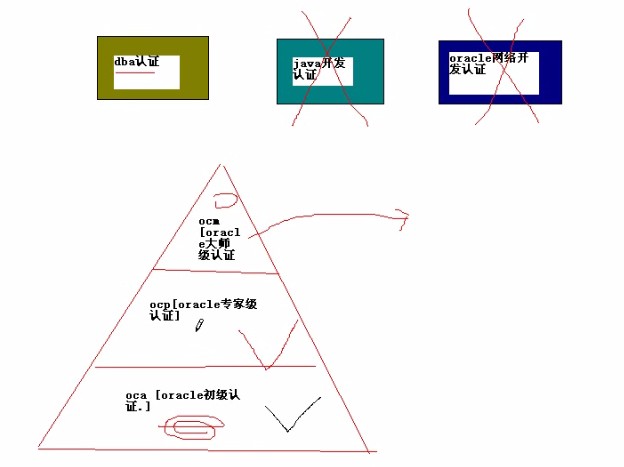
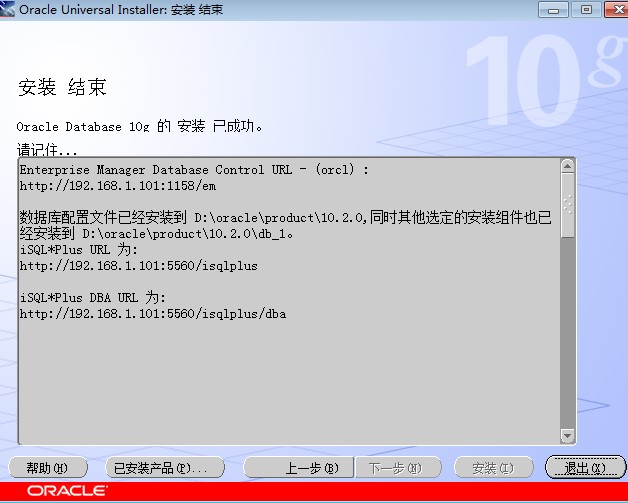
韩顺平玩转Oracle9i

# Oracle认证和安装，与其他数据库比较

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。

# [Oracle的基本使用--基本命令](http://taeky.javaeye.com/blog/605375)

sql\*plus的常用命令

ν **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

ν **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

ν **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

ν **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

**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

# [oracle用户管理](http://taeky.javaeye.com/blog/606336)

ν **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)

ν **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

ν **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;

ν **用户管理的综合案例**

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。

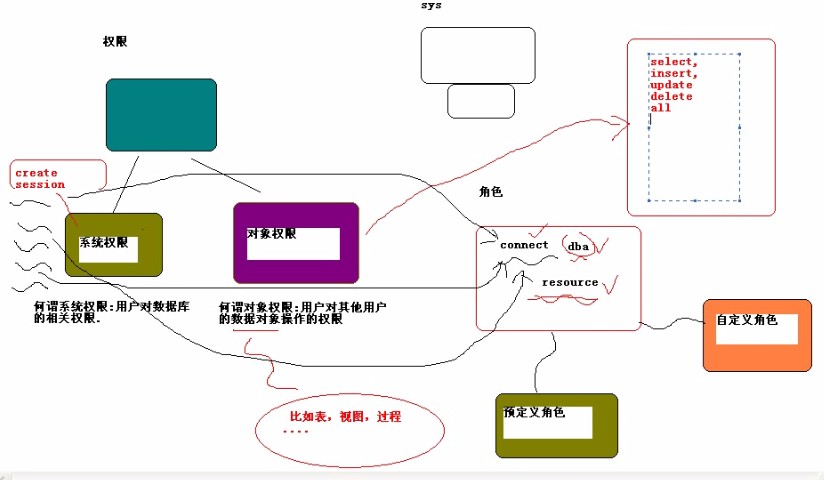
SQL> grant connect to xiaoming;

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。。 看图：   
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。。

ν **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

ν **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

# [oracle表的管理(数据类型，表创建删除，数据 CRUD操作)](http://taeky.javaeye.com/blog/607467)

内容介绍

* 1.上节回顾
* 2.oracle的表的管理
* 3.基本查询
* 4.复杂查询
* 5.oracle数据库的创建  
  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**

ν **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

ν 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

ν 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

ν 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;

ν 修改表的名字   很少有这种需求

SQL>RENAME student TO stu;

ν 删除表

SQL>DROP TABLE student;

添加数据

ν 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);

ν 插入部分字段

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

ν 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

ν 修改一个字段

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

ν 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;

ν 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。

# [oracle表查询(1)](http://taeky.javaeye.com/blog/608779)

ν 介绍

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', 'aaaaaaaaaaaaaaaaaaaaaaa');

--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;

# [oracle表查询(2)](http://taeky.javaeye.com/blog/609924)

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

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

ν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）

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           200   
        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

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

  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？

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。

# [java操作oracle](http://taeky.javaeye.com/blog/611443)

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。

Java 代码

1. package com.sp;
3. import java.sql.Connection;
4. import java.sql.DriverManager;
5. import java.sql.ResultSet;
6. import java.sql.Statement;
8. //演示  如何使用 jdbc\_odbc桥连接方式
9. public class TestOracle {
11. public static void main(String[] args) {
12. try {
14. // 1.加载驱动
15. Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
17. // 2.得到连接
18. Connection ct = DriverManager.getConnection(
19. "jdbc.odbc:testConnectOracle", "scott",
21. "tiger");
23. // 从下面开始，和SQL Server一模一样
24. Statement sm = ct.createStatement();
25. ResultSet rs = sm.executeQuery("select \* from emp");
26. while (rs.next()) {
27. //用户名
28. System.out.println("用户名： "+rs.getString(2));
29. //默认是从1开始编号的
30. }
31. } catch (Exception e) {
32. e.printStackTrace();
33. }
34. }
35. }

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

}

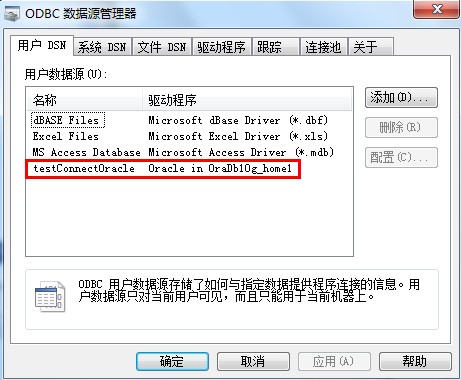
} catch (Exception e) {

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

Java 代码

1. package com.sp;
3. import java.sql.Connection;
4. import java.sql.DriverManager;
5. import java.sql.ResultSet;
6. import java.sql.Statement;
8. //使用 jdbc连接oracle
9. public class TestOracle2 {
11. public static void main(String[] args) {
12. try {
14. // 1.加载驱动
15. Class.forName("oracle.jdbc.driver.OracleDriver");
17. // 2.得到连接
18. Connection ct = DriverManager.getConnection
20. ("jdbc:oracle:thin:@127.0.0.1:1521:orcl", "scott", "tiger");
22. // 从下面开始，和SQL Server一模一样
23. Statement sm = ct.createStatement();
24. ResultSet rs = sm.executeQuery("select \* from emp");
25. while (rs.next()) {
26. //用户名
27. System.out.println("用户名： "+rs.getString(2));
28. //默认是从1开始编号的
29. }
30. } catch (Exception e) {
31. e.printStackTrace();
32. }
33. }
34. }

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) {

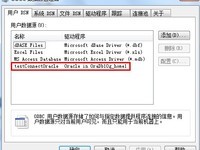
e.printStackTrace();

}

}

}

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';

* [](http://dl.javaeye.com/upload/attachment/214936/1146d7c7-bc19-3b28-9f0a-dee295edce02.jpg)
* 大小: 33.8 KB
* [](http://dl.javaeye.com/upload/attachment/214938/b5003e5e-faf1-395d-9e52-44866b9377be.jpg)
* 大小: 55.5 KB

# [oracle中事务处理](http://taeky.javaeye.com/blog/612273)

**文章分类:**[**数据库**](http://www.javaeye.com/blogs/category/database)

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？

Java 代码

1. package com.sp;
3. import java.sql.Connection;
4. import java.sql.DriverManager;
5. import java.sql.ResultSet;
6. import java.sql.Statement;
8. public class TestTrans {
10. public static void main(String[] args) {
11. try {
13. // 1.加载驱动
14. Class.forName("oracle.jdbc.driver.OracleDriver");
16. // 2.得到连接
17. Connection ct = DriverManager.getConnection(
18. "jdbc:oracle:thin:@127.0.0.1:1521:orcl", "scott", "tiger");
20. Statement sm = ct.createStatement();
22. // 从scott的sal中减去100
23. sm.executeUpdate("update emp set sal=sal-100 where ename='SCOTT'");
25. int i = 7 / 0;
27. // 给smith的sal加上100
28. sm.executeUpdate("update emp set sal=sal+100 where ename='SMITH'");
30. // 关闭打开的资源
31. sm.close();
32. ct.close();
33. } catch (Exception e) {
34. e.printStackTrace();
35. }
37. }
39. }

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？

Java 代码

1. package com.sp;
3. import java.sql.Connection;
4. import java.sql.DriverManager;
5. import java.sql.SQLException;
6. import java.sql.Statement;
8. public class TestTrans {
10. public static void main(String[] args) {
11. Connection ct = null;
12. try {
13. // 1.加载驱动
14. Class.forName("oracle.jdbc.driver.OracleDriver");
16. // 2.得到连接
17. ct = DriverManager.getConnection(
18. "jdbc:oracle:thin:@127.0.0.1:1521:orcl", "scott", "tiger");
20. // 加入事务处理
21. ct.setAutoCommit(false);// 设置不 能默认提交
23. Statement sm = ct.createStatement();
25. // 从scott的sal中减去100
26. sm.executeUpdate("update emp set sal=sal-100 where ename='SCOTT'");
28. int i = 7 / 0;
30. // 给smith的sal加上100
31. sm.executeUpdate("update emp set sal=sal+100 where ename='SMITH'");
33. // 提交事务
34. ct.commit();
36. // 关闭打开的资源
37. sm.close();
38. ct.close();
39. } catch (Exception e) {
40. // 如果发生异常，就回滚
41. try {
42. ct.rollback();
43. } catch (SQLException e1) {
44. e1.printStackTrace();
45. }
46. e.printStackTrace();
47. }
49. }
51. }

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;

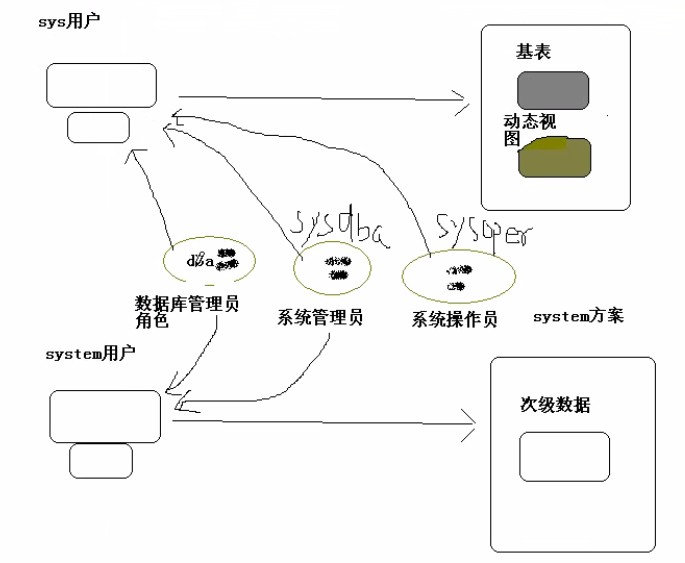
# [oracle的函数](http://taeky.javaeye.com/blog/612783)

**文章分类:**[**数据库**](http://www.javaeye.com/blogs/category/database)

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;   
  
ν 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   
ν 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;

# [数据库管理，表的逻辑备份与恢复](http://taeky.javaeye.com/blog/613937)

**文章分类:**[**数据库**](http://www.javaeye.com/blogs/category/database)

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   
ν 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：   
ν 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   
  
  
ν 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，sysoper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

# [数据字典和动态性能视图](http://taeky.javaeye.com/blog/617732)

**文章分类:**[**数据库**](http://www.javaeye.com/blogs/category/database)

介绍   
ν 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'   
  
  
ν 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%';   
ν 显示当前数据库的全称   
select \* from global\_name;   
  
ν 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。   
  
  
  
ν **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。

# [数据库管理 -- 管理表空间和数据文件](http://taeky.javaeye.com/blog/621312)

**文章分类:**[**数据库**](http://www.javaeye.com/blogs/category/database)

ν 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。

# [约束](http://taeky.javaeye.com/blog/622287)

**文章分类:**[**数据库**](http://www.javaeye.com/blogs/category/database)

玩转oracle实战教程(第五天)   
内容介绍   
1.上节回顾   
2.维护数据的完整性 √   
3.管理索引 √   
4.管理权限和角色 √   
  
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   
介绍   
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));

# [Oracle索引、权限](http://taeky.javaeye.com/blog/622795)

**文章分类:**[**数据库**](http://www.javaeye.com/blogs/category/database)

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）

# [角色](http://taeky.javaeye.com/blog/660050)

**文章分类:**[**数据库**](http://www.javaeye.com/blogs/category/database)

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。

# [PL/SQL 块的结构和实例](http://taeky.javaeye.com/blog/661173)

**文章分类:**[**数据库**](http://www.javaeye.com/blogs/category/database)

韩顺平.玩转oralce第24讲.plsql编程(1)   
玩转oracle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。   
看图：   
  
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

Sql 代码

1. create table mytest(name varchar2(30),passwd varchar2(30));

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

2.创建过程

Sql代 码

1. **create or replace** procedure sp\_pro1 is
2. begin--执行部分
3. insert into mytest values('韩顺平','m1234')；
4. end;
5. /

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。

Sql代 码

1. create or replace procedure sp\_pro2 is
2. begin--执行部分
3. delete from mytest where name='韩顺平'；
4. end;

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

ν 编写规范   
1.注释   
单行注释 --

Sql代 码

1. select \* from emp where empno=7788;  --取得员工信息

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

Sql代 码

1. set serveroutput on --打开输出选项
2. begin
3. dbms\_output.put\_line('hello');
4. end;

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

Sql代 码

1. declare
2. v\_ename varchar2(5); --定义字符串变量
3. begin
4. select ename into v\_ename from emp where empno=&aa;
5. dbms\_output.put\_line('雇员名：'||v\_ename);
6. end;
7. /

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：

Sql代 码

1. select ename,sal into v\_ename,v\_sal from emp where empno=&aa;

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。看图：   
  
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。

Sql代 码

1. declare
2. --定义变量
3. v\_ename varchar2(5);
4. v\_sal  number(7,2);
5. begin
6. --执行部分
7. select ename,sal into v\_ename,v\_sal from emp where empno=&aa;
8. --在控制台显示用户名
9. dbms\_output.put\_line('用户名是：'||v\_ename||' 工资：'||v\_sal);
10. --异常处理
11. exception
12. when no\_data\_found then
13. dbms\_output.put\_line('朋友，你的编号输入有误！')；
14. end;
15. /

# [pl/sql分类 -- 过程，函数，包，触发器](http://taeky.javaeye.com/blog/662707)

**文章分类:**[**数据库**](http://www.javaeye.com/blogs/category/database)

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

Sql代 码

1. begin
2. --执行部分，根据用户名去修改工资
3. update emp set sal=newSal where ename=spName;
4. end;
5. /

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

Java 代码

1. import java.sql.\*;
2. public class TestOraclePro{
3. public static void main(String[] args){
5. try{
6. //1.加载驱动
7. Class.forName("oracle.jdbc.driver.OracleDriver");
8. //2.得到连接
9. Connection ct = DriverManager.getConnection("jdbc:oracle:thin@127.0.0.1:1521:MYORA1","scott","m123");
11. //3.创建CallableStatement
12. CallableStatement cs = ct.prepareCall("{call sp\_pro3(?,?)}");
13. //4.给?赋值
14. cs.setString(1,"SMITH");
15. cs.setInt(2,10);
16. //5.执行
17. cs.execute();
18. //关闭
19. cs.close();
20. ct.close();
21. } catch(Exception e){
22. e.printStackTrace();
23. }
24. }
25. }

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：

Sql代 码

1. --输入雇员的姓名，返回该雇员的年薪
2. create function annual\_incomec(name varchar2)
3. return number is
4. annual\_salazy number(7,2);
5. begin
6. --执行部分
7. select sal\*12+nvl(comm, 0) into annual\_salazy from emp where ename=name;
8. return annual\_salazy;
9. end;
10. /

--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代 码

1. SQL> var income number
2. SQL> call annual\_incomec('scott') into: income;
3. SQL> print income

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(l)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

Sql代 码

1. create package sp\_package is
2. procedure update\_sal(name varchar2, newsal number);
3. function annual\_income(name varchar2) return number;
4. end;

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

Sql代 码

1. create or replace package body sp\_package is
2. procedure update\_sal(name varchar2, newsal number)
3. is
4. begin
5. update emp set sal = newsal where ename = name;
6. end;
7. function annual\_income(name varchar2) return number is
8. annual\_salary number;
9. begin
10. select sal \* 12 + nvl(comm, 0) into annual\_salary from emp
11. where ename = name;
12. return annual\_salary;
13. end;
14. end;
15. /

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。   
  
ν 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。

# [定义并使用变量,复合类型](http://taeky.javaeye.com/blog/663636)

**文章分类:**[**数据库**](http://www.javaeye.com/blogs/category/database)

定义并使用变量   
ν 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)   
  
ν 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。   
  
ν 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;   
  
ν 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。

Sql代 码

1. declare
2. c\_tax\_rate number(3,2):=0.03;
3. --用户名
4. v\_ename varchar2(5);
5. v\_sal number(7,2);
6. v\_tax\_sal number(7,2);
7. begin
8. --执行
9. select ename,sal into v\_ename,v\_sal from emp where empno=&no;
10. --计算所得税
11. v\_tax\_sal := v\_sal\*c\_tax\_rate;
12. --输出
13. dbms\_output.put\_line('姓名是：'||v\_ename||'工资：'||v\_sal||' 交 税：'||v\_tax\_sal);
14. end;
15. /

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;

/

ν 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：

Sql代 码

1. declare
2. --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
3. type emp\_record\_type is record(
4. name   emp.ename%type,
5. salary emp.sal%type,
6. title  emp.job%type);
7. --定义了一个sp\_record变量，这个变量的类型是emp\_record\_type
8. sp\_record emp\_record\_type;
9. begin
10. select ename, sal, job into sp\_record from emp where empno = 7788;
11. dbms\_output.put\_line ('员工名:' || sp\_record.name);
12. end;

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：

Sql代 码

1. declare
2. --定义了一个pl/sql表类型sp\_table\_type，该类型是用于存放emp.ename%type
3. --index by binary\_integer 表示下标是整数
4. type sp\_table\_type is table of emp.ename%type
5. index by binary\_integer;
6. --定义了一个sp\_table变量，这个变量的类型是sp\_table\_type
7. sp\_table sp\_table\_type;
8. begin
9. –将7788这个ename放在下标为-1的sp\_table中
10. select ename into sp\_table(-1) from emp where empno = 7788;
11. dbms\_output.put\_line('员工名:' || sp\_table(-1));
12. end;

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.

Sql代 码

1. declare
2. --定义游标sp\_emp\_cursor
3. type sp\_emp\_cursor is ref cursor;
4. --定义一个游标变量
5. test\_cursor sp\_emp\_cursor;
6. --定义变量
7. v\_ename emp.ename%type;
8. v\_sal emp.sal%type;
9. begin
10. --执行
11. --把test\_cursor和一个select结合
12. open test\_cursor for select ename,sal from emp where deptno=&no;
13. --循环取出
14. loop
15. fetch test\_cursor into v\_ename,v\_sal;
16. --判断是否test\_cursor为空
17. exit when test\_cursor%notfound;
18. dbms\_output.put\_line('名字：'||v\_ename||' 工 资：'||v\_sal);
19. end loop;
20. end;

用游标处理的案例能否用复合变量-PL/SQL记录来处理？

答案：不能

declare

*--定义一个pl/sql记录类型，类型包括2个数据ename，sal*

type myemp\_record\_type is record(

name emp.ename%type,salary emp.sal%type);

*--定义一个变量mytest，这个变量类型是myemp\_record\_type*

mytest myemp\_record\_type;

*--执行部分*

begin

select ename,sal into mytest from emp where DEPTNO=&no;

dbms\_output.put\_line('用工姓名：'||mytest.name||'员工工资：'||mytest.salary);

end;

执行之后：

ORA-01422: 实际返回的行数超出请求的行数

ORA-06512: 在 line 10

这个问题能否用复合变量——PL/SQL表来处理？——不能

declare

*--定义一个pl/sql表类型*

type sp\_table\_type is table of (emp.ename%type,emp.sal%type)

index by binary\_integer;

sp\_table sp\_table\_type;

begin

select ename,sal into sp\_table(0) from emp where DEPTNO=&no;

dbms\_output.put\_line('员工名：'||sp\_table(0));

end;

执行之后：

ORA-06550: 第 4 行, 第 32 列:

PLS-00103: 出现符号 "("在需要下列之一时：

<an identifier>

<a double-quoted delimited-identifier> long double ref char

time timestamp interval date binary national character nchar

符号 "<an identifier>" 被替换为 "(" 后继续。

1. /

# [pl/sql的进阶--控制结构(分支，循环，控制)](http://taeky.javaeye.com/blog/664462)

**文章分类:**[**数据库**](http://www.javaeye.com/blogs/category/database)

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%。

Sql代 码

1. create or replace procedure sp\_pro6(spName varchar2) is
2. --定义
3. v\_sal emp.sal%type;
4. begin
5. --执行
6. select sal into v\_sal from emp where ename=spName;
7. --判断
8. if v\_sal<2000 then
9. update emp set sal=sal+sal\*10% where ename=spName;
10. end if;
11. end;
12. /

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；

Sql代 码

1. create or replace procedure sp\_pro6(spName varchar2) is
2. --定义
3. v\_comm emp.comm%type;
4. begin
5. --执行
6. select comm into v\_comm from emp where ename=spName;
7. --判断
8. if v\_comm<>0 then
9. update emp set comm=comm+100 where ename=spName;
10. else
11. update emp set comm=comm+200 where ename=spName;
12. end if;
13. end;
14. /

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。

Sql代 码

1. create or replace procedure sp\_pro6(spNo number) is
2. --定义
3. v\_job emp.job%type;
4. begin
5. --执行
6. select job into v\_job from emp where empno=spNo;
7. if v\_job='PRESIDENT' then
8. update emp set sal=sal+1000 where empno=spNo;
9. elsif v\_job='MANAGER' then
10. update emp set sal=sal+500 where empno=spNo;
11. else
12. update emp set sal=sal+200 where empno=spNo;
13. end if;
14. end;
15. /

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;

/

ν 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。

Sql代 码

1. create or replace procedure sp\_pro6(spName varchar2) is
2. --定义  :=表示赋值
3. v\_num number:=1;
4. begin
5. loop
6. insert into users values(v\_num,spName);
7. --判断是否要退出循环
8. exit when v\_num=10;
9. --自增
10. v\_num:=v\_num+1;
11. end loop;
12. end;
13. /

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;

/

ν 循环语句 –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。

Sql代 码

1. create or replace procedure sp\_pro6(spName varchar2) is
2. --定义  :=表示赋值
3. v\_num number:=11;
4. begin
5. while v\_num<=20 loop
6. --执行
7. insert into users values(v\_num,spName);
8. v\_num:=v\_num+1;
9. end loop;
10. end;
11. /

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;

/

ν 循环语句 –for循环   
基本for循环的基本结构如下

Sql代 码

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

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。   
  
ν 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，

Sql代 码

1. declare
2. i int := 1;
3. begin
4. loop
5. dbms\_output.put\_line('输出i=' || i);
6. if i = 1{} then
7. goto end\_loop;
8. end if;
9. i := i + 1;
10. end loop;
11. <<end\_loop>>
12. dbms\_output.put\_line('循环结束');
13. end;

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。

Sql代 码

1. declare
2. v\_sal   emp.sal%type;
3. v\_ename emp.ename%type;
4. begin
5. select ename, sal into v\_ename, v\_sal from emp where empno = &no;
6. if v\_sal < 3000 then
7. update emp set comm = sal \* 0.1 where ename = v\_ename;
8. else
9. null;
10. end if;
11. end;

# [PL/SQL分页](http://taeky.javaeye.com/blog/665541)

**文章分类:**[**数据库**](http://www.javaeye.com/blogs/category/database)

编写分页过程   
ν 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：   
  
  
  
ν 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

Sql代 码

1. create or replace procedure sp\_pro7(spBookId in number,spbookName in varchar2,sppublishHouse in varchar2) is
2. begin
3. insert into book values(spBookId,spbookName,sppublishHouse);
4. end;
5. /

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

Java 代码

1. //调用一个无返回值的过程
2. import java.sql.\*;
3. public class Test2{
4. public static void main(String[] args){
6. try{
7. //1.加载驱动
8. Class.forName("oracle.jdbc.driver.OracleDriver");
9. //2.得到连接
10. Connection ct = DriverManager.getConnection("jdbc:oracle:thin@127.0.0.1:1521:MYORA1","scott","m123");
12. //3.创建CallableStatement
13. CallableStatement cs = ct.prepareCall("{call sp\_pro7(?,?,?)}");
14. //4.给?赋值
15. cs.setInt(1,10);
16. cs.setString(2,"笑傲江湖");
17. cs.setString(3,"人民出版社");
18. //5.执行
19. cs.execute();
20. } catch(Exception e){
21. e.printStackTrace();
22. } finally{
23. //6.关闭各个打开的资源
24. cs.close();
25. ct.close();
26. }
27. }
28. }

//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","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");

//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   
  
ν 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。

Sql代 码

1. --有输入和输出的存储过程
2. create or replace procedure sp\_pro8
3. (spno in number, spName out varchar2) is
4. begin
5. select ename into spName from emp where empno=spno;
6. end;
7. /

--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;

/

Java 代码

1. import java.sql.\*;
2. public class Test2{
3. public static void main(String[] args){
5. try{
6. //1.加载驱动
7. Class.forName("oracle.jdbc.driver.OracleDriver");
8. //2.得到连接
9. Connection ct = DriverManager.getConnection("jdbc:oracle:thin@127.0.0.1:1521:MYORA1","scott","m123");
11. //3.创建CallableStatement
12. /\*CallableStatement cs = ct.prepareCall("{call sp\_pro7(?,?,?)}");
13. //4.给?赋值
14. cs.setInt(1,10);
15. cs.setString(2,"笑傲江湖");
16. cs.setString(3,"人民出版社");\*/
18. //看看如何调用有返回值的过程
19. //创建CallableStatement
20. /\*CallableStatement cs = ct.prepareCall("{call sp\_pro8(?,?)}");
22. //给第一个？赋值
23. cs.setInt(1,7788);
24. //给第二个？赋值
25. cs.registerOutParameter(2,oracle.jdbc.OracleTypes.VARCHAR);
27. //5.执行
28. cs.execute();
29. //取出返回值,要注意？的顺序
30. String name=cs.getString(2);
31. System.out.println("7788的名字"+name);
32. } catch(Exception e){
33. e.printStackTrace();
34. } finally{
35. //6.关闭各个打开的资源
36. cs.close();
37. ct.close();
38. }
39. }
40. }

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。

Sql代 码

1. --有输入和输出的存储过程
2. create or replace procedure sp\_pro8
3. (spno in number, spName out varchar2,spSal out number,spJob out varchar2) is
4. begin
5. select ename,sal,job into spName,spSal,spJob from emp where empno=spno;
6. end;
7. /

--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;

/

Java 代码

1. import java.sql.\*;
2. public class Test2{
3. public static void main(String[] args){
5. try{
6. //1.加载驱动
7. Class.forName("oracle.jdbc.driver.OracleDriver");
8. //2.得到连接
9. Connection ct = DriverManager.getConnection("jdbc:oracle:thin@127.0.0.1:1521:MYORA1","scott","m123");
11. //3.创建CallableStatement
12. /\*CallableStatement cs = ct.prepareCall("{call sp\_pro7(?,?,?)}");
13. //4.给?赋值
14. cs.setInt(1,10);
15. cs.setString(2,"笑傲江湖");
16. cs.setString(3,"人民出版社");\*/
18. //看看如何调用有返回值的过程
19. //创建CallableStatement
20. /\*CallableStatement cs = ct.prepareCall("{call sp\_pro8(?,?,?,?)}");
22. //给第一个？赋值
23. cs.setInt(1,7788);
24. //给第二个？赋值
25. cs.registerOutParameter(2,oracle.jdbc.OracleTypes.VARCHAR);
26. //给第三个？赋值
27. cs.registerOutParameter(3,oracle.jdbc.OracleTypes.DOUBLE);
28. //给第四个？赋值
29. cs.registerOutParameter(4,oracle.jdbc.OracleTypes.VARCHAR);
31. //5.执行
32. cs.execute();
33. //取出返回值,要注意？的顺序
34. String name=cs.getString(2);
35. String job=cs.getString(4);
36. System.out.println("7788的名字"+name+" 工 作："+job);
37. } catch(Exception e){
38. e.printStackTrace();
39. } finally{
40. //6.关闭各个打开的资源
41. cs.close();
42. ct.close();
43. }
44. }
45. }

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   
  
ν 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：

Sql代 码

1. create or replace package testpackage as
2. TYPE test\_cursor is ref cursor;
3. end testpackage;

create or replace package testpackage as

TYPE test\_cursor is ref cursor;

end testpackage;

2.建立存储过程。如下：

Sql代 码

1. create or replace procedure sp\_pro9(spNo in number,p\_cursor out testpackage.test\_cursor) is
2. begin
3. open p\_cursor for
4. select \* from emp where deptno = spNo;
5. end sp\_pro9;

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

Java 代码

1. import java.sql.\*;
2. public class Test2{
3. public static void main(String[] args){
5. try{
6. //1.加载驱动
7. Class.forName("oracle.jdbc.driver.OracleDriver");
8. //2.得到连接
9. Connection ct = DriverManager.getConnection("jdbc:oracle:thin@127.0.0.1:1521:MYORA1","scott","m123");
11. //看看如何调用有返回值的过程
12. //3.创建CallableStatement
13. /\*CallableStatement cs = ct.prepareCall("{call sp\_pro9(?,?)}");
15. //4.给第？赋值
16. cs.setInt(1,10);
17. //给第二个？赋值
18. cs.registerOutParameter(2,oracle.jdbc.OracleTypes.CURSOR);
20. //5.执行
21. cs.execute();
22. //得到结果集
23. ResultSet rs=(ResultSet)cs.getObject(2);
24. while(rs.next()){
25. System.out.println(rs.getInt(1)+" "+rs.getString(2));
26. }
27. } catch(Exception e){
28. e.printStackTrace();
29. } finally{
30. //6.关闭各个打开的资源
31. cs.close();
32. ct.close();
33. }
34. }
35. }

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   
  
ν 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

Sql代 码

1. select t1.\*, rownum rn from (select \* from emp) t1 where rownum<=10;
2. --在分页时，大家可以把下面的sql语句当做一个模板使用
3. select \* from
4. (select t1.\*, rownum rn from (select \* from emp) t1 where rownum<=10)
5. where rn>=6;

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：

Sql代 码

1. create or replace package testpackage as
2. TYPE test\_cursor is ref cursor;
3. end testpackage;
4. --开始编写分页的过程
5. create or replace procedure fenye
6. (tableName in varchar2,
7. Pagesize in number,--一页显示记录 数
8. pageNow in number,
9. myrows out number,-- 总记录数
10. myPageCount out number,--总 页数
11. p\_cursor out testpackage.test\_cursor--返回的记录集
12. ) is
13. --定义部分
14. --定义sql语句 字符串
15. v\_sql varchar2(1000);
16. --定义两个整数
17. v\_begin number:=(pageNow-1)\*Pagesize+1;
18. v\_end number:=pageNow\*Pagesize;
19. begin
20. --执行部分
21. v\_sql:='select \* from (select t1.\*, rownum rn from (select \* from '||tableName||') t1 where rownum<='||v\_end||') where rn>='||v\_begin;
22. --把游标和sql关联
23. open p\_cursor for v\_sql;
24. --计算myrows和myPageCount
25. --组织一个sql语句
26. v\_sql:='select count(\*) from '||tableName;
27. --执行sql,并把返回的值，赋给myrows；
28. execute inmediate v\_sql into myrows;
29. --计算myPageCount
30. --if myrows%Pagesize=0 then这样写是错的
31. if mod(myrows,Pagesize)=0 then
32. myPageCount:=myrows/Pagesize;
33. else
34. myPageCount:=myrows/Pagesize+1
35. end if;
36. --关闭游标
37. close p\_cursor;
38. end;
39. /

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

Java 代码

1. import java.sql.\*;
2. public class FenYe{
3. public static void main(String[] args){
5. try{
6. //1.加载驱动
7. Class.forName("oracle.jdbc.driver.OracleDriver");
8. //2.得到连接
9. Connection ct = DriverManager.getConnection("jdbc:oracle:thin@127.0.0.1:1521:MYORA1","scott","m123");
11. //3.创建CallableStatement
12. CallableStatement cs = ct.prepareCall("{call fenye(?,?,?,?,?,?)}");
14. //4.给第？赋值
15. cs.seString(1,"emp");
16. cs.setInt(2,5);
17. cs.setInt(3,2);
19. //注册总记录数
20. cs.registerOutParameter(4,oracle.jdbc.OracleTypes.INTEGER);
21. //注册总页数
22. cs.registerOutParameter(5,oracle.jdbc.OracleTypes.INTEGER);
23. //注册返回的结果集
24. cs.registerOutParameter(6,oracle.jdbc.OracleTypes.CURSOR);
26. //5.执行
27. cs.execute();
29. //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
30. int rowNum=cs.getInt(4);
32. int pageCount = cs.getInt(5);
33. ResultSet rs=(ResultSet)cs.getObject(6);
35. //显示一下，看看对不对
36. System.out.println("rowNum="+rowNum);
37. System.out.println("总页数="+pageCount);
39. while(rs.next()){
40. System.out.println("编号："+rs.getInt(1)+" 名字："+rs.getString(2)+" 工资："+rs.getFloat(6));
41. }
42. } catch(Exception e){
43. e.printStackTrace();
44. } finally{
45. //6.关闭各个打开的资源
46. cs.close();
47. ct.close();
48. }
49. }
50. }

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：

Sql代 码

1. begin
2. --执行部分
3. v\_sql:='select \* from (select t1.\*, rownum rn from (select \* from '||tableName||' order by sal) t1 where rownum<='||v\_end||') where rn>='||v\_begin;

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

# [例外处理](http://taeky.javaeye.com/blog/666144)

**文章分类:**[**数据库**](http://www.javaeye.com/blogs/category/database)

例外处理   
  
ν 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   
  
ν 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？

Sql代 码

1. --例外案例
2. declare
3. --定义
4. v\_ename emp.ename%type;
5. begin
6. --
7. select ename into v\_ename from emp where empno=&gno;
8. dbms\_output.put\_line('名字：'||v\_ename)
9. /

--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：   
  
  
随便输个不存在的编号，回车，会抛出异常，显示：   
ORA-01403: 未找到数据   
ORA-06512: 在line 6

Sql代 码

1. declare
2. --定义
3. v\_ename emp.ename%type;
4. begin
5. --
6. select ename into v\_ename from emp where empno=&gno;
7. dbms\_output.put\_line('名字：'||v\_ename)
8. exception
9. when no\_data\_found then
10. dbms\_output.put\_line('编号没有！');
11. /

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！   
  
ν 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：

Sql代 码

1. create or replace procedure sp\_pro6(spno number) is
2. v\_sal emp.sal%type;
3. begin
4. select sal into v\_sal from emp where empno = spno;
5. case
6. when v\_sal < 1000 then
7. update emp set sal = sal + 100 where empno = spno;
8. when v\_sal < 2000 then
9. update emp set sal = sal + 200 where empno = spno;
10. end case;
11. exception
12. when case\_not\_found then
13. dbms\_output.put\_line('case语句没有与' || v\_sal || '相匹配的条件');
14. end;

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

Sql代 码

1. declare
2. cursor emp\_cursor is select ename, sal from emp;
3. begin
4. open emp\_cursor;
5. for emp\_record1 in emp\_cursor loop
6. dbms\_output.put\_line(emp\_record1.ename);
7. end loop;
8. exception
9. when cursor\_already\_open then
10. dbms\_output.put\_line('游标已经打开');
11. end;
12. /

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

Sql代 码

1. begin
2. insert into dept values (10, '公 关部', '北京');
3. exception
4. when dup\_val\_on\_index then
5. dbms\_output.put\_line('在deptno列上不能出现重复值');
6. end;

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

Sql代 码

1. declare
2. cursor emp\_cursor is select ename, sal from emp;
3. emp\_record emp\_cursor%rowtype;
4. begin
5. --open emp\_cursor; --打开游标
6. fetch emp\_cursor into emp\_record;
7. dbms\_output.put\_line(emp\_record.ename);
8. close emp\_cursor;
9. exception
10. when invalid\_cursor then
11. dbms\_output.put\_line('请检测游标是否打开');
12. end;

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

Sql代 码

1. begin
2. update emp set sal= sal + 'loo';
3. exception
4. when invalid\_number then
5. dbms\_output.put\_line('输入的数字不正确');
6. end;

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

Sql代 码

1. declare
2. v\_sal emp.sal%type;
3. begin
4. select sal into v\_sal from emp
5. when ename='&name';
6. exception
7. when no\_data\_found then
8. dbms\_output.put\_line('不存在该员工')；
9. end;

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。

Sql代 码

1. declare
2. v\_ename emp.ename%type;
3. begin
4. select ename into v\_ename from emp;
5. exception
6. when too\_many\_rows then
7. dbms\_output.put\_line('返回了多行');
8. end;

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 预定义例外 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：

Sql代 码

1. declare
2. v\_ename varchar2(5);
3. begin
4. select ename into v\_ename from emp where empno = &no1;
5. dbms\_output.put\_line(v\_ename);
6. exception
7. when value\_error then
8. dbms\_output.put\_line('变量尺寸不足');
9. end;

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;

ν 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   
  
ν 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。   
ν 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。

Sql代 码

1. --自定义例外
2. create or replace procedure ex\_test(spNo number)
3. is
4. begin
5. --更新用户sal
6. update emp set sal=sal+1000 where empno=spNo;
7. end;
8. /

--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：

Sql代 码

1. --自定义例外
2. create or replace procedure ex\_test(spNo number)
3. is
4. --定义一个例外
5. myex exception;
6. begin
7. --更新用户sal
8. update emp set sal=sal+1000 where empno=spNo;
9. --sql%notfound这是表示没有update
10. --raise myex;触发myex
11. if sql%notfound then
12. raise myex;
13. end if;
14. exception
15. when myex then
16. dbms\_output.put\_line('没有更新任何用户')；
17. end;
18. /

--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;

/

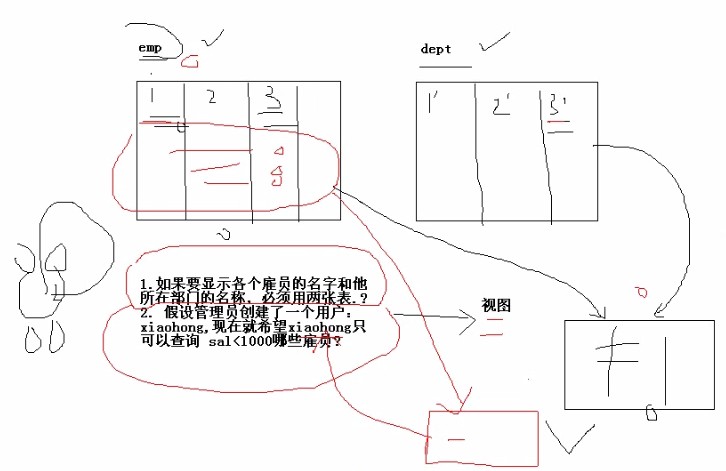
现在再测试一次：   
SQL> exec ex\_test(56);   
没有更新任何用户

# [oracle的视图](http://taeky.javaeye.com/blog/667882)

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：

视图与表的区别

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）

Sql代 码

1. create view myview as select \* from emp where sal<1000;

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

Sql代 码

1. create view myview2 as select emp.empno,emp.ename,dept.dname from emp,dept where emp.deptno=dept.deptno;

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