第4章 表和视图

4.1 表的创建和操作

表由记录(行row)和字段(列column)构成，是数据库中存储数据的结构。要进行数据的存储和管理，首先要在数据库中创建表，即表的字段(列)结构。有了正确的结构，就可以用数据操作命令，插入、删除表中记录或对记录进行修改。比如，要进行图书管理，就需要创建图书和出版社等表，这里给出用于示范和训练的图书和出版社表的结构和内容，如表4-1、表4-2所示。

表4-1 图书表

图书编号	图书名称	出版社编号	作者	出版日期	数量	单价
A0001	计算机原理	01	刘勇	1998年5月7日	8	25.30
A0002	C语言程序设计	02	马丽	2003年1月2日	10	18.75
A0003	汇编语言程序设计	02	黄海明	2001年11月5日	15	20.18

表4-2 出版社表

编　号	出版社名称	地　址	联系电话
01	清华大学出版社	北京	010-83456272
02	西安电子科技大学出版社	西安	029-88201467

4.1.1 表的创建

1．创建表的语法

表的创建需要CREATE TABLE 系统权限，表的基本创建语法如下：

```
CREATE TABLE 表名
(列名 数据类型(宽度)[DEFAULT 表达式][COLUMN CONSTRAINT],
...
[TABLE CONSTRAINT]
[TABLE_PARTITION_CLAUSE]
);
```

由此可见，创建表最主要的是要说明表名、列名、列的数据类型和宽度，多列之间用“，”分隔。可以是用中文或英文作为表名和列名。表名最大长度为30个字符。在同一个用户下，表不能重名，但不同用户表的名称可以相重。另外，表的名称不能使用Oracle的保留字。在一张表中最多可以包含2000列。该语法中的其他部分根据需要添加，作用如下：

DEFAULT 表达式：用来定义列的默认值。

COLUMN CONSTRAINT：用来定义列级的约束条件。

TABLE CONSTRAINT：用来定义表级的约束条件。

TABLE_PARTITION_CLAUSE：定义表的分区子句。

【训练1】 创建图书和出版社表。

步骤1：创建出版社表，输入并执行以下命令：

```
CREATE TABLE 出版社(
编号 VARCHAR2(2),
出版社名称 VARCHAR2(30),
地址 VARCHAR2(30),
联系电话 VARCHAR2(20)
);
```

执行结果：

表已创建。

步骤2：创建图书表，输入并执行以下命令：

```
CREATE TABLE 图书(
图书编号 VARCHAR2(5),
图书名称 VARCHAR2(30),
出版社编号 VARCHAR2(2),
作者 VARCHAR2(10),
出版日期 DATE,
数量 NUMBER(3),
单价 NUMBER(7,2)
);
```

执行结果：

表已创建。

步骤3：使用DESCRIBE显示图书表的结构，输入并执行以下命令：

DESCRIBE 图书

执行结果为：

名称 是否为空? 类型

图书编号 VARCHAR2(5)

图书名称 VARCHAR2(30)

出版社编号 VARCHAR2(2)

作者 VARCHAR2(10)

出版日期 DATE

数量 NUMBER(3)

单价 NUMBER(7,2)

说明：在以上训练中，列名和数据类型之间用空格分隔，数据类型后的括号中为宽度(日期类型除外)。对于有小数的数字型，前一个参数为总宽度，后一个参数为小数位。用逗号分隔各列定义，但最后一列定义后不要加逗号。

2．通过子查询创建表

如果要创建一个同已有的表结构相同或部分相同的表，可以采用以下的语法：

CREATE TABLE 表名(列名...) AS SQL查询语句;

该语法既可以复制表的结构，也可以复制表的内容，并可以为新表命名新的列名。新的列名在表名后的括号中给出，如果省略将采用原来表的列名。复制的内容由查询语句的WHERE条件决定。

【训练2】 通过子查询创建新的图书表。

步骤1：完全复制图书表到“图书1”，输入并执行以下命令：

CREATE TABLE 图书1 AS SELECT * FROM 图书;

执行结果：

表已创建。

步骤2：创建新的图书表“图书2”，只包含书名和单价，输入并执行以下命令：

CREATE TABLE 图书2(书名,单价) AS SELECT 图书名称,单价 FROM 图书;

执行结果：

表已创建。

步骤3：创建新的图书表“图书3”，只包含书名和单价，不复制内容，输入并执行以下命令：

CREATE TABLE 图书3(书名,单价) AS SELECT 图书名称,单价 FROM 图书 WHERE 1=2;

执行结果：

表已创建。

说明：“图书1”表的内容和结构同“图书”表完全一致，相当于表的复制。

“图书2”表只包含“图书”表的两列 “图书名称”和“单价”，并且对字段重新进行了命名，“图书2”表的“书名”对应“图书”表的“图书名称”，“图书2”表的“单价”对应“图书”表的“单价”。

“图书3”表同“图书2”表的结构一样，但表的内容为空。因为WHERE条件始终为假，没有满足条件的记录，所以没有复制表的内容。

3．设置列的默认值

可以在创建表的同时指定列的默认值，这样在插入数据时，如果不插入相应的列，则该列取默认值，默认值由DEFAULT部分说明。

【训练3】 创建表时设置默认值。

步骤1：创建表时，设置表的默认值。

```
CREATE TABLE 图书4(
图书编号 VARCHAR2(5) DEFAULT NULL,
图书名称 VARCHAR2(30) DEFAULT '未知',
出版社编号 VARCHAR2(2) DEFAULT NULL,
出版日期 DATE DEFAULT '01-1月-1900',
作者 VARCHAR2(10) DEFAULT NULL,
数量 NUMBER(3) DEFAULT 0,
单价 NUMBER(7,2) DEFAULT NULL,
借出数量 NUMBER(3) DEFAULT 0
);
```

执行结果：

表已创建。

步骤2：插入数据。

INSERT INTO 图书4(图书编号) VALUES('A0001');

执行结果：

已创建 1 行。

步骤2：查询插入结果。

SELECT * FROM 图书4;

执行结果：

```
图书 图书名称        出版日期   作者   数量      单价     借出数
  量

------- ------------------------------------ ------------ ---------- -----
  ------ ------------------

A0001 未知          01-1月-00          0       0        0
```

说明：本训练中，只插入图书编号，其他部分取的是默认值。图书名称默认为“未知”，出版日期默认为1900年1月1日，数量默认为0，出版社编号、作者和单价的默认值为NULL。

【练习1】创建图书出借信息表，设置适当的默认值，并插入数据。

结构如下：

名称	是否为空?	类型
图书编号		VARCHAR2(10)
借书人		VARCHAR2(10)
借书日期		DATE
归还日期		DATE

4. 删除已创建的表

删除表的语法如下：

DROP TABLE 表名[CASCADE CONSTRAINTS];

表的删除者必须是表的创建者或具有DROP ANY TABLE权限。CASCADE CONSTRAINTS表示当要删除的表被其他表参照时，删除参照此表的约束条件。有关内容请参考下一节。

【训练4】 删除“图书1”表。

DROP TABLE 图书1;

执行结果：

表已丢弃。

【练习2】删除“图书2”、“图书3”和“图书4”表。

4.1.2 表的操作

1. 表的重命名

语法如下：

RENAME 旧表名 TO 新表名;

只有表的拥有者，才能修改表名。

【训练1】 修改“图书”表为“图书5”表：

RENAME 图书 TO 图书5;

执行结果：

表已重命名。

2．清空表

清空表的语法为：

TRUNCATE TABLE 表名；

清空表可删除表的全部数据并释放占用的存储空间。有关训练请参照DELETE语句部分，注意两者的区别。

3．添加注释

(1) 为表添加注释的语法为：

COMMENT ON TABLE 表名 IS '...';

该语法为表添加注释字符串。如IS后的字符串为空，则清除表注释。

【训练2】 为emp表添加注释：“公司雇员列表”。

COMMENT ON TABLE emp IS '公司雇员列表';

执行结果：

注释已创建。

(2) 为列添加注释的语法为：

COMMENT ON COLUMN 表名.列名 IS '...'

该语法为列添加注释字符串。如IS后的字符串为空，则清除列注释。

【训练3】 为emp表的deptno列添加注释：“部门编号”。

```
COMMENT ON COLUMN emp.deptno IS '部门编号';
```

执行结果：

```
注释已创建。
```

【练习1】清除emp表的注释。

4.1.3 查看表

使用以下语法可查看表的结构：

```
DESCRIBE 表名;
```

DESCRIBE可以简写为DESC。

可以通过对数据字典USER_OBJECTS的查询，显示当前模式用户的所有表。

【训练1】 显示当前用户的所有表。

```
SELECT object_name FROM user_objects WHERE object_type='TABLE';
```

执行结果：

```
OBJECT_NAME
--------------------
BONUS
DEPT
EMP
SALGRADE
出版社
图书
```

4.2 数据完整性和约束条件

4.2.1 数据完整性约束

表的数据有一定的取值范围和联系，多表之间的数据有时也有一定的参照关系。在创建表和修改表时，可通过定义约束条件来保证数据的完整性和一致性。约束条件是一些规则，在对数据进行插入、删除和修改时要对这些规则进行验证，从而起到约束作用。

完整性包括数据完整性和参照完整性，数据完整性定义表数据的约束条件，参照完整性定义数据之间的约束条件。数据完整性由主键(PRIMARY KEY)、非空(NOT NULL)、惟一(UNIQUE)和检查(CHECK)约束条件定义，参照完整性由外键(FOREIGN KEY)约束条件定义。

4.2.1 数据完整性约束

表的数据有一定的取值范围和联系，多表之间的数据有时也有一定的参照关系。在创建表和修改表时，可通过定义约束条件来保证数据的完整性和一致性。约束条件是一些规则，在对数据进行插入、删除和修改时要对这些规则进行验证，从而起到约束作用。

完整性包括数据完整性和参照完整性，数据完整性定义表数据的约束条件，参照完整性定义数据之间的约束条件。数据完整性由主键(PRIMARY KEY)、非空(NOT NULL)、惟一(UNIQUE)和检查(CHECK)约束条件定义，参照完整性由外键(FOREIGN KEY)约束条件定义。

4.2.2 表的五种约束

表共有五种约束，它们是主键、非空、惟一、检查和外键。

1．主键(PRIMARY KEY)

主键是表的主要完整性约束条件，主键惟一地标识表的每一行。一般情况下表都要定义主键，而且一个表只能定义一个主键。主键可以包含表的一列或多列，如果包含表的多列，则需要在表级定义。主键包含了主键每一列的非空约束和主键所有列的惟一约束。主键一旦成功定义，系统将自动生成一个B*树惟一索引，用于快速访问主键列。比如图书表中用“图书编号”列作主键，“图书编号”可以惟一地标识图书表的每一行。

主键约束的语法如下：

[CONSTRANT 约束名] PRIMARY KEY

--列级

[CONSTRANT 约束名] PRIMARY KEY(列名1,列名2,...)　--表级

2．非空(NOT NULL)

非空约束指定某列不能为空，它只能在列级定义。在默认情况下，Oracle允许列的内容为空值。比如“图书名称”列要求必须填写，可以为该列设置非空约束条件。

非空约束语法如下：

[CONSTRANT 约束名] NOT NULL

--列级

约束分为两级，一个约束条件根据具体情况，可以在列级或表级定义。

列级约束：约束表的某一列，出现在表的某列定义之后，约束条件只对该列起作用。

表级约束：约束表的一列或多列，如果涉及到多列，则必须在表级定义。表级约束出现在所有列定义之后。

4.2.2 表的五种约束

表共有五种约束，它们是主键、非空、惟一、检查和外键。

1. 主键(PRIMARY KEY)

主键是表的主要完整性约束条件，主键惟一地标识表的每一行。一般情况下表都要定义主键，而且一个表只能定义一个主键。主键可以包含表的一列或多列，如果包含表的多列，则需要在表级定义。主键包含了主键每一列的非空约束和主键所有列的惟一约束。主键一旦成功定义，系统将自动生成一个B*树惟一索引，用于快速访问主键列。比如图书表中用“图书编号”列作主键，“图书编号”可以惟一地标识图书表的每一行。

主键约束的语法如下：

[CONSTRANT 约束名] PRIMARY KEY

--列级

[CONSTRANT 约束名] PRIMARY KEY(列名1,列名2,...) --表级

2．非空(NOT NULL)

非空约束指定某列不能为空，它只能在列级定义。在默认情况下，Oracle允许列的内容为空值。比如“图书名称”列要求必须填写，可以为该列设置非空约束条件。

非空约束语法如下：

[CONSTRANT 约束名] NOT NULL

--列级

3．惟一(UNIQUE)

惟一约束条件要求表的一列或多列的组合内容必须惟一，即不相重，可以在列级或表级定义。但如果惟一约束包含表的多列，则必须在表级定义。比如出版社表的“联系电话”不应该重复，可以为其定义惟一约束。

惟一约束的语法如下：

[CONSTRANT 约束名] UNIQUE

--列级

[CONSTRANT 约束名] UNIQUE(列名1,列名2,...) --表级

4．检查(CHECK)

检查约束条件是用来定义表的一列或多列的一个约束条件，使表的每一列的内容必须满足该条件(列的内容为空除外)。在CHECK条件中，可以调用SYSDATE、USER等系统函数。一个列上可以定义多个CHECK约束条件，一个CHECK约束可以包含一列或多列。如果CHECK约束包含表的多列，则必须在表级定义。比如图书表的“单价”的值必须大于零，就可以设置成CHECK约束条件。

检查约束的语法如下：

[CONSTRANT 约束名] CHECK(约束条件) --列级，约束条件中只包含本列

[CONSTRANT 约束名] CHECK(约束条件) --表级，约束条件中包含多列

5．外键(FOREIGN KEY)

指定表的一列或多列的组合作为外键，外键参照指定的主键或惟一键。外键的值可以为NULL，如果不为NULL，就必须是指定主键或惟一键的值之一。外键通常用来约束两个表之间的数据关系，这两个表含有主键或惟一键的称为主表，定义外键的那张表称为子表。如果外键只包含一列，则可以在列级定义；如果包含多列，则必须在表级定义。

外键的列的个数、列的数据类型和长度，应该和参照的主键或惟一键一致。比如图书表的“出版社编号”列，可以定义成外键，参照出版社表的“编号”列，但“编号”列必须先定义成为主键或惟一键。如果外键定义成功，则出版社表称为主表，图书表称为子表。在表的创建过程中，应该先创建主表，后创建子表。

外键约束的语法如下：

第一种语法，如果子记录存在，则不允许删除主记录：

[CONSTRANT 约束名] FOREIGN KEY(列名1, 列名2,...)REFERENCES 表名(列名1,列名2,...)

第二种语法，如果子记录存在，则删除主记录时，级联删除子记录：

[CONSTRANT 约束名] FOREIGN KEY(列名1, 列名2,...)REFERENCES 表名(列名1,列名2,...)on delete cascade

第三种语法，如果子记录存在，则删除主记录时，将子记录置成空：

[CONSTRANT 约束名] FOREIGN KEY(列名1,列名2,...)REFERENCES 表名(列名1,列名2,...)on delete set null其中的表名为要参照的表名。

在以上5种约束的语法中，CONSTRANT关键字用来定义约束名，如果省略，则系统自动生成以SYS_开头的惟一约束名。约束名的作用是当发生违反约束条件的操作时，系统会显示违反的约束条件名称，这样用户就可以了解到发生错误的原因。

4.2.3 约束条件的创建

在表的创建语法中可以定义约束条件：

CREATE TABLE 表名(列名 数据类型[DEFAULT 表达式][COLUMN CONSTRAINT],...

[TABLE CONSTRAINT]

);

其中，COLUMN CONSTRAINT用来定义列级约束条件；TABLE CONSTRAINT用来定义表级约束条件。

【训练1】 创建带有约束条件的出版社表(如果已经存在，先删除)：

```
CREATE TABLE 出版社(
编号 VARCHAR2(2) CONSTRAINT PK_1 PRIMARY KEY,
出版社名称 VARCHAR2(30) NOT NULL ,
地址 VARCHAR2(30) DEFAULT '未知',
联系电话 VARCHAR2(20)
);
```

执行结果：

表已创建。

说明：出版社表的主键列是“编号”列，主键名为PK_1。“出版社名称”必须填写，地址的默认值为“未知”。

【训练2】 创建带有约束条件(包括外键)的图书表(如果已经存在，先删除):

CREATE TABLE 图书(图书编号 VARCHAR2(5) CONSTRAINT PK_2 PRIMARY KEY,

图书名称 VARCHAR2(30) NOT NULL,

出版社编号 VARCHAR2(2) CHECK(LENGTH(出版社编号)=2) NOT NULL,

作者 VARCHAR2(10) DEFAULT '未知',

出版日期 DATE DEFAULT '01-1月-1900',

数量 NUMBER(3) DEFAULT 1 CHECK(数量>0),

单价 NUMBER(7,2),

CONSTRAINT YS_1 UNIQUE(图书名称,作者),

CONSTRAINT FK_1 FOREIGN KEY(出版社编号) REFERENCES 出版社(编号) ON DELETE CASCADE

);

执行结果：

表已创建。

说明：因为两个表同属于一个用户，故约束名不能相重，图书表的主键为“图书编号”列，主键名为PK_2。其中，约束条件CHECK(LENGTH(出版社编号)=2)表示出版社编号的长度必须是2，约束条件UNIQUE(图书名称,作者)表示“图书名称”和“作者”两列的内容组合必须惟一。FOREIGN KEY(出版社编号) REFERENCES 出版社(编号) 表示图书表的“出版社编号”列参照出版社的“编号”主键列。出版社表为主表，图书表为子表，出版社表必须先创建。ON DELETE CASCADE表示当删除出版社表的记录时，图书表中的相关记录同时删除，比如删除清华大学出版社，则图书表中清华大学出版社的图书也会被删除。

如果同时出现DEFAULT和CHECK，则DEFAULT需要出现在CHECK约束条件之前。

【训练3】 插入数据，验证约束条件。

步骤1：插入出版社信息：

INSERT INTO 出版社 VALUES('01','清华大学出版社','北京','010-83456272');

执行结果：

已创建1行。

继续插入

INSERT INTO 出版社 VALUES('01','电子科技大学出版社','西安','029-88201467');

执行结果：

ERROR 位于第1行:

ORA-00001: 违反惟一约束条件 (SCOTT.PK_1)

第二个插入语句违反约束条件PK_1，即出版社表的主键约束，原因是主键的值必须是惟一的。修改第二个语句的编号为“02”，重新执行：

INSERT INTO 出版社 VALUES('02','电子科技大学出版社','西安','029-88201467');

执行结果：

已创建 1 行。

步骤2：插入图书信息：

INSERT INTO 图书(图书编号,图书名称,出版社编号,作者,单价) VALUES('A0001','计算机原理','01','刘勇',25.30);

执行结果：

已创建1行。

继续插入：

INSERT INTO 图书(图书编号，图书名称，出版社编号，作者，单价) VALUES('A0002',' C语言程序设计','03','马丽', 18.75);

执行结果：

ERROR 位于第 1 行:

ORA-02291: 违反完整约束条件 (SCOTT.FK_1) - 未找到父项关键字

第二个插入语句违反外键约束关系FK_1，因为在出版社表中，被参照的主键列中没有“03”这个出版社，所以产生未找到父项关键字的错误，修改后重新插入：

INSERT INTO 图书(图书编号，图书名称，出版社编号，作者，单价) VALUES(‘A0002′,‘ C语言程序设计′,‘02′,‘马丽′,18.75);

执行结果：

已创建 1 行。

继续插入：

INSERT INTO 图书(图书编号,图书名称,出版社编号,作者,数量,单价) VALUES('A0003','汇编语言程序设计','02','黄海明',0,20.18);

执行结果：

ERROR 位于第 1 行:

ORA-02290: 违反检查约束条件 (SCOTT.SYS_C003114)

插入的数量为0，违反约束条件CHECK(数量>0)。该约束条件没有命名，所以约束名SYS_C003114为系统自动生成。修改后重新执行：

INSERT INTO 图书(图书编号,图书名称,出版社编号,作者,数量,单价) VALUES('A0003','汇编语言程序设计','02','黄海明',15,20.18);

执行结果：

已创建 1 行。

步骤3：显示插入结果：

SELECT * FROM 出版社;

执行结果：

编号 出版社名称 地址 联系电话

------ ------------------------------ ------------------------- --------

----

01 清华大学出版社 北京 010-83456272

02 电子科技大学出版社 西安 029-88201467

继续查询：

SELECT * FROM 图书;

执行结果：

图书编号 图书名称 出版社编号 作者 出版日期 数量 单价

---------- ---------- ---------- ---------- ------ ---------- ---------- ----------

A0001 计算机原理 01 刘勇 01-1月 -00 1 25.3
A0002 C语言程序设计 02 马丽 01-1月 -00 1 18.75
A0003 汇编语言程序设计 02 黄海明 01-1月 -00 15 20.18

步骤4：提交插入的数据：

COMMIT;

执行结果：

提交完成。

说明：在图书表中，没有插入的数量取默认值1，没有插入的出版日期取默认值01-1月-00(即1900年1月1日)。

【训练4】 通过删除数据验证ON DELETE CASCADE的作用。

步骤1：删除出版社01(清华大学)：

```
DELETE FROM 出版社 WHERE 编号='01';
```

执行结果：

```
已删除 1 行。
```

步骤2：显示删除结果：

显示出版社表结果：

SELECT * FROM 出版社;

执行结果：

编号 出版社名称 地址 联系电话

----- ---------- ---------- -------

02 电子科技大学出版社 西安 029-88201467

显示图书表结果：

SELECT * FROM 图书;

执行结果：

图书编号 图书名称 出版社编号 作者 出版日期 数量 单价

---------- ---------- ---------- ---------- ------- ---------- ----------

A0002 C语言程序设计 02 马丽 01-1月 -00 1 18.75

A0003 汇编语言程序设计 02 黄海明 01-1月 -00 15 20.18

步骤3：恢复删除:

ROLLBACK;

回退已完成。

说明：参见训练2，外键约束FK_1带有ON DELETE CASCAD选项，删除清华大学出版社时，对应的图书也自动删除。其他两种情况用户可自行验证。

【练习1】创建学生、系部表，添加必要主键、外键等约束条件。

4.2.4 查看约束条件

数据字典USER_CONSTRAINTS中包含了当前模式用户的约束条件信息。其中，CONSTRAINTS_TYPE 显示的约束类型为：

C：CHECK约束。

P：PRIMARY KEY约束。

U：UNIQUE约束。

R：FOREIGN KEY约束。

其他信息可根据需要进行查询显示，可用DESCRIBE命令查看USER_CONSTRAINTS的结构。

【训练1】 检查表的约束信息：

```
SELECT
    CONSTRAINT_NAME,CONSTRAINT_TYPE,SEARCH_CONDITION
    FROM USER_CONSTRAINTS
WHERE TABLE_NAME='图书';
```

执行结果：

```
CONSTRAINT_NAME      C  SEARCH_CONDITION
---------------------------------- ----- ----------------------------------
SYS_ C003111          C "图书名称" IS NOT NULL
SYS_C003112          C "出版社编号" IS NOT NULL
SYS_C003113          C LENGTH(出版社编号)=2
SYS_C003114          C 数量>0
PK_2                 P
YS_1                 U
FK_1                 R
```

说明：图书表共有 7 个约束条件，一个PRIMARY KEY(P)约束PK_2，一个FOREIGN KEY(R)约束FK_1，一个UNIQUE(R)约束YS_1和4个CHECK(C)约束SYS_C003111、SYS_C003112、SYS_C003113和SYS_C003114，4个CHECK约束的名字是由系统命名的。

4.2.5 使约束生效和失效

约束的作用是保护数据完整性，但有的时候约束的条件可能不再适用或没有必要，如果这个约束条件依然发生作用就会影响操作的效率，比如导出和导入数据时要暂时关闭约束条件，这时可以使用下面的命令关闭或打开约束条件。

使约束条件失效：

ALTER TABLE 表名 DISABLE CONSTRANT 约束名;

使约束条件生效：

ALTER TABLE 表名 ENABLE CONSTRANT 约束名;

【训练1】 使图书表的数量检查失效。

步骤1：使约束条件SYS_C003114(数量>0)失效：

```
ALTER TABLE 图书 DISABLE CONSTRAINT SYS_C003114;
```

执行结果：

表已更改。

步骤2：修改数量为0：

```
UPDATE 图书 SET 数量=0 WHERE 图书编号='A0001';
```

执行结果：

已更新 1 行。

步骤3：使约束条件SYS_C003114生效：

ALTER TABLE 图书 ENABLE CONSTRAINT SYS_C003114;

执行结果：

ERROR 位于第 1 行:

ORA-02293: 无法验证 (SCOTT.SYS_C003114) - 违反检查约束条件

继续执行：

UPDATE 图书 SET 数量=5 WHERE 图书编号='A0001';

执行结果：

已更新 1 行。

继续执行：

```
ALTER TABLE 图书 ENABLE CONSTRAINT SYS_C003114;
```

执行结果：

```
表已更改。
```

说明：在步骤1中，先使名称为SYS_C003114 (数量>0)的检查条件暂时失效，所以步骤2修改第1条记录的数量为0才能成功。步骤3使该约束条件重新生效，但因为表中有数据不满足该约束条件，所以发生错误，通过修改第一条记录的数量为5，使约束条件重新生效。

4.3 修改表结构

4.3.1 增加新列

增加新列的语法如下：

ALTER TABLE 表名

ADD 列名 数据类型[DEFAULT 表达式][COLUMN CONSTRAINT];

如果要为表同时增加多列，可以按以下格式进行：

ALTER TABLE 表名

ADD (列名 数据类型 [DEFAULT 表达式][COLUMN CONSTRAINT]...);

通过增加新列可以指定新列的数据类型、宽度、默认值和约束条件。增加的新列总是位于表的最后。假如新列定义了默认值，则新列的所有行自动填充默认值。对于有数据的表，新增加列的值为NULL，所以有数据的表，新增加列不能指定为NOT NULL约束条件。

【训练1】 为“出版社”增加一列“电子邮件”：

```
ALTER TABLE 出版社
ADD 电子邮件 VARCHAR2(30) CHECK(电子邮件 LIKE '%@%');
```

显示结果：

```
表已更改。
```

说明：为出版社新增加了一列“电子邮件”，数据类型为VARCHAR2，宽度为30。CHECK(电子邮件 LIKE '%@%')表示电子邮件中必须包含字符“@”。可用DESCRIBE命令查看表的新结构。

4.3.2 修改列

修改列的语法如下：

ALTER TABLE 表名

MODIFY 列名 数据类型 [DEFAULT 表达式][COLUMN CONSTRAINT]

如果要对表同时修改多列，可以按以下格式进行：

ALTER TABLE 表名

MODIFY (列名 数据类型 [DEFAULT 表达式][COLUMN CONSTRAINT]...);

其中，列名是要修改的列的标识，不能修改。如果要改变列名，只能先删除该列，然后重新增加。其他部分都可以进行修改，如果没有给出新的定义，表示该部分属性不变。

修改列定义还有以下一些特点：

(1) 列的宽度可以增加或减小，在表的列没有数据或数据为NULL时才能减小宽度。

(2) 在表的列没有数据或数据为NULL时才能改变数据类型，CHAR和VARCHAR2之间可以随意转换。

(3) 只有当列的值非空时，才能增加约束条件NOT NULL。

(4) 修改列的默认值，只影响以后插入的数据。

【训练1】 修改“出版社”表“电子邮件”列的宽度为40。

```
ALTER TABLE 出版社
MODIFY 电子邮件 VARCHAR2(40);
```

执行结果：

```
表已更改。
```

说明：将“电子邮件”列的宽度由原来的30修改为40，约束条件保持不变。可用DESCRIBE命令查看新结构。

4.3.3 删除列

删除列的语法如下：

ALTER TABLE 表名

DROP COLUMN 列名[CASCADE CONSTRAINTS];

如果要同时删除多列，可以按以下格式进行：

ALTER TABLE 表名

DROP(COLUMN 列名 数据类型 [DEFAULT 表达式][COLUMN CONSTRAINT]...)

[CASCADE CONSTRAINTS];

当删除列时，列上的索引和约束条件同时被删除。但如果列是多列约束的一部分，则必须指定CASCADE CONSTRAINTS才能删除约束条件。

【训练1】 删除“出版社”表的“电子邮件”列。

ALTER TABLE 出版社

DROP COLUMN 电子邮件;

执行结果：

表已更改。

说明：此训练将“电子邮件”列删除。可用DESCRIBE命令查看新结构。

使用以下语法，可以将列置成UNUSED状态，这样就不会在表中显示出该列：

ALTER TABLE 表名 SET UNUSED COLUMN 列名 [CASCADE CONSTRAINTS];

以后可以重新使用或删除该列。通过数据字典可以查看标志成UNUSED的列。

删除标志成UNUSED的列：

ALTER TABLE 表名 DROP UNUSED COLUMNS;

【训练2】 将“图书”表的“出版日期”列置成UNUSED，并查看。

步骤1：设置“出版日期”列为UNUSED：

```
ALTER TABLE 图书 SET UNUSED COLUMN 出版日期;
```

步骤2：显示结构：

```
DESC 图书;
```

执行结果：

```
名称                                      是否为空? 类型
----------------------------------------- -------- ----------
---------------
```

图书编号	NOT NULL VARCHAR2(5)
图书名称	NOT NULL VARCHAR2(30)
出版社编号	NOT NULL VARCHAR2(2)
作者	VARCHAR2(10)
数量	NUMBER(3)
单价	NUMBER(7,2)

步骤3：删除UNUSED列：

```
ALTER TABLE 图书 DROP UNUSED COLUMNS;
```

执行结果：

```
表已更改。
```

4.3.4 约束条件的修改

可以为表增加或删除表级约束条件。

1. 增加约束条件

增加约束条件的语法如下：

ALTER TABLE 表名 ADD [CONSTRAINT 约束名] 表级约束条件;

【训练1】 为emp表的mgr列增加外键约束：

```
ALTER TABLE emp ADD CONSTRAINT FK_3 FOREIGN KEY(mgr)
REFERENCES emp(empno);
```

执行结果：

表已更改。

说明：本训练增加的外键为参照自身的外键，含义是mgr(经理编号)列的内容必须是empno(雇员编号)之一。

2．删除约束条件

删除约束条件的语法如下：

ALTER TABLE 表名

DROP PRIMARY_KEY|UNIQUE(列名)|CONSTRAINT 约束名 [CASCADE];

【训练2】 删除为emp表的mgr列增加的外键约束：

ALTER TABLE emp DROP CONSTRAINT FK_3;

执行结果：

表已更改。

4.4 分区表简介

4.4.1 分区的作用

在某些场合会使用非常大的表，比如人口信息统计表。如果一个表很大，就会降低查询的速度，并增加管理的难度。一旦发生磁盘损坏，可能整个表的数据就会丢失，恢复比较困难。根据这一情况，可以创建分区表，把一个大表分成几个区(小段)，对数据的操作和管理都可以针对分区进行，这样就可以提高数据库的运行效率。分区可以存在于不同的表空间上，提高了数据的可用性。

分区的依据可以是一列或多列的值，这一列或多列称为分区关键字或分区列。

所有分区的逻辑属性是一样的(列名、数据类型、约束条件等)，但每个分区可以有自己的物理属性(表空间、存储参数等)。

分区有三种：范围分区、哈斯分区和混合分区。

范围分区(RANGE PARTITIONING)：根据分区关键字值的范围建立分区。比如，根据省份为人口数据表建立分区。

哈斯分区(HASH PARTITIONING)：在分区列上使用HASH算法进行分区。

混合分区(COMPOSITE PARTITIONING)：混合以上两种方法，使用范围分区建立主分区，使用HASH算法建立子分区。

4.4.2 分区的实例

由于分区用到了很多存储参数，故不在这里进行详细讨论，只给出一个范围分区的简单训练实例。

【训练1】 创建和使用分区表。

步骤1：创建按成绩分区的考生表，共分为3个区：

```
CREATE TABLE 考生 (
考号 VARCHAR2(5),
姓名 VARCHAR2(30),
成绩 NUMBER(3)
)
```

```
PARTITION BY RANGE(成绩)
(PARTITION A VALUES LESS THAN (300)
TABLESPACE USERS,
PARTITION B VALUES LESS THAN (500)
TABLESPACE USERS,
PARTITION C VALUES LESS THAN (MAXVALUE)
TABLESPACE USERS
);
```

步骤2：插入不同成绩的若干考生：

```
INSERT INTO 考生 VALUES('10001','王明',280);
INSERT INTO 考生 VALUES('10002','李亮',730);
INSERT INTO 考生 VALUES('10003','赵成',550);
INSERT INTO 考生 VALUES('10004','黄凯',490);
INSERT INTO 考生 VALUES('10005','马新',360);
INSERT INTO 考生 VALUES('10006','杨丽',670);
```

步骤3：检查A区中的考生：

```
SELECT * FROM 考生 PARTITION(A);
```

执行结果：

```
考号  姓名                    成绩
------- -------------------- ------------------------------
10001 王明                    280
```

步骤4：检查全部的考生：

```
SELECT * FROM 考生;
```

执行结果：

```
考号  姓名                    成绩
------- -------------------- ------------------------------
```

10001 王明 280

10004 黄凯 490

10005 马新 360

10002 李亮 730

10003 赵成 550

10006 杨丽 670

说明：共创建A、B、C三个区，A区的分数范围为300分以下，B区的分数范围为300至500分，C区的分数范围为500分以上。共插入6名考生，插入时根据考生分数将自动插入不同的区。

4.5 视图创建和操作

4.5.1 视图的概念

视图是基于一张表或多张表或另外一个视图的逻辑表。视图不同于表，视图本身不包含任何数据。表是实际独立存在的实体，是用于存储数据的基本结构。而视图只是一种定义，对应一个查询语句。视图的数据都来自于某些表，这些表被称为基表。通过视图来查看表，就像是从不同的角度来观察一个(或多个)表。

视图有如下一些优点：

* 可以提高数据访问的安全性，通过视图往往只可以访问数据库中表的特定部分，限制了用户访问表的全部行和列。

* 简化了对数据的查询，隐藏了查询的复杂性。视图的数据来自一个复杂的查询，用户对视图的检索却很简单。

* 一个视图可以检索多张表的数据，因此用户通过访问一个视图，可完成对多个表的访问。

* 视图是相同数据的不同表示，通过为不同的用户创建同一个表的不同视图，使用户可分别访问同一个表的不同部分。

视图可以在表能够使用的任何地方使用，但在对视图的操作上同表相比有些限制，特别是插入和修改操作。对视图的操作将传递到基表，所以在表上定义的约束条件和触发器在视图上将同样起作用。

4.5.2 视图的创建

创建视图需要CREAE VIEW系统权限，视图的创建语法如下：

CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW 视图名[(别名1[，别名2...])]

AS 子查询

[WITH CHECK OPTION [CONSTRAINT 约束名]]

[WITH READ ONLY]

其中：

OR REPLACE 表示替代已经存在的视图。

FORCE表示不管基表是否存在，创建视图。

NOFORCE表示只有基表存在时，才创建视图，是默认值。

别名是为子查询中选中的列新定义的名字，替代查询表中原有的列名。

子查询是一个用于定义视图的SELECT查询语句，可以包含连接、分组及子查询。

WITH CHECK OPTION表示进行视图插入或修改时必须满足子查询的约束条件。后面的约束名是该约束条件的名字。

WITH READ ONLY 表示视图是只读的。

删除视图的语法如下：

DROP VIEW 视图名;

删除视图者需要是视图的建立者或者拥有DROP ANY VIEW权限。视图的删除不影响基表，不会丢失数据。

1．创建简单视图

【训练1】 创建图书作者视图。

步骤1：创建图书作者视图：

```
CREATE VIEW 图书作者(书名,作者)
AS SELECT 图书名称,作者 FROM 图书;
```

输出结果：

视图已建立。

步骤2：查询视图全部内容

```
SELECT * FROM 图书作者;
```

输出结果：

```
书名                          作者
----------------------------- ------------------
```

计算机原理 刘勇

C语言程序设计 马丽

汇编语言程序设计 黄海明

步骤3：查询部分视图：

SELECT 作者 FROM 图书作者;

输出结果：

作者

刘勇

马丽

黄海明

说明：本训练创建的视图名称为“图书作者”，视图只包含两列，为“书名”和“作者”，对应图书表的“图书名称”和“作者”两列。如果省略了视图名称后面的列名，则视图会采用和表一样的列名。对视图查询和对表查询一样，但通过视图最多只能看到表的两列，可见视图隐藏了表的部分内容。

【训练2】 创建清华大学出版社的图书视图。

步骤1：创建清华大学出版社的图书视图：

```
CREATE VIEW 清华图书
AS SELECT 图书名称,作者,单价 FROM 图书 WHERE 出版社编号= '01';
```

执行结果：

```
视图已建立。
```

步骤2：查询图书视图：

```
SELECT * FROM 清华图书;
```

执行结果：

```
图书名称                  作者          单价
```

---------- --------- ----------

计算机原理 刘勇 25.3

步骤3：删除视图：

DROP VIEW 清华图书;

执行结果：

视图已丢掉。

说明：该视图包含了对记录的约束条件。

【练习1】创建部门30的雇员名称和职务的视图，并查询。

【练习2】创建职务为“MANAGER”的雇员名称和工资的视图，并查询。

2．创建复杂视图

【训练3】 修改作者视图，加入出版社名称。

步骤1：重建图书作者视图：

```
CREATE OR REPLACE VIEW 图书作者(书名,作者,出版社)
AS SELECT 图书名称,作者,出版社名称 FROM 图书,出版社
WHERE 图书.出版社编号=出版社.编号;
```

输出结果：

```
视图已建立。
```

步骤2：查询新视图内容：

SELECT * FROM 图书作者;

输出结果：

书名 作者 出版社

---------- --------- ----------

计算机原理 刘勇 清华大学出版社

C语言程序设计 马丽 电子科技大学出版社

汇编语言程序设计 黄海明 电子科技大学出版社

说明：本训练中，使用了OR REPLACE选项，使新的视图替代了同名的原有视图，同时在查询中使用了相等连接，使得视图的列来自于两个不同的基表。

【训练4】 创建一个统计视图。

步骤1：创建emp表的一个统计视图：

```
CREATE VIEW 统计表(部门名,最大工资,最小工资,平均工资)
AS SELECT DNAME,MAX(SAL),MIN(SAL),AVG(SAL) FROM EMP E,DEPT D
WHERE E.DEPTNO=D.DEPTNO GROUP BY DNAME;
```

执行结果：

视图已建立。

步骤2：查询统计表：

SELECT * FROM 统计表;

执行结果：

部门名　　最大工资　最小工资　平均工资

---------- ---------- ---------- ----------

ACCOUNTING　　5000　　1300　　3050

RESEARCH　　3000　　800　　2175

SALES　　2850　　950　1566.66667

说明：本训练中，使用了分组查询和连接查询作为视图的子查询，每次查询该视图都可以得到统计结果。

3．创建只读视图

创建只读视图要用WITH READ ONLY选项。

【训练5】 创建只读视图。

步骤1：创建emp表的经理视图：

```
CREATE OR REPLACE VIEW manager
AS SELECT * FROM emp WHERE job= 'MANAGER'
WITH READ ONLY;
```

执行结果：

```
视图已建立。
```

步骤2：进行删除：

DELETE FROM manager;

执行结果：

ERROR 位于第 1 行:

ORA-01752: 不能从没有一个键值保存表的视图中删除

4．创建基表不存在的视图

正常情况下，不能创建错误的视图，特别是当基表还不存在时。但使用FORCE选项就可以在创建基表前先创建视图。创建的视图是无效视图，当访问无效视图时，Oracle将重新编译无效的视图。

【训练6】 使用FORCE选项创建带有错误的视图：

CREATE FORCE VIEW 班干部 AS SELECT * FROM 班级 WHERE 职务 IS NOT NULL;

执行结果：

警告: 创建的视图带有编译错误。

4.5.3 视图的操作

对视图经常进行的操作是查询操作，但也可以在一定条件下对视图进行插入、删除和修改操作。对视图的这些操作最终传递到基表。但是对视图的操作有很多限定。如果视图设置了只读，则对视图只能进行查询，不能进行修改操作。

1．视图的插入

【训练1】 视图插入练习。

步骤1：创建清华大学出版社的图书视图：

```
CREATE OR REPLACE VIEW 清华图书
AS SELECT * FROM 图书 WHERE 出版社编号= '01';
```

执行结果：

视图已建立。

步骤2：插入新图书：

INSERT INTO 清华图书 VALUES('A0005','软件工程','01','冯娟',5,27.3);

执行结果：

已创建 1 行。

步骤3：显示视图：

SELECT * FROM 清华图书;

执行结果：

图书 图书名称 出 作者 数量 单价

-------- ---------- ---------- -------- ---------- ---------- ----------

A0001 计算机原理 01 刘勇 5 25.3

A0005 软件工程 01 冯娟 5 27.3

步骤4：显示基表

SELECT * FROM 图书;

执行结果：

图书 图书名称 出 作者 数量 单价

-------- ---------- ------- ---------- ----------

------ ----------

A0001 计算机原理 01 刘勇 5 25.3

A0002 C语言程序设计 02 马丽 1 18.75

A0003 汇编语言程序设计 02 黄海明 15 20.18

A0005 软件工程 01 冯娟 5 27.3

说明：通过查看视图，可见新图书插入到了视图中。通过查看基表，看到该图书也出现在基表中，说明成功地进行了插入。新图书的出版社编号为“01”，仍然属于“清华大学出版社”。

但是有一个问题，就是如果在“清华图书”的视图中插入其他出版社的图书，结果会怎么样呢？结果是允许插入，但是在视图中看不见，在基表中可以看见，这显然是不合理的。

2．使用WITH CHECK OPTION选项

为了避免上述情况的发生，可以使用WITH CHECK OPTION选项。使用该选项，可以对视图的插入或更新进行限制，即该数据必须满足视图定义中的子查询中的WHERE条件，否则不允许插入或更新。比如“清华图书”视图的WHERE条件是出版社编号要等于“01”(01是清华大学出版社的编号)，所以如果设置了WITH CHECK OPTION选项，那么只有出版社编号为“01”的图书才能通过清华视图进行插入。

【训练2】 使用WITH CHECK OPTION选项限制视图的插入。

步骤1：重建清华大学出版社的图书视图，带WITH CHECK OPTION选项：

```
CREATE OR REPLACE VIEW 清华图书
AS SELECT * FROM 图书 WHERE 出版社编号= '01'
WITH CHECK OPTION;
```

执行结果：

视图已建立。

步骤2：插入新图书：

INSERT INTO 清华图书 VALUES('A0006','Oracle数据库','02','黄河',3,39.8);

执行结果：

ERROR 位于第 1 行:

ORA-01402: 视图 WITH CHECK OPTIDN 违反 where 子句

说明：可见通过设置了WITH CHECK OPTION选项，“02”出版社的图书插入受到了限制。如果修改已有图书的出版社编号情况会如何？答案是将同样受到限制。要是删除视图中已有图书，结果又将怎样呢？答案是可以，因为删除并不违反WHERE条件。

3．来自基表的限制

除了以上的限制，基表本身的限制和约束也必须要考虑。如果生成子查询的语句是一个分组查询，或查询中出现计算列，这时显然不能对表进行插入。另外，主键和NOT NULL列如果没有出现在视图的子查询中，也不能对视图进行插入。在视图中插入的数据，也必须满足基表的约束条件。

【训练3】 基表本身限制视图的插入。

步骤1：重建图书价格视图：

```
CREATE OR REPLACE VIEW 图书价格
AS SELECT 图书名称,单价 FROM 图书;
```

执行结果：

```
视图已建立。
```

步骤2：插入新图书：

INSERT INTO 图书价格 VALUES('Oracle数据库',39.8);

执行结果：

ERROR 位于第 1 行:

ORA-01400: 无法将 NULL 插入 ("SCOTT"."图书"."图书编号")

说明：在视图中没有出现的基表的列，在对视图插入时，自动默认为NULL。该视图只有两列可以插入，其他列将默认为空。插入出错的原因是，在视图中不能插入图书编号，而图书编号是图书表的主键，是必须插入的列，不能为空，这就产生了矛盾。

4.5.4 视图的查看

USER_VIEWS字典中包含了视图的定义。

USER_UPDATABLE_COLUMNS字典包含了哪些列可以更新、插入、删除。

USER_OBJECTS字典中包含了用户的对象。

可以通过DESCRIBE命令查看字典的其他列信息。在这里给出一个训练例子。

【训练1】 查看清华图书视图的定义：

SELECT TEXT FROM USER_VIEWS WHERE VIEW_NAME='清华图书';

执行结果：

TEXT

SELECT 图书名称,作者,单价 FROM 图书 WHERE 出版社编号='01'

【训练2】 查看用户拥有的视图：

```
SELECT object_name FROM user_objects WHERE object_type='VIEW';
```

执行结果：

```
OBJECT_NAME
--------------------------------------------------------------------------------
-------
清华图书
图书作者
```

4.6 阶段训练

【训练1】 创建学生、系部、课程和成绩表，根据需要设置默认值、约束条件、主键和外键。

步骤1：创建系部表，编号为主键，系部名称非空，电话号码惟一：

```
CREATE TABLE 系部(
编号 NUMBER(5) PRIMARY KEY,
系部名 VARCHAR2(20) NOT NULL,
```

```
地址 VARCHAR2(30),
电话 VARCHAR2(15) UNIQUE,
系主任 VARCHAR2(10)
);
```

步骤2：创建学生表，学号为主键，姓名非空，性别只能是男或女，电子邮件包含@并且惟一，系部编号参照系部表的编号：

```
CREATE TABLE 学生 (
学号 VARCHAR2(10) PRIMARY KEY,
姓名 VARCHAR2(10) NOT NULL,
```

```
    性别 VARCHAR2(2) CHECK(性别='男' OR 性别='女'),
    生日 DATE,
    住址 VARCHAR2(30),
    电子邮件 VARCHAR2(20) CHECK(电子邮件 LIKE '%@%') UNIQUE,
    系部编号 NUMBER(5),
    CONSTRAINT FK_XBBH FOREIGN KEY(系部编号) REFERENCES 系部(编号)
    );
```

步骤3：创建课程表，编号为主键，课程名非空，学分为1到5：

```
CREATE TABLE 课程(
编号 NUMBER(5) PRIMARY KEY,
课程名 VARCHAR2(30) NOT NULL,
学分 NUMBER(1) CHECK(学分>0 AND 学分<=5)
);
```

步骤4：创建成绩表，学号和课程编号为主键，学号参照学生表的学号，课程编号参照课程表的编号：

```
CREATE TABLE 成绩(
学号 VARCHAR2(10),
课程编号 NUMBER(5),
成绩 NUMBER (3),
CONSTRAINT PK PRIMARY KEY(学号,课程编号),
```

CONSTRAINT FK_XH FOREIGN KEY(学号) REFERENCES 学生(学号),

CONSTRAINT FK_KCBH FOREIGN KEY(课程编号) REFERENCES 课程(编号)

);

说明：注意表之间的主从关系，对于系部和学生表，系部表为主表，学生表为子表。学生表的外键表示插入学生的系部编号必须是系部表的编号。对于成绩表，主键是学号和课程编号，表示如果学号相同课程编号必须不同，这样就可以惟一地标识记录。课程表有两个外键，分别参照学生表和课程表，表示成绩表的学号必须是学生表的学号，成绩表的课程编号必须是课程表的编号。

【练习1】向表中插入数据，保证满足约束条件。

4.7 练 习

1. 创建表时，用来说明字段默认值的是：

A. CHECK　　B. CONSTRAINT

C. DEFAULT　　D. UNIQUE

2. 表的主键特点中，说法错误的是：

A. 一个表只能定义一个主键

B. 主键可以定义在表级或列级

C. 主键的每一列都必须非空

D. 主键的每一列都必须惟一

2. 表的主键特点中，说法错误的是：

A. 一个表只能定义一个主键

B. 主键可以定义在表级或列级

C. 主键的每一列都必须非空

D. 主键的每一列都必须惟一

3. 建立外键时添加ON DELETE CASCADE从句的作用是：

A. 删除子表的记录，主表相关记录一同删除

B. 删除主表的记录，子表相关记录一同删除

C. 子表相关记录存在，不能删除主表记录

D. 主表相关记录存在，不能删除子表记录

4. 下面有关表和视图的叙述中错误的是：

A. 视图的数据可以来自多个表

B. 对视图的数据修改最终传递到基表

C. 基表不存在，不能创建视图

D. 删除视图不会影响基表的数据

5. 以下类型的视图中，有可能进行数据修改的视图是：

A. 带WITH READ ONLY选项的视图

B. 子查询中包含分组统计查询的视图

C. 子查询中包含计算列的视图

D. 带WITH CHECK OPTION选项的视图