**Oracle 11g学习**

# 1.1 Oracle的安装卸载与启动

**Oracle有点：性能优越，大型数据库中的典范**

**Oracle是对象关系型的数据库管理系统，应用非常广泛，可移植性好，支持多用户、大事务量的事务处理、跨平台、10g还能传输表空间**

**大型数据库：sybase、Oracle、db2**

**中型数据库：mysql、SqlServer、infomix**

**小型数据库：Access、Visual Foxpro**

**Oracle数据库在数据安全性和数据完整性上性能都非常优越，完整性包括准确性和有效性，数据完整性包括实体完整性（主键非空）、参照完整性（主键外键的参照关系要准确）、用户自定义的完整性（）**

**Oracle数据库基于客户端/服务器**

## 卸载Oracle

1.启动Universal Installer，把所以内容都勾选，包括所有子项

2.删除注册列表

①HKEY\_LOCAL\_MACHINE\SOFTWARE\Oracle

②HKEY\_LOCAL\_MACHINE\SYSTEM\CurentControlSet\Service

\Eventlog\Application\下的oracle项

③HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\下的oracle项

④HKEY\_LOCAL\_MACHINE\SYSTEM\ControlSet001\Services\下的oracle项

⑤HKEY\_LOCAL\_MACHINE\SYSTEM\ControlSet002\Services\下的oracle项

3.删除环境变量中的Oracle对应的path地址，

4.删除安装目录和c盘下的Oracle文件夹

## Oracle服务

OracleDBConsoleorcl 控制台服务

OracleJobSchedulerORCL 定时器服务

OracleOraDb11g\_home1TNSListener 监听服务

OracleServiceORCL ORCL服务 (ORCL是默认全局数据库名)

cmd要以管理员运行

启动/停止监听服务lsnrctl start/shop

启动/停止ORCL服务net start/stop oracleserviceorcl

**访问数据库方式**

**①通过sqlplus**

登录sqlplus scott/scott@orcl

或者先链接到sqlplus：sqlplus/nolog

再登录conn scott/scott@orcl

退出quit

**②通过EM**

开启OracleDBConsoleorcl 服务

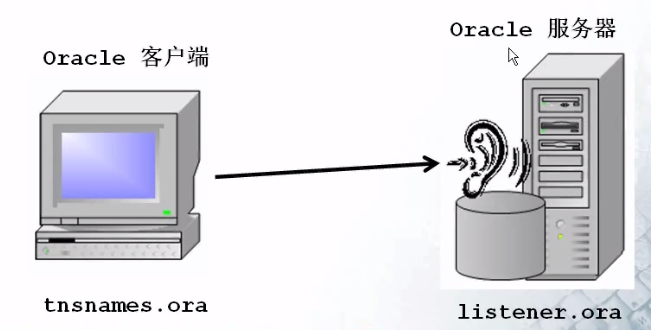
在远程IE的地址输入https://计算机名称:端口/em

**③通过SQL Developer**

**④通过PL Sql/developer**

**⑤通过TOAD**

**Oracle网络配置**



# 1.2常用SQL plus语句

connect切换用户可以简写为conn

|  |
| --- |
| conn scott/tiger@orcl切换到scott |

|  |
| --- |
| show user 显示目前的用户 |

|  |
| --- |
| clear scree 清空屏幕 |

|  |
| --- |
| Spool d:\12.txt  ….  Spool off 把命令和输出的结果保存到12.txt |

|  |
| --- |
| Edit 可以简写为edi对最后一次的sql语句进行修改 |

|  |
| --- |
| Set time on/off 把时间显示/不显示在左边 |

|  |
| --- |
| Show error 显示上次sql语句的错误 |

|  |
| --- |
| Host + doc命令 在sqlplus调用操作系统的命令 |

|  |
| --- |
| @d:\1.sql 执行sql脚本语言 |

# 2.1基本SQL语句和函数

**SQL支持下列类别的命令**

数据定义语言DDL 包括创建表(create)、修改表(alter)、删除表(drop)

数据操作语言DML 包括数据增(insert)删(delete)查(select)改(update)

事务控制语言TCL commit savepoint rollback

数据控制语言DCL grant revoke

**数据类型**

**字符**：char（长度固定1-2000）、varchar2（可变长度1-4000）、long（可变长度2G）

**数值**：number可以存储整数、浮点数和实数，精度是38位，范围是-1038到1038，

number(p,s)中p表示精度，S表示小数点的位数

**日期时间**：date日期+时间，精确到整个秒

|  |
| --- |
| SQL> select sysdate from dual;  SYSDATE  --------------  25-2月 -16  SQL> select to\_char(sysdate,'yyyy-mm-dd hh24:mi:ss') from dual;  TO\_CHAR(SYSDATE,'YY  -------------------  2016-02-25 15:59:47 |

：timestamp 日期+时间+区域信息，精确到秒值得后6位

|  |
| --- |
| SQL> select to\_char(systimestamp,'yyyy-mm-dd hh24:mi:ssxff6') from dual ;  TO\_CHAR(SYSTIMESTAMP,'YYYY-MM  -----------------------------  2016-02-25 16:01:24.273000 |

**RAW/LONG RAW**二进制的存储：

LOB：clob能存储大量字符数据

：blob能存放较大的二进制对象如图形、音频、视频（考虑安全才用）

：bfile用于将二进制数据存储在数据库外部的操作系统文件中

**伪列rowid和rownum**

**rowid是表中行的存储地址，他可以位唯一地标识数据库中的一行，可以使用rowid伪列快速地定位表中的一行**

**rownum是查询返回的结果集中行的序号，可以使用它来限制查询返回的行数（用于分页）**

**伪表dual**

**SQL语句的执行顺序**

**先from， 然后where， 最后select**

**完整的是from→where→group by →having→select→order by**

**exists 用来判断查询所得的结果中，是否有满足条件的记录存在**

**聚合函数一定要放在select后面；**

**case （计算式子结果（可以跟下面判断语句写在一起））**

**when 判断语句 then 前面如果true，然后返回此值**

**when 判断语句 then 前面如果true，然后返回此值**

**else 都不满足就返回此值**

**end 结果的别名**

**select sno,km,score,row\_number() over(order by score desc) from grade;**

**排序编号问题**

**row\_number()排名一样时，也要分不同名次 1 2 3 4 5 6**

**rank()排名一样时，名次一样，站用名次位置 1 2 2 4 5 6**

**dense\_rank()排名一样时，名次一样，不站用名次位置1 2 2 3 4 5**

**decode(字段名,字段值1,返回值1,字段值2，返回值2)**

**decode(字段名,字段值1,返回值1,返回值2)**

**decode(计算式子,字段值1,返回值1,返回值2)**

**行列转换**

|  |  |  |
| --- | --- | --- |
| **商品名称goods\_name** | **季度quarter** | **销售额saleroom** |
| **电视机** | **01** | **100** |
| **电视机** | **02** | **200** |
| **电视机** | **03** | **300** |
| **空调** | **01** | **50** |
| **空调** | **02** | **150** |
| **空调** | **03** | **180** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **商品名称** | **第一季度** | **第二季度** | **第三季度** | **第四季度** |
| **电视机** | **100** | **200** | **300** | **0** |
| **空调** | **50** | **150** | **180** | **0** |

**select a.goods\_name ,**

**sum(decode(a.quarter,’01’,a.** **saleroom,0)) 1st\_quarter,**

**sum(decode(a.quarter,’02’,a.** **saleroom,0)) 2nd\_quarter,**

**sum(decode(a.quarter,’03’,a.** **saleroom,0)) 3ly\_quarter,**

**sum(decode(a.quarter,’04’,a.** **saleroom,0)) 4th\_quarter**

**from t\_sell a**

**group by a.goods\_name;**

**分页显示rownum**

**select \* from (select yggz.\*,rownum rn from yggz) where rn<=5 and rn>=3;**

**rowid里的信息有：数据库对象号、数据文件号、数据块号、行号**

**唯一标识一行**

**删除重复记录**

**t\_table**

|  |  |
| --- | --- |
| **id** | **name** |
| **1** | **aa** |
| **2** | **bb** |
| **3** | **cc** |
| **3** | **cc** |

**delete from t\_table a where a.rowid>(**

**select min(rowid) from t\_table b where a.id=b.id**

**);**

# 同义词synonym

**简化sql语句**

**隐藏对象的名称和所有者**

**提供对对象的公共访问**

创建私有同义词

create synonym s\_emp for emp;

创建公有同义词

create public synonym s\_emp for emp;

查询同义词

select \* from s\_emp;

用system用户赋予scott创建私有同义词的权限

grant create synonym to scott;

用system用户赋予scott创建公有同义词的权限

grant create public synonym to scott;

scott需要给与其他用户对emp表的访问权限，其他用户才能访问scott的同义词

切换到scott用户，对test1给与访问emp的权限

grant select on emp to test1;

删除同义词

drop public synonym s\_emp

# 序列sequence

create sequence stu\_seq

start with 1 从起开始

increment by 1 每次增长1

maxvalue 2000 最大值为2000

minvalue 1 最小值为1

nocycle 不循环

cache 10; 预先在缓冲中存放的个数,默认20

nextval返回下一个值

currval返回当前值

第一次必须要用nextval

select stu\_seq.nextval from dual;

insert into student value(stu\_seq.nextval,’Jack’);

特殊格式2010\_001

‘2010\_’|| lpad(to\_char(stu\_seq.nextval),3,’0’)

修改序列信息

alter sequence stu\_seq maxvalue 3000 cycle;

删除序列

drop sequence stu\_seq

# 视图

视图就是来自于一张表或多张表的数据

视图所依据的表叫“基表”

视图的优点有：

1.提供了另外一种级别的表安全性

2.隐藏的数据的复杂性

3.简化用户的sql命令

4.隔离基表结构的改变

5.通过重命名列，从另一个角度提供数据

创建视图(t\_student表要存在，才能创建成功)

create view v\_student as select \* from t\_student;

创建视图(t\_student表不一定存在，都能创建视图,暴力创建)

create force view v\_student as select \* from t\_student;

为了避免创建需要条件的视图，条件变化会影响视图的数据，使用下面语句后就不能更改条件

create view v\_student as select \* from t\_student where sno=1 with check option

所有视图都可以在user\_views查看

select \* from user\_views;

删除视图

drop view v\_student;

在多表是的视图对数据进行操作，需要建键保留表

键保留表：基表要有主键，建立视图后还能作为视图的主键

非键保留表：基表要有主键，建立视图后不能作为主键

更新时，只能更新键保留表的数据。

触发器可以进行对非键保留表进行更新

视图与同义词的区别

1.视图可以对应一张或多张表，同义词只能对应一张表名称

2.视图可以设置其他约束条件

3.可以在同义词上建立视图

# 索引

索引是与表相关的一个可选结构

用来提高sql语句的执行性能，减少磁盘的读取

索引在逻辑上和物理上独立于表的数据

索引分为：B树索引，位图索引

B树索引分为：唯一索引、组合索引、反向键索引 、基于函数索引

创建一个普通索引

create index ind\_student\_sno on t\_student(sno);

所有索引都可以在user\_indexes查看

select \* from user\_indexes;

查看索引的具体信息

select \* from user\_ind\_columns u where u.index\_name=’ ind\_student\_sno’

**索引的碎片**处理

什么事碎片？

由于索引创建后，对表数据删除会导致索引中出现空缺，从而产生索引的碎片,碎片过多会影响sql执行的效率。

怎么处理碎片呢？

删除旧索引，再重新建立索引，或者替换掉旧索引。

怎么判断碎片的多少，来控制处理碎片呢？

查看index\_stats表中的pct\_used列的值，值过低，说明存在碎片。

先对索引进行分析

analyze index ind\_t validate structure;

在查看(默认有10%空间预留，如果是显示90%，就是没有任何碎片)

select name,pct\_used from index\_stats where name=’IND\_T’;

想要再查看，需要再次进行分析

创建索引

alter index ind\_t rebuild;

如果索引的数据量很大，重新建索引会影响用户的访问

alter index index\_name rebuild online;

重建过程中产生最少的重做条目

alter index index\_name rebuild nologging;

重建过程中避免索引重建之后再进行analyze或dbms\_stats来收集统计信息

alter index index\_name rebuild compute statistics;

**唯一索引**

在列中没有重复值

创建唯一索引(sno为多个null不违反重复的规定)

create unique index ind\_sno on student()sno;

**组合索引**

在表的多个列上创建的索引

建立一个以班级和班内编号关联的索引

create index ind\_student on t\_student(class\_id,class\_inside\_id);

适用于组合条件的查询语句,如：

select \* from t\_student where class\_id=’201201’ and class\_inside\_id=’27’;

**方向键索引**

由于类似于学号1001、1002、1003，前面的几位是固定的而后面是变化的，为了查询速度更高，采用方向键索引，就会变为1001、2001、3001，像这样把变化的部分放在前面，能提高查询的速度。

创建反向键索引

create index ind\_student\_sno on t\_student(sno) reverse;

使一个索引变为反向键索引

alter index ind\_student\_sno rebuild noreverse;

**基于函数索引**

基于一个或多个列上的函数或表达式创建的索引

表达式中不能出现聚合函数

不能再LOB类型的列上创建

创建时必须具有query rewrite

创建一个用学生姓名（用大写字母）关联的索引

create index ind\_student\_sname on student(upper(sname))

**位图索引**

位图索引适用于创建在低基数列上，不被经常改动的数据上(如：数据仓库)

低基数列：列上的数据只有固定的几种。(如：性别)

创建位图索引

create bitmap index ind\_student\_male on t\_student(male);

**分区表创建索引**

创建一个分区表

create table emp(code number,name varchar(20))

partition by range(code)

(

partition p1 values less than(100),

partition p2 values less than(200),

partition p3 values less than(maxvalue)

);

创建局部分区索引：在分区表上创建的索引，在每个表分区上创建独立的索引，索引的分区范围与表一致

create index ind\_emp on emp(code) local;

查看索引是否分区

select u.index\_name,u.table\_name,u.partitioned from user\_indexes u;

select \* from user\_ind\_partition;

创建全局分区索引：在分区表或非分区表上创建的索引，索引单独指定分区的范围，与表的分区范围或是否分区无关

创建全局分区索引，对全局索引进行分区

create index ind\_emp on emp(code) global

partition by range(code)

(

partition ind\_p1 values less than(1500),

partition ind\_p2 values less than(maxvalue)

);

全局非分区索引：在分区表上创建的全局普通索引，索引没有被分区

create index ind\_emp on emp(code) global;

索引的相关数据字典视图有：

user\_indexes:用户创建的索引的信息

user\_ind\_partitions:用户创建的分区索引的信息

user\_ind\_columus:与索引相关的表列的信息

# 事务transaction

事务的4种属性：

原子性（A）：事务是不可分割的，要么一起成功，要么一起失败

一致性（C）：事务必须使数据库从一个一致性状态转到另一个一致性的状态

\*隔离性（I）：多个并发事务之间应该相互隔离

持久性（D）：事务一旦提交，应用永远生效。

隔离性

多个并发事务之间应该相互隔离。

如果并发事务没有隔离，可能会引发以下现象：

**脏读**：一个事务读到了另一个事务没有提交（commit）的更新数据

**不可重复读**：一个事务读到了另一个事务已经提交（commit）的更新数据

**幻读**：一个事务读到了另一个事务已经提交（commit）的插入数据

默认情况下，mysql数据库的隔离级别： repeatable-read

默认情况下，Oracle数据库的隔离级别： serizalizable

任何数据库都可以设置四个隔离级别：

read uncommitted 任何现象都不能防止

read committed 可以防止脏读

repeatable read 可以防止脏读和不可重复读

serizalizable 可以防止所有现象

事务相关命令：commit、savepoint、rollback

开启自动提交set autocommit on/off

创建savepoint,然后回滚到savepoint，只能往前回退，不能往后回退

savepoint mark1;

rollback to savepoint mark1;

# PL/SQL

PL/SQL是过程语言与结构化查询语言结合而成的编程语言，对SQL的扩展

PL/SQL支持多种数据类型，如大对象和集合类型，可使用条件循环等控制结构

可用于创建存储过程、触发器和程序包，给SQL语句执行添加程序逻辑

与Oracle服务器和Oracle工具紧密集成，具备可移植性、灵活性和安全性。

在PL/SQL中可以使用：

1.数据操纵命令

2.事务控制命令

3.游标控制

4.SQL函数和SQL运算符

Oracle需要2种引擎：SQL引擎、PL/SQL引擎

PL/SQL语句的组成部分：声明部分、执行部分、异常处理部分

[DECLARE declarations]

BEGIN

executable statements

[EXCEPTION handlers]

END

声明部分：

identifier [CONSTANT] datatype [NOT NULL] [:=|DEFAULT]

赋值方式：

1.使用赋值语句:=（a:=3）

2.使用select into语句(select count(\*) into a from student;)

把学生的数量赋值给total\_row

declare

total\_row number;

begin

select count(\*) into total\_row from student;

dbms\_output.put\_line(‘一共有’||total\_row||’记录’);

end;

声明一共常量

c\_rate constant number:=0.10;

## 声明变量的类型

binary\_integer(存储有符号整数，所需存储空间少于NUMBER类型值)

子类：natural、naturalln、positive、positiven、signtype

数字 number(存储整数、实数和浮点数)

子类：decimal、float、integer、real

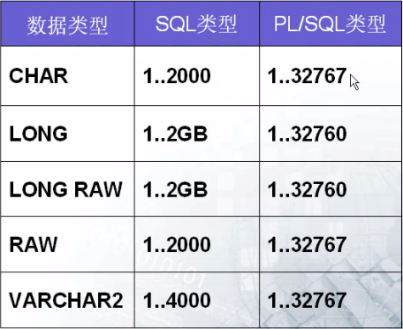
pls\_integer(存储有符号整数，可使算术计算快速而有效)

simple\_integer(11g新数据类型，用于硬件，提高性能)

**标量类型**： 字符：char、varchar2、long、raw、longraw

布尔：boolean(true、null、false)

日期时间:date、timestamp



BLOB (存储大型二进制，如视频、音频、图片)

CLOB (存储大型字符数据)

**LOB类型**：

BFILE(存储操作系统文件的对象)

NCLOB(存储大型nuicode编码的字符数据)

通过dbms\_lob去操作lob的数据

**属性类型**：%TYPE(提供某个变量或数据库表列的数据类型)、

birth student.brithday%type;

%ROWTYPE(提供表示表中一行的记录类型)

用PL/SQL显示表中的CLOB信息

①创建表添加数据

create table test\_clob(c\_id number primary key,c\_info clob);

insert into test\_clob values(1,’我爱java’);

②用PL/SQL显示表中的CLOB信息

declare

clob\_var clob;

length integer; /\*显示的长度\*/

offset integer; /\*从第几个开始显示\*/

output\_var varchar2(10);

begin

select c\_info into clob\_var from test\_clob where c\_id=1;

length:=10;

offset:=1;

dbms\_lob.read(clob\_var,length,offset, output\_var);

dbms\_output.put\_line(output\_var);

end;

用PL/SQL将图片保存到列中

①创建person表

create table person(pid varchar2(20) primary key,photo blob);

②创建目录逻辑(system执行)

create directory PHOTO as ‘d:\photo’;

③在d盘创建photo目录

④给scott赋予读写的权限

grant read,write on directory PHOTO to scott;

⑤用存储过程将图片保存到列中

create or replace procedure insertBlob(id varchar2,imgFile varchar2)

is

img\_file bfile;

img\_blob blob;

lob\_length number;

begin

--先插入一个空值

insert into person varlues(1,empty\_blob());

select photo into img\_blob from person where pid=id;

--读取img\_file中的内容

--指向文件对象

img\_file:=bfilename(‘PHOTO’,imgFile );

--读取文件

dbms\_lob.open(img\_file);

--获取文件长度

lob\_length:=dbms\_lob.getlength(img\_file);

--从文件装载到表中的列

dbms\_lob.loadfromfile(img\_blob,img\_file,lob\_length);

--关闭提交

dbms\_lob.close(img\_file);

commit;

end;

⑥执行存储过程

exec inserBlob(‘1’,’aa.jpg’);

⑦查看图片通过高级语言

## PL/SQL流程控制

条件控制：if语句、case语句

循环控制：loop循环、while循环、for循环

循序控制：goto语句、null语句

其他：exit、continue、when

|  |  |  |
| --- | --- | --- |
| if 条件 then  操作;  end if; | if 条件 then  操作1;  else  操作2;  end if; | if 条件 then  操作1;  elsif  操作2;  else  操作3;  end if; |
| case 数值  when 数值1 then 返回值1  when 数值2 then 返回值2  else 返回值3  end; | loop  循环体  exit when 条件;  end loop; | while 条件 loop  循环体  end loop |
| for 计数器 in 值1..值2  loop  循环体  end loop; | <<a>>  ……  if 条件 then goto a;  endif; |  |

## 动态sql语句

DDL(对表的创建修改删除)语句命令不能直接用pl/sql语句执行，但是可以通过动态sql来执行

begin

execute immediate ‘create table T(t1 int)’;

end;

## 异常处理

异常有两种类型：预定义异常、用户预定义异常

查找学号为5的学生（没有5号学生）

declare

sname student.sname%type;

begin

select sname into sname from student where sno=5;

dbms\_output.put\_line(sname);

exception

when no\_data\_found then

dbms\_output.put\_line(‘没有这个学生。’);

end;

用户预定义异常

非正常的年龄数据处理

declare

birth student.birthday%type;

excep exception;

begin

seclect birthday into birth from student where sno=4;

if birth>to\_date(‘20030101’,’yyyymmdd’) then

raise excep;

else

dbms\_output.put\_line(‘学生的年龄正常’);

end if;

exception

when excep then

dbms\_output.put\_line(‘年龄错误’);

raise\_application\_error(-20001,’年龄错误’);

end;

# 过程和函数

过程：执行某些操作；

函数：执行操作并返回值

子程序的优点：模块化、可重用性、可维护性、安全性

创建存储过程

|  |
| --- |
| create procedure 过程名(参数)  as  申明变量  begin  代码块  [异常]  end; |

参数的三种模式：

in：用于接受调用程序的值，默认的参数模式

out：用于向调用程序返回值

in out：用于接受调用程序的值，并向调用程序返回更新的值

|  |  |
| --- | --- |
| 写一个存储过程，输出  \*  \*\*  \*\*\*  \*\*\*\*  \*\*\*\*\*  \*\*\*\*\*\*  \*\*\*\*\*\*\* | create or replace procedure proc(i in number)  as  a varchar2(50);  begin  a:=’’;  for j in 1..i loop  a:=a||’\*’  dbms\_output.put\_line(a);  end loop;  end;  --打开输出开关  Set serverout on  --执行存储过程  Exec procl(6); |
| 在PL/SQL里调用存储过程  Exec,用来执行存储过程 | create or replace procedure proc(j out int)  as  begin  j:=100;  dbms\_output.put\_line(j);  end;  declare  k number;  begin  proc(k);  end; |

**自主事物处理**

[pragma autonomous\_transaction]，

存储过程中调用别的存储过程不会影响到该过程。

放在as后面；

rollback不能对主程序产生影响

自主事物处理的特征：

与主事务处理的状态无关

提交或回滚操作不影响主事务处理

自主事物处理的结果对其他事务是可见的

能够启动其他自主事务处理

hr用户执行scott的存储过程，先获取权限

grant execute on proc to hr;

创建函数

|  |
| --- |
| create function 函数名 (参数)  return 返回值类型 as  申明变量  begin  代码块  return 返回值;  [异常]  end; |

# 游标

游标是对查询出来的记录进行细致的控制

**三类游标**：

隐式游标：在PL/SQL程序中执行DML SQL语句时自动创建隐式游标，名字固定叫sql

显式游标：用于处理返回多行查询

参照游标：也叫REF游标，用于处理运行时才能确定的动态sql查询结果

**隐式游标的属性有**：

%FOUND：sql语句影响了一行或多行时为true

%NOTFOUND：sql语句没有影响任何行时为true

%ROWCOUNT：sql语句影响的行数

%ISOPEN：游标是否打开，始终未false

|  |
| --- |
| begin  update student set class=1 where sno>4;  if sql%found then  dbms\_output.put\_line(‘更新了’||sql.%rowcount||’行’);  elsif sql%notfound then  dbms\_output.put\_line(‘没有找到行’);  end if;  end; |

显式游标的使用

|  |
| --- |
| declare  sno student.sno%type;  stu student%rowtype;  --声明游标（参数）  cursor mycursor(input\_no number) is select \* from student where sno>input\_no;  begin  sno:=&学生学号;  --打开游标  open mycursor(sno);  --从游标中取一行数据放到内存变量里面  fetch mycursor into stu;  while mycursor%fount loop  dbms\_output.put\_line(‘学号是：’||stu.sno||’，姓名是’||stu.sname);  fetch mycursor into stu;  end loop;  --关闭游标  close mycursor;  end; |

显式游标更新行

|  |
| --- |
| declare  stu student%rowtype;  --声明游标  cursor mycursor is select \* from student where sno=2 or sno=3 for update;  begin  --打开游标  open mycursor;  --从游标中取一行数据放到内存变量里面  fetch mycursor into stu;  while mycursor%fount loop  update student set sno=sno+100 where current of mycursor;  fetch mycursor into stu;  end loop;  --关闭游标  close mycursor;  end; |

循环游标(只能查询用)

|  |
| --- |
| declare  stu student%rowtype;  cursor mysursor is select \* from student;  begin  for cur in mycursor loop  dbms\_output.put\_line(‘学号是：’||cur.sno||’,姓名是：’||cur.sname);  end loop;  end; |

fetch …… bulk collect into 取数据效率远远高于fetch……into

|  |
| --- |
| declare  cursor my\_cursor is select ename from emp where deptno=10;  type ename\_table\_type is table of varchar2(10);  ename\_table ename\_table\_type;  begin  open my\_cursor;  fetch my\_cursor bulk collect into ename\_table;  for I in 1..ename\_table.count loop  dbms\_output.put\_line(ename\_table(i));  end loop;  close my\_cursor;  end; |

参照游标和游标变量用于处理运行时的动态执行的SQL查询

创建游标变量需要两个步骤：

声明参照游标类型

声明参照类型的变量

声明参照游标类型的语法为：

type 参照游标名 is ref cursor [return 返回值类型]

打开游标变量的语法：

open 游标名 for 查询语句

# 触发器

触发器是当特定事件(DML和DDL)出现时自动执行的存储过程

触发器不能被显式调用

触发器的功能：

自动生成数据

自定义复杂的安全权限

提供审计和日志记录

启用复杂的业务逻辑

DDL触发器：在模式中执行DDL语句时执行

数据库级触发器：在发生打开、关闭、登录和退出数据库等系统事件时执行

DML触发器：在对表或视图执行DML语句时执行

语句级触发器：无论受影响的行数是多少，都只执行一次

行级触发器：对DML语句修改的每个行执行一次

instead of触发器：用于用户不能直接使用DML语句修改的视图

语法

create [or replace] trigger 触发器的名字

after|before|instead of

[insert][[or] update [of 列]

[[or] delete]

on 表名或视图名

[referencing{old[as]old/new[as]new}]

[for each row]行级触发

[when(条件)]

pl/sql语句

触发器不能用：rollback commit create drop alter saveopint

创建一个触发器，如果sno<0回滚

|  |
| --- |
| create or replace trigger trig  before insert on student for each row  begin  if :new.sno<0 then  raise\_application\_error(-20001,’学号错误’);  end if;  end; |

:new和:old必须是行级触发器

insert , :new存在,:old不存在

delete , :new不存在,:old存在

update , :new存在,:old存在

# 用户管理

创建用户

create user test1 identified by test1;

给与用户链接权限

grant connect to test1;

查看当前用户的角色

select \* from user\_role\_privs

system查看其他用户的角色

select \* from dba\_role\_privs d where d.grantee=’scott’;

查看角色对应的权限表

select \* from role\_sys\_privs;

数据的备份与恢复

打开cmd

导出scott数据：exp scott/tiger@orcl file=d:\daochu.dmp full=y

Itcast导入数据：imp itcast/itcast@orcl fromuser=scott touser=itcast file=d:\daochu.dmp