第5章

PL/SQL 基础

本章要点

- SQL 和 PL/SQL
- 在 PL/SQL 中的数据定义语言
- 在 PL/SQL 中的数据操纵语言
- 在 PL/SQL 中的数据查询语言

本章介绍 PL/SQL 中的数据定义语言(DDL)、数据操纵语言(DML)和数据查询语言(DQL),由于数据库查询是数据库的核心操作,本章重点讨论使用 SELECT 查询语句对数据库进行各种查询的方法。

5.1 SQL 和 PL/SQL

SQL(Structured Query Language,结构化查询语言)是目前主流的关系型数据库上执行数据操作、数据检索以及数据库维护所需要的标准语言,是用户与数据库之间进行交流的接口,许多关系型数据库管理系统都支持 SQL 语言,但不同的数据库管理系统之间的 SQL 语言不能完全通用,Oracle 数据库使用的 SQL 语言是 Procedural Language/SQL(简称 PL/SQL)。

5.1.1 SQL 语言

SQL 语言是应用于数据库的结构化查询语言,是一种非过程性语言,本身不能脱离数据库而存在。一般高级语言存取数据库时要按照程序顺序处理许多动作,使用 SQL 语言只需简单的几行命令,由数据库系统来完成具体的内部操作。

1. SQL 语言分类

通常将 SQL 语言分为以下 4 类。

(1) 数据定义语言(Data Definition Language, DDL)。用于定义数据库对象,对数据库、数据库中的表、视图、索引等数据库对象进行建立和删除,DDL 包括 CREATE、ALTER、DROP 等语句。

(2) 数据操纵语言(Data Manipulation Language, DML)。用于对数据库中的数据进行插入、修改、删除等操作,DML 包括 INSERT、UPDATE、DELETE 等语句。

(3) 数据查询语言(Data Query Language, DQL)。用于对数据库中的数据进行查询操作,例如用 SELECT 语句进行查询操作。

(4) 数据控制语言(Data Control Language, DCL)。用于控制用户对数据库的操作权

限，DCL包括GRANT、REVOKE等语句。

2. SQL语言的特点

SQL语言具有高度非过程化、应用于数据库的语言、面向集合的操作方式、既是自含式语言又是嵌入式语言、综合统一、语言简洁和易学易用等特点。

(1) 高度非过程化。SQL语言是非过程化语言，进行数据操作，只要提出“做什么”，而无须指明“怎么做”，因此无须说明具体处理过程和存取路径，处理过程和存取路径由系统自动完成。

(2) 应用于数据库的语言。SQL语言本身不能独立于数据库而存在，它是应用于数据库和表的语言，使用SQL语言，应熟悉数据库中的表结构和样本数据。

(3) 面向集合的操作方式。SQL语言采用集合操作方式，不仅操作对象、查找结果可以是记录的集合，而且一次插入、删除、更新操作的对象也可以是记录的集合。

(4) 既是自含式语言、又是嵌入式语言。SQL语言作为自含式语言，它能够用于联机交互的使用方式，用户可以在终端键盘上直接键入SQL命令对数据库进行操作；作为嵌入式语言，SQL语句能够嵌入到高级语言(例如C、C++、Java)程序中，供程序员设计程序时使用。在两种不同的使用方式下，SQL语言的语法结构基本上是一致的，提供了极大的灵活性与方便性。

(5) 综合统一。SQL语言集数据查询(Data Query)、数据操纵(Data Manipulation)、数据定义(Data Definition)和数据控制(Data Control)功能于一体。

(6) 语言简洁，易学易用。SQL语言接近英语口语，易学使用，功能很强，由于设计巧妙，语言简洁，完成核心功能只用了9个动词，如表5.1所示。

表5.1 SQL语言的动词

SQL语言的功能	动　词	SQL语言的功能	动　词
数据定义	CREATE,ALTER,DROP	数据查询	SELECT
数据操纵	INSERT,UPDATE,DELETE	数据控制	GRANT,REVOKE

5.1.2 PL/SQL预备知识

本节介绍使用PL/SQL语言的预备知识：PL/SQL的语法约定，在SQL Developer中执行PL/SQL语句。

1. PL/SQL的语法约定

PL/SQL的语法约定如表5.2所示，在PL/SQL不区分大小写。

表5.2 PL/SQL的基本语法约定

语法约定	说　明
大写	PL/SQL关键字
|	分隔括号或大括号中的语法项，只能选择其中一项
[ ]	可选项。不要键入方括号

续表

语法约定	说　明
{ }	必选项。不要键入方括号
[,...n]	指示前面的项可以重复 n 次,各项由逗号分隔
[...n]	指示前面的项可以重复 n 次,各项由空格分隔
[;]	可选的 Transact-SQL 语句终止符。不要键入方括号
< label >	编写 PL/SQL 语句时设置的值
< *label* >(斜体,下画线)	语法块的名称。此约定用于对可在语句中的多个位置使用的过长语法段或语法单元进行分组和标记,可使用的语法块的每个位置由括在尖括号内的标签指示:< label >

2. 在 SQL Developer 中执行 PL/SQL 语句

在 SQL Developer 中执行 PL/SQL 语句的步骤如下。

(1) 选择“开始”→“所有程序”→Oracle-OraDB12Home1→“应用程序开发”→SQL Developer 命令,启动 SQL Developer 界面。

(2) 在主界面中展开 system_stsys 连接,单击工具栏的 按钮,主界面弹出 SQL 工作表窗口,在窗口中输入或粘贴要运行的 PL/SQL 语句,这里输入:

```
SELECT *
  FROM student;
```

(3) 选中所有语句并单击工具栏的 按钮或直接单击 按钮,即执行语句,在“结果”窗口显示 PL/SQL 语句执行结果,如图 5.1 所示。

图 5.1　在 SQL Developer 的 SQL 工作表窗口中输入 PL/SQL 语句

提示：在SQL工作表窗口中执行PL/SQL语句命令的方法有：

(1) 选中所有语句后单击工具栏的 按钮(“执行语句”按钮)或按F9键；

(2) 直接单击 按钮(“运行脚本”按钮)或按F5键。

5.2 在PL/SQL中的数据定义语言

本节介绍在SQL * Plus中使用PL/SQL语句创建数据库、表空间与表等内容。

5.2.1 数据库操作语句

使用PL/SQL中的DDL语言创建数据库的过程非常复杂，一般情况下应使用图形界面方式的数据库配置向导(Data Base Configuration Assistant，DBCA)创建数据库，不使用PL/SQL语句方式创建数据库。

1. 创建数据库

使用PL/SQL语句创建数据库步骤简介如下。

(1) 设定实例标识符。

建立数据库之前，必须先指定数据库实例的系统标识符，即SID，在SQL * Plus中使用以下命令设定SID：

```
SET ORACLE_SID = stdb
```

(2) 设定数据库管理员的验证方法。

创建Oracle数据库必须经过数据库的验证手续，且被赋予适当系统权限后才可以建立。可以使用密码文件或操作系统的验证方法，下面是密码文件验证方法。

```
orapwd file = D:\app\tao\oradata\DATABASE\PWDstdb.ora
Password = 123456 entries = 5
```

(3) 创建初始化参数。

创建新数据库之前必须新增或编辑的初始化参数如下。

- 全局数据库名称。
- 控制文件名称与路径。
- 数据块大小。
- 影响SGA容量的初始化参数。
- 设定处理程序最大数目。
- 设定空间撤销(Undo)管理方法。

(4) 启动SQL * Plus并以SYSDBA连接到Oracle实例。

```
sqlplus /nolog
connect system/Ora123456 as sysdba
```

(5) 启动实例。

在没有装载数据库情况下启动实例，通常只有在数据库创建期间或在数据库上实施维护操作时才会这么做，使用带有NOMOUNT选项的STARTUP命令。

```
STARTUP NOMOUNT pfile = "D:\app\tao\stdb\pfile\initstdb.ora"
```

(6) 使用 CREATE DATABASE 语句创建数据库。

在 Oracle 中创建数据库,使用 CREATE DATABASE 语句。

语法格式:

```
CREATE DATABASE <数据库名>
  {   USER SYS IDENTIFIED BY <密码>
      | USER SYSTEM IDENTIFIED BY <密码>
      | CONTROLFILE REUSE
      | MAXDATAFILES <最大数据文件数>
      | MAXINSTANCES <最大实例数>
      | {ARCHIVELOG | NO ARCHIVELOG}
      | CHARACTER SET <字符集>
      | NATIONAL CHARACTER SET <民族字符集>
      | SET DEFAULT
          { BIGFILE | SMALLFILE } TABLESPACE
      | [ LOGFILE [ GROUP <数字值> ] <文件选项>
      | MAXLOGFILES <数字值>
      | MAXLOGMEMBERS <数字值>
      | MAXLOGHISTORY <数字值>
      | FORCE LOGGING
      | DATAFILE <文件选项>
        [ AUTOEXTEND [ OFF | ON [ NEXT <数字值>[K | M | G | T ]
        MAXSIZE [ UNLIMITED | <数字值> [K | M | G | T ]]]]
      | DEFAULT TABLESPACE <表空间名> [DATAFILE <文件选项> ]
      | [ BIGFILE | SMALL] UNDO TABLESPACE <表空间名> [ DATAFILE <文件选项>]
      | SET TIME_ZONE = '<时区名>'
}... ;
```

其中:

```
<文件选项>::=
    ('<文件路径>\<文件名>') [ SIZE <数字值> [ K | M | G | T] [ REUSE ]],...n]
```

2. 修改数据库

修改数据库使用 ALTER DATABASE 语句。

语法格式:

```
ALTER DATABASE <数据库名>
  [ARCHIVELOG | NOARCHIVELOG]
  [NO] FORCE LOGGING
  RENAME FILE '<文件名>'[,...n] TO '<新文件名>'[,...n ]
  CREATE DATAFILE '<数据文件名>'
    [ AS {'<新数据文件名>'[ SIZE <数字值> [K | M | G | T ]] [ REUSE ]][,...n]} | NEW ]
  DATAFILE '<文件名>'{ONLINE | OFFLINE [ FOR DROP] | RESIZE <数字值> [ K | M | G | T]
    | END BACKUP | AUTOEXTEND {OFF | ON [NEXT <数字值> [K | M]]
      [MAXSIZE UMLIMITED | <数字值> [K | M ]] ] }}
```

```
  ADD LOGFILE '<文件名>'[ SIZE <数字值> [K | M | G | T ]] [ REUSE ]][,...n]
  DROP LOGFILE '<文件名>'
  ...;
```

3. 删除数据库

删除数据库使用 DROP 语句。

语法格式：

```
DROP DATABASE database_name
```

其中，database_name 是要删除的数据库名称。

5.2.2 表空间操作语句

下面介绍在 SQL * Plus 中使用 PL/SQL 中的 DDL 语言对表空间进行创建、管理和删除。

1. 创建表空间

创建表空间在 SQL * Plus 中使用 CREATE TABLESPACE 语句，创建的用户必须拥有 CREATE TABLESPACE 系统权限，在创建之前必须创建包含表空间的数据库。

语法格式：

```
CREATE TABLESPACE <表空间名>
  DATAFILE '<文件路径>\<文件名>'[SIZE <文件大小> [ K | M ]] [ REUSE ]
    [ AUTOEXTEND [ OFF | ON [ NEXT <磁盘空间大小> [ K | M ]]
    [ MAXSIZE [ UMLIMITED | <最大磁盘空间大小> [ K | M ] ] ] ]
    [ MINMUM EXTENT <数字值>[ K | M ] ]
    [ DEFAULT <存储参数>]
    [ ONLINE | OFFLINE ]
    [ LOGGING | NOLOGGING ]
    [ PERMANENT | TEMPORARY ]
    [ EXTENT MANAGEMENT [ DICTIONARY | LOCAL [ AUTOALLOCATE | UNIFORM [ SIZE <数字值>[ K | M ]
] ] ] ]
```

【例 5.1】 创建表空间 testspace，大小为 40MB，禁止自动扩展数据文件。

```
CREATE TABLESPACE testspace
  LOGGING
  DATAFILE 'I:\app\ora\oradata\orcl\testspace01.DBF' SIZE 40M
  REUSE AUTOEXTEND OFF;
```

该语句运行结果如图 5.2 所示。

【例 5.2】 创建表空间 newspace，允许自动扩展数据文件。

```
CREATE TABLESPACE newspace
  LOGGING
  DATAFILE 'I:\app\ora\oradata\orcl\ newspace01.DBF' SIZE 40M
  REUSE AUTOEXTEND ON NEXT 10M MAXSIZE 300M
  EXTENT MANAGEMENT LOCAL;
```

图 5.2 创建表空间 testspace

该语句运行结果如图 5.3 所示。

图 5.3 创建表空间 newspace

2. 管理表空间

在 SQL * Plus 中使用 ALTER TABLESPACE 命令可以修改表空间或它的一个或多个数据文件、或为数据库中每一个数据文件指定各自的存储扩展参数值。

语法格式:

```
ALTER TABLESPACE <表空间名>
  [ ADD DATAFILE | TEMPFILE '<路径>\<文件名>'[ SIZE <文件大小> [ K | M ] ]
  [ REUSE ]
      [ AUTOEXTEND [ OFF | ON [ NEXT <磁盘空间大小> [ K | M ] ] ] ]
```

```
[MAXSIZE [ UNLIMITED | <最大磁盘空间大小> [ K | M ] ] ]
[ RENAME DATAFILE '<路径>\<文件名>',...n TO '<路径>\<新文件名>'',...n ]
[ DEFAULT STORAGE <存储参数>]
[ ONLINE | OFFLINE [ NORMAL | TEMPORARY | IMMEDIATE ] ]
[ LOGGING | NOLOGGING ]
[ READ ONLY | WRITE ]
[ PERMANENT ]
[ TEMPORARY ]
```

【例 5.3】 通过 ALTER TABLESPACE 命令把一个新的数据文件添加到 newspace 表空间，并指定了 AUTOEXTEND ON 和 MAXSIZE 300M。

```
ALTER TABLESPACE newspace
  ADD DATAFILE 'I:\app\ora\oradata\orcl\DATA02.DBF' SIZE 40M
    REUSE AUTOEXTEND ON NEXT 50M MAXSIZE 300M;
```

3. 删除表空间

在 SQL * Plus 中使用 DROP TABLESPACE 语句删除已经创建的表空间。其语法格式如下。

语法格式：

```
DROP TABLESPACE <表空间名>
  [ INCLUDING CONTENTS [ {AND | KEEP} DATAFILES ]
    [ CASCADE CONSTRAINTS ]
  ] ;
```

【例 5.4】 删除表空间 newspace 和其对应的数据文件。

```
DROP TABLESPACE newspace
  INCLUDING CONTENTS AND DATAFILES;
```

该语句运行结果如图 5.4 所示。

图 5.4　删除表空间 newspace

5.2.3 表操作语句

在 SQL Developer 中使用 PL/SQL 中的 DDL 语言对表进行创建、管理和删除介绍如下。

1. 创建表

使用 CREATE TABLE 语句创建表。

语法格式:

```
CREATE TABLE [<用户方案名>.] <表名>
(
  <列名 1>  <数据类型>  [DEFAULT <默认值>]  [<列约束>]
  <列名 2>  <数据类型>  [DEFAULT <默认值>]  [<列约束>]
  [,...n]
  <表约束>[,...n]
)
  [PCTFREE <数字值>]
  [PCTUSED <数字值> ]
  [INITRANS <数字值>]
  [MAXTRANS <最大并发事务数>]
  [TABLESPACE <表空间名>]
  [STORGE <参数>]
  [AS <子查询>]
```

【例 5.5】 使用 PL/SQL 语句,在 stsys 数据库中创建 student 表。

在 stsys 数据库中创建 student 表语句如下:

```
CREATE TABLE student
(
  sno char(6) NOT NULL PRIMARY KEY,
  sname char(12) NOT NULL,
  ssex char(3) NOT NULL,
  sbirthday date NOT NULL,
  speciality char(18) NULL,
  sclass char(6) NULL,
  tc number NULL
);
```

启动 SQL Developer 界面,在主界面中展开 system_stsys 连接,单击工具栏的 按钮,主界面弹出 SQL 工作表窗口,在窗口中输入上述语句,单击 按钮,在“结果”窗口显示“CREATE TABLE 成功”,如图 5.5 所示。

提示: 由一条或多条 PL/SQL 语句组成一个程序,通常以.sql 为扩展名存储,称为 sql 脚本。SQL 工作表窗口内的 PL/SQL 语句,可用“文件”菜单“另存为”命令命名并存入指定目录。

2. 修改表

使用 ALTER TABLE 语句用来修改表的结构。

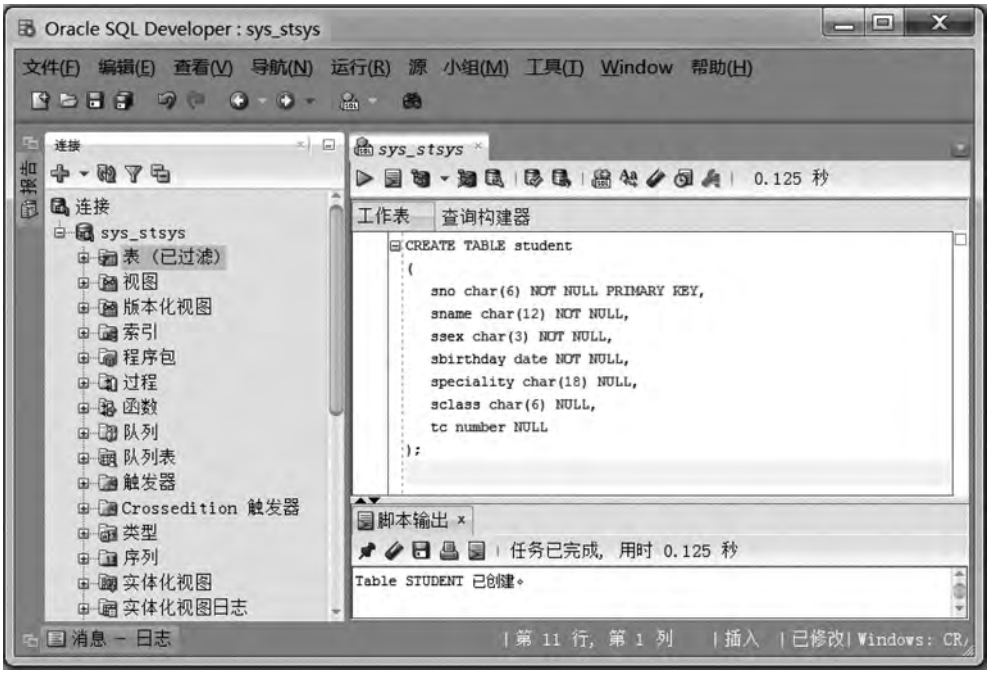

图 5.5 创建 student 表的表结构

语法格式：

```
ALTER TABLE [<用户方案名>.] <表名>
  [ ADD(<新列名> <数据类型> [DEFAULT <默认值>][列约束],...n) ]          /* 增加新列 */
  [ MODIFY([ <列名> [<数据类型>] [DEFAULT <默认值>][列约束],...n) ]  /* 修改已有列的属性 */
  [ STORAGE <存储参数> ]                                                /* 修改存储特征 */
  [<DROP 子句> ]                                                        /* 删除列或约束条件 */
```

其中，<DROP 子句>用于从表中删除列或约束。

语法格式：

```
<DROP 子句>::=
DROP
{
   COLUMN <列名>
    | PRIMARY [KEY]
    | UNIQUE (<列名>,...n)
    | CONSTRAINT <约束名>
    | [ CASCADE ]
}
```

【例 5.6】 使用 ALTER TABLE 语句修改 Stsys 数据库中的 student 表。

(1) 在 student 表中增加一列 remarks(备注)。

```
ALTER TABLE student
  ADD remarks varchar(100);
```

(2) 在 student 表中删除列 remarks。

```
ALTER TABLE student
  DROP COLUMN remarks;
```

3. 删除表

使用 DROP TABLE 语言删除表。

语法格式:

```
DROP TABLE table_name;
```

其中,table_name 是要删除的表的名称。

【例 5.7】 删除 Stsys 数据库中 newstudent 表(已创建)。

```
DROP TABLE newstudent;
```

5.3 在 PL/SQL 中的数据操纵语言

下面介绍在 SQL Developer 中,使用 PL/SQL 中的数据操纵语言 DML 对表进行插入记录、修改记录和删除记录。

5.3.1 插入语句

INSERT 语句用于向数据库的表插入一行,由 VALUES 给定该行各列的值。

语法格式:

```
INSERT INTO <表名>[(<列名 1>,<列名 2>,...n)]
  VALUES(<列值 1>,<列值 2>,...n)
```

其中,列值表必须与列名表一一对应,且数据类型相同。向表的所有列添加数据时,可以省略列名表,但列值表必须与列名表顺序和数据类型一致。

注意:使用 PL/SQL 语言进行插入、修改和删除后,为将数据的改变保存到数据库中,应使用 COMMIT 命令进行提交,使用方法如下:

```
COMMIT;
```

本书后面的 SQL 语句都省略 COMMIT 命令,运行时请读者添加。

【例 5.8】 向 student 表插入表 4.1 各行数据。

向 student 表插入表 4.1 各行数据的语句如下。

```
INSERT INTO student VALUES('181001','宋德成','男',TO_DATE('19971105','YYYYMMDD'),'计算机',
'201805',52);
INSERT INTO student VALUES('181002','何静','女',TO_DATE('19980427','YYYYMMDD'),'计算机','201805',
50);
INSERT INTO student VALUES('181004','刘文韬','男',TO_DATE('19980513','YYYYMMDD'),'计算机','201805',
52);
```

```
INSERT INTO student VALUES('184001','李浩宇','男',TO_DATE('19971024','YYYYMMDD'),'通信','201836',
50);
INSERT INTO student VALUES('184002','谢丽君','女',TO_DATE('19980116','YYYYMMDD'),'通信','201836',
48);
INSERT INTO student VALUES('184003','陈春玉','女',TO_DATE('19970809','YYYYMMDD'),'通信','201836',
52);
```

使用 SELECT 语句查询插入的数据：

```
SELECT *
  FROM student;
```

查询结果：

SNO	SNAME	SSEX	SBIRTHDAY	SPECIALITY	SCLASS	TC
181001	宋德成	男	1997-11-05	计算机	201805	52
181002	何静	女	1998-04-27	计算机	201805	50
181004	刘文韬	男	1998-05-13	计算机	201805	52
184001	李浩宇	男	1997-10-24	通信	201836	50
184002	谢丽君	女	1998-01-16	通信	201836	48
184003	陈春玉	女	1997-08-09	通信	201836	52

5.3.2 修改语句

UPDATE 语句用于修改表中指定记录的列值。

语法格式：

```
UPDATE <表名>
  SET <列名> = {<新值>|<表达式>} [,...n]
  [WHERE <条件表达式>]
```

其中，在满足 WHERE 子句条件的行中，将 SET 子句指定的各列的列值设置为 SET 指定的新值，如果省略 WHERE 子句，则更新所有行的指定列值。

注意： UPDATE 语句修改的是一行或多行中的列。

【例 5.9】 在 student 表中，将所有学生的学分增加 2 分。

```
UPDATE student
  SET tc = tc + 2;
```

5.3.3 删除语句

1. DELETE 语句

DELETE 语句用于删除表中的一行或多行记录。

语法格式：

```
DELETE FROM <表名>
  [WHERE <条件表达式>]
```

该语句的功能为从指定的表或中删除满足 WHERE 子句条件行,若省略 WHERE 子句,则删除所有行。

注意:DELETE 语句删除的是一行或多行。如果删除所有行,表结构仍然存在,即存在一个空表。

【例 5.10】 在 student 表中,删除学号为 184003 的行。

```
DELETE FROM student
  WHERE sno = '184003';
```

2. TRANCATE TABLE 语句

当需要删除一个表里的全部记录,使用 TRUNCATE TABLE 语句,它可以释放表的存储空间,但此操作不可回退。

语法格式:

```
TRUNCATE TABLE <表名>
```

5.4 在 PL/SQL 中的数据查询语言

PL/SQL 语言中最重要的部分是它的查询功能,PL/SQL 的 SELECT 语句具有灵活的使用方式和强大的功能,能够实现选择、投影和连接等操作。

语法格式:

```
SELECT <列>                                  /* SELECT 子句,指定列 */
  FROM  <表或视图>                           /* FROM 子句,指定表或视图 */
  [ WHERE  <条件表达式> ]                    /* WHERE 子句,指定行 */
  [ GROUP BY <分组表达式> ]                  /* GROUP BY 子句,指定分组表达式 */
  [ HAVING <分组条件表达式> ]                /* HAVING 子句,指定分组统计条件 */
  [ ORDER BY <排序表达式> [ ASC | DESC ]]    /* ORDER 子句,指定排序表达式和顺序 */
```

5.4.1 投影查询

投影查询用于选择列,投影查询通过 SELECT 语句的 SELECT 子句来表示。

语法格式:

```
SELECT [ ALL | DISTINCT ] <列名列表>
```

其中,<列名列表>指出了查询结果的形式,其格式为:

```
{   *                                        /* 选择当前表或视图的所有列 */
   |<表名>|<视图>|.*                         /* 选择指定的表或视图的所有列 */
   |{|<列名>|<表达式>}
       [[ AS ] <列别名>]                     /* 选择指定的列,为列指定别名 */
   | <列标题> =   <列名表达式>               /* 选择指定的列并更改列标题,为列指定别名 */
}[,...n ]
```

1. 投影指定的列

使用 SELECT 语句可选择表中的一个列或多个列，如果是多个列，各列名中间要用逗号分开。

语法格式：

```
SELECT <列名 1> [ , <列名 2> [,...n] ]
  FROM <表名>
  [WHERE <条件表达式>]
```

该语句的功能为在 FROM 子句指定表中检索符合条件的列。

【例 5.11】 查询 student 表中所有学生的学号、姓名和班号。

```
SELECT sno, sname, sclass
  FROM student;
```

查询结果：

```
SNO         SNAME        SCLASS
----------  -----------  ----------
181001      宋德成       201805
181002      何静         201805
181004      刘文韬       201805
184001      李浩宇       201836
184002      谢丽君       201836
184003      陈春玉       201836
```

2. 投影全部列

在 SELECT 子句指定列的位置上使用 * 号时，则为查询表中所有列。

【例 5.12】 查询 student 表中所有列。

```
SELECT *
  FROM student;
```

该语句与下面语句等价

```
SELECT sno, sname, ssex, sbirthday, speciality, sclass, tc
  FROM student;
```

查询结果：

```
SNO     SNAME   SSEX   SBIRTHDAY      SPECIALITY   SCLASS    TC
------  ------  -----  -------------  -----------  --------  -----
181001  宋德成  男     1997-11-05     计算机       201805     52
181002  何静    女     1998-04-27     计算机       201805     50
181004  刘文韬  男     1998-05-13     计算机       201805     52
184001  李浩宇  男     1997-10-24     通信         201836     50
184002  谢丽君  女     1998-01-16     通信         201836     48
184003  陈春玉  女     1997-08-09     通信         201836     52
```

3. 修改查询结果的列标题

为了改变查询结果中显示的列标题，可以在列名后使用 AS <列别名>。

【例 5.13】 查询 student 表中所有学生的学生的 sno、sname、speciality,并将结果中各列的标题分别修改为学号、姓名、专业。

```
SELECT sno AS 学号, sname AS 姓名, speciality AS 专业
  FROM student;
```

查询结果:

```
学号          姓名          专业
----------  ----------  ----------
181001        宋德成        计算机
181002        何静          计算机
181004        刘文韬        计算机
184001        李浩宇        通信
184002        谢丽君        通信
184003        陈春玉        通信
```

4. 计算列值

使用 SELECT 子句对列进行查询时,可以对数字类型的列进行计算,可以使用加(+)、减(-)、乘(*)、除(/)等算术运送符,SELECT 子句可使用表达式。

语法格式:

```
SELECT <表达式> [ , <表达式> ]
```

5. 去掉重复行

去掉结果集中的重复行可使用 DISTINCT 关键字。

语法格式:

```
SELECT DISTINCT <列名> [ , <列名>...]
```

【例 5.14】 查询 student 表中 sclass 列,消除结果中的重复行。

```
SELECT DISTINCT sclass
  FROM student;
```

查询结果:

```
SCLASS
----------
201836
201805
```

5.4.2 选择查询

选择查询用于选择行,选择查询通过 WHERE 子句实现,WHERE 子句通过条件表达式给出查询条件,该子句必须紧跟 FROM 子句之后。

语法格式:

```
WHERE <条件表达式>
```

其中，<条件表达式>为查询条件，格式为：

```
<条件表达式>::=
    { [ NOT ] <判定运算> | (<条件表达式> ) }
    [ { AND | OR } [ NOT ] { <判定运算> | (<条件表达式>) } ]
    [ ,...n ]
```

其中，<判定运算>的结果为 TRUE、FALSE 或 UNKNOWN，其格式为：

```
<判定运算>::=
{    <表达式 1> { = | < | <= | > | >= | <> | != } <表达式 2>          /* 比较运算 */
    | <字符串表达式 1> [ NOT ] LIKE <字符串表达式 2> [ ESCAPE '<转义字符>' ]
                                                                    /* 字符串模式匹配 */
    | <表达式> [ NOT ] BETWEEN <表达式 1> AND <表达式 2>             /* 指定范围 */
    | <表达式> IS [ NOT ] NULL                                      /* 是否空值判断 */
    | <表达式> [ NOT ] IN ( <子查询> | <表达式> [,...n] )           /* IN 子句 */
    | EXIST ( <子查询> )                                            /* EXIST 子查询 */
}
```

说明：

(1) 判定运算包括比较运算、模式匹配、指定范围、空值判断、子查询等，判定运算的结果为 TRUE、FALSE 或 UNKNOWN。

(2) 逻辑运算符包括 AND(与)、OR(或)、NOT(非)，NOT、AND 和 OR 的使用是有优先级的，三者之中，NOT 优先级最高，AND 次之，OR 优先级最低。

(3) 条件表达式可以使用多个判定运算通过逻辑运算符成复杂的查询条件。

(4) 字符串和日期必须用单引号括起来。

注意：在 SQL 中，返回逻辑值的运算符或关键字都称为谓词。

1. 表达式比较

比较运算符用于比较两个表达式值，共有 7 个运算符：=(等于)、<(小于)、<=(小于等于)、>(大于)、>=(大于等于)、<>(不等于)、!=(不等于)。

语法格式：

```
<表达式 1> { = | < | <= | > | >= | <> | != } <表达式 2>
```

【例 5.15】 查询 student 表中班号为 201805 或性别为女的学生。

```
SELECT *
  FROM student
  WHERE sclass = '201805' or ssex = '女';
```

查询结果：

SNO	SNAME	SSEX	SBIRTHDAY	SPECIALITY	SCLASS	TC
181001	宋德成	男	1997-11-05	计算机	201805	52
181002	何静	女	1998-04-27	计算机	201805	50
181004	刘文韬	男	1998-05-13	计算机	201805	52
184002	谢丽君	女	1998-01-16	通信	201836	48
184003	陈春玉	女	1997-08-09	通信	201836	52

2. 指定范围

BETWEEN、NOT BETWEEN、IN 是用于指定范围的三个关键字,用于查找字段值在(或不在)指定范围的行。

当要查询的条件是某个值的范围时,可以使用 BETWEEN 关键字。BETWEEN 关键字指出查询范围。

语法格式:

```
<表达式> [ NOT ] BETWEEN <表达式 1 > AND <表达式 2 >
```

【例 5.16】 查询 score 表成绩为 86、92、95 的记录。

```
SELECT *
  FROM score
  WHERE grade in (86,92,95);
```

查询结果:

```
SNO             CNO         GRADE
------------  ---------  -------------
181002          1004        86
184001          4002        92
184001          8001        86
184003          8001        95
181004          1201        92
```

【例 5.17】 查询 student 表中不在 1998 年出生的学生情况。

```
SELECT *
  FROM student
  WHERE sbirthday NOT BETWEEN TO_DATE('19980101','YYYYMMDD') AND
    TO_DATE('19981231','YYYYMMDD');
```

查询结果:

```
SNO        SNAME      SSEX      SBIRTHDAY      SPECIALITY     SCLASS    TC
-------  ---------  ------  -------------  -----------  -------  -------
181001     宋德成     男       1997-11-05     计算机         201805     52
184001     李浩宇     男       1997-10-24     通信           201836     50
184003     陈春玉     女       1997-08-09     通信           201836     52
```

3. 模式匹配

模式匹配使用 LIKE 谓词,LIKE 谓词用于指出一个字符串是否与指定的字符串相匹配,其运算对象可以是 char、varchar2 和 date 类型的数据,返回逻辑值 TRUE 或 FALSE。

语法格式:

```
<字符串表达式 1 > [ NOT ] LIKE <字符串表达式 2 > [ ESCAPE '<转义字符>' ]
```

在使用 LIKE 谓词时,<字符串表达式 2 >可以含有通配符,通配符有以下两种:

%:代表 0 或多个字符;

_：代表一个字符。

LIKE 匹配中使用通配符的查询也称模糊查询。

【例 5.18】 查询 student 表中姓谢的学生情况。

```
SELECT *
  FROM student
  WHERE sname LIKE '谢%';
```

查询结果：

```
SNO        SNAME        SSEX   SBIRTHDAY          SPECIALITY         SCLASS      TC
-------  ----------  -----  --------------  --------------  ----------  -------
184002     谢丽君       女     1998-01-16       通信              201836      48
```

4. 空值判断

判定一个表达式的值是否为空值时，使用 IS NULL 关键字。

语法格式：

```
<表达式> IS [ NOT ] NULL
```

【例 5.19】 查询已选课但未参加考试的学生情况。

```
SELECT *
  FROM score
  WHERE grade IS null;
```

查询结果：

```
SNO             CNO         GRADE
------------  ----------  ------------
184002          8001
```

5.4.3 分组查询和统计计算

查询数据常常需要进行统计计算，本节介绍使用聚合函数、GROUP BY 子句、HAVING 子句进行统计计算的方法。

1. 聚合函数

聚合函数实现数据的统计计算，用于计算表中的数据，返回单个计算结果。聚合函数包括 COUNT、SUM、AVG、MAX、MIN 等函数，下面分别介绍。

(1) COUNT 函数。

COUNT 函数用于计算组中满足条件的行数或总行数。

语法格式：

```
COUNT ( { [ ALL | DISTINCT ] <表达式> } | * )
```

其中，ALL 表示对所有值进行计算，ALL 为默认值，DISTINCT 指去掉重复值，COUNT 函数用于计算时忽略 NULL 值。

【例 5.20】 求学生的总人数。

```
SELECT COUNT( * ) AS 总人数
  FROM student;
```

该语句采用 COUNT(*)计算总行数,总人数与总行数一致。

查询结果:

```
总人数
----------
6
```

【例 5.21】 查询 201836 班学生的总人数。

```
SELECT COUNT( * ) AS 总人数
  FROM student
  WHERE sclass = '201836';
```

该语句采用 COUNT(*)计算总人数,并用 WHERE 子句指定的条件进行限定为 201836。

查询结果:

```
总人数
----------
3
```

(2) SUM 和 AVG 函数。

SUM 函数用于求出一组数据的总和,AVG 函数用于求出一组数据的平均值,这两个函数只能针对数值类型的数据。

语法格式:

```
SUM / AVG ( [ ALL | DISTINCT ] <表达式> )
```

其中,ALL 表示对所有值进行计算,ALL 为默认值; DISTINCT 指去掉重复值; SUM/AVG 函数用于计算时忽略 NULL 值。

【例 5.22】 查询 1201 课程总分。

```
SELECT SUM(grade) AS 课程 1201 总分
  FROM score
  WHERE cno = '1201';
```

该语句采用 SUM ()计算课程总分,并用 WHERE 子句指定的条件进行限定为 1201 课程。

查询结果:

```
课程 1201 总分
-------------
          509
```

(3) MAX 和 MIN 函数。

MAX 函数用于求出一组数据的最大值,MIN 函数用于求出一组数据的最小值,这两个函数都可以适用于任意类型数据。

语法格式:

```
MAX / MIN ( [ ALL | DISTINCT ] <表达式> )
```

其中,ALL 表示对所有值进行计算,ALL 为默认值;DISTINCT 指去掉重复值;MAX/MIN 函数用于计算时忽略 NULL 值。

【例 5.23】 查询 8001 课程的最高分、最低分、平均成绩。

```
SELECT MAX(grade) AS 课程 8001 最高分, MIN(grade) AS 课程 8001 最低分, AVG(grade) AS 课程 8001
平均成绩
  FROM score
  WHERE cno = '8001';
```

该语句采用 MAX 求最高分、MIN 求最低分、AVG 求平均成绩。

查询结果:

```
课程 8001 最高分      课程 8001 最低分   课程 8001 平均成绩
---------------  ---------------  ----------------
             95                85              88.8
```

2. GROUP BY 子句

GROUP BY 子句用于指定需要分组的列。

语法格式:

```
GROUP BY [ ALL ] <分组表达式> [,...n]
```

其中,分组表达式通常包含字段名,ALL 显示所有分组。

注意: 如果 SELECT 子句的列名表包含聚合函数,则该列名表只能包含聚合函数指定的列名和 GROUP BY 子句指定的列名。聚合函数常与 GROUP BY 子句一起使用。

【例 5.24】 查询各门课程的最高分、最低分、平均成绩。

```
SELECT cno AS 课程号, MAX(grade)AS 最高分,MIN (grade)AS 最低分, AVG(grade)AS 平均成绩
  FROM score
  WHERE NOT grade IS null
  GROUP BY cno;
```

该语句采用 MAX、MIN、AVG 等聚合函数,并用 GROUP BY 子句对 cno (课程号)进行分组。

查询结果:

```
课程号        最高分        最低分        平均成绩
--------  ----------  ----------  -------------
8001              95          85           88.8
```

```
4002            92          78        8.6E+01
1201            93          75        8.5E+01
1004            94          86              90
```

3. HAVING 子句

HAVING 子句用于对分组按指定条件进一步进行筛选,过滤出满足指定条件的分组。

语法格式:

```
[ HAVING <条件表达式>]
```

其中,条件表达式为筛选条件,可以使用聚合函数。

注意:HAVING 子句可以使用聚合函数,WHERE 子句不可以使用聚合函数。

当 WHERE 子句、GROUP BY 子句、HAVING 子句、ORDER BY 子句在一个 SELECT 语句中时,执行顺序如下:

(1) 执行 WHERE 子句,在表中选择行;

(2) 执行 GROUP BY 子句,对选取行进行分组;

(3) 执行聚合函数;

(4) 执行 HAVING 子句,筛选满足条件的分组;

(5) 执行 ORDER BY 子句,进行排序。

注意:HAVING 子句要放在 GROUP BY 子句的后面,ORDER BY 子句放在 HAVING 子句后面。

【例 5.25】 查询平均成绩在 90 分以上的学生的学号和平均成绩。

```
SELECT sno AS 学号, AVG(grade) AS 平均成绩
  FROM score
  GROUP BY sno
  HAVING AVG(grade)> 90;
```

该语句采用 COUNT 聚合函数、WHERE 子句、GROUP BY 子句、HAVING 子句;

查询结果:

```
学号                    平均成绩
----------  ----------------
181001                 9.3E+01
184003                 9.2E+01
```

【例 5.26】 查询至少有 5 名学生选修且以 8 开头的课程号和平均分数。

```
SELECT cno AS 课程号, AVG (grade) AS 平均分数
  FROM score
  WHERE cno LIKE '8%'
  GROUP BY cno
  HAVING COUNT( * )> 5;
```

该语句采用 AVG 聚合函数、WHERE 子句、GROUP BY 子句、HAVING 子句。

查询结果：

```
课程号              平均分数
------------  -------------
8001                   88.8
```

5.4.4 排序查询

在 Oracle 中，ORDER BY 子句用于对查询结果进行排序。

语法格式：

```
[ ORDER BY { <排序表达式> [ ASC | DESC ] } [ ,...n ]
```

其中，排序表达式，可以是列名、表达式或一个正整数，ASC 表示升序排列，它是系统默认排序方式，DESC 表示降序排列。

提示： 排序操作可对数值、日期、字符三种数据类型使用，ORDER BY 子句只能出现在整个 SELECT 语句的最后。

【例 5.27】 将 201836 班级的学生按出生时间降序排序。

```
SELECT *
  FROM student
  WHERE sclass = '201836'
  ORDER BY sbirthday DESC;
```

该语句采用 ORDER BY 子句进行排序。

查询结果：

```
SNO       SNAME     SSEX    SBIRTHDAY    SPECIALITY    SCLASS    TC
--------  --------  ------  -----------  ------------  --------  ------
184002    谢丽君    女      1998-01-16   通信          201836    48
184001    李浩宇    男      1997-10-24   通信          201836    50
184003    陈春玉    女      1997-08-09   通信          201836    52
```

5.5 小　　结

本章主要介绍了以下内容。

(1) SQL(Structured Query Language)语言是目前主流的关系型数据库上执行数据操作、数据检索以及数据库维护所需要的标准语言，是用户与数据库之间进行交流的接口，许多关系型数据库管理系统都支持 SQL 语言，但不同的数据库管理系统之间的 SQL 语言不能完全通用，Oracle 数据库使用的 SQL 语言是 Procedural Language/SQL(PL/SQL)。

(2) 通常将 SQL 语言分为以下 4 类：数据定义语言(Data Definition Language，DDL)、数据操纵语言(Data Manipulation Language，DML)、数据查询语言(Data Query

Language, DQL)、数据控制语言(Data Control Language, DCL)。

SQL 语言具有高度非过程化、应用于数据库的语言、面向集合的操作方式、既是自含式语言又是嵌入式语言、综合统一、语言简洁和易学易用等特点。

(3) 在 PL/SQL 中的数据定义语言 DDL。

DDL 中的数据库操作语句有:创建数据库用 CREATE DATABASE 语句、修改数据库用 ALTER DATABASE 语句、删除数据库用 DROP DATABASE 语句。

DDL 中的表空间操作语句有:创建表空间用 CREATE TABLESPACE 语句、修改表空间用 ALTER TABLESPACE 语句、删除表空间用 DROP TABLESPACE 语句。

DDL 中的表操作语句有:创建表用 CREATE TABLE 语句、修改表用 ALTER TABLE 语句、删除表用 DROP TABLE 语句。

(4) 在 PL/SQL 中的数据操纵语言 DML。

在表中插入记录用 INSERT 语句,在表中修改记录或列用 UPDATE 语句,在表中删除记录用 DELETE 语句。

(5) 在 PL/SQL 中的数据查询语言 DQL。

DQL 是 PL/SQL 语言的核心,DQL 使用 SELECT 语句,它包含 SELECT 子句、FROM 子句、WHERE 子句、GROUP BY 子句、HAVING 子句和 ORDER BY 子句等。

5.6 创建表实验

1. 实验目的及要求

(1) 理解数据库和表的基本概念。

(2) 掌握使用 PL/SQL 语句创建表的操作,具备编写和调试创建表、修改表、删除表的代码的能力。

2. 验证性实验

使用 PL/SQL 语句创建表、修改表、删除表。

在 stsys 数据库中分别创建一个 Employee 表(员工表)和 Department 表(部门表),其表结构分别如表 5.3 和表 5.4 所示。

表 5.3 Employee 表的表结构

列 名	数据类型	允许 null 值	是否主键	说 明
EmplID	varchar(4)		主键	员工号
EmplName	varchar(12)			姓名
Sex	varchar(3)			性别
Birthday	date			出生日期
Address	varchar(30)	√		地址
Wages	number			工资
DeptID	varchar(4)	√		部门号

表 5.4 Department 表的表结构

列 名	数据类型	允许 null 值	是否主键	说 明
DeptID	varchar(4)		主键	部门号
DeptName	varchar(30)			部门名称

(1) 创建 Employee 表。

```
CREATE TABLE Employee
(
  EmplID varchar2(4) NOT NULL PRIMARY KEY,
  EmplName varchar2(12) NOT NULL,
  Sex varchar2(3) NOT NULL,
  Birthday date NOT NULL,
  Address varchar2(30) NULL,
  Wages number NOT NULL,
  DeptID varchar2(4) NULL
);
```

(2) 创建 Department 表。

```
CREATE TABLE Department
(
  DeptID varchar2(4) NOT NULL PRIMARY KEY,
  DeptName varchar2(30) NOT NULL
);
```

(3) 将 Employee 表的 EmplName 字段的数据类型改为 char (12)。

```
ALTER TABLE Employee
  MODIFY (EmplName char(12));
```

(4) 将 Employee 表的 Address 字段删除。

```
ALTER TABLE Employee
  DROP COLUMN Address;
```

(5) 在 Employee 表中增加名为 Address 的字段,数据类型为 char(30)。

```
ALTER TABLE Employee
  ADD Address char(30);
```

(6) 删除 Employee 表。

```
DROP TABLE Employee;
```

3. 设计性试验

在 stsys 数据库中分别创建一个 StudentInfo 表(学生信息表)和 ScoreInfo 表(成绩信息表),表结构分别如表 5.5 和表 5.6 所示。

表 5.5 StudentInfo 表的表结构

列 名	数据类型	允许 null 值	是否主键	说 明
StudentID	varchar2(6)		主键	学号
Name	varchar2(12)			姓名
Sex	varchar2(3)			性别
Birthday	date			出生日期
Speciality	Varchar2(18)	√		专业
Address	varchar2(30)	√		家庭地址

表 5.6 ScoreInfo 表的表结构

列 名	数据类型	允许 null 值	是否主键	说 明
StudentID	varchar2 (6)		主键	学号
CourseID	varchar2(4)		主键	课程号
Grade	Number	√		成绩

编写和调试创建表、修改表、删除表的代码,完成以下操作。

(1) 创建 StudentInfo 表。

(2) 创建 ScoreInfo 表。

(3) 将 StudentInfo 表的 Name 字段的数据类型改为 char (12)。

(4) 将 StudentInfo 表的 Address 字段删除。

(5) 在 StudentInfo 表中增加名为 Phone 的字段,数据类型为 char(20)。

(6) 删除 StudentInfo 表。

4. 观察与思考

(1) 在创建表的语句中,NOT NULL 的作用是什么?

(2) 一个表可以设置几个主键?

(3) 主键列能否修改为 NULL?

5.7 表数据的插入、修改和删除实验

1. 实验目的及要求

(1) 掌握表数据的插入、修改和删除操作。

(2) 具备编写和调试插入数据、修改数据和删除数据的代码的能力。

2. 验证性实验

TeacherInfo 表(教师信息表)的表结构如表 5.7 所示,CourseInfo 表(课程信息表)的表结构如表 5.8 所示,在 stsys 数据库中,分别创建 TeacherInfo 表和 CourseInfo 表。

表 5.7 TeacherInfo 表的表结构

列 名	数据类型	允许 null 值	是否主键	说 明
TeacherID	varchar2(6)		主键	教师编号
TeacherName	varchar2(12)			姓名
TeacherSex	varchar2(3)			性别
TeacherBirthday	date			出生日期
School	varchar2(18)			学院
Address	varchar2(30)	√		地址

表 5.8 CourseInfo 表的表结构

列 名	数据类型	允许 null 值	是否主键	说 明
CourseID	varchar2 (4)		主键	课程号
CourseName	varchar2 (24)			课程名
Credit	number	√		学分

TeacherInfo 表的样本数据，如表 5.9 所示。

表 5.9 TeacherInfo 表的样本数据

教师编号	姓名	性别	出生日期	学 院	地 址
100005	李慧强	男	1968-09-25	计算机学院	北京市海淀区
100024	刘 松	男	1976-02-17	计算机学院	北京市海淀区
400021	陈霞飞	女	1975-12-07	通信学院	上海市黄浦区
800004	刘泉明	男	1978-08-16	数学学院	广州市越秀区
120007	张 莉	女	1982-03-21	外国语学院	成都市锦江区

CourseInfo 表的样本数据，如表 5.10 所示。

表 5.10 CourseInfo 表的样本数据

课程号	课 程 名	学分
1004	数据库系统	4
1025	物联网技术	3
4002	数字电路	3
8001	高等数学	4
1201	英语	4

按照下列要求完成表数据的插入、修改和删除操作。

(1) 向 TeacherInfo 表插入样本数据。

```
INSERT INTO TeacherInfo values('100005','李慧强','男',TO_DATE('19680925','YYYYMMDD'),'计算机学院','北京市海淀区');
```

```
INSERT INTO TeacherInfo values('100024','刘松','男',TO_DATE('19760217','YYYYMMDD'),'计算机学院','北京市海淀区');
INSERT INTO TeacherInfo values('400021','陈霞飞','女',TO_DATE('19751207','YYYYMMDD'),'通信学院','上海市黄浦区');
INSERT INTO TeacherInfo values('800004','刘泉明','男',TO_DATE('19780816','YYYYMMDD'),'数学学院','广州市越秀区');
INSERT INTO TeacherInfo values('120007','张莉','女',TO_DATE('19820321','YYYYMMDD'),'外国语学院','成都市锦江区');
```

(2) 向 CourseInfo 表插入样本数据。

```
INSERT INTO CourseInfo VALUES('1004','数据库系统',4);
INSERT INTO CourseInfo VALUES('1025','物联网技术',3);
INSERT INTO CourseInfo VALUES('4002','数字电路',3);
INSERT INTO CourseInfo VALUES('8001','高等数学',4);
INSERT INTO CourseInfo VALUES('1201','英语',4,);
```

(3) 更新教师编号为 120007 的记录,将出生日期改为“1983-09-17”。

```
UPDATE TeacherInfo
  SET TeacherBirthday = '1983 - 09 - 17'
  WHERE TeacherID = '120007';
```

(4) 将性别为“男”的记录的家庭住址都改为“上海市浦东新区”。

```
UPDATE TeacherInfo
  SET address = '上海市浦东新区'
  WHERE TeacherSex = '男';
```

(5) 删除教师编号为 400021 的记录。

```
DELETE FROM TeacherInfo
  WHERE TeacherID = '400021';
```

3. 设计性试验

Goods 表(商品表)的表结构如表 5.11 所示,在 stsys 数据库中,创建 Goods 表。

表 5.11 Goods 表的表结构

列 名	数据类型	允许 null 值	是否主键	说 明
GoodsID	varchar2(4)		主键	商品号
GoodsName	varchar2(30)			商品名称
Classification	varchar2(24)			商品类型
UnitPrice	number			单价
StockQuantity	number			库存量

Goods 表的样本数据如表 5.12 所示。

表 5.12　Goods 表的样本数据

商品号	商 品 名 称	商 品 类 型	单价	库存量
1001	Microsoft Surface Pro 4	笔记本电脑	5488	12
1002	Apple iPad Pro	平板电脑	5888	12
3001	DELL PowerEdgeT130	服务器	6699	10
4001	EPSON L565	打印机	1899	8

编写和调试表数据的插入、修改和删除的代码，完成以下操作。

(1) 向 Goods 表插入样本数据。

采用三种不同的方法，将 Goods 表的样本数据插入 Goods 表中。

① 省略列名表，插入记录。

```
INSERT INTO Goods VALUES('1001','Microsoft Surface Pro 4','笔记本电脑',5488,12);
```

② 不省略列名表，插入记录。

```
INSERT INTO Goods(GoodsID, GoodsName, Classification, UnitPrice, StockQuantity)
  VALUES('1002','Apple iPad Pro','平板电脑',5888,12);
```

③ 同时插入多条记录。

```
INSERT INTO Goods VALUES('3001','DELL PowerEdgeT130','服务器',6699,10);
INSERT INTO Goods VALUES('4001','EPSON L565','打印机',1899,8);
```

(2) 将 Microsoft Surface Pro 4 的类型改为“笔记本平板电脑二合一”。

(3) 将 EPSON L565 的库存量改为 10。

(4) 删除商品类型为平板电脑的记录。

4. 观察与思考

DROP 语句与 DELETE 语句有何区别？

5.8　查 询 实 验

1. 实验目的及要求

(1) 理解 SELECT 语句的语法格式。

(2) 掌握 SELECT 语句的操作和使用方法。

(3) 具备编写和调试 SELECT 语句以进行数据库查询的能力。

2. 验证性实验

对 stsys 数据库的员工表 Employee 表和部门表 Department 上进行信息查询。Employee 表的样本数据如表 5.13 所示，其中，员工号、姓名、性别、出生日期、地址、工资、部门号的列名分别为 EmplID、EmplName、Sex、Birthday、Address、Wages、DeptID。

表 5.13 Employee 表的样本数据

员工号	姓名	性别	出生日期	地　址	工资	部门号
E001	刘思远	男	1980-11-07	北京市海淀区	4100	D001
E002	何莉娟	女	1987-07-18	上海市浦东区	3300	D002
E003	杨　静	女	1984-02-25	上海市浦东区	3700	D003
E004	王贵成	男	1974-09-12	北京市海淀区	6800	D004
E005	孙　燕	女	1985-02-23	NULL	3600	D001
E006	周永杰	男	1979-10-28	成都市锦江区	4300	NULL

Department 表的样本数据如表 5.14 所示,其中,部门号、部门名称的列名分别为 DeptID、DeptName。

表 5.14 Department 表的样本数据

部门号	部门名称	部门号	部门名称
D001	销售部	D004	经理办
D002	人事部	D005	物资部
D003	财务部		

查询要求如下。

(1) 使用两种方式,查询 Employee 表的所有记录。

① 使用列名表。

```
SELECT EmplID, EmplName, Sex, Birthday, Address, Wages, DeptID
  FROM Employee;
```

② 使用 * 。

```
SELECT *
  FROM Employee;
```

(2) 查询 Employee 表有关员工号、姓名和地址的记录。

```
SELECT EmplID, EmplName, Address
  FROM Employee;
```

(3) 从 Department 表查询部门号、部门名称的记录。

```
SELECT DeptID, DeptName
  FROM Department;
```

(4) 通过两种方式查询 Goods 表中价格为 1500～4000 元的商品。

① 通过指定范围关键字。

```
SELECT *
  FROM Goods
```

```
  WHERE UnitPrice BETWEEN 1500 AND 4000;
```

② 通过比较运算符。

```
SELECT *
  FROM Goods
  WHERE UnitPrice >= 1500 AND UnitPrice <= 4000;
```

(5) 查询地址是北京的员工的姓名、出生日期和部门号。

```
SELECT EmplID, EmplName, DeptID
  FROM Employee
  WHERE address LIKE '北京%';
```

(6) 查询各个部门的员工人数。

```
SELECT DeptID AS 部门号, COUNT(EmplID) AS 员工人数
  FROM Employee
  GROUP BY DeptID;
```

(7) 查询每个部门的总工资和最高工资。

```
SELECT DeptID AS 部门号, SUM(Wages) AS 总工资, MAX(Wages) AS 最高工资
  FROM Employee
  GROUP BY DeptID;
```

(8) 查询员工工资,按照工资从高到低的顺序排列。

```
SELECT *
  FROM Employee
  ORDER BY Wages DESC;
```

3. 设计性试验

对 stsys 数据库的学生信息表 StudentInfo 和成绩信息表 ScoreInfo 上进行信息查询。StudentInfo 表的样本数据如表 5.15 所示,其中,学号、姓名、性别、出生日期、专业、地址的列名分别为 StudentID、Name、Sex、Birthday、Speciality、Address。

表 5.15　StudentInfo 表的样本数据

学号	姓名	性别	出生日期	专　业	家庭地址
181001	成志强	男	1998-08-17	计算机	北京市海淀区
181002	孙红梅	女	1997-11-23	计算机	成都市锦江区
181003	朱　丽	女	1998-02-19	计算机	北京市海淀区
184001	王智勇	男	1997-12-05	电子信息工程	NULL
184002	周潞潞	女	1998-02-24	电子信息工程	上海市浦东区
184004	郑永波	男	1997-09-19	电子信息工程	上海市浦东区

ScoreInfo 表的样本数据如表 5.16 所示,其中,学号、课程号、成绩的列名分别为 StudentID、CourseID、Grade。

表 5.16 ScoreInfo 表的样本数据

学号	课程号	成绩	学号	课程号	成绩
181001	1004	95	184001	8001	85
181002	1004	85	184002	8001	NULL
181003	1004	91	184004	8001	94
184001	4002	93	181001	1201	92
184002	4002	76	181002	1201	78
184004	4002	88	181003	1201	94
181001	8001	94	184001	1201	85
181002	8001	89	184002	1201	79
181003	8001	86	184004	1201	94

编写和调试查询语句的代码,完成以下操作。

(1) 使用两种方式,查询 StudentInfo 表的所有记录。

① 使用列名表。

② 使用 * 。

(2) 查询 ScoreInfo 表的所有记录。

(3) 查询高等数学成绩低于 90 分的成绩信息。

(4) 使用两种方式,查询地址为上海市浦东区和成都市锦江区学生的信息。

① 使用 IN 关键字。

② 使用 OR 关键字。

(5) 使用两种方式,查询分数为 90 到 95 分的成绩信息。

① 使用 BETWEEN AND 关键字查询。

② 使用 AND 关键字和比较运算符。

(6) 查询每个专业有多少人。

(7) 查询高等数学的平均成绩、最高分和最低分。

(8) 将英语成绩按从高到低排序。

4. 观察与思考

(1) LIKE 的通配符“%”和“_”有何不同?

(2) IS 能用“=”来代替吗?

(3) “=”与 IN 在什么情况下作用相同?

(4) 空值的使用,可分为哪几种情况?

(5) 聚集函数能否直接使用在 SELECT 子句、WHERE 子句、GROUP BY 子句、HAVING 子句之中?

(6) WHERE 子句与 HAVING 子句有何不同?

(7) COUNT(*)、COUNT(列名)、COUNT(DISTINCT 列名)三者的区别是什么?

习　题　5

一、选择题

1. 以下语句执行出错的原因是_____。

```
SELECT sno AS 学号, AVG(grade) AS 平均分 FROM score GROUP BY 学号;
```

A. 不能对 grade(学分)计算平均值

B. 不能在 GROUP BY 子句中使用别名

C. GROUP BY 子句必须有分组内容

D. score 表没有 sno 列

2. 统计表中记录数,使用_____聚合函数。

A. SUM　　B. AVG　　C. COUNT　　D. MAX

3. 在 SELECT 语句中使用_____关键字去掉结果集中的重复行。

A. ALL　　B. MERGE　　C. UPDATE　　D. DISTINCT

4. 查询 course 表的记录数,使用_____语句。

A. SELECT COUNT(cno) FROM course

B. SELECT COUNT(tno) FROM course

C. SELECT MAX(credit) FROM course

D. SELECT AVG(credit) FROM course

二、填空题

1. 在 DDL 语句中,_____语句可以创建表、ALTER TABLE 语句可以修改表、DROP TABLE 语句可以删除表。

2. 在 DML 语句中,INSERT 语句可以在表中插入记录、_____语句可以在表中修改记录、DELETE 语句可以在表中删除记录。

3. SELECT 语句有 SELECT、FROM、_____、GROUP BY、HAVING、ORDER BY 6 个子句。

4. WHERE 子句可以接收_____子句输出的数据。

三、问答题

1. SQL 语言可分为哪几类? 简述各类包含的语句。

2. SELECT 语句包含哪几个子句? 简述各个子句的功能。

3. 什么是 LIKE 谓词? 通配符有哪几种? 各有何功能?

4. 什么是聚合函数? 简述聚合函数的函数名称和功能。

5. 在一个 SELECT 语句中,当 WHERE 子句、GROUP BY 子句和 HAVING 子句同时出现在一个查询中时,SQL 的执行顺序如何?

四、应用题

1. 查询 score 表中学号为 121004,课程号为 1201 的学生成绩。

2. 列出 goods 表的商品名称、商品价格和打 7 折后的商品价格。

3. 查询 student 表中姓周的学生情况。
4. 查询通信专业的最高学分的学生的情况。
5. 查询 1004 课程的最高分、最低分及平均成绩。
6. 查询至少有 3 名学生选修且以 4 开头的课程号和平均分数。
7. 将计算机专业的学生按出生时间升序排列。
8. 查询各门课程最高分的课程号和分数,并按分数降序排列。
9. 查询选修课程 3 门以上且成绩在 85 分以上的学生的情况。