```

使用 WITH 为子查询定义名称，重复使用
示例，显示部门工资总额超过所有部门平均工资总额的部门名和工资总额：

```
WITH
dept_sum AS
(select d.dname,sum(e.sal) total from dept d,emp e where d.deptno=e.deptno group by d.dname),
dept_avg_sum AS
(select sum(total)/count(*) avg_sum from dept_sum)
select dname,total from dept_sum
where total>(select avg_sum from dept_avg_sum);
DNAME                TOTAL
-------------- ----------------------------
SALES                11400
```

第八章 复杂查询

集合操作符：UNION,UNION ALL,INTERSECT,MINUS
限制：集合操作符不适用用 LOB,VARRAY 和嵌套列表
UNION,INTERSECT,MINUS 不适用于 LONG 列
必须为表达式或函数先定义别名
集合操作符优先级相同，按先后顺序引用，必须确保列数和类型匹配

为下面的示例使用，建立 manager 表和 worker 表：

```
create table manager(id,name,job,sal) AS
select empno,ename,job,sal from emp
where empno IN (select distinct nvl(mgr,0) from emp);
create table worker(id,name,job,sal) AS
select empno,ename,job,sal from emp
where empno NOT IN (select distinct nvl(mgr,0) from emp);
```

UNION：取两个操作结果集的并集
会自动去掉重复行，并以第一列结果升序排序
示例：合并显示 manager 表所有管理者和 emp 表所有的雇员代码，姓名和岗位

```
select id,name,job from manager UNION select empno,ename,job from emp;
   ID NAME           JOB
---- ---------- -------------------------
7499 ALLEN          SALESMAN
7521 WARD           SALESMAN
7566 JONES          MANAGER
```

7654 MARTIN SALESMAN
7698 BLAKE MANAGER
7782 CLARK MANAGER
7788 SCOTT CLERK
7839 KING PRESIDENT
7844 TURNER SALESMAN
7876 ADAMS CLERK
7900 JAMES CLERK
7902 FORD ANALYST
7934 MILLER CLERK

UNION ALL：同 UNION，但不消除重复，而且不排序
select id,name,job from worker UNION ALL select empno,ename,job from emp;

ID	NAME	JOB
7499	ALLEN	SALESMAN
7521	WARD	SALESMAN
7654	MARTIN	SALESMAN
7844	TURNER	SALESMAN
7876	ADAMS	CLERK
7900	JAMES	CLERK
7902	FORD	ANALYST
7934	MILLER	CLERK
7499	ALLEN	SALESMAN
7521	WARD	SALESMAN
7566	JONES	MANAGER
7654	MARTIN	SALESMAN
7698	BLAKE	MANAGER
7782	CLARK	MANAGER
7788	SCOTT	CLERK
7839	KING	PRESIDENT
7844	TURNER	SALESMAN
7876	ADAMS	CLERK
7900	JAMES	CLERK
7902	FORD	ANALYST
7934	MILLER	CLERK

INTERSECT：取交集，以第一列升序排序
select id,name,job from worker INTERSECT select empno,ename,job from emp;

ID	NAME	JOB
7499	ALLEN	SALESMAN
7521	WARD	SALESMAN
7654	MARTIN	SALESMAN
7844	TURNER	SALESMAN
7876	ADAMS	CLERK

7900 JAMES CLERK
7902 FORD ANALYST
7934 MILLER CLERK

MINUS：取差集
A MINUS B，只显示存在于第一个而不存在于第二个的结果集，第一列升序排序
select empno,ename,job from emp MINUS select id,name,job from worker;

EMPNO	ENAME	JOB
7566	JONES	MANAGER
7698	BLAKE	MANAGER
7782	CLARK	MANAGER
7788	SCOTT	CLERK
7839	KING	PRESIDENT

控制排序：列名相同时使用列名，列名不同时使用列位置，如下：
select id,name,job from worker INTERSECT select empno,ename,job from emp order by 2;

ID	NAME	JOB
7876	ADAMS	CLERK
7499	ALLEN	SALESMAN
7902	FORD	ANALYST
7900	JAMES	CLERK
7654	MARTIN	SALESMAN
7934	MILLER	CLERK
7844	TURNER	SALESMAN
7521	WARD	SALESMAN

层次查询

使用条件表达式
DECODE 函数和 CASE 表达式
select ename,deptno,sal,DECODE(deptno,10,sal*1.2,20,sal*1.1,sal) "Actual Salary" from emp;
解释：部门 10 增加 20%，部门 20 增加 10%，部门 30 保持不变

ENAME	DEPTNO	SAL	Actual Salary
ALLEN	30	1600	1600
WARD	30	1250	1250
JONES	20	3272.5	3599.75
MARTIN	30	1250	1250
BLAKE	30	2850	2850
CLARK	10	1500	1800
SCOTT	20	1600	1760
KING	10	5000	6000
TURNER	30	1500	1500
ADAMS	20	1210	1331

```
JAMES                 30          950                950
FORD                  20         3300               3630
MILLER                10         1300               1560
MARY                  30         2000               2000
```

```
select ename,deptno,sal,CASE deptno
when 10 then sal*1.2 when 20 then sal*1.1 else sal end "Actual Salary" from emp;
ENAME                DEPTNO        SAL       Actual Salary
---------- ---------- ---------- ----------------------------------------
ALLEN                 30         1600               1600
WARD                  30         1250               1250
JONES                 20         3272.5             3599.75
MARTIN                30         1250               1250
BLAKE                 30         2850               2850
CLARK                 10         1500               1800
SCOTT                 20         1600               1760
KING                  10         5000               6000
TURNER                30         1500               1500
ADAMS                 20         1210               1331
JAMES                 30          950                950
FORD                  20         3300               3630
MILLER                10         1300               1560
MARY                  30         2000               2000
```

```
select ename,sal,CASE
when sal<2000 then sal*1.2 when sal<3000 then sal*1.1 else sal end "Actual Salary" from emp;
ENAME                SAL       Actual Salary
---------- ---------- ------------------------------------
ALLEN                1600               1920
WARD                 1250               1500
JONES                3272.5             3272.5
MARTIN               1250               1500
BLAKE                2850               3135
CLARK                1500               1800
SCOTT                1600               1920
KING                 5000               5000
TURNER               1500               1800
ADAMS                1210               1452
JAMES                950                1140
FORD                 3300               3300
MILLER               1300               1560
MARY                 2000               2200
```

FlashBack 查询：查询特定时间点或者特定 SCN 的数据

第九章 建立和管理表

常用数据类型：
char(n),char(n byte)----固定长度字符串，最大长度为 2000B(以字节为单位)
char(n char)----固定长度字符串，以字符为单位
varchar2(n),varchar2(n byte)----变长字符串，以字节为单位，最长 4000B
varchar(n char)---同上，但以字符为单位
number(p,s)----数字类型，p 表示数字的总位数，s 表示小数位数
date----日期时间类型，默认格式为 DD-MON-YY
raw(n)----二进制数据，N 上限值为 2000
大对象数据类型----long raw 和 clob
timestamp----date 类型的扩展，包含日期和时间，DD-MON-YY HH.MI.SS AM
timestamp with time zone----包含时区的日期时间格式
timestamp with local time zone
inerval year to month/ interval day to second

伪列 rowid 和 rownum
rowid 用于唯一的标识表行，间接给出了表的物理位置，自动生成，可以查询
select dname,rowid from dept;

DNAME	ROWID
ACCOUNTING	AAAHIDAABAAAUDyAAA
RESEARCH	AAAHIDAABAAAUDyAAB
SALES	AAAHIDAABAAAUDyAAC
OPERATIONS	AAAHIDAABAAAUDyAAD

rownum 返回标识行数据顺序的数字值，以此类推
select rownum,dname from dept;

ROWNUM	DNAME
1	ACCOUNTING
2	RESEARCH
3	SALES
4	OPERATIONS

建表
方案名 schema 与表名 tablename 一致
在当前方案中建表：
conn scott/tiger
create table scott01(no number(2),name varchar2(10),loc varchar2(10));

在其它方案中建表，必须有 create any table 的权限：
conn system/tiger
create table scott.scott02(no number(2),name varchar2(10),loc varchar2(10));

建表时指定默认值

create table scott03(no number(2),name varchar2(10),loc varchar2(20) default '南京');

插入数据(如果没有就使用默认值):
insert into scott03(no,name) values(10,'技术处');

使用子查询建表：
create table emp01(name,salary,job,dno) AS
select ename,sal,job,deptno from emp where deptno=20;

修改表
增加列：
alter table emp01 ADD eno number(4);
修改列定义：
alter table emp01 MODIFY job varchar2(15) default 'CLERK';
删除列：
alter table emp01 DROP COLUMN dno;
修改列名：
alter table emp01 RENAME COLUMN eno TO empno;
修改表名：
RENAME emp01 TO employee;
增加注释：
COMMENT ON TABLE employee IS '存放雇员信息';
COMMENT ON COLUMN employee.name IS '描述雇员姓名';

截断和删除表
截断表：保留表结构，清空数据
TRUNCATE TABLE employee;
删除表：数据和结构都删除了
DROP TABLE employee;

显示表信息
显示当前方案所有表 USER_TABLES：
select table_name from user_tables;
TABLE_NAME

BONUS
DEPT
EMP
MANAGER
MGR_H
NEW
NEW_EMP
SALGRADE
SAL_H
WORKER
显示当前用户所有数据库对象 USER_OBJECTS：

select object_name from user_objects where object_type='TABLE';
OBJECT_NAME

BONUS
DEPT
EMP
MANAGER
MGR_H
NEW
NEW_EMP
SALGRADE
SAL_H
WORKER
显示所有表的注释 USER_TAB_COMMENTS：
select comments from user_tab_comments where table_name=’EMPLOYEE’;
显示当前用户所有表列的注释 USER_COL_COMMENTS：
select comments from user_col_comments
where table_name=’EMPLOYEE’ AND column_name=’NAME’;

第十章 使用约束

五种约束：NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY， CHECK
NOT NULL----用于确保列不能为 NULL，必须为该列提供数据
UNIQUE(唯一约束)----唯一的标识列的数据，该列可以为空 NULL，但不能重复
PRIMARY KEY(主键约束)----唯一的标识表行的数据，不能重复，也不能为 NULL
FOREIGN KEY（外键约束）----定义主从表之间的关系
CHECK(检查约束)----强制表行数据必须满足的条件

定义 NOT NULL 约束
```
create table emp01(
eno INT NOT NULL,
name varchar2(10) CONSTRAINT nn_name NOT NULL,
salary number(6,2)
);
```

定义 UNIQUE 约束
```
create table emp02(
eno INT,name varchar2(10),salary number(6,2),
CONSTRAINT u_name UNIQUE(name)
);
```

定义 PRIMARY KEY 约束
```
create table emp04(
dno INT PRIMARY KEY,
```

dname varchar2(10),loc varchar2(20)
);

定义 FOREIGN KEY 约束
create table emp05(
eno int,name varchar2(10),salary number(6,2),
dno int constraint fk_dno references dept(deptno)
);

定义 CHECK 约束
既可以在列级定义，也可以在表级定义
create table emp06(
eno int,name varchar2(10),salary number(6,2),
CHECK (salary between 1000 and 5000)
);

复合约束，只能在表级定义
create table item(
order_id number(3),item_id number(3),product varchar2(20),
primary key(order_id,item_id)
);

维护约束
增加约束：
unique，primary key，foreign key，check 约束：使用 alter table…add…子句
not null 约束：必须使用 alter table…modify…子句
alter table emp02 modify name not null;
alter table emp04 add constraint u_emp04 unique(dname);
alter table emp05 add primary key(name);
alter table emp01 add dno number(2) references dept(deptno);
alter table emp05 add check(salary between 800 and 5000);

修改约束名：
alter table emp05 rename constraint fk_dno to fkdno;

删除约束：
alter table emp05 drop constraint fkdno;
带 cascade 的删除约束：
当删除特定表的主键约束时，如果该表具有相关的从表，那么在删除主键约束时必须带有 casacade 选项，否则会显示错误信息。
alter table emp05 drop primary key cascade;

禁止约束：
alter table emp04 disable constraint u_emp04;
激活约束：
alter table emp04 enable constraint u_emp04;

显示约束信息
约束信息存放在数据字典 USER_CONSTRAINTS 中
select constraint_name,constraint_type from user_constraints where table_name='EMP';
CONSTRAINT_NAME C
------------------------------ ----------
PK_EMP P
FK_DEPTNO R
P:主键约束，R：外键约束，C：check 约束或 NOT NULL 约束，U：唯一约束
列约束信息：USER_CONS_COLUMNS
示例，显示 pk_emp 的约束列：
select column_name from user_cons_columns where constraint_name='PK_EMP';
COLUMN_NAME

EMPNO

第十一章 使用视图

视图是基于其它表或其它视图的逻辑表，本身没有数据，在视图上进行的 select，insert，update，delete 操作都是针对视图基表完成的。(!视图操作影响基本数据)

视图的作用：
限制数据访问：视图只能访问 select 语句涉及到的列；
简化复杂查询：可以基于常用的复杂查询建立视图。

视图分类：
简单视图：基于单个表建立，不包含函数，表达式和分组数据；
复杂视图：包含函数，表达式和分组数据；
连接视图：基于多个表建立的视图，简化连接查询；
只读视图：只允许 select 操作，禁止任何 DML 操作。

在视图上执行 DML 操作的原则：
如果视图包含 group by 子句，分组函数，distinct 关键字和 rowid 伪列，那么不能在该视图上执行 delete 操作和 update 操作。或者如果视图没有包含视图基表的 NOT NULL 列，也不能执行 insert 操作。

建立视图
当建立视图时，如果不提供视图列别名，那么 oracle 会自动使用子查询的列名或者列别名；如果视图子查询包含函数或表达式，那么必须定义列别名。
建立简单视图：
create view emp_vu AS select empno,ename,sal,job,deptno from emp;
insert into emp_vu values(1234,'MARY',1000,'CLERK',30);
update emp_vu set sal=2000 where empno=1234;
delete from emp_vu where empno=1234;

```
select * from emp_vu where empno=7788;
EMPNO ENAME          SAL     JOB         DEPTNO
----- ---------- ---------- --------- ------------------------------
 7788 SCOTT              1600    CLERK         20
```

建立复杂视图：

```
create view job_vu AS
select job,avg(sal) avgsal,sum(sal) sumsal,max(sal) maxsal,min(sal) minsal
from emp group by job;
select * from job_vu where job='CLERK';
JOB              AVGSAL       SUMSAL     MAXSAL     MINSAL
--------- ---------- ---------- ---------- ----------------------------------------
CLERK             1265            5060           1600            950
```

建立连接视图：

```
create view de_view as
select a.deptno,a.dname,a.loc,b.empno,b.ename,b.sal from dept a,emp b
where a.deptno=b.deptno and a.deptno=20;
select * from de_view;
DEPTNO DNAME          LOC          EMPNO    ENAME        SAL
------ -------------- ------------- ---------- ---------- ----------------------------------
  20       RESEARCH       DALLAS          7566        JONES        3272.5
  20       RESEARCH       DALLAS          7788        SCOTT        1600
  20       RESEARCH       DALLAS          7876        ADAMS        1210
  20       RESEARCH       DALLAS          7902        FORD         3300
```

建立只读视图：

```
create view emp_vur as select * from emp where deptno=20 with read only;
select ename,sal,deptno from emp_vur;
ENAME          SAL     DEPTNO
---------- ---------- ----------------------
JONES            3272.5          20
SCOTT            1600            20
ADAMS            1210            20
FORD             3300            20
```

在视图上定义 CHECK 约束：

```
create view empvc as select * from emp where deptno=10
with check option constraint chk_vu10;
```

建立视图时定义别名：

```
create view empview(name,salary,title) as select ename,sal,job from emp;
select name,salary,title from empview where title='CLERK';
NAME           SALARY       TITLE
---------- ---------- -------------------------
SCOTT            1600          CLERK
```

ADAMS 1210 CLERK
JAMES 950 CLERK
MILLER 1300 CLERK

修改视图
修改视图定义：
create or replace view empview(name,salary,title) as select ename,sal,job from emp;
重新编译试图(当视图基表定义改变之后)：
alter view empview compile;
删除视图：
drop view empview;

显示视图信息：USER_VIEWS,USER_UPDATE_COLUMNS
select view_name from user_views;
确定视图是否允许 DML 操作：
select column_name,insertable,updatable,deletable from user_updatable_columns
where table_name='DE_VIEW' and column_name in ('DNAME','ENAME');
COLUMN_NAME INS UPD DEL
------------------------------ --- --- --------
DNAME NO NO NO
ENAME YES YES YES

第十二章 使用其它对象

使用索引
索引分类：单列索引和复合索引，惟一索引和非惟一索引
当定义主键约束或惟一约束时，oracle 自动在相应的约束列上建立惟一约束。
建立索引的原则：
索引应该建立在 where 子句经常引用的列上
为了提高多表连接的性能，应该在连接列上建立索引
如果经常基于某些列进行排序操作，应该在这些列上建立索引
不要在小表上建立索引
索引会降低 DML 操作的速度，限制索引表的个数

建立单列索引：
create index i_ename on emp(ename);
建立复合索引（不超过 32 列）:
create index i_deptno_job on emp(deptno,job);
建立惟一索引：
create unique index i_dname on dept(dname);
重建索引(执行 delete 或 update 操作后)：
alter index i_ename rebuild;
联机重建索引：
alter index i_ename rebuild online;

删除索引：
drop index i_deptno_job;

显示索引信息：USER_INDEXES，同义词 IND
select index_name,uniqueness,status from ind where table_name='EMP';
INDEX_NAME UNIQUENES STATUS
------------------------------ --------- -------------
I_ENAME NONUNIQUE VALID
PK_EMP UNIQUE VALID

索引列 USER_IND_COLUMNS

使用序列
序列(Sequence)是一种用于生成惟一数字的数据库对象，没有重复自动递增的序列号
建立序列：
create sequence deptno_seq start with 50 increment by 10 maxvalue 99 cache 10;
sequence 指定序列名，increment by 指定序列增量(默认值 1)，start with 指定生成的第一个序列号，maxvalue 指定生成的最大序列号，cache 指定在内存中可以预分配的序列号个数(默认 20)

使用序列：首次引用序列时，只能使用伪列 nextval
insert into dept (deptno,dname,loc) values(deptno_seq.nextval,'DEVELOPMENT',default);
select deptno_seq.currval from dual;
CURRVAL

50

修改序列(序列的初始值不能修改)
alter sequence deptno_seq maxvalue 200 cache 20;
删除序列
drop sequence deptno_seq;

显示序列信息：USER_SEQUENCES，同义词 SEQ

使用同义词
同义词是方案对象的别名，可以简化对象访问，提高访问安全性
例如可以把 emp 设置为 employee 的同义词
公共同义词：所有用户都可以直接引用的同义词
私有同义词：只能由方案用户直接引用

建立公共同义词：
create public synonym pe for scott.emp;
使用同义词：
update pe set sal=3100 where ename='SCOTT';
select ename,sal,job from pe where ename='SCOTT';
ENAME SAL JOB

---------- ---------- --------------
SCOTT 3100 CLERK

建立私有同义词：
create synonym e for scott.emp;
使用私有同义词:
select ename,sal,job from scott.e where ename='SCOTT';
ENAME SAL JOB
---------- ---------- ----------
SCOTT 3100 CLERK

删除公共同义词：
drop public synonym pe;
删除私有同义词：
drop synonym e;

显示同义词信息：user_synonyms,同义词 SYN

第十三章 PL/SQL基础

块(block)是 PL/SQL 的基本程序单元，块结构为：定义部分，执行部分，异常处理部分，如下：
DECLARE
//定义部分：定义常量，变量，复杂数据类型，游标
BEGIN
//执行部分，PL/SQL 语句和 SQL 语句
EXCEPTION
//异常处理----处理运行错误
END

只包含执行部分的 PL/SQL 块：
```
SQL> BEGIN
  2  dbms_output.put_line('HEllo,Oracle!');
  3  END;
  4  /
HEllo,Oracle!
PL/SQL 过程已成功完成。
```

只包含定义部分和执行部分的 PL/SQL 块：
```
SQL> set verify off
SQL> DECLARE
  2  v_ename varchar2(5);
  3  BEGIN
  4  select ename into v_ename from emp where empno=&no;
```

```
  5  dbms_output.put_line('雇员名:'||v_ename);
  6  END;
  7  /
输入 no 的值:  7788
雇员名:SCOTT
PL/SQL 过程已成功完成。
```

包含全部三个部分的 PL/SQL 块：

```
SQL> set serveroutput on
SQL> set verify off
SQL> DECLARE
  2  v_ename varchar2(5);
  3  BEGIN
  4  select ename into v_ename from emp where empno=&no;
  5  dbms_output.put_line('雇员名:'||v_ename);
  6  EXCEPTION
  7  when no_data_found then
  8  dbms_output.put_line('请输入正确的雇员号！');
  9  END;
 10  /
输入 no 的值:  1234
请输入正确的雇员号!
```

PL/SQL 块分类：匿名块，命名块，子程序和触发器

匿名块：

```
SQL> declare
  2  v_avgsal number(6,2);
  3  begin
  4  select avg(sal) into v_avgsal from emp where deptno=&no;
  5  dbms_output.put_line('平均工资：'||v_avgsal);
  6  end;
  7  /
输入 no 的值:  30
平均工资：1628.57
PL/SQL 过程已成功完成。
```

命名块：区分多层嵌套关系，在块前用<< >>标记

```
SQL> <<outer>>
  2  declare
  3  v_deptno number(2);
  4  v_dname varchar2(10);
  5  begin
  6  <<inner>>
  7  begin select deptno into v_deptno from emp where lower(ename)=lower('&name');
  8  end;--<<inner>>
  9  select dname into v_dname from dept where deptno=v_deptno;
```

```
 10     dbms_output.put_line('部门名：'||v_dname);
 11   end;--<<outer>>
 12   /
输入 name 的值:   scott
部门名：RESEARCH
PL/SQL 过程已成功完成。
```

子程序

过程----执行特定操作，create procedure

```
SQL> CREATE PROCEDURE updatesal(name varchar2,newsal number)
  2   IS
  3   BEGIN
  4   update emp set sal=newsal where lower(ename)=lower(name);
  5   end;
  6   /
过程已创建。
SQL> exec updatesal('scott',2000)
PL/SQL 过程已成功完成。
```

函数----返回特定数据，函数头必须包含 return 子句，create function

```
SQL> create function annual_income(name varchar2)
  2   return number is annual_salary number(7,2);
  3   begin
  4   select sal*12+nvl(comm,0) into annual_salary
  5   from emp where lower(ename)=lower(name);
  6   return annual_salary;
  7   end;
  8   /

函数已创建。
SQL> select annual_income('scott') 年收入 from dual;
年收入
----------
24300
```

包：用于逻辑组合相关的过程和函数，由包规范和包体组成

示例，建立包含过程 update_sal 和函数 annual_income 的包 emp_pkg:

```
SQL> create package body emp_pkg is
  2   procedure update_sal(name varchar2,newsal number)
  3   is
  4   begin
  5   update emp set sal=newsal where lower(ename)=lower(name);
  6   end;
  7   function annual_income(name varchar2) return number
  8   is
  9   annual_salary number(7,2);
```

```
 10  begin
 11  select sal*12+nvl(comm,0) into annual_salary from emp
 12  where lower(ename)=lower(name);
 13  return annual_salary;
 14  end;
 15  end;
 16  /
程序包主体已创建。
SQL> exec emp_pkg.update_sal('scott',1500)
PL/SQL 过程已成功完成。
SQL> select emp_pkg.annual_income('scott') 年收入 from dual;
年收入
----------
18300
```

触发器：触发相关事件后隐含执行的 PL/SQL 块
示例：触发器 update_casacade

```
SQL> select ename from emp where deptno=10;
ENAME
----------
CLARK
KING
MILLER
SQL> create trigger update_cascade
  2  after update of deptno on dept for each row
  3  begin
  4  update emp set deptno=:old.deptno;
  5  end;
  6  /
触发器已创建
SQL> update dept set deptno=50 where dname='ACCOUNTING';
已更新 1 行。
SQL> select ename from emp where deptno=50;
ENAME
----------
CLARK
KING
MILLER
```

定义和使用变量
PL/SQL 的数据类型包括：

1. 标量(Scalar)类型：存放单个数值的变量
   varchar2(n)，char(n)，number(p,s)，date，timestamp，boolean
   10g 新增加的：binary_float，binary_double

示例：使用标量变量

```
SQL> declare
  2  v_ename            varchar2(5);
  3  v_sal              number(6,2);
  4  v_tax_rate       constant number(3,2):=0.03;
  5  v_tax_sal        number(6,2);
  6  begin
  7  select ename,sal into v_ename,v_sal from emp where empno=&no;
  8  v_tax_sal:=v_sal*v_tax_rate;
  9  dbms_output.put_line('雇员名：'||v_ename);
 10  dbms_output.put_line('雇员工资:'||v_sal);
 11  dbms_output.put_line('所得税：'||v_tax_sal);
 12  end;
 13  /
输入 no 的值:  7788
雇员名：SCOTT
雇员工资:1500
所得税：45
PL/SQL 过程已成功完成。
```

使用%TYPE 属性：该属性自动根据表列或其它变量的类型和长度定义新变量，可以避免运行错误，提高块的效率和健壮性

```
SQL> declare
  2  v_ename emp.ename%type;
  3  v_sal emp.sal%type;
  4  v_tax_rate       constant number(3,2):=0.03;
  5  v_tax_sal v_sal%type;
  6  begin
  7  select ename,sal into v_ename,v_sal from emp where empno=&no;
  8  v_tax_sal:=v_sal*v_tax_rate;
  9  dbms_output.put_line('雇员名：'||v_ename);
 10  dbms_output.put_line('雇员工资:'||v_sal);
 11  dbms_output.put_line('所得税：'||v_tax_sal);
 12  end;
 13  /
输入 no 的值:  7654
雇员名：MARTIN
雇员工资:1250
所得税：37.5
PL/SQL 过程已成功完成。
```

2. 复合(Composite)类型

用于存放多个数值的变量，包括 PL/SQL 记录，PL/SQL 表，嵌套表和 VARRAY 四种类型

1）PL/SQL 记录

类似于 C 语言的结构，需先定义后使用

```
SQL> declare
```

```
  2  type emprecord is record(
  3  name emp.ename%type,
  4  salary emp.sal%type,
  5  title emp.job%type);
  6  e emprecord;
  7  begin
  8  select ename,sal,job into e from emp where empno=&no;
  9  dbms_output.put_line('姓名：'||e.name);
 10  dbms_output.put_line('工资：'||e.salary);
 11  dbms_output.put_line('岗位：'||e.title);
 12  end;
 13  /
输入 no 的值:  7902
姓名：FORD
工资：3300
岗位：ANALYST
PL/SQL 过程已成功完成。
```

2）PL/SQL 表

类似于 C 语言的数组，下标可以为负值

```
SQL> declare
  2  type etable is table of emp.ename%type index by binary_integer;
  3  et etable;
  4  begin
  5  select ename into et(-1) from emp where empno=&no;
  6  dbms_output.put_line('雇员名：'||et(-1));
  7  end;
  8  /
输入 no 的值:  7499
雇员名：ALLEN
PL/SQL 过程已成功完成。
```

3）嵌套表，Nested Table，类似于 PL/SQL 表，数组类型，元素个数没限制

嵌套表可以作为列的数据类型，但 PL/SQL 不行

示例：建立对象类型 emptype，嵌套表类型 emparray 和表 department

```
SQL> create or replace type emptype as object(
  2  name varchar2(10),salary number(6,2),hiredate date);
  3  /
类型已创建。
SQL> create or replace type emparray is table of emptype;
  2  /
类型已创建。
SQL> create table department(
  2  deptno number(2),dname varchar2(10),
  3  employee emparray
  4  )nested table employee store as employee;
```

表已创建。

4）varray，变长数组，可以作为列和对象类型属性的数据类型，元素个数有限制
示例：建立对象类型 article_type，varray 类型 article_array(20 个元素)和表 author

```
SQL> create type article_type as object(title varchar2(30),pubdate date);
  2  /
类型已创建。
SQL> create type article_array is varray(20) of article_type;
  2  /
类型已创建。
SQL> create table author(id number(6),name varchar2(10),article article_array);
表已创建。
```

3. 引用(Reference)变量
类似于 C 语言的指针变量，包括 ref cursor 和 ref obj_type 两种引用类型
1）ref cursor，游标变量，实现动态游标操作

```
SQL> declare
  2  type c1 is ref cursor;
  3  cur c1;
  4  col1 varchar2(20);
  5  col2 varchar2(20);
  6  begin
  7  open cur for select &col1,&col2 from &tab where &con;
  8  fetch cur into col1,col2;
  9  dbms_output.put_line('col1:'||col1);
 10  dbms_output.put_line('col2:'||col2);
 11  close cur;
 12  end;
 13  /
输入 col1 的值:  dname
输入 col2 的值:  loc
输入 tab 的值:  dept
输入 con 的值:  deptno=10
col1:ACCOUNTING
col2:NEW YORK
PL/SQL 过程已成功完成。
SQL> /
输入 col1 的值:  ename
输入 col2 的值:  sal
输入 tab 的值:  emp
输入 con 的值:  empno=7788
原值      7: open cur for select &col1,&col2 from &tab where &con;
新值      7: open cur for select ename,sal from emp where empno=7788;
col1:SCOTT
col2:1500
PL/SQL 过程已成功完成。
```

2）ref obj_type
用于定义特定对象类型的指针类型，ref 实际是指向对象实例的指针，可以共享对象，节省对象空间占用
示例：建立对象类型 houseType，对象表 houses 和 population（家庭成员具有相同的家庭地址）

```
SQL> create or replace type houseType as object(
  2  street varchar2(50),city varchar2(20),
  3  state varchar2(20),zipcode varchar2(6),
  4  owner varchar2(10)
  5  );
  6  /
```

类型已创建。

```
SQL> create table houses of houseType;
```

表已创建。

```
SQL> insert into houses values('中山路 12 号','南京','江苏','210090','马明');
```

已创建 1 行。

```
SQL> create table population(
  2  id number(6) primary key,
  3  name varchar2(10),addr ref houseType);
```

表已创建。

```
SQL> insert into population select 1,'马明',ref(p) from houses p where p.owner='马明';
```

已创建 1 行。

```
SQL> insert into population select 2,'马五',ref(p) from houses p where p.owner='马明';
```

已创建 1 行。

```
SQL> insert into population select 3,'王铭',ref(p) from houses p where p.owner='马明';
```

已创建 1 行。

4. LOB(Large Object)变量
LOB 变量用于存储大对象的数据，包括 CLOB,BLOB,NCLOB,BFILE 四种类型，其中：
CLOB 和 NCLOB 用于存储大字符数据
BLOB 用于存储大二进制数据
BFILE 存储指向 OS 文件的指针

使用子类型定义变量
1）PL/SQL 子类型

```
SQL> declare
  2  subtype myType is varchar2(20);
  3  vname myType(10);
  4  begin
  5  select ename into vname from emp where empno=&eno;
  6  dbms_output.put_line('姓名：'||vname);
  7  end;
  8  /
```

输入 eno 的值: 7499

姓名：ALLEN
PL/SQL 过程已成功完成。

2）非 PL/SQL 变量
SQL*Plus 变量：
SQL> var name varchar2(10)
SQL> begin
  2 select ename into :name from emp where empno=&eno;
  3 end;
  4 /
输入 eno 的值: 7499
PL/SQL 过程已成功完成。
SQL> print name
NAME

ALLEN

PL/SQL 词汇单元
1）分隔符 delimiter
算数运算： +，-，*，/
赋值操作： ：=
关联操作： =>
连接操作： ||
幂操作符： **
注释： 单行-- 多行： /* */
2）标识符 identifier
每行只能定义一个变量或常量，以；结束
必须以英文字符开始，最大长度为 30
变量和常量名只能使用英文，数字，下划线_，$和#
以其它字符作变量名必须用双引号括住
不能使用 oracle 关键字，如 select 等
3）文本 literal(字面值)
文本是指实际的数值，包括数字文本，字符文本，字符串文本，布尔文本，日期时间文本
4）注释 comment

第十四章 嵌入SQL语句

检索单行数据

使用标量变量接受数据：
SQL> declare
  2 vname emp.ename%type;
  3 vsal emp.sal%type;
  4 begin

```
  5  select ename,sal into vname,vsal from emp where empno=&no;
  6  dbms_output.put_line('姓名：'||vname);
  7  dbms_output.put_line('工资：'||vsal);
  8  end;
  9  /
输入 no 的值:  7902
姓名：FORD
工资：3300
PL/SQL 过程已成功完成。
```

使用记录变量接收数据：

```
SQL> declare
  2  type empRecord is record(
  3  name emp.ename%type,title emp.job%type);
  4  er empRecord;
  5  begin
  6  select ename,job into er from emp where empno=&no;
  7  dbms_output.put_line('姓名：'||er.name);
  8  dbms_output.put_line('岗位：'||er.title);
  9  end;
 10  /
输入 no 的值:  7782
姓名：CLARK
岗位：MANAGER
```

嵌入 SELECT 语句的注意事项：

嵌入 select into 语句时，必须返回一条数据，否则会发生错误

1）NO_DATA_FOUND 异常：select into 没有返回任何数据时

2）TOO_MANY_ROWS 异常：select into 返回多条数据时

使用 where 子句时，变量名不能与列名相同

插入数据

使用 values 子句插入数据：

```
SQL> declare
  2  v_deptno dept.deptno%type;
  3  v_dname dept.dname%type;
  4  begin
  5  v_deptno:=&no;
  6  v_dname:='&name';
  7  insert into dept (deptno,dname)
  8  values(v_deptno,v_dname);
  9  end;
 10  /
```

在 PL/SQL 块中使用子查询插入数据：

```
SQL> declare
```

```
  2  vdno emp.deptno%type:=&no;
  3  begin
  4  insert into employee
  5  select * from emp where deptno=vdno;
  6  end;
  7  /
```

更新数据

使用表达式更新列值：

```
SQL> declare
  2  vdno dept.deptno%type:=&no;
  3  vloc dept.loc%type:='&loc';
  4  begin
  5  update dept set loc=vloc where deptno=vdno;
  6  end;
  7  /
输入 no 的值:  50
输入 loc 的值:  BEIJING
PL/SQL 过程已成功完成。
```

使用子查询更新列值：

```
SQL> declare
  2  vname emp.ename%type:='&name';
  3  begin
  4  update emp set (sal,comm)=
  5  (select sal,comm from emp where ename=vname)
  6  where job=(select job from emp where ename=vname);
  7  end;
  8  /
输入 name 的值:  SCOTT
PL/SQL 过程已成功完成。
```

删除数据

使用变量删除数据：

```
SQL> declare
  2  vdno dept.deptno%type:=&no;
  3  begin
  4  delete from dept where deptno=vdno;
  5  end;
  6  /
输入 no 的值:  50
PL/SQL 过程已成功完成。
```

使用子查询删除数据：

```
SQL> declare
  2  vname emp.ename%type:='&name';
```

```
  3  begin
  4  delete from emp
  5  where deptno=(select deptno from emp where ename=vname);
  6  end;
  7  /
输入 name 的值:  SCOTT
PL/SQL 过程已成功完成。
```

SQL 游标：
SQL%ISOPEN----确定 SQL 游标是否打开
SQL%FOUND 和 SQL%NOTFOUND----确定 SQL 语句执行是否成功
SQL%ROWCOUNT----返回 SQL 语句所作用的总行数
示例：
使用 SQL%FOUND

```
SQL> declare
  2  vdno emp.deptno%type:=&no;
  3  begin
  4  update emp set sal=sal*1.1 where deptno=vdno;
  5  if sql%found then
  6  dbms_output.put_line('删除了'||sql%rowcount||'行');
  7  else
  8  dbms_output.put_line('该部门不存在该雇员');
  9  end if;
 10  end;
 11  /
输入 no 的值:  30
删除了 7 行
PL/SQL 过程已成功完成。
```

使用事务控制语句
在 PL/SQL 块中嵌入 commit 和 rollback 语句
在 PL/SQL 块中嵌入 rollback 和 savepoint 语句

第十五章 控制结构

条件分支语句
IF-THEN，IF-THEN-ELSE，IF-THEN-ELSIF

简单条件分支：

```
SQL> set serveroutput on
SQL> set verify off
SQL> declare
  2  vsal number(6,2);
  3  begin
```

```
  4  select sal into vsal from emp where lower(ename)=lower('&&name');
  5  IF vsal<2000 THEN
  6  update emp set sal=vsal+200 where lower(ename)=('&name');
  7  END IF;
  8  end;
  9  /
输入 name 的值:  miller
PL/SQL 过程已成功完成。
```

二重条件分支：

```
SQL> declare
  2  vcomm number(6,2);
  3  begin
  4  select comm into vcomm from emp where empno=&&no;
  5  IF vcomm<>0 THEN
  6     update emp set comm=vcomm+200 where empno=&no;
  7  ELSE
  8     update emp set comm=vcomm+200 where empno=&no;
  9  END IF;
 10  end;
 11  /
输入 no 的值:  7934
PL/SQL 过程已成功完成。
```

多重条件分支：

```
SQL> declare
  2  vjob varchar2(10);
  3  vsal number(6,2);
  4  begin
  5  select job,sal into vjob,vsal from emp where empno=&&no;
  6  IF vjob='PRESIDENT' THEN
  7     update emp set sal=vsal+1000 where empno=&no;
  8  ELSIF vjob='MANAGER' THEN
  9       update emp set sal=vsal+500 where empno=&no;
 10   ELSE
 11       update emp set sal=vsal+200 where empno=&no;
 12   END IF;
 13  end;
 14  /
输入 no 的值:  7788
PL/SQL 过程已成功完成。
```

CASE 语句：多分支

单一选择符等值比较：

```
declare
vno emp.deptno%type;
```

```
begin
vno:=&no;
CASE vno
   WHEN 10 THEN
      update emp set comm=100 where deptno=vno;
   WHEN 20 THEN
      update emp set comm=80 where deptno=vno;
   WHEN 30 THEN
      update emp set comm=50 where deptno=vno;
   ELSE
    dbms_output.put_line('不存在该部门!');
END CASE;
end;
/
```

CASE 中多种条件比较：

```
SQL> declare
  2  vsal emp.sal%type;
  3  vname emp.ename%type;
  4  begin
  5  select ename,sal into vname,vsal from emp where empno=&no;
  6  CASE
  7     WHEN vsal<1000 THEN
  8        update emp set comm=100 where ename=vname;
  9     WHEN vsal<2000 THEN
 10        update emp set comm=80 where ename=vname;
 11     WHEN vsal<6000 THEN
 12        update emp set comm=50 where ename=vname;
 13  END CASE;
 14  end;
 15  /
输入 no 的值:  7788
PL/SQL 过程已成功完成。
```

循环语句

基本循环：LOOP……END LOOP，语句块至少会被执行一次

```
SQL> declare
  2  i int:=1;
  3  begin
  4     LOOP
  5        insert into temp values(i);
  6        EXIT WHEN i=10;
  7        i:=i+1;
  8     END LOOP;
  9  end;
 10  /
```

PL/SQL 过程已成功完成。

WHILE 循环：

```
SQL> declare
  2  i int:=1;
  3  begin
  4     WHILE i<=10 LOOP
  5        insert into temp01 values(i);
  6        i:=i+1;
  7     END LOOP;
  8  end;
  9  /
```

PL/SQL 过程已成功完成。

FOR 循环：

当使用 for 循环时，oracle 会隐含定义循环控制变量

```
begin
FOR i IN REVERSE 1..10 LOOP
  insert into temp values(i);
END LOOP;
end;
```

嵌套循环和标号

```
SQL> declare
  2  result int;
  3  begin
  4  <<outer>>
  5  for i in 1..100 loop
  6  <<inner>>
  7  for j in 1..100 loop
  8  result:=i*j;
  9  exit outer when result=1000;
 10  exit when result=500;
 11  end loop inner;
 12  dbms_output.put_line(result);
 13  end loop outer;
 14  dbms_output.put_line(result);
 15  end;
 16  /
```

GOTO 语句

NULL 语句：不执行操作，将控制权交给下一语句

第十六章 复合数据类型

PL/SQL 记录
主要用于处理单行多列数据
定义 PL/SQL 记录：

```
DECLARE
   TYPE emp_record_type IS record(
      name         emp.ename%TYPE,
      salary       emp.sal%TYPE,
      dno          emp.deptno%TYPE
);
emp_record       emp_record_type;
…
```

使用%rowtype 定义记录变量：

```
dept_record     dept%rowtype;
emp_record      emp%rowtype;
```

使用 PL/SQL 记录
在 select into 中使用 PL/SQL 记录：

```
SQL> set serveroutput on
SQL> set verify off
SQL> declare
  2   type empRecordType is record(
  3   name emp.ename%type,
  4   salary emp.sal%type,
  5   title emp.job%type);
  6   empRec empRecordType;
  7   begin
  8   select ename,sal,job into empRec from emp where empno=&no;
  9   dbms_output.put_line('姓名：'||empRec.name);
 10   dbms_output.put_line('岗位：'||empRec.title);
 11   dbms_output.put_line('工资：'||empRec.salary);
 12   end;
 13   /
输入 no 的值:  7788
姓名：SCOTT
岗位：CLERK
工资：1700
PL/SQL 过程已成功完成。
```

在 insert 语句中使用记录：

```
SQL> declare
  2   drec dept%rowtype;
  3   begin
  4   drec.deptno:=50;
  5   drec.dname:='ADMINISTRATOR';
```

```
  6  drec.loc:='BEIJING';
  7  insert into dept values drec;
  8  end;
  9  /
```

在 values 子句中使用记录成员：

```
SQL> declare
  2  drec dept%rowtype;
  3  begin
  4  drec.deptno:=60;
  5  drec.dname:='SALES';
  6  insert into dept (deptno,dname) values
  7  (drec.deptno,drec.dname);
  8  end;
  9  /
```

在 update 中使用记录：

在 set 子句中使用记录变量更新数据：

```
SQL> declare
  2  dept_record dept%rowtype;
  3  begin
  4  dept_record.deptno:=30;
  5  dept_record.dname:='SALES';
  6  dept_record.loc:='NANJING';
  7  update dept set row=dept_record where deptno=30;
  8  end;
  9  /
PL/SQL  过程已成功完成。
```

在 set 子句中使用记录成员：

```
SQL> declare
  2  dept_record dept%rowtype;
  3  begin
  4  dept_record.deptno:=30;
  5  dept_record.dname:='SALES';
  6  dept_record.loc:='NANJING';
  7  update dept set row=dept_record where deptno=30;
  8  end;
  9  /
PL/SQL  过程已成功完成。
```

在 delete 子句中使用记录：

```
declare
dr dept%rowtype;
begin
dr.deptno:=50;
```

```
delete from dept where deptno=dr.deptno;
PL/SQL 过程已成功完成。
```

PL/SQL 集合
PL/SQL 表：元素个数无限制，下标可以为负
PL/SQL 表不能作为列的数据类型

使用 binary_integer 或 pls_integer 定义 PL/SQL 表的下标：

```
SQL> set serveroutput on
SQL> declare
  2  type ename_table_type is table of emp.ename%type index by binary_integer;
  3  etab ename_table_type;
  4  begin
  5  select ename into etab(-1) from emp where empno=&no;
  6  dbms_output.put_line('姓名:'||etab(-1));
  7  end;
  8  /
输入 no 的值:  7902
姓名:FORD
PL/SQL 过程已成功完成。
```

使用 varchar2 定义 PL/SQL 表的下标：

```
SQL> set verify off
SQL> declare
  2  type areaTableType is table of number index by varchar(10);
  3  atab areaTableType;
  4  begin
  5  atab('Japan'):=1;
  6  atab('China'):=2;
  7  atab('American'):=3;
  8  atab('England'):=4;
  9  atab('Portugal'):=5;
 10  dbms_output.put_line('第一个元素：'||atab.first);
 11  dbms_output.put_line('最后一个元素：'||atab.last);
 12  end;
 13  /
第一个元素：American
最后一个元素：Portugal
PL/SQL 过程已成功完成。
```

嵌套表：元素个数无限制，下标从 1 开始，可以作为列的数据类型，可以是稀疏的
在 PL/SQL 块中使用嵌套表：

```
SQL> declare
  2  type ename_table_type is table of emp.ename%type;
  3  et ename_table_type;
  4  begin
```

```
  5  et:=ename_table_type('MARY','MARY','MARY');
  6  select ename into et(2) from emp where empno=&no;
  7  dbms_output.put_line('雇员名：'||et(2));
  8  end;
  9  /
输入 no 的值:  7782
雇员名：CLARK
PL/SQL 过程已成功完成。
```

在表列中使用嵌套表：

```
SQL> create or replace type phone_type is table of varchar2(20);
  2  /
类型已创建。
SQL> create table person(
  2  id number(4),name varchar2(10),sal number(6,2),phone phone_type)
  3  nested table phone store as phone_table;
表已创建。
```

为嵌套表插入数据：

```
SQL> begin
  2  insert into person values(1,'SCOTT',800,phone_type('0471-3456788','13804171235'));
  3  end;
  4  /
PL/SQL 过程已成功完成。
```

检索嵌套表列的数据：

```
SQL> declare
  2  pt phone_type;
  3  begin
  4  select phone into pt from person where name='SCOTT';
  5  for i in 1..pt.count loop
  6  dbms_output.put_line('号码：'||i||':'||pt(i));
  7  end loop;
  8  end;
  9  /
号码：1:0471-3456788
号码：2:13804171235
PL/SQL 过程已成功完成。
```

更新嵌套表列的数据：

```
SQL>   declare
  2    pt phone_type:=phone_type('025-58843287',
  3    '15805175201','0714-6575787','13222926585');
  4  begin
  5  update person set phone=pt where id=1;
  6  end;
```

```
  7  /
```
PL/SQL 过程已成功完成。

变长数组 VARRAY
可以作为表列的数据类型，下标从 1 开始，元素个数有限制
在 PL/SQL 块中使用 varray：
```
SQL> declare
  2  type job_array_type is varray(20) of emp.job%type;
  3  jarr job_array_type:=job_array_type('CLERK','CLERK');
  4  begin
  5  select job into jarr(1) from emp where lower(ename)=lower('&name');
  6  dbms_output.put_line('岗位：'||jarr(1));
  7  end;
  8  /
```
输入 name 的值: scott
岗位：CLERK

在表列中使用 varray：
```
SQL> create type phone_array is varray(20) of varchar(20);
  2  /
```
类型已创建。
```
SQL> create table worker(
  2  id number(4),name varchar2(10),
  3  sal number(6,2),phone phone_array);
```
表已创建。

PL/SQL 记录表，类似于多维数组，结合了 PL/SQL 记录和 PL/SQL 集合的有点，可以有效的处理多行多列的数据
```
SQL> declare
  2  type emp_table_type is table of emp%rowtype index by binary_integer;
  3  et emp_table_type;
  4  begin
  5  select * into et(1) from emp where empno=&no;
  6  dbms_output.put_line('姓名：'||et(1).ename);
  7  dbms_output.put_line('工资：'||et(1).sal);
  8  dbms_output.put_line('岗位：'||et(1).job);
  9  dbms_output.put_line('上岗日期：'||et(1).hiredate);
 10  end;
 11  /
```
输入 no 的值: 7782
姓名：CLARK
工资：1500
岗位：MANAGER
上岗日期：09-6 月 -81
PL/SQL 过程已成功完成。

PL/SQL 集合方法
EXISTS 方法：确定特定集合元素是否存在值

```
SQL> declare
  2  type id_table_type is table of number(6);
  3  idt id_table_type:=id_table_type(1,2,5);
  4  no int;
  5  begin
  6  no:=&i;
  7  if idt.exists(no) then
  8    dbms_output.put_line('元素值：'||idt(no));
  9  else
 10    dbms_output.put_line('元素未初始化');
 11  end if;
 12  end;
 13  /
输入 i 的值:  4
元素未初始化
PL/SQL 过程已成功完成。
```

COUNT 方法：返回集合变量的元素总个数

```
SQL> declare
  2  type id_array_type is varray(20) of number(6);
  3  ids id_array_type:=id_array_type(1,null,2,5);
  4  begin
  5  dbms_output.put_line('IDS 的元素总数为：'||ids.count);
  6  end;
  7  /
IDS 的元素总数为：4
PL/SQL 过程已成功完成。
```

LIMIT 方法：返回 varray 变量所允许的最大元素个数

```
SQL> declare
  2  type id_array_type is varray(20) of number(6);
  3  ids id_array_type:=id_array_type(1,null,2,5);
  4  begin
  5  dbms_output.put_line('IDS 的元素总数为：'||ids.count);
  6    dbms_output.put_line('IDS 的最大元素个数为：'||ids.limit);
  7  end;
  8  /
IDS 的元素总数为：4
IDS 的最大元素个数为：20
PL/SQL 过程已成功完成。
```

FIRST 和 LAST：第一个和最后一个元素
示例见前面的 PL/SQL 表

PRIOR 和 NEXT：前一个元素和下一个元素的下标

```
SQL> declare
  2  type ename_table_type is table of emp.ename%type index by binary_integer;
  3  et ename_table_type;
  4  begin
  5  et(-5):='SCOTT';
  6  et(1):='SMITH';
  7  et(5):='MARY';
  8  et(10):='BLACK';
  9  dbms_output.put_line('元素 1 的前一个元素值：'||et(et.prior(1)));
 10  dbms_output.put_line('元素 1 的后一个元素值：'||et(et.next(1)));
 11  end;
 12  /
元素 1 的前一个元素值：SCOTT
元素 1 的后一个元素值：MARY
PL/SQL 过程已成功完成。
```

EXTEND 方法：为集合变量增加元素

三种调用格式：

extend：添加一个 null 元素

extend(n)：添加 n 个 null 元素

extend(n,i)：添加 n 个元素，值与第 i 个元素相同

```
SQL> declare
  2  type id_table_type is table of number(6);
  3  idtab id_table_type:=id_table_type(1);
  4  begin
  5  idtab.extend(&no);
  6  dbms_output.put_line('第一个元素：'||idtab(idtab.first));
  7  dbms_output.put_line('最后一个元素：'||nvl(to_char(idtab(idtab.last)),'null'));
  8  dbms_output.put_line('元素总个数：'||idtab.count);
  9  end;
 10  /
输入 no 的值:  999
第一个元素：1
最后一个元素：null
元素总个数：1000
PL/SQL 过程已成功完成。
```

TRIM 方法：从集合尾部删除元素，删除 n 个：trim(n)

```
SQL> declare
  2  type id_array_type is varray(30) of char;
  3  id id_array_type:=id_array_type
  4  ('A','B','C','D','E','F','G','H','I','J','K','L');
  5  BEGIN
  6  id.trim(&no);
```

```
  7  dbms_output.put_line('第一个元素：'||id(id.first));
  8  dbms_output.put_line('最后一个元素：'||id(id.last));
  9  dbms_output.put_line('元素总个数：'||id.count);
 10  end;
 11  /
输入 no 的值:  5
第一个元素：A
最后一个元素：G
元素总个数：7
PL/SQL 过程已成功完成。
```

DELETE 方法：删除集合元素
delete：删除所有元素
delete(n)：删除第 n 个元素
delete(m,n)：删除 m~n 之间的所有元素

```
SQL> declare
  2  type id_table_type is table of number(6) index by binary_integer;
  3  id id_table_type;
  4  begin
  5  for i in 1..&no loop
  6  id(i):=i;
  7  end loop;
  8  for j in 1..id.count loop
  9  if mod(j,2)=0 then
 10  id.delete(j);
 11  end if;
 12  end loop;
 13  dbms_output.put_line('元素总个数：'||id.count);
 14  dbms_output.put_line('第一个元素：'||id(id.first));
 15  dbms_output.put_line('最后一个元素：'||id(id.last));
 16  end;
 17  /
输入 no 的值:  10000
元素总个数：5000
第一个元素：1
最后一个元素：9999
PL/SQL 过程已成功完成。
```

批量绑定(bulk bind)
批量绑定使用 bulk collect 子句和 forall 子句：
bulk collect 只适用于 select，fetch 和 dml 返回子句
forall 只是用于 dml 语句

示例：
create table demo (id number(6),name varchar2(10));
传统循环输入：

```
SQL> declare
  2  type id_table_type is table of number(6) index by binary_integer;
  3  type name_table_type is table of varchar2(10) index by binary_integer;
  4  ids id_table_type;
  5  names name_table_type;
  6  start_time number(10);
  7  end_time number(10);
  8  begin
  9  for i in 1..100000 loop
 10     ids(i):=i;
 11     names(i):='Name'||to_char(i);
 12  end loop;
 13  start_time:=dbms_utility.get_time;
 14  for i in 1..ids.count loop
 15     insert into demo values(ids(i),names(i));
 16  end loop;
 17  end_time:=dbms_utility.get_time;
 18  dbms_output.put_line('总计时间：'||to_char((end_time-start_time)/100)||'秒');
 19  end;
 20  /
总计时间：5.67 秒
PL/SQL  过程已成功完成。
```

使用批量绑定插入数据：

```
SQL> declare
  2  type id_table_type is table of number(6) index by binary_integer;
  3  type name_table_type is table of varchar2(10) index by binary_integer;
  4  ids id_table_type;
  5  names name_table_type;
  6  start_time number(10);
  7  end_time number(10);
  8  begin
  9  for i in 1..100000 loop
 10     ids(i):=i;
 11     names(i):='Name'||to_char(i);
 12  end loop;
 13  start_time:=dbms_utility.get_time;
 14
 15  FORALL i in 1..ids.count
 16  insert into demo values(ids(i),names(i));
 17  end_time:=dbms_utility.get_time;
 18  dbms_output.put_line('总计时间：'||to_char((end_time-start_time)/100)||'秒');
 19  end;
 20  /
总计时间：.21 秒
PL/SQL  过程已成功完成。
```

bulk collect 子句，用于将批量数据存放到 PL/SQL 集合
在 select into 中使用 bulk collect 子句：

```
SQL> declare
  2  type atp is table of emp%rowtype index by binary_integer;
  3  e atp;
  4  begin
  5  select * BULK COLLECT into e from emp where deptno=&no;
  6  for i in 1..e.count loop
  7  dbms_output.put_line('姓名:'||e(i).ename||'，岗位：'||e(i).job||'，工资:'|e(i).sal);
  8  end loop;
  9  end;
 10  /
输入 no 的值:  30
（部分结果）
姓名:ALLEN，岗位：SALESMAN，工资:1600
姓名:WARD，岗位：SALESMAN，工资:1250
姓名:JONES，岗位：MANAGER，工资:3272.5
PL/SQL 过程已成功完成。
```

在 DML 返回子句中使用 bulk collect 子句：

```
SQL> declare
  2  type etp is table of emp.ename%type;
  3  type stp is table of emp.sal%type;
  4  e etp;
  5  s stp;
  6  begin
  7  update emp set sal=sal*1.1 where deptno=&no
  8  returning ename,sal
  9  BULK COLLECT into e,s;
 10  for i in 1..e.count loop
 11  dbms_output.put_line('姓名：'||e(i)||'，新工资'||s(i));
 12  end loop;
 13  end;
 14  /
输入 no 的值:  30
姓名：ALLEN，新工资 1760
姓名：WARD，新工资 1375
姓名：JONES，新工资 3599.75
PL/SQL 过程已成功完成。
```

使用 FORALL 语句
批量插入数据：

```
SQL> declare
  2  type idtp is table of demo.id%type index by binary_integer;
  3  type nametp is table of demo.name%type index by binary_integer;
```

```
  4  id idtp;
  5  nm nametp;
  6  begin
  7  for i in 1..200000 loop
  8  id(i):=i;
  9  nm(i):='Name'||to_char(i);
 10  end loop;
 11  FORALL i in 1..id.count
 12  insert into demo values(id(i),nm(i));
 13  end;
 14  /
PL/SQL 过程已成功完成。
```

更新批量数据：

```
SQL> declare
  2  type idtp is table of demo.id%type index by binary_integer;
  3  type nametp is table of demo.name%type index by binary_integer;
  4  id idtp;
  5  nm nametp;
  6  begin
  7  for i in 1..20000 loop
  8  id(i):=i;
  9  nm(i):='N'||to_char(i);
 10  end loop;
 11  FORALL i in 1..id.count
 12  update demo set name=nm(i) where id=id(i);
 13  end;
 14  /
PL/SQL 过程已成功完成。
```

删除批量数据：

```
SQL> declare
  2  type idtp is table of number(6) index by binary_integer;
  3  id idtp;
  4  begin
  5  for i in 1..300 loop
  6  id(i):=i;
  7  end loop;
  8  FORALL i in 100..200
  9  delete from demo where id=id(i);
 10  end;
 11  /
PL/SQL 过程已成功完成。
```

SQL%BULK_ROWCOUNT 属性：返回特定元素所作用的行数

```
SQL> set serveroutput on
```

```
SQL> set verify off
SQL> declare
  2  type dtp is table of number(3);
  3  dt dtp:=dtp(10,20);
  4  begin
  5  forall i in 1..dt.count
  6  update emp set sal=sal*1.1 where deptno=dt(i);
  7  dbms_output.put_line('部门 10 被更新了'||SQL%BULK_ROWCOUNT(1)||'行');
  8  dbms_output.put_line('部门 20 被更新了'||SQL%BULK_ROWCOUNT(2)||'行');
  9  end;
 10  /
部门 10 被更新了 0 行
部门 20 被更新了 0 行
PL/SQL 过程已成功完成。
```

在 forall 语句中使用 indices of 子句(跳过 PL/SQL 集合变量中的 NULL 元素，oracle 10g 新增加)

```
SQL> declare
  2  type itp is table of number(6);
  3  id itp;
  4  begin
  5  id:=itp(100,null,300,null,500);
  6  forall i in indices of id
  7  delete from demo where id=id(i);
  8  for i in 1..id.count loop
  9  if SQL%BULK_ROWCOUNT(i)<>0 then
 10  dbms_output.put_line('ID= '||id(i)||'的行被删除');
 11  end if;
 12  end loop;
 13  end;
 14  /
```

在 forall 语句上使用 values of（用于从其它集合变量中取得集合下标值，10g 新增加）子句

PL/SQL 集合高级特性(Oracle 10g 新增加)

赋值操作： ：=

SET 操作符：取消嵌套表变量的重复值

MULTISET UNION 操作符：合并两个嵌套变量的结果

MULTISET UNION DISTINCT：合并嵌套表结果，取消重复

MULTISET INTERSECT 操作符：取两个嵌套表变量的交集

MULTISET EXCEPT 操作符：取两个嵌套表变量的差集

第十七章 使用游标

显示游标
显示游标包括四种属性：
%ISOPEN ： 检测游标是否已经打开
%FOUND ： 检测游标结果集是否存在数据，存在则返回 TRUE
%NOTFOUND ： 检测游标结果集是否不存在数据，不存在则返回 TRUE
%ROWCOUNT ： 返回已提取的实际行数

使用显示游标
定义游标：CURSOR cursor_name IS select_statement；
打开游标：OPEN cursor_name；
提取数据：FETCH cursor_name INTO variable1[,variable2,…];
FETCH INTO 每次只能提取一行数据，批量数据需使用循环

使用游标变量接受数据：

```
SQL> declare
  2  cursor emp_cursor is
  3  select ename,job,sal from emp where deptno=&dno;
  4  vname emp.ename%type;
  5  vsal emp.sal%type;
  6  vjob emp.job%type;
  7  begin
  8  open emp_cursor;
  9  loop
 10  fetch emp_cursor into vname,vjob,vsal;
 11  exit when emp_cursor%notfound;
 12  dbms_output.put_line('姓名：'||vname||'，岗位：'||vjob||'，工资：'||vsal);
 13  end loop;
 14  close emp_cursor;
 15  end;
 16  /
输入 dno 的值:  30
姓名：ALLEN，岗位：SALESMAN，工资：1600
姓名：WARD，岗位：SALESMAN，工资：1250
姓名：JONES，岗位：MANAGER，工资：3272.5
PL/SQL 过程已成功完成。
```

使用 PL/SQL 记录变量接受游标数据：简化单行数据处理

```
SQL> declare
  2  cursor ecur is select ename,sal from emp order by sal desc;
  3  erec ecur%rowtype;
  4  begin
  5  open ecur;
  6  loop
  7  fetch ecur into erec;
```

```
  8  exit when ecur%notfound or ecur%rowcount>&n;
  9  dbms_output.put_line('姓名：'||erec.ename||'，工资：'||erec.sal);
 10  end loop;
 11  close ecur;
 12  end;
 13  /
输入 n 的值:  5
姓名：KING，工资：5000
姓名：FORD，工资：3300
姓名：JONES，工资：3272.5
姓名：BLAKE，工资：2850
姓名：MARY，工资：2000
PL/SQL 过程已成功完成。
```

使用 PL/SQL 集合变量接受游标数据：简化多行多列数据处理

```
SQL> declare
  2  cursor ec is select ename,sal from emp where lower(job)=lower('&job');
  3  type etype is table of ec%rowtype index by binary_integer;
  4  et etype;
  5  i int;
  6  begin
  7  open ec;
  8  loop
  9  i:=ec%rowcount+1;
 10  fetch ec into et(i);
 11  exit when ec%notfound;
 12  dbms_output.put_line('姓名：'||et(i).ename||'，工资：'||et(i).sal);
 13  end loop;
 14  close ec;
 15  end;
 16  /
输入 job 的值:  manager
姓名：JONES，工资：3272.5
姓名：BLAKE，工资：2850
姓名：CLARK，工资：1500
PL/SQL 过程已成功完成。
```

游标 FOR 循环

使用游标 for 循环时，oracle 会隐含的打开游标，提取数据并关闭游标

在游标 for 循环中引用已定义游标：

```
SQL> declare
  2  cursor ec is select ename,hiredate from emp order by hiredate desc;
  3  begin
  4  for erec in ec loop
  5   dbms_output.put_line('姓名：'||erec.ename||'，工作日期：'||erec.hiredate);
  6  exit when ec%rowcount=&n;
```

```
  7  end loop;
  8  end;
  9  /
输入 n 的值:  3
姓名：MARY，工作日期：
姓名：ADAMS，工作日期：23-5 月 -87
姓名：SCOTT，工作日期：01-1 月 -84
PL/SQL 过程已成功完成。
```

在游标 for 循环中直接引用子查询：

```
SQL> begin
  2  for erec in (select ename,hiredate,rownum from emp order by hiredate) loop
  3  dbms_output.put_line('姓名：'||erec.ename||'，工作日期：'||erec.hiredate);
  4  exit when erec.rownum=&n;end loop;
  5  end;
  6  /
输入 n 的值:  2
姓名：ALLEN，工作日期：20-2 月 -81
姓名：WARD，工作日期：22-2 月 -81
PL/SQL 过程已成功完成。
```

参数游标：参数只能指定数据类型，不能指定长度，而且必须在 where 子句中引用参数

```
SQL> declare
  2  cursor ec(dno number) is select ename,job from emp where deptno=dno;
  3  begin
  4  for erec in ec(&dno) loop
  5  dbms_output.put_line('姓名：'||erec.ename||'，岗位：'||erec.job);
  6  end loop;
  7  end;
  8  /
输入 dno 的值:  30
姓名：ALLEN，岗位：SALESMAN
姓名：WARD，岗位：SALESMAN
姓名：JONES，岗位：MANAGER
PL/SQL 过程已成功完成。
```

更新游标行

```
declare
cursor emp_cursor is select ename,sal,deptno from emp for update;
dno int:=&no;
begin
for emp_record in emp_cursor loop
if emp_record.deptno=dno then
dbms_output.put_line('姓名：'||emp_record.ename||'，原工资：'||emp_record.sal);
update emp set sal=sal*1.1 where current of emp_cursor;
end if;
```

```
end loop;
end;
/
```

删除游标行

```
declare
cursor emp_cursor is select ename from emp for update;
name varchar2(10):=lower('&name');
begin
for emp_record in emp_cursor loop
if lower(emp_record.ename)=name then
delete from emp where current of emp_cursor;
else
dbms_output.put_line('姓名：'||emp_record.ename);
end if;
end loop;
end;
/
```

使用 for 子句在特定表上加共享锁(涉及多张表时的同步问题)

```
SQL> declare
  2  cursor emp_cursor is
select a.dname,b.ename from dept a JOIN emp b ON a.deprno=b.deptno;
  3  name varchar2(10):=lower('&name');
  4  begin
  5  for emp_record in emp_cursor loop
  6  if lower(emp_record.dname)=name then
  7  dbms_output.put_line('姓名：'||emp_record.ename);
  8  delete from emp where current of emp_cursor;
  9  end if;
 10  end loop;
 11  end;
 12  /
输入 name 的值:  sales
PL/SQL 过程已完成。
```

游标变量

游标变量是基于 REF CURSOR 类型所定义的变量，它实际上是指向内存地址的指针。使用显式游标只能定义静态游标，而通过使用游标变量可以在打开游标时指定其对应的 select 语句，从而实现动态游标。

使用无返回类型的游标变量

```
SQL> set serveroutput on
SQL> set verify off
SQL> declare
  2  type ref_cursor_type is ref cursor;
```

```
  3  rc ref_cursor_type;
  4  v1 number(6);
  5  v2 varchar2(10);
  6  begin
  7  open rc for
  8  select &col1 col1,&col2 col2 from &table where &cond;
  9  loop
 10  fetch rc into v1,v2;
 11  exit when rc%notfound;
 12  dbms_output.put_line('col1= '||v1||',col2= '||v2);
 13  end loop;
 14  close rc;
 15  end;
 16  /
输入 col1 的值:  empno
输入 col2 的值:  ename
输入 table 的值:  emp
输入 cond 的值:  deptno=10
col1= 7782,col2= CLARK
col1= 7839,col2= KING
col1= 7934,col2= MILLER
PL/SQL 过程已成功完成。
```

使用有返回类型的游标变量

```
SQL> declare
  2  type emp_cursor_type is ref cursor return emp%rowtype;
  3  ec emp_cursor_type;
  4  er emp%rowtype;
  5  begin
  6  open ec for select * from emp where deptno=&dno;
  7  loop
  8  fetch ec into er;
  9  exit when ec%notfound;
 10  dbms_output.put_line('姓名：'||er.ename||'，工资：'||er.sal);
 11  end loop;
 12  close ec;
 13  end;
 14  /
输入 dno 的值:  20
姓名：SMITH，工资：800
姓名：JONES，工资：2975
姓名：SCOTT，工资：3000
姓名：ADAMS，工资：1100
姓名：FORD，工资：3000
PL/SQL 过程已成功完成。
```

使用批量提取

使用 fetch…bulk collect 提取所有数据;

```
SQL> declare
  2  cursor ec is
  3  select * from emp where lower(job)=lower('&job');
  4  type etype is table of emp%rowtype;
  5  et etype;
  6  begin
  7  open ec;
  8  fetch ec bulk collect into et;
  9  close ec;
 10  for i in 1..et.count loop
 11  dbms_output.put_line('姓名：'||et(i).ename||'，工资：'||et(i).sal);
 12  end loop;
 13  end;
 14  /
输入 job 的值:  clerk
姓名：SMITH，工资：800
姓名：ADAMS，工资：1100
姓名：JAMES，工资：950
姓名：MILLER，工资：1300
PL/SQL 过程已成功完成。
```

使用 LIMIT 子句限制提取行数

```
SQL> declare
  2  cursor ec is select * from emp;
  3  type emp_array_type is varray(5) of emp%rowtype;
  4  ea emp_array_type;
  5  begin
  6  open ec;
  7  loop
  8  fetch ec bulk collect into ea limit &rows;
  9  for i in 1..ea.count loop
 10  dbms_output.put_line('姓名：'||ea(i).ename||'，工资：'||ea(i).sal);
 11  end loop;
 12  exit when ec%notfound;
 13  end loop;
 14  close ec;
 15  end;
 16  /
输入 rows 的值:  4
姓名：SMITH，工资：800
姓名：ALLEN，工资：1600
姓名：WARD，工资：1250
姓名：JONES，工资：2975
姓名：MARTIN，工资：1250
```

```
姓名：BLAKE，工资：2850
姓名：CLARK，工资：2450
姓名：SCOTT，工资：3000
姓名：KING，工资：5000
姓名：TURNER，工资：1500
姓名：ADAMS，工资：1100
姓名：JAMES，工资：950
姓名：FORD，工资：3000
姓名：MILLER，工资：1300
PL/SQL 过程已成功完成。
```

使用 cursor 表达式

```
SQL> declare
  2  cursor dept_cursor(no number) is
  3  select a.dname,cursor(select * from emp where deptno=a.deptno)
  4  from dept a where a.deptno=no;
  5  type ref_cursor_type is ref cursor;
  6  ec ref_cursor_type;
  7  er emp%rowtype;
  8  vdname dept.dname%type;
  9  begin
 10  open dept_cursor(&dno);
 11  loop
 12  fetch dept_cursor into vdname,ec;
 13  exit when dept_cursor%notfound;
 14  dbms_output.put_line('部门名：'||vdname);
 15  loop
 16  fetch ec into er;
 17  exit when ec%notfound;
 18  dbms_output.put_line('----雇员名：'||er.ename||'，岗位：'||er.job);
 19  end loop;
 20  end loop;
 21  close dept_cursor;
 22  end;
 23  /
输入 dno 的值:  10
部门名：ACCOUNTING
----雇员名：CLARK，岗位：MANAGER
----雇员名：KING，岗位：PRESIDENT
----雇员名：MILLER，岗位：CLERK
PL/SQL 过程已成功完成。
```

第十八章 异常处理

示例：捕捉并处理异常

```
SQL> declare
  2  vename emp.ename%type;
  3  begin
  4  select ename into vename from emp where empno=&no;
  5  dbms_output.put_line('雇员名：'||vename);
  6  EXCEPTION
  7  when no_data_found then
  8  dbms_output.put_line('雇员号不正确，请核实雇员号！');
  9  end;
 10  /
输入 no 的值:  124
雇员号不正确，请核实雇员号!
PL/SQL 过程已成功完成。
```

Oracle 的预定义异常

捕捉并处理预定义异常：

```
SQL> declare
  2  vn emp.ename%type;
  3  begin
  4  select ename into vn from emp where sal=&salary;
  5  dbms_output.put_line('姓名：'||vn);
  6  EXCEPTION
  7  when NO_DATA_FOUND then
  8  dbms_output.put_line('不存在该工资值的雇员！');
  9  when TOO_MANY_ROWS then
 10  dbms_output.put_line('多个雇员具有该工资！');
 11  end;
 12  /
输入 salary 的值:  800
姓名：SMITH
PL/SQL 过程已成功完成。
输入 salary 的值:  3000
多个雇员具有该工资!
PL/SQL 过程已成功完成。
输入 salary 的值:  123
不存在该工资值的雇员!
PL/SQL 过程已成功完成。
```

使用自定义异常：

```
SQL> declare
  2  E_INTEGRITY EXCEPTION;
  3  E_NO_ROWS EXCEPTION;
  4  pragma EXCEPTION_INIT(E_INTEGRITY,-2291);
```

```
  5  name emp.ename%type:=lower('&name');
  6  dno emp.deptno%type:=&dno;
  7  begin
  8  update emp set deptno=dno where lower(ename)=name;
  9  if SQL%NOTFOUND then
 10  raise E_NO_ROWS;
 11  end if;
 12  EXCEPTION
 13  when E_INTEGRITY then
 14  dbms_output.put_line('该部门不存在');
 15  when E_NO_ROWS then
 16  dbms_output.put_line('该雇员不存在');
 17  end;
 18  /
输入 name 的值:  mary
输入 dno 的值:  80
该雇员不存在
PL/SQL 过程已成功完成。
```

使用函数 SQLCODE 和 SQLERRM

```
SQL> begin
  2  delete from dept where deptno=&dno;
  3  exception
  4  when others then
  5  dbms_output.put_line('错误号：'||SQLCODE);
  6  dbms_output.put_line('错误消息：'||SQLERRM);
  7  end;
  8  /
输入 dno 的值:  10
错误号：-2292
错误消息：ORA-02292: 违反完整约束条件 (SCOTT.FK_DEPTNO) - 已找到子记录日志
PL/SQL 过程已成功完成。
```

第十九章 过程和函数

过程和函数

建立无参过程：

```
SQL> create or replace procedure out_time
  2  is
  3  begin
  4  dbms_session.set_nls('NLS_DATE_FORMAT','''YYYY-MM-DD HH24:MI:SS''');
  5  dbms_output.put_line(sysdate);
  6  end;
```

```
  7  /
过程已创建。
SQL> exec out_time
2008-05-28 16:43:53
PL/SQL 过程已成功完成。
```

建立带输入参数的过程：

```
SQL> create or replace procedure add_emp(
  2  empno emp.empno%type,ename emp.ename%type,
  3  job emp.job%type,mgr emp.mgr%type,
  4  hiredate emp.hiredate%type,sal emp.sal%type,
  5  comm emp.comm%type,deptno emp.deptno%type)
  6  is begin
  7  insert into emp
  8  values(empno,ename,job,mgr,hiredate,sal,comm,deptno);
  9  end;
 10  /
过程已创建。
SQL> exec add_emp(1111,'MARY','CLERK',7369,SYSDATE,1200,null,30)
PL/SQL 过程已成功完成。
```

建立带输出参数的过程：

```
SQL> create or replace procedure update_sal
  2  (eno number,salary number,name out varchar2) is
  3  begin
  4  update emp set sal=salary where empno=eno
  5  return ename into name;
  6  end;
  7  /
过程已创建。
```

调用：

```
SQL> declare
  2  vn emp.ename%type;
  3  begin
  4  update_sal(&eno,&salary,vn);
  5  dbms_output.put_line('姓名：'||vn);
  6  end;
  7  /
输入 eno 的值:  1111
输入 salary 的值:  2500
姓名：MARY
PL/SQL 过程已成功完成。
```

建立带输入输出参数的过程：

```
SQL> create or replace procedure divide
```

```
  2  (num1 in out number,num2 in out number) is
  3  v1 number;
  4  v2 number;
  5  begin
  6  v1:=trunc(num1/num2);
  7  v2:=mod(num1,num2);
  8  num1:=v1;
  9  num2:=v2;
 10  end;
 11  /
```
过程已创建。

调用：

```
SQL> declare
  2  n1 number:=&n1;
  3  n2 number:=&n2;
  4  begin
  5  divide(n1,n2);
  6  dbms_output.put_line('商：'||n1||'，余数：'||n2);
  7  end;
  8  /
```
输入 n1 的值: 100

输入 n2 的值: 30

商：3，余数：10

PL/SQL 过程已成功完成。

删除过程：

```
SQL> drop procedure update_sal;
```
过程已丢弃。

无参数的函数：

```
SQL> create or replace function dtime return varchar2 is
  2  begin
  3  return to_char(sysdate,'YYYY"年"MM"月"DD"日"HH24"时"MI"分"SS"秒"');
  4  end;
  5  /
```
函数已创建。

调用：

```
SQL> begin
  2  dbms_output.put_line(dtime);
  3  end;
  4  /
```
2008 年 05 月 28 日 20 时 05 分 02 秒

PL/SQL 过程已成功完成。

带有输入参数的函数：

```
SQL> create or replace function getsal(name varchar2)
```

```
  2  return number as vsal emp.sal%type;
  3  begin
  4  select sal into vsal from emp where upper(ename)=upper(name);
  5  return vsal;end;
  6  /
函数已创建。
SQL> begin
  2  dbms_output.put_line('工资：'||getsal('&name'));
  3  end;
  4  /
输入 name 的值:  scott
工资：3000
PL/SQL 过程已成功完成。

带有输出参数的函数：
SQL> create or replace function getinfo
  2  (eno number,title out varchar2) return varchar2 as name emp.ename%type;
  3  begin
  4  select ename,job into name,title from emp where empno=eno;
  5  return name;
  6  end;
  7  /
函数已创建。
SQL> declare
  2  vn emp.ename%type;
  3  vj emp.job%type;
  4  begin
  5  vn:=getinfo(&eno,vj);
  6  dbms_output.put_line('姓名：'||vn||'，岗位：'||vj);
  7  end;
  8  /
输入 eno 的值:  7369
姓名：SMITH，岗位：CLERK
PL/SQL 过程已成功完成。

带有输入输出参数的函数：
SQL> create or replace function updateinfo
  2  (eno number,salch in out number) return varchar2 as name emp.ename%type;
  3  begin
  4  update emp set sal=sal+salch where empno=eno
  5  returning ename,sal into name,salch;
  6  return name;
  7  end;
  8  /
函数已创建。
SQL> declare
```

```
  2  ve emp.empno%type;
  3  vn emp.ename%type;
  4  vs emp.sal%type;
  5  begin
  6  ve:=&eno;
  7  vs:=&incre;
  8  vn:=updateinfo(ve,vs);
  9  dbms_output.put_line('姓名：'||vn||'，新工资：'||vs);
 10  end;
 11  /
输入 eno 的值:  7369
输入 incre 的值:  100
姓名：SMITH，新工资：900
PL/SQL 过程已成功完成。
```

删除函数：

```
SQL> drop function dtime;
函数已丢弃。
```

第二十章 触发器

1. DML 触发器

DML 触发器是指基于 DML 操作所建立的触发器

语句触发器：指当执行 DML 语句时被隐含执行的触发器

BEFORE 语句触发器

```
SQL> create or replace trigger tse
  2  before insert or update or delete on emp
  3  begin
  4  if to_char(sysdate,'DY','nls_date_language=AMERICAN')
  5  in ('SAT','SUN') then
  6  raise_application_error(-20001,'不能在休息日改变雇员信息');
  7  end if;
  8  end;
  9  /
触发器已创建
SQL> update emp set sal=sal*1.1 where deptno=10;
已更新 3 行。
```

AFTER 语句触发器

```
SQL> create table ut(
  2  host varchar2(30),statement varchar2(100),exectime date);
表已创建。
SQL> create or replace trigger tru
```

```
  2  after update on emp
  3  declare
  4  sqltxt ora_name_list_t;
  5  vstmt varchar2(100);
  6  m binary_integer;
  7  begin
  8  m:=ora_sql_txt(sqltxt);
  9  for i in 1..m loop
 10  vstmt:=vstmt||sqltxt(i);
 11  end loop;
 12  insert into ut values(
 13  sys_context('userenv','host'),vstmt,sysdate);
 14  end;
 15  /

触发器已创建
SQL> update emp set sal=2000 where empno=7369;
已更新 1 行。
SQL> select statement,exectime from ut;
STATEMENT                                                    EXECTIME
--------------------------------------------------------------------------------
update emp set sal=2000 where empno=7369       2008-05-28 20:42:33
```

条件谓词：
INSERTING：代表 insert 语句触发事件
UPDATING：代表 update 语句触发事件
DELETING：代表 delete 语句触发事件

触发器应用示例：
控制数据安全：

```
SQL> create or replace trigger trtime
  2  before insert or update or delete on emp
  3  begin
  4  if to_char(sysdate,'HH24') not between '9' and '17' then
  5  raise_application_error(-20101,'非工作时间');
  6  end if;
  7  end;
  8  /
触发器已创建
SQL> update emp set sal=3000 where empno=7788;
update emp set sal=3000 where empno=7788
       *
ERROR 位于第 1 行:
ORA-20101: 非工作时间
ORA-06512: 在"SCOTT.TRTIME", line 3
ORA-04088: 触发器 'SCOTT.TRTIME' 执行过程中出错
```

实现数据审计

```
SQL> create table emplog(
  2  name varchar2(10),time date);
表已创建。
SQL> create or replace trigger etr
  2  after delete on emp for each row
  3  begin
  4  insert into emplog values(:old.ename,sysdate);
  5  end;
  6  /
触发器已创建
SQL> delete from emp where empno=7788;
已删除 1 行。
SQL> select * from emplog;
NAME                           TIME
-----------------------------------------
SCOTT          2008-05-28 20:53:01
```

实现数据完整性

```
SQL> create or replace trigger salcheck
  2  before update of sal on emp for each row
  3  when (new.sal<old.sal or new.sal>1.2*old.sal)
  4  begin
  5  raise_application_error(-20931,'工资只升不降，并且升幅不能超过 20%');
  6  end;
  7  /
触发器已创建
SQL> update emp set sal=sal*1.25 where empno=7369;
update emp set sal=sal*1.25 where empno=7369
                                   *
ERROR 位于第 1 行:
ORA-20931: 工资只升不降，并且升幅不能超过 20%
ORA-06512: 在"SCOTT.SALCHECK", line 2
ORA-04088: 触发器 'SCOTT.SALCHECK' 执行过程中出错
```

实现参照完整性

(参照完整性是指在两张表之间具有主从关系。当删除主表数据时，需要首先删除从表的相关数据；当更新主表主键列时，需要首先更新从表相关数据)

利用触发器实现 dept 表的主键列和 emp 表的外部主键列的级联更新：

```
SQL> create or replace trigger update_cascade
  2  after update of deptno on dept for each row
  3  begin
  4  update emp set deptno=:new.deptno where deptno=:old.deptno;
  5  end;
  6  /
```

触发器已创建
SQL> update dept set deptno=50 where deptno=10;
已更新 1 行。
SQL> select ename from emp where deptno=50;
ENAME

CLARK
KING
MILLER

INSTEAD OF 触发器
只适用于视图，不能指定 before 和 after 选项

2. 事件触发器
大量的事件属性函数：
ora_database_name：返回数据库名，类型为 varchar2(50)
ora_login_user：返回登陆用户名，类型为 varchar2(30)
……
系统事件触发器
是指由特定系统事件所触发的触发器，包括四种事件：
STARTUP，SHUTDOWN，DB_ROLE_CHANGE，SERVERERROR
系统事件触发器只能由 sys 用户建立
并且 SHUTDOWN ABORT 命令不会触发 SHUTRDOWN 事件

示例：跟踪例程启动时间
SQL> conn sys/tiger as sysdba
已连接。
SQL> create table elog(event varchar2(30),time date);
表已创建。
SQL> create or replace trigger startuplog
  2 after startup on database
  3 begin
  4 insert into elog values(ora_sysevent,sysdate);
  5 end;
  6 /
触发器已创建
SQL> shutdown
ORACLE 例程已经关闭。
SQL> startup
ORACLE 例程已经启动。
SQL> select * from elog;
EVENT   TIME

STARTUP   28-5 月 -08

客户事件触发器

是指基于客户事件建立的触发器，客户事件是指与用户登录，注销，DDL 及 DCL 相关的事件

示例：建立登录触发器

```
SQL> create table logon(
  2  username varchar2(20),time date,addr varchar2(20));
表已创建。
SQL> create or replace trigger trlogon
  2  after logon on database
  3  begin
  4  insert into logon
  5  values(ora_login_user,sysdate,ora_client_ip_address);
  6  end;
  7  /
触发器已创建
SQL> conn scott/tiger@db
已连接。
SQL> conn sys/tiger as sysdba
已连接。
SQL> select * from logon;
USERNAME        TIME                    ADDR
-------------------- ---------- ----------------------------
SCOTT           28-5 月  -08       192.168.1.100
SYS             28-5 月  -08
```

示例：建立 DDL 触发器

```
SQL> create table tb(
  2  event varchar2(20),username varchar2(10),
  3  owner varchar2(10),objname varchar2(20),
  4  objtype varchar2(10),time date);
表已创建。
SQL> create or replace trigger trddl
  2  after ddl on scott.schema
  3  begin
  4  insert into tb values(
  5  ora_sysevent,ora_login_user,ora_dict_obj_owner,
  6  ora_dict_obj_name,ora_dict_obj_type,sysdate);
  7  end;
  8  /
触发器已创建
SQL> conn scott/tiger@db
已连接。
SQL> create table temp(n int);
表已创建。
SQL> drop table temp;
```

表已丢弃。
SQL> conn sys/tiger as sysdba;
已连接。
SQL> select username,event,objtype,objname from tb;

USERNAME EVENT OBJTYPE OBJNAME

SCOTT CREATE TABLE TEMP
SCOTT DROP TABLE TEMP

显示触发器信息：USER_TRIGGERS
禁止触发器：ALTER TRIGGER…….DISSABLE；
激活触发器：ALTER TRIGGER……ENABLE；
删除触发器：DROP TRIGGER …….；

附录

说明：在Oracle产品页上下载的 600 多M的那些 10g和 9i的版本不是完全版，里面的Samples也没有，建议大家到http://edelivery.oracle.com/下载完整的光盘镜像版。