第5章 数据表的创建和管理

本章学习目标

- 熟悉 Oracle 数据库的常用数据类型
- 掌握数据表的创建、修改和删除命令
- 熟悉数据完整性约束的概念和类型
- 掌握数据完整性约束的定义和管理
- 掌握数据表中数据的插入、更新和删除命令
- 掌握使用 SQL Developer 工具管理数据表的方法
- 掌握使用 SQL Developer 工具完成数据导入的方法

本章首先介绍 Oracle 数据库的常用数据类型；然后介绍数据表的创建、修改和删除命令；接着介绍 Oracle 数据完整性的定义、类型及管理方法；之后介绍对数据表中数据进行插入、更新及删除的命令；最后介绍使用 SQL Developer 工具进行数据表管理以及进行数据导入的方法。

5.1 数 据 类 型

关系数据库通过表(关系)来表示实体及其联系。表是由行和列组成的二维表，每个表包含一组固定的列字段，而列字段由数据类型(DATATYPE)和长度(LENGTH)组成，以描述该表所跟踪的实体的属性。不同的数据类型决定了 Oracle 在存储它们时使用的方式、大小，以及在使用它们时选择什么运算符、函数等。

Oracle 支持多种数据类型，主要有数值类型、日期/时间类型和字符串类型。

(1) 数值类型：包括整数类型和小数类型。

(2) 日期/时间类型：包括 DATE 和 TIMESTAMP。

(3) 字符串类型：包括 CHAR、VARCHAR2、NVARCHAR2、NCHAR 和 LONG 5 种。

5.1.1 数值数据类型

数值数据类型主要用来存储数字。Oracle 提供了多种数值类型，不同的数值类型提供了不同的取值范围。可以存储的值范围越大，其所需要的存储空间就越大。Oracle 的数值类型主要通过 NUMBER 来实现。NUMBER 类型可以存储正数、负数、零、定点数和精度为 30 的浮点数，其使用的语法格式为 NUMBER(p[,s])。

NUMBER(p[,s])是可变长度的数值列,其允许为0、正值及负值。其中,p指所有有效数字的位数,取值范围为1～30;s指小数点以后的位数,正值s为小数位数,负值s表示四舍五入到小数点左部多少位,其取值范围为－84～127。

例如,number(5,3),这个字段的最大值是99.999,如果数值超过了位数限制就按照四舍五入的原则截取多余的位数。如在这个字段输入54.2923,则真正保存的数值是54.292。输入54.2347,则保存的数值是54.235。

如果不需要小数部分,则使用整数来保存数据,可以定义为number(m,0)或者number(m)。如number(3,0)或number(3)都表示保存三位长度的整数。

5.1.2 日期/时间类型

Oracle中表示日期/时间的数据类型主要包括DATE和TIMESTAMP。具体的内容和区别如表5-1所示。

表5-1 日期/时间数据类型

类型名称	说明
DATE	用来存储日期和时间,取值范围是公元前4712年1月1日到公元9999年12月31日
TIMESTAMP	用来存储日期和时间,与DATE类型的区别就是显示日期和时间时更准确,DATE类型精确到秒,而TIMESTAMP的数据类型可以精确到小数秒,TIMESTAMP存放日期和时间还能显示上午、下午和时区

在Oracle中,可以使用SYSDATE来查询数据库的当前时间。查询代码如下:

```
SELECT SYSDATE
FROM DUAL;
```

查询结果如图5-1所示。

```
SYSDATE
-----------
26-7月 -19
```

图5-1 显示系统时间

由图5-1可得,数据库默认的时间格式为“dd-mm-yyyy”。如果用户想要按照指定的格式输入时间,需要修改时间的默认格式,如修改输入格式为“yyyy-mm-dd”,其SQL语句如下:

```
ALTER SESSION SET nls_date_format = 'yyyy-mm-dd';
```

该语句仅修改本次登录会话(SESSION)的情况,不能修改别的会话。如果用户退出了,这些信息就丢失了。当用户重新登录以后,又是一个新的会话,即又恢复到系统默认的设置。

【例5.1】 日期类型数据实例。创建数据表tmp1,定义数据类型为DATE的字段d,向表中插入值。

首先创建表tmp1:

```
CREATE TABLE tmp1(d DATE);
```

向表中插入当前日期:

```
INSERT INTO tmp1 VALUES(SYSDATE);
```

向表中插入数据:

```
D
--------------
26-7月 -19
08-5月 -19
```

图 5-2　例 5.1 数据插入结果 1

```
INSERT INTO tmp1 VALUES('8-5 月-2019');
```

查看表中数据的结果如图 5-2 所示。

修改日期的默认格式：

```
ALTER SESSION SET nls_date_format = 'yyyy-mm-dd';
```

向表中插入'yyyy-mm-dd'格式的数据：

```
INSERT INTO tmp1 VALUES('2019-08-01');
```

向表中插入'yyyymmdd'格式的数据：

```
INSERT INTO tmp1 VALUES('20190802');
```

向表中插入日期和时间并指定格式：

```
INSERT INTO tmp1 VALUES(TO_DATE('2019-05-01 12:30:45','yyyy-mm-dd HH24:mi:ss'));
```

查看表中数据的结果如图 5-3 所示。

由图 5-3 结果可得，只显示日期，时间被省略了。

```
D
----------
2019-07-26
2019-05-08
2019-08-01
2019-08-02
2019-05-01
```

图 5-3　例 5.1 数据插入结果 2

【例 5.2】 创建数据表 tmp2，定义数据类型为 TIMESTAMP 的字段 ts，向表中插入值“2019-05-01 12:30:45”。

创建数据表 tmp2：

```
CREATE TABLE tmp2(ts TIMESTAMP);
```

向表中插入日期和时间并指定格式：

```
INSERT INTO tmp1
VALUES(TO_TIMESTAMP('2019-05-01 12:30:45','yyyy-mm-dd HH24:mi:ss'));
```

上述代码插入数据的结果如图 5-4 所示。

```
TS
---------------------------------------
01-5月 -19 12.30.45.000000 下午
```

图 5-4　【例 5.2】数据插入结果

如果只需要记录日期，则可以使用 DATE 类型；如果需要记录日期和时间，可以使用 TIMESTAMP 类型。特别地，如果需要显示上午、下午或者时区时，必须使用 TIMESTAMP 类型。在上面的例子中，TO_DATE()、TO_TIMESTAMP()是类型转换函数，可按照指定的格式进行类型转换。

5.1.3　字符串数据类型

字符串数据类型用来存储字符串数据。Oracle 中字符串类型指 CHAR、VARCHAR2、NCHAR、NVARCHAR2 和 LONG。字符串数据类型依据存储空间分为固定长度类型(CHAR/NCHAR) 和可变长度类型(VARCHAR2/NVARCHAR2)两种。

(1) 固定长度类型：固定长度类型是指虽然输入的字段值小于该字段的限制长度，但

是实际存储数据时，会先自动向右补足空格，才将字段值的内容存储到数据块中。

(2) 可变长度类型：可变长度类型是指当输入的字段值小于该字段的限制长度时，直接将字段值的内容存储到数据块中，而不会补上空格，这样可以节省数据块空间。

表 5-2 列出了 Oracle 中不同字符串类型的取值范围等信息。

表 5-2　Oracle 中的字符串类型

数据类型	说　明	取值范围
CHAR(n)	固定长度的字符型数据	0～2000
NCHAR(n)	存储 Unicode 的固定长度字符型数据，它的最大长度取决于国家字符集	0～1000
VARCHAR2(n)	可变长度的字符型数据	0～4000
NVARCHAR2(n)	存储 Unicode 的可变长度字符型数据	0～1000
LONG	存储可变长度的字符串	0～2G

VARCHAR2、NVARCHAR2 和 LONG 类型是可变长度类型，对于其存储空间需求取决于列值的实际长度。

【例 5.3】 创建表 tmp3，定义字段 ch 和 vch，数据类型分别定义为 CHAR(4)和 VARCHAR2(4)，向两个字段插入数据“ab”。

创建表 tmp3：

```
CREATE TABLE tmp3(ch CHAR(4), vch VARCHAR2(4));
```

插入数据：

```
INSERT INTO tmp3 VALUES('ab','ab');
```

查询两个字段值的长度：

```
SELECT LENGTH(CH), LENGTH(VCH)
FROM tmp3;
```

查询结果如图 5-5 所示。

```
LENGTH(CH)  LENGTH(VCH)
----------  -----------
         4            2
```

图 5-5　例 5.3 查询结果

可见固定长度的字符串在存储时长度是固定的，而可变长度字符串的存储长度是根据实际插入的数据长度而定的。

5.2　数据表的创建、修改和删除

创建数据表指的是在已经创建好的数据库中建立新表。创建数据表的过程是规定数据列的属性的过程，同时也是实施数据完整性约束的过程。本节将介绍数据表的创建、修改和删除等基本的管理操作。关于数据完整性约束的部分，将在 5.3 节中介绍。

5.2.1 创建数据表

在 Oracle 数据库中，创建表的命令为 CREATE TABLE，语法格式如下：

```
CREATE TABLE<表名>
(
字段名 1,数据类型[列级表约束],
字段名 2,数据类型[列级表约束],
…
[表级约束]
);
```

为创建的表命名时，需要遵循的规则如下：

(1) 表名首字符应该为字母；

(2) 不能使用保留字；

(3) 表名长度不超过 30 个字符；

(4) 同一用户下表名不能重复；

(5) 可以使用下画线、数字、字母，但不能使用空格和单引号；

(6) 表名不区分大小写，系统自动转换为大写。

在表中创建列时，必须为其指定数据类型，列的数据类型决定了数据的取值、范围和存储格式。如果需要创建多个列，列和列之间要用逗号隔开。

【例 5.4】 创建学生表 student，结构如表 5-3 所示。

表 5-3 student 表结构

列 名	数 据 类 型	说 明
Snum	char(10)	学号
Sname	char(20)	姓名
Ssex	char(4)	性别
Sbirth	date	出生日期

创建 student 表的 SQL 语句如下：

```
CREATE TABLE student
(
Snum char(10),
Sname char(20),
Ssex char(4),
Sbirth date
);
```

该 SQL 语句，可以在 SQL Plus 命令环境中执行，结果如图 5-6 所示。

表创建后，可以使用 DESC 命令查看数据表的结构，SQL 语句执行结果如图 5-7 所示。

另外，SQL Developer 工具也提供了执行 SQL 命令的窗口。打开 SQL Developer 工具，连接数据库，单击工具栏上的 按钮，即可打开查询编辑器，如图 5-8 所示。

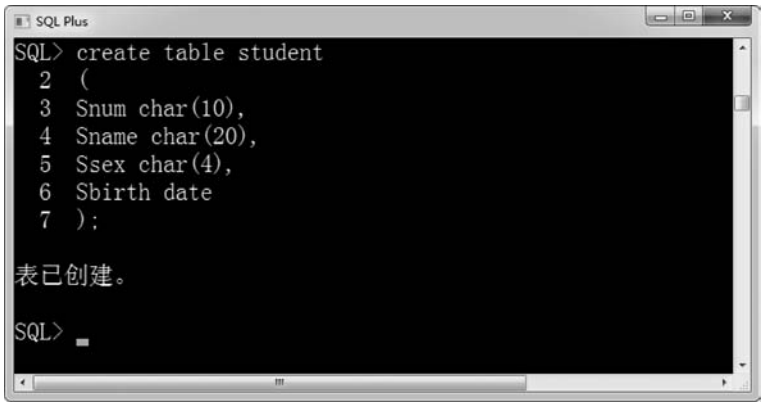

图 5-6　例 5.4 SQL Plus 环境创建 student 表

图 5-7　查看表结构

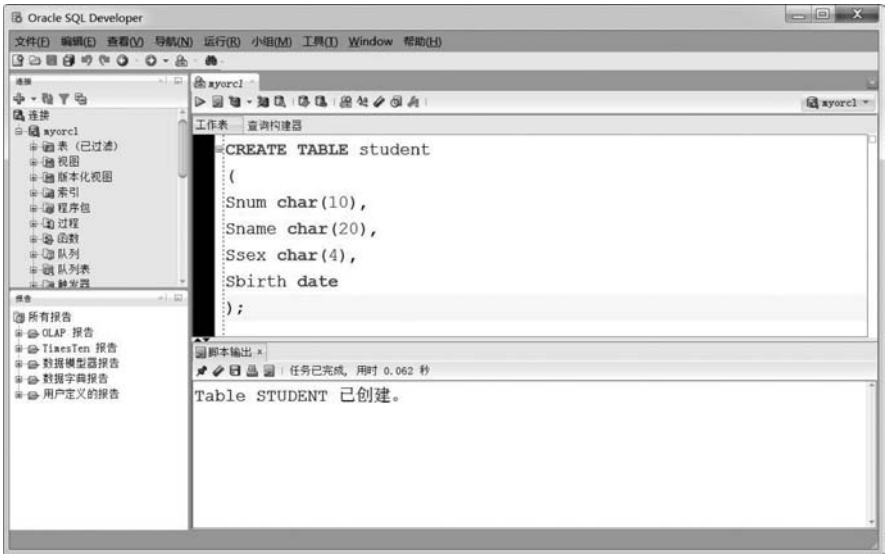

图 5-8　SQL Developer 环境创建 student 表

在工作表中，可以编辑 SQL 命令，单击 ▶ 执行 SQL 命令。执行结果在“脚本输出”区显示。同时，还可以显示“SQL 历史记录”“消息-日志”等。利用 SQL Developer 工具的查询编辑器进行 SQL 命令的编辑，更加灵活。因此，在之后的学习中，本书将主要借助此工具来进行 SQL 命令的编辑和执行。

5.2.2 修改数据表

修改数据表指的是修改数据库中已经存在的数据表的结构。Oracle 使用 ALTER TABLE 命令修改表。常用的修改表的操作有修改表名、修改字段数据类型或字段名、增加和删除字段、修改字段的排列位置、更改表的存储引擎、增减或删除表的约束等。本节将介绍表的基本的修改操作，关于约束的部分，将在 5.3 节中介绍。

ALTER TABLE 命令的语法格式如下：

```
ALTER TABLE<表名>
[RENAME {TO<新表名>|COLUMN<旧字段>TO<新字段>}]
[ADD (<新列名><数据类型>[列级表约束],…,n),]          /*增加新列*/
[ADD (CONSTRAINT 表级约束,…,n)]                      /*增加新的表级约束*/
[MODIFY(<列名>[<数据类型>][列级表约束],…,n,)]        /*修改已有列属性*/
[<DROP 子句>]
```

说明：

(1) RENAME TO：用于数据表或字段的重新命名，可以将旧表名改为新表名。当使用 COLUMN 关键字时，指修改字段名。

(2) ADD 子句：用于向表中增加一个新列或新的表级约束。新的列定义和创建表时定义列的格式一样，一次可以添加多个列，中间用逗号分开。

(3) MODIFY 子句：用于修改表中某列的属性，如数据类型、默认值等。在修改数据类型时需要注意，如果表中该列所存数据的类型与将要修改的列类型冲突，则会发生错误。如原来 CHAR 类型的列要修改为 NUMBER 类型，而原来列中的字符数据无法修改，此时会发生错误。

(4) DROP 子句：该子句用于从表中删除指定的字段或约束。语法格式为：

```
DROP
{
    COLUMN<列名>           /*删除指定的列*/
    |PRIMARY [KEY]         /*删除表的主键*/
    |UNIQUE (<列名>,…,n)   /*删除指定列上的 UNIQUE 约束*/
    |CONSTRAINT<约束名>    /*删除表的完整性约束*/
    |[CASCADE]             /*删除其他所有的完整性约束,这些约束依赖于被删除的完整性约束*/
}
```

注意：当要删除的 UNQIUE 约束和 PRIMARY KEY 约束所在的列被其他表的外键相关联时，如果没有删除外键，则不能删除引用完整性约束中的 UNQIUE 约束和 PRIMARY KEY 约束。

【例 5.5】 假设存在数据表 tmp_old，将数据表 tmp_old 修改为 tmp_new。

SQL 语句如下：

```
ALTER TABLE tmp_old
RENAME TO tmp_new;
```

执行后，表名发生变更，系统提示：Table TMP_OLD 已变更。

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

(1) 在表 student 中增加两列：Sdept(系别)、Snote(备注)。

SQL 语句如下：

```
ALTER TABLE student
ADD ( Sdept varchar(100),
    Snote varchar2(200) );
```

(2) 在表 student 中修改名为 Snote 的列数据类型长度为 1000。

SQL 语句如下：

```
ALTER TABLE student
MODIFY ( Snote varchar2(1000));
```

(3) 将表 student 中的 Snote 列改名为 note。

SQL 语句如下：

```
ALTER TABLE student
RENAME COLUMN Snote TO note;
```

(4) 在表 student 中删除名为 Sdept 和 note 的列。

SQL 语句如下：

```
ALTER TABLE student
DROP COLUMN Sdept;
ALTER TABLE student
DROP COLUMN note;
```

注意：一次只能删除一列。

5.2.3 删除数据表

删除数据表就是将数据库中已经存在的表从数据库中删除。在删除表的同时，表的定义和表中所有的数据均会被删除。因此，在进行表的删除操作前，最好对表中的数据进行备份，以免造成无法挽回的后果。

Oracle 使用 DROP TABLE 命令删除表，其语法格式如下：

```
DROP TABLE<表名>;
```

【例 5.7】 使用 DROP TABLE 命令删除数据库中的表 student。

SQL 语句如下：

```
DROP TABLE student;
```

结果如下：

系统提示：Table STUDENT 已删除。

DROP TABLE 可以一次删除一个或多个没有被其他表关联的数据表。在数据表之间存在外键关联的情况下，如果直接删除父表，结果会显示失败。原因是直接删除会破坏表的参照完整性。如果必须要删除，可以先删除与之关联的子表，再删除父表；或者先解除外键约束条件，再删除父表。

5.3 数据完整性约束

在关系表中，通常对列的取值有不同的约束和限制。符合这些约束的数据是有意义的数据；反之，不符合这些约束的数据往往是错误的、不合法的数据。Oracle 使用完整性约束来防止不合法的数据进入数据表中。完整性约束可以在创建表时定义，也可以创建表后添加。对于已经定义的约束可以进行删除操作。完整性规则定义在表上，存储在数据字典中。如果通过完整性约束增强的数据约束和限制改变了，管理员只需要修改完整性约束。与数据库相关的所有应用都会自动与修改后的约束保持一致。这种设置有力地简化了对数据和应用程序的管理。

5.3.1 数据完整性类型

数据的完整性就是指数据库中数据在逻辑上的一致性和准确性。关系数据库的完整性约束可以分为 3 类：域完整性、实体完整性和参照完整性。

1）域完整性

域完整性又称为列完整性，指定一个数据集对某一个列是否有效、确定是否允许空值。

域完整性通常是通过有效性检查来实现的，还可以通过限制数据类型、格式或者可能的取值范围来实现。

2）实体完整性

实体完整性也可以称为行完整性，要求表中的每行有一个唯一的标识符，这个标识符就是主关键字。通过索引、唯一性约束（UNIQUE）、主键约束（PRIMARY KEY ）可实现数据的实体完整性。

3）参照完整性

参照完整性又称为引用完整性，它保证主表与从表（或称为父表与子表、被参照表与参照表）中数据的一致性。在 Oracle 中，参照完整性的实现是通过定义外键（FOREIGN KEY）与主键（PRIMARY KEY）之间的对应关系实现的。如果被参照表中的一行被某外关键字引用，那么这一行既不能被删除，也不能修改主关键字。参照完整性确保键值在所有表中的一致性。

主键是指在表中能唯一标识表的每个数据行的一个或多个表列。外键是指如果一个表中的一个或若干个字段的组合的值取自另一个表的主键，则称该字段或字段组合为该表的外键。

例如，如图 5-9 所示，对于学生选课数据库中成绩表 SC 中的每条记录的学号 Snum、课程号 Cnum 必须是在学生表 student 和课程表 course 表中已存在的记录才有意义。因此，

将 student 表和 course 表作为主表，SC 表作为从表。将 SC 表中的 Snum 和 Cnum 定义为外键，从而建立主表和从表之间的数据联系，实现参照完整性。

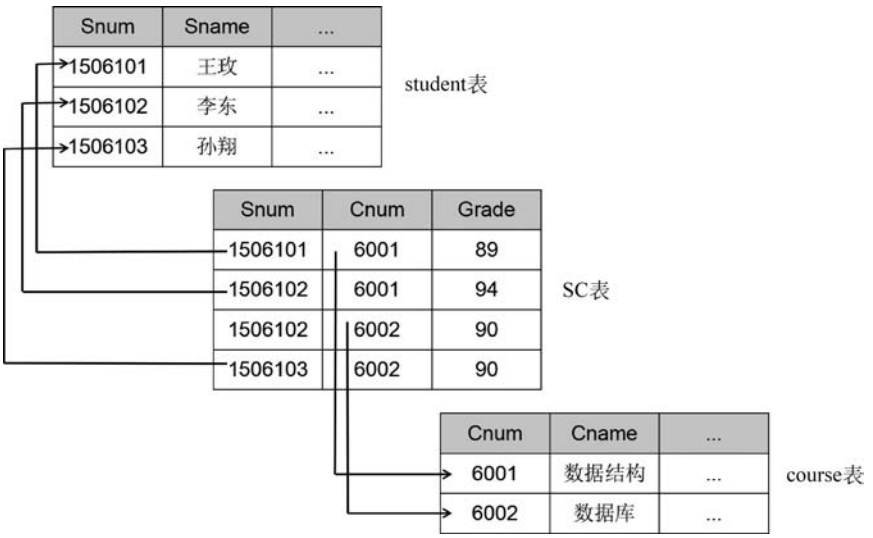

图 5-9　参照完整性

一旦定义了两个表之间的参照完整性，则需要满足如下 4 个条件。

(1) 从表不能引用不存在的键值。

(2) 如果主表中的键值更改了，那么在整个数据库中，对从表中键值的所有引用都要进行一致性的更改。

(3) 如果主表中没有关联的记录，则不能将记录添加到从表。

(4) 如果要删除主表中的某一条记录，应先删除从表中与该记录匹配的相关记录。

完整性约束是通过限制列数据、行数据和表之间数据来保证数据完整性的有效方法。约束是保证数据完整性的标准方法。每种数据的完整性类型，如域完整性、实体完整性和参照完整性，都可以由不同的约束类型来保障。约束确保将有效的数据输入列中，维护表与表之间的关系。

Oracle 中常用的约束包括非空约束(NOT NULL)、默认值约束(DEFAULT)、主键约束(PRIMARY KEY)、唯一性约束(UNIQUE)、检查约束(CHECK)和外键约束(FOREIGN KEY)。可以使用列级定义和表级定义两种方式定义约束。列级定义，即将约束定义为列定义的一部分，这种形式称为 inline 说明，也就是之前提到的列级约束；表级定义，即将约束作为表级的一部分进行定义，这种形式称为 out_of_line 说明。约束类型及说明如表 5-4 所示。

表 5-4　约束类型及说明

约束类型	关键字	定义方式	完整性类型
非空约束	NOT NULL	列级	域完整性
默认值约束	DEFAULT	列级	域完整性
主键约束	PRIMARY KEY	列级、表级	实体完整性

续表

约束类型	关键字	定义方式	完整性类型
唯一性约束	UNIQUE	列级、表级	实体完整性
检查约束	CHECK	列级、表级	域完整性
外键约束	FOREIGN KEY	表级	参照完整性

在 Oracle 中，完整性约束有如下 4 种状态。

(1) 禁止的非校验状态。表示该约束是不起作用的，即使该约束定义依然存储在数据字典中。

(2) 禁止的校验状态。表示对约束列的任何修改都是禁止的。这时，该约束上的索引都被删除，约束也被禁止。但是，这时仍然可以向表中有效地添加数据，即使这些数据与约束有冲突也没有关系。

(3) 允许的非校验状态或强制状态。该状态可以向表中添加数据，但是与约束有冲突的数据不能添加。如果表中已存在的数据与约束冲突，这些数据依然可以存在。

(4) 允许的校验状态。表示约束处于正常状态。这时表中所有的数据，无论是已有的还是新添加的，都必须满足约束条件。

5.3.2 使用非空约束

非空约束(NOT NULL)禁止列包含空值。虽然使用 NULL 不能定义完整性约束，但是可以指定一列允许包含空值。必须在列级使用 NOT NULL 和 NULL 约束，如果既不指定 NOT NULL，也不指定 NULL，则默认值是 NULL。

对于使用了非空约束的字段，如果用户在添加数据时没有指定值，数据库系统会报错。

1. 创建非空约束

非空约束的语法规则如下：

```
字段名 数据类型[NOT NULL|NULL]
```

【例 5.8】 定义数据表 tmp_null，指定学生姓名不能为空。

SQL 语句如下：

```
CREATE TABLE tmp_null
(
id NUMBER(11),
name VARCHAR(25) NOT NULL,
sex CHAR(4),
class VARCHAR(50)
);
```

语句执行后，创建 tmp_null 表，并且 name 列不能插入空值。当为表插入 name 列为空的一行数据时，系统提示“无法将 NULL 插入 ("SYSTEM"."TMP_NULL"."NAME")”，如图 5-10 所示。

2. 使用 ALTER TABLE 修改表添加非空约束

在创建表时如果没有添加非空约束，可以通过修改表添加非空约束。

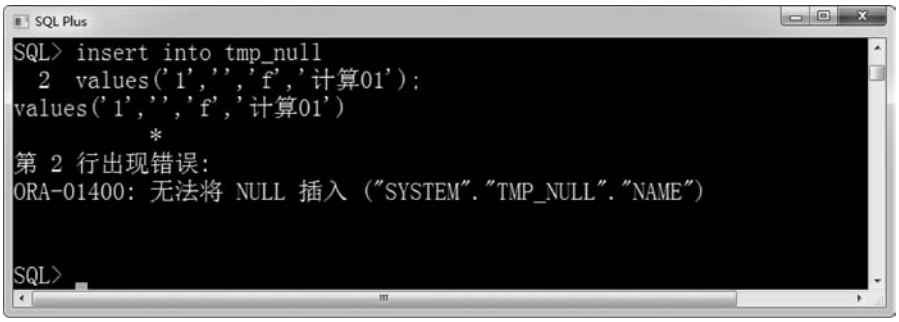

图 5-10　非空约束实例

【例 5.9】 修改数据表 tmp_null,指定班级列不能为空。

SQL 语句如下:

```
ALTER TABLE tmp_null
MODIFY class NOT NULL;
```

语句执行后,class 列不能插入空值。

3. 使用 ALTER TABLE 移除非空约束

对于不需要的非空约束,可以将其移除。

【例 5.10】 修改数据表 tmp_null,移除班级列上的非空约束。

SQL 语句如下:

```
ALTER TABLE tmp_null
MODIFY class NULL;
```

语句执行后,class 列可以插入空值。

当为 class 列添加非空约束后,不能向 class 列插入空值,此约束对定义约束之前插入的数据不限制。当为 class 列移除非空约束后,此列又可以插入空值。也就是说,约束的限制仅在定义后移除前有效。其他约束亦是如此。

5.3.3　使用默认值约束

默认值约束(DEFAULT)指定某列的默认值。如学生表中男性同学较多,性别可以设置默认为“男”。当插入一条新的记录但没有为这个字段赋值时,系统会自动为这个字段赋值为“男”。

默认值约束的语法规则如下:

```
字段名 数据类型 DEFAULT 默认值
```

1. 创建默认值约束

【例 5.11】 定义数据表 tmp_default,指定学生的默认性别为“男”。

SQL 语句如下:

```
CREATE TABLE tmp_default
(
```

```
id NUMBER(11),
name VARCHAR(25) ,
sex CHAR(4) DEFAULT '男',
class VARCHAR(50)
);
```

语句执行后，sex 列的默认值为“男”。当插入新的一行数据并且没有为 sex 列提供数值时，系统自动为此列添加默认值。

2. 使用 ALTER TABLE 修改表时添加默认值约束

在创建表时，如果没有添加默认值约束，可以通过修改表添加默认值约束。

【例 5.12】 修改数据表 tmp_default，指定班级列 class 的默认值为“计算 01”。

SQL 语句如下：

```
ALTER TABLE tmp_default
MODIFY class default '计算 01';
```

语句执行后，class 列的默认值为“计算 01”。

3. 使用 ALTER TABLE 移除默认值约束

对于不需要的默认值约束，可以将其移除。

【例 5.13】 修改数据表 tmp_default，移除班级列上的默认值约束。

SQL 语句如下：

```
ALTER TABLE tmp_default
MODIFY class default NULL;
```

或

```
ALTER TABLE tmp_default
MODIFY class default '';
```

语句执行后，class 列的默认值为空值，即将默认值移除。

5.3.4 使用主键约束

主键约束(PRIMARY KEY)要求主键列的数据唯一，并且不允许为空。主键又称主码，是表中一列或多列的组合。主键能够唯一地标识表中的一条记录，可以结合外键来定义不同数据表之间的关系，并且可以加快数据库查询的速度。主键是多列的组合时，称为复合主键。主键可以定义在表级，也可以定义在列级。但复合主键必须定义在表级。一个表中只能创建一个主键。

Oracle 会为主键建立索引。当删除主键时，由主键产生的索引会一并被删除。如果在建立主键前，该列已经有唯一性约束，则系统自动创建唯一性索引，主键也会共用此索引。这时删除主键，该索引仍然会存在。

1. 单字段主键

主键由一个字段组成，可以建立列级主键约束，也可以建立表级主键约束。语法格式如下。

列级主键约束：

```
字段名 数据类型 PRIMARY KEY
```

表级主键约束：

```
[CONSTRAINT<约束名>] PRIMARY KEY (字段名)
```

【例 5.14】 定义数据表 tmp_primarykey，将 id 列建立为主键。

SQL 语句如下。

列级主键约束：

```
CREATE TABLE tmp_pk
(
id NUMBER(11) PRIMARY KEY,
name VARCHAR(25) ,
sex CHAR(4),
class VARCHAR(50)
);
```

表级主键约束：

```
CREATE TABLE tmp_pk
(
id NUMBER(11),
name VARCHAR(25) ,
sex CHAR(4),
class VARCHAR(50),
PRIMARY KEY (id)
);
```

或

```
CREATE TABLE tmp_pk
(
id NUMBER(11),
name VARCHAR(25) ,
sex CHAR(4),
class VARCHAR(50),
CONSTRAINT tmp_pk PRIMARY KEY (id)
);
```

语句执行后，会在 id 上建立主键，则 id 列不允许出现空值和重复值。有 CONSTRAINT 关键词时，会为主键约束自定义一个约束名称，便于约束的维护。如果没有 CONSTRAINT 关键词，系统会为其定义一个主键约束名字，只是这个名字不太容易识别和记忆。需要注意，自定义的约束名称在同一个表空间中不允许重名。

2. 复合主键

复合主键包含多个字段，只能建立表级主键约束。语法格式如下：

```
[CONSTRAINT<约束名>] PRIMARY KEY (字段名 1,字段名 1,…,字段名 n)
```

【例 5.15】 定义数据表 tmp_pk_1，假设表上没有主键 id，为了唯一确定一个学生，可以把 name 和 class 联合起来作为主键。

SQL 语句如下：

```
CREATE TABLE tmp_pk_1
(
name VARCHAR(25),
sex CHAR(4),
class VARCHAR(50),
PRIMARY KEY (name,class)
);
```

或

```
CREATE TABLE tmp_pk_1
(
id NUMBER(11),
name VARCHAR(25) ,
sex CHAR(4),
class VARCHAR(50),
CONSTRAINT tmp_pk_1 PRIMARY KEY (name,class)
);
```

语句被执行后，创建了(name，class)的复合主键，在表中不能插入两条完全相同的(name，class)组合数据值。

3. 使用 ALTER TABLE 修改表添加主键约束

在创建表时如果没有添加主键约束，可以通过修改表添加主键约束。

【例 5.16】 为数据表 tmp_null 的 id 列添加主键约束。

SQL 语句如下：

```
ALTER TABLE tmp_null
ADD CONSTRAINT tmp_null_pk PRIMARY KEY (id);
```

4. 使用 ALTER TABLE 移除主键约束

对于不需要的主键约束，可以将其移除。

【例 5.17】 移除数据表 tmp_null 中的主键约束 tmp_null_pk。

SQL 语句如下：

```
ALTER TABLE tmp_null
DROP CONSTRAINT tmp_null_pk;
```

上述语句执行完成后，即可移除主键约束 tmp_null_pk。

5.3.5 使用唯一性约束

唯一性约束(UNIQUE)要求该列取值唯一，允许为空，但只能出现一个空值。添加了唯一性约束的单列称为唯一键，也可以定义复合唯一键。复合唯一键是指两个或者两个以上的列共同确定唯一的值。复合唯一键的唯一性约束必须定义在表级。如果不是复合唯一键，那么唯一性约束既可以定义在列级，也可以定义在表级。

1. 创建唯一键约束

唯一键由一个字段组成，可以建立列级唯一键约束，也可以建立表级唯一键约束。语法

格式如下。

列级唯一键约束：

```
字段名 数据类型 UNIQUE
```

表级唯一键约束：

```
UNIQUE(字段名)
```

或

```
[CONSTRAINT<约束名>] UNIQUE (字段名)
```

【例 5.18】 定义数据表 tmp_uq，为 name 列建立唯一键约束。

SQL 语句如下。

列级唯一键约束：

```
CREATE TABLE tmp_uq(
id NUMBER(11),
name VARCHAR(25) UNIQUE,
sex CHAR(4),
class VARCHAR(50)
);
```

表级唯一键约束：

```
CREATE TABLE tmp_uq(
id NUMBER(11),
name VARCHAR(25),
sex CHAR(4),
class VARCHAR(50),
UNIQUE(name)
);
```

或

```
CREATE TABLE tmp_uq(
id NUMBER(11),
name VARCHAR(25),
sex CHAR(4),
class VARCHAR(50),
CONSTRAINT tmp_uq UNIQUE(name)
);
```

语句执行后，会在 name 上建立唯一键约束，name 列不允许出现重复值。关于复合唯一键的使用，同复合主键，在此不再赘述。

2. 使用 ALTER TABLE 修改表添加唯一键约束

假设在创建表时没有添加唯一键约束，可以通过修改表添加唯一键约束。

【例 5.19】 为数据表 tmp_null 的 name 列添加唯一键约束。

SQL 语句如下：

```
ALTER TABLE tmp_null
ADD CONSTRAINT tmp_null_uq UNIQUE (name);
```


3. 使用 ALTER TABLE 语句移除唯一键约束

对于不需要的唯一键约束,可以将其移除。

【例 5.20】 移除数据表 tmp_null 中的唯一键约束 tmp_null_uq。

SQL 语句如下:

```
ALTER TABLE tmp_null
DROP CONSTRAINT tmp_null_uq;
```

上述语句执行完成后,即可移除唯一键约束 tmp_null_uq。

5.3.6 使用检查约束

检查约束(CHECK)规定限定的列值是否满足限定条件,从而确保数值的正确性。例如,性别字段中可以规定只能输入“男”或“女”。限定在单一列上的检查约束可以在列级定义,也可以在表级定义。涉及多个列的检查约束只能在表级定义。

1. 创建检查约束

检查约束的语法规则如下。

列级检查约束:

```
字段名 数据类型 CHECK(检查条件)
表级检查约束:
CHECK (检查条件)
```

或

```
[CONSTRAINT<约束名>] CHECK (检查条件)
```

【例 5.21】 定义数据表 tmp_check,指定学生的性别只能输入“男”或“女”。

SQL 语句如下。

列级检查约束:

```
CREATE TABLE tmp_ck(
id NUMBER(11),
name VARCHAR(25),
sex CHAR(4) CHECK(sex in ('男','女'),
class VARCHAR(50)
);
```

或

```
CREATE TABLE tmp_ck(
id NUMBER(11),
name VARCHAR(25),
sex CHAR(4),
class VARCHAR(50),
CHECK(sex in ('男','女'))
);
```

或

```
CREATE TABLE tmp_ck(
id NUMBER(11),
name VARCHAR(25),
sex CHAR(4),
class VARCHAR(50),
CONSTRAINT tmp_ck_ck CHECK(sex in ('男','女'))
);
```

以上语句执行成功后，表 tmp_ck 上的字段 sex 添加了检查约束，新插入记录时只能输入“男”和“女”。检查条件可以是任意能够确定真假的表达式。

2. 使用 ALTER TABLE 修改表添加检查约束

假设在创建表时没有添加检查约束，可以通过修改表添加检查约束。

【例 5.22】 为数据表 tmp_null 的 sex 列添加检查约束。

SQL 语句如下：

```
ALTER TABLE tmp_null
ADD CONSTRAINT tmp_null_ck CHECK (sex in ('男','女'));
```

3. 使用 ALTER TABLE 移除检查约束

对于不需要的检查约束，可以将其移除。

【例 5.23】 移除数据表 tmp_null 中的检查约束 tmp_null_ck。

SQL 语句如下：

```
ALTER TABLE tmp_null
DROP CONSTRAINT tmp_null_ck;
```

上述语句执行完成后，即可移除检查约束 tmp_null_ck。

5.3.7 使用外键约束

外键用来建立两个表之间的数据关联，它可以是一列或多列。一个表可以有一个或多个外键。外键对应的是参照完整性，其作用是保持表间数据的一致性。外键是表中的一个字段，它可以不是本表的主键，但必须对应另外一个表的主键。对于两个具有关联关系的表而言，相关联字段中主键所在的那个表称为主表，外键所在的表称为从表。一个表（从表）的外键列可以为空值，若不为空值，则必须是另一个表（主表）中的某个值。

1. 创建外键

创建外键的语法规则如下：

```
[CONSTRAINT<外键名>] FOREIGN KEY (字段名 1[,字段名 2,…])
REFERENCES<主表名>(主键名 1,[,主键名 2,…])
[ON DELETE {CASCADE|SET NULL}]
```

其中，“外键名”为定义的外键约束的名称，一个表中不能有相同名称的外键。“字段名”表示需要添加外键约束的字段列。“主表名”即被从表外键所依赖的表的名称。“主键名”表示主表中定义的主键列，或者列组合。

[ON DELETE {CASCADE|SET NULL}]指当主表中的列值被删除时，从表对应列值的操作，可以选择级联删除(CASCADE)，或者设置为空值(SET NULL)。如果没有选定该项，表示主表中的列值有关联数据存在，则不允许进行删除操作。

【例 5.24】 定义数据表 tmp_foreign 和 tmp_pk，建立两个表之间的关联关系。

两个表的结构如表 5-5 和表 5-6 所示。

表 5-5 tmp_foreign 表结构

字段名称	数据类型	备注
id	NUMBER(10)	员工编号
name	VARCHAR2(20)	员工姓名
depID	NUMBER(10)	部门编号

表 5-6 tmp_pk 表结构

字段名称	数据类型	备注
id	NUMBER(10)	部门编号
name	VARCHAR2(20)	部门名称

创建表及关联的 SQL 语句如下。

建立主表，并创建主键：

```
CREATE TABLE tmp_pk
(
id NUMBER(10) PRIMARY KEY,
name VARCHAR2(20)
);
```

建立从表，并创建外键：

```
CREATE TABLE tmp_foreign
(
id NUMBER(10) PRIMARY KEY,
name VARCHAR2(20),
depID NUMBER(10),
FOREIGN KEY(depID) REFERENCES tmp_pk(id)
);
```

或

```
CREATE TABLE tmp_foreign
(
id NUMBER(10) PRIMARY KEY,
name VARCHAR2(20),
depID NUMBER(10),
CONSTRAINT tmp_fk FOREIGN KEY(depID) REFERENCES tmp_pk(id)
);
```

2. 使用 ALTER TABLE 修改表添加外键约束

假设在创建表时没有添加外键约束，可以通过修改表添加外键约束。

【例 5.25】 假设存在 tmp_foreign_1，在创建时没有添加外键约束，为数据表 tmp_

foreign_1 的 depID 列添加外键约束，指定从表级联删除选项。

SQL 语句如下：

```
CREATE TABLE tmp_foreign_1
(
id NUMBER(10) PRIMARY KEY,
name VARCHAR2(20),
depID NUMBER(10)
);
ALTER TABLE tmp_foreign_1
ADD CONSTRAINT tmp_fk_1 FOREIGN KEY(depID) REFERENCES tmp_pk(id)
ON DELETE CASCADE;
```

3. 使用 ALTER TABLE 移除外键约束

对于不需要的外键约束，可以将其移除。

【例 5.26】 移除数据表 tmp_foreign_1 中的外键约束 tmp_fk_1。

SQL 语句如下：

```
ALTER TABLE tmp_foreign_1
DROP CONSTRAINT tmp_fk_1;
```

上述语句执行完成后，即可移除外键约束 tmp_fk_1。

5.3.8 设置表的属性值自动增加

在数据库应用中，当属性列不适合作主键时，可以定义自增约束字段，由系统自动生成主键值。可以通过为表主键添加 GENERATED BY DEFAULT AS IDENTITY 关键字来实现。默认地，在 Oracle 中该列的值的初始值是 1，每新增一条记录，字段值自动加 1。一个表只能有一个字段使用自增约束，且该字段必须为主键的一部分。

设置自增约束的语法规则如下：

```
字段名 数据类型 GENERATED BY DEFAULT AS IDENTITY
```

【例 5.27】 定义数据表 tmp_id，指定学生编号自动递增。

SQL 语句如下：

```
CREATE TABLE tmp_id(
id NUMBER(11) GENERATED BY DEFAULT AS IDENTITY,
name VARCHAR(25),
sex CHAR(4),
class VARCHAR(50)
);
```

上述语句执行后，创建数据表 tmp_id，其 id 列字段的值在进行数据添加时不需要用户提供数据，由系统维护。id 列初始值从 1 开始，每添加一条新记录，该值自动加 1。例如，执行 3 条同样的插入语句：

```
INSERT INTO tmp_id(name)
VALUES('aa');
```

查看表中的数据如图 5-11 所示。

	ID	NAME	SEX	CLASS
1	1	a	(null)	(null)
2	2	a	(null)	(null)
3	3	a	(null)	(null)

图 5-11　自增列示例

5.4　创建案例数据库表

本节以学生选课数据库为例进行讲解。数据库中包括 3 个表，表结构及表中的约束如表 5-7～表 5-9 所示。

表 5-7　学生表 student

列　名	数据类型	可否为空	默认值	说　明
Snum	CHAR(10)	×		学号、主键
Sname	VARCHAR2(20)	×		姓名
Ssex	CHAR(4)	×	“男”	性别
Sbirth	DATE	√		出生日期
Sdept	VARCHAR2(100)	√		系别
Snote	VARCHAR2(1000)	√		备注

创建语句如下：

```
CREATE TABLE student
(
Snum CHAR(10) PRIMARY KEY,
Sname VARCHAR2(20) NOT NULL,
Ssex CHAR(4) DEFAULT '男',
Sbirth DATE,
Sdept VARCHAR2(100),
Snote VARCHAR2(1000)
);
```

表 5-8　课程表 course

列　名	数据类型	可否为空	默认值	数据范围	说　明
Cnum	CHAR(8)	×			课程号、主键
Cname	VARCHR2(50)	×			课程名
Cterm	NUMBER(1)	√			开课学期
Chour	NUMBER(2)	×	48	8～80	学时
Cscore	NUMBER(1)	×	3	1～10	学分

创建语句如下：

```
CREATE TABLE course
(
Cnum CHAR(8) PRIMARY KEY,
```

```
Cname VARCHAR2(50) NOT NULL,
Cterm NUMBER(1),
Chour NUMBER(2) DEFAULT 48,
Cscore NUMBER(1) DEFAULT 3,
CONSTRAINT Course_hour_ck CHECK(Chour>= 8 and Chour<= 80),
CONSTRAINT Course_score_ck CHECK(Cscore>= 1 and Cscore<= 10)
);
```

表 5-9　选课表 SC

列　名	数据类型	可否为空	数据范围	说　明
Snum	CHAR(10)	×		学号、主键、外键
Cnum	CHAR(8)	×		课程号、主键外键
Grade	NUMBER(3)	√	0～100	成绩

创建语句如下：

```
CREATE TABLE SC
(
Snum CHAR(10),
Cnum CHAR(8),
Grade NUMBER(3),
CONSTRAINT SC_PK PRIMARY KEY(Snum,Cnum),
CONSTRAINT SC_FK1 FOREIGN KEY(Snum) REFERENCES student(Snum),
CONSTRAINT SC_FK2 FOREIGN KEY(Cnum) REFERENCES course(Cnum),
CONSTRAINT SC_Grade_ck CHECK(Grade>= 0 and Grade<= 100)
);
```

5.5　数据操作

新建的数据库表是空表，提供了存储和操作数据的结构。针对表中数据的操作主要包括插入、更新、删除、查询等。其中，插入(INSERT)、更新(UPDATE)和删除(DELETE)操作会使数据库中的数据发生变化，因此这 3 种操作在 SQL 语言中被称为数据操作语言(Data Manipulation Language，DML)。

5.5.1　插入数据

Oracle 使用 INSERT 语句向数据库表中插入新的数据记录。常用的插入方式有插入完整的记录、插入记录的一部分、插入多条记录及插入另一个查询的结果。另外，还可以为数据表批量导入数据。

1. INSERT 语句的语法

插入命令 INSERT 语句的语法格式如下：

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

该语句的功能是向指定表中插入一行记录。列名列表“(<列名 1>,<列名 2>,…,<列名 n>)”提供了插入数据对应的列，VALUES 后的列值列表“(<列值 1>,<列值 2>,…,

<列值 n>)”提供了要插入表中的数据值。

说明：

(1) 插入数据时，列名列表可以不与数据表的结构一致。只要保证列值列表中的数据与列名列表中的列名、数据类型一一对应即可。列名列表可以不包含数据表中的所有列，但是缺失的列必须是可以为空或有默认值约束的列。当没有为这些列提供数值时，系统会自动为其填充空值或默认值。

(2) 列名列表可以省略。如果不指定表名后面的列名列表，则 VALUES 子句中要给出每列的值，并且其提供的值要与原表中字段的顺序和数据类型完全一致，而且不能缺少字段。

(3) VALUES 中描述的值可以是一个常量、变量或一个表达式。字符串类型的数值必须用单引号引起来。字符串转换函数 TO_DATE 可以把字符串形式的日期型数据转换成 Oracle 规定的合法的日期型数据。

2. 插入完整的记录

向表中插入完整的记录，指在插入数据时为每个列都指定数据值。可以有两种方式：一种是指定所有字段名，另一种是不指定字段名。

【例 5.28】 向 student 表中插入新记录，指定所有列值。

SQL 语句如下：

```
INSERT INTO student(Snum,Sname,Ssex,Sbirth,Sdept,Snote)
VALUES('1506101','王玫','女','02-1月-1997','计算机系','入学新生');
INSERT INTO student(Snum,Sname,Sbirth,Ssex,Snote,Sdept)
VALUES('1506102','李东','13-3月-1998','男','入学新生','计算机系');
INSERT INTO student
VALUES('1506103','孙翔','男',TO_DATE('19971120','YYYYMMDD'),'计算机系','入学新生');
```

例 5.28 中给出了 3 条插入语句。语句执行后，查询 student 表中的数据。

SQL 语句如下：

```
SELECT *
FROM student;
```

执行结果如下：

```
SNUM      SNAME    SSEX    SBIRTH          SDEPT         SNOTE
------ ------- ---- ---------- ---------------- --------------------
1506101   王玫       女     02-1月-97     计算机系     入学新生
1506102   李东       男     13-3月-98     计算机系     入学新生
1506103   孙翔       男     20-11月-97    计算机系     入学新生
```

例 5.28 中，前两条语句给出了完整的列名列表和值列表。第 1 条语句给出的列名列表和表结构一致；第 2 条语句给出的列名列表和表结构不一致，要保证列名列表和列值列表中的数据一一对应；第 3 条语句省略了列名列表，这时候要保证列值列表与表结构完全一致，如果表结构修改了，则对列进行增加、删除或者位置改变的操作，在进行插入时数值列表的顺序也要相应修改。如果指定列名列表，则不会受到表结构的影响。

日期型数据在输入时需要注意数据库当前的默认格式，需要输入正确的日期格式才能正确插入数据。也可以使用字符串转换函数 TO_DATE()来将字符串表示的数据按照对应的格式转换成日期型数据。

注意：使用命令方式对表数据进行插入、更新和删除后，还需要使用 COMMIT 命令进行提交，这样才会把数据的改变真正保存到数据库中。为方便介绍，本书后面的 SQL 语句均省略 COMMIT 命令，运行时请自行添加。

3. 插入记录的一部分

在很多插入操作中，只提供了部分字段的数据值，其他字段的数据值可以采用 NULL 或默认值填充。对于未提供数据值的列必须是可以为空或者有自增值、默认值约束的列。

【例 5.29】 在 student 表中，插入记录的一部分。

SQL 语句如下：

```
INSERT INTO student(Snum,Sname)
VALUES('1506104','马兰');
```

查询结果如下：

```
SNUM      SNAME    SSEX    SBIRTH          SDEPT          SNOTE
------  -------  -----  ----------  ----------------  --------------------
1506101   王玫       女     02-1月-97     计算机系      入学新生
1506102   李东       男     13-3月-98     计算机系      入学新生
1506103   孙翔       男     20-11月-97    计算机系      入学新生
1506104   马兰       男
```

从例 5.29 中可以看出，提供部分数据值时，需要给出相应数值对应的列名列表。对于未给出的数据，系统自动填充 NULL、默认值或自增值。

4. 插入多条记录

使用多个 INSERT 语句可以向数据表中插入多条记录。

【例 5.30】 向 student 表中插入两条记录。

SQL 语句如下：

```
BEGIN
INSERT INTO student(Snum,Sname,Ssex)
VALUES('1506105','马丁','男');
INSERT INTO student(Snum,Sname,Ssex)
VALUES('1506106','郭阳','女');
END;
```

执行后，查询表中数据如下：

```
SNUM      SNAME    SSEX    SBIRTH          SDEPT          SNOTE
------  -------  -----  ----------  ----------------  --------------------
1506101   王玫       女     02-1月-97     计算机系      入学新生
1506102   李东       男     13-3月-98     计算机系      入学新生
1506103   孙翔       男     20-11月-97    计算机系      入学新生
1506104   马兰       男
1506105   马丁       男
1506106   郭阳       女
```

一次执行多条 INSERT 语句，需要将其放在 BEGIN 和 END 之间，组成一段 PL/SQL 过程。如果想使用一条 INSERT 语句同时插入多条记录，需要配合 SELECT 同时操作。

【例 5.31】 在一条 INSERT 语句中插入两条新记录。

SQL 语句如下：

```
INSERT INTO student(Snum,Sname,Ssex)
SELECT '1506108','刘飞飞','女' FROM DUAL
UNION ALL
SELECT '1506109','王子辰','男' FROM DUAL;
```

系统提示：

2 行已插入。

注意：一个同时插入多行记录的 INSERT 语句可以等同于多个单行插入的 INSERT 语句。但是，多行的 INSERT 语句在处理过程中，效率更高。因为 Oracle 执行单条 INSERT 语句插入多行数据比使用多个单行 INSERT 语句速度快，所以在插入多条记录时，最好选择使用单条 INSERT 语句的方式插入。

---

5. 插入另一个查询的结果

INSERT 语句可以将其他表中已存在的数据以 SELECT 查询结果的形式插入表中，从而快速地从一个或多个表中向另一个表中插入多行。其基本语法格式如下：

```
INSERT INTO 表名 1(列名 1,列名 2,…,列名 n)
SELECT 数值 1,数值 2,…,数值 n
FROM 表 2
WHERE 查询条件
```

其中，SELECT 语句后的“数值 1,数值 2,…数值 n”需要与“(列名 1,列名 2,…,列名 n)”的列的数据类型一一对应。SELECT 语句可以是任意合理的查询语句。SELECT 语句功能强大，将在第 6 章详细介绍。

【例 5.32】 创建一个新表 new_student，将 student 中的部分数据插入新表中。

SQL 语句如下：

```
CREATE TABLE new_student
(
num CHAR(10) NOT NULL,
name VARCHAR2(20) NOT NULL,
sex CHAR(4)
);
```

插入语句如下：

```
INSERT INTO new_student(num,name,sex)
SELECT Snum,Sname,Ssex
FROM student;
```

查询 new_student 表中的数据如下：

```
SNUM      SNAME    SSEX
------  -------  -------
1506101   王玫       女
1506102   李东       男
1506103   孙翔       男
1506104   马兰       男
1506105   马丁       男
1506106   郭阳       女
1506108   刘飞飞     女
1506109   王子辰     男
```

5.5.2 更新数据

对于表中已有的数据进行修改称为更新操作，采用 UPDATE 命令进行。其语法结构如下：

```
UPDATE 表名
SET 字段名 1 = 数值 1,字段名 2 = 数值 2,…,字段名 n = 数值 n
WHERE 条件表达式
```

其中，“字段名 1＝数值 1,字段名 2＝数值 2,…,字段名 n＝数值 n”表示为指定的字段赋予新的数值。更新多个列时，“列＝值”之间要有逗号隔开，最后一列不需要逗号。WHERE 条件表达式代表更新记录需要满足的条件，即只有符合条件的数据才会被修改。保证 UPDATE 语句以 WHERE 子句结束。如果忽略了 WHERE 子句，Oracle 将更新所有的行。

【例 5.33】 在 student 表中，将“刘飞飞”同学的系别改为“计算机系”，备注改为“入学新生”。

SQL 语句如下：

```
UPDATE student
SET Sdept = '计算机系',Snote = '入学新生'
WHERE Sname = '刘飞飞';
```

系统提示：1 行已更新。

如果有多行符合 WHERE 条件的数据，这些数据都将被修改。

5.5.3 删除数据

从数据表中删除数据使用 DELETE 语句，DELETE 语句允许使用 WHERE 子句指定删除条件。其语法格式如下：

```
DELETE FROM 表名
[WHERE 条件表达式]
```

在表名指定的表中，删除符合 WHERE 子句条件的所有记录。如果没有 WHERE 子句，DELETE 语句将删除表中的所有记录。

【例 5.34】 在 student 表中，删除学号为“1506108”和“1506109”的学生记录。

SQL 语句如下：

```
DELETE FROM student
WHERE Snum = '1506108' OR Snum = '1506109';
```

系统提示：2 行已删除。

【例 5.35】 删除 student 表中的所有记录。

SQL 语句如下：

```
DELETE FROM student;
```

系统提示：6 行已删除。

如果想删除表中的所有记录，还可以使用 TRANCATE TABLE 语句，如例 5.35 可以采用语句“TRANCATE TABLE student;”实现。TRANCATE TABLE 命令直接删除原来的表并重新创建一个表，因此执行速度比 DELETE 快。

5.6 使用 SQL Developer 工具管理数据表

SQL Developer 工具提供了方便快捷地管理数据表的方式，本节学习如何使用 SQL Developer 工具进行数据表的各种管理，包括创建数据表、修改数据表、删除数据表、数据导入等。

5.6.1 数据表的管理

1. 使用 SQL Developer 工具创建数据表

在 SQL Developer 中，可以快速完成数据表的创建，步骤如下所示。

(1) 打开 SQL Developer 工具，启动“myorcl”连接，右击“表”，在弹出的菜单中选择“新建表”命令，打开“创建表”对话框，如图 5-12 所示。将表中各列的定义逐一设置，如名称、数据类型、大小、非空、默认值、注释、主键等。单击右上角的“ ”，可以添加一列，单击“ ”按钮，可以删除已有列。

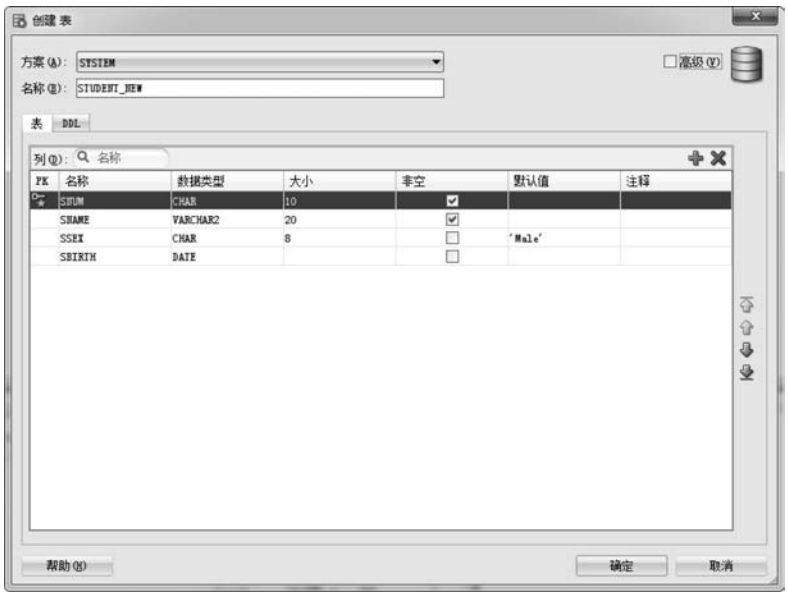

图 5-12 “创建表”基本属性对话框

(2) 要设置表的约束等高级属性,需要选中“高级”复选框,出现如图 5-13 所示的对话框。在此对话框中可以进行高级属性的设置。

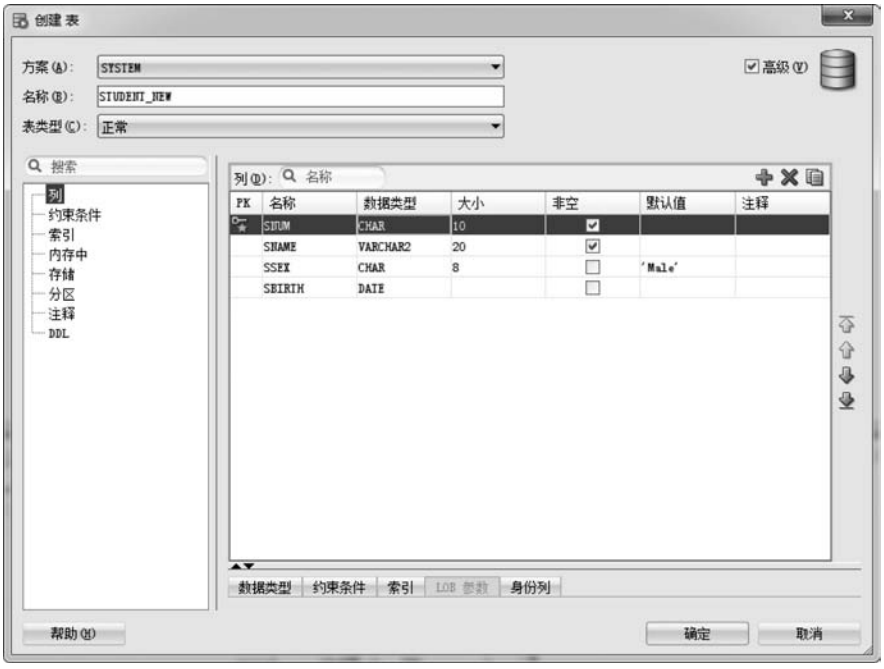

图 5-13 “创建表”高级属性对话框

设置完成后,单击“确定”按钮,表就创建好了。在主界面的“表”节点中可以查看到此表。

2. 使用 SQL Developer 工具修改数据表

在主界面的“表”节点中找到需要查看的数据表,将其选中,在主窗口中显示此表的相关信息,包括列、数据、约束条件、授权、统计信息、触发器、闪回、相关性、详细信息、分区、索引、SQL 等选项卡,如图 5-14 所示。

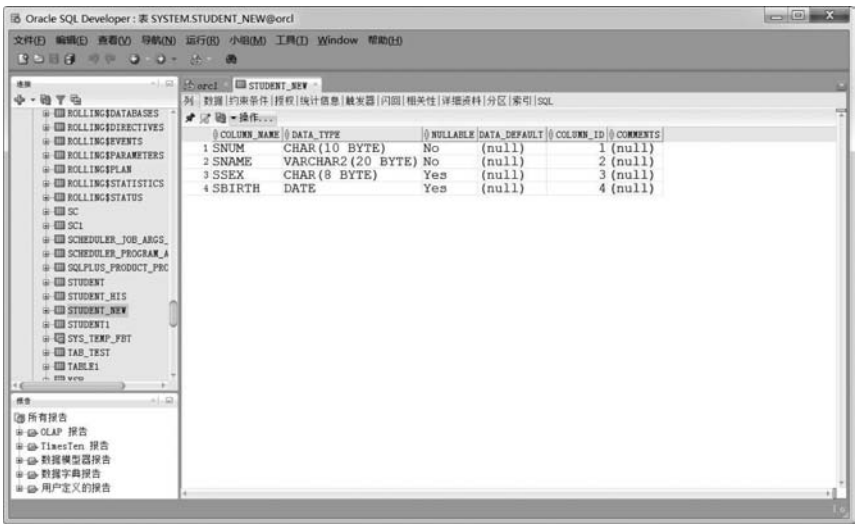

图 5-14 数据表基本信息

选中需要修改的表，右击，在出现的菜单中选择“编辑”命令，出现如图 5-15 所示的“编辑表”对话框。在“编辑表”对话框中，可以重新对表的结构进行设置。

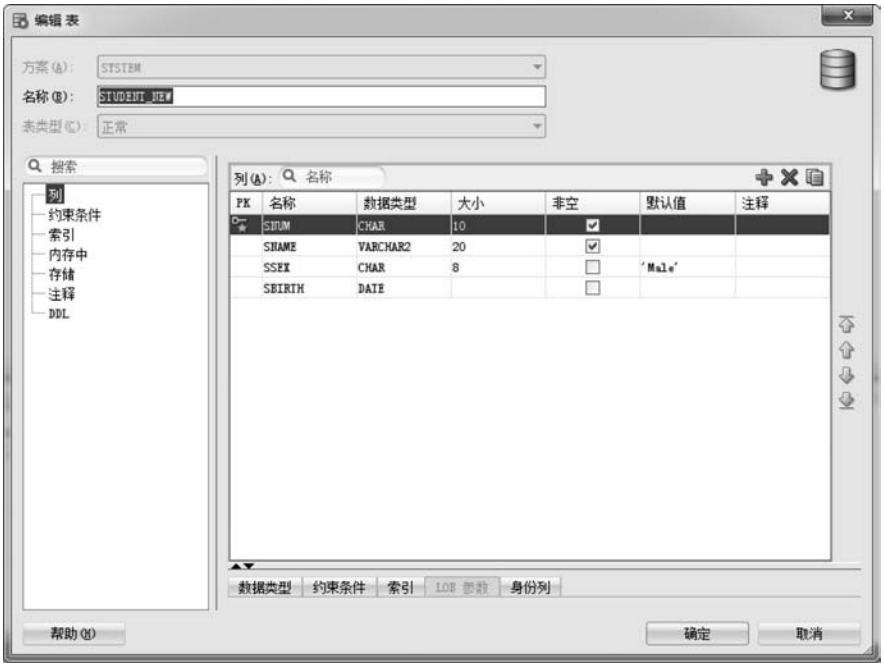

图 5-15 “编辑表”对话框

3. 使用 SQL Developer 工具删除数据表

在主界面的“表”节点下拉列表中选择需要删除的表，右击，在弹出的菜单中选择“表”，在二级菜单中选择“删除”命令，即可开启删除过程，如图 5-16 所示。出现如图 5-17 所示的“删除”对话框时，选中“级联约束条件”“清除”复选框，单击“应用”按钮，进行删除。删除成功后，显示“删除确认”提示框，如图 5-18 所示。

图 5-16 “删除”菜单

图 5-17 “删除”对话框

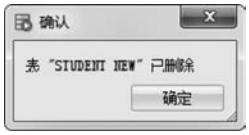

图 5-18 “删除确认”提示框

5.6.2 数据导入

SQL Developer 工具可以将 Excel 表格中的数据导入 Oracle 数据库中。为了后续的学习方便，本书将示例数据采用 Excel 表格的形式导入数据库中。下面以导入 student 表数据为例进行说明。数据导入需要经过如下 7 个步骤。

图 5-19 导入数据菜单

(1) 启动 SQL Developer，在已有的连接“myoracl”下展开“表”节点，单击选中“student”表，右击，在弹出的菜单中选择“导入数据”命令，如图 5-19 所示。

(2) 启动导入数据向导，进入“数据预览”对话框，如图 5-20 所示。在数据预览对话框中，单击“浏览”按钮，选择导入的数据文件。文件中的数据会显示在“文件内容”区域。

(3) 单击“下一步”按钮，进入“导入方法”对话框。选择合适的“导入方法”“导入行限制”等选项，如图 5-21 所示。

(4) 单击“下一步”按钮，进入“选择列”对话框，如图 5-22 所示。Excel 文件中的列会自动显示在“所选列”中，调整顺序使其对应表的结构。

(5) 单击“下一步”按钮，进入“列定义”对话框，如图 5-23 所示。默认匹配条件为“名称”，可以选择其他方式，如位置。对应表中的每列检查数据，保证数据的有效性。如果有数据不符合表定义，则需要重新修改这些数据。

(6) 单击“下一步”按钮，进入“完成”对话框，如图 5-24 所示。导入数据向导设置完成。

(7) 单击“完成”按钮，开启数据导入过程，完成导入后，显示“任务成功，导入已提交”，如图 5-25 所示。

在 SQL Developer 主界面中，再次单击“student”表，在右侧的主窗口中，单击“数据”选项卡，会看到数据已经在“student”表中，如图 5-26 所示。

利用同样的方法，可以插入 course 表和 SC 表需要的数据。

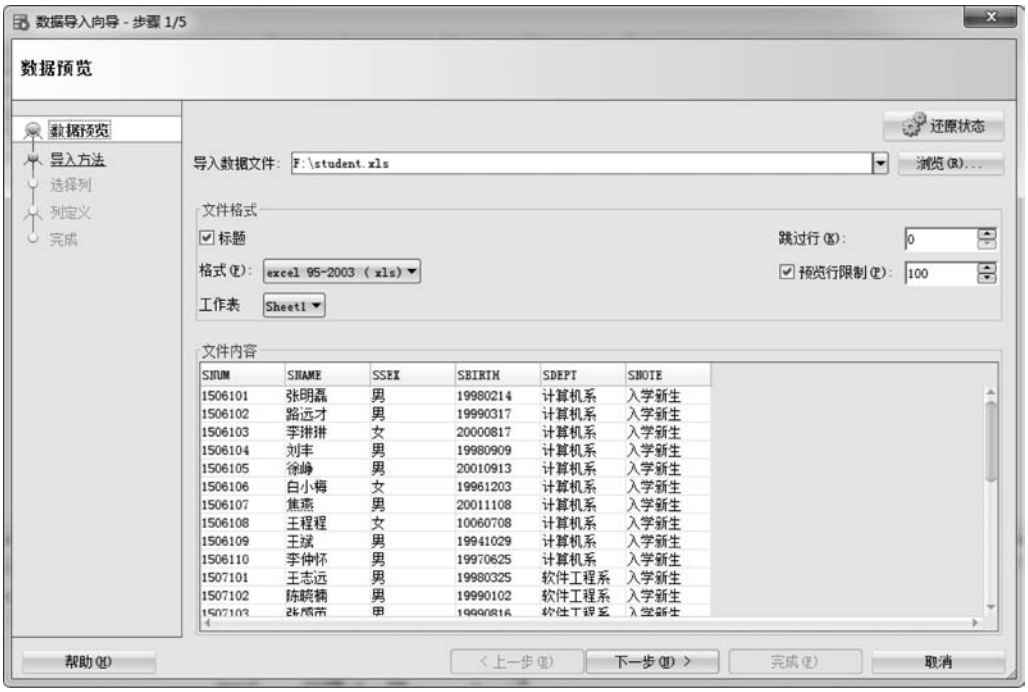

图 5-20 “数据预览”对话框

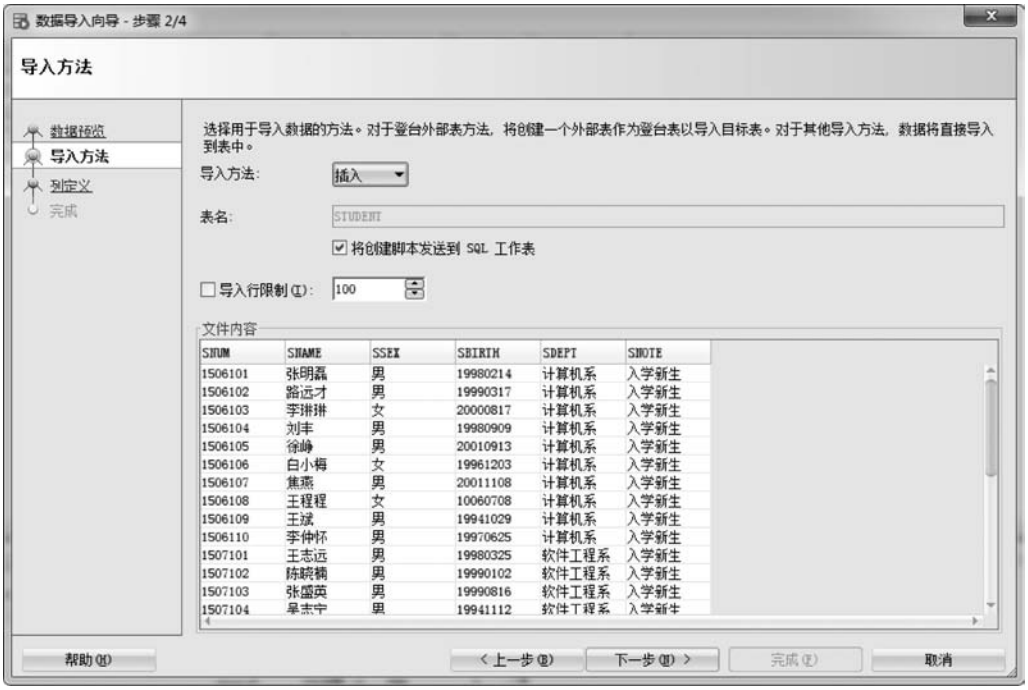

图 5-21 “导入方法”对话框

图 5-22 “选择列”对话框

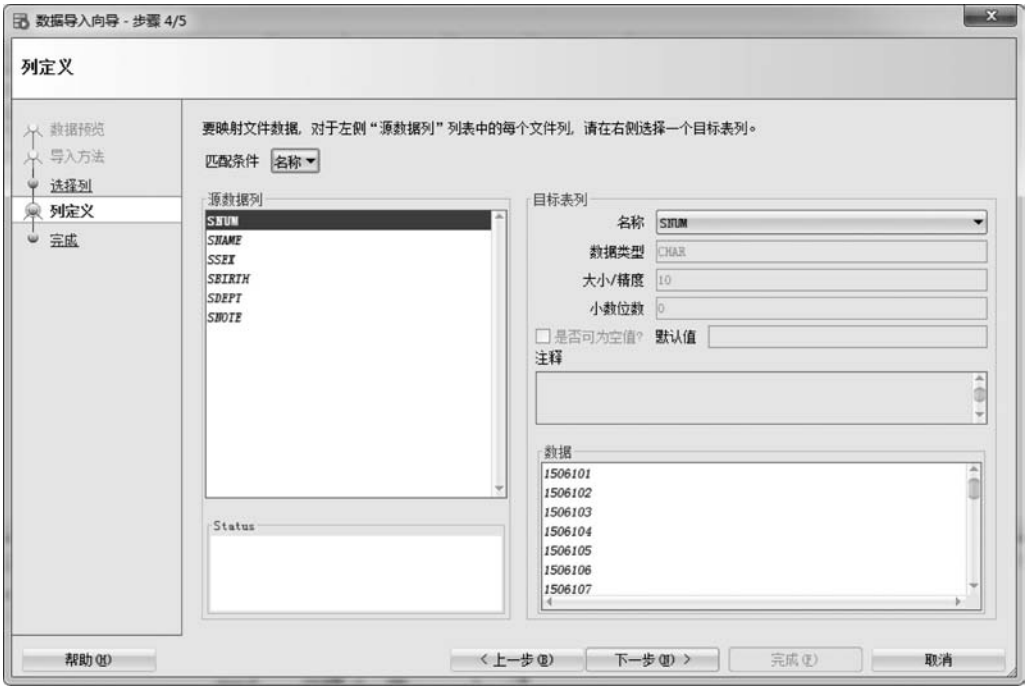

图 5-23 “列定义”对话框

图 5-24 “完成”对话框

图 5-25 “导入数据成功”对话框

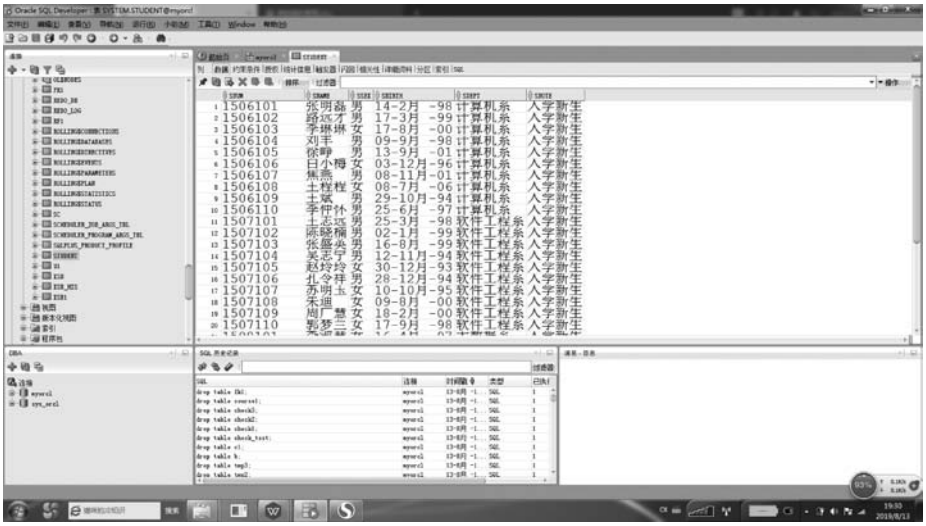

图 5-26 “student”表中的数据

5.7 本章小结

数据表是数据库中存储数据的逻辑结构。在 Oracle 中可以使用 CREATE TABLE、ALTER TABLE 和 DROP TABLE 命令来定义、修改和删除数据表。在定义表时需要指定数据列需要的数据类型和长度。Oracle 常用的数据类型有数值类型、日期/时间类型和字符串类型。通过在表上定义数据完整性约束可以防止不合法的数据进入数据表。完整性约束包括域完整性、实体完整性和参照完整性 3 种类型。在 Oracle 中可以定义非空约束、默认值约束、主键约束、唯一性约束、外键约束及自增值约束。在数据表中插入新数据、更新和删除已有数据分别使用 INSERT、UPDATE 和 DELETE 命令。使用 SQL Developer 工具可以便捷地对数据表进行操作并可以完成数据导入操作。

习 题 5

1. 现有汽车销售数据库，包含 3 个表，分别为汽车表(QCB)、顾客表(GKB)和销售记录(XSJL)，表结构如表 5-10～表 5-12 所示。根据表 5-10～表 5-12 中的要求，建立 3 个表。

表 5-10 汽车表 QCB

列 名	类 型	是否为空	约 束	说 明
汽车编号 QCBH	VARCHAR2(10)	NOT NULL	PRIMARY KEY	
汽车品牌 QCPP	VARCHAR2(50)	NOT NULL		
汽车型号 QCXH	VARCHAR2(50)	NOT NULL		

表 5-11 顾客表 GKB

列 名	类 型	是否为空	约 束	说 明
顾客编号 GKBH	VARCHAR2(10)	NOT NULL	PRIMARY KEY	
顾客姓名 GKXM	VARCHAR2(50)	NOT NULL		
联系电话 GKDH	VARCHAR2(20)			
顾客类型 GKLX	VARCHAR2(10)			

表 5-12 销售记录 XSJL

列 名	类 型	可否为空	默认值	约 束	说 明
销售编号 XSBH	VARCHAR2(10)	NOT NULL		PRIMARY KEY	
顾客编号 GKBH	VARCHAR2(50)	NOT NULL			
汽车编号 QCBH	VARCHAR2(20)	NOT NULL			
汽车价格 QCJG	NUMBER	NOT NULL			单位：万元
付款 FK	VARCHAR2(10)		否	是、否	
保险 BX	VARCHAR2(10)		否	是、否	
完成 WC	VARCHAR2(10)		否	是、否	

2. 为汽车表增加 1 列，出厂时间(CCSJ)，数据类型为 DATE 型，不允许为空。

3. 为顾客表增加 1 列，建档时间(JDSJ)，数据类型为 DATE 型。

4. 删除顾客表中的“建档时间(JDSJ)”列。

5. 为销售记录(XSJL)表添加外键约束,将顾客编号(GKBH)、汽车编号(QCBH)设置为外键,分别参照顾客表中的“顾客编号”、汽车表中的“汽车编号”。

6. 为汽车表插入一条数据,汽车编号为“Q1001”,汽车品牌为“大众”,汽车型号为“朗逸”,出厂日期为“2018 年 12 月 12 日”。

7. 修改汽车表中汽车编号为“Q1001”的出厂日期为“2019 年 3 月 1 日”。

8. 删除汽车表中汽车编号为“Q1001”的数据。

9. 使用 SQL Developer 工具为 3 个表导入若干数据。