# Oracle操作

## 一. Oracle基本操作命令:

### 1. 链接Oracle数据库:

(1)本地连接:

Sys用户连接: **sqlplus / as sysdba**

连接切换: conn scott/root

断开连接: disc. 退出连接: exit

**(2)远程连接:**

**第一步: sqlplus/nolog**

**第二步: conn scott/tiger@1092.168.1.1/orcl**

### 启动/停止Oracle的服务:

启动: Net start Oracle ServiceORCL

停止: Net stop Oracle ServiceORCL

### 显示当前登录用户:

Show user;

### 修改密码: passw/password

(1)修改自己的密码: passw/password

(2)sys修改其他用户密码:

alter user 用户名 identified by 新密码；

alter user Scott identified by 123456;

### 将sqlplus屏幕上的内容输入到指定的文件中去:

**(1)****开启输入模式: spool D:/AA.sql;**

**(2)需要输入到文本里的信息(查询, 插入等等).**

**(3) 关闭输入模式: spool off;**

**(会将中间(2)出现的信息全部输出到AA.sql文件)**

### java链接Oracle数据库

**(1)jdbc\_odbc桥链接**

**(2)JDBC链接**

## 二. Oracle用户管理

### 1.创建用户(sys/system): 新创建用户没有任何权限, 连接数据库的权限都没有

create user Seky identified by m1234;

### 2.删除用户(sys/system)

drop user scott [cascade]

(如果删除的用户有创建表, 那么需要带个cascade参数级联删除)

#### 用户无法删除

如果删除用户时提示: 无法删除当前已链接的用户. 则先kill掉session断开连接后, 再删除用户.

具体可如下：

1)查看用户的连接状况

select username, sid, serial# from **v$session** where username is not null

查询结果为

username          sid                serial#

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

NETBNEW          513              22974   
NETBNEW          514              18183   
NETBNEW          516              21573   
NETBNEW          531              9   
WUZHQ             532              4562

2)找到要删除用户的sid,和serial，并删除.

列：kill掉'WUZHQ'的session:

alter system kill session'532,4562';

3)删除用户:

drop user WUZHQ cascade;

如果在drop 后还提示ORA-01940:无法删除当前已链接的用户，说明还有连接的session，可以通过查看session的状态来确定该session是否被kill 了，用如下语句查看：

select saddr, sid, serial#, paddr, username, status from **v$session where username is not null**

### 3.修改密码:

(1)修改自己密码: passw或password

(2)sys/system用户修改其他用户密码:

alter user Scott identified by 123456

### 4.新建用户授权: grant connect to seky;

(1)connect角色: 连接库; (2)dba角色; (3)resource角色:可以创建表空间(即创建表); (4)视图view

### 5.用户授权问题:

#### (1)一个用户的表授权给另一个用户:

##### 1) select权限

scott用户将自己的emp表的select权限授给seky用户:

grant select on emp to seky;

seky用户查找emp时:

select \* from scott.emp

##### 2) insert权限

scott用户将自己的emp表的insert权限授给seky用户:

grant inset on emp to seky;

seky用户插入emp表时:

insert into scott.emp(XXX) values(XXX);

##### 3) all权限

scott用户将自己的emp表的ALL权限授给seky用户:

grant all on emp to seky;(all包含增删改查)

##### 4)其他权限

alter修改 index 索引 references 外键 execute执行

grant index on scott.emp to seky with grant option允许在scott的emp表上建立索引的权限

##### 5)授权列权限

grant select on emp(name) to seky只允许查询name权限

grant update on emp(name, sal) to seky

#### (2)回收授予的权限:

**revoke** all **on** emp **from** seky; (scott用户回收emp表的all权限)

#### (3)权限的传递:

Scott --授权--> seky --授权--> froc

##### 1.系统授权: with admin option

scott用户: grant connect to seky **with admin option**

seky用户: grant connect to froc

(注意: 系统权限scott收回对seky的权限后, froc的权限不会被收回)

##### 2.对象授权: with grant option

Scott用户: grant all on emp to seky **with grant option**

seky用户: grant all on scott.emp to froc

(注意: Scott收回对seky的对象权限后, froc的权限也被收回)

#### (4)创建和管理JOB权限

oracle中创建job是通过调用dbms\_scheduler.create\_job来做. 用户需要能创建job权限,管理job权限:  
1)创建JOB的权限: **create job**  
2)管理JOB的权限(定时执行): **manager scheduler**

相应命令如下：

grant create job to orace\_user1;  
grant manage scheduler to orace\_user1

### 6.profile管理用户口令

Profile是限制口令, 资源限制集合, 当创建数据库时,Oracle会自动创建名称为default的profile, 当创建用户时没有指定profile时,Oracle会将默认的default分配给用户.

#### (1)用户锁定(dba权限)

列: 指定Scott用户最多尝试3次登陆, 3次失败后锁定2天.

* 1. 创建profile口令:

**Create profile** AA **limit failed\_login\_****attempts** 3 **password\_lock\_time** 2

* 1. 赋给用户profile限制:

**alter user** scott **profile**  AA

#### (2)给用户解锁(dba权限)

**alter user** scott **account unlock**

#### (3)终止口令(dba权限)

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.

列: 要求Scott每隔10要修改登陆密码,宽限期为2天.

a)创建profile文件

**create profile AAA limit password\_life\_time 10 password\_grace\_time 2**

b)赋给用户profile限制:

**alter user** scott **profile**  AA

#### (4)口令历史

如果希望用户在修改密码时,不能使用以前的使用过的密码,可以使用口令历史, Oracle就会将修改的信息存放在数据字典中, 这样当用户修改密码时Oracle就会对新旧密码进行比较, 但发现和旧密码一样时,就会修改不了.

a)创建profile文件

**create profile** password\_history **limit password\_file\_time** 10 **password\_grace\_time** 2 **password\_reuse\_time** 10.

**(password\_reuse\_time指令可重用时间即10天后就可以重用)**

b)赋给用户profile限制:

**alter user** scott **profile**  password\_history

#### (5)删除profile

当profile被删除时,对应约束的用户也被解锁.(有的删除不掉)

**drop profile** password\_history **[cascade]**

**(cascade级联把所有的全部删除掉)**

## 三. Oracle的表管理

### 1.Oracle数据类型

#### (1)char()类型:

最长2000字符,

例: char(200) 如果存放数没有200个字符,不足的用空格填补,也占空间200个字符空间.**(char定长查询速度比varchar2快得很多, 是按整个200字符进行比较)**

#### (2)varchar2()类型:

最长4000字符

例: varchar(200) 如果存放数据没有200个字符, 则按实际的长度进行存取.**(varchar2()查询速度比char慢, varchar2查询时是一个一个字符进行比较)**

#### (3)clob类型(character large object)

字符型大对象, 最大4G

#### (4)数字类型

**Number** 范围 10^-38 到 10^38, 可以表示整数, 也可以表示小数

**Number(5, 2)** 表示一个小数一共有5位, 小数有2位; 范围:-999.99 – 999.99

**Number(5)**  表示5位整数; 范围: -99999-99999

#### (5)日期类型

**Date** 精确到年月日时分秒

**Datestamp** 精确到毫秒级, Oracle9i对date的扩展.

#### (6)图片类型

**Blob** 二进制数据, 可以存放图片/声音, 最大4G(一般是存放路径, 安全性要求高的可以存到数库里)

### 2. 表修改

#### (1)表添加字段

alter table tableName add id number;

或

alter table tableName add (id number(6));

#### (2)修改字段长度, 类型, 名字(没有数据时可以修改类型,名字)

alter table student **modify** classid2 number

#### 删除一个表的字段

alter table student **drop column** classid2

#### 修改表名

**rename** student **to**  student2

#### 表删除操作

(a)delete from student

删除所有表数据, 表结构存在,写日志, 可以恢复, 删除速度较慢.

(b)drop table student 连同表结构一起删除.

(c)truncate table student

删除数据, 表结构存在, 不写日志, 数据不能恢复, 熟读快.

#### 限制表修改(重点)

alter table isp\_info read only;

alter table isp\_info read write;

### 3.系统命令

#### (1) 显示操作耗时:

set timing on; 打开查询时间

#### (2)命令窗口设置显示：

set serveroutput on

#### (3)显示CPU线程数: parallel使用

show parameters cpu;

#### (4)修改oracle默认的时间格式: 默认时间格式 DD-MON-YY

alter session set **nls\_date\_format**='yyyy-mm-dd';

## 四.Oracle查询操作

#### 1 多列子查询

select \* from emp e **where (e.mgr, e.job)** = (select mgr, job from emp b where b.empno =7788)

(说明: 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from使用子查询时必须给子查询一个别名)

#### as 别名: 列可以加别名, 子查询时表的别名不能加as .

#### 3 Oracle的分页: 很多种分页方式

(1). 常规的rownum分页

select empno, ename, sal

from (select rownum rown, empno, ename, sal

from (select \* from emp e order by e.sal desc) a

where rownum <= 2 \* 5) a

where a.rown > 1 \* 5;

**(2).使用rowid进行分页: 分页性能比rownum高 3-4倍.**

select empno, ename, sal from emp where rowid in (select rid

from (select rownum rown, rid

from (select rowid rid from emp e order by e.sal desc)

where rownum <= 2 \* 5)a

where rown > 1 \* 5) order by sal desc;

**(3)ROWID的使用——快速删除重复的记录**

ROWID是数据的详细地址，通过rowid，oracle可以快速的定位某行具体的数据的位置。

ROWID可以分为物理rowid和逻辑rowid两种。普通的表中的rowid是物理rowid，索引组织表(IOT)的rowid是逻辑rowid。当表中有大量重复数据时，可以使用ROWID快速删除重复的记录。

**利用rowid结合max或min函数**

使用rowid快速唯一确定重复行结合max或min函数来实现删除重复行。

delete from stua where rowid not in (select max(b.rowid) from stu b where a.no=b.no and a.name= b.name and a.sex = b.sex);

或者用下面的语句

delete from stua where rowid < (select max(b.rowid) from stu b where a.no=b.no and a.name =b.name and a.sex

跟上面的方法思路基本是一样的，不过使用了group by，减少了显性的比较条件，提高效率。

    delete from stu where rowid not in (select max(rowid) from stu t group by t.no, t.name, t.sex);

**删除重复记录的几种方法：**

1. 适用于有大量重复记录的情况(列上建有索引的时候，用以下语句效率会很高)：

    Delete empa Where empno In (Select empno From empa Group By empno HavingCount(\*) > 1) And ROWID Not In (Select Min(ROWID) From empa Group By empno HavingCount(\*) > 1);

  Delete empa Where ROWID Not In(Select Min(ROWID) From empa Group Byempno);

   2、适用于有少量重复记录的情况(注意，对于有大量重复记录的情况，用以下语句效率会很低)：

    Delete empa a where rowid<>(select max(rowid) from empa where empno=a.empno );

#### 4 使用查询结果创建表:

**create table myemp**(empno, ename, sal) **as select** empno, ename, sal from emp;

**常用操作**

**(1)创建表:**

**create table myemp(empno, ename, sal) as select empno, ename, sal from emp where 1 = 2;**

**(2)数据备份:**

**create table emp\_test as select \* from emp;**

#### 5 批量数据插入

**insert into**  myemp(empno, enname, sal) **select** empno, enname, sal from em**p;**

**全字段**

**insert into** emp\_test(select \* from emp);

#### 6 查询更新操作:

**update emp set** (job, sal, comm)=(select job, sal, comm from myemp where empno = 110) where empname = ‘scott’;

#### 7 合并查询

1.union和union all

union合并查询后的结果去重, union all查询的结果不会去重.

2. intersect

取两个查询结果的交集, 相当于两个条件 AND 操作

select \* from emp e where e.sal > 2000

intersect

select \* from emp e where e.ename = 'SCOTT';

3. minus: 差集, A集合包含B集合, 差集为A集合内, B集合外的部分.

select \* from emp e where e.sal > 2000

minus

select \* from emp e where e.ename = 'SCOTT';

## 五.oracle事务操作

### 1.保存点操作

#### (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 ;

**(备注: 可以设置多个保存点, 但是在设置保存点和回滚之间的这段时间操作, 不能提交事务, 只要commit后, 保存点就会不存在, rollback就回滚不了.)**

### 2. 只读事务

当前用户**在一个时间点**设置只读用户后, 这个时间点其他用户对表的DML操作, 该用户看不到这些操作.

**set transcation read only**

## 六. Oracle的函数

### 1.字符函数

#### (1)lower(char): 将字符串转化为小写的格式.

#### (2)upper(char): 将字符串转化为大写的格式.

#### (3)length(char): 返回字符串的长度.

#### (4)substr(截取的字段, start, end): 截取字符串的子字符串.

#### (5)replace(char1, search\_string, replace\_string): 将char1字符中满足search\_string的字母替换成replace\_string.

#### (6)LPAD(string, length, padd\_str)

在string的左边粘贴字符, length是总长, string不够在左边填补padd\_str

select lpad('AA', 4, '\*') from dual 输出为: \*\*AA

#### RLPAD(string, length, padd\_str)

在string的右边粘贴字符, length是总长, string不够在右边填补padd\_str

select rpad('AA', 4, '\*') from dual 输出: AA\*\*

#### 字符串截取: trim() 只截取字符两边的东东

1. select trim('s' from 'ssqwerwes') from dual 输出: 'qwerwe'
2. select trim(' s sqwerwes ') from dual 输出: ‘s sqwerwes’

#### concat(参数1, 参数2)

concat函数等价于 || 将两个字符拼接在一起.Oracle只能有两个参数,mysql可以多个.

### 2.数学函数

#### trunc(n, [m]): 截取函数

如果省掉m, 则截取到整数;

select trunc(2345.748) from dual 为: 2345

如果m是正数, 则截取到小数点的m位后;

select trunc(2345.748, 2) from dual 为: 2345.74

如果m是负数, 则截取到小数点的m为前.

select trunc(2385.748, -2) from dual 为: 2300

#### mod(m, n): 取模函数.

#### (3)floor(n): 返回小于或等于n的最大整数.

#### (4)ceil(n): 返回大于或等于n的最小整数.

#### (5)round(n, [m]): 该函数用于执行四舍五入.

如果省掉m, 这四舍五入到整数;

select round(2345.748) from dual 为: 2346

如果m是正数, 则四舍五入到小数点的m位后;

select round(2345.748, 2) from dual 为: 2345.75

如果m是负数, 则四舍五入到小数点的m为前.

select round(2345.748, -2) from dual 为: 2300

**[备注: 数学函数里有很多函数, 如: 正弦, 余弦, 幂等函数]**

### 3.日期函数

#### (1)add\_months(date, n): 在指定的date时间上加上n个月.

#### last\_day(date): 返回指定日期所在月份的最后一天.

#### to\_char()

1.将日期转换成字符串: to\_char(sysdate, ‘yy-mm-dd’)

2.数字报表(很有用):

select to\_char(23432433.23,‘L999,999,999.99’)from dual

结果为: ￥23,432,433.23

select to\_char(23432433.23,'L000,000,000**.**000')from dual

结果为: ￥023,432,433.230

格式说明:

9: 显示数字, 并忽略前面的0;

0: 显示数字, 如果位数不足, 则用0补齐;

“**.”**: 在指定位置显示小数点;

“,”: 在指定位置显示逗号, 常用于数字的分割;

“$”: 在数字前加美元符;

L: 在数字前面加上本地的货币符号;

C: 在数字前面加上国际货币符号;

G: 在指定位置显示分隔符;

D: 在指定位置显示小数点符号;

[备注: Oracle中两个日期相减得到的是天数]

#### next\_day(date, weekday): date日期的下一个weekday的日期

#### months\_between(f, s) 日期f和s之间相差的月份.

select months\_between(sysdate, trunc(sysdate-45, 'dd')) from dual

输出为: 1.5

### 4.查询常用函数

#### (1) nvl(columnA, 0)

当columnA为null时, 用0表示.

#### (2) nvl2(columnA, expro1, expro2)

nvl2是对nvl()的增强, 当columnA为null时, 用expro1表示, 不为null时用expro2表示.

#### (3) decode函数的两种使用方法:

(1) decode(条件,值1,返回值1,值2,返回值2,...值n,返回值n, 缺省值)

根if .. else判断标准一致

(2) decode(字段或字段的运算，值1，值2，值3）

这个函数运行的结果是，当字段或字段的运算的值等于值1时，该函数返回值2，否则返回值3

当然值1，值2，值3也可以是表达式，这个函数使得某些sql语句简单了许多.

#### (4) case函数:

**(1)--简单Case函数**

CASE sex

WHEN '1' THEN '男'

WHEN '2' THEN '女'

ELSE '其他' END

**(2)--Case搜索函数**

CASE WHEN sex = '1' THEN '男'

WHEN sex = '2' THEN '女'

ELSE '其他' END

#### (5) COALESCE()函数

COALESCE (expression\_1, expression\_2, ...,expression\_n)

依次参考各参数表达式，遇到非null值即停止并返回该值。如果所有的表达式都是空值，最终将返回一个空值。

#### (6) 排序: order by

多列排序:

select \* from emp order by **id desc**, **birthday asc**

#### (7) group by 和 having子句:

group by用于分组, having用于分组查询后的结果进行筛选.

select max(sal) from emp **group by** sal **having max(sal)** > 200

**顺序: group by -> having -> order by**

#### (8)比较运算符: any, all, in

##### (1) any: 任意一个

select e.empno, e.ename from emp e where e.sal > **any** (select sal from emp b where b.deptno = 30);

等价于

select e.empno, e.ename from emp e where e.sal > (select **min(sal)** from emp b where b.deptno = 30);

##### (2)all: 所有的都满足条件

salary > any(…)大于最小的; salary < ANY(…)小于最大的;

salary > all(…)大于最大的; salary < ALL(…)小于最小;

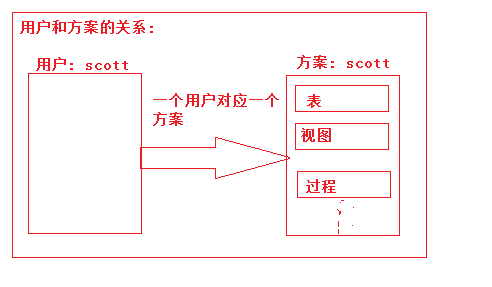
## 七.数据管理

### 1.权限 sysdate > sysoper > dba

sysdate用户具有创建数据库的权限, 其他两个用户没有.

### 2.数据库的导出: 导出表, 导出方案, 导出数据库三种

用户和方案的关系:



导出使用exp命令来完成, 该命令常用的选项:

1. **userid:** 用于指导执行导出操作的用户名, 口令, 连接字符串
2. **tables:** 用于指定导出操作的表.
3. **owner:** 用于指定导出操作的方案.
4. **full=y:** 用于指定导出操作的数据库.
5. **inctype:** 用于指定执行导出操作的增量类型.
6. **rows=n:** 用于指定导出操作是否要导出表中的数据.
7. **file:** 用于指定导出文件名.
8. **direct=y:** 直接导出,用于处理数据量很大的表.

#### 导出表: (导出操作必须是CMD进入到orale的bin目录里, 才能找到exp.exe进行导出)

##### a.导出自己的表:

单张表: exp userid=scott/root@sid tables=(emp) file=D:\emp.dmp

多张表: exp userid=scott/root@sid tables=(emp,dept) file=D:\emp.dmp

[备注:此种方式导出来的dmp文件是二进制文件]

##### b.导出其他方案的表: 如果要导出其他方案的表需要DBA权限

exp userid=system/root@sid tables=(scott.emp) file=D:\emp.dmp

##### c.导出表结构:

exp userid=scott/root@sid tables=(emp) rows=n file=D:\emp.dmp

导出一个用户下的方案: 不导出数据

exp userid=scott/root@sid owner=scott rows=n file=D:\emp.dmp

##### d. 使用直接导出方式: 这种方式比上面的方法要快得多, 导出数据量太大的表.

**exp userid**=scott/root@sid tables=(emp) file=D:\emp.dmp **direct=y**

**(注意:这是需要数据库的字符集合客户端的字符集保持一致, 否则会出错)**

##### e. 从远程服务器上导出表:

**exp userid=plsuser/pls111111@10.156.1.29/plsdb** tables=(business\_audit\_relation) file=D:/business\_audit\_relation.dmp

#### 导出方案(会很慢)

是指使用export工具导出一个方案或多个方案中的所有对象(表, 视图, 索引, 约束…)和数据, 并保存在文件中

##### 导出自己的方案:

exp scott/root@sid owner=scott file=D:\emp.dmp

导出一个用户下的方案**:** **不导出数据**

exp userid=scott/root@sid owner=scott rows=n file=D:\emp.dmp

##### 导出其他方案:

如果要导出其他方案的表需要DBA权限

exp sys/root@sid owner=(sysdate, scott) file=D:\emp.dmp

#### 导出数据库(具备dba权限或exp\_full\_database权限)

exp userid=system/root@ORCL **full=y** **inctype=complete** file=D:/allBase.dmp

### 3.数据库的导入:

导入表, 导入方案, 导入数据库三种

imp导入常用的选项有:

1. **userid:** 用于指导执行导出操作的用户名, 口令, 连接字符串
2. **tables:** 用于指定导出操作的表.
3. **fromuser:** 用于指定源用户.
4. **touser:** 用于指定目标用户.
5. **file:** 用于指定导入文件名.
6. **full=y:** 用于指定导入整个文件.
7. **inctype:** 用于指定执行导入操作的增量类型.
8. **rows=n:** 指定是否导入表的数据.
9. **ignore:** 如果表存在, 则只导入数据.

【注意：导入表的时表有主外键关联时，先要创建外键关联的表，才能导入数据】

#### (1)导入表: (导入操作必须是CMD进入到orale的bin目录里, 才能找到exp.exe进行导入)

##### a.导入到自己的表:

imp userid=scott/root@sid tables=(emp) file=D:/emp.dmp

##### b.导入表到其他用户下:(dba权限)

imp userid=system/root@sid tables=(emp) file=d:/emp.dmp touser=scott

##### c.导入表结构: 只导入表结构而不导入数据

imp userid=scott/root@sid tables=(emp) file=d:/emp.dmp rows=n

imp userid=scott/root@sid owner=scott file=d:/emp.dmp rows=n

##### d.导入数据

如果对象已经存在可以只导入表数据

imp userid=scott/root@sid tables=(emp) file=d:/emp.dmp ignore=y

##### e.导入数据到远程的数据库

**imp userid=plsuser/pls111111@10.156.1.29/plsdb** file=D:/business\_audit\_relation.dmp **full=y**

#### (2)导入方案

##### a.导入自身的方案

imp userid=scott/root@sid file=D:/XX.dmp

##### b.导入其他用户方案(dba权限)

**imp** userid=system/root@sid file=d:/xxx.dmp **fromuser**=system **touser**=scott

#### (3)导入数据库: 默认情况下, 当导入数据库时, 会导入所有对象结构和数据

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

## 八.数据字典和动态性能视图

数据字典: 提供数据库的一些系统信息, 是数据字典基表和数据字典动态视图的集合, 所有者为sys用户; 用户只能在数据字典上进行查询操作,而维护和修改是由系统自己完成.

动态性能视图: 记载了例程启动后的相关信息.

### 1.数据字典视图:

包括三个方面: (1)user\_XXX (2)all\_XXX (3)dba\_XXX

1. **user\_tables**

显示当前用户所拥有的所有表,只返回用户所对应方案的所有表.

如: select \* from user\_tables

1. **all\_tables**

用于返回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他方案的表.

如: select \* from all\_tables

**(3)dba\_tables**

显示所有方案拥有的数据库表.需要dba权限或具有 select any table的权限

#### 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.

##### (1)bda\_users

可以显示所有数据库用户的详细信息.

##### (2)dba\_sys\_privs

数据字典视图, 可以显示用户所具有的系统权限.

##### (3)dba\_tab\_privs

可以显示用户具有的对象权限.

##### (4)dba\_col\_privs

可以显示用户具有的列权限.

##### (5)dba\_role\_privs

查看某个用户具有的角色.

select \* from dba\_role\_privs p where p.grantee = 'SCOTT';

GRANTEE GRANTED\_ROLE ADMIN\_OPTION DEFAULT\_ROLE

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

SCOTT RESOURCE NO YES

SCOTT CONNECT NO YES

说明:

GRANTED\_ROLE表示拥有的角色.

ADMIN\_OPTION表示是否在角色能不能继续派生.

##### (6)Oracle的角色(重点)

查询Oracle有多少角色: **select \* from dba\_roles**

##### (7)oracle中所有的系统权限

select \* from system\_privilege\_map order by name;

##### (8)Oracle中所有对象权限

select distinct privilege from dba\_tab\_privs;

##### (9)查询数据库的表空间

select tablespace\_name from dba\_tablespaces;

##### (10)查看角色里的权限:

###### a.一个角色包含的系统权限

select \* from dba\_sys\_privs where grantee='角色名'

**或者**

select \* from role\_sys\_privs where role='角色名'

列:

select \* from role\_sys\_privs where role='COONNECT'

select \* from dba\_sys\_privs where grantee='COONNECT'; connect要大写

###### b.一个角色包含的对象权限

select \* from dba\_tab\_privs where grantee='角色名';

##### (11)查询一个用户下的角色

select \* from dba\_role\_privs where grantee = ‘SCOTT’;

##### (12)显示当前数据库的全称: SID名

select \* from global\_name;

##### （13）查看表在哪个存储过程中

select \* from user\_dependencies where referenced\_name = 'REJECT\_STAT' --表名

### 2.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的动态性能视图都是以v$开头的]

(1)查询用户的session的连接状态(DBA权限)

select \* from v$session where username is not null

## 九.表空间和数据文件

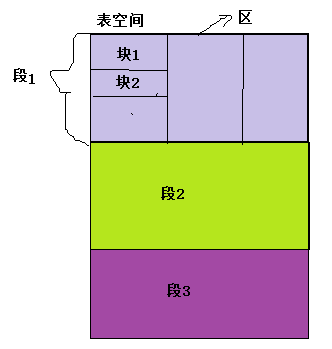
表空间是数据库的逻辑组成部分; 从物理上讲,数据库的数据存放在数据文件中;从逻辑上讲,数据库则是存放在表空间中,表空间是有一个或是多个数据文件组成.

### 1.Oracle存放数据的逻辑结构

Oracle中的逻辑结构包括表空间, 段, 区和块.数据库是由表空间构成,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, 这样可以提高数据库的效率.(表空间是又数据文件构成)

表空间的作用:

DBA将不同的数据存放在不同的位置, 有利于提高i/o性能,同时有利于备份和恢复等.



### 2.表空间管理

#### (1)建立数据表空间

在建立数据库后, 为了便于管理表, 最好建立自己的表空间

create tablespace sp001 datafile ‘D:/sp001.dbf’size 20m uniform size 128k

说明: 执行上述命令后, 会建立名称为sp001的表空间, 并为该表空间建立名称为sp001.dbf的数据文件,区的大小为128K, 表空间的大小为20m

#### (2)表空间的使用

create table emp\_info(deptno number(4)) tablespace sp001

#### (3)改变表空间的状态

建立表空间时,表空间处于联机(online)状态, 此时表空间可以被访问操作,但在系统维护时需要改变表空间的状态.

##### 1)表空间脱机

alter tablespace ‘表空间名’ offline;

脱机后的表空间不能被访问操作

##### 2)表空间联机

alter tablespace ‘表空间名’ online;

##### 3)表空间只读操作

建立表空间时,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 tablesspace ‘表空间名’read only

解除只读:

alter tablesspace ‘表空间名’read write

#### (4)表空间操作

##### 1)查询表空间所包含的表

select \* from all\_tables where tablespace\_name = ‘表空间名’;

##### 2)查询表在哪个表空间

select tablespace\_name, table\_name from user\_tables where table\_name = ‘emp’;

##### 3)删除表空间

drop tablespace ‘表空间名’ including contents and datafiles;

说明:

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

#### (5)扩展表空间

表空间数据增加时,表空间不足, 需要添加更大的存储空间, 有三种方法.

1)增加数据文件

alter tablespace ‘表空间名’ add datafile ‘D:/ sp001.dbf’size 20m

2)增加数据文件的大小

alter tablespace ‘表空间名’‘D:/ sp001.dbf’resize 20m

(注意: 数据文件的大小不能超过500m)

3)设置文件的自增长

alter tablespace ‘表空间名’’D:/sp001.dbf’autoextend on next 10m maxsize 500m;

### 3.数据文件

如果数据文件所在的磁盘损坏时, 需要将该数据文件移动到其他磁盘,然后恢复.

步骤:

1. **确定数据文件所在的表空间**

select tablespace\_name from dba\_data\_files where file\_name = ‘D:/sp001.dbf’;

1. 表空间脱机

alter tablespace sp001 offline;

1. 使用命令移动数据文件到指定的目标位置

host move D:/sp001.dbf C:/ sp001.dbf

1. 执行alter tablspace命令

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

alter tablespace sp001 rename datafile ‘D:/sp001.dbf’ to ‘C:/ sp001.dbf’;

1. 表空间联机

alter tablespace sp001 online;

## 十.数据的完整性

数据的完整性用于确保数据库数据遵从一定的逻辑规则. 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, 在这三种方法中约束易于维护,并且具有最好的性能, 所有作为维护数据完整性的首选.

### 1.约束

#### (1)非空 not null

插入数据时, 必须为列提供数据.(not null只能用在列级定义上)

#### (2)唯一 unique

当定义了唯一约束后, 该列值是不能重复的, 但是可以为null.

#### (3)主键 primary key

用于唯一标示表行数据, 当定义主键约束后, 该列不但不能重复而且不能为null.

**(注意: 一张表最多只能有一个主键, 但是可以有多个****unique约束)**

#### (4)外键 foreign key(表级定义)

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列存在或是null.

constraint foreign\_name foreign key (dept\_id) references dept(id);

**行级定义： references**

#### (5)check

用于强行约束行数据必须满足的条件.

列:创建表

create table customer(

customerId char(8) primary key,

name varchar2(50) not null,

adderss varchar2(50),

email varchar2(50) unique, --email不能重复

sex char(2) default ‘男’check (sex in (‘男’,’女’))--性别只能是男,女;默认为男

);

create table goods(

goodsId char(8) primary key,

unitprice number(10, 2) check (unitprice > 0),--值必须大于0

category varchar2(8),

provider varchar2(30)

)

create table purchase(

customerId char(8) references customer(customerId),--列级定义外键

goodId char(8) references goods(goodsId),--goods表的goodsId列为外键

nums number(10) check(nums between 1 and 30)—nums的范围在1-30之间

)

##### 建表后再建立约束:

1) 为表customer的cardId列添加唯一约束键

alter table customer add constraint unique\_name unique(cardId);

**(alter table 表名add constraint 约束名(随便取) unique(表列))**

2)添加not null约束: 使用**modify**

alter table 表名 **modify** 表列 not null

3)添加check约束

alter table 表名 add constraint 约束名 check(列名 in (1, 3))

#### (6)删除约束

alter table 表名 drop constraint 约束名称

**注:**

**删除主键时, 必须带上cascade选项,解除主从约束关系:**

**alter table 表名 drop primary key cascade**

**或**

**alter table department drop constraint 约束名 cascade;**

**例：**

create table department(

id number primary key,

name varchar2(20) unique,

manager varchar2(30) not null

)

create table employee(

id number primary key,

name char(30),

sex varchar2(2) default '男' check (sex in ('男', '女')),

salary number(10, 2) not null,

deptid number references department(id),

education char(20)

)

删除department的主键时：

alter table department drop constraint SYS\_C009910 cascade;

或者

alter table department drop primary key cascade;

#### (7)列级定义

列级定义在创建表的同时定义约束

如:

create table department(

deptId number(2) constraint pk\_primary primary key,

name varchar2(12)

)

#### (8)表级定义

表级定义指的是在创建表列后,再定义约束. 注意的是: not null约束只能在列级定义上

create table department(

deptId number(2) not null,

name varchar2(12)

)

constraint pk\_primary primary key(deptId);

### 2.索引

索引用于加速数据存取的数据对象,索引的使用可以很大的降低i/o访问的次数,提高性能.按照数据存储方式可以分为B\*树, 反向索引, 位图索引;按照索引的个数分类,可以分为单列索引,复合索引; 等等…

#### 单项索引

单项索引是基于单个列所建的索引.

create index 索引名 on 表名(列名)

#### 复合索引

复合索引是基于多个列所建的索引.

create index emp\_index on emp(name, job);

create index emp\_index on emp(job, name);

上面两个建立的索引不一样, 索引的顺序不一样, 第一个是先对name进行索引, 再对job进行索引.

**sql优化:**

**[SQL执行是按从右到左进行结果集的筛选的,根据这个原理,那些可以过滤掉最大数量记录的条件必须写在WHERE子句的末尾]**

#### 索引缺点

1. 建立索引时,系统要占用大约为表的1.2倍的硬盘和内存空间来保存索引.
2. 更新数据时,系统必须额外的花时间来同时对索引进行更新.

## 十一.角色管理

角色分为预定义角色和自定义角色; 预定义角色是系统自带的角色,自定义角色是用户创建.

### 预定义角色

常见的预定义角色connect, resource, dba.

查看角色里的系统权限:

select \* from dba\_sys\_privs p where p.GRANTEE = 'RESOURCE';

### 自定义角色

创建自定义角色需要DBA权限或是create role权限.建立角色可以设置对角色修改需要密码进行验证; 如果是公用角色,可以采取不验证的方式建立角色.

#### (1)建立角色(不验证)

create role **角色名** **not identified;**

#### (2)建立角色(需要验证)

采用这种方式时, 角色名被修改时,需要提供口令验证.

**create role 角色名 identified by 验证密码;**

### 角色授权

1. 创建角色: create role myRole1 not identified;
2. 角色授权:
   1. grant create session to myRole1 with grant option;
   2. grant select,update,delete,insert on scott.emp to myRole1;

### 分配角色给用户

角色分配需要BDA权限或具有grant any role权限, with admin option用可以将这个角色分配给其他用户.

grant 角色名 to 用户名 **with admin option;**

### 角色删除

需要DBA权限或drop any role的权限; 角色被删除对应赋给用户的权限被删除.

drop role 角色名;

### 角色的信息

#### (1)显示所有的角色

select \* from dba\_roles;

#### (2)角色所具有的权限

##### 1)系统权限

select \* from dba\_sys\_privs where grantee='角色名';

或者

select \* from role\_sys\_privs where role='角色名';

##### 2)对象权限

select \* from dba\_tab\_privs where grantee='角色名';

#### (3)用户具有的角色

select \* from dba\_role\_privs where grantee = ‘SCOTT’;

## 十二.pl/sql编程

### 1.pl/sql简介

pl/sql(procedural language/sql)是Oracle在标准的sql语句上的扩展.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,允许使用列外处理各种错误,功能很强大.

### 2.pl/sql编程概述

#### (1)优点

1)提高应用程序的运行性能.

2)模块化设计思想.(分页, 订单…)

3)减少网络传输量, 提高安全性.

#### (2)缺点

移植性太差, 当数据局库被更换时, 所有的pl/sql编写的过程全要重写。

#### (3). pl/sql编程的分类



上面是pl/sql编程最总要的四个编程

#### (4)pl/sql编程规范

##### 1)注释

a)单行注释: --

b)多行注释: /\*\*/

##### 2)标识符命名规范

a)变量定义: 以**v\_开头**, 如: v\_sal

b)常量定义: 以**c\_开头,** 如: c\_rate

c)游标定义: 以\_**cursor结尾**, 如: emp\_cursor

d)例外(exception异常定义)定义: 以**e\_开头**, 如: e\_error

### 3.过程的调用

(1)exec 过程名(参数1, 参数2, 参数3, …)

**exec my\_procedure(name varchar2, id number);**

(2)all 过程名(参数1, 参数2, 参数3, …)

**call my\_procedure();**

**区别：**

1. 但是exec是sqlplus命令，只能在sqlplus中使用；call为SQL命令，没有限制.

2. 存储过程没有参数时,exec可以直接跟过程名（可以省略()），但call则必须带上().

[注意: 过程和job在命令使用时, 需要/结尾来标识执行]

### 4.异常exception

#### (1)参数输入/注入

declare

v\_name varchar2(100);

v\_sal number;

begin

select e.ename into v\_name from emp e where e.empno = **&aa**;

select e.ename, e.sal into v\_name, v\_sal from emp e where e.empno = **&aa**;

dbms\_output.put\_line(v\_name);

end;

**&: 表示从控制台输入变量aa的值;**

**多个变量注入:**

select e.ename, e.sal into v\_name, v\_sal from emp e where e.empno = 1;

#### (2)异常捕获exception

declare

v\_name varchar2(100);

begin

dbms\_output.put\_line('正常语句1...');

dbms\_output.put\_line(1/0); *--此处的例外下面的exception捕获不了,exception只能捕获自己所在块里前面的异常*

begin

select e.ename into v\_name from emp e where e.empno = 11;

dbms\_output.put\_line(v\_name);

dbms\_output.put\_line('正常语句2...');

exception when others then

dbms\_output.put\_line('异常: ' **||****SQLERRM);--SQLERRM异常信息**

dbms\_output.put\_line(‘11111’);**--如果不抛异常,同一块中exception后面的代码不会执行.**

end;

*--上面抛出异常,不影响下面的执行*

dbms\_output.put\_line('继续执行...');

end;

[例外异常捕获: exception捕获异常只能捕获自己所在块里前面的异常]

### 5.过程

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),通过过程中的参数输入,可以将数据传递到执行部分;通过使用参数的输出,可以将数据传递到应用程序中.

#### (1)过程带输入参数

##### 1)创建procedure

---in表示输入的变量，不写默认是in；out表示输出一个参数。

create or replace procedure my\_procedure(v\_name in varchar2, v\_sal in number) **is/as**

begin

update emp e set e.sal = v\_sal where e.ename = v\_name;

commit;

end;

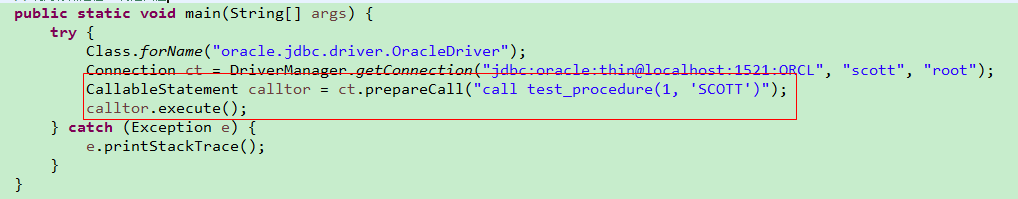
##### 2) sql/sql developer中调用

**call** my\_procedure(‘SCOTT’, 50000);

或

**exec** my\_procedure(‘SCOTT’, 50000);

##### 3) java程序中调用



#### （2）过程带输入，输出参数

##### 1）过程创建

create or replace procedure swap(value1 in out number, value2 in out number) **is/as**

begin

value1 := value1 + value2;

value2 := value1 - value2;

value1 := value1 - value2;

end;

参数既是输入, 又是输出参数

##### 2)调用

declare

a number := 22;

b number := 33;

begin

dbms\_output.put\_line('Before swap: a = ' || a || ', b = ' || b);

swap(a, b);

dbms\_output.put\_line('After swap: a = ' || a || ', b = ' || b);

end;

### 6.函数

创建函数时,在函数的头部必须包含return子句, 而在函数体内必须包含return语句返回数据。

#### （1）函数的创建

create or replace function sp\_func(empName varchar2) return number **is/as** yearSal number(7, 2);

*--return number是返回类型，number(7, 2)是真正返回类型*

begin

select sal\*12 + nvl(comm, 0)\*12 into yearSal from emp where ename=empName;

return yearSal;*--函数返回值*

end;

#### （2）函数的调用

##### 1）pl/sql中调用

declare

yearSal number;

begin

yearSal := emp\_function(‘SCOTT’);

dbms\_output.put\_line(yearSal);

end;

##### 2）java中调用

select SP\_FUNC('SCOTT') from dual；返回后在result.getInt（1）得到返回值。

#### (3)记录类型为返回值: ROWTYPE

##### 1)创建函数

*--使用纪录类型作为返回类型*

CREATE OR REPLACE FUNCTION get\_info(eno NUMBER) RETURN emp%**ROWTYPE**

IS emp\_record emp%**ROWTYPE;**

BEGIN

SELECT \* INTO emp\_record FROM emp e WHERE e.empno=eno;

RETURN emp\_record;

dbms\_output.put\_line('姓名:'||emp\_record.ename||',部门号:'||emp\_record.deptno);

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.put\_line('ERROR: ' || SQLERRM);

END;

##### 2)调用函数

DECLARE

emp\_record emp%ROWTYPE;

BEGIN

emp\_record := get\_info(&a);

dbms\_output.put\_line('姓名:'||emp\_record.ename||',部门号:'||emp\_record.deptno);

END;

### 7.包

用于在逻辑上组合过程和函数，他由**包规范**和**包体**两部分组成。

#### （1）创建包规范：create package

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实体的具体实现，包体用于具体实现。

*--包规范*

create or replace package emp\_package is

procedure emp\_procedure(empId in number, yearSal out number);

function emp\_function(empId number) return number;

end;

#### （2）创建包体：create package body

##### 1）创建包体

注意： 包体创建名必须和包规范名一样，包体里的函数，过程的实现必须和包规范里的一致，包括参数名称。不然会编译出错。

*--包体创建*

create or replace package body emp\_package is

*--包体过程: 过程名，参数名必须和包规范定义的一样,且empId参数不能和表列名empno一样*

procedure emp\_procedure(empId in number, yearSal out number) is

begin

select (e.sal + e.mgr)\*12 into yearSal from emp e where e.empno = empId;

end;

*--包体函数：函数名，参数名必须和包规范定义的一样，且empId参数不能和表列名empId一样*

function emp\_function(empId number) return number is yearSal number(10, 2);

begin

select (e.sal + e.mgr)\*12 into yearSal from emp e where e.empno = empId;

return yearSal;

end;

end;

**【注意：上面过程和函数的输入参数名不能和表中的列名一样，要不样会出错，Oracle不能识别参数和表列名】**

##### 2）调用包体里的过程和函数

（a）pl/sql developer调用

*--调用*

declare

v\_yearSal number;

begin

*--过程调用*

emp\_package.emp\_procedure(7369, v\_yearSal);

dbms\_output.put\_line('年薪：' || v\_yearSal);

*--函数调用*

v\_yearSal := emp\_package.emp\_function(7369);

dbms\_output.put\_line('年薪：' || v\_yearSal);

end;

（b）java中调用

select emp\_package.emp\_function(7369) from dual

### 8.pl/sql变量类型

#### （1）标量类型(scalar)

块编程时，前面定义的变量类型。标量类型赋值时使用 :=

v\_name varchar2(10)

c\_salary number(7, 2) := 1689.6

v\_true boolean is not null default false

##### 1）标量：使用%type类型

declare

v\_name emp.ename%type;

salary emp.sal%type;

begin

XXXXXXXXXXXXXXXXXXX

end;

#### （2）符合类型（composite）

符合类型类似于高级语言中的结构体（java中的类）。用于记录成员变量。

##### 1)复合类型之记录类型：record

记录类型包括: (a) %type (b)%rowtype

###### (a) %type

变量应该拥有与表列相同的类型,将变量类型和列的类型绑定。对应一个字段。

declare

v\_name emp.ename%type;

*--定义一个记录类型：emp\_record*

type emp\_record is record(

name emp.ename%type,

salary emp.sal%type,

dmpno emp.deptno%type

);

*--定义一个emp\_record 类型的变量*

sql\_emp emp\_record;

begin

*--into的参数类型顺序要和记录类型一致*

select e.ename, e.sal, e.deptno into sql\_emp from emp e where e.ename = 'SCOTT';

dbms\_output.put\_line('姓名：'||sql\_emp.name);

dbms\_output.put\_line('薪水：'||sql\_emp.salary);

end;

**【注意：参数into到记录类型时，其顺序要和定义的记录类型的参数列表一致，要不然会乱位】**

###### (b) %rowtype

变量应该拥有与表列相同的类型，对应多个字段。

DECLARE

emp\_record emp%ROWTYPE;

BEGIN

select \* into emp\_record from emp e where e.ename='SCOTT';

dbms\_output.put\_line('姓名：' || emp\_record.ename);

END;

【注意：记录中字段应该和查询结果列表中的字段相匹配。】

##### 2)复合类型之表：table

相当于高级语言的数组，下标可以为负数，且表元素的下标没有限制。

index by binary\_integer 表示下标是整数；table.count总数;table.first开始下标位置;table.last结束下标位置.

declare

*--定义一个标记类型*

type v\_record is record(

empname emp.ename%type,

sal emp.sal%type

);

*--声明一个标记变量*

emp\_record v\_record;

*--定义一个表类型*

type v\_table is table of v\_record;*--声明table里是记录类型变量*

*--type v\_table is table of v\_record index by binary\_integer;*

*--声明一个表变量*

emp\_table v\_table;

begin

select e.ename, e.sal **bulk collect** into emp\_table from emp e where e.deptno = 10;

*--table总数*

dbms\_output.put\_line(emp\_table.count);

for i in emp\_table**.first** .. emp\_table.last loop

dbms\_output.put\_line(emp\_table(i).empname ||' '||emp\_table(i).sal);

end loop;

end;

#### （3）参照变量类型（reference）

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的对象，从而降低占用的空间。参照变量分游标变量（ref cursor）和对象类型变量（ref obj\_type）。

##### 1）游标变量

使用游标时，定义游标时不需要指定相应的select语句，但使用游标时（open时）需要指定select语句和游标结合。

###### (1)分开操作

定义, 声明,与select绑定, 分开定义.

declare

*--声明一个游标类型sq\_cursor*

type sq\_cursor is ref cursor;

*--定义游标变量*

emp\_cursor sq\_cursor;

*--声明一个标记类型*

type sq\_record is record(

name emp.ename%type,

salary emp.sal%type

);

*--定义一个标记类型*

emp\_record sq\_record;

begin

*--把游标和select结合*

open emp\_cursor for select e.ename, e.sal from emp e where e.deptno=&deptno;

loop

*--从游标中取出数据 fetch*

fetch emp\_cursor into emp\_record;

*--当游标nofound时，退出循环*

exit when emp\_cursor%notfound;

dbms\_output.put\_line('姓名：'||emp\_record.name);

dbms\_output.put\_line('薪水：'||emp\_record.salary);

end loop;

*--关闭游标*

close emp\_cursor;

end;

###### (2)同时声明,定义绑定select

游标使用时可以将声明,定义和select语句绑定一起操作.

declare

*--声明,定义,绑定select语句*

cursor emp\_cursor is select e.ename, e.sal from emp e where e.deptno = 10;

begin

*--游标循环*

for r in emp\_cursor loop

dbms\_output.put\_line(r.sal);

end loop;

close emp\_cursor;

end;

##### 2)对象类型变量（ref obj\_type）

很少使用。

#### （4）lob（large object）

很少使用。

### 9.控制结构

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

**【注意：Oracle中的不等于表示有三种：<>, !=, ^=】**

*--循环体*

create or replace procedure add\_subsidy(v\_name varchar2)is

v\_subsidy number;

begin

select e.comm into v\_subsidy from emp e where e.ename=v\_name;

if v\_subsidy <> 0 then

update emp e set e.comm = 200 where e.ename=v\_name;

elsif v\_name='SCOTT' then

update emp e set e.comm = e.comm + 10 where e.ename=v\_name;

else

update emp e set e.comm = e.comm + 1000 where e.ename=v\_name;

end if;

commit;

end;

### 10.循环语句

pl/sql的循环结构包括两种：1.loop----end loop结构；2.while....loop-----end loop结构。3.for循环语句。

#### （1）loop结构

基本的loop循环至少得进入loop里执行一次； loop结构退出使用 exit then 条件。

declare

v\_number number := 1;

begin

loop

*--退出条件*

exit when v\_number = 10;

insert into user\_info(id, name) values(v\_number, 'seky');

v\_number := v\_number +1;

end loop;

commit;

end;

#### （2）while结构

*--while循环*

declare

v\_number number :=1;

begin

while v\_number <= 10 loop

insert into user\_info(id, name) values(v\_number, 'seky');

v\_number := v\_number +1;

end loop;

commit;

end;

#### （3）for循环结构

*--for循环结构*

declare

begin

for r in (select \*from emp) loop

dbms\_output.put\_line('姓名：'||r.ename);

end loop;

end;

*--数字控制*

declare

v\_number number := 10;

begin

for i in 0..v\_number loop

dbms\_output.put\_line(i);

end loop;

end;

### 11.顺序控制语句

顺序控制语句包括1.goto语句，用于跳转到指定的标号去执行语句，可读性较差。2.null语句，什么都不干只是为了增强可读性。

#### （1）顺序控制语句goto

*--goto语句*

declare

v\_number number:=1;

begin

loop

dbms\_output.put\_line(v\_number);

v\_number := v\_number+1;

if v\_number =10 then

*--跳出循环语句*

goto goto\_tag;

end if;

end loop;

*--goto的标记*

<<goto\_tag>>

dbms\_output.put\_line('循环结束！！');

end;

#### （2）顺序控制语句null

*--null语句*

create or replace procedure null\_proced(v\_name varchar2) is

v\_number number;

begin

select e.sal into v\_number from emp e where e.ename = v\_name;

if v\_number < 200 then

dbms\_output.put\_line(v\_number);

else

null;

end if;

end;

### 12.存储过程java中使用

有返回参数或返回一个结果集的存储过程的编写, 在java中调用,结果集中取参数.

#### (1)有返回值(非列表)过程调用

##### 1)存储过程

*--存储过程有返回值: in表示输入值, out表示输出值, 不写默认是in*

create or replace procedure re\_parameter\_pro(v\_name in varchar2, v\_sal out number, v\_job out varchar2) is

begin

select e.sal, e.job into v\_sal, v\_job from emp e where e.ename = v\_name;

end;

##### 2)java中调用



#### (2)有返回值的存储过程(列表[结果集])

由于Oracle过程没有返回值,它所返回的值都是通过out参数来代替,列表也不例外,但由于是集合不能用一般的参数,必须使用package,所有过程分两部分。

##### 1)存储过程

###### a)包的创建

*--创建一个包,包内定义一个游标*

create or replace package re\_collection\_package as

type test\_cursor is ref cursor;

end re\_collection\_package;

###### b)过程创建

*--过程创建: 输出参数是一个游标类型的*

create or replace procedure re\_collection\_procedure(v\_depno in number, re\_cursor out re\_collection\_package.test\_cursor) is

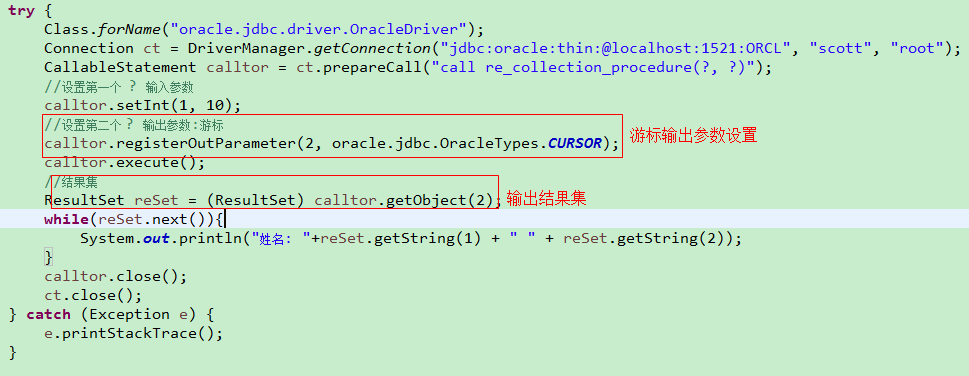
begin

open re\_cursor for

select e.ename, e.job from emp e where e.deptno = v\_depno;

end;

##### 2)java中调用



### 13.视图(view)

视图是一个虚拟表,其内容由查询定义,视图不在数据库中以存储的数据值的形式存在,其查询数据是在引用视图时动态生成的。一般比较复杂的sql语句，常用的使用视图查询。

#### 创建视图

##### 1)授权: 具有创建视图的权限

grant create view to scott

##### 2)创建视图

create or replace view empview as

select e.ename, d.dname, e.job, e.hiredate from emp e

left join dept d on e.deptno = d.deptno;

#### 视图删除

drop view 视图名

### 14.动态执行语句(execute immediate)

解析并马上执行动态的SQL语句或非运行时创建的PL/SQL块.动态创建和执行SQL语句性能超前，EXECUTEIMMEDIATE的目标在于减小企业费用并获得较高的性能.

#### 运行DDL语句

DDL语句包括DML,CREATE, ALTER, DROP等语句

declare

v\_sql varchar(400);

begin

v\_sql := 'update emp e set e.mgr = 200 where e.deptno = 10';

begin

execute immediate v\_sql;

commit;

exception when others then

rollback;

dbms\_output.put\_line(sqlerrm);

end;

end;

#### 从动态语句检索值：检索单个值(INTO子句)

declare

type v\_record is record(

empname emp.ename%type,

sal emp.sal%type

);

emp\_record v\_record;

v\_sql varchar2(400);

begin

v\_sql := 'select e.ename, e.sal from emp e where e.empno = 7788';

*--执行SQL语句*

execute immediate v\_sql into emp\_record;

dbms\_output.put\_line('姓名: '||emp\_record.empname);

end;

#### 从动态语句检索值：批量检索结果

execute immediate 和 bulk collect结合使用.

declare

v\_sql varchar2(400);

*--定义记录类型*

type v\_record is record(

emname emp.ename%type,

sal emp.sal%type

);

*--声明记录类型变量*

emp\_record v\_record;

*--定义表类型*

type v\_table is table of v\_record;*----声明记录类型变量*

emp\_table v\_table;

begin

v\_sql := 'select e.ename, e.sal from emp e where e.deptno = 10';

execute immediate v\_sql **bulk collect** into emp\_table;

dbms\_output.put\_line(emp\_table.count);

end;

#### 动态语句传值(USING 子句)

使用using向动态执行语句输入参数.

##### 1）执行过程

create or replace procedure procedure\_emp(empnumber in number, yearsal out number) is

begin

select 12 \* (e.sal + e.mgr) into yearsal from emp e where e.empno = empnumber;

end;

*--使用动态执行调用过程*

declare

v\_sql varchar2(400);

yearsal number(10, 2);

begin

*--动态执行语句输入的参数用 ‘:参数名’ 表示*

v\_sql := 'begin procedure\_emp(:1, :2); end;';

*--默认为in，in可省，out不可省，可以多个输入输出参数*

execute immediate v\_sql **using in** 7788, **out** yearsal;

dbms\_output.put\_line(yearsal);

end;

##### 2）执行普通的语句

【注意： 传递并检索值时，INTO子句用在USING子句前】

declare

type v\_record is record(

empname emp.ename%type,

sal emp.sal%type

);

emp\_record v\_record;

v\_sql varchar2(400);

begin

v\_sql := 'select e.ename, e.sal from emp e where e.deptno =: 1 and e.ename =: 2';

*--执行SQL语句，多个输入输出参数*

execute immediate v\_sql **into** emp\_record **using in** 20, **in** 'SMITH';

dbms\_output.put\_line('姓名: '||emp\_record.empname);

dbms\_output.put\_line('姓名: '||emp\_record.sal);

end;

#### 多个动态语句：使用begin…end隔开

declare

type v\_record is record(

empname emp.ename%type,

sal emp.sal%type

);

emp\_record v\_record;

v\_sql1 varchar2(400);

v\_sql2 varchar2(300);

ename varchar2(10);

begin

**begin**

v\_sql1 := 'select e.ename, e.sal from emp e where e.deptno =: 1 and e.ename =: 2';

*--执行SQL语句*

execute immediate v\_sql1 into emp\_record using in 20, in 'SMITH';

dbms\_output.put\_line('姓名: '||emp\_record.empname);

dbms\_output.put\_line('工资: '||emp\_record.sal);

**end;**

**begin**

v\_sql2 := 'select e.ename from emp e where e.empno = 7788';

execute immediate v\_sql2 into ename;

dbms\_output.put\_line(ename);

exception when others then

dbms\_output.put\_line(sqlerrm);

**end;**

end;

## 十三.Oracle子句

### 1.retuning子句

RETURNING通常结合DML 语句使用, INSERT: 返回的是添加后的值;UPDATE：返回时更新后的值; DELETE：返回删除前的值.

declare

sal number;

emname varchar2(100);

begin

*--delete返回删除前的值*

delete emp e where e.empno = 7935 **returning** e.sal, e.ename into sal, emname;

commit;

dbms\_output.put\_line(sal);

dbms\_output.put\_line(emname);

end;

### 2. BULK COLLECT 子句

BULK COLLECT 子句会批量检索结果，即一次性将结果集绑定到一个集合变量中，并从SQL引擎发送到PL/SQL引擎。通常可以在SELECT INTO、FETCH INTO以及RETURNING INTO子句中使用BULK COLLECT。

##### (1)SELECT INTO中使用

declare

*--定义表类型*

type v\_table is table of emp%rowtype;

emp\_table v\_table;

begin

select \* **bulk collect into** emp\_table from emp e where e.deptno = 10;

dbms\_output.put\_line(emp\_table.count);

end;

##### (2)FETCH INTO中使用

declare

cursor v\_cursor is select \* from emp e where e.deptno = 10;

type v\_table is table of emp%rowtype;

emp\_table v\_table;

begin

open v\_cursor;

*--fetch批量检索结果*

fetch v\_cursor **bulk collect into** emp\_table;

dbms\_output.put\_line(emp\_table.count);

close v\_cursor;

end;

##### (3)RETURNING INTO中使用

declare

v\_record emp%rowtype;

type v\_nametable is table of emp.ename%type;

nameList v\_nametable;

type v\_saltable is table of emp.sal%type;

salList v\_saltable;

begin

*--每列检索成一个table*

delete emp e where e.deptno = 10 **returning** e.ename, e.sal **bulk collect into** nameList, salList;

dbms\_output.put\_line(nameList.count);

dbms\_output.put\_line(salList.count);

end;

### 3. EXIT、RETURN、CONTINUE子句

return相当于java中的return, 直接跳出存储过程或者函数; exit相当于java中的break结束自己所在的循环体;continue结束本次循环进入下次循环.

### 4.connect by子句

常见的，connect by会在**构造序列**的时候使用

select rownum from dual connect by rownum<10

代替早期版本的

select rownum from all\_objects where rownum <10

### 5.distinct和row\_number() over()区别

#### (1)distinct

用于单个字段和多个字段的去重,多字段去重的是所有该字段值重复的数据进行去重.

#### row\_number() over()

row\_number() 用于给数据库表中的记录进行标号，在使用的时候，其后还跟着一个函数 over()，而函数 over() 的作用是将表中的记录进行分组和排序.

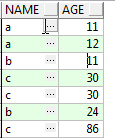
使用: ROW\_NUMBER() OVER(PARTITION BY COLUMN1 ORDER BY COLUMN2)

* PARTITION BY：表示分组,可以对多个字段进行分组.
* ORDER BY：表示排序

例如:

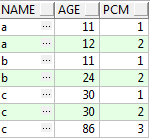
select \* from test\_table

查询结果:



select f.name, f.age, row\_number() over(**partition by** f.name **order by** f.age) rn

from test\_table f



从上面的结果可以看出，其在原表的基础上，多了一列标有数字排序的列;按照name进行分组,age进行排序.可以用标记号筛选出第一条数据:

select \* from (select f.name, f.age, **row\_number() over(partition by f.name order by f.age) rn**

from test\_table f)

**where rn = 1**