第 3 章 数据库定义与更新

SQL 语言由 4 部分组成：数据定义语言 DDL、数据操纵语言 DML、数据控制语言 DCL 和其他，其功能如下：

(1) 数据定义语言(Data Definition Language，DDL)：主要用于定义数据库的逻辑结构，包括定义数据库、基本表、视图和索引等，扩展的 DDL 还包括存储过程、函数、对象、触发器等的定义。

(2) 数据操纵语言(Data Manipulation Language，DML)：主要用于对数据库中的数据进行检索和更新两大类操作，其中更新操作包括插入、删除和修改数据。

(3) 数据控制语言(Data Control Language，DCL)：主要用于对数据库中的对象进行授权、用户维护(包括创建、修改和删除)、完整性规则定义和事务定义等。

(4) 其他：主要是嵌入式 SQL 语言和动态 SQL 语言的定义，规定了 SQL 语言在宿主语言中使用的规则。扩展的 SQL 还包括数据库数据的重新组织、备份与恢复等。

3.1 相 关 知 识

在 SQL Server 2019 中，数据库对象包括表、视图、触发器、存储过程、规则、默认值、用户自定义的数据类型等。

SQL Server 的 DDL 是指用来定义和管理数据库以及数据库中的各种对象的语句，这些语句包括 CREATE、ALTER 和 DROP 等语句。

SQL Server 的 DML 是指用来查询、添加、修改和删除数据库中数据的语句，这些语句包括 SELECT、INSERT、UPDATE、DELETE 等。在默认情况下，只有 sysadmin、dbcreator、db_Owner 或 db_Datawriter 等角色的成员才有权利执行数据操纵语言。

3.1.1 数据库定义语句

本节主要讨论数据库的定义功能。

1. 创建数据库

创建数据库的语法如下：

```
CREATE DATABASE database_name
    [ON [PRIMARY]]
          ( [NAME = logical_file_name, ]
```

```
            FILENAME = 'os_file_name'
            [, SIZE = size]
            [, MAXSIZE = {max_size | UNLIMITED} ]
            [, FILEGROWTH = growth_increment]) [, …n])
    [LOG ON]
            (  [NAME = logical_file_name, ]
               FILENAME = 'os_file_name'
               [, SIZE = size]
               [, MAXSIZE = {max_size | UNLIMITED} ]
               [, FILEGROWTH = growth_increment]) [, …n])
```

其中，

- database_name：被创建的数据库的名字。
- ON：用于指定存储数据库中数据的磁盘文件，除 PRIMARY 文件组外，用户可定义用户的文件组及相关的用户文件。
- PRIMARY：描述在主文件组中定义的相关文件，所有的数据库系统表存放在 PRIMARY 文件组中，同时也存放没有分配具体文件组的对象。在主文件组中第一个文件称为主文件，通常包括数据库的系统表。对于一个数据库来说，只能有一个 PRIMARY 文件组。如果主文件组没有指明，则创建数据库时所描述的第一个文件将作为主文件组成员。
- LOG ON：用来指明存储数据库日志的磁盘文件。如果没有指定 LOG ON，系统将自动创建单个的日志文件，使用系统默认的命名方法。

创建数据库的注意事项：

① 默认情况下，只有系统管理员可以创建新数据库，但是系统管理员可以通过授权将创建数据库的权限授予其他用户。

② 数据库名字必须遵循 SQL Server 命名规范：

- 字符的长度可以为 1～30。
- 名称的第一个字符必须是一个字母或者是下列字符中的某一个：下画线“_”或符号@。
- 在首字母后的字符可以是字母、数字或者前面规则中提到的符号。
- 名称中不能有空格。

③ 所有的新数据库都是 model 数据库的副本，新数据库不可能比 model 数据库当前的容量更小。

④ 单个数据库可以存储在单个文件上，也可以跨越多个文件存储。

⑤ 数据库的大小可以被扩展或者收缩。

⑥ 当新的数据库创建时，SQL Server 自动地更新 master 数据库的 sysdatabases 系统表。

2. 修改数据库

创建数据库后如果想对其定义进行修改，例如增删数据文件、增删文件组等，可以使用 ALTER DATABASE 语句处理。

修改数据库的语法如下：

```
ALTER DATABASE database_name
{
      ADD FILE <filespec>[, …n] [TO FILEGROUP filegroup_name]
    | ADD LOG FILE <filespec>[, …n]
    | REMOVE FILE logical_file_name
    | ADD FILEGROUP filegroup_name
    | REMOVE FILEGROUP filegroup_name
    | MODIFY FILE <filespec>
    | MODIFY FILEGROUP filegroup_name filegroup_property
}
```

其中，

- database_Name：被修改的数据库的名字。
- ADD FILE：指定添加到数据库中的数据文件。
- TO FILEGROUP filegroup_name：指定文件添加到文件组名为 filegroup_name 的文件组。
- ADD LOG FILE：指定添加到数据库中的日志文件。
- REMOVE FILE：从数据库系统表中删除该文件，并且物理删除该文件。
- ADD FILEGROUP：指定添加到数据库的文件组。
- filegroup_name：文件组名。
- REMOVE FILEGROUP：从数据库中删除该文件组，并删除在这个文件组中的文件。
- MODIFY FILE：指定要修改的文件。包含该文件的名称、大小、增长量和最大容量。

注意：一次只可以修改其中的一个选项。

3. 删除数据库

删除数据库的语法如下：

```
DROP DATABASE database_name
```

删除数据库将删除数据库所使用的数据库文件和磁盘文件。

3.1.2 表定义语句

本节主要讨论关系表的定义功能。

1. 创建表

创建表的语法如下：

```
CREATE TABLE <tableName>
    ( <columnName1><dataType>[DEFAULT <defaultValue>] [NULL | NOT NULL] [,
    <columnName2><dataType>[DEFAULT <defaultValue>] [NULL | NOT NULL] … ]
    [, [CONSTRAINT <constraintName1>] {UNIQUE | PRIMARY KEY}
          (<columnName1>[, <columnName2>… ]) [, … n ] ]
    [, [CONSTRAINT <constraintName2>]
          FOREIGN KEY (<columnName1>[, <columnName2>… ])
```

```
        REFERENCE [<dbName>.owner.]<refTable>
            (<refColumn1>[, <refColumn2>… ]) [, … n ] ]
    ) [ON <filegroupName>]
```

其中，

- table Name：新表的名称，表名必须符合标识符规则。
- column Name：表中的列名，列名必须符合标识符规则，并且在表内唯一。
- dateType：列的数据类型。
- default＜defaultValue＞：为列设置默认值，属于可选项。
- NULL | NOT NULL：为列设置是否允许为空值，属于可选项。
- ＜constraintName＞：定义约束的名字，属于可选项。
- UNIQUE：建立唯一索引。
- PRIMARY KEY：建立主码。
- FOREIGN KEY：建立外码。
- ON filegroupName：指定该表属于哪个文件组。

2. 修改表结构

修改表结构的语法如下：

```
ALTER TABLE [database_owner].table_name
          (ADD column_name datatype,
              ......
              ADD CONSTRAINT …,
              DROP CONSTRAINT ...,
              REPLACE column_name DEFAULT expression
)
```

3. 删除表

删除表的语法如下：

```
DROP TABLE table_name
```

3.1.3　索引与视图定义语句

本节主要讨论索引、视图的定义功能。

1. 创建视图

在创建视图前需考虑如下原则。

(1) 只能在当前数据库中创建视图。

(2) 视图名称必须遵循标识符的规则，且对每个用户必须唯一，该名称不得与该用户拥有的任何表的名称相同。

(3) 可以在其他视图上建立视图。

(4) 不能将规则或 DEFAULT 定义与视图相关联。

(5) 定义视图的查询不可以包含 ORDER BY、COMPUTE 或 COMPUTE BY 子句或 INTO 关键字。

（6）不能在视图上定义全文索引。

（7）不能创建临时视图，也不能在临时表上创建视图。

（8）下列情况下必须在视图中指定每列的名称：

① 视图中有任何从算术表达式、内置函数或常量派生出的列。

② 视图中两列或多列具有相同名称。

③ 希望使视图中的列名与它的源列名不同，可在视图中重新命名列。无论重命名与否，视图列都会继承其源列的数据类型。

创建视图的语法如下：

```
CREATE VIEW [<database_name>.][<owner>.] view_name [(column [, …n])]
    [ WITH <view_attribute>[, …n] ]
AS
    select_statement
    [ WITH CHECK OPTION ]
    <view_attribute>::={ encryption | schemabinding | view_metadata }
```

其中，

- view_name：视图的名称，视图名称必须符合标识符规则。
- column：视图中的列名。当列是从算术表达式、函数或常量派生的，或两个或更多的列可能会具有相同的名称（如连接），或视图中的某列被赋予了不同于派生来源列的名称时必须指定列名。如果未指定 column，则视图列将获得与 SELECT 语句中的列相同的名称。
- n：表示可以指定多列的占位符。
- select_statement：定义视图的 SELECT 语句。
- WITH CHECK OPTION：表示当对视图进行更新操作时必须满足视图定义的谓词条件。

2. 修改视图

尽量不要对视图进行更新操作，同时注意以下方面：

① 若建立视图时用了连接和分组，或 DISTICT，或内部函数则不能对视图进行 INSERT、UPDATE 和 DELETE 操作。

② 若视图中的列直接由基本表得到，而不是由 price * 10 这样的表达式组成的列可执行 UPDATE 操作。

修改视图的语法如下：

```
ALTER VIEW [<database_name>.][<owner>.] view_name [(column [, …n]) ]
    [ WITH <view_attribute>[, …n] ]
AS
    select_statement
    [ WITH CHECK OPTION ]
```

3. 删除视图

如果不需要某视图，可以删除该视图。删除视图后，视图所基于的数据并不受到影响。

删除视图的语法如下：

```
DROP VIEW view_name [, …n]
```

4. 创建索引

当为表建立主键和唯一约束时，SQL Server 自动创建唯一索引。如果表中不存在聚集索引，则为主键创建一个唯一的聚集索引。默认情况下对 UNIQUE 约束创建唯一的非聚集索引。

创建索引时须考虑的事项是：

① 只有表的所有者可以在同一个表中创建索引。

② 每个表只能创建一个聚集索引。

③ 每个表可以创建的非聚集索引最多为 249 个(包括 PRIMARY KEY 或 UNIQUE 约束创建的索引)。

④ 包含索引的所有长度固定列的最大大小为 900 字节。

⑤ 包含同一索引的列的最大数目为 16。

创建索引的语法如下：

```
CREATE [UNIQUE] [CLUSTERED | NONCLUSTERED]
INDEX index_name
ON {TABLE | VIEW} (column [ASC | DESC] [, …n])
    [ON filegroup]
```

其中，

- UNIQUE：为表或视图创建唯一索引，聚集索引必须是 UNIQUE 索引。
- CLUSTERED：创建聚集索引，如果没有指定 CLUSTERED，则创建非聚集索引。
- NONCLUSTERED：创建非聚集索引。
- index_name：索引名，索引名必须遵循标识符规则。
- TABLE：要创建索引的表。
- VIEW：要建立索引的视图的名称。
- column：应用索引的列。
- ON filegroup：在给定的 filegroup 上创建指定的索引。该文件组必须已经通过执行 CREATE DATABASE 或 ALTER DATABASE 创建。

5. 删除索引

删除索引的语法如下：

```
DROP INDEX index[, …n] ON <tableName| viewName >
```

或者，

```
DROP INDEX tableName.index | viewName.index [, …n]
```

其中，

- tableName.index | viewName.index：要删除的表或视图的索引名称。
- n：表示可以指定多个索引的占位符。

- ON< tableName| viewName >：指定表名或视图名。

3.1.4 表记录更新语句

DML 语句包括查询、插入、修改和删除数据库中的数据等操纵语句，即 SELECT、INSERT、UPDATE、DELETE 等。本节主要讨论数据库对象的 INSERT、UPDATE、DELETE 功能。

1. 插入数据

插入数据的语法如下：

```
INSERT [INTO] table_name/view_name [(column_list)]
    VALUES {DEFAULT | NULL | expression}
```

其中，

- table_name/view_name：表名/视图名。
- column_list：由逗号分隔的列名列表，用来指定为其提供数据的列。如果没有指定 column_list，表中的所有列都将接收数据。

没有包含在 column_list 的列，将在该列插入一个 NULL 值（或者该列定义的默认值）。

由于 SQL Serve 为以下类型的列自动生成值，INSERT 语句将不为这些类型的列指定值：

① 具有 identity 属性的列，该属性为列生成值。

② 有默认值的列，该列用 newid 函数生成一个唯一的 guid 值。

③ 计算列。

所提供的数据值必须与列的列表匹配。数据值的数目必须与列数相同。

2. 修改数据

修改数据的语法如下：

```
UPDATE table_name/view_name
SET column_name =expression | DEFAULT | NULL
[ FROM <table_source>[, …n] ]
[ WHERE <search_condition>]
```

其中，

- table_name/view_name：需要更新的表/视图的名称。
- column_name：要更改数据的列名。
- expression：返回的值将替换 column_name 的现有值。
- DEFAULT：指定使用对列定义的默认值替换列中的现有值。
- FROM <table_source>：指定用表来为更新操作提供准则。
- WHERE <search_condition>：指定条件来限定所更新的行。

3. 删除数据

删除数据的语法如下：

```
DELETE FROM <table_name/view_name>
[WHERE <search_condition>]
```

其中，

- table_name/view_name：要删除记录的表名/视图名。
- WHERE <search_condition>：指出被删除的记录所满足的条件，若省略，表示删除表中的所有记录。

关于视图的操作(INSERT、DELETE、UPDATE)，应注意以下几个问题：

① 若建立视图时用了连接和分组，DISTINCT 或内部函数则不能对视图进行 INSERT 和 DELETE 操作。

② 若视图中的列直接由基本表得到，而不是计算列就可执行 UPDATE 操作。

③ 对视图插入元组时应注意对 NOT NULL 字段的处理。

④ 若视图由多表连接而成，对视图插入元组时应分别对同一张表中的字段插入元组。

⑤ 尽量不要对视图进行更新操作。

3.2　实验七：数据库与数据表定义

3.2.1　实验目的与要求

(1) 掌握数据库的建立、删除和修改操作。

(2) 理解基本表之间的关系，掌握表结构的建立、修改和删除操作，创建模式导航图。

3.2.2　实验案例

1. 数据库创建与删除

【例 3.1】 创建一个 myorder 数据库，该数据库的主要文件为 myorder.mdf，事务日志为 myorderLog.ldf，它们都位于 e:\mySQL 目录下。

SQL 语句如下：

```
CREATE DATABASE myorder
ON
    (  NAME='myorder',
       FILENAME='e:\mySQL\myorder.mdf',
       SIZE=3,
       MAXSIZE=50,
       FILEGROWTH=1 )
LOG ON
    (  NAME='myorderLog',
       FILENAME='e:\mySQL\myorderLog.ldf',
       SIZE=3,
       MAXSIZE=20,
       FILEGROWTH=1 )
```

本例中，myorder 数据库只有一个主逻辑设备，对应一个物理文件 myorder.mdf，该文件初始大小 3MB，最大可扩展为 50MB，如果初始文件装不下数据，自动按 1MB 进行扩

展，直到50MB为止。日志文件为myorderLog.ldf，该文件初始大小3MB，最大可扩展为20MB，如果初始文件装不下数据，自动按1MB进行扩展。

【例3.2】 建立一个复杂的数据库tmyorder。

SQL语句如下：

```
CREATE DATABASE tmyorder
ON PRIMARY
    (  NAME='tmyorder',
       FILENAME='e:\mySQL\tmyorder.mdf',
       SIZE=100,
       MAXSIZE=300,
       FILEGROWTH=1%),
    (  NAME='tmyorder2',
       FILENAME='e:\temp\tmyorder2.ndf',
       SIZE=50,
       FILEGROWTH=2 ),
    (  NAME='tmyorder3',
       FILENAME='e:\temp\ tmyorder3.ndf',
       SIZE=50,
       FILEGROWTH=2 ),
  FILEGROUP temorder
    (  NAME='temorder',
       FILENAME='e:\temp\temorder.mdf',
       SIZE=6,
       MAXSIZE=10,
       FILEGROWTH=2 )
   LOG ON
   (  NAME='tmyorderLog',
      FILENAME='e:\mySQL\tmyorderLog.ldf',
      SIZE=100,
      MAXSIZE=500,
      FILEGROWTH=2%)
```

在本例中，该数据库由4个数据文件和1个日志文件组成。主设备有1个主要文件tmyorder.mdf和2个次要文件tmyorder2、tmyorder3组成，用户设备有1个文件temorder.mdf，日志有1个文件tmyorderLog.ldf。

【例3.3】 删除数据库tmyorder。

SQL语句如下：

```
DROP DATABASE tmyorder
```

2. 创建表

【例3.4】 创建一张客户表(客户编号、客户姓名、客户电话、客户地址、邮政编码)。

SQL语句如下：

```
CREATE TABLE Customer (
    customerNo      char(9)        NOT NULL  PRIMARY KEY,      /*客户编号*/
    customerName    varchar(40)    NOT NULL,                   /*客户名称*/
    telephone       varchar(20)    NOT NULL,                   /*客户电话*/
    address         char(40)       NOT NULL,                   /*客户住址*/
    zip             char(6)        NULL                        /*邮政编码*/
```

```
)
```

【例 3.5】　建立一张员工表(员工编号、员工姓名、员工性别、员工生日、员工住址、员工电话、雇用日期、所属部门、职称、薪水)。

SQL 语句如下：

```
CREATE TABLE Employee (
    employeeNo      char(8)         NOT NULL  PRIMARY KEY,   /*员工编号*/
    employeeName    varchar(10)     NOT NULL,                /*员工姓名*/
    sex             char(1)         NOT NULL,                /*员工性别*/
    birthday        datetime        NULL,                    /*员工生日*/
    address         varchar(50)     NULL,                    /*员工住址*/
    telephone       varchar(20)     NULL,                    /*员工电话*/
    hireDate        datetime        NOT NULL,                /*雇用日期*/
    department      varchar(30)     NOT NULL,                /*所属部门*/
    title           varchar(6)      NOT NULL,                /*职称*/
    salary          numeric(8,2)    NOT NULL                 /*薪水*/
)
```

【例 3.6】　建立一张订单表(订单编号、客户编号、业务员编号、订货日期、订单金额、发票号码),要求给该表建立主键约束和关于员工表和客户表的外键约束。

SQL 语句如下：

```
CREATE TABLE OrderMaster (
    orderNo         char(12)      NOT NULL  PRIMARY KEY,  /*订单编号*/
    customerNo      char(9)       NOT NULL,               /*客户编号*/
    salerNo         char(8)       NOT NULL,               /*业务员编号*/
    orderDate       datetime      NOT NULL,               /*订货日期*/
    orderSum        numeric(9,2)  NOT NULL,               /*订单金额*/
    invoiceNo       char(10)      NOT NULL,               /*发票号码*/
    CONSTRAINT OrdermasterFK1 FOREIGN KEY(customerNo)
    REFERENCES Customer(customerNo),
    CONSTRAINT OrdermasterFK2 FOREIGN KEY(salerNo)
    REFERENCES Employee(employeeNo)
)
```

3.2.3　实验内容

(1) 创建一个 BookDB 数据库,要求至少有一个数据文件和一个日志文件。

(2) 创建图书借阅管理相关 5 张关系表,表结构如表 3-1～表 3-5 所示。

表 3-1　图书分类表 BookClass

属 性 名	类　型	空 值 约 束	属 性 含 义
classNo	char(4)	NOT NULL	图书分类号
className	varchar(20)	NOT NULL	图书分类名称

表 3-2 图书表 Book

属性名	类型	空值约束	属性含义
bookNo	char(10)	NOT NULL	图书编号
classNo	char(4)	NOT NULL	分类号
bookName	varchar(40)	NOT NULL	图书名称
authorName	varchar(8)	NOT NULL	作者姓名
publisherNo	char(4)	NOT NULL	出版社号
price	numeric(7，2)	NULL	单价
publishingDate	datetime	NULL	出版日期
shopDate	datetime	NULL	入库时间
shopNum	numeric(3)	NULL	入库数量

表 3-3 读者表 Reader

属性名	类型	空值约束	属性含义
readerNo	char(8)	NOT NULL	读者编号
readerName	varchar(8)	NOT NULL	姓名
sex	char(2)	NULL	性别
identifycard	char(18)	NULL	身份证号
workUnit	varchar(50)	NULL	工作单位
borrowCount	tinyint	NULL	读者最大可借书数量

表 3-4 出版社表 Publisher

属性名	类型	空值约束	属性含义
publisherNo	char(4)	NOT NULL	出版社编号
publisherName	varchar(20)	NOT NULL	出版社名称

表 3-5 借阅表 Borrow

属性名	类型	空值约束	属性含义
readerNo	char(8)	NOT NULL	读者编号
bookNo	char(10)	NOT NULL	图书编号
borrowDate	datetime	NOT NULL	借阅日期
shouldDate	datetime	NOT NULL	应归还日期
returnDate	datetime	NULL	归还日期

3.3　实验八：索引与视图定义

3.3.1　实验目的与要求

（1）掌握索引的建立和删除操作。

（2）掌握视图的创建和查询操作。

3.3.2　实验案例

1. 创建索引

【例 3.7】　在员工表中按生日建立一个非聚簇索引 birthdayIdx。

SQL 语句如下：

```
CREATE NONCLUSTERED INDEX birthdayIdx ON Employee(birthday)
```

【例 3.8】　在订单主表中，首先按订单金额的降序，然后按客户编号的升序建立一个非聚簇索引 sumcustIdx。

SQL 语句如下：

```
CREATE INDEX sumcustIdx ON OrderMaster(orderSum DESC, customerNo)
```

【例 3.9】　在订单主表中按发票号码创建一个唯一性索引 uniqincoiceIdx。

SQL 语句如下：

```
CREATE UNIQUE INDEX uniqincoiceIdx ON OrderMaster(invoiceno)
```

【例 3.10】　删除 birthdayIdx 索引。

SQL 语句如下：

```
DROP INDEX birthdayIdx ON Employee
```

2. 定义视图

【例 3.11】　建立一个女员工的视图，要求显示员工编号、姓名、性别和薪水。

SQL 语句如下：

```
CREATE VIEW emp_view
AS
  SELECT employeeNo, employeeName, sex, salary
  FROM Employee
  WHERE sex='f'
```

【例 3.12】　创建一个视图，要求查询每个员工的订单号、员工编号、员工姓名、订单金额、发票号码等信息。

SQL 语句如下：

```
CREATE VIEW emp_ordermast
AS
  SELECT orderNo, employeeNo, employeeName, orderSum, invoiceno
```

```
  FROM Employee, OrderMaster
  WHERE employeeNo=salerNo
```

【例 3.13】 修改 emp_view 视图，要求视图只显示薪水 3000 元以上的女员工信息。SQL 语句如下：

```
ALTER VIEW emp_view
AS
  SELECT employeeNo, employeeName, sex, salary
  FROM Employee
  WHERE sex='f' AND salary>3000
```

【例 3.14】 删除视图 emp_view。

SQL 语句如下：

```
DROP VIEW emp_view
```

3.3.3 实验内容

（1）根据基本表创建以下索引：

- 在图书表中按出版社号建立一个非聚集索引 PublishingnoIdx。
- 在读者表中按身份证号建立一个非聚集索引 IdentifycardIdx。
- 在读者表中，首先按工作单位的升序，然后按最大借书数量降序建立一个非聚集索引 WorkunitCountIdx。

（2）创建一个图书名称中含有“数据”的图书视图 BookView。

（3）创建一个包含读者编号、读者姓名、图书编号、图书名称、借阅日期、归还日期的视图 BorrowView。

（4）创建一个视图，要求显示至少借阅了 3 本书的读者信息 ReaderView。

（5）在视图 BorrowView 中查询 2016 年 3 月 1 日以前借阅的图书。

（6）在视图 ReaderView 中查询姓张的读者信息。

（7）在视图 BorrowView 基础上再建一个只包含“合生元有限公司”的读者所借图书信息的视图 BorrowView1。

（8）删除视图 BorrowView。

3.4 实验九：数据更新操作

3.4.1 实验目的与要求

（1）掌握基本表的 INSERT、UPDATE、DELETE 操作。

（2）正确理解更新操作中涉及的相关约束问题。

3.4.2 实验案例

【例 3.15】 在客户表中插入一条信息（C20220004，双良股份有限公司，0510-

3566021,江阴市,214400)。

SQL 语句如下：

```
INSERT Customer VALUES('C20220004', '双良股份有限公司', '0510-3566021', '江阴市
', '214400')
```

【例 3.16】　删除 1987 年以前出生的员工记录。

SQL 语句如下：

```
DELETE FROM Employee
WHERE year(Birthday)<1987
```

【例 3.17】　删除 E2020002 业务员的订单明细信息。

SQL 语句如下：

```
DELETE FROM OrderDetail
WHERE orderNo IN (
      SELECT orderNo
      FROM OrderMaster
      WHERE salerNo='E2020002' )
```

【例 3.18】　在客户表中把 C20220003 客户的客户名称改为西湖商厦,电话改为 021-6800000。

SQL 语句如下：

```
UPDATE Customer
SET customerName='西湖商厦',Telephone='021-6800000'
WHERE customerNo='C20220003'
```

【例 3.19】　在 OrderMaster 表中找出 E2020003 业务员的订单,将这些订单对应的每一项销售商品的单价打 8 折。

SQL 语句如下：

```
UPDATE OrderDetail
SET price=price*0.8
WHERE orderNo IN (
       SELECT orderNo
       FROM OrderMaster
       WHERE salerNo='E2020003' )
```

3.4.3　实验内容

根据 BookDB 中 5 张关系表,完成以下更新操作：

(1) 分别给这 5 张表添加信息,要求在图书分类表、图书表、出版社表、读者表中各插入 5 个元组,在借阅表中插入 20 个元组。

(2) 将“合生元有限公司”的读者工作单位修改为“联合立华股份有限公司”。

(3) 将入库数量最多的图书单价下调 5%。

(4) 将“经济类”的图书单价提高 10%。

（5）将借阅次数高于2次的图书数量增加50%。

（6）将“兴隆股份有限公司”读者的借书期限延长为3个月。

（7）将至少借了20次书且每次正常还书的读者的最大可借图书数量增加5。

（8）删除价格超过30元的图书借阅信息。

（9）删除借阅了大学英语的借阅记录。

（10）删除从未借过书的读者。