本页已使用福昕阅读器进行编辑。
福昕软件(C)2005-2009，版权所有，
仅供试用。

韩顺平老师 oracle 教程笔记

1.Oracle 认证，与其它数据库比较，安装

Oracle 安装会自动的生成 sys 用户和 system 用户:

（1)sys 用户是超级用户,具有最高权限,具有 sysdba 角色,有 create database 的权限,该用户默认的密码是 change_on_install

（2）system 用户是管理操作员，权限也很大。具有 sysoper 角色，没有 create database 的权限，默认的密码是 manager

（3）一般讲，对数据库维护，使用 system 用户登录就可以拉

也就是说 sys 和 system 这两个用户最大的区别是在于有没有 create database 的权限。

2.Oracle 的基本使用--基本命令

sql*plus 的常用命令

连接命令

1.conn[ect]

用法：conn 用户名/密码@网络服务名[as sysdba/sysoper]当用特权用户身份连接时，必须带上 as sysdba 或是 as sysoper

2.disc[onnect]

说明：该命令用来断开与当前数据库的连接

3.psssw[ord]

说明：该命令用于修改用户的密码，如果要想修改其它用户的密码，需要用 sys/system 登录。

4.show user

说明：显示当前用户名

5.exit

说明：该命令会断开与数据库的连接，同时会退出 sql*plus

文件操作命令

1.start 和@

说明：运行 sql 脚本

案例：sql>@ d:\a.sql 或是 sql>start d:\a.sql

2.edit

说明：该命令可以编辑指定的 sql 脚本

案例：sql>edit d:\a.sql,这样会把 d:\a.sql 这个文件打开

3.spool

说明：该命令可以将 sql*plus 屏幕上的内容输出到指定文件中去。

案例：sql>spool d:\b.sql 并输入 sql>spool off

交互式命令

1.&

说明：可以替代变量，而该变量在执行时，需要用户输入。

```
select * from emp where job='&job';
```

2.edit

说明：该命令可以编辑指定的 sql 脚本

案例：SQL>edit d:\a.sql

3.spool

说明：该命令可以将 sql*plus 屏幕上的内容输出到指定文件中去。

spool d:\b.sql 并输入 spool off

显示和设置环境变量

概述：可以用来控制输出的各种格式，set show 如果希望永久的保存相关的设置，可以去修改 glogin.sql 脚本

本页已使用福昕阅读器进行编辑。
福昕软件(C)2005-2009，版权所有，
仅供试用。

1.linesize

说明：设置显示行的宽度，默认是 80 个字符

show linesize

set linesize 90

2.pagesize 说明：设置每页显示的行数目，默认是 14

用法和 linesize 一样

至于其它环境参数的使用也是大同小异

3.oracle 用户管理

oracle 用户的管理

创建用户

概述：在 oracle 中要创建一个新的用户使用 create user 语句，一般是具有 dba(数据库管理员)的权限才能使用。

create user 用户名 identified by 密码; (oracle 有个毛病，密码必须以字母开头，如果以字母开头，它不会创建用户)

给用户修改密码

概述：如果给自己修改密码可以直接使用

password 用户名

如果给别人修改密码则需要具有 dba 的权限，或是拥有 alter user 的系统权限

SQL> alter user 用户名 identified by 新密码

删除用户

概述：一般以 dba 的身份去删除某个用户，如果用其它用户去删除用户则需要具有 drop user 的权限。

比如 drop user 用户名 【cascade】

在删除用户时，注意：

如果要删除的用户，已经创建了表，那么就需要在删除的时候带一个参数 cascade;

用户管理综合案例

概述：创建的新用户是没有任何权限的，甚至连登陆的数据库的权限都没有，需要为其指定相应的权限。给一个用户赋权限使用命令 grant，回收权限使用命令 revoke。

为了给讲清楚用户的管理，这里我给大家举一个案例。

```
SQL> conn xiaoming/m12;
ERROR:
ORA-01045: user XIAOMING lacks CREATE SESSION privilege; logon denied
警告: 您不再连接到 ORACLE。
SQL> show user;
USER 为 ""
SQL> conn system/p;
已连接。
SQL> grant connect to xiaoming;
授权成功。
SQL> conn xiaoming/m12;
已连接。
SQL>
```

注意：grant connect to xiaoming;在这里，准确的讲，connect 不是权限，而是角色。

现在说下对象权限，现在要做这么件事情：

* 希望 xiaoming 用户可以去查询 emp 表
* 希望 xiaoming 用户可以去查询 scott 的 emp 表

grant select on emp to xiaoming

* 希望 xiaoming 用户可以去修改 scott 的 emp 表

grant update on emp to xiaoming

* 希望 xiaoming 用户可以去修改/删除，查询，添加 scott 的 emp 表

grant all on emp to xiaoming

* scott 希望收回 xiaoming 对 emp 表的查询权限

revoke select on emp from xiaoming

//对权限的维护。

* 希望 xiaoming 用户可以去查询 scott 的 emp 表/还希望 xiaoming 可以把这个权限继续给别人。

--如果是对象权限，就加入 with grant option

grant select on emp to xiaoming with grant option

我的操作过程：

SQL> conn scott/tiger;

已连接。

SQL> grant select on scott.emp to xiaoming with grant option;

授权成功。

SQL> conn system/p;

已连接。

SQL> create user xiaohong identified by m123;

用户已创建。

SQL> grant connect to xiaohong;

授权成功。

SQL> conn xiaoming/m12;

已连接。

SQL> grant select on scott.emp to xiaohong;

授权成功。

--如果是系统权限。

system 给 xiaoming 权限时：

grant connect to xiaoming with admin option

问题：如果 scott 把 xiaoming 对 emp 表的查询权限回收，那么 xiaohong 会怎样？

答案：被回收。

下面是我的操作过程：

SQL> conn scott/tiger;

已连接。

SQL> revoke select on emp from xiaoming;

撤销成功。

SQL> conn xiaohong/m123;

已连接。

SQL> select * from scott.emp;

select * from scott.emp

第 1 行出现错误:

ORA-00942: 表或视图不存在

结果显示：小红受到诛连了。。

使用 profile 管理用户口令

概述：profile 是口令限制，资源限制的命令集合，当建立数据库的，oracle 会自动建立名称为 default 的 profile。当建立用户没

有指定 profile 选项，那么 oracle 就会将 default 分配给用户。

1.账户锁定

概述：指定该账户(用户)登陆时最多可以输入密码的次数，也可以指定用户锁定的时间(天)一般用 dba 的身份去执行该命令。

例子：指定 scott 这个用户最多只能尝试 3 次登陆，锁定时间为 2 天，让我们看看怎么实现。

创建 profile 文件

SQL> create profile lock_account limit failed_login_attempts 3 password_lock_time 2;

SQL> alter user scott profile lock_account;

2.给账户(用户)解锁

SQL> alter user tea account unlock;

3.终止口令

为了让用户定期修改密码可以使用终止口令的指令来完成，同样这个命令也需要 dba 的身份来操作。

例子：给前面创建的用户 tea 创建一个 profile 文件，要求该用户每隔 10 天要修改自己的登陆密码，宽限期为 2 天。看看怎么做。

SQL> create profile myprofile limit password_life_time 10 password_grace_time 2;

SQL> alter user tea profile myprofile;

口令历史

概述：如果希望用户在修改密码时，不能使用以前使用过的密码，可使用口令历史，这样 oracle 就会将口令修改的信息存放到数据字典中，这样当用户修改密码时，oracle 就会对新旧密码进行比较，当发现新旧密码一样时，就提示用户重新输入密码。

例子：

1）建立 profile

SQL>create profile password_history limit password_life_time 10 password_grace_time 2 password_reuse_time 10

password_reuse_time //指定口令可重用时间即 10 天后就可以重用

2）分配给某个用户

删除 profile

概述：当不需要某个 profile 文件时，可以删除该文件。

SQL> drop profile password_history 【casade】

注意：文件删除后，用这个文件去约束的那些用户通通也都被释放了。。

加了 casade，就会把级联的相关东西也给删除掉

4. oracle 表的管理(数据类型，表创建删除，数据 CRUD 操作)

期望目标

- 1. 掌握 oracle 表的管理（创建/维护）
- 2. 掌握对 oracle 表的各种查询技巧
- 3. 学会创建新的 oracle 数据库

oracle 的表的管理

表名和列的命名规则

- 必须以字母开头
- 长度不能超过 30 个字符
- 不能使用 oracle 的保留字
- 只能使用如下字符 A-Z，a-z，0-9，$,#等

oracle 支持的数据类型v

字符类

char　　定长 最大 2000 个字符。

例子：char(10)　‘小韩’前四个字符放‘小韩’，后添 6 个空格补全 如‘小韩　　’

varchar2(20)　变长　最大 4000 个字符。

例子：varchar2（10）　‘小韩’ oracle 分配四个字符。这样可以节省空间。

clob(character large object) 字符型大对象 最大 4G

char 查询的速度极快浪费空间，查询比较多的数据用。

varchar 节省空间

数字型

number 范围 -10 的 38 次方 到 10 的 38 次方

可以表示整数，也可以表示小数

number(5,2)

表示一位小数有 5 位有效数，2 位小数

范围：-999.99 到 999.99

number(5)

表示一个 5 位整数

范围 99999 到-99999

日期类型

date 包含年月日和时分秒　　oracle 默认格式　1-1 月-1999

timestamp 这是 oracle9i 对 date 数据类型的扩展。可以精确到毫秒。

ν 图片

blob 二进制数据 可以存放图片/声音　4G　　一般来讲，在真实项目中是不会把图片和声音真的往数据库里存放，一般存放图片、视频的路径，如果安全需要比较高的话，则放入数据库。

怎样创建表

建表

```
--学生表
create table student (       ---表名
                  xh             number(4),      --学号
                  xm       varchar2(20),      --姓名
                  sex            char(2),          --性别
                  birthday date,                   --出生日期
                  sal            number(7,2)      --奖学金
);
--班级表
CREATE TABLE class(
       classId NUMBER(2),
       cName VARCHAR2(40)
);
```

修改表

ν 添加一个字段

```
SQL>ALTER TABLE student add (classId NUMBER(2));
```

ν 修改一个字段的长度

```
SQL>ALTER TABLE student MODIFY (xm VARCHAR2(30));
```

ν 修改字段的类型/或是名字（不能有数据） 不建议做

```
SQL>ALTER TABLE student modify (xm CHAR(30));
```

ν 删除一个字段　不建议做(删了之后，顺序就变了。加就没问题，应为是加在后面)

```
SQL>ALTER TABLE student DROP COLUMN sal;
```

本页已使用福昕阅读器进行编辑。
福昕软件(C)2005-2009，版权所有，
仅供试用。

v 修改表的名字　　很少有这种需求

SQL>RENAME student TO stu;

v 删除表

SQL>DROP TABLE student;

添加数据

所有字段都插入数据

INSERT INTO student VALUES ('A001', '张三', '男', '01-5 月-05', 10);

oracle 中默认的日期格式 'dd-mon-yy' dd 日子（天） mon 月份 yy 2 位的年 '09-6 月-99' 1999 年 6 月 9 日

修改日期的默认格式（临时修改，数据库重启后仍为默认；如要修改需要修改注册表）

ALTER SESSION SET NLS_DATE_FORMAT ='yyyy-mm-dd';

修改后，可以用我们熟悉的格式添加日期类型：

INSERT INTO student VALUES ('A002', 'MIKE', '男', '1905-05-06', 10);

插入部分字段

INSERT INTO student(xh, xm, sex) VALUES ('A003', 'JOHN', '女');

插入空值

INSERT INTO student(xh, xm, sex, birthday) VALUES ('A004', 'MARTIN', '男', null);

问题来了，如果你要查询 student 表里 birthday 为 null 的记录，怎么写 sql 呢？

错误写法：select * from student where birthday = null;

正确写法：select * from student where birthday is null;

如果要查询 birthday 不为 null,则应该这样写：

select * from student where birthday is not null;

修改数据

v 修改一个字段

UPDATE student SET sex = '女' WHERE xh = 'A001';

v 修改多个字段

UPDATE student SET sex = '男', birthday = '1984-04-01' WHERE xh = 'A001';

修改含有 null 值的数据

不要用 = null 而是用 is null;

SELECT * FROM student WHERE birthday IS null;

v 删除数据

DELETE FROM student;

删除所有记录，表结构还在，写日志，可以恢复的，速度慢。

Delete 的数据可以恢复。

savepoint a; --创建保存点

DELETE FROM student;

rollback to a; --恢复到保存点

一个有经验的 DBA，在确保完成无误的情况下要定期创建还原点。

DROP TABLE student; --删除表的结构和数据;

delete from student WHERE xh = 'A001'; --删除一条记录;

truncate TABLE student; --删除表中的所有记录，表结构还在，不写日志，无法找回删除的记录，速度快。

5.oracle 表查询(1)

在我们讲解的过程中我们利用 scott 用户存在的几张表（emp，dept）为大家演示如何使用 select 语句，select 语句在软件编程中非常有用，希望大家好好的掌握。

emp 雇员表
clerk　普员工
salesman 销售
manager　经理
analyst 分析师
president　总裁

mgr　上级的编号
hiredate 入职时间
sal 月工资
comm 奖金
deptno 部门

dept 部门表
deptno 部门编号
accounting 财务部
research　研发部
operations 业务部
loc 部门所在地点

salgrade　　工资级别
grade　　级别
losal　　最低工资
hisal　　最高工资

简单的查询语句
ν 查看表结构
DESC emp;
ν 查询所有列
SELECT * FROM dept;
切忌动不动就用 select *
SET TIMING ON; 打开显示操作时间的开关，在下面显示查询时间。
CREATE TABLE users(userId VARCHAR2(10), uName VARCHAR2 (20), uPassw VARCHAR2(30));
INSERT INTO users VALUES('a0001', '啊啊啊啊', 'aaaaaaaaaaaaaaaaaaaaaa');
--从自己复制，加大数据量 大概几万行就可以了　可以用来测试 sql 语句执行效率
INSERT INTO users (userId,UNAME,UPASSW) SELECT * FROM users;
SELECT COUNT (*) FROM users;统计行数

ν 查询指定列
SELECT ename, sal, job, deptno FROM emp;
ν 如何取消重复行 DISTINCT
SELECT DISTINCT deptno, job FROM emp;
?查询 SMITH 所在部门，工作，薪水
SELECT deptno,job,sal FROM emp WHERE ename = 'SMITH';
注意：oracle 对内容的大小写是区分的，所以 ename='SMITH'和 ename='smith'是不同的

ν 使用算术表达式　nvl　null

问题：如何显示每个雇员的年工资？

SELECT sal*13+nvl(comm, 0)*13 "年薪" , ename, comm FROM emp;

ν 使用列的别名

SELECT ename "姓名", sal*12 AS "年收入" FROM emp;

ν 如何处理 null 值

使用 nvl 函数来处理

ν 如何连接字符串（||）

SELECT ename　|| ' is a ' || job FROM emp;

ν 使用 where 子句

问题：如何显示工资高于 3000 的 员工？

SELECT * FROM emp WHERE sal > 3000;

问题：如何查找 1982.1.1 后入职的员工？

SELECT ename,hiredate FROM emp WHERE hiredate >'1-1 月-1982';

问题：如何显示工资在 2000 到 3000 的员工？

SELECT ename,sal FROM emp WHERE sal >=2000 AND sal <= 3000;

ν 如何使用 like 操作符

%：表示 0 到多个字符　_：表示任意单个字符

问题：如何显示首字符为 S 的员工姓名和工资？

SELECT ename,sal FROM emp WHERE ename like 'S%';

如何显示第三个字符为大写 O 的所有员工的姓名和工资？

SELECT ename,sal FROM emp WHERE ename like '__O%';

ν 在 where 条件中使用 in

问题：如何显示 empno 为 7844, 7839,123,456 的雇员情况？

SELECT * FROM emp WHERE empno in (7844, 7839,123,456);

ν 使用 is null 的操作符

问题：如何显示没有上级的雇员的情况？

错误写法：select * from emp where mgr = '';

正确写法：SELECT * FROM emp WHERE mgr is null;

6.oracle 表查询(2)

ν 使用逻辑操作符号

问题：查询工资高于 500 或者是岗位为 MANAGER 的雇员，同时还要满足他们的姓名首字母为大写的 J?

SELECT * FROM emp WHERE (sal >500 or job = 'MANAGER') and ename LIKE 'J%';

ν 使用 order by 字句　　默认 asc

问题：如何按照工资的从低到高的顺序显示雇员的信息？

SELECT * FROM emp ORDER by sal;

问题：按照部门号升序而雇员的工资降序排列

SELECT * FROM emp ORDER by deptno, sal DESC;

ν 使用列的别名排序

问题：按年薪排序

select ename, (sal+nvl(comm,0))*12 "年薪" from emp order by "年薪" asc;

别名需要使用“”号圈中,英文不需要“”号

本页已使用福昕阅读器进行编辑。
福昕软件(C)2005-2009，版权所有，
仅供试用。

ν 分页查询

等学了子查询再说吧。。。。。。。。

Clear 清屏命令

oracle 表复杂查询

ν 说明

在实际应用中经常需要执行复杂的数据统计，经常需要显示多张表的数据，现在我们给大家介绍较为复杂的 select 语句

数据分组 ——max，min， avg， sum， count

问题：如何显示所有员工中最高工资和最低工资？

SELECT MAX(sal),min(sal) FROM emp e;

最高工资那个人是谁？

错误写法：select ename, sal from emp where sal=max(sal);

正确写法：select ename, sal from emp where sal=(select max(sal) from emp);

注意：select ename, max(sal) from emp;这语句执行的时候会报错，说 ORA-00937：非单组分组函数。因为 max 是分组函数，而 ename 不是分组函数.....

但是 select min(sal), max(sal) from emp;这句是可以执行的。因为 min 和 max 都是分组函数，就是说：如果列里面有一个分组函数，其它的都必须是分组函数，否则就出错。这是语法规定的

问题：如何显示所有员工的平均工资和工资总和？

问题：如何计算总共有多少员工问题：如何

扩展要求：

查询最高工资员工的名字，工作岗位

SELECT ename, job, sal FROM emp e where sal = (SELECT MAX(sal) FROM emp);

显示工资高于平均工资的员工信息

SELECT * FROM emp e where sal > (SELECT AVG(sal) FROM emp);

ν group by 和 having 子句

group by 用于对查询的结果分组统计，

having 子句用于限制分组显示结果。

问题：如何显示每个部门的平均工资和最高工资？

SELECT AVG(sal), MAX(sal), deptno FROM emp GROUP by deptno;

（注意：这里暗藏了一点，如果你要分组查询的话，分组的字段 deptno 一定要出现在查询的列表里面，否则会报错。因为分组的字段都不出现的话，就没办法分组了）

问题：显示每个部门的每种岗位的平均工资和最低工资？

SELECT min(sal), AVG(sal), deptno, job FROM emp GROUP by deptno, job;

问题：显示平均工资低于 2000 的部门号和它的平均工资？

SELECT AVG(sal), MAX(sal), deptno FROM emp GROUP by deptno having AVG(sal) < 2000;

ν 对数据分组的总结

1 分组函数只能出现在选择列表、having、order by 子句中(不能出现在 where 中)

2 如果在 select 语句中同时包含有 group by, having, order by 那么它们的顺序是 group by, having, order by

3 在选择列中如果有列、表达式和分组函数，那么这些列和表达式必须有一个出现在 group by 子句中，否则就会出错。

如 SELECT deptno, AVG(sal), MAX(sal) FROM emp GROUP by deptno HAVING AVG(sal) < 2000;

这里 deptno 就一定要出现在 group by 中

多表查询

ν 说明

多表查询是指基于两个和两个以上的表或是视图的查询。在实际应用中，查询单个表可能不能满足你的需求，（如显示 sales 部门位置和其员工的姓名），这种情况下需要使用到（dept 表和 emp 表）

本页已使用福昕阅读器进行编辑。
福昕软件(C)2005-2009，版权所有，
仅供试用。

问题：显示雇员名，雇员工资及所在部门的名字【笛卡尔集】？

规定：多表查询的条件是 至少不能少于 表的个数-1 才能排除笛卡尔集

（如果有 N 张表联合查询，必须得有 N-1 个条件，才能避免笛卡尔集合）

```
SELECT e.ename, e.sal, d.dname FROM emp e, dept d WHERE e.deptno = d.deptno;
```

问题：显示部门号为 10 的部门名、员工名和工资？

```
SELECT d.dname, e.ename, e.sal FROM emp e, dept d WHERE e.deptno = d.deptno and e.deptno = 10;
```

问题：显示各个员工的姓名，工资及工资的级别？

先看 salgrade 的表结构和记录

```
SQL>select * from salgrade;
```

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

```
SELECT e.ename, e.sal, s.grade FROM emp e, salgrade s WHERE e.sal BETWEEN s.losal AND s.hisal;
```

扩展要求：

问题：显示雇员名，雇员工资及所在部门的名字，并按部门排序？

```
SELECT e.ename, e.sal, d.dname FROM emp e, dept d WHERE e.deptno = d.deptno ORDER by e.deptno;
```

（注意：如果用 group by，一定要把 e.deptno 放到查询列里面）

ν 自连接

自连接是指在同一张表的连接查询

问题：显示某个员工的上级领导的姓名？

比如显示员工‘FORD’的上级

```
SELECT worker.ename, boss.ename FROM emp worker,emp boss WHERE worker.mgr = boss.empno AND worker.ename = 'FORD';
```

子查询

ν 什么是子查询

子查询是指嵌入在其他 sql 语句中的 select 语句，也叫嵌套查询。

ν 单行子查询

单行子查询是指只返回一行数据的子查询语句

请思考：显示与 SMITH 同部门的所有员工？

思路：

1 查询出 SMITH 的部门号

```
select deptno from emp WHERE ename = 'SMITH';
```

2 显示

```
SELECT * FROM emp WHERE deptno = (select deptno from emp WHERE ename = 'SMITH');
```

数据库在执行 sql 是从左到右扫描的， 如果有括号的话，括号里面的先被优先执行。

ν 多行子查询

多行子查询指返回多行数据的子查询

请思考：如何查询和部门 10 的工作相同的雇员的名字、岗位、工资、部门号

```
SELECT DISTINCT job FROM emp WHERE deptno = 10;
SELECT * FROM emp WHERE job IN (SELECT DISTINCT job FROM emp WHERE deptno = 10);
```

（注意：不能用 job=..，因为等号=是一对一的）

ν 在多行子查询中使用 all 操作符

本页已使用福昕阅读器进行编辑。
福昕软件(C)2005-2009，版权所有，
仅供试用。

问题：如何显示工资比部门 30 的所有员工的工资高的员工的姓名、工资和部门号？

SELECT ename, sal, deptno FROM emp WHERE sal > all (SELECT sal FROM emp WHERE deptno = 30);

扩展要求：

大家想想还有没有别的查询方法。

SELECT ename, sal, deptno FROM emp WHERE sal > (SELECT MAX(sal) FROM emp WHERE deptno = 30);

执行效率上， 函数高得多

ν 在多行子查询中使用 any 操作符

问题：如何显示工资比部门 30 的任意一个员工的工资高的员工姓名、工资和部门号？

SELECT ename, sal, deptno FROM emp WHERE sal > ANY (SELECT sal FROM emp WHERE deptno = 30);

扩展要求：

大家想想还有没有别的查询方法。

SELECT ename, sal, deptno FROM emp WHERE sal > (SELECT min(sal) FROM emp WHERE deptno = 30);

ν 多列子查询

单行子查询是指子查询只返回单列、单行数据，多行子查询是指返回单列多行数据，都是针对单列而言的，而多列子查询是指查询返回多个列数据的子查询语句。

请思考如何查询与 SMITH 的部门和岗位完全相同的所有雇员。

SELECT deptno, job FROM emp WHERE ename = 'SMITH';

SELECT * FROM emp WHERE (deptno, job) = (SELECT deptno, job FROM emp WHERE ename = 'SMITH');

ν 在 from 子句中使用子查询

请思考：如何显示高于自己部门平均工资的员工的信息

思路：

1. 查出各个部门的平均工资和部门号

SELECT deptno, AVG(sal) mysal FROM emp GROUP by deptno;

2. 把上面的查询结果看做是一张子表

SELECT e.ename, e.deptno, e.sal, ds.mysal FROM emp e, (SELECT deptno, AVG(sal) mysal FROM emp GROUP by deptno) ds WHERE e.deptno = ds.deptno AND e.sal > ds.mysal;

如何衡量一个程序员的水平？

网络处理能力， 数据库， 程序代码的优化程序的效率要很高

小总结：

在这里需要说明的当在 from 子句中使用子查询时，该子查询会被作为一个视图来对待，因此叫做内嵌视图，当在 from 子句中使用子查询时，必须给子查询指定别名。

注意：别名不能用 as，如：SELECT e.ename, e.deptno, e.sal, ds.mysal FROM emp e, (SELECT deptno, AVG(sal) mysal FROM emp GROUP by deptno) as ds WHERE e.deptno = ds.deptno AND e.sal > ds.mysal;

在 ds 前不能加 as，否则会报错 （给表取别名的时候，不能加 as；但是给列取别名，是可以加 as 的）

ν 分页查询

按雇员的 id 号升序取出

oracle 的分页一共有三种方式

1.根据 rowid 来分

select * from t_xiaoxi where rowid in (select rid from (select rownum rn, rid from(select rowid rid, cid from t_xiaoxi order by cid desc) where rownum<10000) where rn>9980) order by cid desc;

执行时间 0.03 秒

2.按分析函数来分

select * from (select t.*, row_number() over(order by cid desc) rk from t_xiaoxi t) where rk<10000 and rk>9980;

执行时间 1.01 秒

3.按 rownum 来分

本页已使用福昕阅读器进行编辑。
福昕软件(C)2005-2009，版权所有，
仅供试用。

select * from (select t.*,rownum rn from(select * from t_xiaoxi order by cid desc)t where rownum<10000) where rn>9980;

执行时间 0.1 秒

其中 t_xiaoxi 为表名称，cid 为表的关键字段，取按 cid 降序排序后的第 9981-9999 条记录，t_xiaoxi 表有 70000 多条记录。

个人感觉 1 的效率最好，3 次之，2 最差。

//测试通过的分页查询 okokok

select * from (select a1.*, rownum rn from(select ename,job from emp) a1 where rownum<=10)where rn>=5;

下面最主要介绍第三种：按 rownum 来分

1. rownum 分页

SELECT * FROM emp;

2. 显示 rownum[oracle 分配的]

SELECT e.*, ROWNUM rn FROM (SELECT * FROM emp) e;

rn 相当于 Oracle 分配的行的 ID 号

3.挑选出 6—10 条记录

先查出 1-10 条记录

SELECT e.*, ROWNUM rn FROM (SELECT * FROM emp) e WHERE ROWNUM <= 10;

如果后面加上 rownum>=6 是不行的，

4. 然后查出 6-10 条记录

SELECT * FROM (SELECT e.*, ROWNUM rn FROM (SELECT * FROM emp) e WHERE ROWNUM <= 10) WHERE rn >= 6;

5. 几个查询变化

a. 指定查询列，只需要修改最里层的子查询

只查询雇员的编号和工资

SELECT * FROM (SELECT e.*, ROWNUM rn FROM (SELECT ename, sal FROM emp) e WHERE ROWNUM <= 10) WHERE rn >= 6;

b. 排序查询，只需要修改最里层的子查询

工资排序后查询 6-10 条数据

SELECT * FROM (SELECT e.*, ROWNUM rn FROM (SELECT ename, sal FROM emp ORDER by sal) e WHERE ROWNUM <= 10) WHERE rn >= 6;

ν 用查询结果创建新表

这个命令是一种快捷的建表方式

CREATE TABLE mytable (id, name, sal, job, deptno) as SELECT empno, ename, sal, job, deptno FROM emp;

创建好之后，desc mytable; 和 select * from mytable;看看结果如何？

合并查询

ν 合并查询

有时在实际应用中，为了合并多个 select 语句的结果，可以使用集合操作符号 union，union all，intersect，minus

多用于数据量比较大的数据局库，运行速度快。

1). union

该操作符用于取得两个结果集的并集。当使用该操作符时，会自动去掉结果集中重复行。

SELECT ename, sal, job FROM emp WHERE sal >2500

UNION

SELECT ename, sal, job FROM emp WHERE job = 'MANAGER';

2).union all

该操作符与 union 相似，但是它不会取消重复行，而且不会排序。

SELECT ename, sal, job FROM emp WHERE sal >2500

UNION ALL

SELECT ename, sal, job FROM emp WHERE job = 'MANAGER';

该操作符用于取得两个结果集的并集。当使用该操作符时，会自动去掉结果集中重复行。

3). intersect

使用该操作符用于取得两个结果集的交集。

```
SELECT ename, sal, job FROM emp WHERE sal >2500
INTERSECT
SELECT ename, sal, job FROM emp WHERE job = 'MANAGER';
```

4). minus

使用改操作符用于取得两个结果集的差集，他只会显示存在第一个集合中，而不存在第二个集合中的数据。

```
SELECT ename, sal, job FROM emp WHERE sal >2500
MINUS
SELECT ename, sal, job FROM emp WHERE job = 'MANAGER';
```

（MINUS 就是减法的意思）

创建数据库有两种方法：

1). 通过 oracle 提供的向导工具。 √

database Configuration Assistant 【数据库配置助手】

2).我们可以用手工步骤直接创建。

7.java 操作 oracle

内容介绍

1.上节回顾

2.java 程序如何操作 oracle √

3.如何在 oracle 中操作数据

4.oracle 事务处理

5.sql 函数的使用 √

期望目标

1.掌握 oracle 表对数据操作技巧

2.掌握在 java 程序中操作 oracle

3.理解 oracle 事物概念

4.掌握 oracle 各种 sql 函数

java 连接 oracle

ν 介绍

前面我们一直在 plsql 中操作 oracle，那么如何在 java 程序中操作数据库呢？

下面我们举例说明，写一个 java，分页显示 emp 表的用户信息。

```
package com.sp;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
//演示  如何使用 jdbc_odbc 桥连接方式
public class TestOracle {
	public static void main(String[] args) {
		try {
			// 1.加载驱动
			Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

```
                        // 2.得到连接
                        Connection ct = DriverManager.getConnection(
                                        "jdbc.odbc:testConnectOracle", "scott",
"tiger");
                        // 从下面开始，和 SQL Server 一模一样
                        Statement sm = ct.createStatement();
                        ResultSet rs = sm.executeQuery("select * from emp");
                        while (rs.next()) {
                                //用户名
                                System.out.println("用户名： "+rs.getString(2));
                                //默认是从 1 开始编号的
                        }
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }
}
```

在得到连接那里，要去配置数据源，点击控制面板-->系统和安全-->管理工具-->数据源(ODBC),
打开后点添加,如图：
可以看到，有个 Oracle in OraDb10g_home1 的驱动，它是 Oracle 安装完后自动加上去的。 选中后，点完成，再填如下信息,如图：
这样配好后基本就可以了，但为了安全起见，建议大家测试一下，点击 Test Connection 按钮，测试通过后点 ok,然后数据源就生成了，如图：
然后把数据源名称写进 jdbc.odbc:???里。
这里要注意：jdbcodbc 能不能远程连接呢？不能远程连接，也就是你这样写的话就意味着 java 程序和 oracle 数据库应该是在同一台机器上，因为这里没有指定 IP 地址，肯定默认就是本地。 如果要远程连，就用 jdbc,jdbc 是可以远程连的。
运行 TestOracle.java,控制台输出.....
可惜我没运行成功，说

```
java.sql.SQLException: No suitable driver found for jdbc.odbc:testConnectOracle
at java.sql.DriverManager.getConnection(Unknown Source)
at java.sql.DriverManager.getConnection(Unknown Source)
at com.sp.TestOracle.main(TestOracle.java:18)
```

不知道为什么。。。
接下来讲解用 JDBC 的方式连接 Oracle

```
package com.sp;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
//使用 jdbc 连接 oracle
public class TestOracle2 {
        public static void main(String[] args) {
                try {
                        // 1.加载驱动
```

```
                        Class.forName("oracle.jdbc.driver.OracleDriver");
                        // 2.得到连接
                        Connection ct = DriverManager.getConnection
("jdbc:oracle:thin:@127.0.0.1:1521:orcl", "scott", "tiger");
                        // 从下面开始，和 SQL Server 一模一样
                        Statement sm = ct.createStatement();
                        ResultSet rs = sm.executeQuery("select * from emp");
                        while (rs.next()) {
                                //用户名
                                System.out.println("用户名： "+rs.getString(2));
                                //默认是从 1 开始编号的
                        }
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }
}
```

记得要把驱动包引入，classes12.jar

运行，。。。。 再次可惜，我还是没运行成功，错误是：

```
java.sql.SQLException: Io 异常: The Network Adapter could not establish the
connection
at oracle.jdbc.dbaccess.DBError.throwSqlException(DBError.java:134)
at oracle.jdbc.dbaccess.DBError.throwSqlException(DBError.java:179)
at oracle.jdbc.dbaccess.DBError.throwSqlException(DBError.java:334)
at oracle.jdbc.driver.OracleConnection.<init>(OracleConnection.java:418)
at oracle.jdbc.driver.OracleDriver.getConnectionInstance
(OracleDriver.java:521)
at oracle.jdbc.driver.OracleDriver.connect(OracleDriver.java:325)
at java.sql.DriverManager.getConnection(Unknown Source)
at java.sql.DriverManager.getConnection(Unknown Source)
at com.sp.TestOracle2.main(TestOracle2.java:18)
```

我也不知道为什么。。。 幽怨了。。

接下来建个 web project，来测试 oracle 的分页，挺麻烦，不记录了。。

在 oracle 中操作数据 - 使用特定格式插入日期值

ν 使用 to_date 函数

请大家思考： 如何插入列带有日期的表，并按照年-月-日的格式插入？

```
insert into emp values (9998, 'xiaohong', 'MANAGER', 7782, to_date('1988-12-
12', 'yyyy-mm-dd'), 78.9, 55.33, 10);
```

注意：

```
insert into emp values (9998, 'xiaohong', 'MANAGER', 7782, '12-12 月-1988',
78.9, 55.33, 10);
```

这句语句是可以成功运行的

使用子查询插入数据

ν 介绍

当使用 valus 子句时，一次只能插入一行数据，当使用子查询插入数据时，一条 inset 语句可以插

入大量的数据。当处理行迁移或者装载外部表的数据到数据库时，可以使用子查询来插入数据。
把 emp 表中 10 号部门的数据导入到新表中
create table kkk(myId number(4), myName varchar2(50), myDept number(5));
insert into kkk (myId, myName, myDept) select empno, ename, deptno from emp where deptno = 10;
ν 介绍
使用 update 语句更新数据时，既可以使用表达式或者数值直接修改数据，也可以使用子查询修改数据。
问题：希望员工 SCOTT 的岗位、工资、补助与 SMITH 员工一样。
update emp set(job, sal, comm)=(select job, sal, comm from emp where ename='SMITH')
where ename='SCOTT';

8.oracle 中事务处理

ν 什么是事务
事务用于保证数据的一致性，它由一组相关的 dml 语句组成，该组的 dml(数据操作语言，增删改，没有查询)语句要么全部成功，要么全部失败。
如：网上转账就是典型的要用事务来处理，用于保证数据的一致性。
dml 数据操作语言
银行转账、QQ 申请、车票购买
ν 事务和锁
当执行事务操作时（dml 语句），oracle 会在被作用的表上加锁，防止其它用户修改表的结构。这里对我们的用户来来讲是非常重要的。
.....其它进程排序，知道 1 号进程完成，锁打开，2 号进程进入。依次进行，如果有进程级别较高的，可以插队。
ν 提交事务
当执行用 commit 语句可以提交事务。当执行了 commit 语句之后，会确认事务的变化、结束事务。删除保存点、释放锁，当使用 commit 语句结束事务之后，其它会话将可以查看到事务变化后的新数据。
保存点就是为回退做的。保存点的个数没有限制
ν 回退事务
在介绍回退事务前，我们先介绍一下保存点（savepoint）的概念和作用。保存点是事务中的一点。用于取消部分事务，当结束事务时，会自动的删除该事务所定义的所有保存点。当执行 rollback 时，通过指定保存点可以回退到指定的点，这里我们作图说明。
ν 事务的几个重要操作
1.设置保存点 savepoint a
2.取消部分事务 rollback to a
3.取消全部事务 rollback
注意：这个回退事务，必须是没有 commit 前使用的；如果事务提交了，那么无论你刚才做了多少个保存点，都统统没有。
如果没有手动执行 commit,而是 exit 了，那么会自动提交
ν java 程序中如何使用事务
在 java 操作数据库时，为了保证数据的一致性，比如账户操作（1）从一个账户中减掉 10$（2）在另一个账户上加入 10$,我们看看如何使用事务？

```
package com.sp;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
```

```
import java.sql.Statement;
public class TestTrans {
	public static void main(String[] args) {
		try {
			// 1.加载驱动
			Class.forName("oracle.jdbc.driver.OracleDriver");
			// 2.得到连接
			Connection ct = DriverManager.getConnection(
					"jdbc:oracle:thin:@127.0.0.1:1521:orcl", "scott", "tiger");
			Statement sm = ct.createStatement();
			// 从 scott 的 sal 中减去 100
			sm.executeUpdate("update emp set sal=sal-100 where ename='SCOTT'");
			int i = 7 / 0;
			// 给 smith 的 sal 加上 100
			sm.executeUpdate("update emp set sal=sal+100 where ename='SMITH'");
			// 关闭打开的资源
			sm.close();
			ct.close();
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
}
```

运行，会出现异常，查看数据库，SCOTT 的 sal 减了 100，但是 SMITH 的 sal 却不变，很可怕。。。

我们怎样才能保证，这两个操作要么同时成功，要么同时失败呢？

```
package com.sp;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
public class TestTrans {
	public static void main(String[] args) {
		Connection ct = null;
		try {
			// 1.加载驱动
			Class.forName("oracle.jdbc.driver.OracleDriver");
			// 2.得到连接
			ct = DriverManager.getConnection(
					"jdbc:oracle:thin:@127.0.0.1:1521:orcl", "scott", "tiger");
			// 加入事务处理
			ct.setAutoCommit(false);// 设置不能默认提交
			Statement sm = ct.createStatement();
			// 从 scott 的 sal 中减去 100
			sm.executeUpdate("update emp set sal=sal-100 where ename='SCOTT'");
			int i = 7 / 0;
```

```
                    // 给 smith 的 sal 加上 100
                    sm.executeUpdate("update emp set sal=sal+100 where ename='SMITH'");
                    // 提交事务
                    ct.commit();
                    // 关闭打开的资源
                    sm.close();
                    ct.close();
                } catch (Exception e) {
                    // 如果发生异常，就回滚
                    try {
                        ct.rollback();
                    } catch (SQLException e1) {
                        e1.printStackTrace();
                    }
                    e.printStackTrace();
                }
            }
}
```

再运行一下，会出现异常，查看数据库，数据没变化。。

ν 只读事务

只读事务是指只允许执行查询的操作，而不允许执行任何其它 dml 操作的事务，使用只读事务可以确保用户只能取得某时间点的数据。假定机票代售点每天 18 点开始统计今天的销售情况，这时可以使用只读事务。在设置了只读事务后，尽管其它会话可能会提交新的事务，但是只读事务将不会取得最新数据的变化，从而可以保证取得特定时间点的数据信息。

ν 设置只读事务

```
set transaction read only;
```

9.oracle 的函数

sql 函数的使用

字符函数ν

介绍

字符函数是 oracle 中最常用的函数，我们来看看有哪些字符函数：

υ lower(char)：将字符串转化为小写的格式。

υ upper(char)：将字符串转化为大写的格式。

υ length(char)：返回字符串的长度。

υ substr(char,m,n)：取字符串的子串　　n 代表取 n 个的意思，不是代表取到第 n 个

υ replace(char1,search_string,replace_string)

υ instr(char1,char2,[,n[,m]])取子串在字符串的位置

问题：将所有员工的名字按小写的方式显示

```
SQL> select lower(ename) from emp;
```

问题：将所有员工的名字按大写的方式显示。

```
SQL> select upper(ename) from emp;
```

问题：显示正好为 5 个字符的员工的姓名。

```
SQL> select * from emp where length(ename)=5;
```

问题：显示所有员工姓名的前三个字符。

SQL> select substr(ename,1,3) from emp;

问题：以首字母大写,后面小写的方式显示所有员工的姓名。

SQL> select upper(substr(ename,1,1)) || lower(substr(ename,2,length(ename)-1)) from emp;

问题：以首字母小写,后面大写的方式显示所有员工的姓名。

SQL> select lower(substr(ename,1,1)) || upper(substr(ename,2,length(ename)-1)) from emp;

问题：显示所有员工的姓名，用“我是老虎”替换所有“A”

SQL> select replace(ename,'A', '我是老虎') from emp;

数学函数ν 介绍

数学函数的输入参数和返回值的数据类型都是数字类型的。数学函数包括 cos，cosh，exp，ln，log，sin，sinh，sqrt，tan，tanh，acos，asin，atan，round，我们讲最常用的：

υ round(n,[m]) 该函数用于执行四舍五入，如果省掉 m，则四舍五入到整数，如果 m 是正数，则四舍五入到小数点的 m 位后。如果 m 是负数，则四舍五入到小数点的 m 位前。

υ trunc(n,[m]) 该函数用于截取数字。如果省掉 m，就截去小数部分，如果 m 是正数就截取到小数点的 m 位后，如果 m 是负数，则截取到小数点的前 m 位。

υ mod(m,n)

υ floor(n) 返回小于或是等于 n 的最大整数

υ ceil(n) 返回大于或是等于 n 的最小整数

对数字的处理，在财务系统或银行系统中用的最多，不同的处理方法，对财务报表有不同的结果。

问题：显示在一个月为 30 天的情况下，所有员工的日薪金，忽略余数。

SQL> select trunc(sal/30), ename from emp;

or

SQL> select floor(sal/30), ename from emp;

在做 oracle 测试的时候，可以使用 dual 表

select mod(10,2) from dual;结果是 0

select mod(10,3) from dual;结果是 1

其它的数学函数，有兴趣的同学可以自己去看看：

abs(n)： 返回数字 n 的绝对值

select abs(-13) from dual;

acos(n)： 返回数字的反余弦值

asin(n)： 返回数字的反正弦值

atan(n)： 返回数字的反正切值

cos(n)：

exp(n)： 返回 e 的 n 次幂

log(m,n)： 返回对数值

power(m,n)： 返回 m 的 n 次幂

日期函数ν

介绍

日期函数用于处理 date 类型的数据。

默认情况下日期格式是 dd-mon-yy 即 12-7 月-78

(1)sysdate: 该函数返回系统时间

(2)add_months(d,n)

(3)last_day(d)：返回指定日期所在月份的最后一天

问题：查找已经入职 8 个月多的员工

SQL> select * from emp where sysdate>=add_months(hiredate,8);

问题：显示满 10 年服务年限的员工的姓名和受雇日期。

SQL> select ename, hiredate from emp where sysdate>=add_months(hiredate,12*10);

问题：对于每个员工，显示其加入公司的天数。

SQL> select floor(sysdate-hiredate) "入职天数",ename from emp;

or

SQL> select trunc(sysdate-hiredate) "入职天数",ename from emp;

问题：找出各月倒数第 3 天受雇的所有员工。

SQL> select hiredate,ename from emp where last_day(hiredate)-2=hiredate;

转换函数

ν 介绍 √

转换函数用于将数据类型从一种转为另外一种。在某些情况下，oracle server 允许值的数据类型和实际的不一样，这时 oracle server 会隐含的转化数据类型

比如：

create table t1(id int);

insert into t1 values('10');-->这样 oracle 会自动的将'10' -->10

create table t2 (id varchar2(10));

insert into t2 values(1); -->这样 oracle 就会自动的将 1 -->'1';

我们要说的是尽管 oracle 可以进行隐含的数据类型的转换，但是它并不适应所有的情况，为了提高程序的可靠性，我们应该使用转换函数进行转换。

ν to_char

你可以使用 select ename, hiredate, sal from emp where deptno = 10;显示信息，可是，在某些情况下，这个并不能满足你的需求。

问题：日期是否可以显示 时/分/秒

SQL> select ename, to_char(hiredate, 'yyyy-mm-dd hh24:mi:ss') from emp;

问题：薪水是否可以显示指定的货币符号

SQL>

yy：两位数字的年份 2004-->04

yyyy：四位数字的年份　2004 年

mm：两位数字的月份 8 月-->08

dd：两位数字的天 30 号-->30

hh24：　8 点-->20

hh12：8 点-->08

mi、ss-->显示分钟\秒

9：显示数字，并忽略前面 0

0：显示数字，如位数不足，则用 0 补齐

.：在指定位置显示小数点

,：在指定位置显示逗号

$：在数字前加美元

L：在数字前面加本地货币符号

C：在数字前面加国际货币符号

G：在指定位置显示组分隔符、

D：在指定位置显示小数点符号(.)

问题：显示薪水的时候，把本地货币单位加在前面

SQL> select ename, to_char(hiredate, 'yyyy-mm-dd hh24:mi:ss'), to_char(sal,'L99999.99') from emp;

问题：显示 1980 年入职的所有员工

SQL> select * from emp where to_char(hiredate, 'yyyy')=1980;

问题：显示所有 12 月份入职的员工

SQL> select * from emp where to_char(hiredate, 'mm')=12;

v to_date

函数 to_date 用于将字符串转换成 date 类型的数据。

问题：能否按照中国人习惯的方式年—月—日添加日期。

系统函数

v sys_context

1）terminal：当前会话客户所对应的终端的标示符

2）lanuage：语言

3）db_name： 当前数据库名称

4）nls_date_format： 当前会话客户所对应的日期格式

5）session_user： 当前会话客户所对应的数据库用户名

6）current_schema： 当前会话客户所对应的默认方案名

7）host： 返回数据库所在主机的名称

通过该函数，可以查询一些重要信息，比如你正在使用哪个数据库？

select sys_context('USERENV','db_name') from dual;

注意：USERENV 是固定的，不能改的，db_name 可以换成其它,比如 select sys_context('USERENV','lanuage') from dual;又比如 select sys_context('USERENV','current_schema') from dual;

10.数据库管理，表的逻辑备份与恢复

内容介绍

1.上节回顾

2.数据库管理员

3.数据库（表）的逻辑备份与恢复 √

4.数据字典和动态性能视图 √

5.管理表空间和数据文件 √

期望目标

1.了解 oracle 管理员的基本职责

2.掌握备份和恢复数据库/表的方法

3.理解表空间、数据字典、性能视图

数据库管理员

v 介绍

每个 oracle 数据库应该至少有一个数据库管理员（dba），对于一个小的数据库，一个 dba 就够了，但是对于一个大的数据库可能需要多个 dba 分担不同的管理职责。那么一个数据库管理员的主要工作是什么呢：

v 职责

1.安装和升级 oracle 数据库

2.建库，表空间，表，视图，索引…

3.制定并实施备份和恢复计划

4.数据库权限管理，调优，故障排除

5.对于高级 dba，要求能参与项目开发，会编写 sql 语句、存储过程、触发器、规则、约束、包

v 管理数据库的用户主要是 sys 和 system

（sys 好像是董事长，system 好像是总经理，董事长比总经理大，但是通常是总经理干事）

在前面我们已经提到这两个用户，区别主要是：

1.最重要的区别，存储的数据的重要性不同

sys：所有 oracle 的数据字典的基表和视图都存放在 sys 用户中，这些基表和视图对于 oracle 的运行是至关重要的，由数据库自己维护，任何用户都不能手动更改。sys 用户拥有 dba，sysdba，sysoper 角色或权限，是 oracle 权限最高的用户。

system：用于存放次一级的内部数据，如 oracle 的一些特性或工具的管理信息。system 用户拥有 dba，sysdba 角色或系统权限。

看图：

sysdba 可以建数据库，sysope 不能建数据库

2. 其次的区别，权限的不同。

sys 用户必须以 as sysdba 或 as sysoper 形式登录。不能以 normal 方式登录数据库

system 如果正常登录，它其实就是一个普通的 dba 用户，但是如果以 as sysdba 登录，其结果实际上它是作为 sys 用户登录的，从登录信息里面我们可以看出来。

sysdba 和 sysoper 权限区别图，看图：

sysdba>sysoper>dba

可以看到：只要是 sysoper 拥有的权限，sysdba 都有；蓝色是它们区别的地方。（它们的最大区别是：sysdba 可以创建数据库，sysoper 不可以创建数据库）

ν dba 权限的用户

dba 用户是指具有 dba 角色的数据库用户。特权用户可以执行启动实例，关闭实例等特殊操作，而 dba 用户只有在启动数据库后才能执行各种管理工作。

（相当于说 dba 连 startup 和 shutdown 这两个权限都没有）

两个主要的用户，三个重要权限，他们的区别和联系，大家要弄清楚

管理初始化参数

ν 管理初始化参数（调优的一个重要知识点，凭什么可以对数据库进行调优呢？是因为它可以对数据库的一些参数进行修改修正）

初始化参数用于设置实例或是数据库的特征。oracle9i 提供了 200 多个初始化参数，并且每个初始化参数都有默认值。

λ 显示初始化参数

(1) show parameter 命令

λ 如何修改参数

需要说明的如果你希望修改这些初始化的参数，可以到文件 D:\oracle\admin\myoral\pfile\init.ora 文件中去修改比如要修改实例的名字

数据库（表）的逻辑备份与恢复

介绍

ν 介绍

逻辑备份是指使用工具 export 将数据对象的结构和数据导出到文件的过程，逻辑恢复是指当数据库对象被误操作而损坏后使用工具 import 利用备份的文件把数据对象导入到数据库的过程。

物理备份即可在数据库 open 的状态下进行也可在关闭数据库后进行，但是逻辑备份和恢复只能在 open 的状态下进行。

看图：

ν 导出

导出具体的分为：导出表，导出方案，导出数据库三种方式。

导出使用 exp 命令来完成的，该命令常用的选项有：

userid： 用于指定执行导出操作的用户名，口令，连接字符串

tables： 用于指定执行导出操作的表

owner： 用于指定执行导出操作的方案

full=y： 用于指定执行导出操作的数据库

inctype： 用于指定执行导出操作的增量类型

rows： 用于指定执行导出操作是否要导出表中的数据

file： 用于指定导出文件名

ν 导出表

1.导出自己的表

exp userid=scott/tiger@myoral tables=(emp,dept) file=d:\e1.dmp

2.导出其它方案的表

如果用户要导出其它方案的表，则需要 dba 的权限或是 exp_full_database 的权限，比如 system 就可以导出 scott 的表

E:\oracle\ora92\bin>exp userid=system/manager@myoral tables=(scott.emp) file=d:\e2.emp

特别说明：在导入和导出的时候，要到 oracle 目录的 bin 目录下。

3. 导出表的结构

exp userid=scott/tiger@accp tables=(emp) file=d:\e3.dmp rows=n

4. 使用直接导出方式

exp userid=scott/tiger@accp tables=(emp) file=d:\e4.dmp direct=y

这种方式比默认的常规方式速度要快，当数据量大时，可以考虑使用这样的方法。

这时需要数据库的字符集要与客户端字符集完全一致，否则会报错...

ν 导出方案

导出方案是指使用 export 工具导出一个方案或是多个方案中的所有对象（表，索引，约束...）和数据。并存放到文件中。

1. 导出自己的方案

exp userid=scott/tiger@myorcl owner=scott file=d:\scott.dmp

2. 导出其它方案

如果用户要导出其它方案，则需要 dba 的权限或是 exp_full_database 的权限，比如 system 用户就可以导出任何方案

exp userid=system/manager@myorcl owner=(system,scott) file=d:\system.dmp

ν 导出数据库

导出数据库是指利用 export 导出所有数据库中的对象及数据，要求该用户具有 dba 的权限或者是 exp_full_database 权限

增量备份（好处是第一次备份后，第二次备份就快很多了）

exp userid=system/manager@myorcl full=y inctype=complete file=d:\all.dmp

导入

ν 介绍

导入就是使用工具 import 将文件中的对象和数据导入到数据库中，但是导入要使用的文件必须是 export 所导出的文件。与导出相似，导入也分为导入表，导入方案，导入数据库三种方式。

imp 常用的选项有

userid： 用于指定执行导入操作的用户名，口令，连接字符串

tables： 用于指定执行导入操作的表

formuser： 用于指定源用户

touser： 用于指定目标用户

file： 用于指定导入文件名

full=y： 用于指定执行导入整个文件

inctype： 用于指定执行导入操作的增量类型

rows： 指定是否要导入表行（数据）

ignore： 如果表存在，则只导入数据

ν 导入表

1. 导入自己的表

imp userid=scott/tiger@myorcl tables=(emp) file=d:\xx.dmp

2. 导入表到其它用户

要求该用户具有 dba 的权限，或是 imp_full_database

imp userid=system/tiger@myorcl tables=(emp) file=d:\xx.dmp touser=scott

3. 导入表的结构

只导入表的结构而不导入数据

imp userid=scott/tiger@myorcl tables=(emp) file=d:\xx.dmp rows=n

4. 导入数据

如果对象（如比表）已经存在可以只导入表的数据

imp userid=scott/tiger@myorcl tables=(emp) file=d:\xx.dmp ignore=y

ν 导入方案

导入方案是指使用 import 工具将文件中的对象和数据导入到一个或是多个方案中。如果要导入其它方案，要求该用户具有 dba 的权限，或者 imp_full_database

1. 导入自身的方案

imp userid=scott/tiger file=d:\xxx.dmp

2. 导入其它方案

要求该用户具有 dba 的权限

imp userid=system/manager file=d:\xxx.dmp fromuser=system touser=scott

ν 导入数据库

在默认情况下，当导入数据库时，会导入所有对象结构和数据，案例如下：

imp userid=system/manager full=y file=d:\xxx.dmp

11.数据字典和动态性能视图

介绍：数据字典是什么

数据字典是 oracle 数据库中最重要的组成部分，它提供了数据库的一些系统信息。

动态性能视图记载了例程启动后的相关信息。

ν **数据字典**

数据字典记录了数据库的系统信息，它是只读表和视图的集合，数据字典的所有者为 sys 用户。

用户只能在数据字典上执行查询操作（select 语句），而其维护和修改是由系统自动完成的。

这里我们谈谈数据字典的组成：数据字典包括数据字典基表和数据字典视图，其中基表存储数据库的基本信息，普通用户不能直接访问数据字典的基表。数据字典视图是基于数据字典基表所建立的视图，普通用户可以通过查询数据字典视图取得系统信息。数据字典视图主要包括 user_xxx，all_xxx，dba_xxx 三种类型。

ν user_tables;

用于显示当前用户所拥有的所有表，它只返回用户所对应方案的所有表

比如：select table_name from user_tables;

ν all_tables;

用于显示当前用户可以访问的所有表，它不仅会返回当前用户方案的所有表，还会返回当前用户可以访问的其它方案的表：

比如：select table_name from all_tables;

ν dba_tables;

它会显示所有方案拥有的数据库表。但是查询这种数据库字典视图，要求用户必须是 dba 角色或是有 select any table 系统权限。

例如：当用 system 用户查询数据字典视图 dba_tables 时，会返回 system，sys，scott...方案所对应的数据库表。

ν 用户名，权限，角色

在建立用户时，oracle 会把用户的信息存放到数据字典中，当给用户授予权限或是角色时，oracle 会将权限和角色的信息存放到数据字典。

通过查询 dba_users 可以显示所有数据库用户的详细信息；

通过查询数据字典视图 dba_sys_privs，可以显示用户所具有的系统权限；

通过查询数据字典视图 dba_tab_privs，可以显示用户具有的对象权限；

通过查询数据字典 dba_col_privs 可以显示用户具有的列权限；

通过查询数据库字典视图 dba_role_privs 可以显示用户所具有的角色。

这里给大家再讲讲角色和权限的关系。

例如：要查看 scott 具有的角色，可查询 dba_role_privs；

SQL> select * from dba_role_privs where grantee='SCOTT';

//查询 orale 中所有的系统权限，一般是 dba

select * from system_privilege_map order by name;

//查询 oracle 中所有对象权限，一般是 dba

select distinct privilege from dba_tab_privs;

//查询 oracle 中所有的角色，一般是 dba

select * from dba_roles;

//查询数据库的表空间

select tablespace_name from dba_tablespaces;

问题 1：如何查询一个角色包括的权限？

a. 一个角色包含的系统权限

select * from dba_sys_privs where grantee='角色名'

另外也可以这样查看：

select * from role_sys_privs where role='角色名'

b. 一个角色包含的对象权限

select * from dba_tab_privs where grantee='角色名'

问题 2：oracle 究竟有多少种角色？

SQL> select * from dba_roles;

问题 3：如何查看某个用户，具有什么样的角色？

select * from dba_role_privs where grantee='用户名'

v 显示当前用户可以访问的所有数据字典视图。

select * from dict where comments like '%grant%';

v 显示当前数据库的全称

select * from global_name;

v 其它说明

数据字典记录有 oracle 数据库的所有系统信息。通过查询数据字典可以取得以下系统信息：比如

1. 对象定义情况

2. 对象占用空间大小

3. 列信息

4. 约束信息

...

但是因为这些个信息，可以通过 pl/sql developer 工具查询得到，所以这里我就飘过。

v 动态性能视图

动态性能视图用于记录当前例程的活动信息，当启动 oracle server 时，系统会建立动态性能视图；当停止 oracle server 时，系统会删除动态性能视图。oracle 的所有动态性能视图都是以 v_$开始的，并且 oracle 为每个动态性能视图都提供了相应的同义词，并且其同义词是以 V$开始的，例如 v_$datafile 的同义词为 v$datafile;动态性能视图的所有者为 sys，一般情况下，由 dba 或是特权用户来查询动态性能视图。

因为这个在实际中用的较少，所以飞过。

12. 数据库管理 -- 管理表空间和数据文件

v 介绍

表空间是数据库的逻辑组成部分。从物理上讲，数据库数据存放在数据文件中；从逻辑上讲，数据库则是存放在表空间中，表空间由一个或多个数据文件组成。

数据库的逻辑结构

ν 介绍

oracle 中逻辑结构包括表空间、段、区和块。

说明一下数据库由表空间构成，而表空间又是由段构成，而段又是由区构成，而区又是由 oracle 块构成的这样的一种结构，可以提高数据库的效率。

为了让大家明白，我们画图说明逻辑关系：看图：

表空间

ν 介绍

表空间用于从逻辑上组织数据库的数据。数据库逻辑上是由一个或是多个表空间组成的。通过表空间可以达到以下作用：

1. 控制数据库占用的磁盘空间

2. dba 可以将不同数据类型部署到不同的位置，这样有利于提高 i/o 性能，同时利于备份和恢复等管理操作。

ν 建立表空间

建立表空间是使用 crate tablespace 命令完成的，需要注意的是，一般情况下，建立表空间是特权用户或是 dba 来执行的，如果用其它用户来创建表空间，则用户必须要具有 create tablespace 的系统权限。

ν 建立数据表空间

在建立数据库后，为便于管理表，最好建立自己的表空间

create tablespace data01 datafile 'd:\test\dada01.dbf' size 20m uniform size 128k;

说明：执行完上述命令后，会建立名称为 data01 的表空间，并为该表空间建立名称为 data01.dbf 的数据文件，区的大小为 128k

ν 使用数据表空间

create table mypart(deptno number(4), dname varchar2(14), loc varchar2(13)) tablespace data01;

ν 改变表空间的状态

当建立表空间时，表空间处于联机的（online）状态，此时该表空间是可以访问的，并且该表空间是可以读写的，即可以查询该表空间的数据，而且还可以在表空间执行各种语句。但是在进行系统维护或是数据维护时，可能需要改变表空间的状态。一般情况下，由特权用户或是 dba 来操作。

1. 使表空间脱机

alter tablespace 表空间名 offline;

2. 使表空间联机

alter tablespace 表空间名 online;

3. 只读表空间

当建立表空间时，表空间可以读写，如果不希望在该表空间上执行 update，delete，insert 操作，那么可以将表空间修改为只读

alter tablespace 表空间名 read only;

(修改为可写是 alter tablespace 表空间名 read write;)

ν 改变表空间的状态

我们给大家举一个实例，说明只读特性：

1. 知道表空间名，显示该表空间包括的所有表

select * from all_tables where tablespace_name='表空间名';

2. 知道表名，查看该表属于那个表空间

select tablespace_name, table_name from user_tables where table_name='emp';

通过 2.我们可以知道 scott.emp 是在 system 这个表空间上,现在我们可以将 system 改为只读的但是我们不会成功,因为 system 是系统表空间，如果是普通表空间，那么我们就可以将其设为只读的，给大家做一个演示，可以加强理解。

3.

4. 使表空间可读写

alter tablespace 表空间名 read write;

ν 删除表空间

一般情况下，由特权用户或是 dba 来操作，如果是其它用户操作，那么要求用户具有 drop tablespace 系统权限。

drop tablespace ‘表空间’ including contents and datafiles;

说明：including contents 表示删除表空间时，删除该空间的所有数据库对象，而 datafiles 表示将数据库文件也删除。

ν 扩展表空间

表空间是由数据文件组成的，表空间的大小实际上就是数据文件相加后的大小。那么我们可以想象，假定表 employee 存放到 data01 表空间上，初始大小就是 2M，当数据满 2M 空间后，如果在向 employee 表插入数据，这样就会显示空间不足的错误。

案例说明：

1. 建立一个表空间 sp01
2. 在该表空间上建立一个普通表 mydment 其结构和 dept 一样
3. 向该表中加入数据 insert into mydment select * from dept;
4. 当一定时候就会出现无法扩展的问题，怎么办?
5. 就扩展该表空间，为其增加更多的存储空间。有三种方法：

1. 增加数据文件

SQL> alter tablespace sp01 add datafile ‘d:\test\sp01.dbf’ size 20m;

2. 增加数据文件的大小

SQL> alter tablespace 表空间名 ‘d:\test\sp01.dbf’ resize 20m;

这里需要注意的是数据文件的大小不要超过 500m。

3. 设置文件的自动增长。

SQL> alter tablespace 表空间名 ‘d:\test\sp01.dbf’ autoextend on next 10m maxsize 500m;

ν 移动数据文件

有时，如果你的数据文件所在的磁盘损坏时，该数据文件将不能再使用，为了能够重新使用，需要将这些文件的副本移动到其它的磁盘，然后恢复。

下面以移动数据文件 sp01.dbf 为例来说明：

1. 确定数据文件所在的表空间

select tablespace_name from dba_data_files where file_name=’d:\test\sp01.dbf’;

2. 使表空间脱机

确保数据文件的一致性，将表空间转变为 offline 的状态。

alter tablespace sp01(表空间名) offline;

3. 使用命令移动数据文件到指定的目标位置

host move d:\test\sp01.dbf c:\test\sp01.dbf

4. 执行 alter tablespace 命令

在物理上移动了数据后，还必须执行 alter tablespace 命令对数据库文件进行逻辑修改：

alter tablespace sp01 rename datafile ‘d:\test\sp01.dbf’ to ‘c:\test\sp01.dbf’;

5. 使得表空间联机

在移动了数据文件后，为了使用户可以访问该表空间，必须将其转变为 online 状态。

alter tablespace sp01(表空间名) online;

ν 显示表空间信息

查询数据字典视图 dba_tablespaces，显示表空间的信息：

select tablespace_name from dba_tablespaces;

ν 显示表空间所包含的数据文件

查询数据字典视图 dba_data_files,可显示表空间所包含的数据文件，如下：

select file_name, bytes from dba_data_files where tablespce_name=’表空间’;

ν 表空间小结

1. 了解表空间和数据文件的作用

2. 掌握常用表空间，undo 表空间和临时表空间的建立方法
3. 了解表空间的各个状态(online, offline, read write, read only)的作用，及如何改变表空间的状态的方法。
4. 了解移动数据文件的原因，及使用 alter tablespace 和 alter datatable 命令移动数据文件的方法。

ν 其它表空间

除了最常用的数据表空间外，还有其它类型表空间：

1. 索引表空间
2. undo 表空间
3. 临时表空间
4. 非标准块的表空间

这几种表空间，大家伙可以自己参考书籍研究，这里我就不讲。

ν 其它说明

关于表空间的组成部分 段/区/块，我们在后面给大家讲解。

13.约束

玩转 oracle 实战教程(第五天)

期望目标

1.掌握维护 oracle 数据完整性的技巧

2.理解索引概念，会建立索引

3.管理 oracle 的权限和角色

维护数据的完整性

ν 介绍

数据的完整性用于确保数据库数据遵从一定的商业和逻辑规则，在 oracle 中，数据完整性可以使用约束、触发器、应用程序（过程、函数）三种方法来实现，在这三种方法中，因为约束易于维护，并且具有最好的性能，所以作为维护数据完整性的首选。

约束

ν 约束

约束用于确保数据库数据满足特定的商业规则。在 oracle 中，约束包括：not null、 unique， primary key， foreign key，和 check 五种。

使用

ν not null（非空）

如果在列上定义了 not null，那么当插入数据时，必须为列提供数据。

ν unique（唯一）

当定义了唯一约束后，该列值是不能重复的，但是可以为 null。

ν primary key（主键）

用于唯一的标示表行的数据，当定义主键约束后，该列不但不能重复而且不能为 null。

需要说明的是：一张表最多只能有一个主键，但是可以有多个 unqiue 约束。

ν foreign key（外键）

用于定义主表和从表之间的关系。外键约束要定义在从表上，主表则必须具有主键约束或是 unique 约束，当定义外键约束后，要求外键列数据必须在主表的主键列存在或是为 null。

ν check

用于强制行数据必须满足的条件，假定在 sal 列上定义了 check 约束，并要求 sal 列值在 1000-2000 之间如果不在 1000-2000 之间就会提示出错。

ν 商店售货系统表设计案例

现有一个商店的数据库，记录客户及其购物情况，由下面三个表组成：商品 goods（商品号 goodsId，商品名 goodsName，单价 unitprice，商品类别 category，供应商 provider）；

客户 customer（客户号 customerId，姓名 name，住在 address，电邮 email，性别 sex，身份证 cardId）；

购买 purchase（客户号 customerId，商品号 goodsId，购买数量 nums）；

请用 SQL 语言完成下列功能：

1. 建表，在定义中要求声明：

(1). 每个表的主外键；

(2). 客户的姓名不能为空值；

(3). 单价必须大于 0，购买数量必须在 1 到 30 之间；

(4). 电邮不能够重复；

(5). 客户的性别必须是 男 或者 女，默认是男；

```
SQL> create table goods(goodsId char(8) primary key, --主键
                        goodsName varchar2(30),
                        unitprice number(10,2) check(unitprice>0),
                        category varchar2(8),
                        provider varchar2(30)
         );
SQL> create table customer( customerId char(8) primary key, --主键
                        name varchar2(50) not null,   --不为空
                        address varchar2(50),
                        email varchar2(50) unique,
                        sex char(2) default '男' check(sex in ('男','女')),  -- 一个 char 能存半个汉字，两位 char
能存一个汉字
                        cardId char(18)
         );
SQL> create table purchase( customerId char(8) references customer(customerId),
                        goodsId char(8) references goods(goodsId),
                        nums number(10) check (nums between 1 and 30)
         );
```

表是默认建在 SYSTEM 表空间的

维护

ν 商店售货系统表设计案例（2）

如果在建表时忘记建立必要的约束，则可以在建表后使用 alter table 命令为表增加约束。但是要注意：增加 not null 约束时，需要使用 modify 选项，而增加其它四种约束使用 add 选项。

1. 增加商品名也不能为空

```
SQL> alter table goods modify goodsName not null;
```

2. 增加身份证也不能重复

```
SQL> alter table customer add constraint xxxxxx unique(cardId);
```

3. 增加客户的住址只能是’海淀’,’朝阳’,’东城’,’西城’,’通州’,’崇文’,’昌平’；

```
SQL> alter table customer add constraint yyyyyy check (address in (’海淀’,’朝阳’,’东城’,’西城’,’通州’,’
崇文’,’昌平’));
```

ν 删除约束

当不再需要某个约束时，可以删除。

alter table 表名 drop constraint 约束名称；

特别说明一下：

在删除主键约束的时候，可能有错误，比如：

alter table 表名 drop primary key；

这是因为如果在两张表存在主从关系，那么在删除主表的主键约束时，必须带上 cascade 选项 如像：

alter table 表名 drop primary key cascade;

ν 显示约束信息

1.显示约束信息

通过查询数据字典视图 user_constraints，可以显示当前用户所有的约束的信息。

select constraint_name, constraint_type, status, validated from user_constraints where table_name = '表名';

2.显示约束列

通过查询数据字典视图 user_cons_columns，可以显示约束所对应的表列信息。

select column_name, position from user_cons_columns where constraint_name = '约束名';

3.当然也有更容易的方法，直接用 pl/sql developer 查看即可。简单演示一下下...

表级定义 列级定义

ν 列级定义

列级定义是在定义列的同时定义约束。

如果在 department 表定义主键约束

create table department4(dept_id number(12) constraint pk_department primary key,

name varchar2(12), loc varchar2(12));

ν 表级定义

表级定义是指在定义了所有列后，再定义约束。这里需要注意：

not null 约束只能在列级上定义。

以在建立 employee2 表时定义主键约束和外键约束为例：

create table employee2(emp_id number(4), name varchar2(15),

dept_id number(2), constraint pk_employee primary key (emp_id),

constraint fk_department foreign key (dept_id) references department4(dept_id));

14.Oracle 索引、权限

管理索引-原理介绍

ν 介绍

索引是用于加速数据存取的数据对象。合理的使用索引可以大大降低 i/o 次数，从而提高数据访问性能。索引有很多种我们主要介绍常用的几种：

为什么添加了索引后，会加快查询速度呢？

创建索引

ν 单列索引

单列索引是基于单个列所建立的索引，比如：

create index 索引名 on 表名(列名);

ν 复合索引

复合索引是基于两列或是多列的索引。在同一张表上可以有多个索引，但是要求列的组合必须不同，比如：

create index emp_idx1 on emp (ename, job);

create index emp_idx1 on emp (job, ename);

使用原则

ν 使用原则

1. 在大表上建立索引才有意义

2. 在 where 子句或是连接条件上经常引用的列上建立索引

3. 索引的层次不要超过 4 层

这里能不能给学生演示这个效果呢？

如何构建一个大表呢？

索引的缺点

ν 索引缺点分析

索引有一些先天不足：

1. 建立索引，系统要占用大约为表 1.2 倍的硬盘和内存空间来保存索引。

2. 更新数据的时候，系统必须要有额外的时间来同时对索引进行更新，以维持数据和索引的一致性。

实践表明，不恰当的索引不但于事无补，反而会降低系统性能。因为大量的索引在进行插入、修改和删除操作时比没有索引花费更多的系统时间。

比如在如下字段建立索引应该是不恰当的：

1. 很少或从不引用的字段；

2. 逻辑型的字段，如男或女（是或否）等。

综上所述，提高查询效率是以消耗一定的系统资源为代价的，索引不能盲目的建立，这是考验一个 DBA 是否优秀的很重要的指标。

其它索引

ν 介绍

按照数据存储方式，可以分为 B*树、反向索引、位图索引；

按照索引列的个数分类，可以分为单列索引、复合索引；

按照索引列值的唯一性，可以分为唯一索引和非唯一索引。

此外还有函数索引，全局索引，分区索引...

对于索引我还要说：

在不同的情况，我们会在不同的列上建立索引，甚至建立不同种类的索引，请记住，技术是死的，人是活的。比如：

B*树索引建立在重复值很少的列上，而位图索引则建立在重复值很多、不同值相对固定的列上。

显示索引信息

ν 显示表的所有索引

在同一张表上可以有多个索引，通过查询数据字典视图 dba_indexs 和 user_indexs，可以显示索引信息。其中 dba_indexs 用于显示数据库所有的索引信息，而 user_indexs 用于显示当前用户的索引信息：

```
select index_name, index_type from user_indexes where table_name = '表名';
```

ν 显示索引列

通过查询数据字典视图 user_ind_columns,可以显示索引对应的列的信息

```
select table_name, column_name from user_ind_columns where index_name = 'IND_ENAME';
```

ν 你也可以通过 pl/sql developer 工具查看索引信息

管理权限和角色

介绍

ν 介绍

这一部分我们主要看看 oracle 中如何管理权限和角色，权限和角色的区别在那里。

当刚刚建立用户时，用户没有任何权限，也不能执行任何操作。如果要执行某种特定的数据库操作，则必须为其授予系统的权限；如果用户要访问其它方案的对象，则必须为其授予对象的权限。为了简化权限的管理，可以使用角色。这里我们会详细的介绍。看图：

权限

ν 权限

权限是指执行特定类型 sql 命令或是访问其它方案对象的权利，包括系统权限和对象权限两种。

系统权限

ν 系统权限介绍

系统权限是指执行特定类型 sql 命令的权利。它用于控制用户可以执行的一个或是一组数据库操作。比如当用户具有 create table 权限时，可以在其方案中建表，当用户具有 create any table 权限时，可以在任何方案中建表。oracle 提供了 100 多种系统权限。

常用的有：

create session 连接数据库 create table 建表

create view 建视图 create public synonym 建同义词

create procedure 建过程、函数、包 create trigger 建触发器

create cluster 建簇

ν 显示系统权限

oracle 提供了 100 多种系统权限，而且 oracle 的版本越高，提供的系统权限就越多，我们可以查询数据字典视图 system_privilege_map，可以显示所有系统权限。

```
select * from system_privilege_map order by name;
```

ν 授予系统权限

一般情况，授予系统权限是由 dba 完成的，如果用其他用户来授予系统权限，则要求该用户必须具有 grant any privilege 的系统权限。在授予系统权限时，可以带有 with admin option 选项，这样，被授予权限的用户或是角色还可以将该系统权限授予其它的用户或是角色。为了让大家快速理解，我们举例说明：

1.创建两个用户 ken，tom。初始阶段他们没有任何权限，如果登录就会给出错误的信息。

```
create user ken identfied by ken;
```

2 给用户 ken 授权

```
1). grant create session, create table to ken with admin option;
2). grant create view to ken;
```

3 给用户 tom 授权

我们可以通过 ken 给 tom 授权，因为 with admin option 是加上的。当然也可以通过 dba 给 tom 授权，我们就用 ken 给 tom 授权：

```
1. grant create session, create table to tom;
2. grant create view to ken;   --ok 吗？不 ok
```

ν 回收系统权限

一般情况下，回收系统权限是 dba 来完成的，如果其它的用户来回收系统权限，要求该用户必须具有相应系统权限及转授系统权限的选项（with admin option）。回收系统权限使用 revoke 来完成。

当回收了系统权限后，用户就不能执行相应的操作了，但是请注意，系统权限级联收回的问题?[不是级联回收！]

```
system --------->ken ---------->tom
(create session)(create session)( create session)
```

用 system 执行如下操作：

```
revoke create session from ken;   --请思考 tom 还能登录吗？
```

答案：能，可以登录

对象权限

ν 对象权限介绍

指访问其它方案对象的权利，用户可以直接访问自己方案的对象，但是如果要访问别的方案的对象，则必须具有对象的权限。比如 smith 用户要访问 scott.emp 表（scott：方案，emp：表）

常用的有：

alter 修改 delete 删除 select 查询 insert 添加

update 修改 index 索引 references 引用 execute 执行

ν 显示对象权限

通过数据字段视图可以显示用户或是角色所具有的对象权限。视图为 dba_tab_privs

```
SQL> conn system/manager;
SQL> select distinct privilege from dba_tab_privs;
SQL> select grantor, owner, table_name, privilege from dba_tab_privs where grantee = 'BLAKE';
```

1.授予对象权限

在 oracle9i 前，授予对象权限是由对象的所有者来完成的，如果用其它的用户来操作，则需要用户具有相应的（with grant option）权限，从 oracle9i 开始，dba 用户（sys，system）可以将任何对象上的对象权限授予其它用户。授予对象权限是用 grant 命令来完成的。

对象权限可以授予用户，角色，和 public。在授予权限时，如果带有 with grant option 选项，则可以将该权限转授给其它用户。但是要注意 with grant option 选项不能被授予角色。

1.monkey 用户要操作 scott.emp 表，则必须授予相应的对象权限

1). 希望 monkey 可以查询 scott.emp 表的数据，怎样操作？

```
grant select on emp to monkey;
```

2). 希望 monkey 可以修改 scott.emp 的表数据，怎样操作？

```
grant update on emp to monkey;
```

3). 希望 monkey 可以删除 scott.emp 的表数据，怎样操作？

```
grant delete on emp to monkey;
```

4). 有没有更加简单的方法，一次把所有权限赋给 monkey？

```
grant all on emp to monkey;
```

2.能否对 monkey 访问权限更加精细控制。（授予列权限）

1). 希望 monkey 只可以修改 scott.emp 的表的 sal 字段，怎样操作？

```
grant update on emp(sal) to monkey
```

2).希望 monkey 只可以查询 scott.emp 的表的 ename，sal 数据，怎样操作？

```
grant select on emp(ename,sal) to monkey
```

...

3.授予 alter 权限

如果 black 用户要修改 scott.emp 表的结构，则必须授予 alter 对象权限

```
SQL> conn scott/tiger
SQL> grant alter on emp to blake;
```

当然也可以用 system，sys 来完成这件事。

4.授予 execute 权限

如果用户想要执行其它方案的包/过程/函数，则须有 execute 权限。

比如为了让 ken 可以执行包 dbms_transaction，可以授予 execute 权限。

```
SQL> conn system/manager
SQL> grant execute on dbms_transaction to ken;
```

5.授予 index 权限

如果想在别的方案的表上建立索引，则必须具有 index 对象权限。

如果为了让 black 可以在 scott.emp 表上建立索引，就给其 index 的对象权限

```
SQL> conn scott/tiger
SQL> grant index on scott.emp to blake;
```

6.使用 with grant option 选项

该选项用于转授对象权限。但是该选项只能被授予用户，而不能授予角色

```
SQL> conn scott/tiger;
SQL> grant select on emp to blake with grant option;
SQL> conn black/shunping
SQL> grant select on scott.emp to jones;
```

ν 回收对象权限

在 oracle9i 中，收回对象的权限可以由对象的所有者来完成，也可以用 dba 用户（sys，system）来完成。

这里要说明的是：收回对象权限后，用户就不能执行相应的 sql 命令，但是要注意的是对象的权限是否会被级联收回？【级

联回收】

如：scott-------------->blake-------------->jones

select on emp　　select on emp　　select on emp

SQL> conn scott/tiger@accp

SQL> revoke select on emp from blake

请大家思考，jones 能否查询 scott.emp 表数据。

答案：查不了了（和系统权限不一样，刚好相反）

15.角色

ν 介绍

角色就是相关权限的命令集合，使用角色的主要目的就是为了简化权限的管理，假定有用户 a，b，c 为了让他们都拥有权限

1. 连接数据库

2. 在 scott.emp 表上 select，insert，update。

如果采用直接授权操作，则需要进行 12 次授权。

因为要进行 12 次授权操作，所以比较麻烦喔！怎么办？

如果我们采用角色就可以简化：

首先将 creat session，select on scott.emp，insert on scott.emp， update on scott.emp 授予角色，然后将该角色授予 a，b，c 用户，这样就可以三次授权搞定。

角色分为预定义和自定义角色两类：

ν 预定义角色

预定义角色是指 oracle 所提供的角色，每种角色都用于执行一些特定的管理任务，下面我们介绍常用的预定义角色 connect，resource，dba

1.connect 角色

connect 角色具有一般应用开发人员需要的大部分权限,当建立了一个用户后,多数情况下,只要给用户授予 connect 和 resource 角色就够了，那么 connect 角色具有哪些系统权限呢？

alter session

create cluster

create　database link

create session

create table

create view

create sequence

2.resource 角色

resource 角色具有应用开发人员所需要的其它权限，比如建立存储过程，触发器等。这里需要注意的是 resource 角色隐含了 unlimited tablespace 系统权限。

resource 角色包含以下系统权限：

create cluster

create indextype

create table

create sequence

create type

create procedure

create trigger

3.dba 角色

dba 角色具有所有的系统权限，及 with admin option 选项，默认的 dba 用户为 sys 和 system，它们可以将任何系统权限授予其他用户。但是要注意的是 dba 角色不具备 sysdba 和 sysoper 的特权（启动和关闭数据库）。

ν 自定义角色

顾名思义就是自己定义的角色，根据自己的需要来定义。一般是 dba 来建立，如果用别的用户来建立，则需要具有 create role 的系统权限。在建立角色时可以指定验证方式（不验证，数据库验证等）。

1.建立角色（不验证）

如果角色是公用的角色，可以采用不验证的方式建立角色。

```
create role 角色名 not identified;
```

2.建立角色（数据库验证）

采用这样的方式时，角色名、口令存放在数据库中。当激活该角色时，必须提供口令。在建立这种角色时，需要为其提供口令。

```
create role 角色名 identified by 密码;
```

角色授权

当建立角色时，角色没有任何权限，为了使得角色完成特定任务，必须为其授予相应的系统权限和对象权限。

1.给角色授权

给角色授予权限和给用户授权没有太多区别，但是要注意，系统权限的 unlimited tablespace 和对象权限的 with grant option 选项是不能授予角色的。

```
SQL> conn system/manager;
SQL> grant create session to 角色名 with admin option
SQL> conn scott/tiger@myoral;
SQL> grant select on scott.emp to 角色名;
SQL> grant insert, update, delete on scott.emp to 角色名;
```

通过上面的步骤，就给角色授权了。

2.分配角色给某个用户

一般分配角色是由 dba 来完成的，如果要以其它用户身份分配角色，则要求用户必须具有 grant any role 的系统权限。

```
SQL> conn system/manager;
SQL> grant 角色名 to blake with admin option;
```

因为我给了 with admin option 选项，所以，blake 可以把 system 分配给它的角色分配给别的用户。

ν 删除角色

使用 drop role，一般是 dba 来执行，如果其它用户则要求该用户具有 drop any role 系统权限。

```
SQL> conn system/manager;
SQL> drop role 角色名;
```

问题：如果角色被删除，那么被授予角色的用户是否还具有之前角色里的权限？

答案：不具有了

ν 显示角色信息

1.显示所有角色

```
SQL> select * from dba_roles;
```

2.显示角色具有的系统权限

```
SQL> select privilege, admin_option from role_sys_privs where role='角色名';
```

3.显示角色具有的对象权限

通过查询数据字典视图 dba_tab_privs 可以查看角色具有的对象权限或是列的权限。

4.显示用户具有的角色，及默认角色

当以用户的身份连接到数据库时，oracle 会自动的激活默认的角色，通过查询数据字典视图 dba_role_privs 可以显示某个用户具有的所有角色及当前默认的角色

SQL> select granted_role, default_role from dba_role_privs where grantee = ‘用户名’;

ν 精细访问控制

精细访问控制是指用户可以使用函数，策略实现更加细微的安全访问控制。如果使用精细访问控制，则当在客户端发出 sql 语句（select，insert，update，delete）时，oracle 会自动在 sql 语句后追加谓词（where 子句），并执行新的 sql 语句，通过这样的控制，可以使得不同的数据库用户在访问相同表时，返回不同的数据信息，如：

用户 scott　　blake　　jones

策略　　emp_access

　　数据库表 emp

如上图所示，通过策略 emp_access，用户 scott，black，jones 在执行相同的 sql 语句时，可以返回不同的结果。例如：当执行 select ename from emp; 时，根据实际情况可以返回不同的结果。

16.PL/SQL 块的结构和实例

韩顺平.玩转 oralce 第 24 讲.plsql 编程(1)

玩转 orcle 实战教程(第六天)

内容介绍

1.上节回顾

2.pl/sql 的介绍 √

3.pl/sql 的基础 √

期望目标

1.理解 oracle 的 pl/sql 概念

2.掌握 pl/sql 编程技术（包括编写过程、函数、触发器...）

pl/sql 的介绍

pl/sql 是什么

pl/sql(procedural language/sql)是 oracle 在标准的 sql 语言上的扩展。pl/sql 不仅允许嵌入 sql 语言，还可以定义变量和常量，允许使用条件语句和循环语句，允许使用例外处理各种错误，这样使得它的功能变得更加强大。

为什么学 pl/sql

ν 学习必要性

1.提高应用程序的运行性能

2.模块化的设计思想【分页的过程，订单的过程，转账的过程。。】

3.减少网络传输量

4.提高安全性（sql 会包括表名，有时还可能有密码，传输的时候会泄露。PL/SQL 就不会）

为什么 PL/SQL 会快呢？看图：

不好的地方：

移植性不好（换数据库就用不了），

用什么编写 pl/sql

ν sqlplus 开发工具

sqlplus 是 oracle 公司提供的一个工具，这个因为我们在以前介绍过的：

举一个简单的案例：

编写一个存储过程，该过程可以向某表中添加记录。

1.创建一个简单的表

```
create table mytest(name varchar2(30),passwd varchar2(30));
```

2.创建过程

```
  create or replace procedure sp_pro1 is
```

```
begin--执行部分
insert into mytest values('韩顺平','m1234');
end;
/
```

replace:表示如果有 sp_pro1，就替换

如何查看错误信息：show error;

如何调用该过程：

1）exec 过程名（参数值 1，参数值 2...）；

2）call 过程名（参数值 1，参数值 2...）；

ν pl/sql developer 开发工具

pl/sql developer 是用于开发 pl/sql 块的集成开发环境(ide)，它是一个独立的产品，而不是 oracle 的一个附带品。

举一个简单案例：

编写一个存储过程，该过程可以删除某表记录。

```
create or replace procedure sp_pro2 is
begin--执行部分
delete from mytest where name='韩顺平';
end;
```

pl/sql 基础

pl/sql 介绍

ν 介绍

开发人员使用 pl/sql 编写应用模块时，不仅需要掌握 sql 语句的编写方法，还要掌握 pl/sql 语句及语法规则。pl/sql 编程可以使用变量和逻辑控制语句，从而可以编写非常有用的功能模块。比如：分页存储过程模块、订单处理存储过程模块、转账存储过程模块。而且如果使用 pl/sql 编程，我们可以轻松地完成非常复杂的查询要求。

pl/sql 可以做什么

ν 简单分类

```
                                    |————过程(存储过程)
                                    |
                                    |————函数
块(编程)————————|
                                    |————触发器
                                    |
                                    |————包
```

编写规范

ν 编写规范

1.注释

单行注释 --

```
select * from emp where empno=7788;  --取得员工信息
```

多行注释 /*...*/来划分

2.标志符号的命名规范

1).当定义变量时，建议用 v_作为前缀 v_sal

2).当定义常量时，建议用 c_作为前缀 c_rate

3).当定义游标时，建议用_cursor 作为后缀 emp_cursor

4).当定义例外时，建议用 e_作为前缀 e_error

pl/sql 块介绍

ν 介绍

块(block)是 pl/sql 的基本程序单元，编写 pl/sql 程序实际上就是编写 pl/sql 块，要完成相对简单的应用功能，可能只需要编写一个 pl/sql 块，但是如果想要实现复杂的功能，可能需要在一个 pl/sql 块中嵌套其它的 pl/sql 块。

ν 块结构示意图

pl/sql 块由三个部分构成：定义部分，执行部分，例外处理部分。

如下所示：

```
declare
/*定义部分——定义常量、变量、游标、例外、复杂数据类型*/
begin
/*执行部分——要执行的 pl/sql 语句和 sql 语句*/
exception
/*例外处理部分——处理运行的各种错误*/
end;
```

定义部分是从 declare 开始的，该部分是可选的；

执行部分是从 begin 开始的，该部分是必须的；

例外处理部分是从 exception 开始的，该部分是可选的。

可以和 java 编程结构做一个简单的比较。

pl/sql 块的实例(1)

ν 实例 1-只包括执行部分的 pl/sql 块

```
set serveroutput on --打开输出选项
begin
    dbms_output.put_line('hello');
end;
```

相关说明：

dbms_output 是 oracle 所提供的包(类似 java 的开发包)，该包包含一些过程，put_line 就是 dbms_output 包的一个过程。

pl/sql 块的实例(2)

ν 实例 2-包含定义部分和执行部分的 pl/sql 块

```
declare
    v_ename varchar2(5); --定义字符串变量
begin
    select ename into v_ename from emp where empno=&aa;
    dbms_output.put_line('雇员名：'||v_ename);
end;
/
```

如果要把薪水也显示出来，那么执行部分就应该这么写：

```
select ename,sal into v_ename,v_sal from emp where empno=&aa;
```

相关说明：

& 表示要接收从控制台输入的变量。

pl/sql 块的实例(3)

ν 实例 3-包含定义部分，执行部分和例外处理部分

为了避免 pl/sql 程序的运行错误，提高 pl/sql 的健壮性，应该对可能的错误进行处理，这个很有必要。

1.比如在实例 2 中，如果输入了不存在的雇员号，应当做例外处理。

2.有时出现异常，希望用另外的逻辑处理，[网示]

我们看看如何完成 1 的要求。

相关说明：

oracle 事先预定义了一些例外，no_data_found 就是找不到数据的例外。

```
declare
    --定义变量
    v_ename varchar2(5);
    v_sal  number(7,2);
begin
    --执行部分
    select ename,sal into v_ename,v_sal from emp where empno=&aa;
    --在控制台显示用户名
dbms_output.put_line('用户名是：'||v_ename||' 工资：'||v_sal);
    --异常处理
exception
when no_data_found then
    dbms_output.put_line('朋友，你的编号输入有误！');
end;
/
```

17.pl/sql 分类 -- 过程，函数，包，触发器

ν 过程

过程用于执行特定的操作，当建立过程时，既可以指定输入参数(in)，也可以指定输出参数(out)， 通过在过程中使用输入参数，可以将数据传递到执行部分；通过使用输出参数，可以将执行部分的数据传递到应用环境。在 sqlplus 中可以使用 create procedure 命令来建立过程。

实例如下：

1.请考虑编写一个过程，可以输入雇员名，新工资，可修改雇员的工资

2.如何调用过程有两种方法； exec call

3.如何在 java 程序中调用一个存储过程

问题：如何使用过程返回值？

特别说明：

对于过程我们会在以后给大家详细具体的介绍，现在请大家先有一个概念。

```
create procedure sp_pro3(spName varchar2, newSal number) is
--不要写成 number(3,2),表明类型就可以了，不需要大小。就好像 Java 写方法时的参数一样
begin
  --执行部分，根据用户名去修改工资
update emp set sal=newSal where ename=spName;
end;
/
```

java 程序中调用一个存储过程

```
//演示 java 程序去调用 oracle 的存储过程案例
import java.sql.*;
public class TestOraclePro{
    public static void main(String[] args){
        try{
            //1.加载驱动
            Class.forName("oracle.jdbc.driver.OracleDriver");
            //2.得到连接
```

```
            Connection ct =
DriverManager.getConnection("jdbc:oracle:thin@127.0.0.1:1521:MYORA1","scott","m123");
            //3.创建 CallableStatement
            CallableStatement cs = ct.prepareCall("{call sp_pro3(?,?)}");
            //4.给?赋值
            cs.setString(1,"SMITH");
            cs.setInt(2,10);
            //5.执行
            cs.execute();
            //关闭
            cs.close();
            ct.close();
        } catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

ν 函数

函数用于返回特定的数据，当建立函数时，在函数头部必须包含 return 子句。而在函数体内必须包含 return 语句返回的数据。我们可以使用 create function 来建立函数，实际案例：

```
--输入雇员的姓名，返回该雇员的年薪
create function annual_incomec(name varchar2)
return number is
annual_salazy number(7,2);
begin
    --执行部分
    select sal*12+nvl(comm, 0) into annual_salazy from emp where ename=name;
    return annual_salazy;
end;
/
```

如果函数创建过程有编译错误，可以使用 show error;命令显示错误

在 sqlplus 中调用函数

```
SQL> var income number
SQL> call annual_incomec('scott') into: income;
SQL> print income
```

同样我们可以在 java 程序中调用该函数

```
select annual_income('SCOTT') from dual;
```

这样可以通过 rs.getInt(1)得到返回的结果。

ν 包

包用于在逻辑上组合过程和函数，它由包规范和包体两部分组成。

1.我们可以使用 create package 命令来创建包。

实例：

```
--创建一个包 sp_package
--声明该包有一个过程 update_sal
--声明该包有一个函数 annual_income
```

```
create package sp_package is
  procedure update_sal(name varchar2, newsal number);
  function annual_income(name varchar2) return number;
end;
```

包的规范只包含了过程和函数的说明，但是没有过程和函数的实现代码。包体用于实现包规范中的过程和函数。

2.建立包体可以使用 create package body 命令

```
--给包 sp_package 实现包体
create or replace package body sp_package is
  procedure update_sal(name varchar2, newsal number)
  is
  begin
    update emp set sal = newsal where ename = name;
  end;
  function annual_income(name varchar2) return number is
    annual_salary number;
  begin
    select sal * 12 + nvl(comm, 0) into annual_salary from emp
     where ename = name;
    return annual_salary;
  end;
end;
/
```

3.如何调用包的过程或是函数

当调用包的过程或是函数时，在过程和函数前需要带有包名，如果要访问其它方案的包，还需要在包名前加方案名。

如：

```
SQL> call sp_package.update_sal('SCOTT', 1500);
```

特别说明：

包是 pl/sql 中非常重要的部分，我们在使用过程分页时，将会再次体验它的威力呵呵。

v 触发器

触发器是指隐含的执行的存储过程。当定义触发器时，必须要指定触发的事件和触发的操作，常用的触发事件包括 insert,update,delete 语句，而触发操作实际就是一个 pl/sql 块。可以使用 create trigger 来建立触发器。

特别说明：

我们会在后面详细为大家介绍触发器的使用，因为触发器是非常有用的，可维护数据库的安全和一致性。

18.定义并使用变量,复合类型

定义并使用变量

v 介绍

在编写 pl/sql 程序时，可以定义变量和常量；在 pl/sql 程序中包括有：

1.标量类型(scalar)

2.复合类型(composite)

3.参照类型(reference)

4.lob(large object)

v 标量(scalar)——常用类型

在编写 pl/sql 块时，如果要使用变量，需在定义部分定义变量。pl/sql 中定义变量和常量的语法如下：

```
identifier [constant] datatype [not null] [:=| default expr]
```

identifier ： 名称
constant ： 指定常量。需要指定它的初始值，且其值是不能改变的
datatype ： 数据类型
not null ： 指定变量值不能为 null
:= 给变量或是常量指定初始值
default 用于指定初始值
expr ： 指定初始值的 pl/sql 表达式，可以是文本值、其它变量、函数等。
v 标量定义的案例
1.定义一个变长字符串
v_ename varchar2(10);
2.定义一个小数,范围 -9999.99~9999.99
v_sal number(6,2);
3.定义一个小数并给一个初始值为 5.4 :=是 pl/sql 的赋值号
v_sal2 number(6,2):=5.4;
4.定义一个日期类型的数据
v_hiredate date;
5.定义一个布尔变量，不能为空，初始值为 false
v_valid boolean not null default false;
v 标量(scalar)——使用标量
在定义好变量后，就可以使用这些变量。这里需要说明的是 pl/sql 块为变量赋值不同于其它的编程语言，需要在等号前面加冒号(:=)
下面以输入员工号，显示雇员姓名、工资、个人所得税(税率为 0.03)为例。说明变量的使用，看看如何编写。

```
declare
  c_tax_rate number(3,2):=0.03;
  --用户名
  v_ename varchar2(5);
  v_sal number(7,2);
  v_tax_sal number(7,2);
begin
--执行
    select ename,sal into v_ename,v_sal from emp where empno=&no;
--计算所得税
    v_tax_sal := v_sal*c_tax_rate;
--输出
    dbms_output.put_line('姓名是：'||v_ename||'工资：'||v_sal||' 交税：'||v_tax_sal);
end;
/
```

v 标量(scalar)——使用%type 类型
对于上面的 pl/sql 块有一个问题：
就是如果员工的姓名超过了 5 个字符的话，就会有错误，为了降低 pl/sql 程序的维护工作量，可以使用%type 属性定义变量，这样它会按照数据库列来确定你定义的变量的类型和长度。
我们看看这个怎么使用：
标识符名 表名.列名%type;
比如上例的 v_ename，这样定义： v_ename emp.ename%type;

ν 复合变量(composite)——介绍

用于存放多个值的变量。主要包括这几种：

1.pl/sql 记录

2.pl/sql 表

3.嵌套表

4.varray

ν 复合类型——pl/sql 记录

类似于高级语言中的结构体，需要注意的是，当引用 pl/sql 记录成员时，必须要加记录变量作为前缀(记录变量.记录成员)如下：

```
declare
  --定义一个 pl/sql 记录类型 emp_record_type,类型包含 3 个数据 name,salary,title。说白了，就是一个类型可以存放 3 个数据，主要是为了好管理
  type emp_record_type is record(
    name   emp.ename%type,
    salary emp.sal%type,
    title  emp.job%type);
--定义了一个 sp_record 变量，这个变量的类型是 emp_record_type
  sp_record emp_record_type;
begin
  select ename, sal, job into sp_record from emp where empno = 7788;
  dbms_output.put_line ('员工名:' || sp_record.name);
end;
```

ν 复合类型-pl/sql 表

相当于高级语言中的数组，但是需要注意的是在高级语言中数组的下标不能为负数，而 pl/sql 是可以为负数的，并且表元素的下标没有限制。实例如下：

```
declare
--定义了一个 pl/sql 表类型 sp_table_type，该类型是用于存放 emp.ename%type
--index by binary_integer 表示下标是整数
  type sp_table_type is table of emp.ename%type
  index by binary_integer;
--定义了一个 sp_table 变量，这个变量的类型是 sp_table_type
  sp_table sp_table_type;
begin
  select ename into sp_table(-1) from emp where empno = 7788;
  dbms_output.put_line('员工名:' || sp_table(-1));
end;
```

说明：

sp_table_type 是 pl/sql 表类型

emp.ename%type 指定了表的元素的类型和长度

sp_table 为 pl/sql 表变量

sp_table(0) 则表示下标为 0 的元素

注意：如果把 select ename into sp_table(-1) from emp where empno = 7788;变成 select ename into sp_table(-1) from emp;则运行时会出现错误，错误如下：

ORA-01422:实际返回的行数超出请求的行数

解决方法是：使用参照变量（这里不讲）

ν 复合变量——嵌套表(nested table)

ν 复合变量——变长数组(varray)

ν 参照变量——介绍

参照变量是指用于存放数值指针的变量。通过使用参照变量，可以使得应用程序共享相同对象，从而降低占用的空间。在编写 pl/sql 程序时，可以使用游标变量(ref cursor)和对象类型变量(ref obj_type)两种参照变量类型。

ν 参照变量——ref cursor 游标变量

使用游标时，当定义游标时不需要指定相应的 select 语句，但是当使用游标时（open 时）需要指定 select 语句，这样一个游标就与一个 select 语句结合了。实例如下：

1.请使用 pl/sql 编写一个块，可以输入部门号，并显示该部门所有员工姓名和他的工资。

2.在 1 的基础上，如果某个员工的工资低于 200 元，就添加 100 元。

1.

```
declare
--定义游标 sp_emp_cursor
    type sp_emp_cursor is ref cursor;
--定义一个游标变量
    test_cursor sp_emp_cursor;
--定义变量
v_ename emp.ename%type;
v_sal emp.sal%type;
begin
--执行
--把 test_cursor 和一个 select 结合
open test_cursor for select ename,sal from emp where deptno=&no;
--循环取出
loop
    fetch test_cursor into v_ename,v_sal;
    --判断是否 test_cursor 为空
    exit when test_cursor%notfound;
    dbms_output.put_line('名字：'||v_ename||' 工资：'||v_sal);
end loop;
end;
/
```

19.pl/sql 的进阶--控制结构(分支，循环，控制)

玩转 oracle 实战教程(第七天)

内容介绍

1.上节回顾

2.pl/sql 的进阶 √

3.oracle 的视图(具有安全性，和简化复杂查询的功能) √

4.oracle 的触发器 √

期望目标

1.掌握 pl/sql 的高级用法（能缩写分页过程模块，下订单过程模块...）

2.会处理 oracle 常见的例外

3.会编写 oracle 各种触发器

4.理解视图的概念并能灵活使用视图

pl/sql 的进阶--控制结构

ν 介绍

在任何计算机语言(c，java，pascal)都有各种控制语句（条件语句，循环结构，顺序控制结构...）在 pl/sql 中也存在这样的控制结构。

在本部分学习完成后，希望大家达到：

1.使用各种 if 语句

2.使用循环语句

3.使用控制语句——goto 和 null；

ν 条件分支语句

pl/sql 中提供了三种条件分支语句 if—then，if - then - else，if - then - elsif - then

这里我们可以和 java 语句进行一个比较

ν 简单的条件判断 if - then

问题：编写一个过程，可以输入一个雇员名，如果该雇员的工资低于 2000，就给该员工工资增加 10%。

```
create or replace procedure sp_pro6(spName varchar2) is
--定义
v_sal emp.sal%type;
begin
    --执行
    select sal into v_sal from emp where ename=spName;
    --判断
    if v_sal<2000 then
        update emp set sal=sal+sal*10% where ename=spName;
    end if;
end;
/
```

ν 二重条件分支 if - then - else

问题：编写一个过程，可以输入一个雇员名，如果该雇员的补助不是 0 就在原来的基础上增加 100；如果补助为 0 就把补助设为 200；

```
create or replace procedure sp_pro6(spName varchar2) is
--定义
v_comm emp.comm%type;
begin
    --执行
    select comm into v_comm from emp where ename=spName;
    --判断
    if v_comm<>0 then
        update emp set comm=comm+100 where ename=spName;
    else
        update emp set comm=comm+200 where ename=spName;
    end if;
end;
/
```

ν 多重条件分支 if - then - elsif - then

问题：编写一个过程，可以输入一个雇员编号，如果该雇员的职位是 PRESIDENT 就给他的工资增加 1000，如果该雇员的职位是 MANAGER 就给他的工资增加 500，其它职位的雇员工资增加 200。

```
create or replace procedure sp_pro6(spNo number) is
    --定义
    v_job emp.job%type;
begin
    --执行
    select job into v_job from emp where empno=spNo;
    if v_job='PRESIDENT' then
        update emp set sal=sal+1000 where empno=spNo;
    elsif v_job='MANAGER' then
        update emp set sal=sal+500 where empno=spNo;
    else
        update emp set sal=sal+200 where empno=spNo;
    end if;
end;
/
```

v 循环语句 - loop

是 pl/sql 中最简单的循环语句，这种循环语句以 loop 开头，以 end loop 结尾，这种循环至少会被执行一次。

案例：现有一张表 users，表结构如下：

用户 id | 用户名

|

请编写一个过程，可以输入用户名，并循环添加 10 个用户到 users 表中，用户编号从 1 开始增加。

```
create or replace procedure sp_pro6(spName varchar2) is
--定义  :=表示赋值
    v_num number:=1;
begin
    loop
        insert into users values(v_num,spName);
        --判断是否要退出循环
        exit when v_num=10;
        --自增
        v_num:=v_num+1;
    end loop;
end;
/
```

v 环语句 - while 循环

基本循环至少要执行循环体一次，而对于 while 循环来说，只有条件为 true 时，才会执行循环体语句，while 循环以 while...loop 开始，以 end loop 结束。

案例：现有一张表 users，表结构如下：

用户 id 用户名

问题：请编写一个过程，可以输入用户名，并循环添加 10 个用户到 users 表中，用户编号从 11 开始增加。

```
create or replace procedure sp_pro6(spName varchar2) is
--定义  :=表示赋值
    v_num number:=11;
```

```
begin
    while v_num<=20 loop
    --执行
       insert into users values(v_num,spName);
       v_num:=v_num+1;
    end loop;
end;
/
```

v 循环语句 –for 循环

基本 for 循环的基本结构如下

```
begin
  for i in reverse 1..10 loop
    insert into users values (i, 'shunping');
  end loop;
end;
```

我们可以看到控制变量 i，在隐含中就在不停地增加。

v 顺序控制语句 –goto，null

1.goto 语句

goto 语句用于跳转到特定符号去执行语句。注意由于使用 goto 语句会增加程序的复杂性，并使得应用程序可读性变差，所以在做一般应用开发时，建议大家不要使用 goto 语句。

基本语法如下 goto lable，其中 lable 是已经定义好的标号名，

```
declare
  i int := 1;
begin
  loop
    dbms_output.put_line('输出 i=' || i);
    if i = 1{} then
      goto end_loop;
    end if;
    i := i + 1;
  end loop;
  <<end_loop>>
  dbms_output.put_line('循环结束');
end;
```

2.null

null 语句不会执行任何操作，并且会直接将控制传递到下一条语句。使用 null 语句的主要好处是可以提高 pl/sql 的可读性。

```
declare
  v_sal   emp.sal%type;
  v_ename emp.ename%type;
begin
  select ename, sal into v_ename, v_sal from emp where empno = &no;
  if v_sal < 3000 then
    update emp set comm = sal * 0.1 where ename = v_ename;
  else
    null;
```

```
  end if;
end;
```

20.PL/SQL 分页

编写分页过程

v 介绍

分页是任何一个网站(bbs,网上商城，blog)都会使用到的技术，因此学习 pl/sql 编程开发就一定要掌握该技术。看图：

v 无返回值的存储过程

古人云：欲速则不达，为了让大家伙比较容易接受分页过程编写，我还是从简单到复杂，循序渐进的给大家讲解。首先是掌握最简单的存储过程，无返回值的存储过程：

案例：现有一张表 book，表结构如下：看图：

书号 书名 出版社

请写一个过程，可以向 book 表添加书，要求通过 java 程序调用该过程。

```
--in:表示这是一个输入参数，默认为 in
--out:表示一个输出参数

create or replace procedure sp_pro7(spBookId in number,spbookName in varchar2,sppublishHouse in varchar2) is
begin
    insert into book values(spBookId,spbookName,sppublishHouse);
end;
/
--在 java 中调用
//调用一个无返回值的过程
import java.sql.*;
public class Test2{
    public static void main(String[] args){
        try{
            //1.加载驱动
            Class.forName("oracle.jdbc.driver.OracleDriver");
            //2.得到连接
            Connection ct =
DriverManager.getConnection("jdbc:oracle:thin@127.0.0.1:1521:MYORA1","scott","ml23");
            //3.创建 CallableStatement
            CallableStatement cs = ct.prepareCall("{call sp_pro7(?,?,?)}");
            //4.给?赋值
            cs.setInt(1,10);
            cs.setString(2,"笑傲江湖");
            cs.setString(3,"人民出版社");
            //5.执行
            cs.execute();
        } catch(Exception e){
            e.printStackTrace();
        } finally{
            //6.关闭各个打开的资源
            cs.close();
```

```
            ct.close();
        }
    }
}
```

执行，记录被加进去了

v 有返回值的存储过程（非列表）

再看如何处理有返回值的存储过程：

案例：编写一个过程，可以输入雇员的编号，返回该雇员的姓名。

案例扩张：编写一个过程，可以输入雇员的编号，返回该雇员的姓名、工资和岗位。

```
--有输入和输出的存储过程
create or replace procedure sp_pro8
(spno in number, spName out varchar2) is
begin
    select ename into spName from emp where empno=spno;
end;
/
import java.sql.*;
public class Test2{
    public static void main(String[] args){
        try{
            //1.加载驱动
            Class.forName("oracle.jdbc.driver.OracleDriver");
            //2.得到连接
            Connection ct =
DriverManager.getConnection("jdbc:oracle:thin@127.0.0.1:1521:MYORA1","scott","m123");
            //3.创建 CallableStatement
            /*CallableStatement cs = ct.prepareCall("{call sp_pro7(?,?,?)}");
            //4.给?赋值
            cs.setInt(1,10);
            cs.setString(2,"笑傲江湖");
            cs.setString(3,"人民出版社");*/
            //看看如何调用有返回值的过程
            //创建 CallableStatement
            /*CallableStatement cs = ct.prepareCall("{call sp_pro8(?,?)}");
            //给第一个? 赋值
            cs.setInt(1,7788);
            //给第二个? 赋值
            cs.registerOutParameter(2,oracle.jdbc.OracleTypes.VARCHAR);
            //5.执行
            cs.execute();
            //取出返回值,要注意? 的顺序
            String name=cs.getString(2);
            System.out.println("7788 的名字"+name);
        } catch(Exception e){
            e.printStackTrace();
```

```
        } finally{
            //6.关闭各个打开的资源
            cs.close();
            ct.close();
        }
    }
}
```

运行，成功得出结果。。

案例扩张：编写一个过程，可以输入雇员的编号，返回该雇员的姓名、工资和岗位。

```
--有输入和输出的存储过程
create or replace procedure sp_pro8
(spno in number, spName out varchar2,spSal out number,spJob out varchar2) is
begin
    select ename,sal,job into spName,spSal,spJob from emp where empno=spno;
end;
/
import java.sql.*;
public class Test2{
    public static void main(String[] args){
        try{
            //1.加载驱动
            Class.forName("oracle.jdbc.driver.OracleDriver");
            //2.得到连接
            Connection ct =
DriverManager.getConnection("jdbc:oracle:thin@127.0.0.1:1521:MYORA1","scott","m123");
            //3.创建 CallableStatement
            /*CallableStatement cs = ct.prepareCall("{call sp_pro7(?,?,?)}");
            //4.给?赋值
            cs.setInt(1,10);
            cs.setString(2,"笑傲江湖");
            cs.setString(3,"人民出版社");*/
            //看看如何调用有返回值的过程
            //创建 CallableStatement
            /*CallableStatement cs = ct.prepareCall("{call sp_pro8(?,?,?,?)}");
            //给第一个? 赋值
            cs.setInt(1,7788);
            //给第二个? 赋值
            cs.registerOutParameter(2,oracle.jdbc.OracleTypes.VARCHAR);
            //给第三个? 赋值
            cs.registerOutParameter(3,oracle.jdbc.OracleTypes.DOUBLE);
            //给第四个? 赋值
            cs.registerOutParameter(4,oracle.jdbc.OracleTypes.VARCHAR);
            //5.执行
            cs.execute();
            //取出返回值,要注意? 的顺序
```

```
            String name=cs.getString(2);
            String job=cs.getString(4);
            System.out.println("7788 的名字"+name+" 工作："+job);
        } catch(Exception e){
            e.printStackTrace();
        } finally{
            //6.关闭各个打开的资源
            cs.close();
            ct.close();
        }
    }
}
```

运行，成功找出记录

v 有返回值的存储过程（列表[结果集]）

案例：编写一个过程，输入部门号，返回该部门所有雇员信息。

对该题分析如下：

由于 oracle 存储过程没有返回值，它的所有返回值都是通过 out 参数来替代的，列表同样也不例外，但由于是集合，所以不能用一般的参数，必须要用 pagkage 了。所以要分两部分：

返回结果集的过程

1.建立一个包，在该包中，我定义类型 test_cursor,是个游标。 如下：

```
create or replace package testpackage as
  TYPE test_cursor is ref cursor;
end testpackage;
```

2.建立存储过程。如下：

```
create or replace procedure sp_pro9(spNo in number,p_cursor out testpackage.test_cursor) is
begin
  open p_cursor for
    select * from emp where deptno = spNo;
end sp_pro9;
```

3.如何在 java 程序中调用该过程

```
import java.sql.*;
public class Test2{
    public static void main(String[] args){
        try{
            //1.加载驱动
            Class.forName("oracle.jdbc.driver.OracleDriver");
            //2.得到连接
            Connection ct =
DriverManager.getConnection("jdbc:oracle:thin@127.0.0.1:1521:MYORA1","scott","m123");
            //看看如何调用有返回值的过程
            //3.创建 CallableStatement
            /*CallableStatement cs = ct.prepareCall("{call sp_pro9(?,?)}");
            //4.给第？赋值
            cs.setInt(1,10);
            //给第二个？赋值
```

```
            cs.registerOutParameter(2,oracle.jdbc.OracleTypes.CURSOR);
            //5.执行
            cs.execute();
            //得到结果集
            ResultSet rs=(ResultSet)cs.getObject(2);
            while(rs.next()){
                System.out.println(rs.getInt(1)+" "+rs.getString(2));
            }
        } catch(Exception e){
            e.printStackTrace();
        } finally{
            //6.关闭各个打开的资源
            cs.close();
            ct.close();
        }
    }
}
```

运行，成功得出部门号是 10 的所有用户

v 编写分页过程

有了上面的基础，相信大家可以完成分页存储过程了。

要求，请大家编写一个存储过程，要求可以输入表名、每页显示记录数、当前页。返回总记录数，总页数，和返回的结果集。如果大家忘了 oracle 中如何分页，请参考第三天的内容。

先自己完成，老师在后面给出答案，并讲解。

```
--oracle 的分页
select t1.*, rownum rn from (select * from emp) t1 where rownum<=10;
--在分页时，大家可以把下面的 sql 语句当做一个模板使用
select * from
  (select t1.*, rownum rn from (select * from emp) t1 where rownum<=10)
where rn>=6;
--开发一个包
--建立一个包，在该包中，我定义类型 test_cursor,是个游标。 如下：
create or replace package testpackage as
  TYPE test_cursor is ref cursor;
end testpackage;
--开始编写分页的过程
create or replace procedure fenye
    (tableName in varchar2,
     Pagesize in number,--一页显示记录数
     pageNow in number,
     myrows out number,--总记录数
     myPageCount out number,--总页数
     p_cursor out testpackage.test_cursor--返回的记录集
    ) is
--定义部分
--定义 sql 语句 字符串
```

```
v_sql varchar2(1000);
--定义两个整数
v_begin number:=(pageNow-1)*Pagesize+1;
v_end number:=pageNow*Pagesize;
begin
--执行部分
v_sql:='select * from (select t1.*, rownum rn from (select * from '||tableName||') t1 where rownum<='||v_end||')
where rn>='||v_begin;
--把游标和 sql 关联
open p_cursor for v_sql;
--计算 myrows 和 myPageCount
--组织一个 sql 语句
v_sql:='select count(*) from '||tableName;
--执行 sql,并把返回的值，赋给 myrows;
execute inmediate v_sql into myrows;
--计算 myPageCount
--if myrows%Pagesize=0 then 这样写是错的
if mod(myrows,Pagesize)=0 then
  myPageCount:=myrows/Pagesize;
else
  myPageCount:=myrows/Pagesize+1
end if;
--关闭游标
close p_cursor;
end;
/
--使用 java 测试
//测试分页
import java.sql.*;
public class FenYe{
    public static void main(String[] args){

        try{
            //1.加载驱动
            Class.forName("oracle.jdbc.driver.OracleDriver");
            //2.得到连接
            Connection ct =
DriverManager.getConnection("jdbc:oracle:thin@127.0.0.1:1521:MYORA1","scott","m123");
            //3.创建 CallableStatement
            CallableStatement cs = ct.prepareCall("{call fenye(?,?,?,?,?,?)}");
            //4.给第? 赋值
            cs.seString(1,"emp");
            cs.setInt(2,5);
            cs.setInt(3,2);
            //注册总记录数
```

```
            cs.registerOutParameter(4,oracle.jdbc.OracleTypes.INTEGER);
            //注册总页数
            cs.registerOutParameter(5,oracle.jdbc.OracleTypes.INTEGER);
            //注册返回的结果集
            cs.registerOutParameter(6,oracle.jdbc.OracleTypes.CURSOR);
            //5.执行
            cs.execute();
            //取出总记录数   /这里要注意，getInt(4)中 4，是由该参数的位置决定的
            int rowNum=cs.getInt(4);
            int pageCount = cs.getInt(5);
            ResultSet rs=(ResultSet)cs.getObject(6);
            //显示一下，看看对不对
            System.out.println("rowNum="+rowNum);
            System.out.println("总页数="+pageCount);
            while(rs.next()){
                System.out.println("编号："+rs.getInt(1)+" 名字："+rs.getString(2)+" 工资："+rs.getFloat(6));
            }
        } catch(Exception e){
            e.printStackTrace();
        } finally{
            //6.关闭各个打开的资源
            cs.close();
            ct.close();
        }
    }
}
```

运行，控制台输出：

```
rowNum=19
总页数：4
编号：7369 名字：SMITH 工资：2850.0
编号：7499 名字：ALLEN 工资：2450.0
编号：7521 名字：WARD 工资：1562.0
编号：7566 名字：JONES 工资：7200.0
编号：7654 名字：MARTIN 工资：1500.0
```

--新的需要，要求按照薪水从低到高排序，然后取出 6-10

过程的执行部分做下改动，如下：

```
begin
--执行部分
v_sql:='select * from (select t1.*, rownum rn from (select * from '||tableName||' order by sal) t1 where
rownum<='||v_end||') where rn>='||v_begin;
```

重新执行一次 procedure,java 不用改变，运行，控制台输出：

```
rowNum=19
总页数：4
编号：7900 名字：JAMES 工资：950.0
编号：7876 名字：ADAMS 工资：1100.0
```

编号：7521 名字：WARD 工资：1250.0
编号：7654 名字：MARTIN 工资：1250.0
编号：7934 名字：MILLER 工资：1300.0

21.例外处理

例外处理
v 例外的分类
oracle 将例外分为预定义例外，非预定义例外和自定义例外三种。
预定义例外用于处理常见的 oracle 错误
非预定义例外用于处理预定义例外不能处理的例外
自定义例外用于处理与 oracle 错误无关的其它情况
v 例外传递
如果不处理例外我们看看会出现什么情况：
案例，编写一个过程，可接收雇员的编号，并显示该雇员的姓名。
问题是，如果输入的雇员编号不存在，怎样去处理呢？

```
--例外案例
declare
--定义
v_ename emp.ename%type;
begin
--
select ename into v_ename from emp where empno=&gno;
dbms_output.put_line('名字：'||v_ename)
/
```

执行，弹出框，看图：
随便输个不在的编号，回车，会抛出异常，显示：
ORA-01403: 未找到数据
ORA-06512: 在 line 6

```
declare
--定义
v_ename emp.ename%type;
begin
--
select ename into v_ename from emp where empno=&gno;
dbms_output.put_line('名字：'||v_ename)
exception
    when no_data_found then
    dbms_output.put_line('编号没有！');
/
```

执行，输入一个不存在的编号，回车，显示：
编号没有！
v 处理预定义例外
预定义例外是由 pl/sql 所提供的系统例外。当 pl/sql 应用程序违反了 oracle 规定的限制时，则会隐含的触发一个内部例外。
pl/sql 为开发人员提供了二十多个预定义例外。我们给大家介绍常用的例外。

ν 预定义例外 case_not_found

在开发 pl/sql 块中编写 case 语句时，如果在 when 子句中没有包含必须的条件分支，就会触发 case_not_found 的例外：

```
create or replace procedure sp_pro6(spno number) is
  v_sal emp.sal%type;
begin
  select sal into v_sal from emp where empno = spno;
  case
    when v_sal < 1000 then
      update emp set sal = sal + 100 where empno = spno;
    when v_sal < 2000 then
      update emp set sal = sal + 200 where empno = spno;
  end case;
exception
  when case_not_found then
    dbms_output.put_line('case 语句没有与' || v_sal || '相匹配的条件');
end;
```

ν 预定义例外 cursor_already_open

当重新打开已经打开的游标时，会隐含的触发例外 cursor_already_open

```
declare
  cursor emp_cursor is select ename, sal from emp;
begin
  open emp_cursor;
  for emp_record1 in emp_cursor loop
    dbms_output.put_line(emp_record1.ename);
  end loop;
exception
  when cursor_already_open then
    dbms_output.put_line('游标已经打开');
end;
/
```

ν 预定义例外 dup_val_on_index

在唯一索引所对应的列上插入重复的值时，会隐含的触发例外 dup_val_on_index 例外

```
begin
  insert into dept values (10, '公关部', '北京');
exception
  when dup_val_on_index then
    dbms_output.put_line('在 deptno 列上不能出现重复值');
end;
```

ν 预定义例外 invalid_cursor

当试图在不合法的游标上执行操作时，会触发该例外

例如：试图从没有打开的游标提取数据，或是关闭没有打开的游标。则会触发该例外

```
declare
  cursor emp_cursor is select ename, sal from emp;
  emp_record emp_cursor%rowtype;
begin
```

```
  --open emp_cursor; --打开游标
  fetch emp_cursor into emp_record;
  dbms_output.put_line(emp_record.ename);
  close emp_cursor;
exception
  when invalid_cursor then
    dbms_output.put_line('请检测游标是否打开');
end;
```

ν 预定义例外 invalid_number

当输入的数据有误时，会触发该例外

比如：数字 100 写成了 loo 就会触发该例外

```
begin
  update emp set sal= sal + 'loo';
exception
  when invalid_number then
    dbms_output.put_line('输入的数字不正确');
end;
```

预定义例外 no_data_found

下面是一个 pl/sql 块，当执行 select into 没有返回行，就会触发该例外

```
declare
    v_sal emp.sal%type;
begin
    select sal into v_sal from emp
    when ename='&name';
exception
    when no_data_found then
    dbms_output.put_line('不存在该员工');
end;
```

ν 预定义例外 too_many_rows

当执行 select into 语句时，如果返回超过了一行，则会触发该例外。

```
declare
  v_ename emp.ename%type;
begin
  select ename into v_ename from emp;
exception
  when too_many_rows then
    dbms_output.put_line('返回了多行');
end;
```

ν 预义例外 zero_divide

当执行 2/0 语句时，则会触发该例外。

ν 预定义例外 value_error

当在执行赋值操作时，如果变量的长度不足以容纳实际数据，则会触发该例外 value_error，比如：

```
declare
  v_ename varchar2(5);
begin
```

```
  select ename into v_ename from emp where empno = &no1;
  dbms_output.put_line(v_ename);
exception
  when value_error then
    dbms_output.put_line('变量尺寸不足');
end;
```

v 其它预定义例外(这些例外不是在 pl/sql 里触发的，而是在用 oracle 时触发的，所以取名叫其它预定义例外)

1.login_denied

当用户非法登录时，会触发该例外

2.not_logged_on

如果用户没有登录就执行 dml 操作，就会触发该例外

3.storage_error

如果超过了内存空间或是内存被损坏，就触发该例外

4.timeout_on_resource

如果 oracle 在等待资源时，出现了超时就触发该例外

v 非预定义例外

非预定义例外用于处理与预定义例外无关的 oracle 错误。使用预定义例外只能处理 21 个 oracle 错误，而当使用 pl/sql 开发应用程序时，可能会遇到其它的一些 oracle 错误。比如在 pl/sql 块中执行 dml 语句时，违反了约束规定等等。在这样的情况下，也可以处理 oracle 的各种例外，因为非预定义例外用的不多，这里我就不举例了。

v 处理自定义例外

预定义例外和自定义例外都是与 oracle 错误相关的，并且出现的 oracle 错误会隐含的触发相应的例外；而自定义例外与 oracle 错误没有任何关联，它是由开发人员为特定情况所定义的例外.

问题：请编写一个 pl/sql 块，接收一个雇员的编号，并给该雇员工资增加 1000 元，如果该雇员不存在，请提示。

```
--自定义例外
create or replace procedure ex_test(spNo number)
is
begin
--更新用户 sal
update emp set sal=sal+1000 where empno=spNo;
end;
/
```

运行，该过程被成功创建。

```
SQL> exec ex_test(56);
```

PL/SQL 过程被成功完成

这里，编号为 56 是不存在的，刚才的报异常了，为什么现在不报异常呢？

因为刚才的是 select 语句

怎么解决这个问题呢？ 修改代码，如下：

```
--自定义例外
create or replace procedure ex_test(spNo number)
is
--定义一个例外
myex exception;
begin
--更新用户 sal
update emp set sal=sal+1000 where empno=spNo;
```

```
--sql%notfound 这是表示没有 update
--raise myex;触发 myex
if sql%notfound then
raise myex;
end if;
exception
when myex then
dbms_output.put_line('没有更新任何用户');
end;
/
```

现在再测试一次：

```
SQL> exec ex_test(56);
```

没有更新任何用户

22.oracle 的视图

oracle 的视图

ν 介绍

视图是一个虚拟表，其内容由查询定义，同真实的表一样，视图包含一系列带有名称的列和行数据。但是，视图并不在数据库中以存储的数据值集形式存在。行和列数据来自由定义视图的查询所引用的表，并且在引用视图时动态生成。（视图不是真实存在磁盘上的）

看图：

视与表的区别

ν 视图与表的区别

1.表需要占用磁盘空间，视图不需要

2.视图不能添加索引（所以查询速度略微慢点）

3.使用视图可以简化，复杂查询

比如：学生选课系统

4.视图的使用利于提高安全性

比如：不同用户查看不同视图

创建/修改视图

ν 创建视图

```
create view 视图名 as select 语句 [with read only]
```

ν 创建或修改视图

```
create or replace view 视图名 as select 语句 [with read only]
```

ν 删除视图

```
drop view 视图名
```

当表结构国语复杂，请使用视图吧！

```
--创建视图，把 emp 表的 sal<1000 的雇员映射到该视图（view）

create view myview as select * from emp where sal<1000;
--为简化操作，用一个视图解决 显示雇员编号，姓名和部门名称
create view myview2 as select emp.empno,emp.ename,dept.dname from emp,dept where emp.deptno=dept.deptno;
```

视图之间也可以做联合查询