第一章 SQL*Plus

这一章，我们以 SQL*Plus3.1 版本为例来介绍 SQL 语言以及用于生成报表的 SQL*Plus 命令。

第一节 SQL*Plus 概述

SQL*Plus 是 ORACLE 数据库软件的主要产品，是 ORACLE 系统提供的一种用户接口。用户可以在 SQL*Plus 环境下使用 SQL 命令交互式地访问数据库，也可以使用 SQL*Plus 命令生成报表。

一、启动和退出 SQL*Plus

1. 启动 SQL*Plus

如果SQL*Plus是运行在操作系统环境下，则在操作系统提示符后面输入如下命令即可：

SQLPLUS (回车)

这时，系统将显示其版本号、日期和版权信息，并提示输入用户名和口令字(口令字不在屏幕上显示)，如果连续三次输入错误，则拒绝进入 SQL*Plus 环境。也可以使用下面简单的方法启动 SQL*Plus：

SQLPLUS 用户名/口令字(回车)

如果 SQL*Plus 是 Windows 环境下的，则在程序管理器中找到并打开 ORACLE 程序组图标，在此组中找到并打开 SQL*Plus3.1 图标，在系统显示其版本号、日期和版权信息后，会弹出连接对话框，让用户输入用户名和口令字，同样，如果连续三次输入错误，则拒绝进入 SQL*Plus 环境。

如果提供的用户名和口令字是合法的，则进入 SQL*Plus 环境，系统显示 SQL*Plus 命令提示符：

SQL>

这表明 SQL*Plus 已经准备好，可以输入命令了。

2. 退出 SQL*Plus

如果想退出 SQL*Plus 环境，则在 SQL*Plus 提示符后面输入命令 EXIT 即可：

SQL>EXIT(回车)

这样，就可以返回到操作系统(如果 SQL*Plus 运行在操作系统环境下)或返回到 ORACLE 程序组了(如果 SQL*Plus 运行在 Windows 环境下)。

二、SQL*Plus 环境下，可以执行的命令

1. SQL 命令

利用 SQL 语言命令可以操纵数据库中的数据。例如，可以利用 SELECT 语句查询 emp 表中 20 号部门职工的编号、姓名和工资，命令如下：

```
SQL> SELECT empno, ename, sal(回车)
  2  FROM emp(回车)
  3  WHERE empno=20;(回车)
```

说明：

- 该命令共有 3 行，第 1 行、第 2 行输入回车后，出现下一行的行号，提示用户输入下一行命令，第 3 行中的分号(;)表示命令结束，按回车键后， SQL*Plus 将处理该命令，并把查询结果显示在屏幕上。之后，再次出现 SQL 命令提示符以接受新的命令。
- SQL 命令可以输入在一行上，也可以象上面所示输入在多行上；
- 结束 SQL 命令的方法有三种，即可以用一个分号(;)，可以在单独一行上输入一个反斜杠(/)，也可以用一个空行来结束命令。但这三种方法是有区别的，前两种方法首先把 SQL 命令存入 SQL 缓冲区，然后 SQL*Plus 处理缓冲区中的命令；而后一种方法只是把命令存入缓冲区，但并不执行。要想执行缓冲区中的命令，可以在 SQL 命令提示符后输入反斜杠(/)，或输入命令 RUN，按回车键后 SQL*Plus 就对该命令进行处理。
- 如果用分号结束命令，则在分号的同一行上不能输入注释(/* */)。

2. PL/SQL 块

除了 SQL 命令外，还可以使用 PL/SQL 块来操纵数据库中的数据。有两种方法可以进入 PL/SQL 方式：

(1) DECLARE 或 BEGIN 方式

在 SQL*Plus 命令提示符下输入 DECLARE 或 BEGIN 命令，就可以进入 PL/SQL 方式，在这种方式下，我们可以输入 PL/SQL 块的剩余部分，所建立的是一个无名的 PL/SQL 块。

(2) CREATE 方式

在 SQL*Plus 命令提示符下输入 CREATE 命令，进入 PL/SQL 方式，在这种方式下我们可以建立存储子程序。如：

用 CREATE FUNCTION 可以建立 PL/SQL 函数；

用 CREATE PACKAGE 可以建立包说明；

用 CREATE PACKAGE BODY 可以建立包体；

用 CREATE PROCEDURE 可以建立 PL/SQL 过程；

用 CREATE TRIGGER 可以建立触发器等等。

结束 PL/SQL 方式的方法是：在单独一行上输入一个句点。

SQL*Plus 把在 SQL 命令提示符下输入的 PL/SQL 块或子程序存入缓冲区中，当用户在 SQL 命令提示符下发出 RUN 命令或反斜杠(/)后，就可以执行当前子程序。有关 PL/SQL 子程序的信息可以参见本书第二章的内容。

3. SQL*Plus 命令

可以利用 SQL*Plus 命令来操作 SQL 命令和 PL/SQL 块，格式化和打印查询结果。有关 SQL*Plus 的命令及其使用可以参见本章第八节的内容。这里我们通过一个简单的例子来说明 SQL*Plus 命令怎样格式化查询结果。

首先在命令行中输入 SQL*Plus 命令：

```
SQL> COLUMN sal HEADING salary FORMAT $99,999.99
```

输入并运行查询命令：

```
SQL> SELECT empno,ename,sal FROM emp WHERE deptno=20;
```

结果如下：

```
EMPNO    ENAME        SALARY
------ -------- ------------
7369     SMITH         $800.00
7566     JONES       $2,975.00
7788     SCOTT       $3,000.00
7876     ADAMS       $1,100.00
7902     FORD        $3,000.00
```

从上面的结果可以看出，COLUMN 命令已经将 SAL 列格式化，给它一个新的标题 SALARY，并且在数据中使用美元符号($)，逗号(,)和句点(.)。

注意：SQL*Plus 只将 SQL 命令存入 SQL 缓冲区中，并不把 SQL*Plus 命令存入缓冲区。

如果一个 SQL*Plus 命令比较长，可以输入在连续的多行上。为了换行，可以在一行的末尾输入一个连字符(-)，然后按回车键，SQL*Plus 就显示一个大于号作为续行的提示符。

如果要结束一个 SQL*Plus 命令，只需按回车键即可。当然也可以在命令的末尾输入一

个分号。

4. 操作系统命令

在 SQL*Plus 命令提示符下也可以执行主机的操作系统命令。要想运行一个主机操作系统命令，可以在 SQL*Plus 的 HOST 命令后面跟一个主机操作系统命令，例如：

```
SQL> HOST DIR *.SQL
```

当该主机操作系统命令运行结束后，将再次显示 SQL*Plus 命令提示符。

5. 运行 ORACLE*Forms 的格式文件

如果在 SQL*Plus 安装时使用了 RUNFORM 选项，就可以在 SQL*Plus 的命令提示符下运行 ORACLE*Forms 的格式文件，例如，要运行一个文件名为 ORDER 的格式文件，可以发出如下命令：

```
SQL> RUNFORM ORDER
```

6. 求助命令

使用求助命令可以帮助我们获得所需要的信息。例如：

```
SQL> HELP(回车)   /* 这条命令执行的结果将显示所有 SQL 和 SQL*Plus 命令 */
SQL> HELP SELECT(回车) /* 这条命令将显示指定命令的信息 */
SQL> DESC EMP(回车)    /* 显示基表 EMP 中各列的定义 */
SQL> DESC proc_name(回车)  /* 显示该过程的定义 */
SQL> SHOW ALL(回车)    /* 显示当前 SQL*Plus 环境参数的设置情况 */
SQL> SHOW PAUSE(回车)  /* 显示参数 PAUSE 的值 */
```

三、SQL*Plus 的限制

这里给出了使用 SQL*Plus 时的各种限制，见表 1-1 。这些值对大多数操作系统都有效，除了 PDP11 和 MC68000 。

表 1-1 SQL*Plus 的限制

文件名长度	与系统有关
用户名长度	30 字节
用户变量名长度	30 字节
用户变量值长度	1024 字节
SQL INSERT 命令的 INTO 表中的变量数	50
每个 SQL 命令的变量数	100
命令行长度	2500 字符
通过 SQL*Plus 输入的 LONG 值的长度	LINESIZE 值

续表 1-1

LINESIZE 值	与系统有关
LONGCHUNKSIZE 值(要求 ORACLE7)	MAXDATA 值
MAXDATA 值	与系统有关
输出行大小	与系统有关
变量替代后的行大小	1000 字符
每个 SQL 命令的行数	500(假定每行 80 字符)
每页行数	50000
总行宽	32767 字符,VMS 为 60000 字符
一次取数组的行数	5000
嵌套的命令文件数	5,VMS 、 CMS 和 UNIX 为 20
页数	99,999
PL/SQL 错误消息缓冲区	2K(ORACLE7),512(ORACLE6)

四、SQL 缓冲区及 SQL 命令的编辑

1. SQL 缓冲区

当输入 SQL 命令或 PL/SQL 块时， SQL*Plus 将其保存在内部缓冲区中，这个内部缓冲区称为 SQL 缓冲区。 SQL 缓冲区中只能存放一条 SQL 命令,即当输入一条新的 SQL 命令后，SQL*Plus 将清除原来保存的 SQL 命令，而把新的 SQL 命令存入缓冲区中。 SQL 缓冲区中不能保存 SQL*Plus 命令。我们可以编辑和运行存放在 SQL 缓冲区中的命令而不用重新输入该命令。

2. SQL 命令的编辑

对于存放在 SQL 缓冲区中的 SQL 命令或 PL/SQL 块，可以用主机操作系统提供的编辑程序进行编辑，也可以使用 SQL*Plus 命令进行编辑。 SQL*Plus 的编辑命令见表 1-2 。

表 1-2　SQL*Plus 的编辑命令

命令	缩写	用途
APPEND text	A text	将 text 追加到当前行末尾
CHANGE /old/new/	C /old/new/	将当前行中的字符串 old 修改为 new
CHANGE /text	C /text	从当前行中删除 text
CLEAR BUFFER	CL BUFF	删除全部行，即把缓冲区清空
DEL	(无)	删除当前行
INPUT	I	在当前行下面增加一行或若干行

续表 1-2

INPUT text	I text	在当前行下面增加由 text 组成的一行
LIST	L	列出 SQL 缓冲区中的所有行
LIST n	L n 或 n	列出指定行
LIST *	L *	列出当前行
LIST LAST	L LAST	列出最后一行
LIST m n	L m n	列出从 m 到 n 的所有行

例如，输入命令：

```
SQL> SELECT empno,ename,sal
  2  FROM emp;
```

(1) 列出缓冲区中的命令

```
SQL> L(回车) /* 列出所有行 */
  1  SELECT empno,ename,sal
  2* FROM emp        /* 这里的星号(*)标识当前行 */

SQL> L1(回车) /* 列出第一行 */
  1* SELECT empno,ename,sal
```

(2) 编辑当前行

① 字符串替代

假如要将第一行中的 ename 修改为 job，首先用 LIST 命令使第一行成为当前行，然后发出如下命令即可：

```
SQL> C /ename/job/(回车)
  1* SELECT empno,job,sal
```

② 将正文加到行尾

假如还要列出 comm 列，首先使第一行成为当前行，然后发出如下命令即可：

```
SQL> A ,comm(回车)
  1* SELECT empno,job,sal,comm
```

(3) 增加新行

如果要查看 20 号部门的职工情况，就需要再增加一个 WHERE 子句加以限定，首先用 LIST 命令列出缓冲区中的内容，使第二行成为当前行，然后再发出如下命令即可：

```
SQL> INPUT(回车) /* 在下一行上出现一行号，提示用户输入 */
  3i WHERE deptno=20(回车)
  4i      /* 可以继续输入正文，也可以直接按回车键退出插入操作 */
```

这时再列出缓冲区中的内容如下：

```
SQL> LIST(回车)
  1  SELECT empno, job, sal, comm
  2  FROM emp
  3* WHERE deptno=20
```

(4) 删除行

如果要删除缓冲区中的某一行，首先使该行成为当前行，然后用 DEL 命令即可删除之。如要删除缓冲区中的第三行，可以发出下面的命令：

```
SQL> L 3(回车)
  3* WHERE deptno=20
SQL> DEL(回车)
```

这时原来的第三行就被删除，它的下一行将成为当前行(如果有的话)。

如果要删除缓冲区中的所有行，可以使用 CLEAR BUFFER 命令。如：

```
SQL> CLEAR BUFFER(回车)
```

这条命令执行的结果是清除 SQL 缓冲区中的全部内容。

(5) 系统编辑程序

在主机操作系统中，一般都提供一个或几个正文编辑程序供用户使用。用户可以通过 SQL*Plus 命令 EDIT 进入主机操作系统的缺省正文编辑程序而不必退出 SQL*Plus，如：

```
SQL> EDIT(回车)
```

如果用户不想使用系统缺省的编辑程序，而是使用某一个指定的编辑程序的话，可以用 SQL*Plus 的 DEFINE 命令，定义参数_EDITOR 的值为该编辑程序的名字。例如：

```
SQL> DEFINE _EDITOR = PE3(回车)
```

这样，当用户使用 EDIT 命令时，就进入 PE3 编辑程序。

五、本章使用的基表和视图

这里给出本章例子中所使用的基表 DEPT 和 EMP 。

1. DEPT 表

DEPTNNO	DNNAME	LOC
10	ACCOUNNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

2. 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	1,600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1,250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2,975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1,250	1,400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2,850		30
7782	CLARK	MANAGER	7839	09-JUN-82	2,450		10
7788	SCOTT	ANALYST	7566	09-NOV-81	3,000		20
7839	KING	PRESIDENT		17-NOV-81	5,000		10
7844	TURNER	SALESMANN	7698	08-SEP-81	1,500	0	30
7876	ADAMS	CLERK	7788	23-SEP-81	1,100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORDS	ANALYST	7566	03-DEC-81	3,000		20
7934	MILLER	CLERK	7782	23-JAN-82	1,300		10

六、语法格式定义中所用符号说明

所有关键字用大写表示。

…|…：表示竖线两边的内容可以选择出现，即“或”的关系。

[…]：表示方括号中的内容可以出现 0 次或 1 次。

{…}：表示花括号中的内容必须出现 1 次。

{…}*：表示花括号中的内容可以出现 0 次或多次。

第二节 SQL 语言概述

SQL 语言是一组命令的集合。在 ORACLE 数据库中，所有的程序和用户都必须使用这些命令来存取数据。应用程序和 ORACLE 的工具包不允许用户直接使用 SQL 命令来访问 ORACLE 数据库，但是这些应用包在执行用户的要求时必须使用 SQL 命令。

SQL 语言最初是七十年代中期由 IBM 公司在关系数据库管理系统 System R 上开发的语言，一经问世就受到广泛重视与欢迎，现在已经成为工业上标准的关系数据库存取语言，并定义在 ANSI X3.125-1989 “具有完整性增强特征的数据库语言 SQL ” 文档中。

一、SQL 语言的特点

SQL 语言之所以受欢迎并得到广泛应用，主要源于以下几个特点：

1. SQL 是一种非过程化程度高的语言

用户只需在程序中指出要作什么就可以了，而不必告诉系统怎么去作。

2. 用户性能好

SQL 语言对所有的用户都提供了一致的、可用的、易学的命令。基本的 SQL 命令，仅用几个小时就可以掌握，即使复杂的高级命令也能够在几天内掌握。

3. SQL 语言是一种公用语言

所有类型的用户都可以使用 SQL 语言进行任何类型的数据库活动。这些用户可以是：系统管理员，数据库管理员，安全管理员，应用程序员，决策支持系统人员以及其它类型的最终用户。

4. SQL 语言是所有关系数据库的共用语言

所有的主要关系数据库管理系统都支持 SQL 语言，用 SQL 语言编写的程序具有可移植性，即仅需很少的修改就可以把它从一个数据库移植到另一个数据库。

5. SQL 语言具有很强的功能

SQL 语言是一种关系数据语言。它允许用户仅与高层次的数据结构打交道，即 SQL 命令操纵的可以是行的集合而不仅仅是一行记录。 SQL 命令接收集合或行作为输入，同样返回相应的集合作为输出。

6. 提供了“视图”数据结构

视图是由数据库中满足一定条件的数据组成的虚表。用户可以使用 SQL 语言对视图进行操作，由系统把对视图的操作转换为对相应基表的操作，这样既便于用户使用，又提高了数据的独立性和安全性。

7. SQL 语言有两种使用方式

用户可以通过两种方式使用 SQL 语言。一种是在 SQL*Plus 环境下，采用命令方式交互式地使用 SQL 语言，即用户每发出一条命令，系统就执行它并显示执行结果；另一种是嵌

入高级程序设计语言(如 PASCAL 、 C 、 ADA 、 FORTRAN 等)中，组成一个完整的程序。用户可以根据自己的需要选择相应的方式来使用 SQL 语言。

8. SQL 语言提供了数据控制功能

SQL 语言提供了事务控制功能，以保证数据的共享及并发使用，有利于数据库的恢复；SQL 语言还提供了授权控制功能，以保证数据的安全与保密。

二、SQL 命令中使用的运算符、函数、表达式和条件

1. 运算符

SQL 语言中使用的运算符有算术运算符、字符运算符、比较运算符、逻辑运算符等，见表 1-3 。

表 1-3 SQL 运算符

运算符		优先级
一元的+ -运算符	PRIOR 运算符	最高
二元的* /运算符		
二元的+ -运算符	||字符运算符	
全部的比较运算符		
NOT 逻辑运算符		
AND 逻辑运算符		
OR 逻辑运算符		最低

表 1-4 至表 1-9 详细列出了各种 SQL 运算符的符号及其作用，并给出了使用例子。

(1) 算术运算符

表 1-4 算术运算符

运算符	作用	使用举例
一元+ -	表示正负表达式	SELECT * FROM emp WHERE -saL<0
二元* /	进行乘和除运算	UPDATE emp SET saL=sal*1.1
二元+ -	进行加和减运算	SELECT empno, sal+comm FROM emp

说明：不能用两个连续的负号(-)表示双否定或负值的减法。因为在 SQL 语句中“--”表示注释的开始。如果要表示双否定或负值的减法，则必须用空格或括号把二个连续的负号分开。

(2) 字符运算符

表 1-5 字符运算符

运算符	作用	使用举例
||	字符串连接	SELECT 'No.' ||empno,sal FROM emp

(3) 比较运算符

可以用比较运算符把一个表达式和另一个表达式进行比较，结果可能是 TRUE 、 FALSE 或 NULL(未知的、不可用的)。

表 1-6 比较运算符

运算符	作用	使用举例
=	相等	SELECT * FROM emp WHERE deptno=20
!= ^= <>	不等	SELECT * FROM emp WHERE job!='SALESMAN'
>	大于	SELECT * FROM emp WHERE sal>2000
<	小于	SELECT * FROM emp WHERE comm<1000
>=	大于或等于	SELECT * FROM emp WHERE sal>=1500
<=	小于或等于	SELECT * FROM emp WHERE sal<=2000
IN =ANY	等于任一元素则结果为 TRUE	SELECT * FROM emp WHERE deptno IN (20,30)
NOT IN !=ALL	等于任一元素则结果为 FALSE	SELECT * FROM emp WHERE deptno NOT IN (20,30)
ANY SOME	把一个值与表中的每一个值进行比较，有一个关系成立，则结果为 TRUE	SELECT * FROM emp WHERE sal=ANY(SELECT saL FROM emp WHERE deptno=20)
ALL	把一个值和表中的所有值进行比较，当所有关系均成立时，结果为 TRUE	SELECT * FROM emp WHERE sal>ALL(SELECT * FROM emp WHERE deptno=20)
[NOT]BETWEEN num1 AND num2	(不)大于等于 num1，并且小于等于 num2	SELECT * FROM emp WHERE sal BETWEEN 1500 AND 3000
EXISTS	如果子查询至少返回一行，则结果为 TRUE	SELECT * FROM dept WHERE EXIST(SELECT * FROM emp WHERE dept.deptno=emp.deptno)
X [NOT]LIKE Y [ESCAPE Z]	如果 X(不)匹配 Y，则结果为 TRUE ESCAPE 指明 Z 为转义字符	1. SELECT * FROM emp WHERE job LIKE 'SALES%' 2. SELECT * FROM emp WHERE ename LIKE '%A_B%' ESCAPE '\'
IS [NOT] NULL	如果(不)为空，则结果为 TRUE	SELECT * FROM emp WHERE comm IS NULL

说明：

- 如果在 NOT IN 后面的表中有一项为空，则结果为空。
- 在模式匹配中，百分号(%)表示和具有 0 个或多个字符的任一字符串匹配，下划线(_)表示和任一个字符匹配。

(4) 逻辑运算符

表 1-7 逻辑运算符

运算符	作用	使用举例
NOT	逻辑非	SELECT * FROM emp WHERE NOT (comm IS NULL)
AND	逻辑与	SELECT * FROM emp WHERE job LIKE 'SALES%' AND comm>2000
OR	逻辑或	SELECT * FROM emp WHERE sal>2500 OR job='MANAGER'

说明：• NOT 真值表

NOT	TRUE	FALSE	NULL
	FALSE	TRUE	NULL

• AND 真值表

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

• OR 真值表

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

(5) 集合运算符

利用集合运算符，见表 1-8，可以把两个查询的结果合并为一个结果集合。

表 1-8 集合运算符

运算符	作用	举例
UNION	集合的并，不包括重复行	SELECT * FROM emp WHERE sal>2000 UNION SELECT * FROM emp WHERE comm>1000
UNION ALL	集合的并，包括重复行	SELECT * FROM emp WHERE deptno=10 UNION ALL SELECT * FROM emp WHERE job='MANAGER'
INTERSECT	集合的交，不包括重复行	SELECT * FROM emp WHERE sal>2500 INTERSECT SELECT * FROM emp WHERE comm>1000
MINUS	集合的差，不包括重复行	SELECT * FROM emp WHERE sal>2000 MINUS SELECT * FROM emp WHERE job='CLERK'

(6) 其它运算符

表 1-9　其它运算符

运算符	作用	举例
(+)	表示在连接中优先列是外部连接列	SELECT ename, emp.deptno, loc FROM emp, dept WHERE dept.deptno=emp.deptno(+)
PRIOR	在分层或树型结构的查询中，为当前的父行计算PRIOR后的表达式	SELECT empno, ename, mgr FROM emp CONNECT BY PRIOR empno=mgr

2. 函数

SQL 函数一般分为单行函数和聚组函数，单行函数为被查询的每一行返回一个结果，而聚组函数为被查询行的每一组返回一个结果。单行函数可以使用在SELECT表、WHERE子句、START WITH 子句和 CONNECT BY 子句中(注意：该 SELECT 语句中不能有 GROUP BY 子句)；聚组函数可以出现在 SELECT 表和 HAVING 子句中。 SQL 函数可以分为如下几类：

单行函数 包括：
- 数字函数
- 字符函数 包括：
  - 返回字符值的函数
  - 返回数字值的函数
- 日期函数
- 转换函数
- 其它函数

聚组函数

(1) 数字函数

数字函数的自变量是数字型的，并且返回结果也为数字型的，见表 1-10 。

表 1-10　数字函数

函数	功能	使用举例
ABS(n)	返回 n 的绝对值	SELECT ABS(comm) FROM emp
CEIL(n)	返回大于或等于 n 的最小整数	SELECT CEIL(sal) FROM emp
COS(n)	返回 n(弧度表示的角)的余弦值	SELECT COS(9.42) FROM DUAL
COSH(n)	返回数 n 的双曲余弦值	SELECT COSH(0) FROM DUAL
EXP(n)	返回自然对数底 e 的 n 次幂值	SELECT EXP(3) FROM DUAL
FLOOR(n)	返回等于或小于数 n 的最大整数	SELECT FLOOR(521.4) FROM DUAL
LN(n)	返回数 n(n>0)的自然对数	SELECT LN(100) FROM DUAL
LOG(m,n)	返回数 n 的以 m 为底的对数	SELECT LOG(2,6) FROM DUAL
MOD(m,n)	返回 m 除以 n 所得的余数	SELECT MOD(7,2) FROM DUAL
POWER(m,n)	返回 m 的 n 次幂	SELECT POWER(3,2) FROM DUAL

续表 1-10

ROUND(m[,n])	返回 m 舍入到 n 位结果	SELECT ROUND(4.123,2) FROM DUAL SELECT ROUND(1234.56,-2) FROM DUAL
SIGN(n)	返回数 n 的符号(若 n=0,返回 0)	SELECT SIGN(-12) FROM DUAL
SIN(n)	返回弧度数 n 的正弦值	SELECT SIN(3.14) FROM DUAL
SINH(n)	返回数 n 的双曲正弦值	SELECT SINH(2) FROM DUAL
SQRT(n)	返回数 n(n>=0)的平方根	SELECT SQRT(9) FROM DUAL
TAN(n)	返回弧度数 n 的正切值	SELECT TAN(45.2) FROM DUAL
TANH(n)	返回数 n 的双曲正切值	SELECT TANH(4) FROM DUAL
TRUNC(m[,n])	返回将数 m 截断到 n 位	SELECT TRUNC(1234.56,1) FROM DUAL SELECT TRUNC(1234.56,-2) FROM DUAL

(2) 字符函数

字符函数的自变量都是字符型的，但返回的结果可以为数字型的，见表 1-11，也可以是字符型的，见表 1-12。

表 1-11 返回数字值的字符函数

函数	功能	使用举例
ASCII(char)	返回字符 char 的核准代码	SELECT ASCII('A') FROM DUAL
INSTR(str1,str2[,pos[,n]])	从第 pos(缺省为 1)个字符开始查找 str2 在 str1 中第 n(缺省为 1)次出现的位置，如果搜索失败则返回 0。	SELECT INSTR('MADEINCHINA','IN') FROM DUAL
INSTRB(str1,str2[,pos[,n]])	从第 pos(缺省为 1)个字节位置开始查找 str2 在 str1 中第 n(缺省为 1)次出现的位置，如果 pos 为负，则从 str1 的右端开始搜索，如搜索失败返回 0。	SELECT INSTRB('数据库','据') FROM DUAL
LENGTH(str)	返回 str 的字符数，当 str 取 CHAR 值时，长度包括尾部的空格；如 str 为空值，则返回结果也为空值。	SELECT * FROM emp WHERE LENGTH(ename)=5
LENGTHB(str)	返回 str 的字节数，当 str 取 CHAR 值时，长度包括尾部的空格；如 str 为空值，则返回结果也为空值。对于单字节字符集，与前一函数等价。	SELECT * FROM emp WHERE LENGTHB(ename)=6

续表 1-11

NLSSORT(str[,nlsparms])	将 str 按 nlsparams 所指定的排序序列排序后返回	SELECT * FROM emp WHERE NLSSORT(ename,'NLS_SORT=German')> NLSSORT('H',NLS_SORT=German')

表 1-12　返回字符值的字符函数

函数	功能	使用举例
CHR(num)	返回核准代码 num 所表示的字符。	SELECT CHR(65) FROM DUAL
CONCAT(str1,str2)	返回将 str2 拼接在 str1 之后所形成的串。	SELECT CONCAT(ename,'-'),job FROM emp
INITCAP(str)	将 str 中每个字的第一个字母变为大写，其余字母均为小写。	SELECT INITCAP(ename) FROM emp
LOWER(str)	使 str 的所有字符均为小写，下同。	SELECT LOWER(ename) FROM emp
LPAD(str,len[,pad])	用 pad 中的字符序列对 str 从左边开始填充到长度 len 处，pad 缺省为单个空格，如果 str 的长度大于 len，则返回它的前 len 个字符。	SELECT LPAD(ename,20),job FROM emp
LTRIM(str[,set])	从 str 左端开始截去出现在 set 中的字符，直到遇到一个不属于 set 的字符为止，set 缺省为单个空格。	SELECT LTRIM(ename,'SM') FROM emp
NLS_INITCAP(str[,nlsparms])	将 str 中每个字的第一个字母大写，其余字母小写后返回。	SELECT NLS_INITCAP('ijsland','NLS_SORT =Xdutch') FROM DUAL
NLS_LOWER(str[,nlsparms])	将 str 的所有字母小写后返回。	SELECT NLS_LOWER('CITTA','NLS_SORT= XItalian') FROM DUAL
NLS_UPPER(str[,nlsparms])	将 str 的所有字母大写后返回。	SELECT NLS_UPPER('gro β', 'NLS_SORT=XGerman') FROM DUAL
REPLACE(str1,str2[,str3])	将 str1 中的每个子串 str2 用 str3 代替后返回。	SELECT REPLACE('JACK and JUE','J','BL') FROM DUAL

续表 1-12

RPAD(str, len[, pad])	用 pad 中的字符序列对 str 从右边开始填充到长度 len 处， pad 缺省为单个空格，如果 str 的长度大于 len ，则返回它的前 len 个字符。	SELECT RPAD(ename, 30, '.'), job FROM emp
RTRIM(str[, set])	从 str 右端开始截去出现在 set 中的字符，直到遇到一个不属于 set 的字符为止， set 缺省为单个空格。	SELECT RTRIM('TURNERjiijj', 'ij') FROM DUAL
SOUNDEX(str)	返回包含 str 的语音表示的字符串。	SELECT ename FROM emp WHERE SOUNDEX(ename)= SOUNDEX('SMYTHE')
SUBSTR(str, pos[, len])	返回 str 中从字符位置 pos 开始，长度为 len 的子串，若 pos 为负，则反向计数。	SELECT SUBSTR(dname, 1, 4) FROM dept
SUBSTRB(str, pos[, len])	返回 str 中从字节位置 pos 开始，长度为 len 的子串，若 pos 为负，则反向计数。	SELECT SUBSTRB('数据库原理', 7, 4) FROM DUAL
TRANSLATE(str, set1, set2)	将 str 中的属于 set1 的字符用 set2 中相对应的字符替代，其余的字符不受影响，如果 set1 包含的字符比 set2 多，则这些多余的字符在 set2 中无对应的字符，因此该函数将从 str 中删除这样的字符。	SELECT TRANSLATE('SMITH', 'HJIKLMN', 'ABCD') FROM DUAL 结果为: SCTA
UPPER(str)	将 str 中所有字母变成大写后返回。	SELECT UPPER('Large') FROM DUAL

(3) 日期函数

日期函数在DATE类型的值上进行操作，返回结果也为DATE类型的，只有MONTH_BETWEEN返回数字型结果，见表 1-13 。

表 1-13 日期函数

函数	功能	使用举例
ADD_MONTHS(dte, num) ADD_MONTHS(num, dte)	返回日期 dte 加上或减去 num 月后所得日期。	SELECT empno, ADD_MONTHS(hiredate, 1) FROM emp

续表 1-13

LAST_DAY(dte)	返回 dte 所在的月的最后一天。	SELECT LAST_DAY(SYSDATE) FROM DUAL
MONTHS_BETWEEN(dte1, dte2)	返回日期 dte1 和 dte2 之间所差的月数。	SELECT empno, MONTHS_BETWEEN(SYSDATE, hiredate) FROM emp
NEW_TIME(dte, zon1, zon2)	返回与时区 zon1 的日期和时间 dte 相对应的时区 zon2 的日期和时间。	SELECT TO_CHAR(NEW_TIME(TO_DATE('17:47', 'hh24:mi'), 'PST', 'GMT'), 'hh24:mi') FROM DUAL
NEXT_DAY(dte, day)	返回由 day 所命名的周的第一天(day 在 dte 之后)。	SELECT NEXT_DAY('1-MAY-96', 'MONDAY') FROM DUAL 结果为：6-MAY-96
ROUND(dte[, fmt])	将 dte 舍入到由格式 fmt 所指定的单位后返回。	SELECT empno, ROUND(hiredate, 'YEAR') FROM emp
SYSDATE	返回当前的系统日期和时间。	SELECT SYSDATE FROM DUAL
TRUNC(dte[, fmt])	根据格式 fmt 所指定的单位对日期 dte 进行截断后返回。	SELECT empno, TRUNC(hiredate, 'YYYY') FROM emp

(4) 转换函数

转换函数把一个值从一种数据类型转换为另一种数据类型。SQL 的转换函数有以下几个，见表 1-14 。

表 1-14　转换函数

函数	功能	使用举例
CHARTOROWID(str)	将 str 从 CHAR 或 VARCHAR2 类型转换为 ROWID 类型。	SELECT ename, FROM emp WHERE ROWID= CHARTOROWID('0000000F.0003.0002')
ROWIDTOCHAR(bin)	将二进制值 bin 从 ROWID 类型转换为 VARCHAR2 类型的 18 字节的十六进制的字符串。	SELECT ROWID FROM emp WHERE ROWIDTOCHAR(ROWID) LIKE '%F21%'
CONVERT(str, set1 [, set2])	将字符集 set2 中的 str 转换为字符集 set1 的字符串。	SELECT CONVERT('Gro β', 'WE8HP', 'WE8DEC') FROM DUAL