# Oracle

在cmd命令提示符窗口登录oracle，输入**sqlplus / as sysdba**。



## 一、MySQL和Oracle的区别

### 1.区别

#### ⑴权限

①Oracle相对于mysql更安全。在Oracle中几乎所有的操作如创建用户、创建表、删除表等等一些增删改查在没有获得权限之前是不允许执行的。

②Oracle中一个数据库对应多个用户，每个用户拥有各自的表(数据库对象)，一个用户不能操作多个数据库。而MySQL中，一个用户对应多个数据库，每个数据库拥有各自的表。

#### ⑵数据库操作

MySQL可以使用show databases或者show tables等语句查看数据库信息。而Oracle不支持这些操作，采用**数据字典**(表格或者视图)查看数据库信息。

#### ⑶数据类型

①Oracle中不定长字符串用varchar2(m)表示，MySQL中则是用varchar(m)表示。

②**Oracle中只能使用单引号。而MySQL中可以使用单引号也可以使用双引号**。对于字符串，Oracle中一定要记得用单引号，不能用双引号。

③Oracle中数字类型较少，有int、double、number。其中number(m)表示m位的整数，number(m,n)中m表示数字总位数，n表示小数点后位数。MySQL中则是smallint、int、float、double、decimal等。

④特别注意：MySQL中使用**limit m,n**语句完成分页查询。而Oracle中则是使用**rownum**。

oracle每个表都有一个虚列rownum，它用来显示结果中记录的行号，保持1，2，3...的顺序的序列。特别注意，**不能直接使用rownum>m的条件形式**，**只能直接使用rownum<m的形式**。若要执行大于的操作，可借助子查询。

假设每页5条数据

第一页：**select \* from emp where rownum<=5;**

第二页：**select \* from (select rownum rn,e.\* from emp e ) t where rn<=10 and rn>5;**

第二页的sql也可写成这样：**select \* from (select rownum rn,e.\* from emp e where rownum<=10) where rn>5;**

⑤MySQL中有date、datetime、time、timestamp等。Oracle中有date和timestamp。

#### ⑷主键自增

MySQL中给主键设置自增是通过给主键字段设置auto\_increment属性。

Oracle中是由数据库对象序列(sequence)来完成自增的需求。

创建一个序列：生成的序列可以在左边的表格栏目中看到。



create **sequence** idseq start with 1 increment by 1 [min 1 max 1000 nocache nocycle];

使用序列的两个**currval**和**nextval**属性来完成数据的获取。

select idseq.currval,idseq.nextval from dual;

比如创建一个表格，并且表格中的idn属性为自增的形式：

create table MYTABLE2(idn int primary key,namen varchar2(20));

create sequence idseq start with 1 increment by 1 nocache nocycle;

insert into MYTABLE2(idn,namen) values(idseq.nextval,'小王');

#### ⑸事务处理

在事物处理上，Mysql默认是自动提交的，但是oracle不是，必须自己提交，即必须commit。

Mysql支持四种隔离级别，默认隔离级别是repeatable read。

Mysql对资源锁定的粒度大，表级锁。

Oracle只支持两种read committed和serializable，默认隔离级别是read committed 。

Oracle对资源锁定的粒度小，支持行级锁。

#### ⑹MySQL和Oracle区别总结

①Oracle相对于mysql更安全。

②在事务处理上， Mysql默认是自动提交的，但是oracle不是，必须手动提交。

③MySQL默认隔离级别是repeatable read。Oracle默认的隔离级别是read committed。

④Mysql对资源锁定的粒度大，表锁。Oracle对资源锁定的粒度小，行锁。

⑤Oracle中只能使用单引号。而MySQL中可以使用单引号也可以使用双引号。

⑥Oracle中不定长字符串用varchar2(m)表示，MySQL中则是用varchar(m)表示。

#### 2.Oracle中truncate和delete的区别

DDL语句是直接提交的不能回滚。DML语句是可以回滚的，不是自动提交的。

**truncate是DDL语句，truncate是清空表格数据，不能加条件即不能跟着where。一旦执行了truncate不可以回滚。执行起来比delete快**。

**delete是DML语句。delete是删除表格中指定的内容的数据，后面可跟where条件，执行之后可以回滚**。

drop是将整个表都删除，包括删除表结构。

## 二、Oracle

### 1.JDBC连接Oracle数据库

Oracle中dba\_tables中的表格只有管理员能操作，user\_tables则是普通用户可操作的，all\_tables则是所有人可操作的。

查询Oracle中普通用户可操作的表格：select \* from user\_tables;

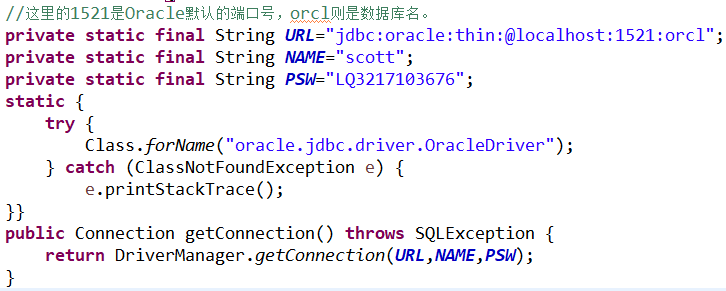
Oracle中默认的端口是**1521**。

java代码连接Oracle数据库时，要将jar包复制到项目中，并BuildPath。

加载Oracle驱动：**oracle.jdbc.driver.OracleDriver**

获取Connection对象的url：**jdbc:oracle:thin:@localhost:1521:orcl。**这里1521是Oracle默认的端口号，orcl则要操作的是数据库名。

**JDBC中连接Oracle数据库**：



### 2.数据类型

**char(n)**：n为1-2000个字节。定长字符串，n字节长，若不指定长度，缺省为1个字节长(一个汉字为2字节)。

**varchar2(n)**：n为1-4000个字节。可变长的字符串。

**number(m,n)**：m是有效数字的位数，n是小数点以后的位数。如number(5,2)中输入575.316，则保存的是575.32。

date：默认格式为DD-MON-YY，如07-11月-00 表示2000年11月7日。

### 3.数值型函数

abs()、sqrt()：开根号、power()：求幂函数、cos()：求余弦三角函数、**mod()**：求余函数、**ceil()**：求大于等于某数的最小整数。**floor()**：求小于等于某数的最大整数。

**round(m,n)**：对m进行四舍五入。n为保留几位小数。round(523.32,1)则为523.3。

**trunc(m,n)**：对m进行直接截断，不会进行四舍五入。如trunc(53.38,1)的值为53.3。

**ascii()**：获得字符的ASCII码值。如ascii(‘A’)为65。若参数为字符串，则返回该字符串首字母对应的ASCII码值，而忽略其它字符。如select ascii('ABC')from dual;的结果为65。

**chr()**：返回与ASCII对应的字符。如chr(65)为‘A’。

**lower()、upper()**字符串转大小写。

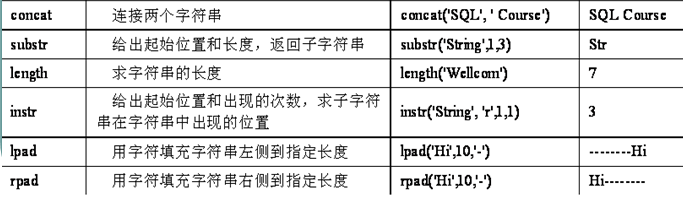
**initcap()**：将字符串转换成每个单词以大写字母开头。initcap(“hello world”)则为Hello World。

**trim**(**leading** ‘a’ **from** ‘abca’)：**leading**代表去除左边的，**trailing**代表去除右边的，**both**代表去除两边的。

**replace()**：替换字符串。

**concat()、substr()、length()、instr()、lpad()、rpad()**。

select replace('abc','b','de') from dual;结果为：adec。



### 4.日期型函数

sysdate：返回当前的日期时间。

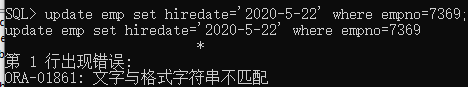
好像没多大用处，没做笔记。

### 5.类型转换函数

**To\_char()**：转换成字符串类型。如将Oracle中的日期转换成指定格式的形式：如将emp表中的日期转换成年月日的形式。

select To\_char(hiredate,'yyyy-mm-dd') from emp;

**To\_date()**：转换成日期格式。注意其它格式的日期要存放在Oracle数据库中必须转换成Oracle中的日期格式。即**日-月年**的形式。否则会报错。如更新emp表中的hirdate为2020-5-22，下面语句报错。



而采用To\_date()将日期格式转换成Oracle中的日期格式才可以正常更新。

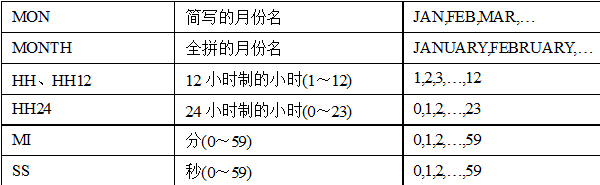


**To\_number()**：转换成数值类型。如To\_number(‘234’)则为234。

#### ⑴日期类型转换

将日期格式转换成字符串，则可以按指定的格式显示。如格式YYYY-MM-DD HH24:MI:SS表示“年-月-日 小时:分钟:秒”。





#### ⑵数值转换符

没懂。

#### ⑶其它常用函数

**nvl(null,xxx)**：空值转换函数。如果括号内为null，则返回指定的值。如果第一个参数为空，则返回第二个参数。第二个参数相当于第一个参数的替换对象。

**decode()**：实现分支功能。**select decode(x,1,’x is 1’,2,’x is 2’,’others’) from dual；当x=1时返回x is 1，当x=2时则返回x is 2，否则返回others**。

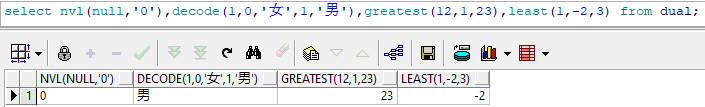
select \* from student;

select s.name,decode(s.sex,1,'男',0,'女') sex from student s;

**greatest()**：返回参数中的最大值。

**least()**：返回参数中的最小值。



## 三、PL/SQL基本介绍

PL/SQL程序的基本单元是块(BLOCK)，块就是实现一定功能的逻辑模块。一个PL/SQL程序由一个或多个块组成。块有固定的结构，也可以嵌套。一个块可以包括三个部分，每个部分由一个关键字标识。

### 1.块中的各部分：

**declare**：声明部分标志。

**begin**：可执行部分标志。

**exception**：异常处理部分标志。(可选)

**end**；：程序结束标志。

### 2.PL/SQL程序的输出：

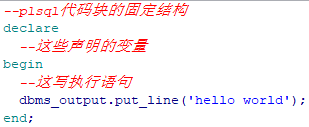
dbms\_output.put();输出括号中的内容。

dbms\_output.put\_line();输出括号中的内容。

dbms\_output.new\_line();回车

注意在Oracle中执行insert、update、delete操作，要使用commit进行提交。

PL/SQL中的--是单行注释，/\*...\*/则是多行注释。



### 3.PL/SQL中的变量

以字母开头，后跟其他的字符序列，字符序列中可以包含字母、数值、下划线等符号。

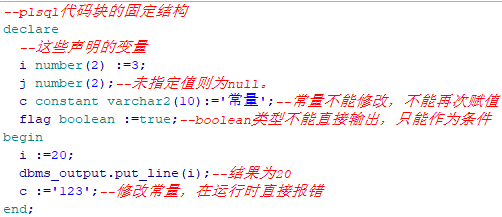
最大长度为30个字符，不区分大小写。

不能使用Oracle的保留字作为变量名。

变量名不要和在程序中引用的字段名相重，如果相重，变量名会被当作列名来使用。Oracle中的代码块赋值用**:=**，判断相等用**=**。一个变量只有声明没有赋值，则获取的值是null。

**定义变量语法**：**变量名 [constant] 类型标识符 [not null][:=|default]**

其中**constant**用来说明定义的变量是常量，如果是常量，必须有赋值部分进行赋值。**:=**用来给变量赋值。default则是赋默认值。



### 4.select...into...

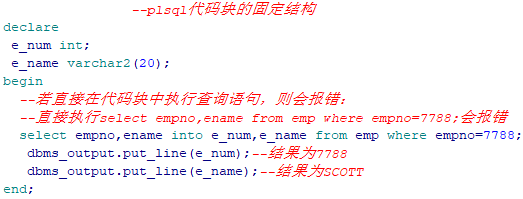
在PL/SQL中select语句总是和into相配合，into后跟用于接收查询结果的变量，select..into...用于将查询数据存入变量中。

形如**：select 列名1，列名2，...into 变量1，变量2...from 表名 where....**

注意：接收查询结果的变量类型、顺序和个数与查询的字段的类型、顺序和个数应该完全一致。select..into..语句必须查询到一行记录，不能多也不能少，否则运行时报错。若接收多行结果，可以采用游标的方式。

Oracle中不是自动提交语句，需要手动commit。

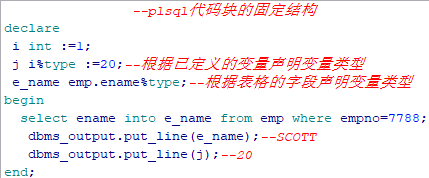
注意：不能直接在代码块中执行select查询，否则报错，必须通过select...into...将查询到的数据存入变量。但是增删改可以直接执行。



### 5.特殊的定义变量方式

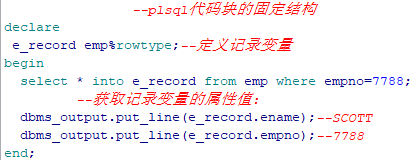
**①%type**：根据表的字段或变量类型定义变量。

在声明变量的类型时，还可以根据数据库表的字段进行定义或根据已经定义的变量进行定义。形式：表字段名或者已定义的变量%type。



**②%rowtype**：记录变量。可以根据表或视图的一个记录中的所有字段定义变量，称为记录变量。

记录变量包含若干个字段，其字段名就是表的字段名，数据类型也一致。形式：**记录变量名 表名%rowtype**。获得记录变量的字段的方法是：记录变量名.字段名。相当于是对象，有很多属性，通过**.**获取属性的值，属性的名字参照表格的列名。



### 6.运算符

算术运算：+ - \* / \*\*(指数)

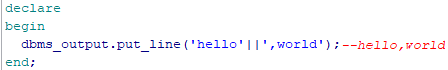
比较运算： > < = !=

逻辑运算： and or not

null的判断： is null 或者is not null

**注意PL/SQL中没有+=、-=、++、--运算符**。

连接运算：**||** 特别注意在代码块中将字符串进行连接不是使用+号，而是使用||号。

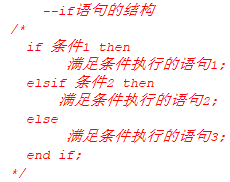


注意**对空值的算术和比较运算的结果都是空，但对空值可以进行连接运算，结果是另外一部分的字符串**。

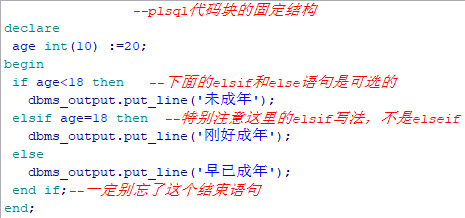
null+5的结果为null。null>5的结果为null。null||’ABC’的结果为‘ABC’。

### 7.if语句

**if语句的形式**：



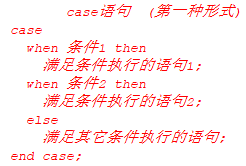
要特别注意elsif的写法，不是elseif。其中elsif和else是可选的。



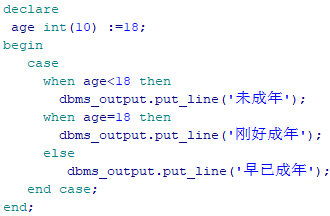
### 8.case语句

#### ⑴第一种形式

结构形式：

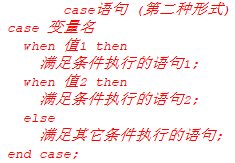


例子：

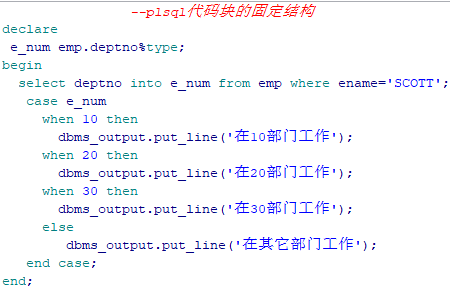


#### ⑵第二种形式

结构形式：



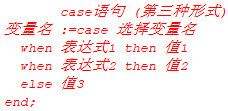
例子：



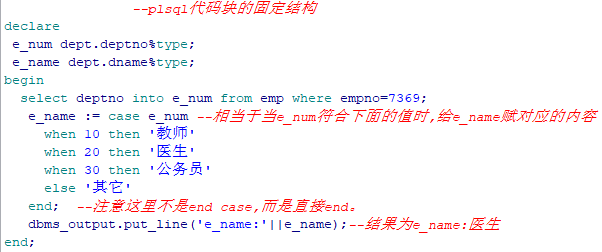
#### ⑶第三种形式

case结构还能以赋值表达式的形式出现，它根据选择变量的值求得不同的结果。

结构形式：

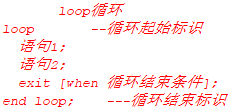


例子：



### 9.loop循环

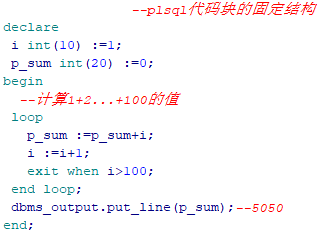
结构：



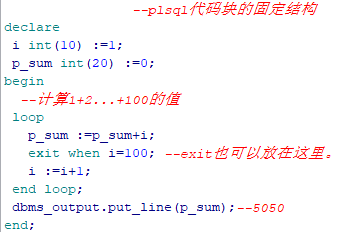
该循环的作用是反复执行loop与end loop之间的语句。

exit用于在循环过程中退出循环，when用于定义exit的退出条件。如果没有when条件，遇到exit语句则无条件退出循环。**注意用于退出循环的exit可以放在loop和end loop之间的任何位置**。

例子1：

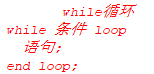


例2：



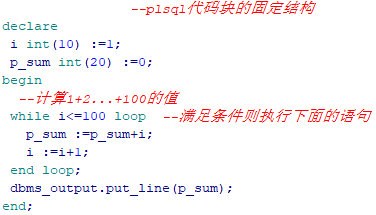
### 10.while loop循环

结构形式：



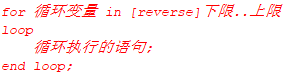
退出循环可以用exit。

例子：



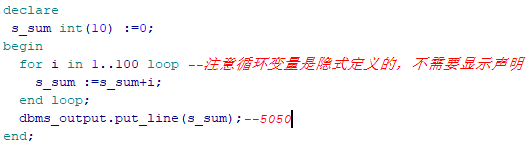
### 11.for loop循环

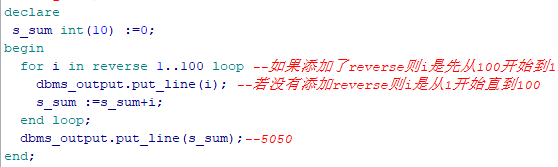
结构形式：



注意：**循环变量是隐式定义的，不需要显示声明**。下限和上限用于指明循环次数。正常情况下，循环变量取值是从下限到上限递增的，添加了reverse之后，循环变量取值由上限到下限递减的。注意下限的值不能比上限的值大，否则循环不会执行该for loop循环。**若要退出循环，可以用exit**。

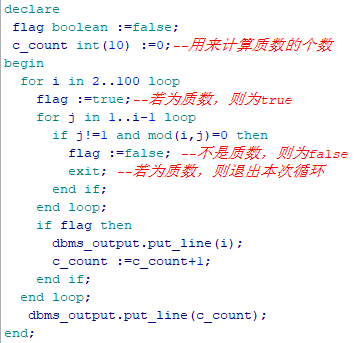
例子：





质数：大于1的自然数中，除了1和它本身之外，没有其它的因子的自然数。

例子：输出1-100之间的质数，和质数的个数。



注意：不管是loop循环，还是while loop循环，还是for loop循环，退出循环都可以用exit退出。

### 12.变量的作用域

PL/SQL代码块是可以嵌套使用，一个变量的作用域在定义它的代码块中，外部代码块定义的变量内部也可以用，内部代码块定义的变量只能在内部中使用。

## 四、游标

游标是SQL的一个内存工作区，由系统或用户以变量的形式定义。

作用：用于临时存储从数据库中提取的数据块。

游标有两种类型：**显式游标**、**隐式游标**

游标一旦打开，数据就从数据库中传送到游标变量中，然后应用程序再从游标变量中分解出需要的数据，并进行处理。

### 1.隐式游标

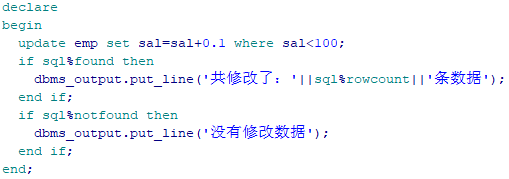
DML操作(insert、delete、update)和单行查询操作(select ...into...)系统会使用一个隐式游标。

隐式游标可以使用名字SQL来访问。注意，通过SQL游标名总是只能访问前一个DML操作或单行SELECT操作的游标属性。所以通常在刚刚执行完操作之后，立即使用SQL游标名来访问属性。

**隐式游标的属性：**

|  |  |  |
| --- | --- | --- |
| 隐式游标的属性 | 返回值 | 意义 |
| **sql%rowcount** | 整型 | 代表DML操作成功执行的行数 |
| **sql%found** | 布尔型 | 值为true代表insert、update、  delete、select...into...操作成功 |
| **sql%notfound** | 布尔型 | 值为true，代表上面的操作没成功 |
| **sql%isopen** | 布尔型 | DML操作执行过程中为true，结束后为false |

例子：



### 2.显示游标

若要提取多行数据，则要自定义一个显示游标。显式游标对应一个返回结果为多行多列的select语句。

获取显示游标的属性：**游标名%属性**。

显示游标的属性：

|  |  |  |
| --- | --- | --- |
| 显示游标的属性 | 返回值 | 意义 |
| **游标名%rowcount** | 整型 | 获得fetch语句返回的数据行数 |
| **游标名%found** | 布尔型 | 最近的fetch语句返回一行数据则为true |
| **游标名%notfound** | 布尔型 | 最近的fetch语句不返回数据则为true |
| **游标名%isopen** | 布尔型 | 游标已经打开时值为true，否则false |

**自定义显示游标有4个步骤**：

**①声明游标**

在declare中声明，语法如下：

**cursor 游标名 [参数 数据类型] is select语句;**

参数是可选的，所定义的参数可以出现在select语句的where子句中。若定义了参数，则必须在打开游标时传递相应的实际参数。

注意这里的select 语句可以跟where条件，但是**不能使用into子句**。在select语句中可以使用在定义游标之前定义的变量。**注意这里有参游标的参数的数据类型不用指定大小**，如cursor emp\_cursor(c\_name varchar2,c\_id number)....不用c\_name varchar2(20)，否则会报错。

**②打开游标**

**open 游标名[(实参)]**

采用for循环遍历不用打开游标、不用采用fetch...into子句提取数据、不用关闭游标。

**③提取数据**

**法一：fetch 游标名 into 变量名1,变量名2;**

**法二：fetch 游标名 into 记录变量;**

按上面格式将游标工作区中的数据取到变量中，提取操作必须在打开游标之后进行。

游标打开后有一个指针指向数据区，fetch语句一次只返回指针所指的一行数据，要返回多行需重复执行，可以使用循环语句。

获取数据的方式：

方式一：将从游标中获取到的数据放到指定的多个变量中，即上面的法一。注意这里的变量要事先定义好，且变量的个数和类型应与select语句中的字段变量的个数和类型一致。

方式二：一次将一行数据存放到记录变量中，需要事先使用%rowtype定义记录变量。

定义记录变量的方式：**记录变量名 表名(或游标名)%rowtype**。注意这里的表名或游标名必须事先存在或者事先定义了。

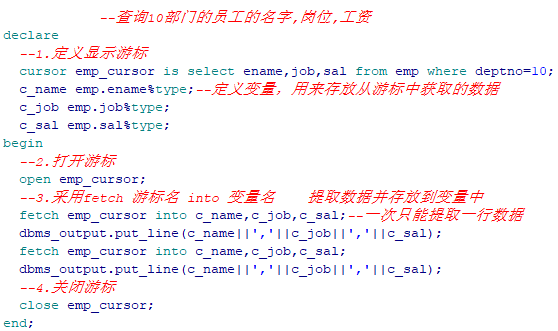
**④关闭游标**

**close 游标名;**

显式游标打开后，必须显式地关闭。游标一旦关闭，游标占用的资源就被释放，游标变成无效，必须重新打开才能使用。

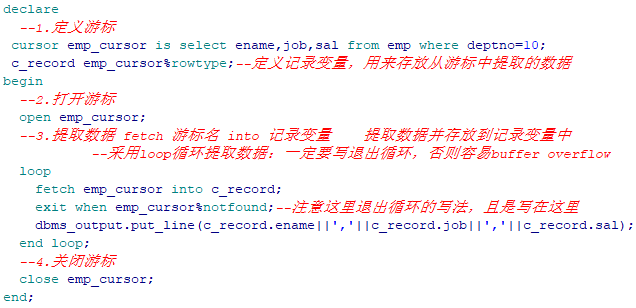
#### 例子1

无参显示游标，提取的数据存放到多个变量中



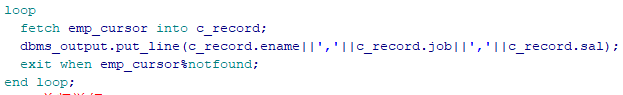
#### 例子2

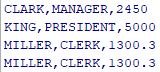
无参显示游标，**loop循环**提取数据存放到记录变量中





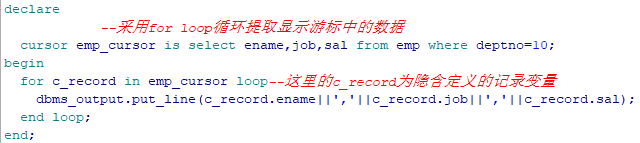
若上面退出loop循环的位置修改成如下，则输出结果会有点问题。可以看到下面的重复输出了一行数据。





#### 例子3

无参显示游标，采用**for loop循环**提取数据。

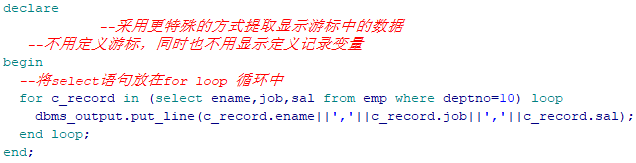




从上面可以看到，采用for loop循环的方式特别简单，**不用显示的定义记录变量，循环中隐式的定义了记录变量。不用打开游标、提取数据、关闭游标**这几个步骤。采用for loop循环，其循环的执行次数与游标取得的数据的行数相一致。

#### 例子4

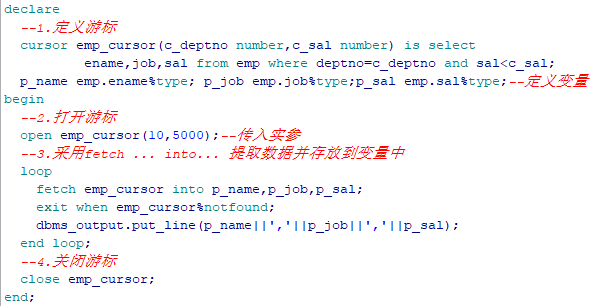
无参显示游标，更特殊的for loop提取数据



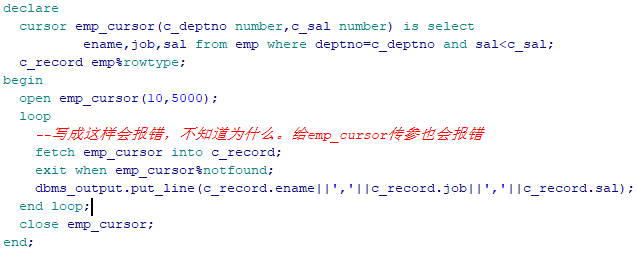
这种方式，更简单，不用再定义游标，也不用显示定义记录变量。**游标的查询语句直接放在for loop循环中**。

#### 例子5

有参显示游标，采用loop循环提取数据：

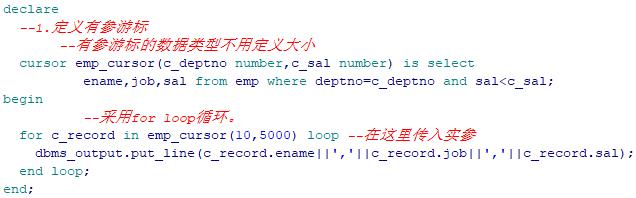


下面这种写法会报错：不知道为什么



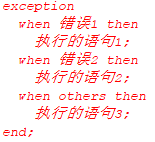
#### 例子6

有参显示游标，采用for loop循环提取数据



## 五、异常处理

异常处理的语法形式：



注意在exception和end之间，都是属于异常处理的范围。

others必须是exception部分的最后一个错误处理分支，如要在该分支中进一步判断错误种类，可以通过使用预定义函数**sqlcode()**和**sqlerrm()**来获得系统错误号和错误信息。

**常见的异常**：

Too\_many\_rows 行过多

No\_data\_found 数据为空

Case\_not\_found 建case时无分支语句

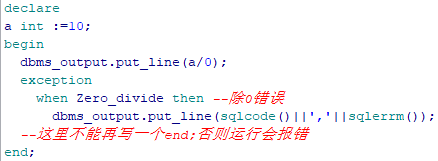
Cursor\_already\_open 重新打开已打开的游标

Dup\_val\_on\_index 主键冲突

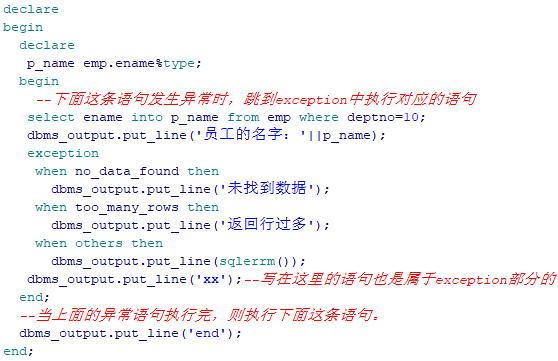
Invalid\_cursor 要对未打开或已关闭的游标读取数据

Zero\_divide 除0异常

例1：



例2：

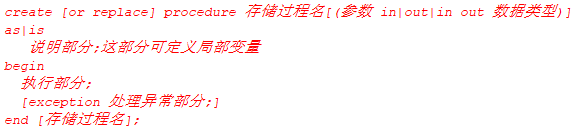


## 六、存储过程和函数

存储过程和函数以***命名的数据库对象形式***存储于数据库的数据字典中，事先经过编译排除错误。可以重复调用。

### 1.存储过程

**⑴创建一个存储过程的基本语法**：

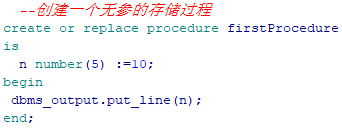


可选关键字or replace 表示如果存储过程已经存在，则用新的存储过程覆盖，通常用于存储过程的重建。

可以给存储过程定义参数，参数有三种形式：in 、out、in out。若没有指明参数的形式，则默认为in。参数的作用是向存储过程传递数据，或从存储过程获得返回结果。in 定义输入变量，传入参数给存储过程，可以是变量或者值。out 定义输出变量，从存储过程中拿到数据，只能是变量。in out则即是输入参数也是输出参数，只能是变量。

可用as或 is，后跟过程的说明部分，可以在此定义过程的局部变量。

例子：



**⑵删除一个存储过程**：

语法：**drop procedure 存储过程名**

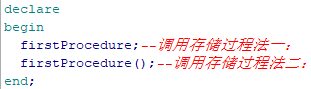


**⑶重新编译存储过程：**

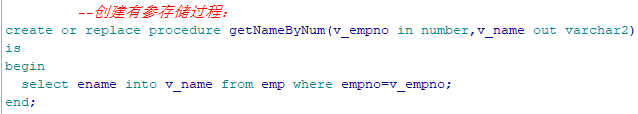
语法：**alter procedure 存储过程名 compile;**

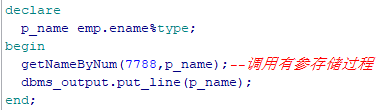


**(4)执行存储过程：**



**例2：**



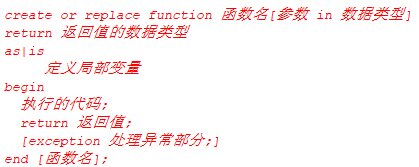


注意传递的参数必须与定义的参数类型、个数和顺序一致(如果参数定义了默认值，则调用时可以省略参数)。参数可以是变量、常量或表达式。存储过程没有参数，在调用时，直接写过程名即可。

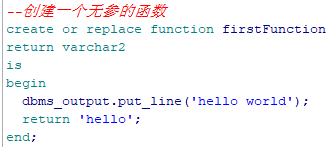
### 2.函数

**⑴创建一个函数的语法**：

**注意函数必须有返回值**。



**例子：**



**⑵删除函数**：

语法：**drop function 函数名;**



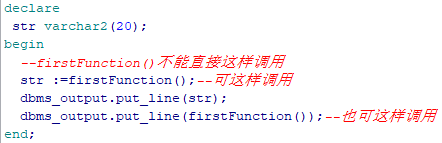
**⑶重新编译函数**：

语法：**alter function 函数名 compile;**



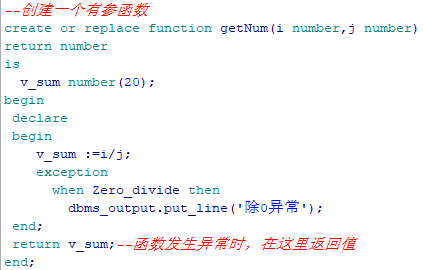
**⑷调用函数**：

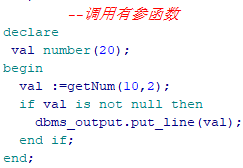
函数调用不能直接调用。



**例子2**：

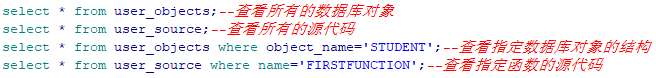
函数的异常处理：不管是否发生异常，都必须有返回值。





注意：存储过程和函数都是数据库对象，处理时用DDL语句，比如创建用create，修改用alter，删除用drop。

通过数据字典进行查询操作。查看所有的数据库对象用user\_objects，查看存储过程或函数的源代码用user\_source。



## 七、触发器

触发器是一种特殊的存储过程，它在插入、删除或修改特定表中的数据时(事件)触发执行。触发器的触发事件分可为3类，分别是DML事件、DDL事件和数据库事件。在创建触发器时要指明触发的事件。

### 1.触发事件

**DML事件(3种)**：insert、delete、update分别在插入数据、删除数据、修改数据时触发事件。

**DDL事件(3种)**：create、alter、drop分别在创建新对象、修改数据库或数据库对象以及删除对象时触发。

数据库事件(5种)：startup数据库打开触发，shutdown在使用normal或immediate选项关闭数据库时触，logon当用户连接到数据库并建立会话时触发，logoff当一个会话从数据库中断开时触发，servererror发生服务器错误时触发。

### 2.触发器

触发器分为4种，**DML触发器、DDL触发器、数据库事件触发器、instead of触发器**(替代触发器)。

#### ⑴DML触发器

DML触发器是是定义在表上的触发器，由DML事件引发。

**触发器的要素**：

**①确定触发的表**

**②确定触发的事件**，DML触发器的触发事件有insert、update、delete

**③确定触发时间**，触发的时间有**before**和**after**两种，**分别表示触发动作发生在DML语句执行之前和语句执行之后**。

**④确定触发级别**，**有语句级触发器和行级触发器两种**。语句级触发器表示SQL语句只触发一次触发器，行级触发器表示SQL语句影响的每一行都要触发一次。

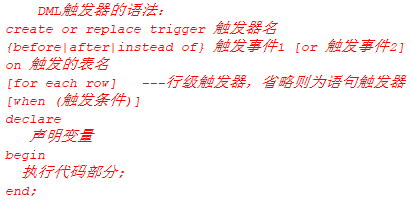
若有多个触发器被定义成为相同时间、相同事件触发,，最后定义的触发器被触发，其他触发器不执行。

一个DML触发器可由多个不同的DML操作触发，采用**inserting**、**deleting**、**updating**关键字区别不同的DML操作，这些关键字用在if分支语句中作为条件使用。

**在行级触发器中，**用**:new**和**:old**(称为伪记录)来访问数据变更前后的值。**insert**插入一条新记录，所以**没有:old记录**；**delete**删除一条已经存在的记录，**没有:new 记录**。**update**语句**既有:new记录也有:old记录**，代表修改前后的记录。引用具体某一列的值采用**:new.字段名**和**:old.字段名**。

注意：触发器体内禁止commit、rollback语句等，同时也禁止直接或间接地调用含有上述语句的存储过程。

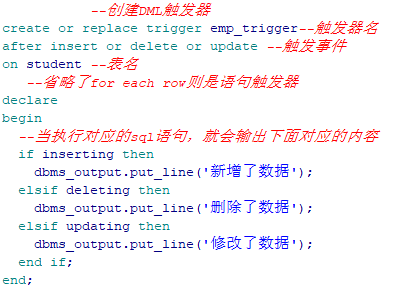
**DML触发器的语法**：



when 后面跟着触发条件表示当满足条件时，该触发器才执行。

for each row表示触发器为行级触发器，省略则为语句触发器。

**例子**：

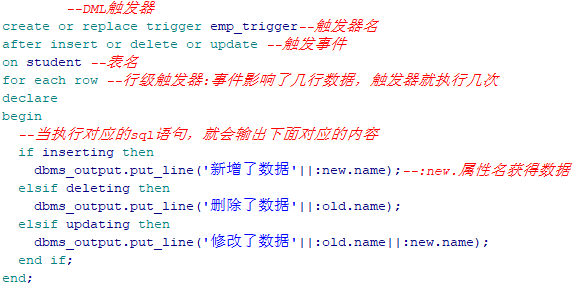


当运行上面的触发器代码之后，创建了触发器，然后，执行下面的代码，就会执行相应触发器中的操作。

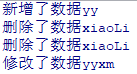




例子2：







**删除触发器**：**drop trigger 触发器名;**



设置触发器暂时关闭或重新打开：

语法：**alter trigger 触发器名{disable|enable};**



## 八、包

包是用来存储相关程序结构的对象，它存储于数据字典中。包由**包头(package)**和**包体(package body)**。

包头是包的说明部分，是对外的操作接口，对应用是可见的；包体是包的代码和实现部分。

出现在包头中的称为公有元素(public)，出现在包体中的为私有元素(private)，出现在包体中的过程(或函数)中的称为局部变量(local)。

**包的优点**：

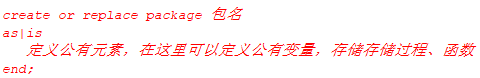
包可以方便地将存储过程和函数组织到一起，每个包又是相互独立的，可以解决存储过程和函数的命名冲突问题。

在同一个会话中，公用变量的值将被保留，直到会话结束。

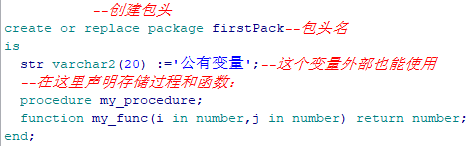
包在被首次调用时，就作为一个整体被全部调入内存，减少了多次访问过程或函数的I/O次数。

### 1.创建包头

**语法：**



**创建包头例子：**



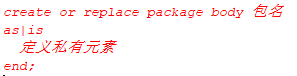
**删除包头**：**drop package 包头名;**

**重新编译包头**：**alter package 包头名 compile package;**

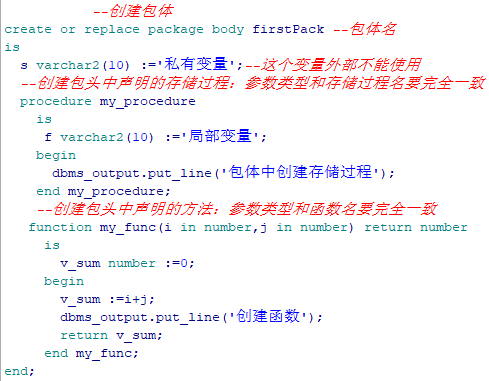


### 2.创建包体

**语法：**



**创建包体例子**：



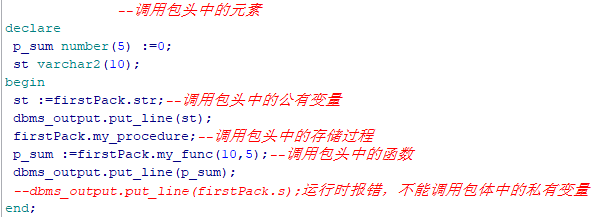
**删除包体**：**drop package body 包体名;**

**重新编译包体**：**alter package 包体名 compile package body;**



在包头中说明的对象可以在包外调用，调用包头中的公有变量和存储过程、函数需要使用**包头.变量、包头.存储过程、包头.函数**。

**调用包头中定义的元素：**



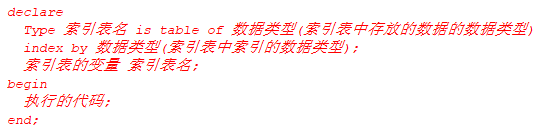
## 九、索引表和批量处理

### 1.索引表

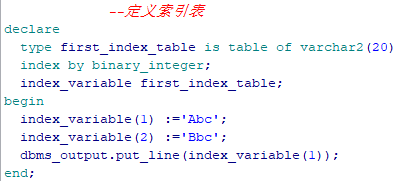
索引表的下标可以为负值，而且其元素个数没有限制。Oracle9i之前，索引表的元素下标只能使用数字类型，即：**binary\_integer**和**PLS\_integer**。Oracle9i开始，索引表的元素下标也允许使用varchar2类型。注意索引表不能做为表列的数据类型使用。

遍历索引表可用for loop循环，索引表对应的索引变量有有first和last属性。如果索引表中存储记录，就是记录表。

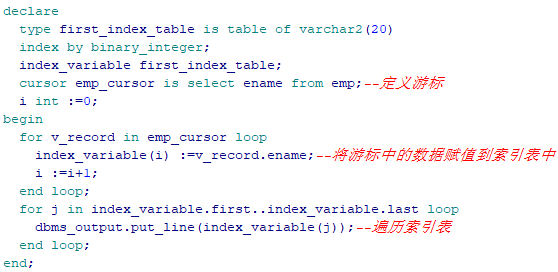
**定义索引表的结构：**



**例子：**



例子2：



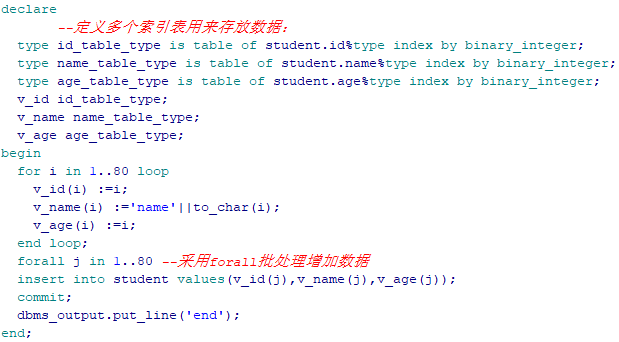


### 2.批量处理

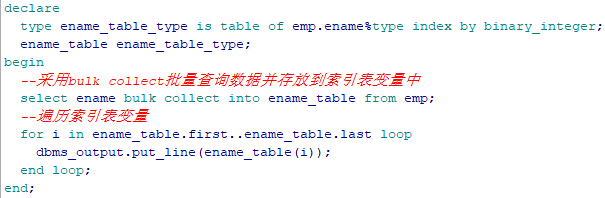
在select、insert、update、delete 语句上处理批量数据时，通过批量绑定，可以极大加快数据处理速度。

forall语句和bulk collect可用来批量绑定。forall一般用于insert、delete、update操作。bulk collect一般用于select.. .into... 和fetch...into...

**从索引表中获取数据并进行批量增加例子：**



**批量查询数据并存放到索引表中：**



## 十、PL/SQL中的集合

**①索引表②嵌套表**，需要先进行初始化才能使用**③数组**，下标从1开始，长度不限，可作为表列的数据类型使用**④变长数组**，下标从1开始，长度有限，需先初始化才能使用，可作为表列的数据类型使用。**⑤记录表**。

