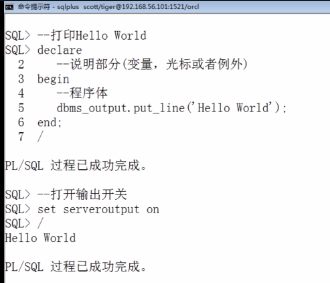
PLSQL是程序设计语言，可以用来操作Oracle数据库！

学习PL/SQL目的：  
1）操作ORACLE数据库，效率更高；  
2）更好去了解后面的存储过程，存储函数，触发器。



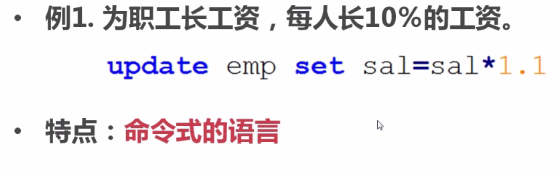
[回到顶部](http://www.cnblogs.com/Qian123/p/5587149.html#_labelTop)

**❤1、PL/SQL简介**

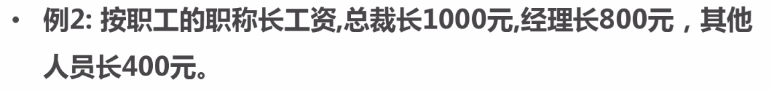
　　PL/SQL (Procedure Language/SQL)过程语言的SQL；

**PL/SQL是Oracle对sql语言的过程化扩展**；  
　　　　--指在SQL命令语言中增加了过程处理语句（如分支、循环等），使SQL语言具有过程处理能力。  
　　**PL/SQL是面向过程的语言**。

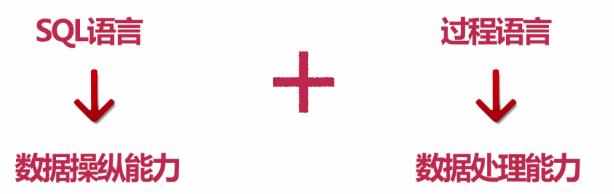
PL/SQL的提出：



但是，对于下面，无法采用命令式的语言完成，要通过程序的方式来完成：（学习PL/SQL的目的）



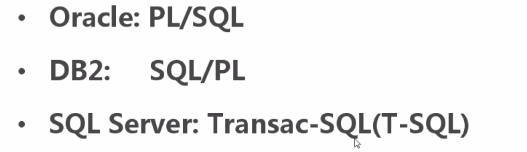
（数据操纵能力：SQL语言的插入、编辑、删除或者查询）



上面两个结合，使得PL/SQL面向过程，但是比过程语言简单、高效、灵活、实用。

操作数据库效率最高的语言就是PL/SQL，而不是其他编程语言。

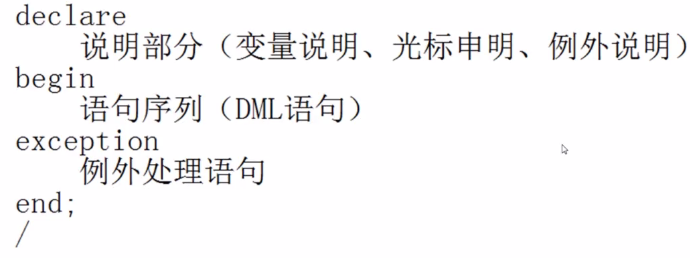
注意，不同的数据库对SQL语言有不同的扩展，从而形成不同的编程程序语言。



[回到顶部](http://www.cnblogs.com/Qian123/p/5587149.html#_labelTop)

**❤2、PL/SQL的基础语法**

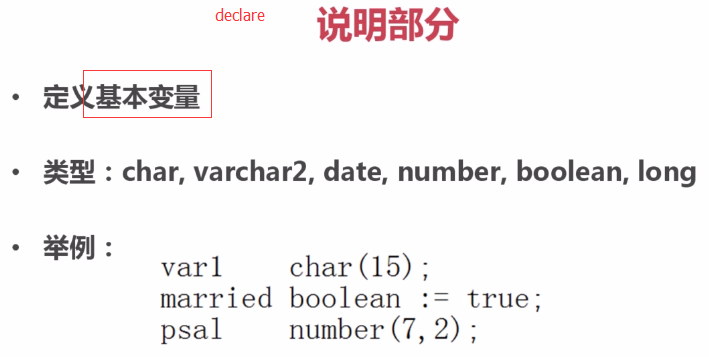
　　PL/SQL的程序结构：



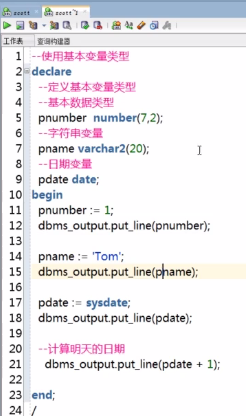
注：如果没有说明部分，declare可以不写。以begin开始，end结束

具体介绍如下:

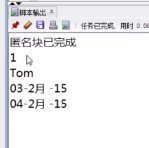
* **基本变量类型**：



注意，在PL/SQL程序中，定义基本变量，变量名在前面，类型在后面



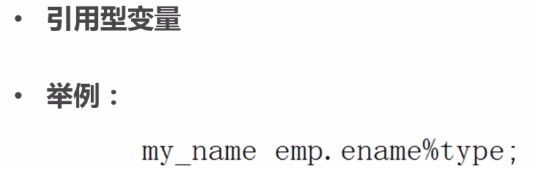
执行结果：



* **引用型变量和纪录型变量**：

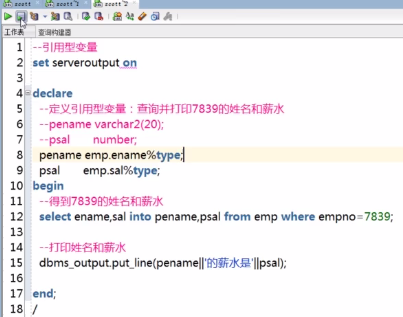
--引用型变量

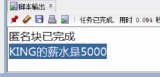
变量名称 变量类型 （引用emp表ename列的类型作为my\_name的类型）



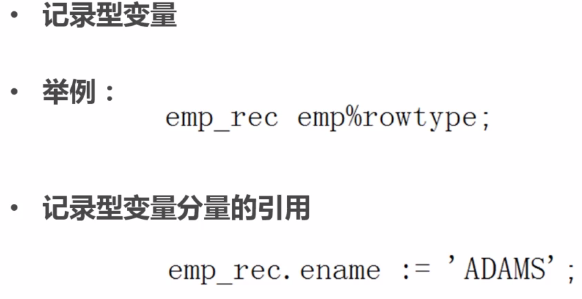
注：PL/SQL赋值有两种方式：1、：= 2、into（into后面的变量顺序一定是跟前面的是一一对应）

示例：

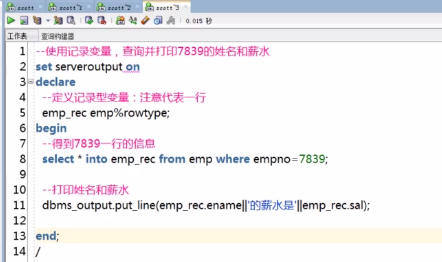




--纪录型变量



示例：

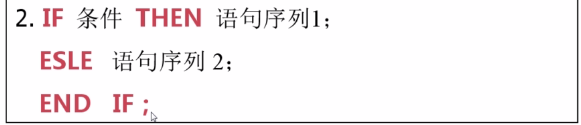


结果：

* if 语句的使用

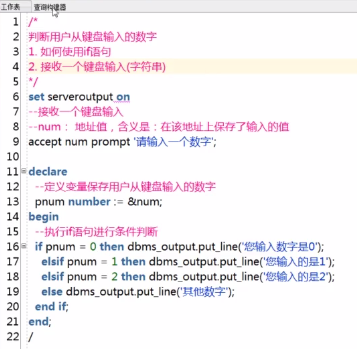
if语句的三种形式





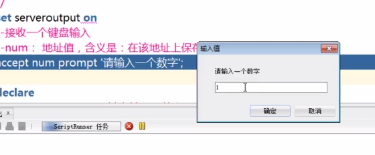


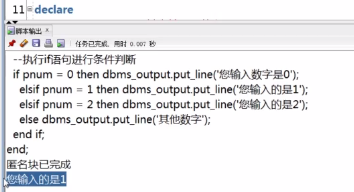
if 语句案例：判断用户从键盘输入的数字



分析：从键盘输入数值：accept num prompt '请输入数字'  
PL/SQL中接受数字：declare pnum := &num;(注意：其中的num为地址值。&num表示取出该地址中的值)

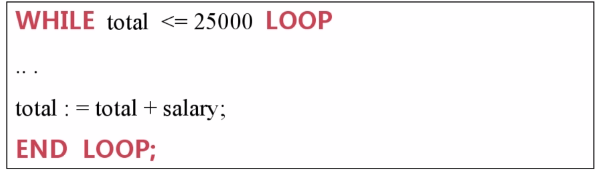
结果：



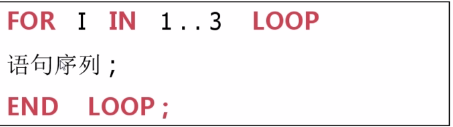


* 循环语句的使用：

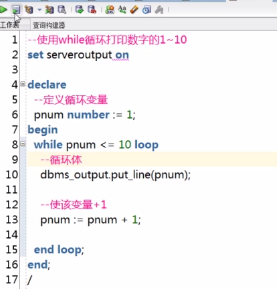
三种循环：



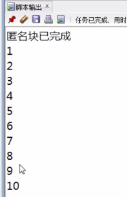




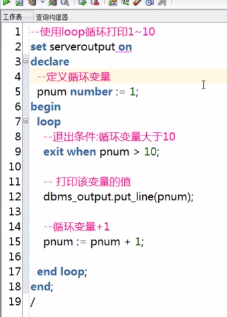
示例：



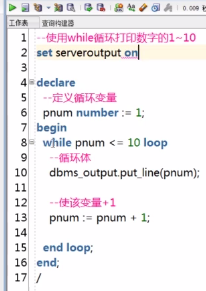
结果：



示例2：



示例3：



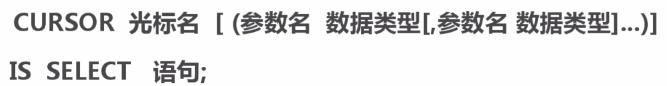
建议使用loop循环，在控制光标的时候比较方便。

[回到顶部](http://www.cnblogs.com/Qian123/p/5587149.html#_labelTop)

**❤3、光标**

光标（游标）就是一个结果集（Result Set）

* **光标的语法：**



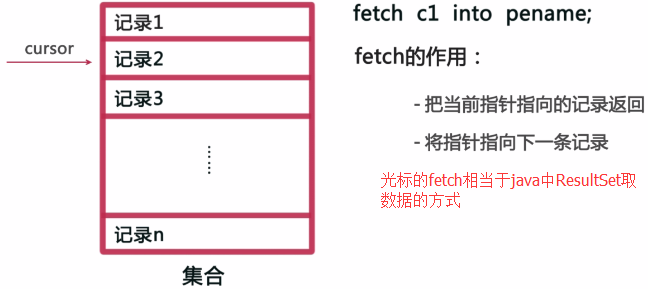
一个具体的光标：http://images2015.cnblogs.com/blog/690102/201606/690102-20160623095838328-334154058.png

光标在使用之前，需要先打开光标http://images2015.cnblogs.com/blog/690102/201606/690102-20160623100008985-2118131142.png

关闭光标http://images2015.cnblogs.com/blog/690102/201606/690102-20160623095939750-1696108136.png

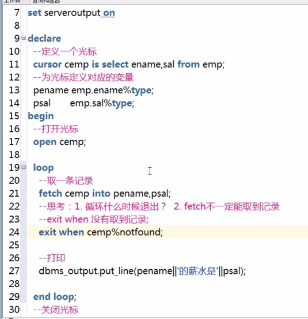
光标打开以后如何从光标这个集合当中取出每一行记录





光标的具体案例1：http://images2015.cnblogs.com/blog/690102/201606/690102-20160623100423547-53803461.png





http://images2015.cnblogs.com/blog/690102/201606/690102-20160623100750625-823573115.png

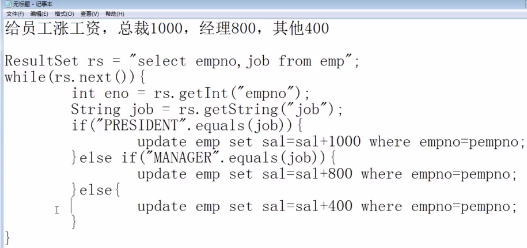
结果：



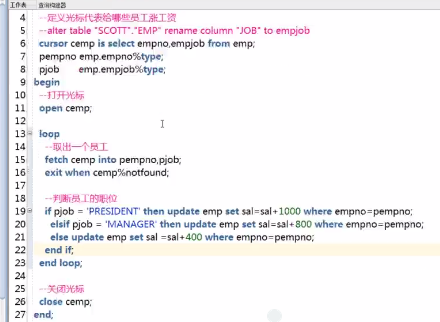
光标的具体案例2：



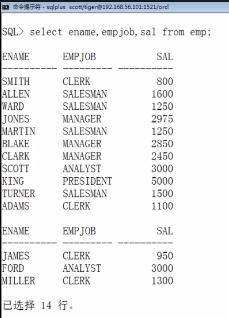
代码程序：



PL/SQL程序：

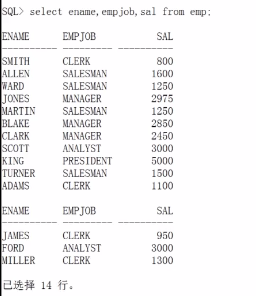


涨工资之前先查询原先的工资：



然后执行程序：http://images2015.cnblogs.com/blog/690102/201606/690102-20160623102306063-1273845134.png

然后再查询：



结果还是和原来一样，但是程序也并没有出错。那问题是什么？

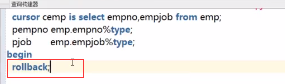
主要是Oracle数据库是默认开启事务的，也就是说上面的PL/SQL程序会在一个事务当中来执行，而我们的查询语句也是会在一个事务当中来查询信息。既然两个事务同时访问数据库，我们就必须要考虑事务的隔离性和隔离级别的问题。

对于Oracle数据库来说，默认的事务隔离级别是read（读）commited（提交）

如果我们PL/SQL程序中如果不提交，我们就当然查询不到结果。所以，注意，写了PL/SQL程序后，如果程序当中对数据做了更改，最后一定要记住需要提交commit。



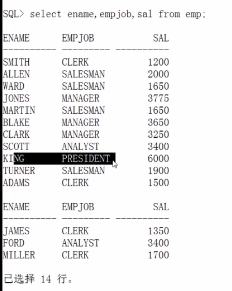
由于我们之前已经执行了PL/SQL程序（没有加commit的程序），所以为了不影响实现的效果，采用rollback把刚才的数据给回退掉。



然后再来执行下这个完整的程序



现在我们在命令行上再来查询下

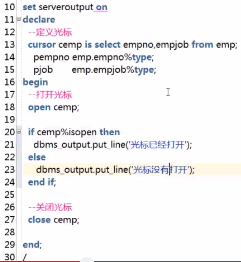


* **光标的属性和光标数的限制**

**--光标的属性：**

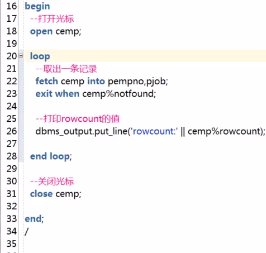
　　%found %notfound  
　　%isopen : 判断光标是否打开  
　　%rowcount : 影响的行数（不是总行数）

示例：



结果：

继续补充上面的程序：

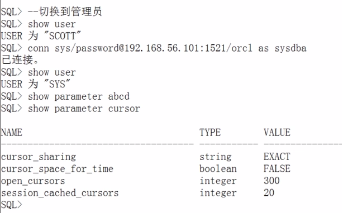


执行结果：



**--光标数的限制：**默认情况下,oracle数据库只允许在同一个会话中,打开**300**个光标（我们可以通过查询Oracle初始化参数查询到这个值）

 不允许无限制的打开光标，也就是光标打开之后，不关闭的话，当你光标数目打开到一定程度后，就不能再打开新的光标了。



show paramter 后面输入的查询值会在值前后自动加上% 模糊查询

如果系统比较小，光标数300可以，但是对于一个大型的系统，往往是不够用的。下面可以修改光标数的限制：

**alter system set open\_cursors =**400**scope = both;**

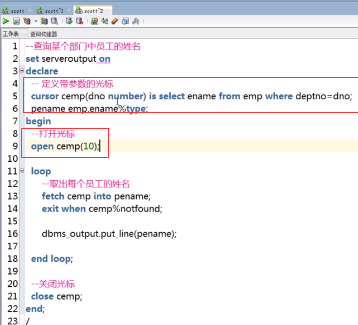
scope 的取值范围：both,memory,spfile(数据库需要重启)  
memory:只更改当前实例，不更改参数文件  
spfile:只更改参数文件，不更改当前实例（数据库需要重启才会生效）  
both: memory 跟spfile同时被更改

* **带参数的光标**

语法：



带参数的光标打开光标以及定义光标是跟不带参数的光标不一样的





[回到顶部](http://www.cnblogs.com/Qian123/p/5587149.html#_labelTop)

**❤4、Oracle中的例外**

例外是程序设计语言提供的一种功能，用来增强程序的健壮性和容错性。

Oracle数据库中有两种意外：**系统例外**，**自定义例外**

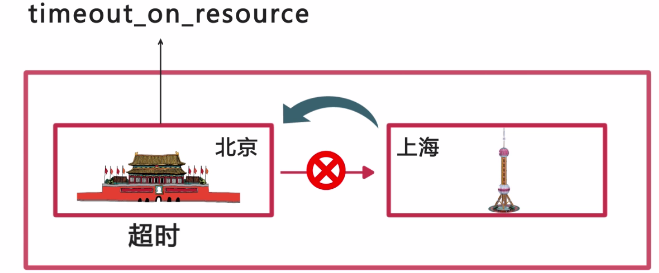
1）系统例外:No\_data\_found（没有找到数据）、

　　　　　　Too\_many\_rows（select ... into 语句匹配多个行）、

　　　　　　Zero\_Divide(被零除，0做除数产生的)、

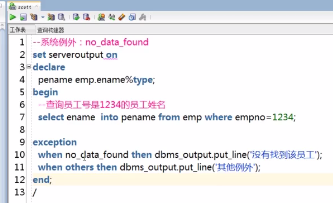
　　　　　　Value\_error（算数或转换错误）、

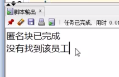
　　　　　　Timeout\_on\_resource（在等待资源时发生超时，比较典型的场景就是分布式数据库）



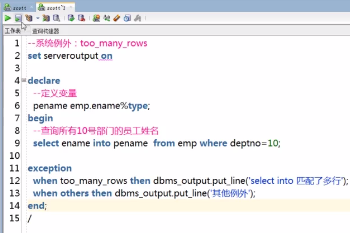
-- **no\_data\_found** (没有找到数据)

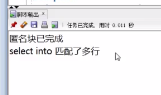
异常的捕获通过“exception”模块来完成：  
...  
exception  
when ... then ...  
when others then ...  
...  
注意：  
1. “then”之后可以跟多条处理该异常的语句；  
2. “when others”代表除以上所列举的异常之外的所有其他异常；





-- **too\_many\_rows** (select ... into 语句匹配多个行)





--**zero\_Divide** (被零除，0做除数产生的)

