PL/SQL程序员必须熟悉下面这些最基本的操作：

* 在数据库中导航；
* 创建和编辑PL/SQL源代码；
* 编译PL/SQL源代码，修正编译器发现的代码错误（或者，可选的，处理警告信息）；
* 在某些环境下运行编译后的程序；
* 检查程序执行结果（可能是从屏幕上的输出，或者是根据表中的数据变化，等等）。

PL/SQL是寄生在Oracle执行环境中的（因此说它是“嵌入式语言”），这一点和那些独立语言比如C语言不同。也正是因为这一点不同，在PL/SQL中完后上述任务就有一些细微差别需要注意。这一章将演示如何在一个最基本的层面（使用SQL\*Plus这个最原生态的工具）完成这些任务，并对这些细小差别做一个大致的浏览。这一章也包括了在常见编程环境中（比如PHP、C）该如何调用PL/SQL的例子。

## 1、在数据库中导航

在开发过程中，你会需要检查数据库里面的数据结构（比如表、列、序列、用户自定义类型等），当然还有那些你将要调用的已有的存储过程。你可能还需要知道表中实际内容（比如列、约束等）。

两种方法在数据库里导航：

1. 使用IDE工具（Integrated Development Environment，集成开发环境），比较流行的有Toad、SQL Developer、PL/SQL Developer、SQL Navigator等。这些工具都提供了图形化的浏览器，可以在上面通过点击操作进行导航。
2. 命令行方式：这种方法是在SQL\*Plus这种命令行环境下运行脚本来查看数据库内容。比如查看ALL\_OBJECTS或USER\_OBJECTS两个数据字典视图的内容。

## 2、创建和编辑源代码

PL/SQL所特有的是，存储过程的源代码在编译和执行之前必须先被加载到数据库中，这份保留在数据库内的代码拷贝可以被有权限的程序员查看。说到这里，我们立即会想到一堆代码管理的问题，比如：

* 该如何、从哪里发现存储过程代码的“原始”拷贝？
* 这些代码到底是保存在磁盘上还是在数据库里？
* 如何做备份，多长时间做一次备份？
* 如何管理多个开发人员对代码的访问？也就是说，我们是不是也需要一个软件版本控制系统呢？

源代码保存到文件里——建议不要把数据库作为你的代码仓库。

## 3、SQL\*Plus

Oracle的SQL\*Plus提供了一个命令行方式的解释器，同时支持SQL和PL/SQL。SQL\*Plus可以接收用户输入的语句，然后把语句发送给Oracle服务器，并显示处理结果。

这些年来Oracle已经提供了几种版本的SQL\*Plus，包括：

* 一个控制台程序

这是一个从SHELL或者命令行提示符下运行的程序。

* 一个“伪GUI”程序

已经不再提供支持。

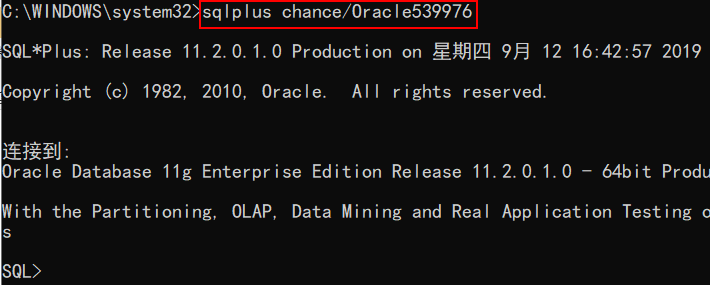
* Isql\*Plus

这个程序是以Web浏览器方式运行的，Web浏览器连接到中间层的Oracle HTTP Server和iSQL\*Plus服务器。

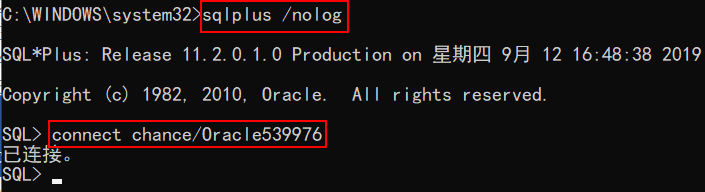
从Oralce 11g开始，Oralce只提供控制台的SQL\*Plus（sqlplus.exe）。

### 3.1启动SQL\*Plus

要启动控制版本的SQL\*Plus，你只需要在操作系统命令提示符下输入“sqlplus”即可。这种启动方式在UNIX和Windows操作系统下都适用。

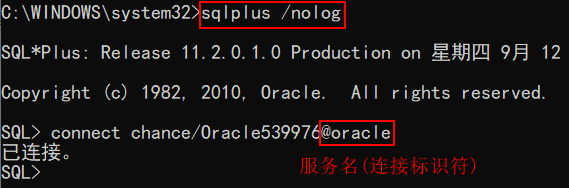
也可以在命令行中提供用户名和密码的方式进入SQL\*Plus：  


在多用户系统，适用/NOLOG选项来启动SQL\*Plus，这种方式只是进入SQL\*Plus环境，不会连接到数据库。接下来通过CONNECT命令，并提供用户名密码进入数据库。



如果运行SQL\*Plus的这台计算机上已经正确配置了Oracle Net，并且远程数据库的管理员已经授权了你的访问，你就可以用SQL\*Plus连接到这些远程数据库上。要想这么做，需要知道远程数据库的连接标识符（也叫做服务名），当然也要提供你的用户名和密码。

需要把连接标识符跟在用户名和密码的后面，中间用符号“@”分隔。



SQL\*Plus运行起来后，可以进行各种操作了：

* 运行一个SQL语句；
* 编译一个PL/SQL程序，并保存到数据库中；
* 运行一个PL/SQL程序；
* 执行一个SQL\*Plus专有的命令；
* 运行一个掺杂了上面这些内容的脚本。

### 3.2运行一个SQL语句

SELECT deptno,dname,loc FROM dept;

在SQL\*Plus环境中，缺省的情况下是把分号“；”作为一个SQL语句的结束标志，不过也可以修改这个结束字符。

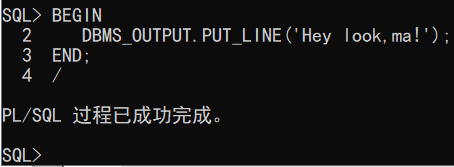
### 3.3运行PL/SQL程序

输入下段PL/SQL代码：

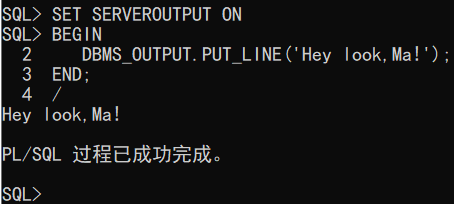
BEGIN

DBMS\_OUTPUT.PUT\_LINE(‘Hey look,ma!’);

END;



糟糕，这段代码调用了PL/SQL的一个内置方法以显示一些文本，虽然这段代码成功的结束了，但是屏幕上却没有期待的输出。这就是SQL\*Plus第一个让人不爽的行为：SQL\*Plus缺省会“吃掉”输出内容。为了正常地显示内容，你必须要通过一个SQL\*Plus的命令打开SERVEROUTPUT。像下面这样：



通常我会把SERVEROUTPUT命令放在我的启动脚本里（见“在启动时加载你的自定义环境”一节），让输出一直启用直到遇到下面的情况：

* 你通过断开、退出或者是其他什么方式结束了会话。
* 明确地通过SET SERVEROUTPUT OFF关闭输出；
* Oracle数据库或者因为你自己的主动请求或者因为编译错误而丢弃了会话状态（见“重编译失效的程序单元”一节）。

每当在控制台的SQL\*Plus环境中录入一条SQL或者PL/SQL语句，SQL\*Plus会为第一行以后的每行代码分配一个数字行号。提供行号有两个好处：首先，如果你使用内置的行编辑器（你总有一天会用到的），有行号的帮助可以让你知道要编辑哪一行；其次，如果数据库发现代码中有错误，通常会在错误报告中给出行号。

要想让SQL\*Plus知道你已经完成了PL/SQL程序的录入，你必须使用一个斜杠“/”作为结尾。尽管“/”是基本无害的，但是斜杠也有一些重要的特点：

* 斜杠“/”的含义是“执行刚录入的这个语句”，不管刚录入的是一个SQL语句还是一个PL/SQL代码；
* 斜杠“/”不属于PL/SQL语句，也不属于SQL语法，它是SQL\*Plus特有的命令；
* 斜杠“/”只能自己出现在一行中，这一行不能再有其他的命令；

SQL\*Plus为PL/SQL用户提供了一个EXECUTE命令，这就避免了输入BEGIN、END、结尾的斜杠“/”。所以，下面的写法是和之前那段代码完全等价的简写形式：

SQL> EXECUTE DBMS\_OUTPUT.PUT\_LINE(‘Hey look,Ma!’)

使用这种方式，结尾的分号“;”也是可选的。SQL\*Plus提供的许多命令都有缩写形式，并且不区分大小写，EXECUTE命令也一样。所以很多时候这个命令都是这么用的：

SQL> EXEC DBMS\_OUTPUT.PUT\_LINE(‘Hey look,Ma!’)

### 3.4运行脚本

基本上所有可以在SQL\*Plus环境中以交互方式运行的语句都可以保存到一个文件，然后重复执行。使用SQL\*Plus的“at符号”（@）命令运行脚本。下面例子就是运行abc.pkg文件中的所有命令：

SQL> @abc.pkg

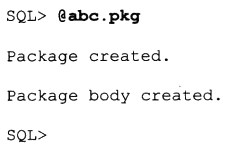
使用这种方式，这个文件必须在我的当前目录下（或者用SQLPATH指定的目录下）。如果偏爱使用单词而不是@符号，可以使用START命令：

SQL> START abc.pkg

SQL\*Plus会做如下的事情：

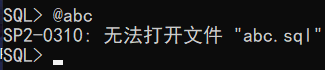
1. 打开名叫abc.pkg文件。
2. 尝试顺序执行这个文件中的所有命令，包括SQL、PL/SQL、SQL\*Plus的命令。
3. 当所有命令都完成后，关闭文件，并返回到SQL\*Plus提示符下（如果文件中调用了EXIT语句，则会直接退出SQL\*Plus环境）。

比如：



默认的行为是仅在屏幕上吧每个语句的结果显示出来，如果想看到文件的原始内容，可以使用SET ECHO ON。

如果没有添加文件扩展名，就会出现下面的结果：



如图所示，如果没有添加扩展名，那么会自动添加默认的文件扩展名.sql。

### 3.5什么是“当前目录”

每次从操作系统命令行提示符进入SQL\*Plus时，SQL\*Plus会把当时所在的操作系统目录作为自己的“当前目录”，换句话说，如果我们是像下面这样启动SQL\*Plus的：

C:\BOB\FILES>sqlplus

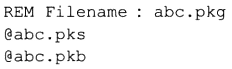
那么在SQL\*Plus中的所有文件操作，都会被认为操作的是这个目录“C:\BOB\FILES”下的文件，“C:\BOB\FILES”就是当前目录。

如果是通过快捷方式或者菜单为的方式进入到SQL\*Plus，那么“当前目录”就取决于操作系统的行为了。进入了SQL\*Plus之后又该如何更改“当前目录”呢？这就取决于版本，如果是控制台版的SQL\*Plus，是没有办法修改的。你必须退出SQL\*Plus、在操作系统中改变目录，然后再次启动SQL\*Plus。

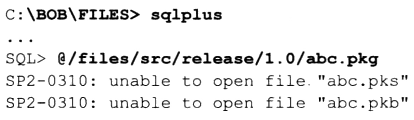
可以在文件为名前加上执行路径，比如：

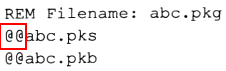
SQL>@/files/src/release/1.0/abc.pkg

运行其他目录下的脚本又引出了一个有趣的问题。如果abc.pkg脚本位于某个目录下，然后它又调用了其他脚本会怎样？比如该脚本可能包含下面这些行：



（REM开头的行是一个注释，会被SQL\*Plus忽略掉。）执行abc.pkg脚本会调用abc.pks和abc.pkb两个脚本，但是我们没有提供目录信息，SQL\*Plus会去哪里找这些文件呢？



SQL\*Plus只在当前目录下寻找脚本。为解决这个问题，Oracle提供了@@命令。两个atat符号意味着，在本次调用中“请假设当前目录已经改成当前执行文件所在的目录”。所以，abc.pkg脚本更好的写法应该是：  


### 3.6其他SQL\*Plus任务

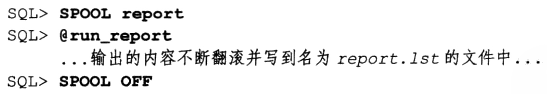
1）个性化设置

对于SQL\*Plus而言，可以通过命令行的方式改变它的一些内置变量或配置的值来改变它的行为。比如SET SUFFIX（改变默认的文件扩展名）、SET LINESIZE n（设置显示时每一行最大字符数，多出的折行处理）。

查看当前会话的各种设置的值，使用SHOW ALL命令。

2）输出保存到文件

把一个SQL\*Plus会话的输出保存到文件中——比如你想生成报告、或者记录下你的活动，或者要动态的生成命令稍后执行。一个简单的方法就是利用SPOOL命令，比如：



SPOOL report告诉SQL\*Plus把以后的所有输出内容保存到report.lst文件中。这个扩展名.lst是缺省的扩展名，也可以在SPOOL命令中提供自己的扩展名而不使用默认的，比如：

SQL> SPOOL report.txt

SPOOL OFF命令告诉SQL\*Plus结束这个输出保存行为，并关闭文件。

3）退出SQL\*Plus

使用EXIT命令，可以退出SQL\*Plus，并回到操作系统。

退出数据库但不退出SQL\*Plus，使用DISCONNECT命令。

4）编辑语句

SQL\*Plus会在缓存中保留最后执行的语句，你可以使用内置的行编辑器或者任何外部的编辑器来修改这条语句，接下来，如何设置使用一个外部的编辑器。

使用EDIT命令会让SQL\*Plus把当前命令缓存中的内容保存到一个文件（默认为afiedt.buf），然后暂停SQL\*Plus，然后调用编辑器：如果想编辑一个已有的文件，直接把文件名作为参数跟在EDIT后面即可。一旦保存文件并退出编辑器，SQL\*Plus会话就把编辑后的最新文件内容读到缓存中，然后会话继续。

Oracle使用的外部编辑器默认是：

* UNIX、Linux相关系统使用ed；
* Windows下使用的是Notepad。

SQL\*Plus所选的外部编辑器已经被硬编码到sqlplus可执行文件中了，尽管如此，我们仍然可以很容易地改变当前编辑器，通过修改SQL\*Plus的\_EDITOR变量值就可以。

5）在启动时加载你的自定义环境

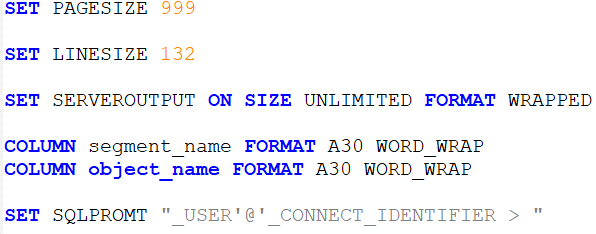
要自定义SQL\*Plus环境，需要编辑一个或者全部的自动启动脚本。

SQL\*Plus启动时的行为：

1. 首先寻找$ORACLE\_HOME/sqlplus/admin/glogin.sql脚本，找到执行这个脚本中的命令。这是一个“全局”登录的脚本，不管这些人是从哪个目运行的。
2. 接下来，SQL\*Plus就会运行当前目录下的login.sql脚本，如果有这个脚本的话。

启动脚本和其他SQL\*Plus脚本一样，可以包含同类语句：SET命令、SQL语句、列格式化命令等。

这两个文件都不是必须的。如果两个都有，则SQL\*Plus会先执行glogin.sql，然后执行login.sql，如果两个脚本中发生了变量冲突，会以最后一个配置为主。



### 3.7SQL\*Plus中的错误处理

SQL\*Plus表现命令成功的方法根据运行命令类型不同。对于大部分SQL\*Plus专有命令，只要没有错误消息就代表成功了。另一方面，对于PL/SQL命令而言，通常会用某种正面内容来标识命令成功执行。

如果遇到了一个SQL或者PL/SQL的错误，缺省行为时报告这个错误然后继续处理。在交互式场景下，这种处理行为是很理想的。但如果运行的是一个脚本，很多时候我们更希望出现错误后中止SQL\*Plus运行。使用如下命令：

WHENEVER SQLERROR EXIT SQL.SQLCODE

这样一来，如果数据库返回任何SQL语句或者PL/SQL语句的错误，SQL\*Plus就会中止。而SQL.SQLCODE的含义是：当SQL\*Plus中止后，把返回值设置成一个非0值，你可以在调用SQL\*Plus的环境中得到这个值。否则，SQL\*Plus会以返回值0结束，这会导致误以为脚本执行成功了。

另一种写法：

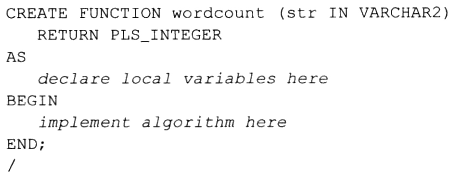
WHENEVER SQLERROR SQL.SQLCODE EXIT ROLLBACK

这个命令除了之前的含义，还包括让SQL\*Plus在退出之前把所有还未提交的变化回滚处理。

## 4、执行必要的PL/SQL任务

### 4.1创建存储过程

比如想要创建一个存储函数，这个函数能够对一个字符串中的单词个数计数，你就可以使用CREATE FUNCTION语句：



只是一个简单的BEGIN-END块，要在SQL\*Plus中运行这个语句还需要一个结尾的斜杠，这个斜杠自己占一行。

如果DBA已经把Oracle的CREATE PROCEDURE权限赋予给你了（这个权限也允许创建函数），这个语句会让Oracle编译并把这个存储过程保存到你的模式下，一旦代码编译完成，你就会看到如下这样一个成功信息：Function created。

如果在你的模式下已经有另一个数据库对象也叫wordcount，比如一个表或者一个包，CREATE FUNCTION这个命令就会失败，并给出错误消息ORA-0955：name is already used by an existing object。这也是Oracle提供了OR REPLACE选项的原因，而且99%的情况下你都会使用这个选项。

CREATE OR REPLACE FUNCTION wordcount (str IN VARCHAR2)

RETURN PLS\_INTEGER

AS

same as before

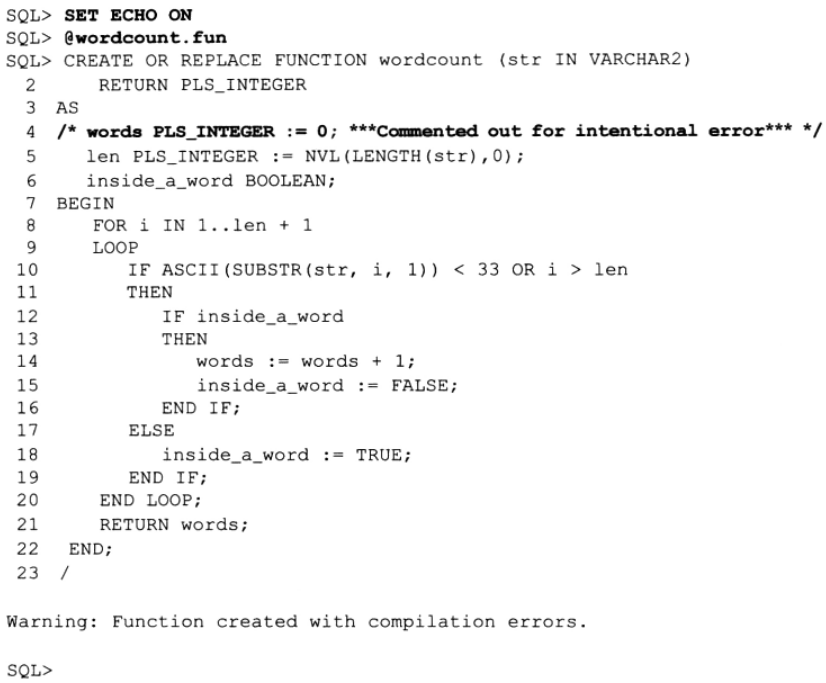
使用OR REPLACE选项，能够避免先删除然后重建程序的方法所带来副作用；换句话说，它能够保留你已经赋予其他用户或者角色的针对这个对象的权限。幸运的是，这个命令只会替换相同类型的对象，所以如果你想要创建的是一个函数，它不会自动删除一个叫做workcount的数据表。

和之前多次使用的匿名代码块一样，程序员通常会把这些语句保存到一个操作系统文件中。我为这个函数创建一个名叫wordcount.fun的文件，然后使用SQL\*Plus的@命令来运行这个文件：

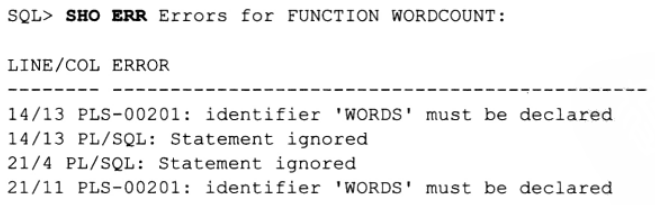
SQL> @wordcount.fun

Function created.

SQL\*Plus默认的时候并不回显脚本的内容。你可以开启SET ECHO ON，然后代码在屏幕上滚动，还带着数据库所分配的行号，这个选项在问题诊断时非常有用。让我们注释掉一个变量声明，这就给程序中引入一个错误：

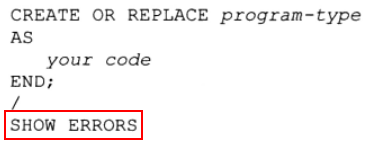


这个消息高速我们，函数已经创建了，但却带有编译错误，因此这个函数还不能使用。我们已经成功地把函数的源代码保存到数据库中，现在我们需要对数据库给出的错误信息的细节进行梳理。查看错误消息的完整文本的最快捷方式就是使用SHOW ERRORS命令，简写成SHO ERR：



其实在后台，SHOW ERRORS命令也仅仅是查询Oracle数据字典的USER\_ERRORS视图。

习惯上，在每一个创建PL/SQL存储过程的CREATE语句后面都要加上一个SHOW ERRORS命令。因此有个用于SQL\*Plus环境的，创建存储过程的“最佳实践”模板应该是这样的：



如果你的程序包含的是编译器能够检测到的错误，CREATE命令仍然会让Oracle把程序保存到数据库中，只不过程序的状态是无效的。但如果拼错了CREATE语法，数据库无法分辨你想做什么，因此也不会在数据库中保存代码。

### 4.2执行存储过程

两种调用存储过程的方法：

1. 封装在一个PL/SQL代码块中。
2. 或者使用SQL\*Plus的EXECUTE命令。

你也可以在存储过程中使用其它存储过程。比如，你可以在任何一处能够使用正式表达式的地方使用wordcount函数。下面代码简单演示了，如何使用一个较怪的输入（CHR(9)是一个，ASCII的“tab”字符）来测试wordcount函数的。

BEGIN

DBMS\_OUTPUT.PUT\_LINE(‘There are ’ || wordcount(CHR(9)) || ‘ words in a tab’);

END;

/

wordcount表达式的返回值是一个整数。PL/SQL会自动把整数转换成字符串，然后和其他两个字符串拼接起来。

在一个SQL语句里可以调用多个PL/SQL函数，下面是使用wordcount函数的一些例子：

* 在select子句中使用这个函数，以计算某一表列的词的数量；

SELECT Iisbn, wordcount(description) FROM books;

* 使用ANSI兼容的CALL语句，把函数的返回值绑定到一个SQL\*Plus变量，然后显示结果；

VARIABLE words NUMBER

CALL wordcount(‘some text’) INTO :words;

PRINT :words

* 和上一个例子相同，但是执行的是一个远程数据库的函数，指向这个远程数据库的数据库链接叫做test.newyork.ora.com：

CALL wordcount@test.newyork.ora.com:

* 函数是属于bob用户，但是以其他有相应授权的用户登录后执行函数：