第一篇　SQL * Plus 3.1

第一章　SQL * Plus 概述

§1.1　数据库的基本概念

1.1.1　什么是数据库和数据库管理系统

数据库是数据的集合，数据库管理系统是为管理数据库而配置的软件工具，数据库管理系统的主要功能有：

- 存储、检索和修改数据
- 保护数据的一致性
- 解决并发性问题
- 为数据提供一个通用的界面
- 限制用户访问数据库的权限

1.1.2　什么是表和视图

每个 ORACLE 数据库都具有相应的物理结构和逻辑结构。物理结构决定了数据库信息在磁盘上的真正的物理存储结构，而逻辑结构是用户所涉及的数据库结构。ORACLE 数据库的逻辑结构由两类因素决定：

- 逻辑存储结构(如表空间、段和范围等)
- 数据库模式对象(如表、视图等)

逻辑存储结构将支配一个数据库的物理空间如何使用，而数据库模式对象及它们之间的相互联系组成了一个数据库的关系设计。

表和视图是比较常用的两种数据库模式对象。在多数情况下，用户在建立、检索和修改数据库时，直接面对的就是表和视图。

表是数据库中数据存储的基本单位，其数据按行和列存储。每个表都有一个表名，表中的每一列都有一个列名，每一列中各行的数据都具有相同数据类型和宽度，而每一行中的数据则是对应于某个单个记录的列的信息的集合。

下面给出两个表的例子，它们在本书后面的章节中将多次被用到。

例 1.1　职工情况表(EMP 表)

EMP 表：

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

该例中，表名为 EMP。各列的列名分别为 EMPNO、ENAME、JOB、MGR、HIREDATE、SAL、COMM 和 DEPTNO。每一列的数据都具有相同的数据类型，如 HIREDATE 列的数据均为日期型，而每一行的数据则记录了每个特定职工的职工号及相应的姓名、工作、工资等个人情况。

例 1.2 部门情况表

DEPT 表：

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

本例中，表名为 DEPT，表中的数据记录了每个部门的部门号及相应的名称和地址。

视图是另一种常用的模式对象。视图是一种虚拟表，其数据是由其它表和视图中提取的。视图具有很多与表相类似的特点，它也可以被查询、修改或删除，但是视图和表的一个重要差别在于：视图并不真正的包含数据，一个视图不分配任何存储空间。通常，将视图所基于的表称为基表。

引入视图的好处在于：

(1)提供了附加的安全级

提供视图，使得用户只能通过视图看到其有权看到的信息，而无权看到基表中的全部信息。

(2)隐藏数据的复杂性

通过构造视图，使得用户只看到所需的信息，而不必查看所有的信息，从而使所提供的信息更加简单。

可以提供变换列名的手段，而无需真正地改变基表的定义。

在第 5 章中将详细介绍如何建立视图，从视图的建立过程中，可以更清楚地理解视图的含义和优点。

1.1.3 什么是数据字典

数据字典是ORACLE数据库的重要组成部分之一。数据字典由一组只读的表及其视图组成，它提供有关该数据库的信息，如每个ORACLE用户的名字及被授予的特权和角色、每个模式对象的名字、数据库中对象的空间分布信息及当前的使用情况等等。数据字典是只读的，其信息只允许被查询。

§1.2 SQL语言概述

1.2.1 什么是SQL语言

SQL(结构化查询语言)语言是一种标准的关系数据库语言，用于建立、存储、修改、检索和管理ORACLE数据库中的信息。它有三个显著的优点：

(1)它是一种非过程化的语言。因为它每次处理一个记录集而不是每次处理一个单个记录，而且SQL对数据提供导航，这意味着用户可以在高层的数据结构上工作，而不必指定对数据的存取方法；

(2)SQL是一种所有用户都可以使用的统一语言，包括系统管理员、数据库管理员、应用程序员，决策支持系统人员及许多其它类型的终端用户，SQL可以用于各种数据库活动，包括数据查询、在表中插入、修改和删除行、建立、修改和删除数据对象、控制对数据库和数据对象的存取，保证数据库的一致性和完整性等。

(3)SQL是所有关系数据库的公共语言，用户可以方便地移植用SQL语言编写的程序。

1.2.2 SQL语言简介

本节介绍SQL语言的一些基本概念。

(1)模式对象

数据库模式对象及它们之间的相互联系组成了一个数据库的关系模型。在模式对象中最常用的是表和视图。

(2)SQL语言的常量

SQL语言的常量包括文本常量、整数常量和数值常量。文本常量用于指定一个文本或字符串，例如'JACK'、'101'均为文本常量；整数常量用于指定一个正整数；数值常量用于指定一个整数或实常数。

(3)SQL语言的数据类型

由ORACLE操作的每一个常量或列值都属于某一种数据类型，一个值的数据类型将一组固定的特性与该值相联系起来，这些特性使得ORACLE以不同的方式来处理不同类型的数据。

ORACLE支持的内部数据类型如下：

VARCHAR2(SIZE)：变长字符串，长度SIZE由用户指定，SIZE最大为2000字节。

NUMBER：数值型。

LONG：变长字符数据，最长为2G字节。

DATE：日期型，有效日期范围从 B. C. 4712 年 1 月 1 日到 A. D. 4712 年 12 月 31 日。

RAW(SIZE)：长度为 SIZE 的二进制数据，SIZE 由用户指定，最长不能超过 255 字节。

LONG RAW：变长二进制数据，最长为 2G 字节。

RAWID：十六进制串，用于表示表的行的唯一地址。

CHAR(SIZE)：固定长度的字符数据，长度为 SIZE 字节，SIZE 最大为 255，缺省值为 1。

(4)空值(NULL)

如果一行数据在某列没有值，则称该列是空值(NULL)或称该列包含一个空值(NULL)。应当注意，不能用空值来表示取值为 0，它们并不等价。任何包含空值的表达式的计算结果仍为空值。

(5)SQL 语言的运算符

SQL 语言的运算符包括算术运算符、字符运算符、比较运算符、逻辑运算符、集合运算符以及(+)等其它类型的运算符。

算术运算符：

符　号	功　能	例　子
+，-	表示正的或负的表达式	-SAL
*，/	乘除运算	SAL * 1.1
+，-	加减运算	SAL+COMM

字符运算符：

符　号	功　能	例　子
||	用于连接两个字符串	'NAME IS' || 'JOHN' 等价于 'NAME IS JOHN'

比较运算符：

符号	含　义
=	相等测试
!=，^=，<>	不等测试
>	大于
<	小于
>=	大于或等于
<=	小于或等于

此外，比较运算符还包括 IN、BETWEEN…AND、LIKE、IS NULL、NOT 等，它们的作用将在第 3 章中详细介绍。

逻辑运算符：

符号	含义
NOT	逻辑非
AND	逻辑与
OR	逻辑或

集合运算符包括 UNION、INTERSECT 和 MINUS 等，它们的作用将在第 3 章中详细介绍。

(6)SQL 语言的函数

SQL 语言提供了各种各样的内部函数供用户使用，第 6 章中将详细介绍 SQL 语言的函数。

(7)SQL 语言的表达式

表达式由一个或多个值(常量或列值)、运算符和函数组合而成，并且可以计算出一个值，表达式的计算结果的数据类型通常与它的成分的数据类型相同。例如，COMM/SAL 就是一个表达式。

(8)SQL 语言的条件

条件是由一个或多个含有比较运算符的表达式及逻辑运算符组合而成的，其计算值是 TRUE、FALSE 或 NULL。例如，SAL>1000 AND COMM<1000 就是一个条件。

(9)SQL 语言的命令

SQL 语言提供了用于建立、存储、修改、检索和管理 ORACLE 数据库的多种命令，本书将在后面的章节中予以详细介绍。

§1.3 SQL * Plus 和 SQL、PL/SQL

SQL 是一种关系数据库语言，而 PL/SQL 是 SQL 的过程化扩充语言，它将非过程化的关系数据库语言 SQL 和过程化的程序语言结合起来，使得用户可以用过程化的逻辑将 SQL 语言的若干命令连接起来，从而使用过程化方法(如循环、分支等)来处理数据。SQL * Plus 是 ORACLE 提供的一种开发工具，在 SQL * Plus 环境中，可以输入、修改、编辑和运行关系数据库语言 SQL 及其过程化扩充语言 PL/SQL 的全部命令，还可以完成许多附加的功能，如格式化 SQL 的查询结果、在 SQL 数据库之间拷贝数据等。

SQL 命令和 PL/SQL 块都可以在 SQL * Plus 这一开发工具中使用，除此之外，SQL * Plus 还提供了不同于 SQL 命令和 PL/SQL 块的专有命令，用于完成一些附加功能。

SQL * Plus 的主要功能有：

- 输入、编辑、存储、检索和运行 SQL 命令及 PL/SQL 块；
- 对查询结果进行计算并且以报表的形式显示和输出；
- 显示表的列定义；
- 在 SQL 数据库中拷贝数据；
- 向终端用户发送信息或从终端用户接收应答。

SQL 语言及其过程化扩充语言 PL/SQL 是使用 SQL * Plus 的重要基础。

在 SQL * Plus 中，SQL 命令、PL/SQL 块和 SQL * Plus 命令在输入、编辑和运行方法上有一些差异。因此，在使用 SQL * Plus 时，应当区分运行的是哪一种类型的命令。本书将在后面的有关章节中详细介绍。

参考文献

(1) SQL * Plus User's Guide and Reference Version 3.1 , Part No. 5142－31－1192
(2) ORACLE 7 Server SQL Language Reference Manaual ,Part No. 778－70－1292
(3) ORACLE 7 Server SQL Language Quick Reference ,Part No. 5421－70－1292
(4) SQL * Plus Quick Reference Version 3.1 Part No. 3703－31－1192

第二章　SQL * Plus 界面

SQL * Plus 为用户提供了很方便的界面环境，使得用户可以在 SQL * Plus 环境中输入、编辑和运行 SQL，SQL * Plus 命令和 PL/SQL 块，还可以使用户随时获得帮助信息。本章将介绍如何利用 SQL * Plus 的界面环境进行最基本的操作，包括登录和退出 SQL * Plus、命令的编辑与运行及如何获得帮助信息。

§2.1　登录和退出 SQL * Plus

2.1.1　登录 SQL * Plus

登录 SQL * Plus 的具体步骤如下：

(1)在操作系统提示符下，输入 SQLPLUS，然后回车。

这时屏幕将显示版本号、日期和版权信息，并提示用户输入用户名，如下所示。

SQL * Plus：Version 3.1.3 — Production on Fri April 10 09:39:26 1992

Copyright (c) Oracle Corporation 1979，1992. All rights reserved.

Enter user—name：

(2)输入正确的 ORACLE 用户名并回车，这时 SQL * Plus 将提示用户输入口令，即显示：

Enter password：

(3)输入用户口令并回车；

这时 SQL * Plus 将检查用户名和口令，如果正确，SQL * Plus 将显示用户所连接的 ORACLE 版本号及用户可使用的开发工具的版本，然后 SQL * Plus 显示命令提示符：

SQL>

这意味着用户已经成功地登录到 SQL * Plus 中，可以使用各种 SQL * Plus 允许的命令。

如果输入的用户名或口令不正确，SQL * Plus 将提示用户重新输入用户名和口令，如果三次登录失败，SQL * Plus 将自动返回到操作系统下。

用户也可以用另一种简化的方法登录 SQL * Plus，即在系统提示符下直接输入：

SQLPLUS 用户名/口令（回车）

这样可以直接进入 SQL * Plus 提示符状态。

2.1.2　退出 SQL * Plus

在 SQL * Plus 提示符后输入 EXIT，可以退出 SQL * Plus，返回到操作系统。

§2.2　命令的编辑与运行

在 SQL * Plus 中，编辑和运行 SQL 命令、PL/SQL 块和 SQL * Plus 命令的方法并不完

全相同。对于每一种命令，SQL * Plus 都提供了多种编辑和运行方法。下面详细介绍。

2.2.1 编辑和运行 SQL 命令

(1)在命令提示符后输入 SQL 命令并运行

为了使命令更具可读性，SQL * Plus 允许将每一个 SQL 命令划分为多行，划分点可以在任意两个单词之间，但不能把一个单词分写在两行中，输入完一行并回车后，SQL * Plus 将显示新的行号并允许在新的一行上输入命令，直到输入某个特殊的字符表示结束一个 SQL 命令的输入过程。SQL * Plus 可以采用下述三种方法结束一个 SQL 命令：

· 在命令的最后一行的尾部加上分号(;)并回车，这时系统将所输入命令的全部内容存入 SQL 缓冲区中以备后用并运行该命令，然后显示运行结果。运行完毕后显示新的命令提示符

SQL>

这时可以输入新的命令。

· 在独自一行输入斜杠 / 并回车(即在命令的最后一行输完后回车，在新的行号提示符后输入斜杠/并回车)，则系统将所有输入的命令存入 SQL 缓冲区并运行该命令，然后显示运行结果。运行完毕后显示新的命令提示符

SQL>

· 在新的输入行输入一空行并回车，则系统将所有输入的命令存入 SQL 缓冲区，并显示新的命令提示符。与前面两种结束 SQL 命令的方法不同，这种方法并不运行 SQL 命令。

例 2.1 从 EMP 表中找出所有工资大于 2500 元的职工的职工号、职工姓名、工作和工资记录。这时需要输入一个 SQL 命令：

```
SELECT EMPNO,ENAME,JOB,SAL FROM EMP WHERE SAL>2500
```

我们将这个命令分为多行输入，并以上述三种方式结束命令的输入过程。

· 以分号结束输入

```
1  SQL>SELECT
2  EMPNO,ENAME,JOB,SAL
3  FROM EMP
4  WHERE SAL>2500;
```

· 以斜杠结束输入

```
SQL>SELECT
2  EMPNO,ENAME,JOB,SAL
3  FROM EMP
4  WHERE SAL>2500
5  /
```

· 以空行结束输入

```
SQL>SELECT
2  EMPNO,ENAME,JOB,SAL
3  FROM EMP
4  WHERE SAL>2500
```

前两种方法的运行结果如下(第三种方法并不能运行所输入的 SQL 命令):

EMPNO	ENAME	JOB	SAL
7566	JONES	MANAGER	2975
7698	BLAKE	MANAGER	2850
7788	SCOTT	ANALYST	3000
7839	KING	PRESIDENT	5000
7902	FORD	ANALYST	3000

需要注意的是,每当结束一个新的 SQL 命令时,SQL 缓冲区中旧的命令就被清除,而新的 SQL 命令将被存入 SQL 缓冲区中。SQL 命令存入缓冲区时,结束符(;)和(/)将被自动取消。

(2)利用 SQL 缓冲区进行命令编辑和运行

前面介绍的三种结束 SQL 命令的方法都可以将 SQL 命令存入 SQL 缓冲区中,后面还将介绍一些其它的将 SQL 命令存入缓冲区中的方法。为什么要将 SQL 命令存入缓冲区呢?这是因为将命令存入 SQL 缓冲区后,用户如果想修改正在输入的 SQL 命令,可以不必重新输入该命令,只需在 SQL 缓冲区中对该命令进行修改。然后重新运行 SQL 缓冲区中的命令。为此,SQL * Plus 提供了多种编辑 SQL 缓冲区中命令的方法。

· 利用编辑命令编辑缓冲区中的内容

表 1.1 列出了 SQL * Plus 提供的可以用于编辑缓冲区内容的命令及其含义。这些命令都是 SQL * Plus 命令,运行它们的方法很简单,只需在命令提示符(SQL>)后面输入命令并回车即可。

表 1.1 SQL * Plus 编辑命令

命令	缩写	说明
APPEND text	A text	在当前行的最后增加 text
CHANGE /old/new	C /old /new	在当前行中将 old 变为 new
CHANGE /text	C /text	删除当前行的 text
CLEAR BUFFER	CL BUFF	删除所有的行
DEL	(无)	删除当前行
INPUT	I P	增加一行或多行
INPUT text	I text	增加一行 text
LIST	L	显示 SQL 缓冲区中的所有行
LIST n	L n 或 n	显示一行
LIST *	L *	显示当前行
LIST LAST	L LAST	显示最后一行
LIST m,n	L m,n	显示 m 行到 n 行

下面举例介绍表 1.1 中的命令的使用方法

例 2.2 清除缓冲区的内容—CLEAR BUFF

在命令提示符后输入 CLEAR BUFFER 并回车，即

SQL>CLEAR BUFFER

这时，SQL 缓冲区中的所有内容都被清除。

例 2.3 显示缓冲区的内容—LIST 命令

假设缓冲区中存放着如下的 SQL 命令：

```
1  SELECT
2  EMPNO,ENAME,JOB,SAL
3  FROM EMP
4  WHERE SAL>2500
```

(a)SQL>LIST

SQL * Plus 将显示

```
1  SELECT
2  EMPNO,ENAME,JOB,SAL
3  FROM EMP
4 * WHERE SAL>2500
```

(b)SQL>LIST 1,2

将显示

```
1  SELECT
2 * EMPNO,ENAME,JOB,SAL
```

(c)SQL>LIST 2

将显示

```
2 * EMPNO,ENAME,JOB,SAL
```

(d)在(c)的基础上，输入如下命令：

SQL>LIST *

将显示

```
2 * EMPNO,ENAME,JOB,SAL
```

各例中的命令含义显而易见，需要说明的是每个例子中的运行结果中都有一行 SQL 命令带有 *，它的含义是说明该命令行为当前行。从例 2.2 可以看出 LIST 、LIST m,n 和 LIST n 都将所显示缓冲区内容的最后一行作为当前行，而 LIST * 不影响当前行的位置，因为它只是显示当前行的内容。定义当前行是为了方便表 1.1 中其它的编辑命令的使用，因为除了 LIST 和 CLEAR BUFFER 外，表 1.1 中的其它命令都是只对当前行有效(即这些命令只能修改当前行的内容)。可以用 List 改变当前行的方法来决定编辑哪一行。

决定哪一行是当前行的 SQL * Plus 命令还有 RUN(后面将详细介绍其功能)，它将运行缓冲区中的命令，并使缓冲区的最后一行成为当前行。

使用表 1.1 中的其它的编辑命令时，都应该先使 SQL 缓冲区中将被修改的行置成为当前行，然后再进行修改。

将命令存入缓冲区时，SQL * Plus 自动将(;)或(/)去掉，这样处理可以给编辑缓冲区带来方便。例如，可以直接在缓冲区末端加入新行。缓冲区的命令不需要用分号(;)来运行，而是有专门的运行命令。

例 2.4 编辑缓冲区的当前行－CHANGE

CHANGE 命令可以修改当前行的内容。

假设缓冲区的内容同例 2.3

(a)假设第 4 行为当前行，现在修改第 4 行，将 2500 该成 2000。

SQL>CHANGE /2500/2000

屏幕上显示修改后的结果：

4 * WHERE SQL>2000

(b)假设第 2 行为当前行，现在对第 2 行进行修改，将 EMPNO 去掉。

SQL>CHANGE /EMPNO,

屏幕上显示修改结果：

2 * ENAME,JOB,SAL

例 2.5 增加一个新行－INPUT 命令

INPUT 命令在 SQL 缓冲区的当前行后面增加新行。

假设缓冲区的内容同例 2.3，第 4 行为当前行，现在要增加第 5 行

SQL>INPUT

SQL * Plus 显示行号提示符 5，提示用户输入新命令，在行号提示符后输入新的内容，即：

5 ORDER BY SAL

回车后，SQL * Plus 显示行号提示符 6，可以继续加入新的一行，也可以输入空行结束插入新行的过程。

INPUT 命令的另一个作用是直接将 SQL 命令写入缓冲区(在命令提示符后输入 SQL 命令的方法只适在全部 SQL 命令输入完才将所输入的命令存入缓冲区)。

例如，按照下述步骤运行：

SQL>CLEAR BUFFER

SQL>INPUT

就可以直接在缓冲区中输入命令并进一步用各种编辑方法来修改。

例 2.6 在缓冲区当前行的尾部追加文本－APPEND 命令

APPEND 命令可以在缓冲区当前行的尾部添加文本。

假设缓冲区的内容同例 2.5，第 5 行为当前行。

在第 5 行的尾部添加 DESC。

SQL>APPEND DESC

SQL * Plus 显示修改结果：

5 * ORDER BY SAL DESC

例 2.7 删除缓冲区的当前行－DEL 命令

假设缓冲区的内容同例 2.6，第 5 行为当前行，现在要删除第 5 行。

SQL>DEL

·利用系统文本编辑器编辑缓冲区内容

很多读者都使用过一种或多种文本编辑器，在 SQL * Plus 中也可以使用系统提供的文本编辑器来修改缓冲区中的 SQL 命令。当运行系统的文本编辑器后，SQL 缓冲区的内容将

作为被编辑的文本自动调入文本编辑器。可以利用文本编辑器的命令对其中的内容进行全屏幕编辑，当用户保存编辑过的文本并退出时，SQL * Plus 会自动将修改后的文本存入缓冲区以替代缓冲区中的旧命令。

运行系统文本编辑器的方法是：

SQL>EDIT

· 运行缓冲区中的命令

用 SQL * Plus 的 RUN 或 / 命令都可以运行 SQL 缓冲区中的命令。它们的区别是：RUN 将显示 SQL 缓冲区中的命令，并使缓冲区中的最后一行成为当前行，而 / 只是执行 SQL 缓冲区中的 SQL 命令，它并不显示缓冲区内容，也不改变当前行的位置。

(3)利用命令文件进行命令的编辑和运行

SQL * Plus 可以把一个或多个 SQL 命令存入文本文件，称为命令文件。用户可以用系统的编辑程序直接对命令文件进行编辑，然后将命令文件的内容调入缓冲区，以便运行 SQL 命令。

· 命令文件的生成

生成一个命令文件有两种途径：

(a)SAVE 命令将缓冲区的内容存入指定文件。命令格式为：

SAVE filename

其中 filename 为文件名，SQL * Plus 自动为其加上. SQL 的后缀。

(b)直接利用系统编辑程序建立命令文件。命令格式为：

EDIT filename

进入文本编辑器后，可以输入并编辑 SQL 命令，输入完毕后存盘退出，SQL * Plus 将自动生成 filename. SQL 的命令文件。

· 命令文件的运行

运行一个命令文件有两种方法：

(1)用 GET 命令将指定的命令文件调入 SQL 缓冲区，然后再执行缓冲区中的命令。命令格式为：

SQL>GET filename

SQL>RUN

(2)用 START 命令可直接执行指定的命令文件的内容，命令格式为：

SQL> START filename

2.2.2 编辑和运行 PL/SQL 块

(1)在命令提示符后输入 PL/SQL 块

与输入 SQL 命令不同，输入 PL/SQL 块之前，首先应当进入 PL/SQL 模式。可以采用两种方式进入 PL/SQL 模式：

· 在命令提示符后输入 PL/SQL 的关键词 DECLARE 或 BEGIN 并回车。

· 用 SQL 命令 CREATE 建立一个存贮过程(如 CREATE FUNCTION，CREATE PACKAGE，CREATE PROCEDURE 等)。

进入 PL/SQL 模式后，用户就可以继续输入 PL/SQL 块的其它内容，输入的过程同输

入 SQL 命令相同，即每输完一行并回车后，SQL * Plus 将显示新的行号提示符。结束一个 PL/SQL 块的方法与结束一个 SQL 命令的方法有所不同。(;)或独自一行的(/)不能用于标识一个 PL/SQL 块的结束。SQL * Plus 用独自一行的(.)来结束 PL/SQL 块，并将 PL/SQL 块存入缓冲区，但不执行，可以通过 RUN 命令执行存在缓冲区中的 PL/SQL 块。

(2)利用 SQL 缓冲区进行 PL/SQL 块的编辑和运行

在 2.2.1 中介绍的利用 SQL 缓冲区进行 SQL 命令的编辑和运行的方法同样适用于 PL/SQL 块的编辑和运行。使 PL/SQL 块存入缓冲区的方法主要有：

· 在命令提示符后输入 PL/SQL 块并用独自一行的(.)作为结束的标志，则 PL/SQL 块将存入缓冲区。

· 利用 INPUT 命令，将 PL/SQL 块直接存入缓冲区。

· 用 GET 命令将命令文件中的 PL/SQL 块调入缓冲区。

利用 SQL * Plus 的编辑命令(表 1.1)和系统文本编辑程序来编辑缓冲区中的 PL/SQL 块的方法同 2.2.1 中的介绍完全相同。RUN 和 / 同样可用于执行缓冲区中的 PL/SQL 块。

(3)利用命令文件进行 PL/SQL 块的编辑和运行与 2.2.1 中介绍的方法相同，可以采用同样的方法来建立、编辑和运行 PL/SQL 命令文件。

可以用 SAVE 命令将缓冲区中的 PL/SQL 块存入命令文件或直接用 EDIT 命令来建立包含 PL/SQL 块的命令文件，并利用系统编辑程序 EEIT 来修改和编辑命令文件，GET 和 START 可以分别将 PL/SQL 块的命令文件调入内存及执行。

2.2.3 编辑和运行 SQL * Plus 命令

(1)在命令提示符后输入和运行 SQL * Plus 命令

在命令提示符后可以直接输入 SQL * Plus 命令，其输入方法与输入 SQL 命令有两点不同：

· 续行的方法不同，如果要将一个 SQL * Plus 命令分为多行输入，则在每一行的结尾用连字符'—'来告诉 SQL * Plus 还要输入新的一行，SQL * Plus 则显示提示符>来告诉用户输入新的一行；

· 结束命令的方法不同：当最后一行 SQL * Plus 命令输入完毕后，直接回车即可。SQL * Plus 将执行所输入的 SQL * Plus 命令。但是，SQL * Plus 不将输入的命令存入缓冲区，这是输入 SQL * Plus 命令与输入 SQL 命令及 PL/SQL 块的一个主要区别。

例 2.8 执行 SQL * Plus 格式化查询命令

有两种输入方法：

(a) SQL>COLUMN SAL FORMAT $99999 HEADING SALARY

(b) SQL> COLUMN SAL —

>FORMAT $99999 HEADING SALARY

(2)利用 SQL 缓冲区编辑和运行 SQL * Plus 命令的方法

由于在命令提示符后输入的 SQL * Plus 命令不能存入缓冲区，所以使 SQL * Plus 命令进入缓冲区的方法有两种：

· 利用 INPUT 命令，将 INPUT 命令以后的内容直接写入缓冲区；

· 用 GET 命令将命令文件中的 SQL * Plus 命令调入缓冲区。

前面 2.2.1 中介绍的方法同样适用于编辑、修改和运行缓冲区中的 SQL * Plus 命令，本节不再详述。

(3)利用命令文件进行 SQL * Plus 命令的编辑和运行

前面介绍的方法同样适用于建立、编辑和运行包含 SQL * Plus 命令的命令文件，这里不再详述。

2.2.4 建立和编辑批处理命令文件

前面介绍了命令文件中只包含一个命令(SQL 或 SQL * Plus 命令)或只包含一个 PL/SQL 块的情况。事实上，命令文件中可以同时包含多个 SQL、SQL * Plus 命令及多个 PL/SQL 块，这样的文件类似于操作系统中的批处理文件。运行批处理命令文件时，可以一次执行多个 SQL 或 SQL * Plus 命令及 PL/SQL 块，为用户带来方便。可以用系统编辑程序 EDIT 来建立批处理命令文件。应当注意的是，每一个 SQL 命令后都要用(;)或独自一行的(/)来结束，而每一个 PL/SQL 块都要用独自一行的(/)来结束。

§2.3 获得帮助信息

在 SQL * Plus 环境中，可以获得以下几类系统帮助：

2.3.1 SQL * Plus 命令帮助

(1)在 SQL * Plus 提示符下输入 HELP，即：

```
SQL>HELP
```

SQL * Plus 将列出所有的 SQL * Plus 和 SQL 命令。

(2)SQL>HELP command—name

将列出命令名为 command—name 的帮助信息，如 HELP SELECT 将显示有关 SELECT 的帮助信息。

(3)SQL>> HELP TOPICS

将列出当前的所有帮助话题。

2.3.2 列出一个表的列定义

利用 DESCRIBE 命令可以显示一个指定的表的每一个列的定义(包括列名称、数据类型等)。这无疑会方便用户查询和编程。

例如，列出表 DEPT 的列定义：

```
SQL>DESCRIBE DEPT
```

运行结果如下所示：

Name	Null?	Type
DEPTNO	NOT NULL	NUMBER(2)
DNAME		CHAR(14)
LOC		CHAR(13)

同样，DESCRIBE 还可以提示 PL/SQL 的函数、包、过程的定义。

2.3.3 解释出错信息

当 SQL * Plus 检测到命令中的错误时，会返回一个错误信息。用户可以通过查看有关手册进一步了解出错的原因及修正的方法。

(1)对于 SQL * Plus COPY 命令，如果错误是一个数字编码的错误，可以参考 SQL * Plus USER'S GUIDE AND REFERENCE(V3.1)的附录 A；

(2)如果错误是以"ORA"为开头的数字编码错误，参考 ORACLE7 serve Messages and Codes

(3)如果错误是非数字编码错误，参考 SQL LANGUAGE REFERENCE MANUAL 或 PL/SQL USER'S GUIDE。

第三章　数据库查询

从数据库中检索信息是数据库操作的重要内容之一。SQL 语言提供的 SELECT 命令，可以使用户通过多种方式来灵活地从一个或多个表和视图中检索所需要的数据。一个完整的 SELECT 命令由若干个子句构成，即

```
SELECT ……
FROM ……
[WHERE …
GROUP BY …
HAVING …
ORDER BY …];
```

其中，SELECT 和 FROM 子句是不可缺省的，而方括号中的子句可以根据用户的需要灵活地选择。值得注意的是，SELECT 命令是一个 SQL 命令，因此应该用第 2 章中介绍的 SQL 命令的输入和运行方法来输入、编辑和运行它。在下面的各节中将详细介绍各子句的用法。

本章中，被查询的表均以 EMP 表为例，它包含了某单位职员的详细情况。EMP 表的内容见第 1 章。

§3.1　普通查询

3.1.1　SELECT 子句和 FROM 子句

在 SELECT 命令中，SELECT 子句用于指定检索数据库的哪些列，FROM 子句用于指定从哪一个表或视图中检索数据。

(1)选择所有的列

例 3.1 查看 EMP 表中所有的列

```
SQL> SELECT *
  2  FROM EMP;
```

其中，* 表示要查询所有的列。查询结果如下：

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	8[illegible]0		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1[illegible]0	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1[illegible]0	500	30
7566	JONES	MANAGER	7839	02-APR-81	29[illegible]5		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1[illegible]0	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	28[illegible]0		30
7782	CLARK	MANAGER	7839	09-JUN-81	24[illegible]0		10
7788	SCOTT	ANALYST	7566	09-DEC-82	30[illegible]0		20

7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

(2)选择指定的列

通过 SELECT 子句可以告诉 SQL * Plus，在检索时只查询表中某一个或若干个特定的列。

例 3.2 查看表 EMP 中的所有职工姓名(ENAME)

```
SQL>SELECT ENAME
  2  FROM EMP;
```

其中，ENAME 指明了查询时所要选择的列名。查询结果如下：

```
ENAME
SMITH
ALLEN
WARD
JONES
MARTIN
BLAKE
CLARK
SCOTT
KING
TURNER
ADAMS
JAMES
FORD
MILLER
```

例 3.3 查看 EMP 表中的职工代码(EMPNO)、职工姓名(ENAME)和工作名称(JOB)。

```
SQL> SELECT EMPNO,ENAME,JOB
  2  FROM EMP;
```

其中，SELECT 子句中的各个列名之间要以逗号分隔。

查询结果如下：

EMPNO	ENAME	JOB
7369	SMITH	CLERK
7499	ALLEN	SALESMAN
7521	WARD	SALESMAN
7566	JONES	MANAGER
7654	MARTIN	SALESMAN