

第2章 PL/SQL开发和运行环境

PL/SQL块可以在多种不同特点及功能的环境下运行。我们在本章主要讨论定位 PL/SQL引擎的有关问题。除此之外，我们还将讨论各种可用来开发 PL/SQL应用的环境，其中包括 Oracle本身提供的开发工具和由第三方开发商提供的开发工具。

2.1 应用模式和PL/SQL

一般的数据库应用可以分为三个部分：

- 用户界面，负责提供应用的外观和使用方式。该部分负责处理用户的输入信息和显示处理结果。
- 应用逻辑，这层的主要功能是控制应用的处理流程。
- 数据库，该层负责可靠地存储应用数据。

目前可以将上述三部分功能分配到不同位置的数据库应用设计模式主要有两类。

为了编译并运行一个PL/SQL块，程序员必须将该块提交给 PL/SQL引擎来处理。与Java语言的虚拟机相类似，PL/SQL引擎也是由编译器和运行时系统组成。借助于 Oracle公司和其他开发商提供的开发工具，PL/SQL可以用于应用的各个层次，并且PL/SQL引擎也可以宿主在不同的系统中。

2.1.1 两层模式

两层模式，即客户/服务器模式，是传统的应用设计模式。在这种模式中，应用由客户端程序和服务器端程序两部分组成。客户端负责处理用户界面，而服务器端管理数据库。这种模式的应用逻辑分为客户端和服务端两部分。通常，PL/SQL引擎驻留在服务器端，在个别情况下，PL/SQL引擎也可以驻留在客户端。

1. 服务器端的PL/SQL

从Oracle6.0版开始，PL/SQL就驻留在数据库服务器端，同时，该服务器也是 PL/SQL引擎的默认位置。由于数据库服务器可以处理 SQL语句，所以SQL语句和PL/SQL块都可以送到该服务器进行处理。一个客户应用，不管是用 Oracle开发工具实现的或使用其他开发工具编制的，都可以向数据库服务器提交SQL语句和PL/SQL块。SQL *Plus就是一个这种客户应用的案例，该程序可以在SQL提示符下接收交互输入的SQL语句和PL/SQL命令并将其送往服务器执行。

例如，我们可以假设在SQL *Plus与服务器建立了连接的情况下输入下列的SQL，PL/SQL命令：

```
SQL_PLSQL.sql
SQL> CREATE OR REPLACE PROCEDURE ServerProcedure AS
  2 BEGIN
  3 NULL;
  4 END ServerProcedure;
```

```
5 /
Procedure created.

SQL> DECLARE
2 v_StudentRecord students%ROWTYPE;
3 v_Counter BINARY_INTEGER;
4 BEGIN
5 v_Counter := 7;
6
7 SELECT *
8 INTO v_StudentRecord
9 FROM students
10 WHERE id = 10001;
11
12 ServerProcedure;
13
14 END;
15 /
PL/SQL procedure successfully completed.

SQL> UPDATE classes
2 SET max_students = 70
3 WHERE department = 'HIS'
4 AND course = 101;
1 row updated.
```

注意 可以在本书的CD-ROM中找到上面的例子的代码。本书中存储在CD-ROM中的程序案例的文件名称在该例子的第一行注释中给予了说明，读者可以根据注释中提供的XXX.sql在CD-ROM中找到对应的程序文件。CD-ROM中CODE目录下的reademe.html文件提供了对这些程序案例的说明。

图2-1演示了在服务器端的PL/SQL引擎对PL/SQL块的处理过程。客户应用可以向服务器提交PL/SQL块（该块可以带过程和包括调用服务器端存储过程的SQL语句），以及单独的SQL语句。如图所示，PL/SQL块和SQL语句通过网络送往服务器。一旦服务器收到了这些内容，SQL语句将直接进入服务器内含的SQL语句执行器，而PL/SQL块则送往PL/SQL引擎进行语法分析。在该块的运行期间，PL/SQL引擎负责执行过程语句（如赋值语句和存储过程调用）。对于该块中出现的SQL语句（如SELECT语句等），PL/SQL引擎将它们送往SQL语句执行器执行。

2. 客户端的PL/SQL

除了在服务器端的PL/SQL引擎外，几种Oracle开发工具也带有PL/SQL引擎。由于这些开发工具是运行在客户端的，所以它们的PL/SQL引擎也可以运行在客户端。这样一来，借助于客户端的PL/SQL支持，PL/SQL块中的过程语句就可以在本地运行，而没有必要送到服务器端。例如开发工具Oracle Forms（该工具是Oracle Developer的一部分）自身带有PL/SQL引擎；在Oracle Developer工具包中，如Oracle Reports也带有PL/SQL引擎。需要指出的是这种引擎与PL/SQL服务器端的引擎有所不同。PL/SQL块只能出现在客户端应用中，并且该块必须用开发工具来编制。假设一个Oracle Forms应用包括了触发器和过程，这些语句都在客户端执行。只有该程序中的

SQL语句和调用服务器端存储子程序的语句被送往服务器进行处理。如图 2-2所示，客户端的PL/SQL引擎可以处理过程语句。

与服务器端的PL/SQL一样，应用程序提交的单独的SQL语句（如UPDATE语句）直接通过网络送往服务器端的SQL语句执行器。不同的是，PL/SQL块是在客户端直接处理。任何过程语句（如赋值语句）的处理都不会引起网络传输。PL/SQL块中的SQL语句要提交给SQL语句执行器，对服务器端的存储子程序的调用则是送到服务器端的PL/SQL引擎执行。

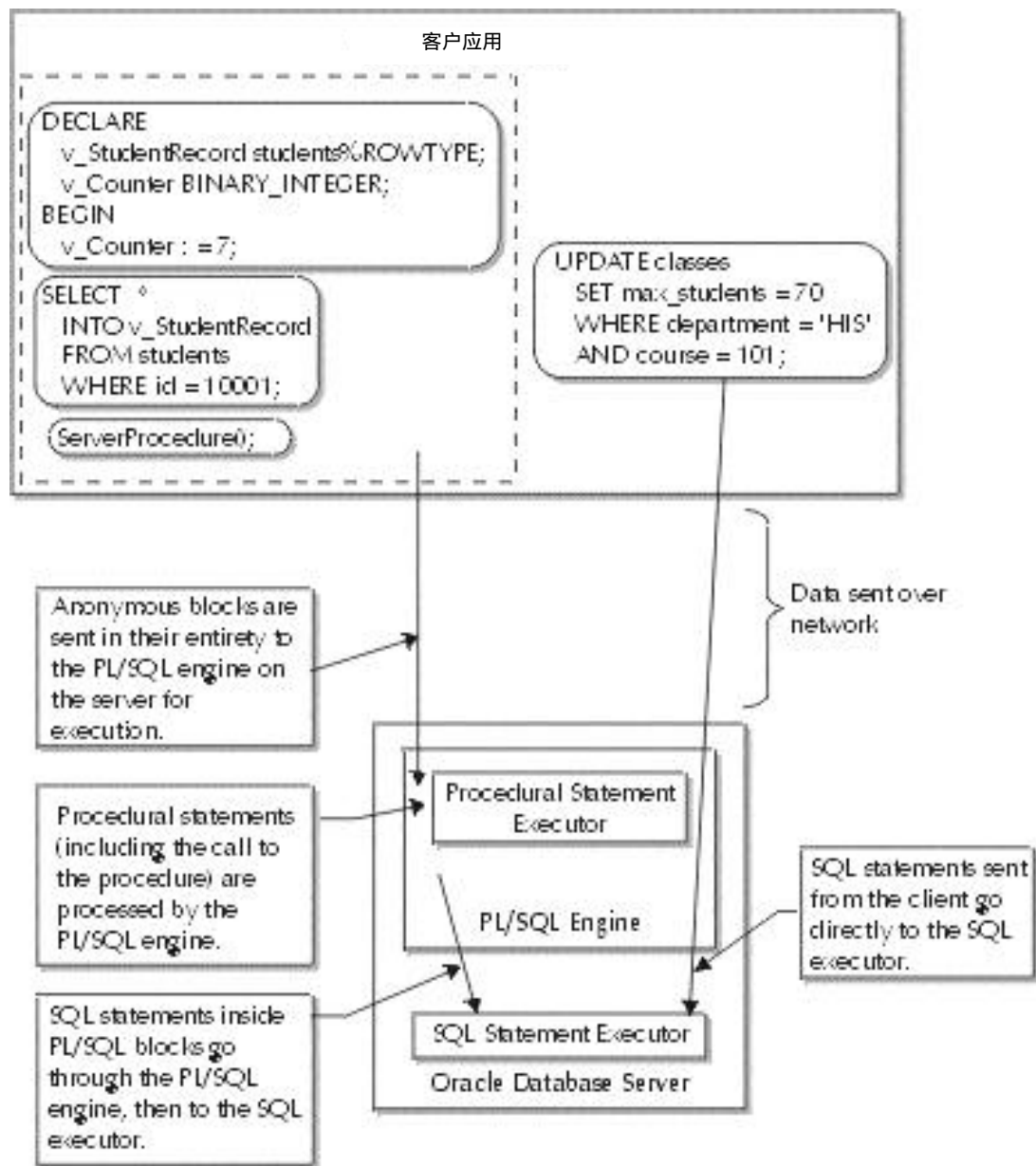


图2-1 服务器端PL/SQL引擎

3. Oracle 预编译器

程序员可以使用 Pro *C/C++ 和 Pro *COBOL 一类的 Oracle 预编译器创建运行在服务器端的应用程序。用这种方法实现的应用将不包括 PL/SQL 引擎，因此，由这种应用提交的 SQL 和 PL/SQL 语句都要送到服务器进行处理。

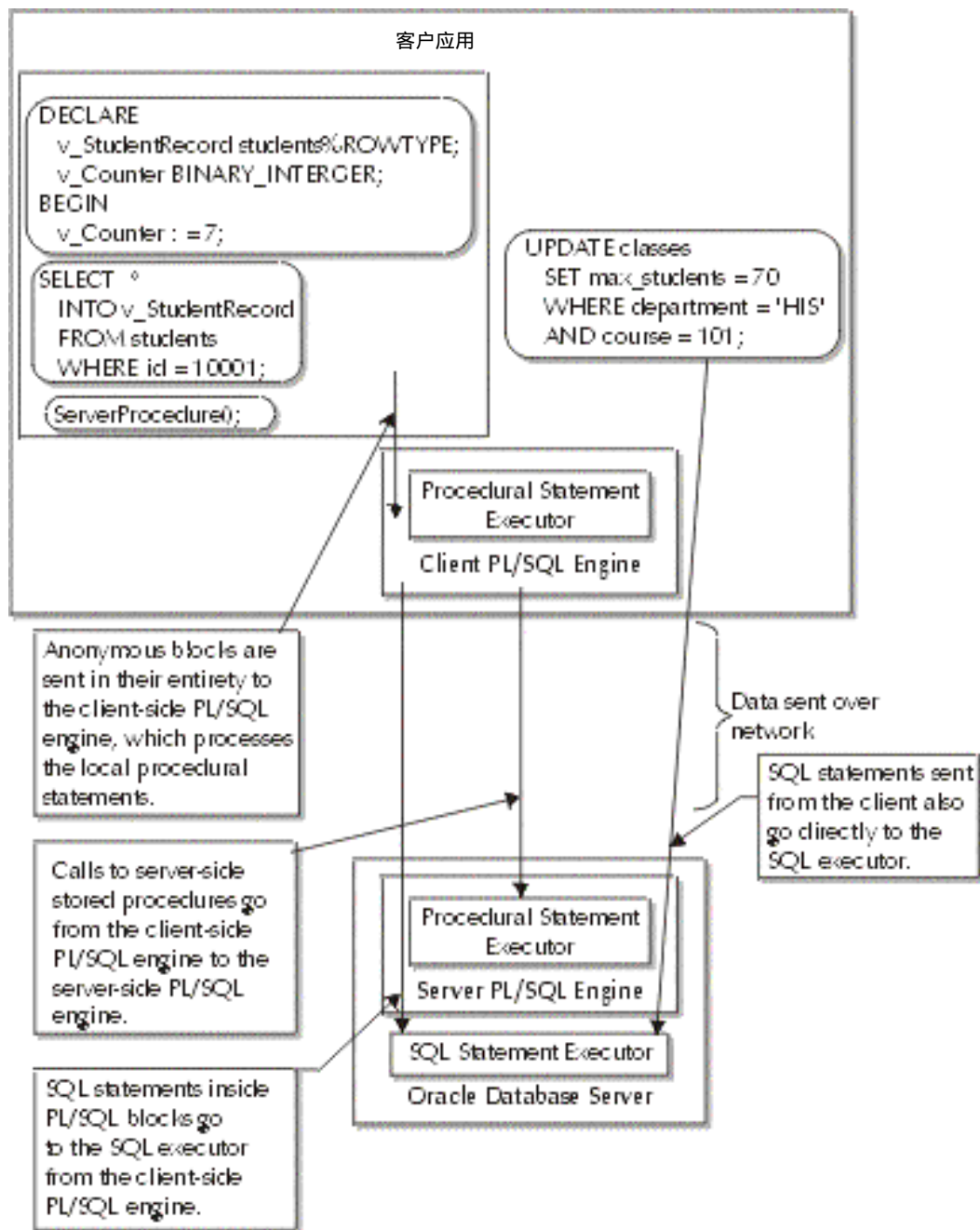


图2-2 客户端PL/SQL引擎

预编译器本身包括了 PL/SQL 引擎，该引擎在预编译中用来校验应用代码中匿名块的语法和语义是否正确。预编译功能是 Oracle 开发工具的一大特点。

4. 引擎间的通信

在图2-2所示的流程图中，有两个各自独立但又相互通信的 PL/SQL 引擎。例如，一个运行在客户端 PL/SQL 下的报表（Form）中的触发器可以调用在服务器端 PL/SQL 下运行的存储过程，这一类的网络通信是通过远程过程调用实现的。借助于类似的机制，通过数据库连接可以实现不同 PL/SQL 引擎之间的通信。

在上述情况下，不同引擎中的 PL/SQL 对象可以相互依赖。这种存在依赖关系的类型工作方式与在同一个数据库中的 PL/SQL 对象几乎完全一样，不同之处是程序中必须提供一些防止产生误解的声明。本书的第5章提供了进一步的信息。

总的来说，两个 PL/SQL 引擎可以是不同的版本。例如，Oracle 开发器 1.2 版使用了 PL/SQL 第 1 版，而服务器使用了 PL/SQL 版本 2（如果使用了 Oracle8，就需要使用 PL/SQL 版本 8），这就意味着 PL/SQL 版本 2 或更高的版本提供的功能，如用户自定义表和记录，定长的 CHAR 数据类型和其他一些功能可能不会出现在客户端引擎中。尽管 Oracle 开发器第二版以上的工具中包括了 PL/SQL 8.0 版，但在引擎之间仍然存在着版本差别，因此，每一个版本可用的功能也不一样。

2.1.2 三层模式

在三层模式中，用户界面，应用逻辑和数据库是三个各自独立的部分。该模式下的客户是典型的瘦客户类型，如浏览器一类的客户软件。应用层逻辑全部位于称为应用服务器的独立层中。在这种环境下，PL/SQL 引擎通常只放置在服务器中。

Oracle 应用服务器（OAS）具有一般应用服务器的全部功能。通过 PL/SQL 盒式磁带，读者可以运行服务器上的存储过程并返回 HTML 页面形式的处理结果，这项功能可以借助于 OAS 提供的 PL/SQL Web 工具箱实现。图 2-3 演示了三层模式的内部结构。有关 PL/SQL 盒式磁带和 Web 工具箱的进一步介绍，请读者参考 Oracle 文档。

2.2 PL/SQL 开发工具介绍

开发调试 PL/SQL 应用可以使用多种不同的开发工具，每种开发工具都有其优点与不足。表

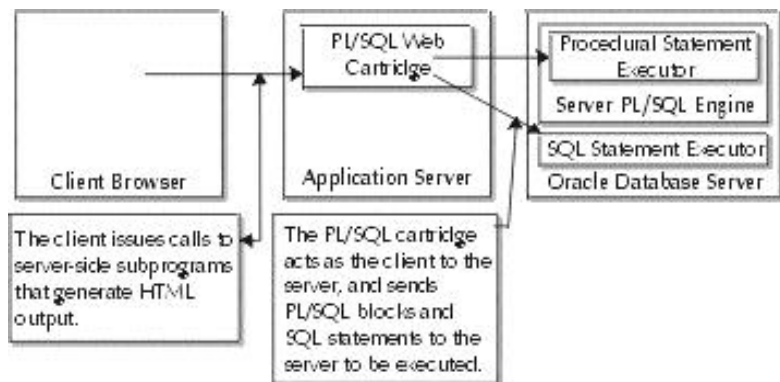


图2-3 三层模型示意图

2-1介绍了我们本章详细讨论的开发工具的基本情况。从表中我们可以看出，SQL *Plus是Oracle公司与服务器一起提供的开发工具，其他的开发工具则是由第三方开发商提供的。本书的CD-ROM中带有第三方开发工具的试用版，我们将在下面几节中将介绍这些开发工具的使用方法。在第3章，我们将详细介绍这些开发工具的调试功能。

注意 CD-ROM中的开发工具存放在Development Tools目录中，CD-ROM中根目录下的readme.html文件中有对开发工具的说明。

表2-1 PL/SQL开发环境一览

工具名称	开发商	Web站点地址	是否随CD-ROM提供
SQL *Plus	Oracle公司	www.oracle.com	不
Rapid SQL	Embarcadero技术公司	www.embarcadero.com	是
XPEDITER/SQL	Compuware	www.compuware.com	是
SQL Navigator	Quest Software	www.quest.com	是
TOAD	Quest Software	www.quest.com	是
		www.toadsoft.com	
SQL-Programmer	Sylvain Faust International	www.sfi-software.com	是

为了保持一致性，下面讨论的每一种开发工具都采用相同的模式，即使用几种不同类型的PL/SQL对象案例来进行说明，以便了解每一种工具在同一个环境下的优缺点。表2-2描述了样本模式的具体内容，该样本模式的每一项都可以在执行安装脚本relTables.sql后运行表中指示的脚本文件来自动创建。本书的在线文件Setup.sql也可以用来运行这些脚本文件。需要指出的是，在创建外部过程和函数时，需要提供系统特权CREATE LIBRARY。

注意 在下面几节中，每种开发工具都带有图形窗口。由于受到本书空间的限制，某些工具的图形窗口只能存放在本书的CD-ROM中，有关这些图形窗口的说明，请看CD-ROM中目录online Chapters中的文件ch02ScreenShots.html。

表2-2 样本模式一览表

对象名称	对象类型	可运行的脚本
AddNewStudent	过程	ch04/AddNewStudent.sql
AlmostFull	函数	ch04/AlmostFull.sql
ModeTest	过程	ch04/ModeTest.sql
ClassPackage	包和包体	ch04/ClassPackage.sql
Point	对象类型和类型体	ch12/Point.sql
OutputString	外部过程和函数	ch10/OutputString.sql

2.2.1 SQL *Plus

SQL *Plus可能是最简单的PL/SQL开发工具。该工具允许用户交互式地从输入提示符输入SQL语句和PL/SQL块，这些输入的语句直接送到数据库执行，并将执行结果显示在终端屏幕上。该工具支持字符环境，没有内置的本地PL/SQL引擎。

一般，SQL *Plus是与Oracle服务器捆绑销售，并作为标准 Oracle安装过程的一部分执行。读者可以参阅SQL *Plus用户指南和参考手册来了解该工具的详细说明和有关命令。

SQL *Plus的命令不区分大小写字母。例如，下面三个命令都可以声明连接变量：

```
SQL> VARIABLE v_Num NUMBER
SQL> variable v_Char char(3)
SQL> vaRIAbLe v_Varchar VarCHAR2(5)
```

1. 连接数据库

在SQL *Plus下输入SQL或PL/SQL命令之前，必须先实现与数据库服务器的连接。下面是实现数据库连接的两种常用方法：

- 在SQL *Plus命令行输入用户标识（Userid）和口令，或输入连接字符串。
- 进入SQL *Plus后使用CONNECT语句。

在下面的连接例子中，读者可以看到，如果没有定义口令的话，SQL *Plus将不会为用户提示口令内容并且不将输入字符回送到屏幕显示。

```
$ sqlplus example/example
```

```
SQL*Plus: Release 8.0.6.0.0 - Production on Wed Nov 3 10:29:11 1999
(c) Copyright 1999 Oracle Corporation. All rights reserved.
```

```
Connected to:
```

```
Oracle8 Enterprise Edition Release 8.0.6.0.0 - Production
With the Objects option
PL/SQL Release 8.0.6.0.0 - Production
```

```
SQL> exit
```

```
Disconnected from Oracle8 Enterprise Edition Release 8.0.6.0.0 -Production
With the Objects option
PL/SQL Release 8.0.6.0.0 - Production
```

```
$ sqlplus example
```

```
SQL*Plus: Release 8.0.6.0.0 - Production on Wed Nov 3 10:29:15 1999
(c) Copyright 1999 Oracle Corporation. All rights reserved.
```

```
Enter password:
```

```
Connected to:
```

```
Oracle8 Enterprise Edition Release 8.0.6.0.0 - Production
With the Objects option
PL/SQL Release 8.0.6.0.0 - Production
```

```
SQL> connect example/example
```

```
Connected.
```

```
SQL> connect example/example@v806_tcp
```

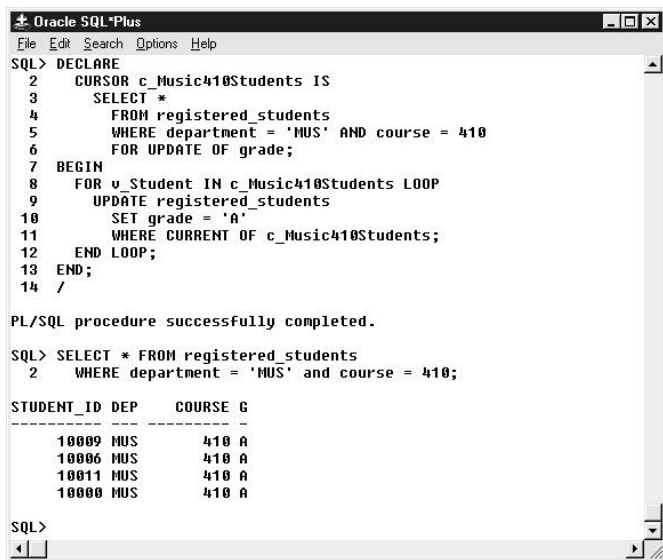
```
Connected.
```

2. SQL *Plus中的块操作

当在SQL *Plus中执行一个SQL语句时，语句末尾的分号标识了该语句的结束。该分号不属于语句本身，它是一个独立的语句终止符。当SQL *Plus读到一个分号时，它就知道该语句已经

结束并将其发送到数据库执行。另一方面，对于 PL/SQL块来说，分号则是块本身的语法部分，而不是语句终止符。当输入关键字 DELCARE或BEGIN时，SQL *Plus就能够检测到该关键字并确认正在输入的是 PL/SQL块，不是SQL语句。在这种情况下，SQL *Plus仍然需要确认输入的 PL/SQL块何时结束，我们使用一个正斜杠来表示块结束，这种正斜杠是 SQL *Plus RUN命令的缩写形式。

请注意，图 2-4中更新表 registered_students的PL/SQL块之后有一个正斜杠。该块后面的 SELECT语句由于使用了分号而不用再输入斜杠。（如果需要的话，读者也可以在 SQL语句中使用斜杠来代替分号。）



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> DECLARE
2  CURSOR c_Music410Students IS
3  SELECT *
4  FROM registered_students
5  WHERE department = 'MUS' AND course = 410
6  FOR UPDATE OF grade;
7  BEGIN
8  FOR v_Student IN c_Music410Students LOOP
9  UPDATE registered_students
10 SET grade = 'A'
11 WHERE CURRENT OF c_Music410Students;
12 END LOOP;
13 END;
14 /

PL/SQL procedure successfully completed.

SQL> SELECT * FROM registered_students
2  WHERE department = 'MUS' and course = 410;

STUDENT_ID DEP    COURSE G
-----
10009 MUS        410 A
10006 MUS        410 A
10011 MUS        410 A
10000 MUS        410 A

SQL>
```

图2-4 在SQL *Plus中输入PL/SQL块

3. 替换和连接（bind）变量

由于PL/SQL是服务器端的专用语言，所以该语言不支持用户输入和输出。文件输入输出操作是通过包 UTL_FILE实现的（需PL/SQL 2.3版和更高版本支持，本书第7章有专门介绍），借助于BFILE接口，Oracle8可以读入外部文件（第15～16章有专文论述）。除此之外，与SQL *Plus一样，使用DBMS_OUTPUT包可以实现有限的屏幕输出操作（详细内容请参阅第3章）。

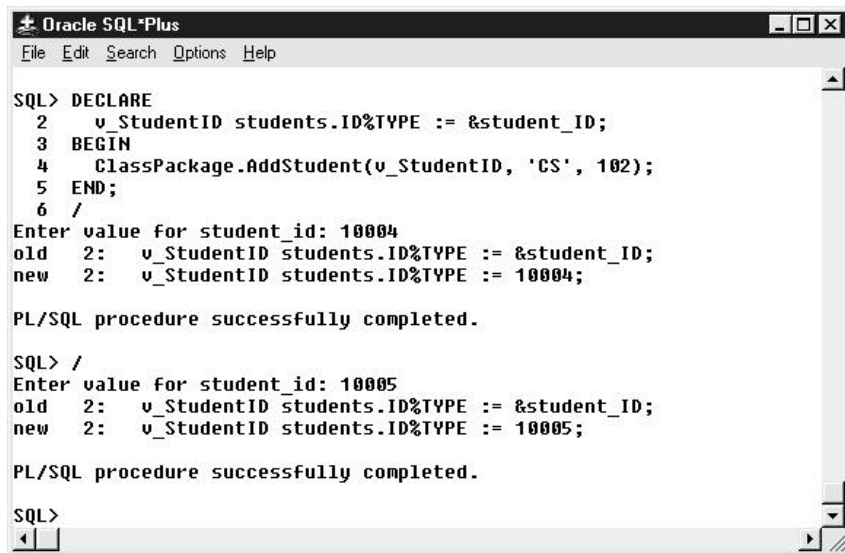
上述的各种方法都不支持接收用户的输入，其主要原因是 PL/SQL所在的运行环境所致。例如，Pro *C程序可以使用C语言的 I/O库函数功能来接收用户输入并将其传入 PL/SQL中。

SQL *Plus提供了两种不同类型的变量来接收用户的输入并通过多个运行存储信息，这两种变量就是替换和连接变量。

替换变量 替换变量在 PL/SQL块或SQL语句中是用字符符号“&”描述的（使用 SET DEFINE命令可以指定替换“&”的字符）。SQL *Plus在将PL/SQL块或SQL语句发送到服务器前要对变量进行彻底的原文替换，类似于C语言对宏的处理。

图2-5中的PL/SQL块中使用了替换变量。该块运行了两次，每次都对变量 v_StudentID给予了不同的初始值。用户分别输入了数值 10004和10005,这两个数值在该块中被 &student-id替换。

需要注意的是替换变量实际上不会占用内存空间。SQL *Plus在把该块发送到数据库执行前使用用户输入的数值来置换替换变量。由于这种原因，替换变量只能用于输入。下面讨论的连接变量则可以用于输入和输出双向操作。



```
Oracle SQL*Plus
File Edit Search Options Help

SQL> DECLARE
  2   v_StudentID students.ID%TYPE := &student_ID;
  3 BEGIN
  4   ClassPackage.AddStudent(v_StudentID, 'CS', 102);
  5 END;
  6 /
Enter value for student_id: 10004
old  2:   v_StudentID students.ID%TYPE := &student_ID;
new  2:   v_StudentID students.ID%TYPE := 10004;

PL/SQL procedure successfully completed.

SQL> /
Enter value for student_id: 10005
old  2:   v_StudentID students.ID%TYPE := &student_ID;
new  2:   v_StudentID students.ID%TYPE := 10005;

PL/SQL procedure successfully completed.

SQL>
```

图2-5 SQL*PLUS替换变量

提示 假设现在从提示符SQL>下输入下列SQL语句：

```
SQL> SELECT *
      FROM students
      WHERE first_name = &first_name;
```

输入该语句后，当SQL *Plus提示输入时，用户必须要在输入的字符串两端使用单引号，如‘SCOTT’。现在请比较下面的语句：

```
SQL> SELECT *
      FROM students
      WHERE first_name = '&first_name';
```

在这种情况下，用户不必使用单引号，因为该语句中已经有了单引号。单引号的使用与否取决于要替换的内容。

连接变量 我们在上面讲过，替换变量不会占用内存空间，然而，SQL *Plus可以分配内存空间供PL/SQL块和SQL语句内部使用。因为这种存储空间位于块的外部，所以它可以为一个以上的连续的块和SQL语句共用，并且该存储空间的内容还可以在块结束退出后打印显示，我们描述的这种变量就是连接变量。该类变量的图解说明在CD-ROM中提供的图CD2-1中。连接变量v_Count是用SQL*Plus的VARIABLE命令为其分配内存的。请注意，VARIABLE命令只能在SQL提示符下使用，它不能用在PL/SQL块中。在块内部，连接变量是用起始处的冒号来定界的。在该块结束后，命令PRINT将显示该变量的值。SQL*Plus允许使用的合法连接变量类型包括VARCHAR2,CHAR和NUMBER。连接变量REFCURSOR只能用于SQL *Plus3.2版和更高版本。

连接变量 NCHAR, NVARCHAR2, CLOB 和 NCLOB 只能用于 SQL *Plus 8.0 版及更高版本。如果没有特殊指定连接变量类型 VARCHAR2, CHAR, NCHAR 或 NVARCHAR2 的长度, 这些变量的默认长度都为 1, NUMBER 类型的连接变量不受精度和比例的限制。

4. 变量和数据库对象

由于替换变量在 PL/SQL 块被发送到服务器之前就被进行了替换, 所以这类变量可以作为 SQL 语句和数据库对象的语法部分使用, 而连接变量则不能在这种情况下使用。下面的 SQL *Plus 对话演示了这种规则。

```
--节选自在线代码Variables.sql
```

```
SQL> SELECT &columns
```

```
2 FROM classes;
```

```
Enter value for columns: department, course
```

```
old 1: SELECT &columns
```

```
new 1: SELECT department, course
```

```
DEP COURSE
```

```
--- -----
```

```
HIS 101
```

```
HIS 301
```

```
CS 101
```

```
ECN 203
```

```
CS 102
```

```
MUS 410
```

```
ECN 101
```

```
NUT 307
```

```
MUS 100
```

```
9 rows selected.
```

```
SQL> SELECT first_name, last_name
```

```
2 FROM students
```

```
3 WHERE &where_clause;
```

```
Enter value for where_clause: ID = 10001
```

```
old 3: WHERE &where_clause
```

```
new 3: WHERE ID = 10001
```

```
FIRST_NAME LAST_NAME
```

```
-----
```

```
Margaret Mason
```

```
SQL> -- But a bind variable cannot be used in this manner:
```

```
SQL> VARIABLE where_clause VARCHAR2(100)
```

```
SQL>
```

```
SQL> BEGIN
```

```
2 :where_clause := 'WHERE ID = 10001';
```

```
3 END;
```

```
4 /
```

```
PL/SQL procedure successfully completed.
```

```
SQL> PRINT :where_clause
```

```
WHERE_CLAUSE
```

```
-----  
WHERE ID = 10001
```

```
SQL> SELECT first_name, last_name
```

```
2 FROM students
```

```
3 WHERE :where_clause;
```

```
WHERE :where_clause
```

```
*
```

```
ERROR at line 3:
```

```
ORA-00920: invalid relational operator
```

读者可以使用动态SQL在运行期间建立SQL语句和PL/SQL块，并使用PL/SQL变量来构造这些语句。有关详细内容，请看本书的第8章。

5. 使用EXECUTE命令调用存储过程

存储过程调用只能从PL/SQL块的可执行语句部分或其异常处理部分中执行。SQL *Plus为这种调用提供了一个有用的简写方式，即命令EXECUTE。该命令实现的功能是在其命令参数之前加上BEGIN，而在参数之后加上END。并同时在调用语句后加入一个分号，表示调用语句结束。经过这种处理后形成的块就被提交给数据库执行。例如，假设我们在SQL命令提示符下输入下列命令：

```
SQL> EXECUTE ClassPackage.AddStudent(10006,'CS',102)
```

则SQL *Plus就可以生成下面的PL/SQL块：

```
BEGIN ClassPackage.AddStudent(10006,'CS',102); END;
```

该块被发送到数据库执行。关键字EXECUTE之后的分号是该命令的选择项，这对所有的SQL *Plus命令都是适用的。并且如果使用了该分号的话，它将被忽略不记，就像命令PRINT，VARIABLE一样，EXECUTE是SQL *Plus的命令，它不能用在PL/SQL块内部。

提示 EXECUTE命令并不在SQL缓冲区中存储生成的匿名块，但如果匿名块是直接输入的话，该命令可以实现这种功能。

6. 使用文件

SQL *Plus可以把当前的PL/SQL块或SQL语句保存在文件中，并在需要时，将该文件读入系统执行。SQL *Plus的这种功能对于实现先开发PL/SQL程序后运行调试来说非常有用。例如，读者可以把命令CREATE或REPLACE存储在一个文件中。利用这种方法，任何对该过程的修改都可以通过该文件实现。为了保存数据库的修改，我们可以简单地把该文件读如到SQL *Plus中。文件中可以存放一个以上的命令。

SQL *Plus的GET命令将文件从磁盘读入到本地缓冲区中存放。输入一个正斜杠就可以运行读入的文件，就像该文件中的命令是从键盘直接输入的一样。如果文件的结尾处有一个正斜杠的话，就可以使用GET命令的缩写形式@将该文件读入并执行。例如，假设文件file.sql包含下列

命令行：

```
--节选自在线代码File.sql
BEGIN
  FOR v_Count IN 1..10 LOOP
    INSERT INTO temp_Table (num_col, char_col)
      VALUES (v_Count, 'Hello World!');
  END LOOP;
END;
/

SELECT * FROM temp_table;
```

现在，我们可以从SQL提示符模式下使用下面的命令运行该文件。

```
SQL> @file
```

执行该文件的输出结果在本书 CD-ROM中的图CD2-2所示。命令SET ECHO ON（该图中的第一条命令）告诉SQL *Plus在读入该文件时把其内容送到屏幕显示。

7. 使用SHOW ERRORS命令

在创建存储过程时，有关该过程的信息存储在数据字典中。所有的编译错误都存储在user_errors的数据字典中。SQL *Plus提供了一个可以查询该数据字典并报告错误的命令 SHOW ERRORS。本书的CD-ROM上的图CD2-3演示了该命令的使用方法。命令 SHOW ERRORS可以用在SQL *Plus报告了下列错误信息后：

```
Warning: Procedure created with compilation errors.
```

2.2.2 Rapid SQL

由Embarcadero Technologies公司开发的Rapid SQL（快速SQL）提供了图形用户界面的开发环境，其主要功能如下：

- PL/SQL和SQL语句的自动格式化
- PL/SQL调试器
- 支持Oracle8的对象类型和表分区
- SQL任务调度
- 工程管理
- 支持Windows活动脚本（active scripting）
- 支持第三方版本控制系统
- 集成数据库开发和Web程序设计

有关该工具的安装和使用方法，请参考有关在线帮助。

1. 连接数据库

当第一次启动运行Rapid SQL时，该程序将显示如图2-6所示的窗口。该窗口左面的窗格中显示了按数据源分类的可浏览数据库对象。右面的窗格是工作窗口，用来显示正在浏览的不同类型的对象，该窗口中的数据源记录了远程数据库的所有相关信息，如数据库类型、用户标识、

口令和连接信息。当 Rapid SQL 第一次运行时，它将通过检索当前计算机上的 SQL *Net 或 Net8 配置自动地来发现可用的数据源。双击某个特殊的数据源将会启动一个对话框来接收连接使用的用户标识和口令，读者可以参考 CD-ROM 中的图 CD2-4 来了解该操作过程。除此之外，也可以通过数据源菜单增加新的数据源。

如果选择已建立的数据源的话，Rapid SQL 将自动使用用户标识和口令，而不需要再次输入。Rapid SQL 允许同时打开多个数据库连接。

2. 浏览数据库对象

一旦建立了数据库连接，我们就可以浏览左窗格中的对象。单击对象类型将使显示的目录中增加所选对象，这时，我们就可以进一步观察列表目录中某个单独的对象。双击某个对象将会启动创建该对象的 SQL 或 PL/SQL。例如，图 2-7 中显示了对象浏览器中的包和表以及在右面的工作区中显示了表 classe 的 DDL。

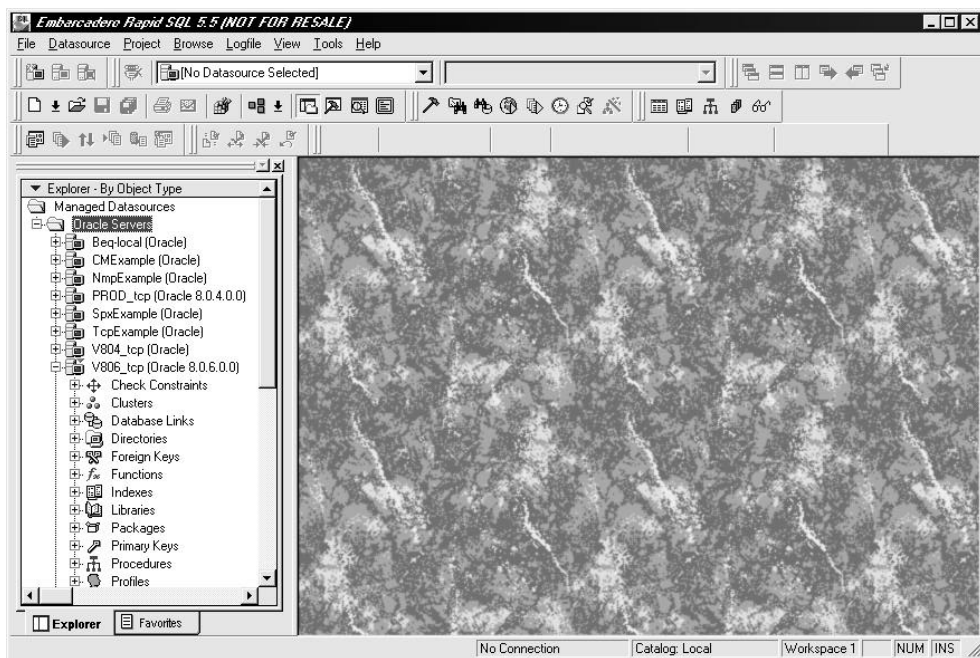


图2-6 Rapid SQL的主窗口

3. 编辑器类型

Rapid SQL 提供了两类不同的窗口供用户输入 SQL 语句和 PL/SQL 块，每种窗口适用于不同的任务：

- SQL 编辑器是通用的编辑器，适用于输入 SQL DML 和 DDL 语句，以及匿名 PL/SQL 块和 CREATE 语句。
- DDL 编辑器允许用户选择待创建的对象类型并自动地用结构对其进行填充。

这两种编辑器可以从主窗口的 File | New 的子菜单中选择。CD-ROM 中的图 CD2-5 图示了使

用匿名块的PL/SQL编辑器。单击绿色的三角形将把该块发送到服务器运行。

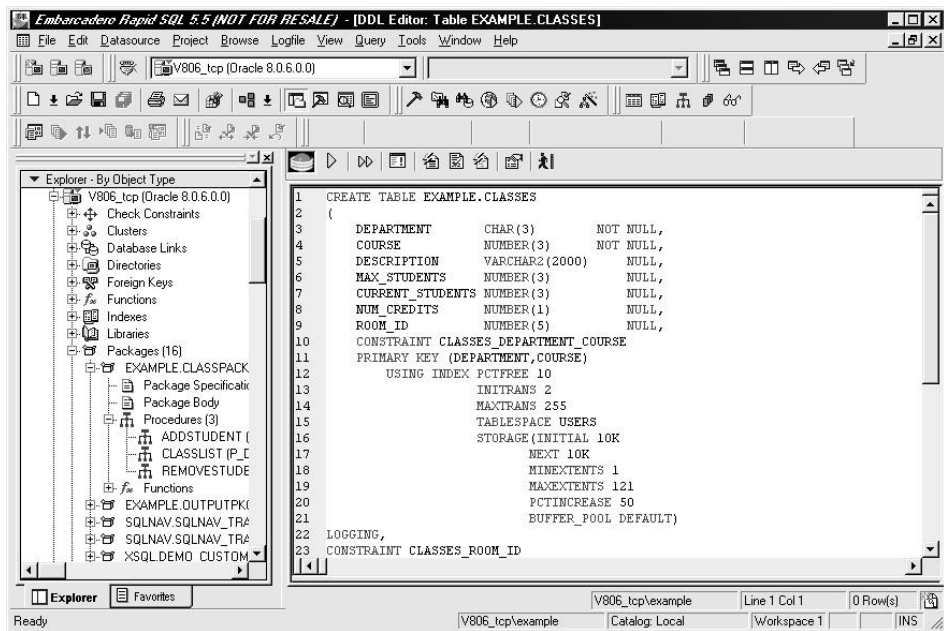


图2-7 浏览表classes

4. 自动格式

任何编辑器都可以对SQL和PL/SQL语句进行自动格式。自动格式操作包括缩进和各种语法元素的识别。这种功能非常有助于根据程序员自己的程序设计风格来格式化 PL/SQL块。CD-ROM中的图CD2-6中列举了自动格式的选择项。

5. 观察错误

把PL/SQL块提交给服务器后，只要单击 Errors工具条就可以显示任何错误信息。这时浏览器中与错误对应的对象上也标有 X标记，表示该对象有语法错误。CD-ROM中图2-7演示了显示错误的窗口图形。错误将显示在 PL/SQL文本下面的窗格中。单击某个错误将使光标移动到产生错误的语句所在的位置。

6. 作业调度

Rapid SQL将Windows98和NT提供的作业调度合并入该软件中，这种功能的引入可以使用户在确定的时间独立于 Rapid SQL Pro运行作业。

7. 集成Web开发功能

借助于附加的编辑器，Rapid SQL还可以编辑Java代码和HTML页面。有关这方面的信息，请参阅在线文档。

2.2.3 XPEDITOR/SQL

XPEDITOR/SQL是由Compuware 公司发行的开发工具，它也提供图形用户界面开发环境。

该工具的主要功能如下：

- 自动格式PL/SQL和SQL语句
- 提供PL/SQL调试器
- 支持Oracle8对象类型
- 工程管理
- 对任何应用提供调试和跟踪功能
- 支持第三方版本控制系统

1. 连接数据库

当XPEDITER/SQL首次运行时，该软件将提示用户建立数据库连接。为了方便建立数据库连接，我们可以事先存储不同数据库连接所需的配置信息，这些信息包括用户标识和数据库连接字符串。可以将用户标识的格式指定为“用户标识 / 口令”的格式，这样一来，用户口令也可以存储在配置文件中。建立数据库连接的对话框的窗口在本书 CD-ROM中的图CD2-8显示。XPEDITER/SQL允许同时建立多个连接。

2. 服务器端的安装

为了正确的使用XPEDITER/SQL，必须在服务器端进行安装操作。该安装将创建一个包括存储XPEDITER/SQL所需信息的表的数据库用户。该数据库用户的创建可以在安装过程中完成，也可以在安装结束后，使用数据库安装向导来实现。该向导的窗口在 CD-ROM中的图CD2-9中演示。数据库安装向导允许用户安装，卸载或为多个数据库更新服务器端对象。

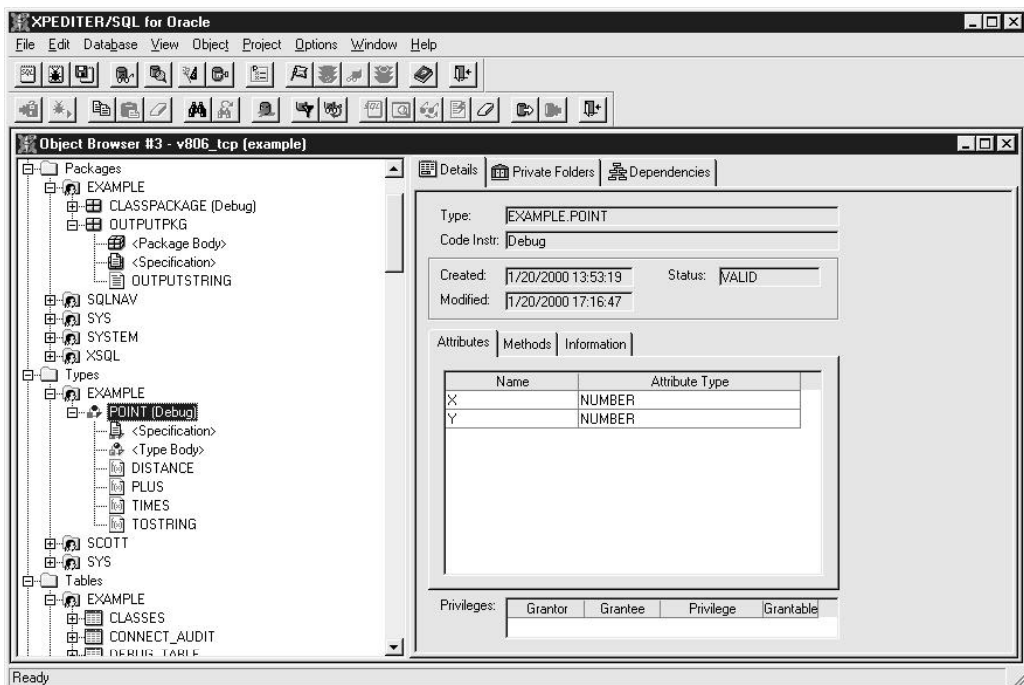


图2-8 浏览对象Point类型的窗口

3. 浏览数据库对象

XPEDITER/SQL的对象浏览窗口是用来观察数据库对象信息和对话特权的专用窗口。XPEDITER/SQL允许用户为同一个数据库连接打开多个浏览器，但每个浏览器只能显示一个连接的信息。对象浏览器的窗口如图 2-8所示。该窗口的左窗格显示了对象的树形目录，用户可以查看希望的对象类型和拥有者。该窗口右面的窗格则显示选择的对象的有关信息，包括该对象是否带有经过编译的调试信息和其他相关信息。（用户可以通过选择菜单项 Object | Debug On 或 Object | Debug Off来编译带有或不带有调试信息的对象。）

4. 编辑数据库对象

双击对象浏览器中的某个对象将会在 SQL的Notepad编辑器窗口中打开该对象文件。SQL的Notepad编辑器窗口可以用来编辑所有类型的数据库对象，及 PL/SQL存储过程和匿名块。例如，图2-9中所示的窗口显示了在 Notepad编辑器中对象Point的类型说明。编辑器对 PL/SQL代码自动进行了格式设置。除此之外，单击该窗口中的红色的三角形按钮可以运行编辑器中的 SQL语句，如果单击窗口中的绿色三角形按钮可以对 SQL语句进行调试。

5. 显示错误

单击位于SQL Notepad编辑器底部窗格中的SQL Errors标签，用户可以查看编译后报告的对象错误信息。这时显示的是当前对象或块的错误信息。如果单击某个错误，该编辑器将把光标定位到发生错误所在过程的语句行上。CD-ROM中图CD2-10显示了过程TooManyErrors的错误。除此之外，非法的对象也在对象浏览中被标明。

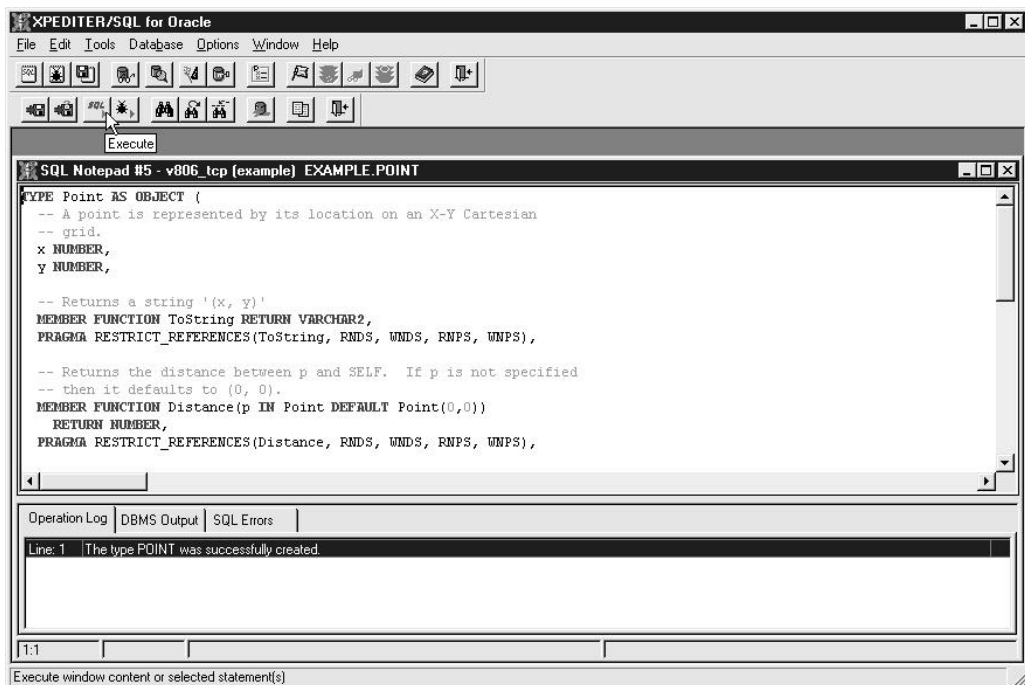


图2-9 编辑对象Point的窗口

6. 使用模板

XPEDITER/SQL的另一个常用的工具是模板编辑器。该编辑窗口可以在 SQL Notepad编辑器窗口中通过选择Tools | Templates Editor子菜单激活。在该窗口中，用户可以把予定义的 PL/SQL 代码段插入到激活的Notepad中。除了可以使用创建新过程，函数，触发器，包和类型外的语法外，模板编辑器还包括对公用内置包（如 DBMS_SQL,DBMS_OUTPUT等）的调用语句。本书 CD-ROM中的图CD2-11有模板编辑器的图形窗口的样板。除此之外，在该窗口中双击某个模板将把该模板的文本复制到Notepad窗口中。

7. DBPartner

我们讨论的XPEDITER/SQL的最后一个功能是工具 DBPartner。该实用工具允许用户从任何程序中捕捉SQL语句并在XPEDITER/SQL中对其进行编辑和调试。利用这种功能，用户可以在没有源程序的情况下调整或调试程序。

2.2.4 SQL浏览器

SQL浏览器是由Quest Software公司提供的开发工具。它也提供了图形用户界面，其主要功能如下：

- 自动格式PL/SQL和SQL语句
- 提供PL/SQL调试器
- 数据库浏览器
- 支持Oracle8对象类型和Oracle8i类型
- 代码模板
- 支持第三方版本控制系统

1. 连接数据库

与上述XPEDITER/SQL开发工具一样，SQL浏览器在其第一次启动运行时也要请求用户建立数据库连接。SQL浏览器可以自动保存连接配置文件，但用户的口令不能与该文件一起保存。用来建立数据库连接的程序窗口如 CD-ROM中图CD2-12所示。如果用户在初次启动时没有建立数据库连接，该工具将在一个编辑窗口或浏览窗口中用同一个对话框请求用户建立连接。SQL浏览器支持同时对不同数据库的多连接操作。

2. 安装服务器端的对象

SQL浏览器的多个选择项要求在服务器中创建一个名为 SQLNAV的用户。CD-ROM中图CD2-13所示的服务器端安装向导可以帮助建立必要的用户和对象。该向导可以作为安装程序的一部分运行，也可以在安装完成后，选择菜单项 Tools | Server Side Installation Wizard启动该程序运行。服务器端的安装是实现 SQL浏览器的程序员组编程，第三方版本控制，和 SQL浏览器Tuner等功能的必要步骤。

3. 浏览数据库对象

SQL浏览器的DB 浏览器窗口是用来浏览数据库对象的工具。用户可以从该窗口提供的树形视图中选择要查看的对象的类型和所有者。该浏览器的一个独特的功能是提供了过滤器。使用过滤器，用户可以选择希望显示的那些对象类型。该浏览器提供了多个预定义的过滤器，用户

也可以创建自己的过滤器。图 2-10所示的DB 浏览器窗口中使用了三个过滤器。

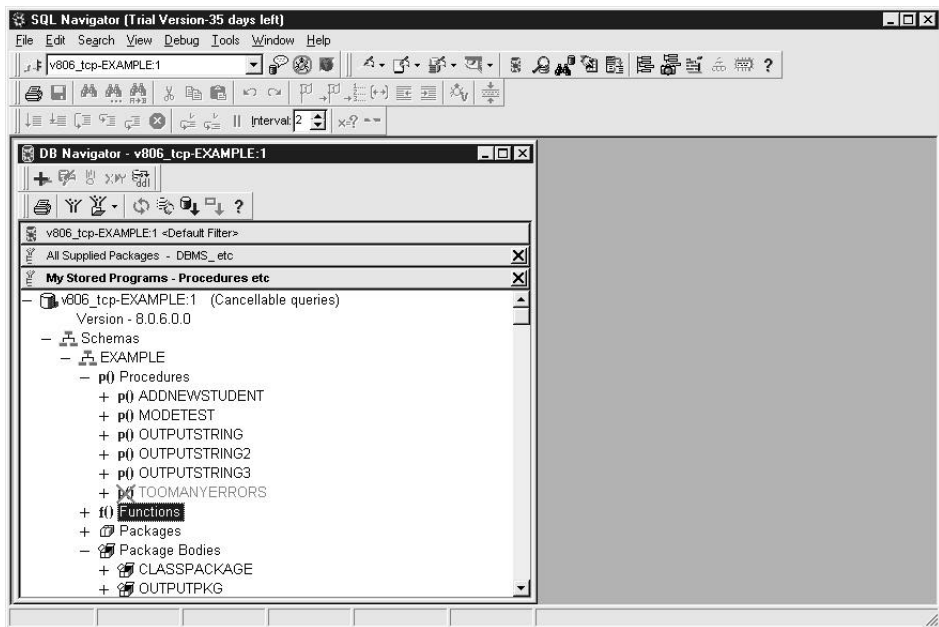


图2-10 带有三个过滤器的DB 浏览器窗口

如果用户建立了与 Oracle8i 数据库的连接，SQL浏览器还在其数据库浏览中支持 Oracle8i Java存储过程（JSPs）。这时，在浏览器的树形目录中将增加有关 Java类，Java 数据源，和资源的目录入口。虽然用户不能直接在 SQL浏览器中对Java代码进行编辑，但可以对已在数据库中的 Java 数据源进行编译。有关在 Oracle8i中使用JSPs的方法，请看本书第10章的内容。

4. 编辑数据库对象

SQL浏览器提供了三种编辑窗口：

- SQL 编辑器，该编辑器只能编辑一个 SQL语句或脚本。
- 触发器编辑器，该编辑器用来创建或编辑数据库触发器。除此之外，该编辑器还支持 Oracle8的Instead-of 触发器类型。
- 存储程序编辑器，该编辑器用来创建或编辑存储的 PL/SQL代码。

当用户单击DB浏览器中的对象时，相应的编辑器就会开始运行。用户也可以使用 SQL浏览器的工具条直接打开各种编辑器。图 2-11显示了在存储程序编辑器中的函数 AlmostFull。该窗口突出显示了该函数中的PL/SQL代码。

5. 显示错误

如果在编译PL/SQL对象的过程中出现错误的话，这些错误将在用户在存储程序编辑器中编译对象时显示出来。单击突出显示错误的源程序行，以及双击一个错误时都将启动一个来自于 Oracle文档资料中指示错误原因和方式的窗口。CD-ROM中的图 CD2-14显示了过程 TooManyErrors的错误信息。

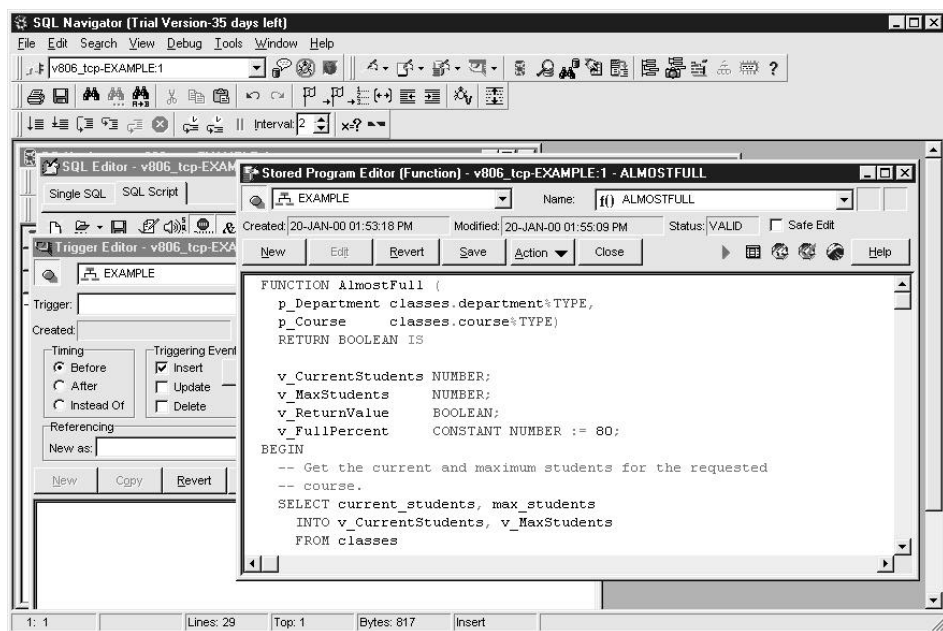


图2-11 编辑函数AlmostFull的窗口

6. 代码模板

可以在SQL浏览器工具菜单中启动的代码助理 (Code Assistant) 提供了PL/SQL和SQL结构中通用库。加亮显示某个结构将在代码助理信息窗口显示该结构的描述信息，双击该描述区将把该信息复制到现行的编辑窗口中并可由用户进行定制。代码助理的窗口图形显示在 CD-ROM 中图CD2-15中。

2.2.5 TOAD

TOAD是Oracle应用开发工具的缩写。该工具最早是从SQL浏览器分离出来的。现在，该工具是由Quest Software和SQL Navigator共同开发销售。这样做的结果导致这两个工具在某些方面（如许可机制）具有相同的功能，但是，我们马上也会看到，它们之间的不同。TOAD提供了下述功能：

- 自动格式PL/SQL和SQL语句
- 提供PL/SQL调试器
- 数据库浏览器
- 支持Oracle8对象类型
- 代码模板
- 支持第三方版本控制系统

TOAD是一种功能完善的轻量级开发工具。该工具所需的磁盘和内存空间都比其他工具要小的多。

1. 连接数据库

TOAD可支持多数据库连接。当该应用首次启动运行时，它使用一个对话框提示用户建立数

数据库连接（该对话框在CD-ROM中的图CD2-16中）。如果需要建立更多的连接的话，用户可以通过菜单命令 File | New Connection来实现。一旦建立了连接，该连接就将保持到用户通过执行菜单命令 File | Close Connection来将其关闭为止。连接使用的口令可以存储在连接配置文件中。

TOAD管理连接的一种特殊功能是与任何窗口关联的连接可以动态地进行变更。该功能使应用程序可以工作在多数据库环境下时将其他不用的窗口最小化。但对于一个给定的工作窗口来说，它只能与一个对话相关联。

2. 浏览数据库对象

TOAD提供了两种不同的数据库浏览器，它们是模式浏览器（Schema Browser）和对象浏览器（Object Browser）。图2-12所示的模式浏览器允许用户选择 Oracle7类型的表、过程和包。该浏览器提供的目录结构与我们在上面介绍的其他三种开发工具所提供的树形目录结构不同，模式浏览器中有选择对象类型的标签，这些标签显示在窗口中左面的窗格中。该窗口的右窗格显示对象的详细内容，用户可以从该浏览器中编译或删除对象。

对象浏览器只能用来查看 Oracle8的对象类型和类型体。类似于模式浏览器，对象浏览器允许用户查看和修改对象类型和类型体的属性。显示对象 Point的对象浏览器的图形窗口在CD-ROM中的图CD2-17中。

3. 编辑数据库对象

TOAD提供了两种编辑窗口：SQL编辑窗口和存储过程编辑窗口。正如这两个窗口的名称所表示的含意那样，SQL编辑窗口是用于编辑单个SQL语句或SQL脚本的，而存储过程编辑窗口则用来编辑存储过程、函数、包和触发器。在存储过程编辑窗口下，用户可以编译，运行，或调试过程。图2-13所示的是在存储过程编辑窗口中显示的过程 OutputString，该过程是一个用C语

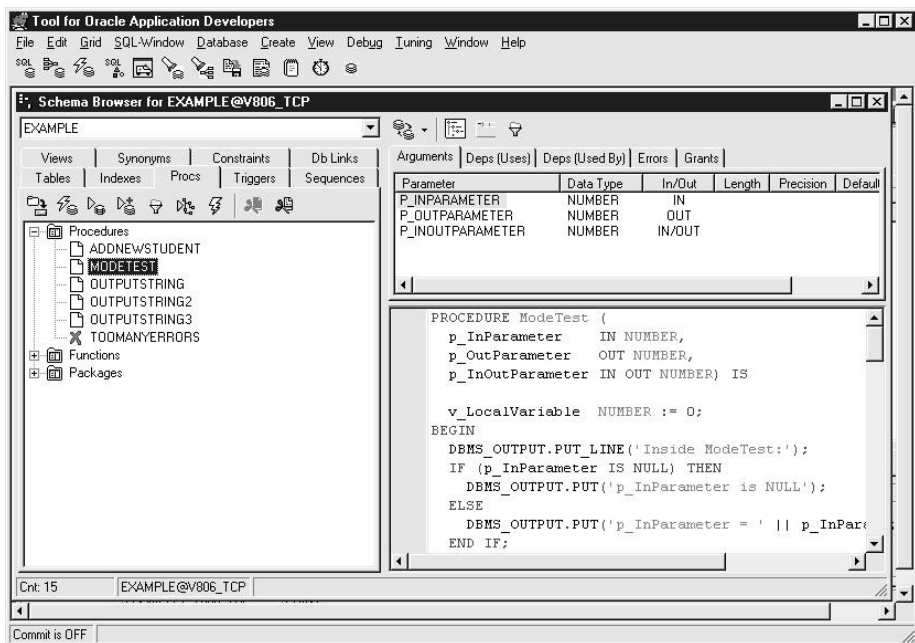


图2-12 模式浏览器窗口

言实现的外部过程。存储过程编辑窗口可以从数据库对象或文件中载入并用来创建新的对象。

4. 显示错误信息

如果在编译过程中出现了编译错误，这些错误将显示在存储过程编辑窗口中下面的错误显示窗格中。当使用蓝色三角形按钮单击错误时，与产生这些错误有关的源程序行将突出显示，本书CD-ROM中的图CD2-18演示了这种情况。除此之外，也可以使用模式浏览器来查看非法对象的错误。

5. 代码模板

TOAD支持通用PL/SQL和SQL结构使用的代码模板。在该代码模板下，可以不用输入所需的结构，用户只要使用键盘快捷键就可以实现。其具体操作是，先在编辑窗口中输入键盘快捷键，然后再按CTRL-SPACEBAR，这时该快捷键将被一个完整的结构替代。代码模板可以在编辑选择项窗口中查看，如CD-ROM中的图CD2-19所示。除此之外，用户还可以编辑现存的模板，或加入自己建立的模板。

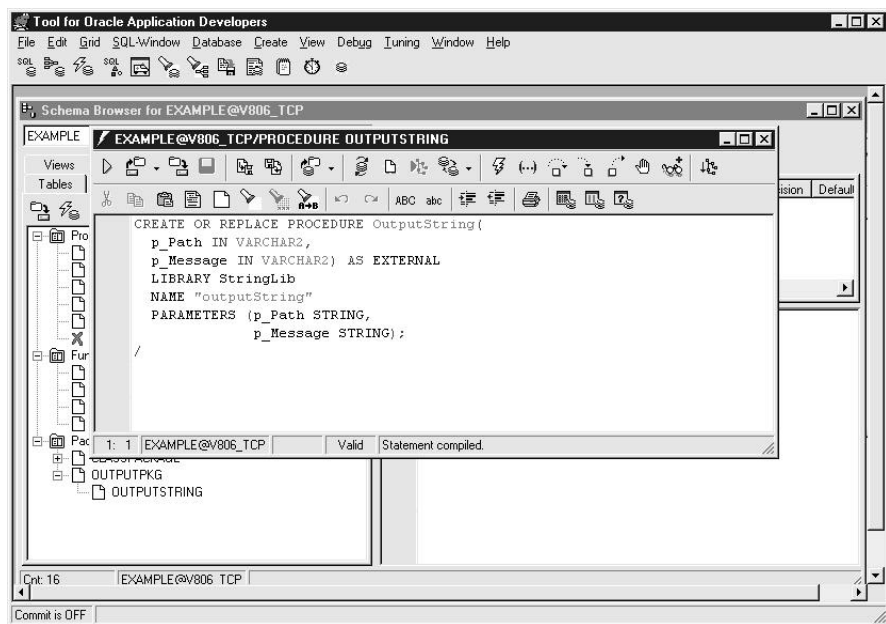


图2-13 存储过程编辑窗口

2.2.6 SQL-Programmer

最后一个介绍的图形用户界面开发工具是 SQL-Programmer，该工具是由 Sylvain Faust International开发的。它支持下列功能：

- 自动格式PL/SQL和SQL语句
- 提供PL/SQL调试器
- 数据库浏览器

- 支持Oracle8对象类型
- 代码模板
- 支持第三方版本控制系统
- 数据库对象脚本编制

1. 连接数据库

SQL-Programmer支持对不同数据库的多点同时连接。CD-ROM中的图CD2-20是连接对话框的窗口显示。虽然连接口令不能存储在连接配置文件中,但系统可以为不同的服务器存储不同的连接配置文件。连接对话框还可以显示已打开的连接。

2. 浏览数据库对象

SQL-Programmer的SQL浏览窗口提供了查看数据库对象的功能,该窗口中显示的对象按模式和对象类型进行排序。如果当前有一个以上的连接处于活动状态的话,则单个 SQL浏览窗口就可以浏览几个连接中的对象。图2-14中的窗口演示了正在显示函数 AlmostFull的SQL浏览窗口。从该浏览器中,用户可以观察对象的属性,编译对象,或者将对象删除。

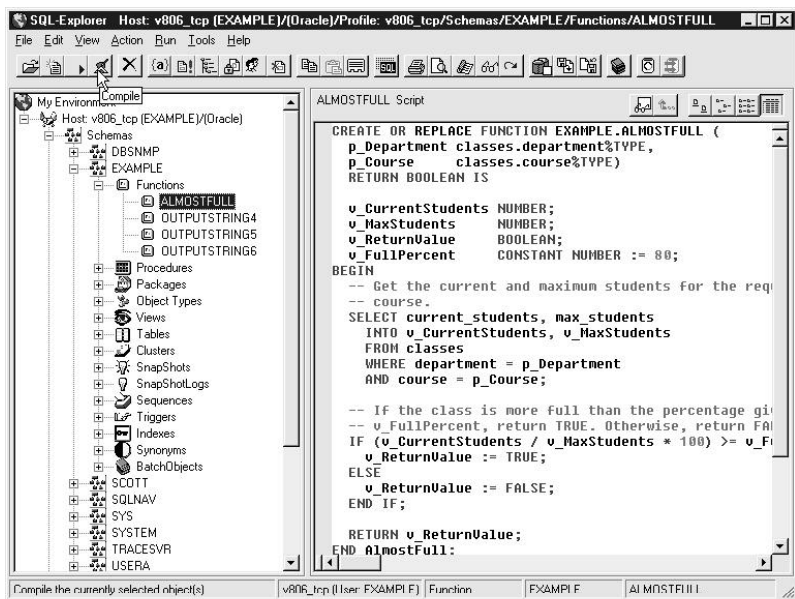


图2-14 SQL浏览窗口

3. 编辑数据库对象

双击SQL浏览窗口中显示的某个对象将启动 SQL-Programmer的开发窗口 (SPDW) 对其进行编辑。SPDW提供查询或修改所显示对象信息的功能。图2-15是显示函数 AlmostFull的SPDW窗口,如图所示,该窗口中还显示了变量。用户可以直接从SPDW中执行过程,这时,该窗口将显示对话框等待用户输入匿名块中使用的输入参数的值。

4. 显示错误信息

如果过程在编译出现错误的话,SPDW将在文本窗口下的错误窗格中显示这些错误。CD-ROM中图CD2-21就是演示SPDW该功能的图形窗口。双击该错误窗格中的某个错误将使与该错

误有关的代码段突出显示。同时，SPDW还将过程标识为非法。

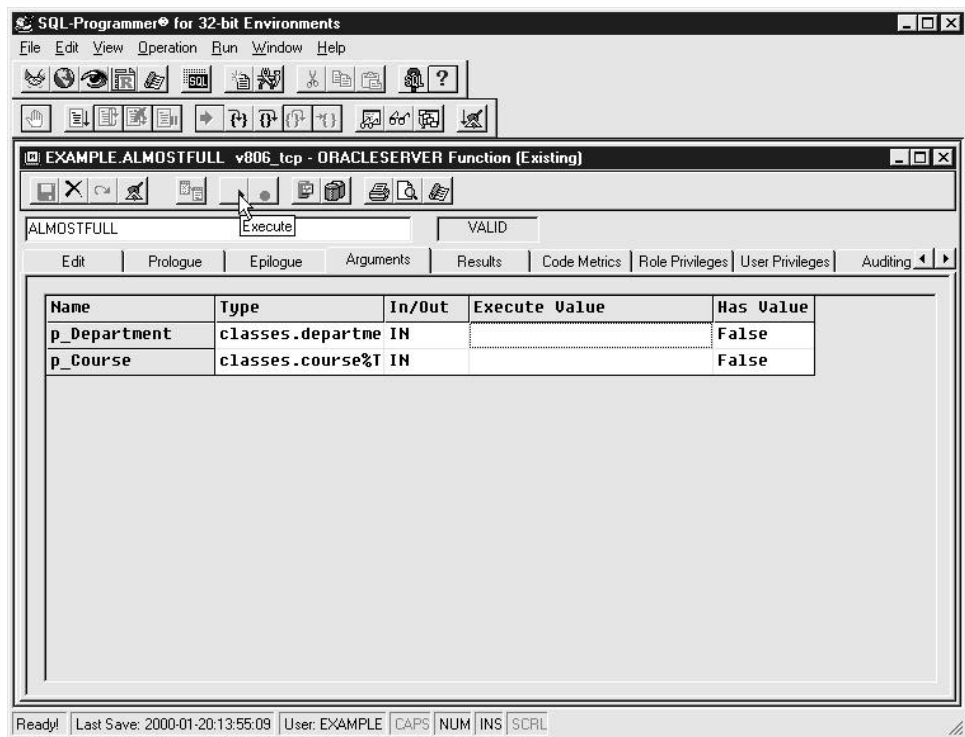


图2-15 显示AlmostFull的SPDW窗口

5. 代码模板

SQL-Programmer可以为创建新的对象提供模板。当用户在 SQL浏览窗口创建新的对象时，该浏览器将自动为创建的对象填充相应的模板。用户可以在窗口中的选择菜单中指定模板。CD-ROM中的图CD2-22显示了过程使用的默认模板。在该窗口中，除了过程代码外还有代码注解。

6. 脚本

SQL-Programmer的脚本界面允许用户为任何数据库对象的组合创建脚本。这种脚本中包括了可以自动地重建任何其他服务器所需的代码。因此，这种脚本提供了一种可以在数据库之间复制单独的对象，或全部模式的机制。CD-ROM中图CD2-23显示了这种脚本的窗口界面。

2.2.7 PL/SQL开发工具小结

到目前为止，我们所讨论的所有开发工具都没有提供客户端的 PL/SQL引擎。这些开发工具都可以用于开发和调试 PL/SQL应用。表2-3给出了这些工具之间主要性能的比较。有关这些开发工具的详细信息，请看本书 CD-ROM中提供的联机帮助和访问本章表 2-1中列出的开发商站点。

支持版本控制的开发工具通常借助于第三方数据源控制系统实现开发功能。不同的开发工具支持不同的数据源控制系统。有关进一步的信息，请参阅 CD-ROM中的联机帮助。

表2-3 PL/SQL开发工具性能比较

功能	SQL*PLUS	Rapid SQL	XPEDITER /SQL	SQL Navigator	TOAD	SQL- Programmer
提供方式	随Oracle提供	单独出售	单独出售	单独出售	单独出售	单独出售
GUI界面环境	无	有	有	有	有	有
对象浏览器	无	有	有	有	有	有
代码模板	无	有	有	有	有	有
工程管理	无	有	有	无	无	无
代码格式化	无	有	有	有	有	有
作业调度	无	有	无	无	无	无
版本控制	无	有	有	有	有	有
支持Oracle8 类型	有	有	有	有	有	有
支持Oracle8i 类型	有	有	无	有	无	无
安装服务器端要求	无	无	有	有	无	无
支持同时多个连接	无	有	无	有	有	有

2.3 小结

我们在本章介绍了六个用于创建 PL/SQL应用的开发工具。它们分别是 Oracle的SQL*Plus,Embarcadero Technologies公司的Rapid SQL, Compuware公司的XPEDITER/SQL,Qest Software公司的TOAD和SQL Navigator,以及Sylvain-Faust International的SQL-Programmer。这些工具中,除了SQL*Plus是随Oracle数据库一起提供的外,其他工具的试用版程序都在本书的CD-ROM中。我们在下一章将介绍这些开发工具提供的调试功能。