

第六部分 Oracle接口和应用工具

第26章 管理员使用的SQL*Plus

本章要点：

- 系统管理的SQL*Plus

- 使用SQL*Plus的COPY命令

- 使用SQL创建SQL

- 在SQL*Plus中对用户权限的限制

- 追踪SQL语句

- SQL*Plus 8.1版的增强

本章讨论SQL*Plus，它是用于访问 Oracle数据库的交互式应用程序，它也是数据库系统管理员最好的朋友和不可缺少的工具。

除了SQL*Plus外，数据库系统管理员还可以在另外两种工具中与数据库交互并使用SQL*Plus命令（但仅仅是SQL*Plus命令的一个子集）：

- 服务器管理器。

- 企业管理器中的工作表工具。

注意 从8.1.3版本开始，所有的服务器管理器命令在SQL*Plus中都是可用的。Oracle正计划放弃服务器管理器，并提供一个一体化的工具完成所有的数据库系统管理和 SQL命令（参见26.6节）。

因为SQL*Plus是个很大的主题，不是在一个章节中能完成的，我将讨论主要集中在 DBA感兴趣的特性方面，包括EXECUTE、AUTOTRACE和各种各样的新特性，还有少为人知且较少用到的特性（COPY命令、禁止命令）。

26.1 系统管理的SQL*Plus

有两个文件 glogin.sql和login.sql用于管理 SQL*Plus。glogin.sql文件是全局设置文件，login.sql文件是为个人使用准备的。这两个文件包含每次 Oracle用户激活SQL*Plus所执行的SQL*Plus命令和/或SQL语句。首先读和执行glogin.sql文件，然后是用户的login.sql文件。

glogin.sql文件位于\$ORACLE_HOME/sqlplus/admin目录下。Oracle数据库系统管理员可以自定义该文件含有SQL*Plus命令、SQL语句和PL/SQL块，它们在每个SQL*Plus用户开始他或她的SQL*Plus会话时被执行。glogin.sql文件也称作节点资源文件。

注意 在Windows 95/NT 4.0环境下，glogin.sql文件位于%ORACLE_HOME%\PLUSnn目录下，这里的nn代表机器上安装的SQL*Plus版本号。例如，如果安装的是SQL*Plus 3.3版，目录名将是%ORACLE_HOME%\PLUS33。

26.1.1 使用SQL*Plus环境变量

SQL*Plus里使用两个环境变量：

SQLPATH。

编辑器。

SQL*Plus使用环境变量SQLPATH指示login.sql文件的所在目录。换句话说，SQL*Plus将查看SQLPATH中定义的每个目录以找到login.sql，使用本地目录（当你启动SQL*Plus时所在的目录）启动SQL*Plus。

例如，如果你希望SQL*Plus首先在本地目录下查找login.sql文件，然后在你的根目录，最后再查找其他目录，那么如下设置SQLPATH：

```
$ SQLPATH=".:$HOME:<other_directory>"; export SQLPATH      — (Bourne/Korn shell)
$ set SQLPATH=(. $HOME <other_directory>)                  — C shell
```

在Windows 95/NT 4.0环境下，SQLPATH在注册表中定义，缺省值是 \$ORACLE_HOME\DBS（在安装期间设置的）。ORACLE_HOME的值在Windows 95中为C:\ORAWIN 95，在NT 4.0中为C:\ORANT（如果你没有将Oracle安装在C：驱动器上，使用你安装Oracle的磁盘驱动器符号代替C：）。

注意 在Windows 95/NT 4.0环境下，login.sql文件位于%ORACLE_HOME%\DBS目录下，这是SQLPATH的缺省值。

在Windows 95/NT 4.0环境下要设置或改变SQLPATH的值，遵照以下步骤：

- 1) 在Start菜单中选择Run。
- 2) 输入regedit.exe/regedit32.exe（分别对应于Windows 95/NT 4.0环境）。
- 3) 点击OK。
- 4) 双击HKEY_LOCAL_MACHINE。
- 5) 双击SOFTWARE。
- 6) 双击Oracle。
- 7) 双击SQLPATH。
- 8) Edit String 对话框出现，在Value Data域输入新的SQLPATH值。
- 9) 点击OK。
- 10) 在Registry菜单，选择Exit。
- 11) 重新启动你的机器，以便让新的值起作用（或者注销并再次登录进入Windows NT 4.0）。

注意 SQLPATH还被SQL*Plus用于指示你在SQL*Plus中运行的SQL脚本的位置。

在glogin.sql或login.sql中还可以设置另外一个环境变量，这个环境变量称为_editor，它定义了你用于编辑SQL*Plus命令的编辑器。

要设置编辑器为vi文本编辑器，在glogin.sql或login.sql文件中输入下面的代码行：

```
define _editor=vi
```

注意 在Oracle 8.1.x中，可以在工作过程中更改编辑器，进入Edit | Editor | Define editor并输入“Notepad”（笔记本）或“vi”。

如果你使用任何其他的文本编辑器，将“vi”替换为相应的名称。有关使用不同编辑器的

更多信息，在稍后的26.1.5节中讨论。

26.1.2 激活/访问SQL*Plus

要在操作系统提示符状态下激活 SQL*Plus，请使用下面的命令：

```
$ sqlplus [-S{ILENT}] [logon] [start]]!-?
```

当从一个壳脚本中运行 SQL*Plus时，使用-S{ILENT}参数，因为它消除了 SQL*Plus被激活时显示的全部信息，例如 SQL*Plus标志、提示信息和命令行提示符。

[logon]部分需要下面的语法结构：

```
username[/password] [@connect_string]!//NOLOG
```

[start]子句让你能够启动 SQL*Plus并运行一个包含 SQL*Plus命令、SQL语句和/或PL/SQL块的任意组合的命令文件。另外，你可以向这个命令文件传递参数。start子句要求下面的语法：

```
@file_name[.ext] [arg...]
```

如果你没有输入用户名和/或口令，SQL*Plus提示你输入它们。

在你成功地启动 SQL*Plus后，可以在 SQL*Plus提示符（SQL>）状态下输入三种类型的命令：

用于处理数据库对象和操作数据库中存储的数据的 SQL命令/语句。

用于处理数据库对象和操作数据库中存储的数据的 PL/SQL（过程化语言/SQL）块。

用于设置选项、编辑、存储和提取 SQL命令与PL/SQL块以及格式化查询输出结果的 SQL*Plus命令。

要向SQL*Plus提交一条SQL命令，请在命令的结尾处输入一个分号（；）并且按下回车键。SQL*Plus执行该命令，显示查询的结果并且返回到提示符状态。

要结束一个PL/SQL块，在它自己的行上输入一个小数点即可。要提交一个 PL/SQL块用于执行，在PL/SQL的行上用一条斜线（/）结束它并按下回车键。

26.1.3 编辑SQL命令

如果你在输入一条 SQL命令时出现一个错误并且希望改正它，或者如果你希望运行仅仅稍加改动的最后一条命令，你非常幸运！SQL*Plus将最近输入的SQL命令存储在一个缓冲区中，该缓冲区被称为 SQL缓冲区。SQL*Plus提供了一系列命令来提取并编辑存储在该缓冲区中的SQL语句和PL/SQL块。

注意SQL*Plus命令没有存储在该缓冲区中，所以，它们不能够被提取和修改。但是，有一种方法可以在该缓冲区中存储 SQL*Plus命令，这种方法在稍后的26.1.4节中讨论。

表26-1列出用于查看、编辑和运行 SQL*Plus缓冲区内容的命令。

除LIST命令外，所有其他的编辑命令仅影响缓冲区中的一行，该行被称为当前行，当列出SQL命令或PL/SQL块时，当前行用星号标记出来。当运行 LIST命令时，当前行总是缓冲区中的最后一行。还要注意终结一条 SQL命令的分号（；）没有存储在缓冲区中，如清单 26-1所示。

如果你得到一条错误信息，含有错误的行成为当前行，这样你就可以立即编辑该行并纠正错误。清单26-2显示了带有错误的程序清单。

表26-1 用于SQL缓冲区的SQL*Plus命令

命 令	缩 写	作 用
APPEND 文本	A 文本	在一行尾部添加文本
CHANGE 旧文本/新文本	C 旧文本/新文本	使用新文本更改一行中的旧文本
CHANGE /文本	C /文本	从一行中删除文本
CLEAR BUFFER	CL BUFF	删除全部行
DEL	(无)	删除缓冲区中的当前行
INPUT	I	在缓冲区中添加一行或多行
INPUT 文本	I 文本	添加一条文本组成的行
LIST	L	列出SQL*Plus缓冲区中的内容
LIST n	L n 或n	列出第n行
LIST *	L *	列出当前行
LIST m n	L m n	列出第m行到第n行
LIST LAST	L LAST	列出缓冲区中的最后一行

清单26-1 存储在缓冲区中的SQL语句

```
SQL> LIST
1 SELECT empno, ename, deptno, job, sal, comm
2 FROM emp
3* WHERE comm IS NOT NULL
```

清单26-2 带有错误的行成为当前行

```
SQL> SELECT empno, empname, deptno, job, sal, comm
2 FROM emp
3 WHERE comm IS NOT NULL;
SELECT empno, empname, deptno, job, sal, comm
*
ERROR at line 1:
ORA-00904: invalid column name
```

26.1.4 输入并编辑SQL*Plus命令

直接在SQL*Plus提示符下输入的SQL*Plus命令（例如DESCRIBE）不保存在SQL缓冲区中。所以，你不能够提取最后输入的SQL*Plus命令来编辑并/或再次运行它。试图从缓冲区中列出一条SQL*Plus命令的结果如清单26-3所示。

清单26-3 SQL*Plus命令没有被保存在缓冲区中

```
SQL> DESCRIBE EMP
Name Null? Type
-----
EMPNO NOT NULL NUMBER(4)
ENAME VARCHAR2(10)
JOB VARCHAR2(9)
MGR NUMBER(4)
HIREDATE DATE
SAL NUMBER(7,2)
COMM NUMBER(7,2)
DEPTNO NOT NULL NUMBER(2)
SQL> LIST
No lines in SQL buffer.
```

要在缓冲区中存储SQL*Plus命令，请输入不带有文本的INPUT命令并且按下回车键。

SQL*Plus使用一个行号提示你可以输入命令。当你完成命令输入时，按下回车键输入一行空行。清单 26-4显示了一个示例。

清单 26-4 在缓冲区中存储SQL*Plus命令

```
SQL> INPUT
  1 DESCRIBE EMP
  2
SQL> L
  1* DESCRIBE EMP
```

注意 参阅SQL*Plus命令表，缩写“L”代表“List”命令。

不可以从缓冲区中执行该命令，但是可以将它保存在一个文件中，可以以后在 SQL*Plus 中检索并执行这个文件。清单 26-5显示了从缓冲区中执行一个 SQL*Plus命令的尝试失败，后来成功地将它保存到文件中。

清单 26-5 保存SQL*Plus命令到文件

```
SQL> RUN
  1* DESCRIBE EMP
DESCRIBE EMP
*
ERROR at line 1:
ORA-00900: invalid SQL statement
SQL> SAVE test
Created file test
SQL> GET test
  1* DESCRIBE EMP
```

新创建的文件被保存在由SQLPATH环境变量指定的目录中。

26.1.5 在SQL*Plus中使用你的操作系统编辑器

SQL*Plus提供的编辑能力非常有限，而且相对于其他的文本编辑器也不直观（例如，不能使用箭头键、HOME键或END键）。所以，许多用户喜欢使用他们自己感到满意的编辑器来创建命令文件，然后在SQL*Plus中使用START命令或@命令运行这些文件。

如果你宁愿使用你的操作系统编辑器而不愿使用 SQL*Plus的编辑功能，你可以在SQL*Plus中使用EDIT命令实现这种操作。EDIT命令的语法为：

```
EDIT [file_name [.ext]]
```

上述命令将会使用glogin.sql或login.sql文件中变量_editor指定的编辑器打开名为file_name的文件。如果你想在SQL*Plus会话里使用另一种编辑器，你可以使用SQL*Plus命令DEFINE重新定义_editor变量：

```
DEFINE _editor=emacs
```

如果_editor变量没有定义，EDIT命令尝试使用操作系统缺省的编辑器（例如，在Windows 95环境下的记事本（Notepad））。

当你发出EDIT命令时，从SQL*Plus内部激活一个编辑器而不需要退出SQL*Plus，这样是非常方便的。

如果你运行EDIT命令时没有提供一个文件名，SQL*Plus将在一个文件中保存SQL缓冲区中的内容并使用编辑器打开这个文件。在缺省情况下，该文件的名称为afiedt.buf，它创建在

当前目录下或由SQLPATH环境变量定义的目录下。当打开编辑器的时候，你可以使用你希望编辑的文件的全路径名，例如：EDIT C:\MYDIR\MYFILE.SQL。

```
SQL> EDIT
Wrote file afiedt.buf
```

通过在editfile变量中设置合适的值，可以更改SQL*Plus用于保存缓冲区内容的文件的名称，如清单26-6所示。

清单26-6 被编辑的文件有一个缺省名称

```
SQL> SHOW editfile
editfile "afiedt.buf"
SQL> SET editfile "buffer.txt"
SQL> SHOW editfile
editfile "buffer.txt"
```

当运行EDIT命令时，如果你没有输入一个文件名并且SQL缓冲区为空，SQL*Plus返回一个通知消息，如清单26-7所示。

清单26-7 没有保存任何东西的空缓冲区

```
SQL> CLEAR BUFFER
buffer cleared
SQL> EDIT
Nothing to save.
```

缺省的文件名称的扩展名是.sql。所以，如果你没有指定一个文件扩展名，SQL*Plus将查找名为file_name.sql的文件。如果你想编辑一个扩展名不是.sql的文件，你必须明确地指定文件的扩展名。你还可以通过SUFFIX变量更改文件扩展名的缺省值，如清单26-8所示。

清单26-8 更改缺省的扩展名

```
SQL> SHOW SUFFIX
suffix "SQL"
SQL> SET SUFFIX sh
SQL> SHOW SUFFIX
suffix "sh"
```

注意 SUFFIX变量只适用于命令文件，不适用于输出文件。依赖于操作系统，输出文件的缺省扩展名是.lst或.lis。

26.1.6 运行SQL*Plus/SQL命令

你可以采用三种方式运行SQL命令和PL/SQL块：

命令行方式。

SQL缓冲区方式。

命令文件方式（非正式地称为SQL脚本）。

为了以缓冲区方式执行SQL命令或PL/SQL块，SQL*Plus提供了RUN命令和/（前斜线）命令。RUN命令的语法是：

```
R[UN]
```

RUN命令列出并执行当前存储在缓冲区中的SQL命令或PL/SQL块。

我们假设缓冲区中包含以下查询：

```
SELECT empno, ename FROM emp
```

如果你使用RUN命令运行这个查询，它看起来如清单 26-9所示。

清单26-9 使用RUN命令运行一个查询

```
SQL> RUN
      1* SELECT empno, ename FROM emp
EMPNO      ENAME
-----
      7369 SMITH
      7499 ALLEN
      7521 WARD
      7566 JONES
      7654 MARTIN
      7698 BLAKE
      7782 CLARK
      7788 SCOTT
      7839 KING
      7844 TURNER
      7876 ADAMS
      7900 JAMES
      7902 FORD
      7934 MILLER
14 rows selected.
```

RUN命令显示缓冲区中的命令并返回查询的结果。另外，RUN命令使得缓冲区中的最后一行成为当前行。

/命令类似于RUN命令，它执行存储在缓冲区中的SQL命令或PL/SQL块，但它不显示缓冲区的内容，如清单 26-10所示。

清单26-10 使用/命令运行一个查询

```
SQL> /
EMPNO      ENAME
-----
      7369 SMITH
      7499 ALLEN
      7521 WARD
      7566 JONES
      7654 MARTIN
      7698 BLAKE
      7782 CLARK
      7788 SCOTT
      7839 KING
      7844 TURNER
      7876 ADAMS
      7900 JAMES
      7902 FORD
      7934 MILLER
14 rows selected.
```

注意 不同于RUN命令，/命令不会使缓冲区中的最后一行成为当前行。

要以命令行形式运行一个SQL命令、一个SQL*Plus命令或一个PL/SQL块，有两种命令：

START

@ (“ at ”)

START命令的语法是：

```
START file_name[.ext] [arg1 arg2...]
```

参数file_name[.ext]代表你想运行的命令文件，如果省略扩展名，SQL*Plus使用缺省的命令文件扩展名（通常为.sql）。

SQL*Plus在当前目录下查找具有你在START命令中指定的文件名和扩展名的文件。如果没有找到符合条件的文件，SQL*Plus将在SQLPATH环境变量定义的目录中查找该文件。也可以包括文件的全路径名，例如：C:\MYDYR\MYFILE.SQL。

可以在一个命令文件中包括正常情况下交互地输入的任何SQL命令、SQL*Plus命令或PL/SQL块。命令文件中的EXIT或QUIT命令用于退出SQL*Plus。

参数部分（[arg1 arg2...]）代表你希望传递给命令文件中的参数的值，命令文件中的参数必须使用如下格式声明：&1、&2、...（或&&1、&&2、...）。如果你输入一个或多个参数，SQL*Plus使用这些值替换命令文件中的参数。第一个参数替代每个&1，第二个参数替代每个&2，以此类推。

START命令使用参数的值定义参数。如果你在同一个SQL*Plus会话中再次运行这个命令文件，你可以输入新的参数或省略参数以使用参数的当前值。要运行名称为DELTBL.SQL的命令文件，输入以下代码：

```
SQL> START DELTBL
```

@（“at”）命令的功能与START命令非常类似，唯一的区别就是@命令既可以在SQL*Plus会话内部运行，又可以在启动SQL*Plus时的命令行级别运行，而START命令只能在SQL*Plus会话内部运行。要启动一个SQL*Plus会话并执行一个命令文件中的命令，输入下列代码：

```
$ sqlplus [username/password] @file_name[.ext] [arg1 arg2...]
```

如果START命令被禁止，同样会禁止@命令。本章后面的26.4节含有关于禁止SQL*Plus命令的详细信息。

1. 使用EXECUTE命令

从SQL*Plus 3.2开始，有一个新的命令——EXECUTE，该命令能够直接在SQL*Plus提示符状态下执行单条PL/SQL语句，而不需要从缓冲区或命令文件中执行。EXECUTE的主要用途是运行涉及一个函数或存储过程的PL/SQL语句，如清单26-11所示。

清单26-11 使用EXECUTE命令运行存储过程

```
SQL> VARIABLE id NUMBER           -- Define a bind variable
SQL> EXECUTE :id := ADD_CASES(10);
PL/SQL procedure successfully completed.
SQL> PRINT id
      ID
-----
      10
SQL> EXECUTE :id := ADD_CASES(3);
PL/SQL procedure successfully completed.
SQL> PRINT id
      ID
-----
       0
```

存储过程ADD_CASES返回的值存储在绑定变量:id中。

2. 保存SQL*Plus/SQL命令

使用SAVE命令，你可以将缓冲区中存储的SQL*Plus或SQL命令保存到一个操作系统文件（称为命令文件）中。SAVE命令的语法为：

```
SAV[E] file_name[.ext] [CRE[ATE] | [REP[LACE] | APP[END]]
```

file_name[.ext]是用于存储SQL缓冲区内容的操作系统文件的名称。要命名该文件，应使用SQL*Plus运行的操作系统的文件命名约定。如果你没有提供文件的扩展名，SQL*Plus使用缺省的扩展名.sql。你还可以指定一条路径作为文件名称的一部分。如果没有指明路径，SAVE命令将使用SQLPATH环境变量中指定的目录作为文件的路径。

CRE[ATE]参数创建命令文件。如果这个文件已经存在，你将收到一条错误消息。

REP[LACE]参数使用SQL缓冲区中的内容替换一个已有文件。如果文件不存在，SAVE...REPLACE命令创建它。

APP[END]参数将SQL缓冲区中的内容添加到文件的尾部。如果文件不存在，SAVE...APPEND命令创建该文件。

3. 提取SQL*Plus/SQL命令

要提取SQL*Plus或SQL命令，使用GET命令。该命令的语法是：

```
GET file_name[.ext] [LIS[T] | NOL[IST]]
```

GET命令加载包含SQL*Plus和/或SQL命令的操作系统文件file_name到SQL缓冲区，那么你就可以编辑文件中的命令或运行它们。文件的缺省扩展名为.sql。

LIS[T]参数在SQL*Plus加载文件到缓冲区时列出文件的内容。LIST参数是缺省设置。NOL[IST]参数不列出文件的内容。

如果在GET命令中没有指定文件名的全路径，SQL*Plus首先在当前目录下查找该文件，然后在环境变量SQLPATH中列出的目录中查找文件。

提示 如果使用Windows 95/NT，通过在指定目录下启动SQL*Plus应用，可以让SQL*Plus在该目录下查找命令文件。要实现它，更改Windows快捷方式的Start In属性为包含这些文件的目录。没有指定全路径的文件将在SQL*Plus启动时所在的目录中被打开、创建或保存。

26.2 使用SQL*Plus的COPY命令

虽然由于Oracle在7.2版中推出的CREATE TABLE...UNRECOVERABLE AS SELECT...命令在某种程度上削弱了COPY命令所起的作用，但COPY命令仍然是一条最为有用的SQL*Plus命令；但是人们对它不甚了解，所以不经常使用它。COPY命令可以用来实现几个功能：

从一个本地数据库将一个或多个表或整个模式拷贝到一个远程数据库或另外一个本地数据库。这可以用于从一个数据库将整个模式移动到另外一个数据库，而不需要使用导出/导入工具，当导出大于操作系统文件限制的文件时这尤其有用。

将一个表中指定的记录（基于查询）拷贝到远程数据库或本地数据库的其他表中。

将包含LONG类型数据列的表的内容拷贝到其他表。因为LONG类型的列不能用于SELECT语句中，所以这是解决此问题的唯一方法。

从一个Oracle数据库向一个非Oracle数据库拷贝表。

COPY命令的语法为：

```
COPY {FROM username[/password]@database_specification| TO
username[/password]@database_specification| FROM
username[/password]@database_specification TO
username[/password]@database_specification} {APPEND|CREATE|INSERT|REPLACE}
destination_table [(column, column, column ...)] USING query
```

username[/password]代表你希望拷出 /拷入的 Oracle 用户名和口令。在 FROM 子句中, username/password 标识数据的源; 在 TO 子句中 username/password 标识数据的目的地。如果你既没有在 FROM 子句中指定口令也没有在 TO 子句中指定口令, SQL*Plus 会提示你输入口令。SQL*Plus 不显示对这些提示的用户响应。

database-specification (数据库声明) 是数据库链接名、Net 8 服务名或 SQL*Net V2 服务名。在 COPY 命令中, 必须提供数据库声明子句。在 FROM 子句中, 数据库声明代表源端数据库; 在 TO 子句中, 数据库声明代表目的端数据库。

destination_table (目标表) 是你希望创建的表或向其中添加数据的表, [(column, column, column...)] 指示了目标表中的列名。

如果指定列, 列的数目必须与查询选择的列的数目相同。如果没有指定任何列, 如果 COPY 命令创建目标表, 被拷贝的列在目标表中将具有与源表中的列相同的名字。

USING query 参数指定一个提取 COPY 命令拷贝的行和列的 SELECT 语句。

FROM username[/password]@database_specification 部分定义用户名、口令和含有需要拷贝的数据的数据库。如果省略 FROM 子句, 源数据库在缺省情况下为 SQL*Plus 连接的数据库。你必须包含一个 FROM 子句以指定缺省数据库以外的源数据库。

TO username[/password]@database_specification 部分指明包含目标表的数据库。如果省略 TO 子句, 目标表在缺省情况下为 SQL*Plus 连接的数据库。必须包含 TO 子句以指定一个缺省数据库以外的目标数据库。

有几个参数用于控制 COPY 命令如何从一个表向另一个表中拷贝数据。这些参数如下所示:

如果目标表存在, APPEND 将查询出的记录插入到目标表。如果目标表不存在, COPY 命令创建该表。即使该表为空 (不含数据), APPEND 命令也插入记录。

CREATE 首先创建目标表, 然后向目标表中插入取自查询的记录。如果目标表已经存在, COPY 命令返回一个错误。

INSERT 将取自查询的记录插入目标表。如果目标表不存在, COPY 命令返回一个错误。当使用 INSERT 命令时, USING 查询必须为目标表中的每列选择一行。

REPLACE 使用查询提取的记录替换目标表及其内容。如果目标表存在, COPY 删除它并用含有拷贝数据的表替换它。如果目标表不存在, COPY 命令创建它。

有三个 SQL*Plus SET 变量控制 COPY 命令的行为:

LONG

COPYCOMMIT

ARRAYSIZE

LONG 变量决定你所拷贝的 LONG 类型列的长度。如果 LONG 类型列包含的数据长度大于 LONG 变量的值, COPY 命令将截断数据。

在缺省情况下, SQL*Plus 在每个成功执行的 COPY 命令后面执行一次提交。如果设置 SQL*Plus 的 SET 变量 COPYCOMMIT 为一个正整型值 n, SQL*Plus 将在每拷贝 n 批记录后执行

一次提交。SET变量ARRAYSIZE决定批操作的大小。

注意 当使用COPY命令时，一些操作系统环境（特别是VAX/VMX、OpenVMX）要求以双引号括起数据库声明。

注意 如果在COPY命令的（column column, ...）子句中定义的任一列名含有小写字母或空格，列名必须使用双引号括起来。

注意 当使用COPY命令拷贝含有数字型（NUMBER）列的表时要非常小心！这些列在目标数据库中变为精度为38位的小数类型（DECIMAL）。

26.3 使用SQL创建SQL

在许多情况下数据库系统管理员或一个普通的用户需要对许多数据库对象（表、索引等等）执行SQL语句。例如，数据库系统管理员可能会需要周期性地从一个成品数据库加载数据到开发数据库或测试数据库。在加载数据之前，他或她可能需要删除开发数据库或测试数据库中所有表中的数据。基于单个表来做这件工作会非常繁琐，尤其当数据库中有几百个表时。在这种情形下，一种简单快捷的解决方案是使用SQL*Plus创建所需的SQL语句，将它们保存到一个操作系统文件中，并在SQL*Plus中运行该文件。

例如，要删除属于用户SCOTT的所有表中的所有记录，SYSTEM用户可以使用清单26-12中所示的SQL语句。

清单26-12 SQL生成SQL

```
SQL> CONNECT SYSTEM
Enter password: *****
Connected.
SQL> SET PAGESIZE 0
SQL> SET HEADING OFF
SQL> SET FEEDBACK OFF
SQL> SET VERIFY OFF
SQL> SET ECHO OFF
SQL> SPOOL DELTBL.SQL
SQL> SELECT 'DELETE '||TABLE_NAME||';'
2 FROM DBA_TABLES
3 WHERE OWNER='SCOTT';
DELETE ACCTS;
DELETE ACCT_ADDRS;
DELETE BONUS;
DELETE CITIES;
DELETE COMPANY_SUMMARY;
DELETE CUSTOMER;
DELETE DEPT;
DELETE DUMMY;
DELETE EMP;
DELETE FUNDS;
DELETE FUND_CONTRIB;
DELETE FUND_XACT;
DELETE F_EMPCOMP;
DELETE F_XACT_TYPE;
DELETE INVINFO;
DELETE INVREQUEST;
DELETE ITEM;
DELETE ORD;
DELETE ORDER_HISTORY;
```

```
DELETE PRICE;  
DELETE PRODUCT;  
DELETE SALES_REVENUE;  
DELETE SALGRADE;  
DELETE STOCK_HISTORY;  
SQL> SPOOL OFF
```

当你运行上述文件时，将得到文件 DELTBL.SQL，如下所示：

```
DELETE ACCTS;  
DELETE ACCT_ADDRS;  
DELETE BONUS;  
DELETE CITIES;  
DELETE COMPANY_SUMMARY;  
DELETE CUSTOMER;  
DELETE DEPT;  
DELETE DUMMY;  
DELETE EMP;  
DELETE FUNDS;  
DELETE FUND_CONTRIB;  
DELETE FUND_XACT;  
DELETE F_EMPCOMP;  
DELETE F_XACT_TYPE;  
DELETE INVINFO;  
DELETE INVREQUEST;  
DELETE ITEM;  
DELETE ORD;  
DELETE ORDER_HISTORY;  
DELETE PRICE;  
DELETE PRODUCT;  
DELETE SALES_REVENUE;  
DELETE SALGRADE;  
DELETE STOCK_HISTORY;
```

发出命令 @DELTBL.SQL 运行 DELTBL.SQL 文件将删除所示表中的记录。其他的命令用于格式化输出和创建命令文件：

SET PAGESIZE 0 关闭全部页格式化信息，例如列标题、标题、起始空白行、分页符等等。

SET HEADING OFF 关闭列标题显示（你可以使用 SET HEADING OFF 或 SET PAGESIZE 0 实现）。

SET FEEDBACK OFF 隐藏查询返回的记录号。

SET VERIFY OFF 不在 SQL*Plus 使用实际值替换变量的前后显示 SQL 命令的文本。

SET ECHO OFF 可以在文件执行时，隐藏 DELETE.SQL 文件中的 SQL 命令清单。

SPOOL DELTBL.SQL 开始将在 SQL*Plus 提示符下输入的命令查询结果保存入名为 DELTBL.SQL 的文件中。命令和查询结果继续在终端上显示。如果没有指明一个文件扩展名，SQL*Plus 为输出文件使用缺省的扩展名（.lst 或 .lis）。

SPOOL OFF 停止保存并关闭 DELTBL.SQL 文件。

如果 SYSTEM 想删除 SCOTT 用户的全部表中的记录，他所要做的就是 在 SQL*Plus 中执行 DELTBL.SQL 文件，可以按照以下方式实现：

```
SQL> @DELTBL
```

26.4 在 SQL*Plus 中对用户权限的限制

Oracle 为数据管理员提供了一个工具，让他们能够禁止在 SQL*Plus 环境下指定的 SQL 和

SQL*Plus命令的执行，它基于单个用户实现。事实上，这个工具是一个表——PRODUCT_USER_PROFILE，由用户SYSTEM拥有。在表26-2中给出PRODUCT_USER_PROFILE的表说明；表26-3说明PRODUCT_USER_PROFILE表中每列的用途。

表26-2 PRODUCT_USER_PROFILE表定义

名 称	能否为空	类 型
PRODUCT	不能为空	VARCHAR2(30)
USERID		VARCHAR2(30)
ATTRIBUTE		VARCHAR2(240)
SCOPE		VARCHAR2(240)
NUMERIC_VALUE		NUMBER(15, 2)
CHAR_VALUE		VARCHAR2(240)
DATE_VALUE		DATE
LONG_VALUE		LONG

表26-3 PRODUCT_USER_PROFILE表的列的用途

列	用 途
PRODUCT	必须含有产品的名称（在本例中为SQL*Plus）。在该列中不允许使用通配符（%）或空（NULL）值
USERID	必须含有你希望禁止命令操作的用户的名称（大写）。如果针对几个用户禁止某个命令，使用通配符（%）或输入多条记录。如果针对全部用户禁止某个命令，只要在该列中插入一个通配符（%）即可
ATTRIBUTE	必须包含被禁用的SQL或SQL*Plus命令（例如ALTER）的名称（大写）。该列不允许使用通配符值
SCOPE	SQL*Plus忽略该列。应该在该列输入一个空（NULL）值。该列被保留用于非SQL*Plus产品
NUMERIC_VALUE	SQL*Plus忽略该列。应该在该列输入一个空（NULL）值。该列被保留用于非SQL*Plus产品
CHAR_VALUE	必须含有字符串"DISABLED"
DATE_VALUE	SQL*Plus忽略该列。应该在该列输入一个空（NULL）值。该列被保留用于非SQL*Plus产品
LONG_VALUE	SQL*Plus忽略该列。应该在该列输入一个空（NULL）值。该列被保留用于非SQL*Plus产品

作为SYSTEM用户运行脚本PUPBLD.SQL创建PRODUCT_USER_PROFILE表。这个文件的位置根据操作平台的不同而不同。例如，对于Windows 95下的个人Oracle系统，PUPBLD.SQL文件位于目录C:\ORAWIN 95\DBS中（C:\ORAWIN95是Oracle产品在Windows 95操作系统中的缺省ORACLE_HOME目录）。在UNIX平台，这个文件位于\$ORACLE_HOME/sqlplus/admin目录下。

在安装Oracle软件与创建起始数据库期间（对于UNIX平台在crdb脚本中，对于Windows 95/NT平台在buildall脚本中），PUPBLD.SQL命令文件被运行。如果在安装过程中没有创建起始数据库，必须作为SYSTEM用户运行PUPBLD.SQL脚本以创建PRODUCT_USER_PROFILE表。

如果PUPBLD.SQL没有运行，每当用户登录SQL*Plus时，他或她将收到下面的警告消息：

```
Warning: Product user profile information not loaded !
Error in disabling roles in product user profile.
```

（警告：产品用户环境资源文件信息没有加载！在产品用户环境资源文件中禁止角色产生错误。）

26.4.1 禁用一个SQL命令

要禁止某给定用户使用某个 SQL或SQL*Plus命令，SYSTEM用户必须插入与表 26-4所示列和值匹配的一条记录。

表26-4 用于限制SQL*Plus命令的值

列	值
PRODUCT	SQL*Plus
USERID	被限制的用户的用户名
ATTRIBUTE	被限制的命令名称
CHAR_VALUE	DISABLED
SCOPE	NULL (空)
NUMERIC_VALUE	NULL (空)
DATE_VALUE	NULL (空)
LONG_VALUE	NULL (空)

例如，要禁止用户 SCOTT从SQL*Plus访问操作系统，需要作为 SYSTEM用户登录到SQL*Plus并执行清单 26-13中的SQL语句。

清单26-13 禁止SCOTT用户使用HOST命令

```
INSERT INTO
product_user_profile
(product, userid, attribute, scope, numeric_value,
char_value, date_value, long_value)
VALUES
('SQL*Plus', 'SCOTT', 'HOST', NULL, NULL, 'DISABLED', NULL, NULL);
```

当用户访问SQL*Plus时，SQL*Plus从PRODUCT_USER_PROFILE表读取用户限制，并在用户对话期间强制实施这些限制。

注意 如果在PRODUCT_USER_PROFILE表中做出影响用户的任何改动，这些改变将不会影响用户的当前会话。用户可以在下一次登录到SQL*Plus时看到这些改变的效果。

26.4.2 重新允许使用一个SQL命令

要重新允许使用一个 SQL命令，需要删除含有限制的记录。清单 26-14重新允许使用在清单26-13中被禁止的命令。

清单26-14 允许用户SCOTT使用HOST命令

```
DELETE [FROM] product_user_profile
WHERE userid='SCOTT' and attribute='HOST';
```

可以使用PRODUCT_USER_PROFILE表禁止下面的SQL*Plus命令（以字母顺序排列）：

```
COPY
EDIT
EXECUTE
EXIT
GET
```

HOST (或者操作系统HOST的别名, 例如UNIX上的 ' ! ' 或VMS上的 ' \$ ')

QUIT

PASSWORD

RUN

SAVE

SET (参见下面的注释)

SPOOL

START (参见下面的注释)

注意 禁止SQL*Plus的SET命令还会禁止SET ROLE和SET TRANSACTION命令。

注意 禁止SQL*Plus的START命令还会禁止SQL*Plus的@和@@命令。

下面的SQL命令可以被禁止：

ALTER

ANALYZE

AUDIT

CONNECT

CREATE

DELETE

DROP

GRANT

INSERT

LOCK

NOAUDIT

RENAME

REVOKE

SELECT

SET ROLE

SET TRANSACTION

TRUNCATE

UPDATE

你还可以禁止下面的PL/SQL命令：

BEGIN

DECLARE

注意 禁止BEGIN和DECLARE命令不妨碍SQL*Plus的EXECUTE命令的使用。

EXECUTE命令必须被单独禁止。

26.4.3 禁用SET ROLE

SQL命令SET ROLE使应用的用户能够访问获取应用的角色授予的特权，这样做可能会产生安全问题。要阻止应用的用户在SQL*Plus中获取应用的角色，数据库系统管理员可以使用

PRODUCT_USER_PROFILE表禁用SET ROLE命令。当他或她启动SQL*Plus时，该命令限制SQL*Plus用户只能使用角色允许的相关的权限。

26.4.4 禁用角色

要禁止一个指定用户的角色，应在PRODUCT_USER_PROFILE表中插入一条记录，该记录的UserID列中含有用户的用户名，Attribute（属性）列含有ROLES，Char_Value列含有角色名称。

注意 当禁用角色时，如果在UserID列中键入PUBLIC或%，便禁止了全部用户的角色。%或PUBLIC只能用于被授权为PUBLIC的角色。

Oracle用户SYSTEM拥有PRODUCT_USER_PROFILE表。当SYSTEM登录时，SQL*Plus不读取PRODUCT_USER_PROFILE表。因此，没有任何SQL*Plus限制应用到SYSTEM用户上。

注意 PRODUCT_USER_PROFILE表提供的产品级安全策略只由SQL*Plus实施，而不是由Oracle RDBMS实行。

26.5 追踪SQL语句

在SQL*Plus 3.3之前，获得有关在SQL*Plus中SQL语句的执行情况的统计手段非常有限。用于统计目的的命令只有SET TIME和SET TIMING。

SET TIME命令控制SQL*Plus如何显示当前时间。ON参数在每个SQL*Plus提示符前面显示当前时间，OFF参数关闭当前时间的显示。SET TIME命令的语法是：

SET TIME [OFF | ON] （OFF是缺省值）

清单26-15显示了使用SET TIME命令的一个示例。

清单26-15 估算运行时间

```
SQL> SHOW TIME
time OFF
SQL> SET TIME ON
01:17:57 SQL> SELECT t1.dname, t2.ename, t2.sal, t2.job
01:18:11 2 FROM dept t1, emp t2
01:18:23 3 WHERE t1.deptno = t2.deptno;
```

DNAME	ENAME	SAL	JOB
RESEARCH	SMITH	800	CLERK
SALES	ALLEN	1600	SALESMAN
SALES	WARD	1250	SALESMAN
RESEARCH	JONES	2975	MANAGER
SALES	MARTIN	1250	SALESMAN
SALES	BLAKE	2850	MANAGER
ACCOUNTING	CLARK	2450	MANAGER
RESEARCH	SCOTT	3000	ANALYST
ACCOUNTING	KING	5000	PRESIDENT
SALES	TURNER	1500	SALESMAN
RESEARCH	ADAMS	1100	CLERK
SALES	JAMES	950	CLERK
RESEARCH	FORD	3000	ANALYST
ACCOUNTING	MILLER	1300	CLERK

```
14 rows selected.
01:18:43 SQL>
```

通过用查询执行后第一个提示符显示的时间（01:18:43）减去最后一个输入行处显示的时间（01:18:23），用户可以得到一个执行这个查询所消耗的时间估算值（包含用于输入WHERE子句的时间）。

使用这种方法计算得出的运行时间不能反映 Oracle 执行一个查询的真正速度，因为流逝的时间包含将查询结果显示到 SQL*Plus 界面的时间消耗，运行查询的实际时间要短得多。

SET TIMING 命令控制执行时间统计的显示，SET TIMING 命令的语法为：

```
SET TIMING[ON | OFF] (OFF 是缺省值)
```

SET TIMING ON 命令显示 SQL*Plus 中执行的每一条 SQL 语句或 PL/SQL 块的执行时间统计。SET TIMING OFF 关闭时间统计的显示。

清单 26-16 显示了使用 SET TIMING 命令的一个示例。

清单 26-16 执行的计时

```
SQL> SHOW TIMING
timing OFF
SQL> SET TIMING ON
SQL> SELECT t1.dname, t2.ename, t2.sal, t2.job
       2 FROM dept t1, emp t2
       3 WHERE t1.deptno = t2.deptno;
DNAME          ENAME          SAL      JOB
-----
RESEARCH       SMITH             800      CLERK
SALES          ALLEN            1600     SALESMAN
SALES          WARD             1250     SALESMAN
RESEARCH       JONES            2975     MANAGER
SALES          MARTIN           1250     SALESMAN
SALES          BLAKE            2850     MANAGER
ACCOUNTING     CLARK            2450     MANAGER
RESEARCH       SCOTT            3000     ANALYST
ACCOUNTING     KING             5000     PRESIDENT
SALES          TURNER           1500     SALESMAN
RESEARCH       ADAMS            1100     CLERK
SALES          JAMES             950      CLERK
RESEARCH       FORD             3000     ANALYST
ACCOUNTING     MILLER           1300     CLERK
14 rows selected.
real: 2690
```

如清单 26-15 和清单 26-16 所示，SET TIME 和 SET TIMING 命令提供了在 SQL*Plus 中 SQL 语句和（或）PL/SQL 块执行情况的非常基本且不是非常精确的计时统计。在 SQL*Plus 3.3 以前版本的 SQL*Plus 中，不能直接获得关于优化器使用的执行（访问）路径的任何信息。

SQL*Plus 3.3 版以及更高版本中使用了一个称为 AUTOTRACE 的新功能。AUTOTRACE 使 SQL*Plus 用户可以在 SQL*Plus 中查看数据库管理语言（DML）语句（SELECT、INSERT、UPDATE 和 DELETE）的执行计划，而不必使用 Oracle 提供的普通工具（SQL_TRACE、EXPLAIN PLAN 和 TKPROF 工具）来追踪 SQL 语句的执行情况。

要使用该功能，数据库系统管理员必须作为 SYS 用户运行一个称为 PLUSTRACE.SQL 的 SQL 脚本。PLUSTRACE.SQL 文件的位置依赖于操作系统。PLUSTRACE 脚本完成下列任务：

创建一个名为 PLUSTRACE 的角色。

为角色 PLUSTRACE 授予 V\$SESSTAT、V\$STATNAME 和 V\$SESSION 表的 SELECT 权限。

使用 ADMIN 选项将 PLUSTRACE 角色授权给 DBA 角色。

要让用户具有 AUTOTRACE 能力，必须满足三个条件：

用户必须被任何具有 DBA 角色的用户授予 PLUSTRACE 角色。

用户必须在自己的模式中创建一个 PLAN_TABLE 表。用户可以运行 utlxplan.sql 脚本完成这项工作。在 UNIX 操作平台下该脚本文件位于目录 \$ORACLE_HOME/rdbms/admin 中，在 Windows 95/NT 操作系统下该脚本位于目录 %ORACLE_HOME%\Rdbms8x\Admin 下。

必须适当地设置 SET 系统变量 AUTOTRACE。

在这些步骤执行后，在成功地运行任何 DML 语句（SELECT、INSERT、DELETE 和 UPDATE）后，用户能够得到关于优化器使用的执行路径的一个报告和执行统计。报告的输出由系统变量 AUTOTRACE 控制。

在表 26-5 中列出系统变量 AUTOTRACE 允许的設置和它們的使用結果。

表 26-5 AUTOTRACE 值

值	结 果
SET AUTOTRACE OFF	缺省值。不生成任何报告
SET AUTOTRACE ON EXPLAIN	追踪报告只显示执行路径，不显示执行统计
SET AUTOTRACE ON STATISTICS	追踪报告只显示执行统计，不显示执行路径
SET AUTOTRACE ON	追踪报告同时显示执行路径和执行统计
SET AUTOTRACE TRACEONLY	与 SET AUTOTRACE ON 相同，但是不显示查询的结果

26.5.1 理解执行计划

执行计划显示当执行一个查询时优化器使用的访问路径。使用 EXPLAIN PLAN 命令生成执行计划的输出。执行计划中显示的每一行都有一个连续的行号，也显示父操作的行号。执行计划由四列组成（对于标准查询，只显示三列，只有在分布式查询或使用并行查询选项（Parallel Query Option, PQO）运行查询的情况下才显示第四列）。

表 26-6 显示列的名称和它们的说明。

表 26-6 执行计划列说明

列 名	说 明
ID_PLUS_EXP	显示每步执行步骤的行号
PARENT_ID_PLUS_EXP	显示某步骤和它的父步骤之间的关系
PLAN_PLUS_EXP	显示报告的每步骤；例如 TABLE ACCESS (FULL) OF 'DEPT'
OBJECT_NODE_PLUS_EXP	显示使用的数据库链接或并行查询服务器（只有在运行分布式查询或使用并行查询选项（PQO）时）

缺省的列格式通常在站点环境资源文件（glogin.sql 文件）中设置。

列的格式可以使用 SQL*Plus 命令 COLUMN 加以改变。可以更改列的宽度、重新命名列或停止显示列。例如，要禁止显示 ID_PLUS_EXP 列，应键入：

```
SQL> COLUMN ID_PLUS_EXP NOPRINT
```

此语句追踪报告的第二部分显示语句执行的统计。它们显示用于执行查询的系统资源和它们的用法。与执行路径不同，不能改变统计报告的缺省格式。

26.5.2 使用AUTOTRACE功能

本节详细介绍为使SCOTT用户能够在SQL*Plus中使用AUTOTRACE功能所需要的步骤。

首先，用户SYSTEM登录到SQL*Plus，并授予用户SCOTT PLUSTRACE角色（SYS用户已在先前运行过 \$ORACLE_HOME/sqlplus/admin/plustrce.sql脚本，在数据库中创建了PLUSTRACE角色）。清单26-17显示该项工作已经完成。

清单26-17 授权PLUSTRACE角色

```
SQL>

C:\>sqlplus system

SQL*Plus: Release 8.1.3.0.0 - Beta on Sun Mar 7 09:25:09 1999

(c) Copyright 1998 Oracle Corporation. All rights reserved.

Enter password:

Connected to:
Oracle8 Enterprise Edition Release 8.1.3.0.0 - Beta
With the Partitioning and Objects options
PL/SQL Release 8.1.3.0.0 - Beta
SQL> GRANT plustrace TO scott;
Grant succeeded.
```

接着，用户SCOTT登录到SQL*Plus，并通过运行 \$ORACLE_HOME/rdbms80/admin/utlxplan.sql脚本在他的模式中创建PLAN_TABLE（这是EXPLAIN PLAN命令需要的），如清单26-18所示。

清单26-18 创建PLAN_TABLE

```
SQL> CONNECT scott
Enter password: *****
Connected.
SQL> @$ORACLE_HOME/rdbms80/admin/utlxplan
Table created.
SQL> L
 1 create table PLAN_TABLE (
 2   statement_id    varchar2(30),
 3   timestamp       date,
 4   remarks         varchar2(80),
 5   operation       varchar2(30),
 6   options         varchar2(30),
 7   object_node     varchar2(128),
 8   object_owner    varchar2(30),
 9   object_name     varchar2(30),
10   object_instance numeric,
11   object_type     varchar2(30),
12   optimizer       varchar2(255),
13   search_columns  numeric,
14   id              numeric,
15   parent_id       numeric,
16   position        numeric,
17   cost            numeric,
18   cardinality     numeric,
19   bytes           numeric,
20   other_tag       varchar2(255),
21*  other           long)
```

L[IST]命令列出 SQL 缓冲区中的内容，它含有从 ORACLE_HOME/rdbms80/admin/utlxplan.sql 脚本中执行的最后一条 SQL 命令——CREATE TABLE plan_table 语句。

接着，SCOTT 用户输入并执行他希望追踪的查询。清单 26-19 显示了一个示例。

清单 26-19 追踪一个查询

```
SQL> SELECT t1.dname, t2.ename, t2.sal, t2.job
2 FROM dept t1, emp t2
3 WHERE t1.deptno = t2.deptno;
DNAME          ENAME          SAL      JOB
-----
RESEARCH        SMITH           800      CLERK
SALES            ALLEN           1600     SALESMAN
SALES            WARD            1250     SALESMAN
RESEARCH        JONES           2975     MANAGER
SALES            MARTIN          1250     SALESMAN
SALES            BLAKE           2850     MANAGER
ACCOUNTING      CLARK           2450     MANAGER
RESEARCH        SCOTT           3000     ANALYST
ACCOUNTING      KING            5000     PRESIDENT
SALES            TURNER          1500     SALESMAN
RESEARCH        ADAMS           1100     CLERK
SALES            JAMES           950      CLERK
RESEARCH        FORD            3000     ANALYST
ACCOUNTING      MILLER          1300     CLERK
14 rows selected.
```

要同时得到执行路径和执行统计的信息，用户 SCOTT 必须正确地设置 AUTOTRACE 变量。清单 26-20 显示了用于设置 AUTOTRACE 为打开状态并从缓冲区中运行先前的查询的命令。

清单 26-20 追踪缓冲区中的查询

```
SQL> SET AUTOTRACE ON
SQL> /
Execution Plan
-----
0      SELECT STATEMENT Optimizer=CHOOSE
1  0    NESTED LOOPS
2    1    TABLE ACCESS (FULL) OF 'EMP'
3    1    TABLE ACCESS (BY ROWID) OF 'DEPT'
4    3    INDEX (UNIQUE SCAN) OF 'DEPT_PRIMARY_KEY' (UNIQUE)
Statistics
-----
0  recursive calls
2  db block gets
43 consistent gets
0  physical reads
0  redo size
726 bytes sent via SQL*Net to client
376 bytes received via SQL*Net from client
3  SQL*Net roundtrips to/from client
0  sorts (memory)
0  sorts (disk)
14 rows processed
```

不显示查询数据，追踪相同的语句，如下所示：

```
SQL> SET AUTOTRACE TRACEONLY
SQL> /          - slash command: doesn't display query text
14 rows selected.
Execution Plan
-----
```

```

0      SELECT STATEMENT Optimizer=CHOOSE
1      0      NESTED LOOPS
2      1      TABLE ACCESS (FULL) OF 'EMP'
3      1      TABLE ACCESS (BY ROWID) OF 'DEPT'
4      3      INDEX (UNIQUE SCAN) OF 'DEPT_PRIMARY_KEY' (UNIQUE)
Statistics
-----
0 recursive calls
2 db block gets
43 consistent gets
0 physical reads
0 redo size
726 bytes sent via SQL*Net to client
376 bytes received via SQL*Net from client
3 SQL*Net roundtrips to/from client
0 sorts (memory)
0 sorts (disk)
14 rows processed

```

提示 当调整一个返回大量记录的查询时，TRACEONLY选项特别有用。

执行统计中涉及的客户是 SQL*Plus。不管 SQL*Net 是否被安装并使用，SQL*Net 是指客户（SQL*Plus）和服务器（Oracle RDBMS）之间通常的进程间的通信。

26.6 SQL*Plus 8.1版的增强

SQL*Plus 8.1版给予用户实现下列功能的能力：

访问所有的 Oracle 8i 数据类型和用户定义的数据类型。

访问所有 Oracle 8i 引入的 SQL 新特性。

SQL*Plus 8.1 版拥有 8.0 版具有的全部特性。

而且，服务管理器功能目前在 SQL*Plus 中已可使用。这些功能包括：启动、关闭、显示参数、显示系统全局区（SGA）、归档日志、恢复数据库和内部连接等。

有一个名为 ARCHIVE LOG 的新命令，ARCHIVE LOG 命令使重做日志文件被归档，它还显示重做日志文件的信息。

例如：

```

SQL> connect internal
Connected.
SQL> archive log list
Database log mode          No Archive Mode
Automatic archival         Enabled
Archive destination        D:\ORANT\oradata\orcl2\archive
Oldest online log sequence 2
Current log sequence       5
SQL>

```

你现在可以在 SQL*Plus 中使用 RECOVER 命令恢复一个数据库。RECOVER 命令执行表空间、数据文件或整个数据库的介质恢复。

例如：

```

SQL> recover database
ORA-00283: recovery session canceled due to errors
ORA-01124: cannot recover data file 1 - file is in use or recovery
ORA-01110: data file 1: 'D:\ORANT\ORADATA\ORCL2\SYSTEM01.DBF'

```

SET 命令现在有一个 INSTANCE 子句，INSTANCE 子句将会话的缺省实例改变为指定的实例。SET 命令现在还有一个 LOGSOURCE 子句，LOGSOURCE 子句指定数据库恢复期间检索

归档日志的路径。

现在你可以在SQL*Plus中使用SHOW PARAMETERS命令查看init.ora参数。

例如：

```
SQL> show parameters block
```

NAME	TYPE	VALUE
db_block_buffers	integer	1000
db_block_checking	boolean	TRUE
db_block_checksum	boolean	FALSE
db_block_lru_latches	integer	1
db_block_max_dirty_target	integer	1000
db_block_size	integer	2048
db_file_multiblock_read_count	integer	8
hash_multiblock_io_count	integer	1
sort_multiblock_read_count	integer	2

可以在SQL*Plus中查看系统全局区（SGA）信息。

例如：

```
SQL> show sga
```

```
Total System Global Area  20396276 bytes
Fixed Size                  63732 bytes
Variable Size              18210816 bytes
Database Buffers          2048000 bytes
Redo Buffers               73728 bytes
SQL>
```

可以在SQL*Plus中使用带有安装和打开一个数据库的选项的STARTUP命令来启动数据库。

例如：

```
SQL> startup
ORACLE instance started.
Total System Global Area      20396276 bytes
Fixed Size                    63732 bytes
Variable Size                 18210816 bytes
Database Buffers              2048000 bytes
Redo Buffers                  73728 bytes
Database mounted.
Database opened.
SQL>
```

可以在SQL*Plus中使用SHUTDOWN命令关闭一个数据库，SHUTDOWN命令关闭当前正在运行的一个Oracle实例。该命令也可以关闭和卸载一个数据库。

例如：

```
SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL>
```

SQL*Plus是最经常用来和数据库交互并执行系统管理功能如启动、关闭或恢复数据库等的唯一工具。它是一个交互式的工具，允许用户创建、修改、删除或查询数据库对象如表、视图、索引和约束。它还允许具有适当权限的用户创建新的用户，向他们授予特权或修改现有的用户。拥有适当权限的用户能够在SQL*Plus中删除其他用户和他们的权限。在Oracle 8i中，所有的服务器管理器命令都合并入SQL*Plus，所以数据库系统管理员能够使用这个工具有效地管理他们的数据库。