第 3 章　关系数据库语言 SQL 与 PLSQL

SQL 是关系数据库提供的一种人机接口语言，SQL 的目标是实现对关系数据库灵活、方便地操纵。

本章主要介绍 SQL 的特点、基本组成，用 SQL 定义与创建各种数据库对象的命令及其用法，以及用 SQL 对关系数据库数据进行基本操作：插入、修改、删除数据以及查询数据的命令及其用法，在此基础上介绍 PLSQL 的内容，包括 PLSQL 块的结构、基本组成，以及用 PLSQL 编程的基本语句及编程方法。最后简要介绍 Oracle 数据库操作环境的安装、配置及操作方法。

3.1　SQL 概述

SQL 早期被称为 SEQUEL(Structured English QUEry Language，结构化查询语言)，因功能丰富，操作方便，容易学习和使用，1986 年被美国国家标准化研究所(ANSI)采纳为关系数据库管理系统的标准语言，后被国际标准化组织接纳为国际标准，1992 年推出 SQL 的更新版本 SQL-92 简称 SQL2，之后又引入了很多新特征被称为 SQL3。

SQL 是一种非过程化语言，主要由数据定义语言(Data Definition Language，DDL)、数据操纵语言(Data Manipulate Language，DML)和数据控制语言(Data Control Language，DCL)等组成。数据定义语言提供对各种数据库对象，如关系表、视图、索引、聚集、存储过程、存储函数、触发器等的定义及创建；数据操纵语言提供对关系表的基本操作(查询、修改、删除)；数据控制语言提供对数据库的控制，包括权限控制等。

3.1.1　SQL 的特点

SQL 是一种综合的、通用的、功能较强的关系数据库语言。SQL 有以下特点：

1. 非过程化

SQL 是非过程化的，被称为第四代计算机语言，以区别于面向过程的第三代语言。SQL 只需要用户描述“做什么”，而无须指出“怎么做”，语句操作的过程由系统自动完成，语言简洁，容易学习和使用。

2. 一体化

SQL 集数据定义、数据操作、数据控制为一体，用统一的语言操作数据、管理维护数据库实施及运行阶段的各种活动，包括建库、装载数据、查询数据、更新数据、维护数据库的安全性、数据的完整性、一致性和可靠性。

目前，很多商用关系数据库管理系统其 SQL 提供两种使用方式：一种是交互式命令语言方式，即用户输入一条数据操纵命令，系统返回一组运行结果；另一种是程序方式的 SQL，是对标准 SQL 的扩充。如 Oracle 的 PLSQL，SQL Server 的 T-SQL 都属于标准 SQL 的扩充版，即在标准 SQL 中增加了过程处理语句，使得 SQL 处理数据更加高效、灵活和方

便。有些大型商用数据库管理系统还支持嵌入式 SQL，即把基本的 SQL 语句嵌入到多种高级程序设计语言（如 C，C++，Java，Fortran 等）中，把 SQL 的数据操纵能力和高级语言的数据处理能力结合起来以支持更为复杂的数据库应用。用户可以根据不同的使用需求选择合适的运行方式，不论哪种使用方式，其 SQL 的语法结构是统一的，容易学习和使用。

总而言之，SQL 是关系数据库统一的界面语言，可用于支持各种用户的数据库活动，其中也包括数据库系统管理员的一些活动，例如，启动数据库、关闭数据库及系统的管理维护活动等。

3.1.2　SQL 的基本成分

本节将参照 SQL 标准，以 Oracle 数据库为实例，介绍 SQL 的基本成分。

1. 数据类型

SQL 和一般的计算机语言一样，也有自己的词法和句法。在关系模式中，所有的属性名必须定义数据类型。SQL 支持的数据类型有：

（1）数值型（NUMBER）。

可以存储整数数字值或实数值（最大存储 38 位数字）。

可用下面的语法定义数值型数据类型：

```
NUMBER(p,s)
```

其中，p 表示所有数字的精度（范围为 1～38），s 定义小数点后面的位数。

例如，

```
NUMBER                          //存储一个浮点数
NUMBER(2)                       //存储两位整数
NUMBER(7,2)                     //表示将存储 5 位整数,两位小数
```

Oracle 根据不同的精度及小数位数的定义存储数据的示范，如表 3.1 所示。

表 3.1　不同精度及小数位数对存储数据的影响

输 入 数 据	数 据 类 型	存储的数据
7 456 123.89	NUMBER	7 456 123.89
7 456 123.89	NUMBER(9,1)	7 456 123.9（保留 1 位小数）
7 456 123.89	NUMBER(9)	7 456 124（取整，小数四舍五入）
7 456 123.89	NUMBER(9,2)	7 456 123.89（保留两位小数）
7 456 123.89	NUMBER(6)	******（不接收超长数据）
0.0012	NUMBER(2,4)	.0012

（2）字符（Character）数据类型。

字符数据类型以串的形式存储字符数据，串的字节值与字符编码模式（字符集）相对应。数据库的字符集是在创建数据库的时候建立的，通常不会改变。Oracle 支持单字节和多字节字符编码模式，单字节字符集如 7 位编码的 ASCII 码，多字节编码模式包括支持汉字的标准代码 CGB2312—80 及国际通用编码 Unicode 等。

Oracle 支持的字符数据类型有：

① CHAR。

定长字符型。所谓定长是指当输入的字符数目比指定的长度小时，Oracle 用空格填充补齐。如果输入的字符数目大于指定字符位数，Oracle 会发出相应的错误信息拒绝输入。对于 CHAR 类型的列必须指定列的长度(字节数，范围为 1～2000)，CHAR 的默认长度为 1，即存放一个字符。通常一个汉字占两个字节。

② VARCHAR2。

变长字符型。对于变长列必须指定一个最大的长度(字节数，范围为 1～4000)，变长表列占用的存储空间依赖于实际输入的字符数。

③ LONG。

最大可存储 2GB 的变长文本数据。

(3) LOB(Large Object Block)数据类型。

大对象数据类型有 BLOB、CLOB、BCLOB、BFILE，最大存储数据长度为 4GB，可用来存储大量的无结构的数据块，如文本、图像、声音、视频等多媒体信息。

① BLOB。

BLOB 数据类型在数据库中存储无结构的二进制数据，存储长度可达 4GB。

② CLOB 和 NCLOB。

CLOB 和 NCLOB 数据类型在数据库中可存储多达 4GB 的字符数据，CLOB 存储单字节字符集数据，NCLOB 存储多字节字符集数据。

③ BFILE。

BFILE 数据类型在数据库外的操作系统文件中存储无结构的二进制数据，存储长度可达 4GB。BFILE 类型的表列中仅存储文件定位符，由目录名和文件名组成文件定位指针映射到文件系统。

(4) 日期(DATE)数据类型。

日期型用于存储日期和时间信息。Oracle 的日期数据按 7 字节定长字段，使用其内部格式存储，对应于：世纪、年、月、日、小时、分钟和秒。

Oracle 默认的日期格式是“DD-MON-YYYY”，如 05-MAR-2007 表示 2007 年 3 月 5 日，如果通过中文环境输入日期，其值应输入为：05-3 月-2007。如果想改变日期的默认格式，可以用 TO_CHAR 或 TO_DATA 函数。例如，以非默认格式输入日期数据：

```
SQL> Insert into mytable (hiredate) values (to_date('28/11/1998 8:11:32',
'DD/MM/YYYY HH24:MI:SS'))
已创建 1 行;
```

以非默认格式显示(查询)日期结果：

```
SQL> select to_char(hiredate,'day Month year HH24:MI:SS') from mytable;

TO_CHAR(HIREDATE,'DAYMONTHYEARHH24:MI:SS')
------------------------------------------------------------------
星期六 11 月 nineteen ninety-eight 08:11:32
```

2. 数据类型转换

与大部分程序设计语言一样，SQL 也不支持不同类型数据的运算。例如，$a \times b + 5$，如

果这个表达式能够正确求值的话，两个变量 a 和 b 的数据类型都必须是数值型的。SQL 的语法规定，在插入数据的时候对于数值型的数据不需要用引号把值括起来，而字符型的数据要用引号把值括起来。例如：

语句 1：

```
Insert into course (credit, quota) values (3,90);
```

语句 2：

```
Insert into course (credit, quota) values ('3','90');
```

这两条语句都能够完成数据的插入，但语句 2 在插入数据的时候，系统先要把字符“3”和字符“90”转换成数字后再插入。显然，语句 2 也正确执行了，但会增加系统的开销。

Oracle 提供了一些类型转换函数，如表 3.2 所示，需要时可以调用。

表 3.2　Oracle 的数据类型转换函数

FROM \ TO	CHAR	NUMBER	DATE	RAW	ROWID
CHAR		TO_NUMBER	TO_DATE	HEXTORAW	CHARTOROWID
NUMBER	TO_CHAR		TO_DATE		
DATE	TO_CHAR				
RAW	RAWTOHEX				
ROWID	ROWIDTOCHAR				

3. 空值的用法

空值是一个不是零，也不是空格的特殊值，代表列值未知，很多数据库管理系统支持空值(Null)的存储和处理。Oracle 系统将一个长度为零的字符值作为空值，这意味着，如果一个表列的值为 Null，则它不占据存储空间。Oracle 系统按照下列规则处理空值：

(1) 任何包含空值的算术表达式，其结果为 Null。例如，

```
sal+Null=Null                    //不管 sal 的值是什么结果都为 Null
```

(2) 用 NVL 函数可以处理空值。

有时，为了得到希望的运算结果，可调用 NVL 函数处理表列中的空值，例如，按年收入升序列出教师的姓名和年收入，可输入下面的 SQL 语句：

```
SELECT tname, (sal+ nvl(bonus,0)) * 12  年收入
FROM teacher  ORDER BY 年收入;
```

语句中的 NVL 函数用于判断 bonus 的值，如果不空则 bonus 的取值与 sal 相加，否则使用参数 0 值与 sal 相加。

(3) Null 值的排序。

具有 Null 值的表列在按照升序排列时，Null 值被排在最后面；降序排列时，Null 值被排列在最前面。

(4) 用 IS NULL 运算符判断表列中的空值。

例如，查询没有奖金的教师信息，列出其姓名和职称，输入下面的语句：

```
SELECT tname, title FROM teacher
```

```
WHERE bonus is NULL;
```

4. 注释

在一个程序中使用注释可以增加语句的可读性和程序的可维护性。在 SQL 语句中可以使用单行注释和跨行注释。

(1) 单行注释。

用两个连字符--,后面跟着注释内容表示单行注释。例如:

```
SELECT tname 教师名,sal 工资        --教师名是 tname 列的别名,工资是 sal 列的别名
FROM teacher
WHERE DEPTNO=50 and sal>2800
ORDER BY 2 DESC
--ORDER BY 子句中的 2 表示按照 SELECT 列表中定义的第 2 个列名 sal 的值排序
```

(2) 跨行注释。

SQL 使用一对符号:/ * 和 * /表示跨(多)行注释。如果程序中需要注释的内容在一行内写不下,可以用符号/ * 和 * /把需要注释的内容括起来。

例如:

```
/ * This is a program ……
…… * /
```

5. 运算符

在 SQL 语句中允许使用的运算符有:

算术运算符:+,-,*,/

逻辑运算符:AND,OR,NOT

比较符:=,!=,>,<,>=,<=

集合运算符:[NOT] IN,ANY,ALL

判断列值是否在指定的范围内:[NOT] BETWEEN…AND…

模糊匹配符:[NOT] LIKE

测试空值:IS [NOT] NULL

字符串拼接符:‖

6. SQL 的语句不区分大小写,但插入的值区分大小写

例如:

```
INSERT into student (sno,sname,sex)
    values (991100, 'BLAKE', '男');
```

3.1.3 实例

本章讨论的大部分实例基于下面的关系模式:

```
Dep(dno,dname, tel,director)                                    --系关系模式
Student(sno, sname, sex, birth, dno)                            --学生关系模式
Course(cno, cname, credit)                                      --课程关系模式
Sc(sno, cno, grade)                                             --成绩关系模式
Teacher(tno, tname, title, hiredate, sal, bonus, mgr, deptno)   --教师关系模式
```

```
Teaching(tno, cno, evaluate)                                    --授课关系模式
```

3.2　数据定义语言

数据定义语言(Data Definition Language,DDL)用于定义和建立各种数据库对象。在 Oracle 数据库中,数据定义语言可用来定义下列的数据库对象:关系表(table)、视图(view)、索引(index)、聚集(cluster)、存储过程(procedure)、存储函数(function)、触发器(trigger)、数据库链路(database link)、对象表(object table)等。

3.2.1　关系表的创建与维护

表是关系数据库中最基本的对象,用数据定义语句定义和创建。每张表有一个名字,通常称为表名或关系名。Oracle 数据库限定表名必须以字母开头,最多不超过 30 个字符。一张表可以由一列或多列组成,列名以字母开头,最多不超过 30 个字符。

1. 建表语句(CREATE TABLE)

建表语句的语法为:

```
CREATE    TABLE  表名
(  [关系完整性约束,],
   列名 1  数据类型  [列完整性约束],
   列名 2  数据类型  [列完整性约束],
   ⋮  )
   [存储子句]
   ⋮
```

例如,建立关系表 dep(系)的语句为:

```
CREATE  TABLE  dep
(dno number(2),
 dname varchar2(30),
 tel  char(8),
 director number(4))                          --存储系主任的工作证号
```

建立关系表 student 的语句为:

```
CREATE  TABLE  student
(sno number(6) constraint PK_sno PRIMARY KEY,       --sno 列是主码,约束名为 PK_sno
sname char(6),
sex char(2) CONSTRAINT CHECK_sex
CHECK(sex IN ('男', '女')),                  --限制 sex 的值只能取值"男",或"女"
birth date  default sysdate,                 --birth 列的默认值为系统当前日期
dno number(2) constraint fK_dno REFERENCES DEP(DNO)   --dno 列上定义了引用完整性约束
```

注:语句中的 sysdate 是一个日期函数,它返回系统现在时刻的日期值。当输入一条记录时,如果此列没有提供具体值,系统就使用默认值。

上面建立关系表 student 的语句中实现了下列完整性约束:

(1) sno 是主码。

(2) sex 属性的取值只能为“男”或“女”。

(3) 出生年月的默认值为 sysdate(系统当前日期)。

(4) 外来码 dno 的值必须在被引用的原关系对应的列值中存在。

2. 修改表的定义(ALTER TABLE)

修改表的语句格式为:

```
ALTER TABLE 表名
ADD     (列说明)  增加新的列定义
MODIFY  (列说明)  修改表中已有列的定义
DROP    (列说明)  删除表中指定列的定义
```

例 3.1 在 dep 表中增加新列 dmgr,使用语句:

```
ALTER  TABLE  dep
  ADD(dmgr  CHAR(8));
```

例 3.2 修改 dep 表的结构,将 dname 列的最大长度扩展到 40 个字符,使用语句:

```
ALTER TABLE dep
MODIFY(dname  VARCHAR2(40));
```

例 3.3 从 dep 表中删除列 dmgr,使用语句:

```
ALTER TABLE dep
DROP (dmgr);
```

例 3.4 在关系表 dep 上增加主码约束,使用语句:

```
ALTER TABLE dep
ADD CONSTRAINT pk_dno PRIMARY KEY (dno);
```

3. 删除表对象(DROP TABLE)

当一些表不再需要时,可以将它们删除。

删除表的语句格式为:

```
DROP  TABLE  表名
```

例如:

```
DROP TABLE dep;
```

该语句将 dep 表(包括它里面的数据)从数据库中删除。

3.2.2 视图的定义与维护

视图是用一个查询定义的、由基表导出的表。视图像一个窗口,通过这个窗口用户可以查看或操作由视图定义的指定表中的信息。

1. 建立视图的语法

```
CREATE VIEW 视图名 [((列名 1,列名 2,…)]
AS  子查询
```

例如，建立 50 系学生信息的视图，使用下面的命令：

```
CREATE VIEW  student50
  AS  SELECT * FROM student
  WHERE  dno=50
  ORDER BY birth;                          --视图中的数据按出生日期升序排列
```

2. 删除视图的语法

```
DROP VIEW 视图名
```

例如，删除名为 student1 的视图，使用命令：

```
DROP VIEW student1;
```

3.3　数据更新

3.3.1　INSERT 语句

插入数据操作 INSERT 语句可将一条记录或查询结果插入指定的关系(表)中。

语句格式为：

```
INSERT  INTO 表名[(列名 1,列名 2,…)]
  VALUES (表达式 1,表达式 2,…);
```

INSERT 语句在插入一条记录到数据库中时，将表达式的值作为对应属性(表列)的值插入。插入的值除了数值型的数据以外，包括日期型在内的其他数据要用单引号括起来，空值用 null 表示，下面举例说明插入数据语句的用法。

例 3.5　插入一条完整的学生数据到学生表中，语句如下：

```
INSERT INTO student
  VALUES(980001, '张冬梅', '女', '15-7月-1987', 10);
```

如果在插入语句的 VALUES 子句中没有提供表全部列的列值，那么在表名的后面要说明插入语句将包含哪些列名，例如：

```
INSERT INTO student (sno, sname, sex, birth)
VALUES(990001, '王小朋', '男', '01-12月-1988');
```

INSERT 语句中的日期值要用单引号括起来，在中文操作环境下，日期值用格式“01-12 月-1988”，西文环境用格式“01-DEC-1988”为日期列提供值。

若暂时不输入出生日期，可以用下面的语句插入数据，例如：

```
INSERT INTO student
VALUES(980001, '张冬梅', '女', null, 10);
```

由于插入数据时，学生张冬梅的出生日期尚不清楚需要确认，可先用关系模式定义的默认值 sysdate(系统当前日期)插入此学生的出生日期，例如：

```
INSERT INTO student(sno,sname,sex,dno)
VALUES(980003, '张冬梅', '女', 10);
```

将一个查询结果插入指定表 teacher1 中，可以用下面的语句，例如：

```
Insert  into teacher1  Select * from teacher;        --teacher1 表必须存在
```

把 teacher 表中的数据，复制到 teacher2 表中，使用语句：

```
create table teacher2  as select * from teacher;      --系统会先建表,然后插入数据
```

3.3.2 UPDATE 语句

UPDATE 语句对表中已有的数据进行修改及更新操作。

语句格式为：

```
UPDATE 表名
SET 列名 1=表达式 1,列名 2=表达式 2,…
WHERE 条件;
```

其中，SET 子句定义需要修改值的列名和修改以后列的取值，WHERE 子句指定更新操作满足的条件，下面的例子说明 UPDATE 语句的用法：

例 3.6 将课程“数据结构”的学分改为 4 学分：

```
UPDATE course
SET credit=4
WHERE cname='数据结构';
```

如果语句中省略了 WHERE 子句，则修改指定表中全部的记录。

例 3.7 在原有工资基础上为全体教师涨 10％的工资：

```
UPDATE  teacher
SET sal=sal * 1.1;
```

这条语句将教师表中所有行 sal 列的值更新为“sal×1.1”。

3.3.3 DELETE 语句

语句格式为：

```
DELETE  FROM 表名 WHERE 条件;
```

DELETE 语句删除指定表中满足条件的记录。如果不指定条件，该语句将删除指定表中所有的记录，下面的例子说明 DELETE 语句的用法：

例 3.8 删除课程名为“化学试验”且学分为 1 学分的课程信息：

```
Delete from course
Where cname='化学试验' and credit=1;
```

例 3.9 删除课程表中的全部记录，即将该表清空，语句如下：

```
DELETE  FROM  course;
```

3.4　数据查询

SQL 对数据库的检索操作是通过 SELECT 语句实现的，本节讨论 SELECT 语句的功能及用法。

查询语句的一般格式为：

```
SELECT [DISTINCT|ALL] { * |[列表达式],[列表达式],…}
FROM {[表名|视图名], [表名|视图名], …}
[WHERE 条件]
[START  WITH 条件  CONNECT  BY 条件]
[GROUP BY 表达式[,表达式] …[HAVING 条件]
[UNION|UNION ALL |INTERSECT|MINUS]SELECT 命令
[ORDER BY{表达式|位置} [ASC|DESC] [, {表达式|位置[ASC|DESC]}]…]
```

3.4.1　SELECT 及其子句的用法

1. SELECT FROM 子句

这是最基本的查询表达式，即从指定的表中查找出全部的记录。

例 3.10　查询全部学生的信息：

```
SELECT * FROM  student;
```

这条语句查询学生表中所有的数据(行和列)。语句中的 * 表示查询结果将包含全部列，这条语句还使用了 SELECT 语句的默认选项(ALL 带下划线为默认项)，表示查询结果选择表中所有的行。

在查询语句中如果指定 DISTINCT 项，则可以过滤重复的记录，即相同的记录只输出一条，DISTINCT 项的用法如下：

例 3.11　查询现有教师职称的种类：

```
SELECT  DISTINCT  title  FROM  Teacher;
```

2. 选择操作——WHERE 子句

当从关系表中查询满足条件的记录时，可以用 WHERE 子句，WHERE 子句的用法如下：

例 3.12　查询 10 系男同学的信息：

```
SELECT * FROM  student
WHERE  sex='男'  AND dno=10;
```

例 3.13　列出所有男同学的学号、姓名和所在系号：

```
SELECT sno,sname,dno FROM student
WHERE sex='男';
```