

【例 2.1】创建一个 **uname** 用户定义数据类型，其基于的系统数据类型是变长为 8 的字符，不允许空。

```
Use Northwind
Exec sp_addtype uname,'Varchar(8)', 'Not Null '
```

【例 2.2】创建一个用户定义的数据类型 **birthday**，其基于的系统数据类型是 **DateTime**，允许空。

```
Use Northwind
Exec sp_addtype birthday,datetime,'Null'
```

【例 2.3】创建两个数据类型 **telephone** 和 **fax**，为电话及传真号码设置专门的数据类型。

```
Use Sales
Exec sp_addtype telephone,'varchar(24) ','Not Null'
Exec sp_addtype fax,'varchar(24)', 'Null'
```

【例 2.4】删除用户定义数据类型 **uname**

```
Use Northwind
Exec sp_droptype 'uname '
```

【例 3.1】使用企业管理器创建一个数据库。名字为 **Sales**，数据文件名为 **Sales_Data.Mdf**，存储在 **E:**下，初始大小为 2MB，最大为 10MB，文件增量以 1MB 增长，事务日志文件名为 **Sales_Log.Ldf**，存储在在 **E:**下，初始大小为 1MB，最大为 5MB，文件增量以 1MB 增长。

- (1) 展开服务器，右击“数据库”，在弹出的菜单中选择“新建数据库”命令。
- (2) 单击数据库属性窗口的“常规”选项卡，在“名称”栏输入销售数据库的名字 **Sales**，结果如图 3.3 所示。



图 3.3 Sales 数据库属性窗口

(3) 单击数据库属性窗口的“数据文件”选项卡，在文件名和位置栏输入文件名及其存放的位置，也可以通过单击“...”按钮后进行修改。本例采用系统默认的数据文件名 Sales_Data，将路径修改为“E: \”，将数据文件的初始大小修改为 2MB。

(4) 文件组采用系统给出的默认值 PRIMARY，它表示当前的这个数据文件是在主文件组中。

(5) 数据文件增长采用“按兆字节”，文件增量为 1MB，将文件最大容量限制为 10MB，设置结果如图 3.4 所示。



图 3.4 完成设置后的数据文件属性页

(6)单击数据库属性窗口的“事务日志”选项卡，出现如图 3.5 所示的对话框，事务日志文件名采用系统默认的名字 Sales_Log，将路径修改为“E:\”，将事务日志文件的初始大小修改为 1MB。

(7)事务日志文件增长采用“按兆字节”，文件增量为 1MB，将文件最大容量限制为 5MB。

(8)单击“确定”按钮，完成数据库的创建。

【例 3.2】在查询分析器中使用 CREATE DATABASE 语句创建一个数据库。名字为 NewSales，数据文件名为 NewSales_Data.Mdf，存储在 E:\下，初始大小为 4MB,最大为 10MB，文件增量以 1MB 增长，事务文件为 NewSales_Log.Ldf，存储在 E:\下，初始大小为 2MB，最大为 5MB，文件增量以 1MB 增长。

(1) 启动查询分析器，输入登录名和登录密码，按“确定”按钮后屏幕出现 SQL 查询分析器窗口。

(2) 在查询分析器中运行以下命令：

```
CREATE DATABASE NewSales
ON PRIMARY
```

```

(NAME = NewSales_Data ,
  FILENAME = 'E:\ NewSales_Data.Mdf',
  SIZE = 4MB,
  MAXSIZE = 10MB,
  FILEGROWTH =1 MB)
LOG ON
(NAME = NewSales_Log ,
  FILENAME = 'E:\NewSales_Log.Ldf',
  SIZE = 2MB,
  MAXSIZE =5MB,
  FILEGROWTH = 1MB)
GO

```

【例 3.3】使用查询分析器创建名为 Mydb 的数据库，它有容量为 12MB、8MB、6MB 的 3 个数据文件，其中 Mydb_Data1.Mdf 是主文件，Mydb_Data2.Ndf、Mydb_Data3.Ndf 是次文件，数据库中有两个容量分别是 6MB、5MB 的事务日志文件，文件名分别为 Mydb_Log1.Ldf、Mydb_Log2.Ldf。数据文件和事务日志文件均存储在 E: \, 最大容量均为 20MB，文件增量均为 2MB。

在查询分析器中运行以下命令：

```

CREATE DATABASE Mydb
ON PRIMARY
( NAME = Mydb_Data1,
  FILENAME ='E:\ Mydb_Data1.Mdf',
  SIZE = 12MB,
  MAXSIZE = 20MB,
  FILEGROWTH = 2MB),
(NAME = Mydb_Data2 ,
  FILENAME ='E:\ Mydb_Data2.Ndf',
  SIZE = 8MB,
  MAXSIZE = 20MB,
  FILEGROWTH = 2MB),
(NAME = Mydb_Data3,
  FILENAME = 'E:\ Mydb_Data3.Ndf',
  SIZE = 6MB,
  MAXSIZE = 20MB,
  FILEGROWTH = 2MB)
LOG ON
(NAME = Mydb_Log1,
  FILENAME = 'E:\ Mydb_Log1.Ldf',
  SIZE = 6MB,

```

```

MAXSIZE = 20MB,
FILEGROWTH = 2MB),
(NAME = Mydb_Log2,
FILENAME = 'E:\ Mydb_Log2.Ldf',
SIZE = 5MB,
MAXSIZE = 20MB,
FILEGROWTH = 2MB)
GO

```

【例 3.4】在查询分析器中使用存储过程 SP_HELPDB 显示数据库 Sales 的信息。

在查询分析器中运行如下命令：

```
SP_HELPDB Sales
```

【例 3.5】在查询分析器中使用存储过程 SP_HELPDB 显示 SQL Server 上所有数据库的信息。

在查询分析器中运行如下命令：

```
SP_HELPDB
```

【例 3.6】使用企业管理器将销售数据库 Sales 的数据文件 Sales_Data 由原来的 2MB 扩充为 4MB；增加一个次文件 Sales_Data.Ndf, 存储在 E:\, 初始容量为 2MB, 最大容量为 10MB, 文件增量为 1MB；事务日志文件由原来的 1MB 扩充为 2MB。

(1) 展开服务器。

(2) 右击 Sales 数据库，在弹出的菜单中选择“属性”命令。

(3) 单击“属性”窗口的“数据文件”选项卡，将数据文件 Sales_Data 分配的空间修改为 4MB。单击下一个空行，输入增加的次文件名 Sales_1_Data.Ndf, 位置为“E:\”，分配的空间为 2MB，文件增长采用“按兆字节”，将文件最大限制为 10MB，文件增量为 1MB。

(4) 单击“属性”窗口的“事务日志”选项卡，将事务日志文件 Sales_Log 分配的空间修改为 2MB

(5) 单击“确定”按钮，完成数据库的扩充。

【例 3.7】使用查询分析器将 NewSales 数据库中的数据文件 NewSales_Data 由原来的 4MB 扩充为 8MB，事务日志文件 NewSales_Log 由原来的 2MB 扩充为 4MB。

在查询分析器中运行如下命令：

```

USE MASTER
GO
ALTER DATABASE NewSales
MODIFY FILE (NAME='NewSales_Data', SIZE=8MB)
GO
ALTER DATABASE NewSales
MODIFY FILE (NAME='NewSales_Log', SIZE=4MB)
GO

```

【例 3.8】使用企业管理器将 NewSales 数据库文件 NewSales_Data 收缩为 4MB。

(1)展开服务器。

(2)右击 Sales 数据库，在弹出菜单中选择“所在任务”，然后选择“收缩数据库”。

(3)在图 3.13 中单击“文件”按钮。

选择“收缩操作”下方的“收缩文件至”，输入“4MB”。

(4)单击“确定”按钮，屏幕显示“收缩数据文件顺利完成”对话框。

(5)单击“确定”按钮，屏幕显示“数据库已顺利收缩”对话框。再单击“确定”按钮，结束收缩数据库操作。

【例 3.9】 将数据库 Sales 的大小压缩到原来的 20%。

在查询分析器中运行如下命令：

```
USE Sales
GO
DBCC SHRINKDATABASE (Sales,20)
GO
```

【例 3.10】将数据库 NewSales 中的数据文件 NewSales_Data 的大小由原来的 8MB 压缩为 4MB。

在查询分析器中运行如下命令：

```
USE NewSales
GO
DBCC SHRINKFILE (NewSales_Data,4)
GO
```

【例 3.11】 显示数据库可以进行重新设置的选项。

在查询分析器中运行如下命令：

```
SP_DBOPTION
```

【例 3.12】 将 NewSales 数据库设置为只读。

在查询分析器中运行如下命令：

```
USE Sales
GO
SP_DBOPTION 'NewSales','read only','true'
GO
```

【例 3.13】 将 Sales 数据库设置为单用户方式。

在查询分析器中运行如下命令：

```
USE Sales
GO
SP_DBOPTION 'Sales','single user','true'
GO
```

【例 3.14】 将 Sales 数据库名字修改为 MySales。

在查询分析器中运行如下命令：

```
SP_RENAMEDB 'Sales','MySales'
```

【例 3.15】 删除 MySales 数据库。

在查询分析器中运行如下命令：

```
DROP DATABASE MySales
```

```
GO
```

【例 3. 16】使用企业管理器在 Sales 数据库中创建 Employees 数据表。Employees 数据表的结构如表 3.1 所示。

表 3.1 Employees 数据表的数据结构

列名	数据类型	是否为空
编号	Char(6)	No
姓名	Char(8)	No
性别	Bit	No
部门	Varchar(16)	Yes
电话	Varchar(20)	Yes
地址	Varchar(50)	Yes

下面使用企业管理器创建 Employees 数据表。

- (1) 展开 Sales 数据库。
- (2) 右击“表”，在弹出的菜单中选择“新建表”命令，屏幕显示表设计器对话框。
- (3) 在表设计器对话框中，根据表 3.1 输入对应的列名、数据类型、长度和是否为空，结果如图 3.19 所示。

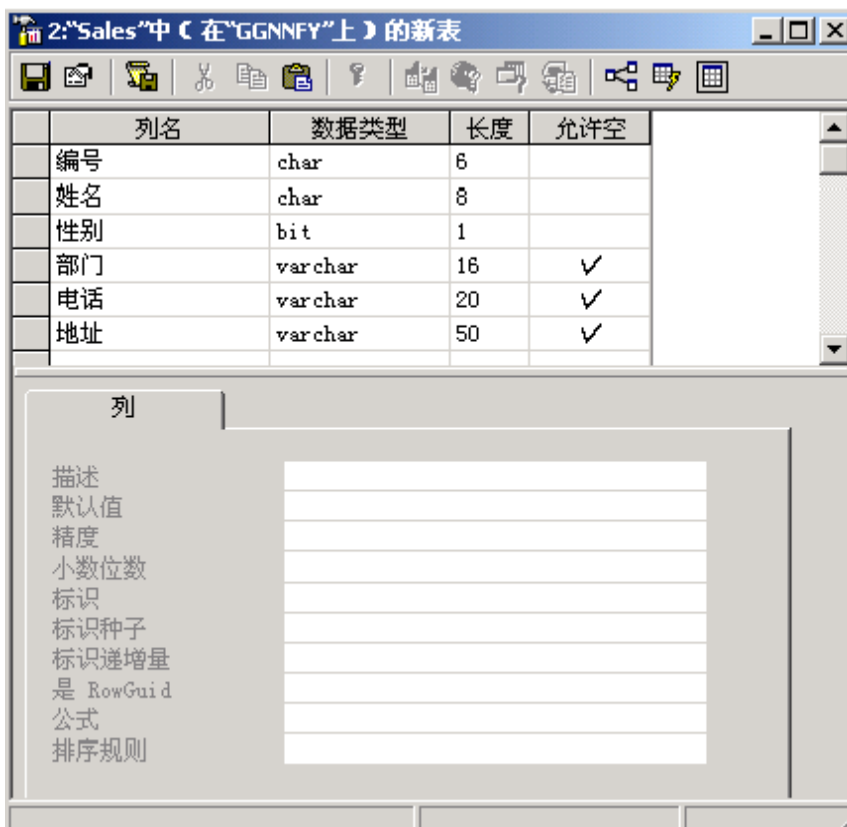


图 3.19 设置 Employees 数据表

(4) 单击工具栏中的“存盘”按钮，在弹出的对话框中输入数据表名“Employees”，按“确定”按钮，即可完成 Employees 数据表的创建。

【例 3. 17】使用查询分析器在 Sales 数据库中创建数据表 Goods、Sell。数据表 Goods、Sell 的数据结构分别如表 3. 2、3. 3 所示。

表 3. 2 数据表 Goods 的数据结构

列名	数据类型	是否为空
商品编号	Int	No
商品名称	Varchar (20)	No
生产厂商	Varchar (30)	No
(续表) 列名	数据类型	是否为空
进货价	Money	No
零售价	Money	No
数量	Int	No
进货时间	DateTime	No
进货员工编号	Varchar (20)	No

表 3. 3 数据表 Sell 的数据结构

列名	数据类型	是否为空
销售编号	Int	No
商品编号	Int	No
数量	Int	No
售出时间	DateTime	No
售货员工编号	Char (6)	No

在查询分析器中运行如下命令：

```
USE Sales
GO
--创建进货表 Goods
CREATE TABLE Goods
( 商品编号 Int NOT NULL,
  商品名称 Varchar (20) NOT NULL,
  生产厂商 Varchar (20) NOT NULL,
  进货价 Money NOT NULL,
  零售价 Money NOT NULL,
  数量 Int NOT NULL,
  进货时间 DateTime NOT NULL,
  进货员工编号 Char (6) NOT NULL
)
```

```

GO
--创建已售商品表 Sell
CREATE TABLE Sell
( 销售编号 Int NOT NULL,
  商品编号 Int NOT NULL ,
  数量 Int NOT NULL,
  售出时间 DateTime NOT NULL,
  售货员工编号 Char(6) NOT NULL
)
GO

```

【例 3.18】 分别使用企业管理器、查询分析器在 Sales 数据库中为 Employees 数据表创建名为 IX_EmployeesNo 的 PRIMARY KEY 约束，以保证不会出现编号相同的员工。

使用企业管理器：

- (1) 展开 Sales 数据库，单击“表”。
- (2) 在详细列表中右击“Employees”，在弹出菜单中选择“设计表”命令。
- (3) 选择“属性”对话框中的“索引/键”选项卡，单击“新建”按钮。
- (4) 在“索引名”栏中为索引命名。这里输入“IX_EmployeesNo”。
- (5) 在“列名”列表中选择“编号”列，按“升序”排列。
- (6) 选中“创建 UNIQUE”。
- (7) 选中“约束”表示创建 PRIMARY KEY 约束。最后设置结果如图 3.20 所示。
- (8) 完成设置后单击“关闭”按钮。
- (9) 单击工具栏上的“保存”按钮。

使用查询分析器：

在查询分析器中输入如下命令：

```

USE Sales
GO
ALTER TABLE Employees
ADD CONSTRAINT IX_EmployeesNo PRIMARY KEY (编号)
GO

```

【例 3.19】 分别使用企业管理器、查询分析器删除例 3.18 建立的名为 IX_EmployeesNo 的 PRIMARY KEY 约束。

使用企业管理器：

- (1) 展开 Sales 数据库，单击“表”。
- (2) 在详细列表中右击“Employees”，在弹出菜单中选择“设计表”命令。
- (3) 单击工具栏上的“表和索引属性”按钮。
- (4) 选择“属性”对话框中的“索引/键”选项卡，在“选定的索引”下拉列表中选择 IX_EmployeesNo。
- (5) 单击“删除”按钮。
- (6) 单击“关闭”按钮。

(7)单击工具栏上的“保存”按钮。

使用查询分析器：

在查询分析器中输入如下命令：

```
USE Sales
GO
ALTER TABLE Employees
DROP CONSTRAINT IX_EmployeesNo
GO
```

【例 3.20】分别使用企业管理器、查询分析器在 Sales 数据库中为 Goods 数据表创建名为 FK_Goods_ Employees1 的 FOREIGN KEY 约束，该约束限制“进货员工编号”列的数据只能是 Employees 数据表“编号”列中存在的数。

使用企业管理器查询分析器：

- (1)展开 Sales 数据库，单击“表”。
- (2)在详细列表中右击“Goods”，在弹出菜单中选择“设计表”命令。
- (3)单击工具栏上的“表和索引属性”按钮。
- (4)选择“属性”对话框中的“关系”选项卡，单击“新建”按钮。
- (5)在“外键表”下拉列表中选择“Goods”表。
- (6)在“外键表”下拉列表下方的字段列表中选择“进货员工编号”。
- (7)在“主键表”下拉列表中选择“Employees”表。
- (8)在“主键表”下拉列表下方的字段列表中选择“编号”。
- (9)在“关系名”栏中为关系命名。这里输入“FK_Goods_ Employees1”。
- (10)设置结果如图 3.21 所示。单击“关闭”按钮。

图 3.21 创建 FOREIGN KEY 约束

(11)单击工具栏上的“保存”按钮。

使用查询分析器：

在查询分析器中输入如下命令：

```
USE Sales
GO
ALTER TABLE Goods
ADD CONSTRAINT FK_Goods_ Employees1 FOREIGN KEY (进货员工编号)
REFERENCES Employees(编号)
GO
```

单击“运行”按钮，返回“命令已成功完成”的信息，说明已成功创建 FOREIGN KEY 约束。

【例 3.21】分别使用企业管理器、查询分析器在 Sales 数据库中删除例 3.20 建立的名为 FK_Goods_ Employees1 的 FOREIGN KEY 约束。

使用企业管理器：

- (1) 展开 Sales 数据库，单击“表”。
- (2) 在详细列表中右击“Goods”，在弹出菜单中选择“设计表”命令。
- (3) 单击工具栏上的“表和索引属性”按钮。
- (4) 选择“属性”对话框中的“关系”选项卡。
- (5) 在“选定的关系”下拉列表中选择“FK_Goods_Employees1”。
- (6) 单击“删除”按钮，删除关系。
- (7) 单击“关闭”按钮。
- (8) 单击工具栏上的“保存”按钮。

使用查询分析器：

在查询分析器中输入如下命令：

```
USE Sales
GO
ALTER TABLE Goods
DROP CONSTRAINT FK_Goods_Employees1
GO
```

【例 3. 22】 分别使用企业管理器、查询分析器在 Sales 数据库中为 Employees 数据表创建名为 IX_EmployeesTeNo 的 UNIQUE 约束，以保证列名为“电话”的取值不相同。

使用企业管理器：

- (1) 展开 Sales 数据库，单击“表”。
- (2) 在详细列表中右击“Employees”，在弹出菜单中选择“设计表”命令。
- (3) 选择“属性”对话框中的“索引/键”选项卡，单击“新建”按钮。
- (4) 选中“创建 UNIQUE”和“约束”。
- (5) 在“索引名”栏中为索引命名。这里输入“IX_EmployeesTeNo”。
- (6) 在“列名”列表中选择“电话”列，按“升序”排列。
- (7) 单击“关闭”按钮。
- (8) 单击工具栏上的“保存”按钮。

使用查询分析器：

在查询分析器中输入如下命令：

```
USE Sales
GO
ALTER TABLE Employees
ADD CONSTRAINT IX_EmployeesTeNo UNIQUE(电话)
GO
```

【例 3. 23】 分别使用企业管理器、查询分析器在 Sales 数据库中删除例 3. 22 建立的名为 IX_EmployeesTeNo 的 UNIQUE 约束。

使用企业管理器：

- (1) 展开 Sales 数据库，单击“表”。
- (2) 在详细列表中右击“Employees”，在弹出菜单中选择“设计表”命令。
- (3) 单击工具栏上的“表和索引属性”按钮。

- (4) 选择“属性”对话框中的“索引/键”选项卡。
- (5) 在“选定的索引”下拉列表中选择“IX_EmployeesTeNo”。
- (6) 单击“删除”按钮。
- (7) 单击“关闭”按钮。
- (8) 单击工具栏上的“保存”按钮。

使用查询分析器：

在查询分析器中输入如下命令：

```
USE Sales
GO
ALTER TABLE Employees
DROP CONSTRAINT IX_EmployeesTeNo
GO
```

【例 3. 24】 分别使用企业管理器、查询分析器在 Sales 数据库中为 Employees 数据表创建名为 CK_EmployeesNo 的 CHECK 约束，该约束限制“编号”列中只允许 6 位数字（不能为字母）。

使用企业管理器：

- (1) 展开 Sales 数据库，单击“表”。
- (2) 在详细列表中右击“Employees”，在弹出菜单中选择“设计表”命令。
- (3) 选择“属性”对话框中的“CHECK 约束”选项卡，单击“新建”按钮。
- (4) 在“约束表达式”栏中输入“编号 like ‘[0-9][0-9][0-9][0-9][0-9][0-9]’”。
- (5) 在“约束名”栏中为约束命名，这里输入“CK_EmployeesNo”。
- (6) 完成设置后结果如图 3. 23 所示，单击“关闭”按钮。
- (7) 单击工具栏上的“保存”按钮。

使用查询分析器：

在查询分析器中输入如下命令：

```
USE Sales
GO
ALTER TABLE Employees
ADD CONSTRAINT CK_EmployeesNo CHECK (编号
like ' [0-9][0-9][0-9][0-9][0-9][0-9]')
GO
```

【例 3. 25】 分别使用企业管理器、查询分析器在 Sales 数据库中删除例 3. 24 建立的名为 CK_EmployeesNo 的 CHECK 约束。

使用企业管理器：

- (1) 展开 Sales 数据库，单击“表”。
- (2) 在详细列表中右击“Employees”，在弹出菜单中选择“设计表”命令。
- (3) 单击工具栏上的“表和索引属性”按钮。
- (4) 选择“属性”对话框中的“CHECK 约束”选项卡。

(5)在“选定的约束”下拉列表中选择“CK_EmployeesNo”。

(6)单击“删除”按钮。

(7)单击“关闭”按钮。

(8)单击工具栏上的“保存”按钮。

使用查询分析器：

在查询分析器中输入如下命令：

```
USE Sales
GO

ALTER TABLE Employees
DROP CONSTRAINT CK_EmployeesNo
GO
```

【例 3. 26】分别使用企业管理器、查询分析器在 Sales 数据库中为 Goods 数据表创建名为 DF_GoodsDate 的 DEFAULT 约束，该约束使“进货时间”列的默认值为当前的日期。

使用企业管理器：

(1) 展开 Sales 数据库，单击“表”。

(2) 在详细列表中右击“Goods”，在弹出菜单中选择“设计表”命令。

(3)将光标定位在“进货时间”行。

(4)在“列”属性页下的“默认值”编辑框中输入“GETDATE（）”。

图 3.24 创建 DEFAULT 约束

(5)单击工具栏上的“保存”按钮。

使用查询分析器：

在查询分析器中输入如下命令：

```
USE Sales
GO

ALTER TABLE Goods
ADD CONSTRAINT DF_GoodsDate DEFAULT(GETDATE()) FOR 进货时间
```

单击“运行”按钮，返回“命令已成功完成”的信息，说明已成功创建 DEFAULT 约束。

【例 3. 27】分别使用企业管理器、查询分析器在 Sales 数据库中删除例 3. 26 建立的名为 DF_GoodsTime DEFAULT 约束。

使用企业管理器：

(1)展开 Sales 数据库，单击“表”。

(2)在详细列表中右击“Goods”，在弹出菜单中选择“设计表”命令。

(3)选中“进货时间”列。

(4)清除“列”属性页下的“默认值”编辑框的值。

(5)单击工具栏上的“保存”按钮。

使用查询分析器：

在查询分析器中输入如下命令：

```
USE Sales
GO
ALTER TABLE Goods
DROP CONSTRAINT DF_GoodsDate
GO
```

【例 3. 28】使用查询分析器给 Sales 数据库的 Employees 数据表增加一列，列名为：邮箱，数据类型：VarChar(20)，并将该列设置为惟一约束。

在查询分析器中输入如下命令：

```
USE Sales
GO
ALTER TABLE Employees
ADD COLUMN 邮箱 VarChar(20) NULL CONSTRAINT U_Email UNIQUE
GO
```

【例 3. 29】使用查询分析器删除例 3. 28 给 Employees 数据表增加的列。
在查询分析器中输入如下命令：

```
USE Sales
GO
--首先删除约束
ALTER TABLE Employees
DROP CONSTRAINT U_Email
GO
--然后删除列
ALTER TABLE Employees
DROP COLUMN 邮箱
GO
```

【例 3. 30】使用查询分析器将数据表 Goods 重新命名为 MyGoods。
在查询分析器中输入如下命令：

```
USE Sales
GO
EXEC SP_RENAME 'Goods', 'MyGoods'
GO
```

【例 3. 31】使用查询分析器删除数据表 MyGoods。
在查询分析器中输入如下命令：

```
USE Sales
GO
DROP TABLE MyGoods
GO
```

【例 3. 32】使用 INSERT 语句向数据表 Employees 插入一条记录。

在查询分析器中输入如下命令：

```
USE Sales
GO
INSERT INTO Employees (编号, 姓名, 性别)
VALUES ('000016', '李明', 1)
GO
```

【例 3. 33】使用 INSERT 语句将数据表 Employees1 中性别=1 的记录插入数据表 Employees(数据表 Employees1 的结构与 Employees 完全相同)。

在查询分析器中输入如下命令：

```
USE Sales
GO
INSERT INTO Employees
SELECT *
FROM Employees1
WHERE Employees1. 性别=1
GO
```

【例 3. 34】使用 SELECT INTO 语句生成一张新数据表，新数据表的名称为 Employees2，数据来自于 Employees 数据表中性别=1 的编号，姓名，性别等字段。

在查询分析器中输入如下命令：

```
USE Sales
GO
SELECT 编号, 姓名, 性别
INTO Employees2
FROM Employees WHERE 性别=1
GO
```

【例 3. 35】使用 UPDATE 语句将数据表 Employees 中编号为“000006”记录的电话更改为 0771-3836386。

在查询分析器中输入如下命令：

```
USE Sales
GO
UPDATE Employees
SET 电话='0771-3836386'
WHERE 编号='000006'
GO
```

【例 3. 36】使用 UPDAT 语句将数据表 Goods 中李明 2005 年 5 月 20 日进货的商品零售价调整为九五折。

在查询分析器中输入如下命令：

```
USE Sales
```

```
GO
UPDATE Goods
SET Goods. 零售价= Goods. 零售价*0.95
FROM Goods, Employees
WHERE Goods. 进货时间='2005-05-20' AND Employees. 姓名='李明'
AND Employees. 编号= Goods. 进货员工编号
GO
```

【例 3. 37】使用 DELETE 语句删除数据表 Sell 中售出时间为 1995 年 1 月 1 日以前的记录。

在查询分析器中输入如下命令：

```
USE Sales
GO
DELETE Sell where 售出时间 < '1995-01-01'
GO
```

【例 3. 38】使用 DELETE 语句删除数据表 Goods 中李明 1995 年 1 月 1 日以前的进货记录。

在查询分析器中输入如下命令：

```
USE Sales
GO
DELETE Goods
FROM Goods, Employees
WHERE Goods. 进货时间 < '1995-01-01' AND Employees. 姓名='李明'
AND Employees. 编号= Goods. 进货员工编号
GO
```

【例 3. 39】使用企业管理器分离数据库 Sales。

(1)展开服务器，展开数据库。

(2)右击 Sales 数据库，在弹出菜单中选择“所有任务”下的“分离数据库”。

(3)系统检查数据库的连接状态，如果存在使用本数据库的连接，单击“清除”按钮断开这些连接。

(4)如果数据库的状态为“该数据库已就绪，可以分离。”，单击“确定”按钮，出现分离数据库顺利完成对话框，再单击“确定”按钮，结束分离操作。

【例 3. 40】使用企业管理器将例 3. 39 分离出来的数据库 Sales 附加到另一台计算机。

在附加数据库前，将例 3. 39 分离出来的数据库文件 Sales_Data. Mdf、Sales_Log. Ldf 复制到另一台计算机的 E:\。

(1)展开服务器。

(2)右击数据库，在弹出菜单中选择“所有任务”下的“附加数据库”。

(3)输入要附加的数据文件名。也可以单击“”按钮，从图 3-3-2 所示的对话框中选择数据文件名，单击“确定”按钮。

(4)在“附加为”框内输入数据库的名称（默认为分离时的数据库名），输入的数据

库名不能与本计算机上任何现有的数据库名称相同。

(5)在指定数据库所有者下拉列表框中选择数据库的所有者，设置如图 3.27 所示。

(6)单击“确定”按钮，出现附加数据库顺利完成对话框。单击“确定”按钮结束附加工作。

【例 3. 41】使用查询分析器分离数据库 NewSales，分离后不进行更新统计。

在查询分析器中运行如下命令：

```
EXEC SP_DETACH_DB 'NewSales', 'true'
```

【例 3. 42】使用查询分析器将例 3. 43 分离出来的数据库 NewSales 附加到另一台计算机。

在附加数据库前，将例 3. 41 分离出来的数据库文件 NewSales_Data.Mdf、NewSales_Log.Ldf 复制到另一台计算机的 E:\。然后在查询分析器中运行如下命令：

```
CREATE DATABASE NewSales
ON PRIMARY
(NAME= NewSales_Data ,
FILENAME='E:\NewSales_Data.Mdf')
LOG ON
(NAME=NewSales_Log,
FILENAME='E:\NewSales_Log.Ldf')
FOR ATTACH
GO
```

【例 4. 1】 查询员工表中所有员工的姓名和联系电话，可以写为：

```
SELECT 姓名,电话 FROM employees
```

程序执行结果为：

姓名	电话
赵飞燕	01032198454
刘德发	01032298726
李建国	01032147588
李圆圆	01032358697
刘金武	01032298726
万兴国	01032658325
孟全	01058546230
黎美丽	01058964357
冯晓丹	01036571568
王峰	01032987564
陈吉轩	01058796545

（所影响的行数为 11 行）

【例 4. 2】 查询员工表中的所有记录，程序为：

```
SELECT * FROM employees
```

程序执行结果为：

编号	姓名	性别	部门	电话	地址
1001	赵飞燕	0	采购部	01032198454	北京市南京东路 55 号
1002	刘德发	1	采购部	01032298726	北京市建国路 101 号
1003	李建国	1	采购部	01032147588	北京市民主路 6 号
1101	李圆圆	0	财务部	01032358697	北京市仁爱路一巷 41 号
1102	刘金武	1	财务部	01032298726	北京市建国路 101 号
1103	万兴国	1	财务部	01032658325	北京市南大街南巷 250 号
1201	孟全	1	库存部	01058546230	北京市南大街南巷 115 号
1202	黎美丽	0	库存部	01058964357	北京市教育路 32 号
1301	冯晓丹	0	销售部	01036571568	北京市育才路 78 号
1302	王峰	1	销售部	01032987564	北京市沿江路 123 号
1303	陈吉轩	1	销售部	01058796545	北京市德外大街 19 号

（所影响的行数为 11 行）

【例 4.3】 查询进货表中的所有的生产厂商，去掉重复值，程序为：

```
SELECT DISTINCT 生产厂商 FROM goods
```

程序执行结果为：

生产厂商
dell 公司
TCL 公司
惠普公司
佳能公司
联想公司
三星公司
索尼公司

（所影响的行数为 7 行）

【例 4.4】 查询进货表中商品名称、单价和数量的前 4 条记录，程序为：

```
SELECT TOP 4 商品名称, 进货价, 数量 FROM goods
```

程序执行结果为：

商品名称	进货价	数量
打印机	1205.0000	10
液晶显示器	2210.0000	12
数码相机	2380.0000	8
扫描仪	998.0000	16

（所影响的行数为 4 行）

【例 4.5】 使用列的别名，查询员工表中所有记录的员工编号（别名为 **number**），姓名（别名为 **name**）和电话（别名为 **telephone**），程序为：

```
SELECT 编号 AS number, name=姓名, 电话 telephone FROM employees
```

程序执行结果为：

number	name	telephone
1001	赵飞燕	01032198454
1002	刘德发	01032298726
1003	李建国	01032147588
1101	李圆圆	01032358697
1102	刘金武	01032298726
1103	万兴国	01032658325
1201	孟全	01058546230
1202	黎美丽	01058964357
1301	冯晓丹	01036571568
1302	王峰	01032987564
1303	陈吉轩	01058796545

（所影响的行数为 11 行）

【例 4.6】 查询各件商品的进货总金额，可以写为：

```
SELECT 商品名称, 进货价*数量 AS 总金额 FROM goods
```

程序执行结果为：

商品名称	总金额
打印机	12050.0000
液晶显示器	26520.0000
数码相机	19040.0000
扫描仪	15968.0000
笔记本电脑	156000.0000
MP3 播放器	8244.0000
摄像机	35100.0000
台式电脑	68500.0000
CRT 显示器	4740.0000

（所影响的行数为 9 行）

【例 4.7】 在 **Employees** 表中查询姓名为王峰的员工的联系电话，程序为：

```
SELECT 姓名, 电话 FROM employees AS c WHERE c.姓名='王峰'
```

程序执行结果为：

姓名	电话
王峰	01032987564

（所影响的行数为 1 行）

【例 4.8】 查询笔记本电脑的进货信息，程序为：

```
SELECT * FROM goods WHERE 商品名称='笔记本电脑'
```

【例 4.9】 查询在 2005 年 1 月 1 日以前销售的商品信息，可以写为：

```
SELECT 商品编号, 数量, 售出时间 FROM sell WHERE 售出时间<'2005-1-1'
```

程序执行结果为：

商品编号	数量	售出时间
2	1	2004-10-15 00:00:00.000
2	1	2004-10-16 00:00:00.000
5	2	2004-10-26 00:00:00.000

（所影响的行数为 3 行）

【例 4.10】 查询进货总金额小于 10000 元的商品名称，可以写为：

```
SELECT 商品名称 FROM goods WHERE 进货价*数量<10000
```

程序执行结果为：

商品名称
MP3 播放器
CRT 显示器

（所影响的行数为 2 行）

【例 4.11】 查询 2005 年 1 月 1 日以前进货且进货价大于 1000 元的商品，可以写为：

```
SELECT 商品名称 FROM goods WHERE 进货时间<'2005-1-1' AND 进货价>1000
```

程序执行结果为：

商品名称
打印机

（所影响的行数为 1 行）3. LIKE 关键字

【例 4.12】 查询“李”姓员工的基本信息，可以写为：

```
SELECT * FROM employees WHERE 姓名 LIKE '李%'
```

程序执行结果为：

编号	姓名	性别	部门	电话	地址
1003	李建国	1	采购部	01032147588	北京市民主路 6 号
1101	李圆圆	0	财务部	01032358697	北京市仁爱路一巷 41 号

（所影响的行数为 2 行）

【例 4.13】 查询零售价格在 2000 到 3000 元之间的所有商品，可以写为：

```
SELECT 商品名称, 零售价 FROM goods WHERE 零售价 BETWEEN 2000 AND 3000
```

程序执行结果为：

商品名称	零售价
液晶显示器	2980.0000
数码相机	3000.0000
CRT 显示器	2200.0000

（所影响的行数为 3 行）

【例 4.14】 查询打印机、摄像机的进货价格，程序为：

```
SELECT 商品名称, 进货价  
FROM goods  
WHERE 商品名称 IN ('打印机', '摄像机')
```

程序执行结果为：

商品名称	进货价
打印机	1205.0000
摄像机	5850.0000

（所影响的行数为 2 行）

【例 4.15】 查询电话不为空的员工信息，可以写为：

```
SELECT * FROM employees WHERE 电话 IS NOT NULL
```

【例 4.16】 查询商品的进货价格并按从大到小排序，程序为：

```
SELECT 商品名称, 进货价 FROM goods ORDER BY 进货价 DESC
```

程序执行结果为：

商品名称	进货价
笔记本电脑	7800.0000
台式电脑	6850.0000
摄像机	5850.0000
数码相机	2380.0000
液晶显示器	2210.0000
CRT 显示器	1580.0000
打印机	1205.0000
扫描仪	998.0000
MP3 播放器	458.0000

（所影响的行数为 9 行）

【例 4.17】 按照商品进货数量的升序排序，在同一数量内，将按照进货价的降序排列，程序为：

```
SELECT 商品名称, 进货价, 数量 FROM goods ORDER BY 3, 2 DESC
```

程序执行结果为：

商品名称	进货价	数量
CRT 显示器	1580.0000	3
摄像机	5850.0000	6
数码相机	2380.0000	8
台式电脑	6850.0000	10
打印机	1205.0000	10
液晶显示器	2210.0000	12
扫描仪	998.0000	16
MP3 播放器	458.0000	18
笔记本电脑	7800.0000	20

（所影响的行数为 9 行）

【例 4.18】 使用 INTO 子句创建一个新表，可以写为：

```
SELECT TOP 15 PERCENT 商品名称,进货价*数量 AS 总金额 INTO 金额表
```

FROM goods

【例 4.19】 查询财务部的员工人数，可以写为：

```
SELECT COUNT(*) AS 人数 FROM employees WHERE 部门='财务部'
```

程序执行结果为：

人数
3

（所影响的行数为 1 行）

【例 4.20】 查询商品编号为 2 的的商品的销售数量，可以写为：

```
SELECT SUM(数量) as 销售数量 FROM sell WHERE 商品编号='2'
```

程序执行结果为：

销售数量
2

（所影响的行数为 1 行）

【例 4.21】 统计各部门的人数，可以写为：

```
SELECT 部门,COUNT(*) AS 人数 FROM Employees GROUP BY 部门
```

程序执行结果为：

部门	人数
财务部	3
采购部	3
库存部	2

销售部 3

(所影响的行数为 4 行)

【例 4.22】对员工表按性别统计各部门人数，可以写为：

```
SELECT 性别, 部门, COUNT(部门) FROM Employees GROUP BY 性别, 部门
```

程序执行结果为：

性别	部门	人数
0	财务部	1
1	财务部	2
0	采购部	1
1	采购部	2
0	库存部	1
1	库存部	1
0	销售部	1
1	销售部	2

(所影响的行数为 8 行)

【例 4.23】使用 WITH CUBE，可以写为：

```
SELECT 性别, 部门, COUNT(部门) AS 人数  
FROM Employees GROUP BY 性别, 部门 WITH CUBE
```

程序执行结果为：

性别	部门	人数
0	财务部	1
0	采购部	1
0	库存部	1
0	销售部	1
0	NULL	4
1	财务部	2
1	采购部	2
1	库存部	1
1	销售部	2
1	NULL	7
NULL	NULL	11
NULL	财务部	3
NULL	采购部	3
NULL	库存部	2
NULL	销售部	3

(所影响的行数为 15 行)

【例 4.24】 使用 WITH ROLLUP, 可以写为:

```
SELECT 性别, 部门, COUNT(部门) AS 人数
FROM Employees GROUP BY 性别, 部门 WITH ROLLUP
```

程序执行结果为:

性别	部门	人数
0	财务部	1
0	采购部	1
0	库存部	1
0	销售部	1
0	NULL	4
1	财务部	2
1	采购部	2
1	库存部	1
1	销售部	2
1	NULL	7
NULL	NULL	11

(所影响的行数为 11 行)

【例 4.25】 统计各部门的男性人数, 可以写为:

```
SELECT 性别, 部门, COUNT(部门) AS 人数
FROM Employees GROUP BY 性别, 部门 HAVING 性别='1'
```

程序执行结果为:

性别	部门	人数
1	财务部	2
1	采购部	2
1	库存部	1
1	销售部	2

(所影响的行数为 4 行)

【例 4.26】 统计销售总数量, 可以写为:

```
SELECT 售货员工编号, 商品编号, 数量
FROM sell COMPUTE SUM(数量)
```

程序执行结果为:

售货员工编号	商品编号	数量
1301	2	1
1302	2	1

1303	5	2
1301	6	1
1301	7	2
1302	3	2
1302	3	1
1302	9	1
1303	6	1
1303	6	1
SUM		
13		

（所影响的行数为 11 行）

【例 4.27】 分别统计各员工的销售总数，可以写为：

```
SELECT 售货员工编号, 商品编号, 数量
FROM sell order by 售货员工编号 COMPUTE SUM(数量) BY 售货员工编号
```

程序执行结果为：

售货员工编号	商品编号	数量
1301	2	1
1301	6	1
1301	7	2
SUM		
4		
1302	3	2
1302	3	1
1302	9	1
1302	2	1
SUM		
5		

售货员工编号	商品编号	数量
1303	5	2

1303	6	1
1303	6	1

SUM
4

（所影响的行数为 13 行）

【例 4.28】 使用 ANSI 联接形式（INNER JOIN），程序为：

```
SELECT 销售编号, 商品名称, sell. 数量 as 销售数量
FROM goods INNER JOIN sell ON goods. 商品编号=sell. 商品编号
```

程序执行结果为：

销售编号	商品名称	销售数量
1	打印机	1
2	打印机	1
3	扫描仪	2
4	笔记本电脑	1
5	MP3 播放器	2
6	液晶显示器	2
7	液晶显示器	1
8	台式电脑	1
9	笔记本电脑	1
10	笔记本电脑	1

（所影响的行数为 10 行）

【例 4.29】 使用 SQL Server 联接形式，程序为：

```
SELECT 销售编号, 商品名称, sell. 数量 as 销售数量
FROM goods, sell
WHERE goods. 商品编号=sell. 商品编号
```

【例 4.30】 使用左外联接（LEFT [OUTER] JOIN），程序为：

```
SELECT 销售编号, 商品名称, sell. 数量 as 销售数量
FROM goods LEFT JOIN sell ON goods. 商品编号=sell. 商品编号
```

程序执行结果为：

销售编号	商品名称	销售数量
1	打印机	1
2	打印机	1
6	液晶显示器	2
7	液晶显示器	1

NULL	数码相机	NULL
3	扫描仪	2
4	笔记本电脑	1
9	笔记本电脑	1
10	笔记本电脑	1
5	MP3 播放器	2
NULL	摄像机	NULL
8	台式电脑	1
NULL	CRT 显示器	NULL

（所影响的行数为 13 行）

【例 4.31】 使用 SQL Server 联接形式，程序为：

```
SELECT 销售编号, 商品名称, sell. 数量 as 销售数量
FROM goods, sell
WHERE goods. 商品编号*=sell. 商品编号
```

【例 4.32】 使用左外联接（RIGHT [OUTER] JOIN），程序为：

```
SELECT 销售编号, 商品名称, sell. 数量 as 销售数量
FROM goods RIGHT JOIN sell ON goods. 商品编号=sell. 商品编号
```

程序执行结果为：

销售编号	商品名称	销售数量
1	打印机	1
2	打印机	1
3	扫描仪	2
4	笔记本电脑	1
5	MP3 播放器	2
6	液晶显示器	2
7	液晶显示器	1
8	台式电脑	1
9	笔记本电脑	1
10	笔记本电脑	1

（所影响的行数为 10 行）

【例 4.33】 使用左外联接（FULL [OUTER] JOIN），程序为：

```
SELECT 销售编号, 商品名称, sell. 数量 as 销售数量
FROM goods FULL JOIN sell ON goods. 商品编号=sell. 商品编号
```

程序执行结果为：

销售编号	商品名称	销售数量
------	------	------

1	打印机	1
2	打印机	1
6	液晶显示器	2
7	液晶显示器	1
NULL	数码相机	NULL
3	扫描仪	2
4	笔记本电脑	1
9	笔记本电脑	1
10	笔记本电脑	1
5	MP3 播放器	2
NULL	摄像机	NULL
8	台式电脑	1
NULL	CRT 显示器	NULL

(所影响的行数为 13 行)

【例 4.34】 使用交叉联接 (CROSS JOIN)，程序为：

```
SELECT 销售编号, 商品名称, sell. 数量 as 销售数量
FROM goods CROSS JOIN sell
```

【例 4.35】 联合查询进货员工和销售员工，可以写为：

```
SELECT 售货员工编号 AS 业务员 FROM sell
UNION SELECT 进货员工编号 FROM goods
```

程序执行结果为：

业务员
1001
1002
1003
1301
1302
1303

(所影响的行数为 6 行)

【例 4.36】 查询进货员工的基本信息，可以写为：

```
SELECT * FROM employees
WHERE 编号=ANY (SELECT 进货员工编号 FROM goods)
```

程序执行结果为：

编号	姓名	性别	部门	电话	地址
1001	赵飞燕	0	采购部	01032198454	北京市南京东路 55 号

1002	刘德发	1	采购部	01032298726	北京市建国路 101 号
1003	李建国	1	采购部	01032147588	北京市民主路 6 号

(所影响的行数为 3 行)

【例 5.1】为新建的表创建一个唯一性聚集索引，其语句如下：

```
USE sales
go
CREATE TABLE 生产厂商表 (
    厂商编号 int IDENTITY(1,1) NOT NULL,
    生产厂商 VARCHAR(30) NOT NULL,
    法人代表 VARCHAR(8),
    厂商地址 VARCHAR(50)
)
CREATE UNIQUE CLUSTERED INDEX I_厂商编号
ON 生产厂商表(厂商编号)
```

【例 5.2】为 Employees 表的“姓名”字段创建一个名为“I_姓名”的非聚集索引，使用降序排列，填充因子为 30，其语句如下：

```
USE sales
go
CREATE INDEX I_姓名
    ON Employees (姓名 DESC)
WITH FILLFACTOR=30
```

【例 5.3】为 goods 表的“商品编号”和“商品名称”创建一个复合索引“I_goods”设定各相应参数，其语句如下：

```
USE sales
go
CREATE INDEX I_goods
    ON goods (商品编号, 商品名称)
WITH
    PAD_INDEX,
    FILLFACTOR=50,
    IGNORE_DUP_KEY,
    STATISTICS_NORECOMPUTE
```

【例 5.4】删除 goods 中的索引 I_进货时间和销售表中的索引 I_售出时间，其语句如下：

```
USE sales
go
DROP INDEX goods. I_进货时间, sell. I_售出时间
```

【例 5.5】显示 Products 表的 PK_Products 索引的碎片统计信息。

```
USE Northwind
```

```
DBCC SHOWCONTIG (Products, PK_Products)
```

【例 5.6】整理 Northwind 数据库中 Orders 表的 CustomersOrders 索引碎片。

```
DBCC INDEXDEFRAG (Northwind, Orders, CustomersOrders)
```

【例 5.7】重新创建 Customers 表的 CompanyName 索引并配置新的填充因子

```
USE Northwind
```

```
GO
```

```
CREATE INDEX CompanyName
```

```
ON Customers (CompanyName) WITH DROP_EXISTING, FILLFACTOR=100
```

【例 6.1】创建一个新视图 v1，要求基表选择 goods，sell，employees，来源字段为 sell 表中的销售编号、商品编号和数量；goods 表中的商品名称；employees 表中编号和姓名，要求查询采购部的赵飞燕所采购商品的销售情况，程序为：

```
use sales
```

```
go
```

```
create view v1
```

```
as
```

```
select sell.销售编号,sell.商品编号,sell.数量,
```

```
goods.商品名称,employees.编号,employees.姓名
```

```
from sell,goods,employees
```

```
where goods.商品编号=sell.商品编号
```

```
and goods.进货员工编号=employees.编号
```

```
and employees.姓名='赵飞燕'
```

【例 6.2】创建一个新视图 v2，要求基表选择 goods，sell，employees，来源字段为 sell 表中的销售编号、商品编号和数量；goods 表中的商品名称；employees 表中编号和姓名，要求查询销售部的王峰所销售商品的情况，并对视图的定义进行加密，程序为：

```
use sales
```

```
go
```

```
create view v2
```

```
with encryption
```

```
as
```

```
select sell.销售编号,sell.商品编号,sell.数量,
```

```
goods.商品名称,employees.编号,employees.姓名
```

```
from sell,goods,employees
```

```
where goods.商品编号=sell.商品编号
```

```
and sell.售货员工编号=employees.编号
```

```
and employees.姓名='王峰'
```

【例 6.3】创建一个新视图 v3，要求基表选择 goods，sell，，来源字段为 sell 表中的销售编号、商品编号、数量和售出时间；goods 表中的商品名称、进货价和零售价；再增加一列“该笔销售纯利润”；要求查询该公司 2004 年 10 月份商品的销售情况和每

一笔销售的纯利润，并对视图的定义进行加密，程序为：

```
use sales
go
create view v3
with encryption
as
select sell.销售编号,sell.商品编号,sell.数量,sell.售出时间,
goods.商品名称,goods.进货价,goods.零售价,
(goods.零售价-goods.进货价)*sell.数量 as 该笔销售纯利润
from sell,goods
where goods.商品编号=sell.商品编号
and datepart(yy,sell.售出时间)=2004
and datepart(mm,sell.售出时间)=10
```

【例 6.4】 修改视图 v2，在该视图中增加一个新的限制条件，要求查询王峰所销售的液晶显示器的销售情况，并对视图 v2 取消加密，程序为：

```
use sales
go
alter view v2
as
select sell.销售编号,sell.商品编号,sell.数量,
goods.商品名称,employees.编号,employees.姓名
from sell,goods,employees
where goods.商品编号=sell.商品编号
and sell.售货员工编号=employees.编号
and employees.姓名='王峰'
and goods.商品名称='液晶显示器'
```

【例 6.5】： 查询视图 v1 的基表中的数据，程序为：

```
select * from v1
```

【例 6.6】 创建一个视图 v4，该视图的基表为 employees，要求在视图中显示采购部的所有男员工的详细信息，程序为：

```
use sales
go
create view v4
as
select employees.*
from employees
where employees.性别=1
and employees.部门='采购部'
```

【例 6.7】 创建一个视图 v5，该视图的基表为 employees，要求在视图中显示财务

部的所有男员工的详细信息，程序为：

```
use sales
go
create view v5
as
select employees.*
from employees
where employees.性别=1
and employees.部门='财务部'
with check option
go
```

【例 6.8】 创建一个视图 v6，该视图的基表为 goods，在视图中显示 goods 表中的商品编号，商品名称，生产厂商，进货价和零售价五个字段，要求只显示进货时间在 2004 年的进货情况，程序为：

```
create view v6
as
select 商品编号, 商品名称, 生产厂商, 进货价, 零售价
from goods
where year(进货时间)=2004
```

【例 6.9】 利用视图 v5，删除编号为 1007 的员工的记录，程序为：

```
use sales
go
delete from v5
where 编号=1007
```

【例 7.1】 下面的示例使用行内注释语句来解释正在进行的语句。

```
USE Sales  --选择数据库
GO
--使用 SELECT 语句查询进货时间在 2005 年 1 月的所有商品信息
SELECT *  --选择所有字段
FROM Goods  --选择 Goods 表
WHERE YEAR(进货时间)=2005 and MONTH(进货时间)=1
--查询条件满足进货时间是 2005 年 1 月
GO
```

【例 7.2】 下面的示例使用行内注释语句来解释正在进行的语句。

```
USE Sales  --选择数据库
GO
/* 使用 SELECT 语句查询所有员工的信息
用*表示选择表中的所有字段
没有使用 WHERE 子句则显示表中的全部记录 */
```

```

SELECT * /* 选择所有字段 */
FROM employees /* 选择 employee 表 */
GO
/* 调试程序时在一个 T-SQL 语句内部使用注释，
临时禁止使用“电话”字段 */
SELECT 编号, 姓名, /*电话, */地址
FROM employees
GO

```

【例 7.3】 本例列出一些合法和不合法的标识符：

以下是合法的标识符：

Table1、TABLE1、table1、stu_proc、_abc、@varname、#proc、##temptbl
A\$bc、"Employees"、"Table 1"

以下是不合法的标识符：

Table 1、@@@、SELECT、TABLE

【例 7.4】 假如某公司数据库中有一张北京分公司所有员工的信息表，表名为 Employees In BeiJing，现要查询 Employees In BeiJing 表中的所有员工信息，则查询语句应为：

```
SELECT * FROM "Employees In BeiJing"
```

或者

```
SELECT * FROM [Employees In BeiJing]
```

【例 7.5】 使用 @@ERROR 变量在一个 UPDATE 语句中检测限制检查冲突（错误代码为#547）。

```

USE Sales
GO
--将编号为 1001 的员工所进货品的数量更新为 1100。
UPDATE Goods
SET 数量=1100
WHERE 进货员工编号='1001'
--检查是否出现限制检查冲突
IF @@ERROR=547
PRINT '出现限制检查冲突，请检查需要更新的数据限制'
GO

```

(3) @@SPID: 返回当前用户进程的服务器进程 ID。

【例 7.6】 下面这个程序段返回当前用户进程的 ID、登录名和用户：

```
SELECT @@SPID AS 'ID', SYSTEM_USER AS 'Login Name', USER AS 'User Name'
```

(4) @@TRANCOUNT: 返回事务嵌套的级别。

(5) @@SERVERNAME: 返回本地服务器的名称。

(6) @@VERSION: 返回当前安装的 SQL Server 版本、日期、及处理器类型。

(7) @@IDENTITY: 返回上次 INSERT 操作中插入到 IDENTITY 列的值。

(8) @@LANGUAGE: 返回当前所用语言的名称。

【例 7.7】 本例使用 DECLARE 语句声明一个用于保存计数值的整型变量。

```
DECLARE @cnt int
```

【例 7.8】 本例使用一条 DECLARE 语句同时声明多个变量。

```
DECLARE @empid char(6), @empname char(8), @tel varchar(20)
```

【例 7.9】 声明一个名为 now 的局部变量并赋值，用此变量返回当前系统的日期和时间。

```
--声明两个局部变量  
DECLARE @now datetime  
--对局部变量赋值  
SET @now=GETDATE()  
--显示局部变量的值  
SELECT @now
```

【例 7.10】 本例演示了使用查询给变量赋值的方法。

```
USE Sales  
GO  
DECLARE @cnt int  
SET @cnt=(SELECT count(编号) FROM employees)  
/*使用查询给变量赋值，注意这里的查询语句只能在返回单个值的情况下才能赋值成功*/  
SELECT @cnt AS 公司员工总数 /*将员工总数输出到屏幕*/  
GO
```

本例也可以不用 SET 语句赋值，而使用 SELECT 语句赋值，改写为：

```
USE Sales  
GO  
DECLARE @cnt int  
SELECT @cnt= count(编号) FROM employees  
SELECT @cnt AS 公司员工总数  
GO
```

【例 7.11】 本例演示了局部变量的作用域。

在查询分析器中建立以下两个批处理：

```
DECLARE @msg varchar(50)  
SET @msg= '欢迎您使用 Transact-SQL 语句编程'  
GO  
PRINT @msg  
GO
```

【例 7.12】 求现有进货表中每种商品的总价。

```
USE Sales  
GO
```

```
SELECT 商品编号, 商品名称, 进货价*数量 AS 总价
FROM Goods
```

【例 7.13】 本例用于查询指定名称的商品还有没有货。

```
USE Sales
GO
DECLARE @goodname varchar(20), @num numeric
SET @goodname='HP喷墨打印机 photosmart 7268'
SELECT @num=数量
FROM goods
WHERE 商品名称=@goodname
IF @num>0 PRINT '还有货'
ELSE PRINT '无库存, 请尽快进货!'
```

【例 7.14】 本例演示逻辑运算符 ALL 的使用。查询单笔商品销售量高于王峰最高销售量的员工姓名、所销售的商品名称、售出时间及销售量。

```
USE Sales
GO
SELECT 姓名, 商品名称, 售出时间, Sell. 数量
FROM Employees JOIN Sell ON Employees. 编号=Sell. 售货员工编号 JOIN Goods ON Goods.
商品编号=Sell. 商品编号
WHERE Sell. 数量>ALL(SELECT Sell. 数量
FROM Employees JOIN Sell
ON Employees. 编号=Sell. 售货员工编号
JOIN Goods ON Goods. 商品编号=Sell. 商品编号
WHERE 姓名='王峰')
```

【例 7.15】 本例演示多个字符串的连接。

```
USE Sales
GO
SELECT RTRIM(姓名)+' ':' '+SPACE(1)+电话 AS 姓名及电话
FROM Employees
```

【例 7.16】 在 WHILE 循环中, 包含两条语句, 需要 BEGIN...END 语句将这两条语句封闭起来组成一个语句块。

```
DECLARE @counter int
SELECT @counter=0
WHILE @counter<10
BEGIN
    SELECT @counter=@counter+1 --计数器循环累加
    PRINT @counter --将计数器的值显示出屏幕来
END
```

【例 7.17】 测试Sales数据库的Goods表中是否有“HP喷墨打印机 photosmart 7268”这种商品，有的话显示该商品的信息，否则显示“该商品无进货记录!”。

```
USE Sales
GO
DECLARE @Goodname varchar(20)
SET @Goodname=' HP喷墨打印机 photosmart 7268'
IF EXISTS(SELECT * FROM Goods WHERE 商品名称=@Goodname)
    BEGIN
        PRINT @Goodname+' 商品的信息如下: '
        SELECT * FROM Goods WHERE 商品名称=@Goodname
    END
ELSE
    BEGIN
        PRINT ' 该商品无进货记录! '
    END
GO
```

【例 7.18】 判断 Sales 数据库是否有商品的库存量少于 10，如果有，则将每一商品都入货 50，直到所有商品的库存量都多于 10 或者有商品的库存量超过 200。

```
USE Sales
GO
WHILE EXISTS(SELECT * FROM Goods WHERE 数量<10)
    BEGIN
        UPDATE Goods
        SET 数量=数量+50
        IF (SELECT MAX(数量) FROM Goods)>200
            BREAK
    END
GO
```

【例 7.19】 在 Sales 数据库中查询每个员工在 2005 年 1 月的销售商品数量并发布奖罚信息，销售商品数量低于 30 的为不合格员工，扣除当月提成；30 到 60 之间的，为合格员工；高于 60 的，为优秀员工，凭多出的销量获取奖金；其他情况视为无销量记录。

```
USE Sales
GO
SELECT 售货员工编号, 姓名, 奖罚信息=
CASE
    WHEN sum(数量)<30 THEN ' 不合格员工，扣除当月提成'
    WHEN sum(数量)>=30 AND sum(数量)<60 THEN ' 合格员工'
    WHEN sum(数量)>=60 THEN ' 优秀员工，凭多出的销量获取奖金'
```

```

ELSE '无销售记录'
END
FROM Sell JOIN Employees ON Employees.编号=Sell.售货员工编号
WHERE YEAR(售出时间)=2005 AND MONTH(售出时间)=1
GROUP BY 售货员工编号, 姓名
GO

```

【例 7.20】 使用 DELAY 关键字指定在执行 SELECT 语句之前等待五秒。

```

USE Sales
GO
WAITFOR DELAY '00:00:05'
SELECT 商品名称, 生产厂商, 进货价
FROM Goods
WHERE 进货时间>='2005-1-1'
GO

```

【例 7.21】 本例是符合 SQL-92 标准的游标声明语句。声明一个游标，用于访问 Sales 数据库中的所有进货商品信息。

```

USE Sales
GO
--声明游标
DECLARE Goods_cursor CURSOR
FOR
SELECT * FROM Goods
FOR READ ONLY

```

【例 7.22】 声明一个全局滚动动态游标 sales_cursor,它用于获取所有员工销售“8”号商品的信息，其中包括员工姓名、销售数量和售出时间三列。

```

USE Sales
GO
DECLARE sales_cursor CURSOR
GLOBAL SCROLL DYNAMIC
FOR
SELECT 姓名,数量,售出时间
FROM Employees JOIN Sell ON Employees.编号=Sell.售货员工编号
WHERE 商品编号=8

```

【例 7.23】 将【例 7.22】中声明的游标 sales_cursor 打开，输出这个游标中的记录行数。

```

OPEN sales_cursor
SELECT @@CURSOR_ROWS AS '游标 sales_cursor 记录行数'

```

【例 7.24】 将【例 7.22】中声明的游标打开，并使用该游标读取结果集中的所有记录。

```

--打开游标
OPEN sales_cursor
--第一次读取，得到结果集的首行记录
FETCH NEXT FROM sales_cursor
--循环读取结果集中剩余的数据行
WHILE @@FETCH_STATUS=0
BEGIN
    FETCH NEXT FROM sales_cursor
END

```

【例 7.25】 关闭和释放【例 7.22】中声明的游标 sales_cursor。

```

--关闭游标 sales_cursor
CLOSE sales_cursor
--释放（删除）游标 sales_cursor
DEALLOCATE sales_cursor

```

【例 8.1】 本例创建一个简单的无参数的存储过程：在 Sales 数据库中，创建存储过程 proc_Employees，查询所有的员工信息。

创建和执行存储过程的脚本内容如下：

```

USE Sales
GO
CREATE PROC proc_Employees
AS
SELECT * FROM Employees
--执行存储过程
EXEC proc_Employees

```

【例 8.2】 创建一个带有输入参数的存储过程 proc_goods，查询指定员工所进商品信息。

创建和执行存储过程的脚本内容如下：

```

USE Sales
GO
CREATE PROC proc_goods
    @员工编号 char(6)='1001'
AS
    SELECT 商品编号, 商品名称, 生产厂商, 进货价, 零售价, 数量, 进货时间
    FROM Goods
    WHERE 进货员工编号=@员工编号

```

--执行存储过程，查询 1001 号员工所进的商品的信息

```
EXEC proc_goods @员工编号=default
```

--或

```
EXEC proc_goods @员工编号='1001'
```

【例 8.3】 创建一个带有输入和输出参数的存储过程 proc_GNO，查询指定厂商指定名称的商品所对应的商品编号。

```
USE Sales
GO
CREATE PROC proc_GNO
    @商品名称 varchar(20), @生产厂商 varchar(30),
    @商品编号 int OUTPUT
AS
    SELECT @商品编号=商品编号
    FROM Goods
    WHERE 商品名称=@商品名称 AND 生产厂商=@生产厂商
```

--执行存储过程，查询惠普公司打印机商品编号

```
DECLARE @商品编号 int
EXEC proc_GNO '打印机', '惠普公司', @商品编号 OUTPUT
PRINT '该商品编号为: ' + CAST(@商品编号 AS char(6))
```

【例 8.4】 创建带有参数和返回值的存储过程：在 Sales 数据库中创建存储过程 ProcSumByGoods。查询指定厂商指定名称的商品在某年某月的总销售量。

```
USE Sales
GO
CREATE PROC ProcSumByGoods
    @goodname varchar(20), @corp varchar(30), @year int, @month int,
    @sum int OUTPUT
AS
    --声明和初始化一个局部变量，用于保存系统函数@@ERROR 的返回值
    DECLARE @ErrorSave int
    SET @ErrorSave=0
    --统计指定厂商指定名称的商品在指定年份月份总的销售量
    SELECT @sum=SUM(Sell.数量)
    FROM Sell JOIN Goods ON Sell.商品编号=Goods.商品编号
    WHERE 商品名称=@goodname AND 生产厂商=@corp AND YEAR(售出时间)=@year AND MONTH(售出时间)=@month
    IF (@@ERROR<>0)
        SET @ErrorSave=@@ERROR
    RETURN @ErrorSave
GO
```

--执行存储过程，查询惠普公司 2004 年 10 月的打印机销售总量

```

DECLARE @ret int,@sum int
EXEC @ret=ProcSumByGoods '打印机','惠普公司',2004,10,@sum OUTPUT
PRINT '该存储过程执行结果如下: '
PRINT '返回值='+CAST(@ret AS char(1))
PRINT '总销售量='+CAST(@sum AS char(4))

```

【例 8.5】 本例演示了存储过程的嵌套调用：创建存储过程 `proc_GoodsSell`,嵌套调用存储过程 `proc_GNO`，查询指定厂商指定名称商品的销售情况。

```

CREATE PROC proc_GoodsSell
    @商品名称 varchar(20),@指定厂商 varchar(30)
AS
    DECLARE @商品编号 int
    EXEC proc_GNO @商品名称,@指定厂商,@商品编号 OUTPUT
    SELECT 商品编号,数量 AS 销售数量,售出时间,售货员工编号
    FROM Sell
    WHERE 商品编号=@商品编号

```

--执行存储过程，查询惠普公司打印机的销售情况

```
EXEC proc_GoodsSell '打印机','惠普公司'
```

【例 8.6】 本例是用 `EXECUTE` 语句执行字符串的示例。

```

USE Sales
GO
DECLARE @sqlstr VARCHAR(40)
SET @sqlstr='SELECT * FROM Employees ORDER BY 姓名 '
EXEC(@sqlstr)

```

【例 8.7】 使用相关的系统存储过程查看 `Sales` 数据库中 `proc_GoodsSell` 存储过程的定义、相关性和参数。

```

USE Sales
GO
EXEC sp_helptext proc_GoodsSell  --查看存储过程的定义
EXEC sp_depends proc_GoodsSell  --查看存储过程的相关性
EXEC sp_help proc_GoodsSell      --查看存储过程的参数

```

【例 8.8】 将存储过程 `ProcSumByGoods` 重命名为 `proc_单月总销量`。

```
EXEC sp_rename ProcSumByGoods, proc_单月总销量
```

【例 8.9】 修改【例 8.2】存储过程，要求加密存储过程的定义文本，不指定参数的默认值，只查询指定员工所进商品的名称、生产厂商、进货价和数量信息。

```

USE Sales
GO
CREATE PROC proc_goods
    @员工编号 char(6)

```

```

WITH ENCRYPTION
AS
    SELECT 商品名称, 生产厂商, 进货价, 数量
    FROM Goods
    WHERE 进货员工编号=@员工编号

```

【例 8.10】 删除【例 8.3】和【例 8.5】创建的存储过程。

```
DROP PROC proc_GN0, proc_GoodsSell
```

【例 9.1】 创建一个简单的触发器，当有人试图更新 Sales 数据库的商品信息时，利用触发器产生提示信息。

```

--创建触发器
USE Sales
GO
CREATE TRIGGER tri_UpdateGoods
ON Goods
FOR UPDATE
AS
    RAISERROR('更新表数据', 16, 10)
--测试触发器
UPDATE Goods
SET 数量=8
where 商品编号=5

```

结果如下：

```

服务器: 消息 50000, 级别 16, 状态 10, 过程 tri_UpdateGoods, 行 5
更新表数据

```

【例 9.2】 创建一个 AFTER INSERT 触发器，当在 Sales 数据库的 employees 表中插入一条新员工记录时，如果不是“采购部”、“财务部”、“销售部”或“库存部”的员工，则撤消该插入操作，并返回出错消息

```

USE Sales
GO
CREATE TRIGGER tri_InsertEmp
ON Employees
FOR INSERT
AS
    declare @depart varchar(16)
    select @depart=Employees. 部门
    from Employees, inserted
    where Employees. 编号=inserted. 编号
    if @depart<>'采购部' OR @depart<>'财务部' OR @depart<>'销售部' OR @depart<>'库

```

存部’

```

begin
    rollback transaction
    --返回用户定义的错误信息并设系统标志，记录发生错误
    raiserror('不能插入非本公司设定部门的员工信息！', 16, 10)
end
--测试 insert 触发器
INSERT Employees(编号, 姓名, 性别, 部门, 电话, 地址)
VALUES('1501', '彭真', 1, '人事部', '3988563', '北京市南京路 756 号')

```

【例 9.3】在 Sales 数据库的 employees 表和 Sell 表之间具有逻辑上的主外键关系，要求当删除或更新员工记录的时候，要激发触发器 tri_Delete，在 Sell 表中也删除或更新相对应的记录行。

```

USE Sales
GO
CREATE TRIGGER tri_Update_Delete
ON Employees
FOR UPDATE, DELETE
AS
    SET NOCOUNT OFF    --不返回结果
    DECLARE @delcount INT
    DECLARE @Empid CHAR(6)
    --更新
    IF UPDATE(编号)
    BEGIN
        UPDATE Sell
        SET 售货员工编号=(SELECT 编号 FROM inserted)
        WHERE 售货员工编号 in (SELECT 编号 FROM deleted)
    END
    --删除
    SELECT @delcount=COUNT(*) FROM deleted
    IF @delcount>0
    BEGIN
        --从临时表 deleted 中获取要删除的售货员工编号
        SELECT @Empid=编号 FROM deleted
        --从 Sell 表中删除该员工的销售记录
        DELETE FROM Sell WHERE 售货员工编号=@Empid
    END
END

```

【例 9.4】查看 Sales 数据库中 employees 表的触发器类型。

```

USE Sales
GO

```

```
EXEC sp_helptrigger 'employees'
EXEC sp_helptrigger 'employees', 'DELETE'
```

【例 9.5】 查看【例 9.3】创建的触发器 tri_Update_Delete 的定义文本。

```
EXEC sp_helptext 'tri_Update_Delete'
```

【例 9.6】 查看【例 9.3】创建的触发器 tri_Update_Delete 的所有者和创建日期。

```
EXEC sp_help 'tri_Update_Delete'
```

【例 9.7】 修改触发器 tri_UpdateGoods，加密触发器文本定义。

```
USE Sales
GO
ALTER TRIGGER tri_UpdateGoods
ON Goods
WITH ENCRYPTION
FOR UPDATE
AS
RAISERROR('更新表数据', 16, 10)
```

【例 9.8】 修改触发器 tri_UpdateGoods 的名称，更名为“triUpdGood”。

```
EXEC sp_rename 'tri_UpdateGoods', 'triUpdGood'
```

【例 9.9】 禁用触发器 tri_Update_Delete 的使用。

```
ALTER TABLE employees
DISABLE TRIGGER tri_Update_Delete
```

/*测试，看 tri_Update_Delete 触发器禁用后删除一个员工记录还有没有使得触发器触发执行*/

```
DELETE employees
WHERE 编号='1301'
```

【例 9.10】 删除触发器 triUpdGood 和 tri_Update_Delete

```
DROP TRIGGER triUpdGood, tri_Update_Delete
```

【例 10.1】 使用查询分析器在 Sales 数据库创建名为 Fn_Cost 的自定义函数，用于计算 Goods 数据表的进货金额，并将其绑定到 Goods 数据表。

(1)定义函数

在查询分析器中运行以下命令：

```
USE Sales
GO
CREATE FUNCTION Fn_Cost
(@x Int , @y Money )
RETURNS Money
AS
BEGIN
RETURN (@x*@y)
END
```

(2)在 Goods 数据表中增加一列，列名为金额，并将函数 **Fn-Cost** 与其绑定。

在查询分析器中运行以下命令：

```
ALTER TABLE Goods ADD 金额 AS dbo.Fn_Cost(数量,进货价)
```

(3)测试

在查询分析器中运行以下命令：

```
SELECT * FROM Goods
```

【例 10.2】使用查询分析器在 Sales 数据库创建名为 **Fn_Total** 的自定义函数，用于统计 Sell 数据表在某一时间段内的销售情况。

(1)定义函数

在查询分析器中运行以下命令：

```
USE Sales
GO
CREATE FUNCTION Fn_Total
(@bt DateTime,@et DateTime)
RETURNS TABLE
AS
RETURN
    (SELECT Goods.商品名称,Sell.数量
    FROM Goods,Sell
    WHERE (Sell.售出时间>=@bt AND Sell.售出时间<=@et
        AND Goods.商品编号=Sell.商品编号)
    )
```

(2)测试

在查询分析器中运行以下命令：

```
SELECT * FROM dbo.Fn_Total('2005-1-1','2005-1-31')
```

从返回结果可以看到 1 月份的销售记录。

【例 10.3】使用查询分析器在 Sales 数据库创建名为 **Fn_Lan** 的自定义函数，该函数生成一张数据表，数据表的内容为进货价为指定价格以上的商品。

(1)定义函数

在查询分析器中运行以下命令：

```
CREATE FUNCTION Fn_Lan
(@price Money)
RETURNS @Fn_Lan TABLE
(商品编号 Int PRIMARY KEY NOT NULL,
  商品名称 VarChar(20) NOT NULL,
  生产厂商 VarChar(30) NOT NULL,
  进货价 Money NOT NULL,
  进货时间 DateTime NOT NULL
)
```

```

AS
BEGIN
    INSERT @Fn_Lan
    SELECT 商品编号, 商品名称, 生产厂商, 进货价, 进货时间
    FROM Goods
    WHERE (进货价>=@price)
RETURN
END

```

(2)测试

在查询分析器中运行以下命令：

```
SELECT * FROM dbo.Fn_Lan(1000)
```

返回结果都是进货价为 1000 元以上的商品。

【例 10.4】定义一个事务，向数据表 Sell 插入 3 条记录，最后提交该事务。

在查询分析器中运行如下命令：

```

USE Sales
GO
--事务开始
BEGIN TRANSACTION
    INSERT Sell (销售编号, 商品编号, 数量, 售出时间, 销售员工编号)
        VALUES (20, 18, 2, '2005-5-20', '000008')
    INSERT Sell (销售编号, 商品编号, 数量, 售出时间, 销售员工编号)
        VALUES (30, 18, 2, '2005-5-20', '000008')
    INSERT Sell (销售编号, 商品编号, 数量, 售出时间, 销售员工编号)
        VALUES (40, 18, 2, '2005-5-20', '000008')

```

--提交事务

```
COMMIT TRANSACTION
```

验证结果：在查询分析器中运行如下命令：

```
SELECT * FROM Sell WHERE 销售员工编号='000008'
```

可以看到以上 3 条记录确实已经成功添加。

【例 10.5】定义一个事务，向数据表 Sell 插入 3 条记录，最后回滚该事务。

在查询分析器中运行如下命令：

```

USE Sales
GO
--事务开始
BEGIN TRANSACTION
    INSERT Sell (销售编号, 商品编号, 数量, 售出时间, 销售员工编号)
        VALUES (13, 18, 2, '2005-5-20', '000009')
    INSERT Sell (销售编号, 商品编号, 数量, 售出时间, 销售员工编号)
        VALUES (14, 12, 5, '2005-5-20', '000009')

```

```
INSERT Sell (销售编号, 商品编号, 数量, 售出时间, 销售员工编号)
VALUES (15, 16, 3, '2005-5-20', '000009')
```

--回滚事务

```
ROLLBACK TRANSACTION
```

验证结果：在查询分析器中运行如下命令：

```
SELECT * FROM Sell WHERE 销售员工编号='000009'
```

可以看到以上 3 条记录确实没有添加。

【例 10.6】定义一个事务，向数据表 **Sell** 插入 1 条记录后设置一个保存点，然后再向数据表 **Sell** 插入 3 条记录，最后将事务回滚到该保存点。

在查询分析器中运行如下命令：

```
USE Sales
```

```
GO
```

--事务开始

```
BEGIN TRANSACTION
```

```
INSERT Sell (销售编号, 商品编号, 数量, 售出时间, 销售员工编号)
VALUES (16, 18, 2, '2005-5-20', '000019')
```

```
SAVE TRANSACTION s1
```

```
INSERT Sell (销售编号, 商品编号, 数量, 售出时间, 销售员工编号)
VALUES (17, 12, 5, '2005-5-20', '000019')
```

```
INSERT Sell (销售编号, 商品编号, 数量, 售出时间, 销售员工编号)
VALUES (18, 16, 3, '2005-5-20', '000019')
```

```
INSERT Sell (销售编号, 商品编号, 数量, 售出时间, 销售员工编号)
VALUES (19, 16, 3, '2005-5-20', '000019')
```

--回滚事务到保存点 s1

```
ROLLBACK TRANSACTION s1
```

验证结果：在查询分析器中运行如下命令：

```
SELECT * FROM Sell WHERE 销售员工编号='000019'
```

从显示结果可以看到只有 1 条记录被添加，另外 3 条记录确实没有添加。

【例 10.7】使用企业管理器浏览系统中的锁。

- (1) 展开服务器组，展开服务器。
- (2) 展开“管理”，然后展开“当前活动”。
- (3) 展开“锁/进程 ID”，可以查看每个连接的当前锁。
- (4) 展开“锁/对象”，可以查看每个对象的当前锁。
- (5) 单击要查看的锁，当前锁显示在详细信息栏目中。

在查询分析器中，使用系统存储过程 **SP_LOCK** 可以查看正在运行的某一进程拥有的锁的信息。系统存储过程 **SP_LOCK** 的语法格式如下：

```
SP_LOCK [[@spid1=] 'spid1' ] [, [@spid2=] 'spid2' ]
```

其中存储过程的参数指定的是进程的标识号，该标识号存储在 **master.dbo.sysprocess** 中。如果没有指定参数，存储过程将返回所有锁的信息。

【例 10.8】在查询分析器中，使用系统存储过程 SP_LOCK 查看当前持有锁的信息。

在查询分析器中运行如下命令：

```
USE MASTER
GO
EXEC SP_LOCK
GO
```

【例 11.1】XX 学院局域网内部 Windows 2000 Server 服务器上新装了 SQL Server 2000 系统，且原服务器中已为单位员工设置了用户账户，现要求为其 SQL Server 2000 系统设置认证模式。

分析：由于原服务器上已经建立了员工的账户，为了充分利用 Windows 2000 Server 的安全管理机制，可以选用 Windows 认证模式。这样，不用在 SQL Server 上另建一套账户，对账户的管理可交给 Windows 2000 Server。

【例 11.2】某单位要开发一个对外发布的网站，后台数据库用 SQL Server 2000,请问应该用那种验证模式。

分析：由于网站中应用程序是基于 Internet 运行的，如果使用 Windows 认证模式，则需要为分布在各个位置的用户建立登录信息，这不太现实。如果使用 SQL Server 登录认证，所有用户登录信息都可以由服务器端的 SQL Server 来维护，才可实现。

【例 11.3】XX 学院有两类用户，一类是只要在校内访问 SQL Server 2000 服务器中的 Sales 数据库，如财务处的用户；另一类要求出差在校外也要访问 SQL Server 2000 服务器的 Sales 数据库。问：应如何为这两类用户设置登录账户和数据用户。

分析：从前面的知识可知，要访问 SQL Server 数据库，用户必须先能够登录 SQL Server，即拥有合法的登录账户，登录 SQL Server 后还必须具有访问 Sales 数据库的账户。因此可按如下思路解决：

(1) 要使两类用户都能访问 SQL Server，必须将 SQL Server 服务器的身份验证模式设为“SQL Server 和 Windows(S)”。

(2) 对校内用户，只要将其原有的 Windows 账户设为 SQL Server 登录账户。

(3) 对校外用户，要为其设置 SQL Server 登录账户

(4) 设置完登录者后，可在 Sales 数据库中用户中加入上述登录者，即可将其映射为合法的数据库用户。

【例 11.4】要删除一个使用 SQL Server 身份验证的登录账户“whb”的 SQL 语句为：

```
EXECUTE sp_droplogin whb
```

【例 11.5】要删除一个使用 Windows 身份验证的登录账户“work1\jyb”的 SQL 语句为：

```
EXECUTE sp_revokelogin 'work1\jyb'
```

【例 11.6】假如有使用 SQL Server 身份验证的登录账户“whb”，为其添加一个 sales 数据库用户账户的方法为：

(1) 在 SQL 查询分析器中，选择当前数据库为 Sales；

(2) EXECUTE sp_grantdbaccess 'whb','whb'



图 11.9 使用存储过程创建数据库用户

【例 11.7】将登录账户“whb”添加到 sales 数据库，使其成为数据库用户“whb”，然后将“whb”添到 sales 数据库的 db_accessadmin 角色中。

```
Use sales
EXECUTE sp_grantdbaccess 'whb', 'whb'
EXECUTE sp_addrolemember 'db_accessadmin ', 'whb'
EXECUTE sp_helpuser 'whb'
```

【例 11.8】从当前数据库 sales 中删除用户账户 whb

```
EXECUTE sp_revokedbaccess 'whb'
```

【例 11.9】XX 学院需要一名对 SQL Server 服务器进行系统管理的人员，他现在拥有 SQL Server 身份认证模式的登录名 whb，现要升级他的的权限，让其拥有对系统的所有操作权。

分析：在服务器固定角色中存在 sysadmin 角色，其可执行 SQL Server 中的任何任务，是 SQL Server 的管理员组，根据需求，可在该角色中添加成员 whb 即可。

1、使用企业管理器实现上述目标

操作步骤：

- (1) 展开服务器组，然后展开需要进行角色管理的服务器。
- (2) 展开“安全性”，单击“服务器角色”。
- (3) 在右边窗口中选中 sysadmin 角色。
- (4) 单击右键，选择“属性”，弹出“服务器角色属性”对话框。
- (5) 选择“常规选项卡”后单击“添加”，然后在弹出的“添加成员”对话框中列表中选择要添加的成员（如：whb），如图 11.12 所示。

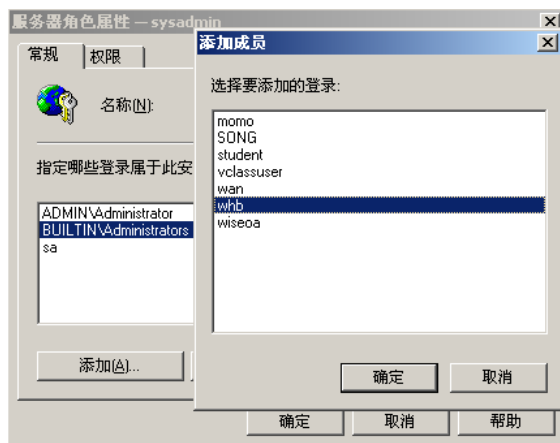


图 11.12 添加成员

(6) 单击“确定”，返回“服务器角色属性”，新成员成功添加在登录列表中。

(7) 单击“确定”完成本任务。

2、使用存储过程 `sp_addsrvrolemember` 实现上述目标

```
EXECUTE sp_addsrvrolemember 'whb', 'sysadmin'
```

【例 11.10】如果因工作原因，登录名为 `whb` 的用户不应该再拥有管理 `XX` 学院 `SQL Server` 服务器的全部权力，但其仍能登录 `SQL Server` 服务器，应如何设置？

分析：可以删除登录账户 `whb` 在固定服务器角色 `sysadmin` 中的映射，不能删除其登录名，因为这样 `whb` 将不能登录 `SQL Server`。

1、使用企业管理器删除固定服务器角色中的成员

操作步骤：

- (1) 展开服务器组，然后展开需要进行角色管理的服务器。
- (2) 展开“安全性”，单击“服务器角色”。
- (3) 在右边窗口中选中 `sysadmin` 角色。
- (4) 右击角色 `sysadmin`，选择“属性”。
- (5) 在弹出的“服务器角色属性”中选择要删除的登录成员（如：`whb`）。
- (6) 单击“删除”，然后单击“确定”即可成功删除。

2、使用存储过程 `sp_dropsrvrolemember` 删除固定服务器角色中的成员

```
EXECUTE sp_dropsrvrolemember 'whb', 'sysadmin'
```

【例 11.11】某用户的登录账户为 `work1\jyb`，在数据库 `sales` 中的数据库用户账户为 `work1\jyb`，如果他要数据库 `Sales` 拥有任意操作权限，应如何设置？

分析：服务器固定角色中存在 `db_owner`，该角色的权限跨越所有其他固定数据库角色，只要把数据库用户 `work1\jyb` 添加到该角色中即可满足要求。

1、用企业管理器实现向固定数据库角色添加成员

操作步骤：

- (1) 在企业管理器选择要管理的数据库，如 `Sales`。
- (2) 展开 `sales`，单击“角色”。

(3) 在右边窗口的角色列表中选择 `db_owner`，右击该角色，然后选择“属性”，如图 11.13。

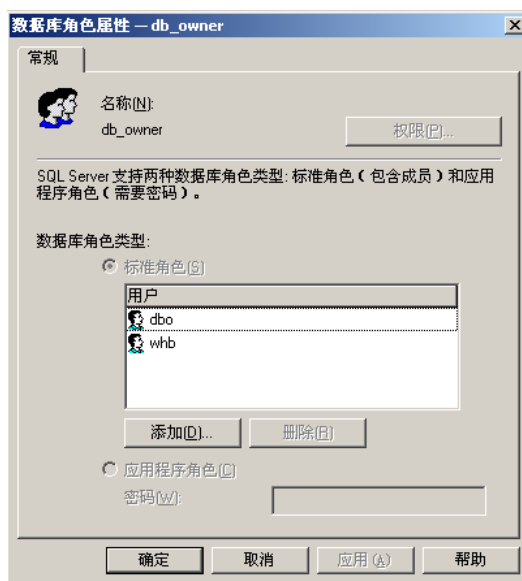


图 11.13 数据库角色属性

(4) 在“常规”选项卡单击“添加”弹出“添加成员”对话框。

(5) 在“添加成员”列表中选择要添加的数据库用户，如选择 `work1\jyb`，然后按“确定”。

(6) 此时在“数据库角色属性”对话框中出现新添加的成员，然后按“确定”完成数据库角色添加。

2、使用系统存储过程 `sp_addrolemember` 实现向固定数据库角色添加成员

```
EXECUTE sp_addrolemember 'db_owner', 'WORK1\jyb'
```

【例 11.12】如果 XX 学院的高级系统管理人员登录 SQL Server 服务器时需要对数据库 `sales` 有一些特殊的访问要求，如何进行权限设置？

分析：对这类人员进行权限设置，为避免大量重复劳动，可设置用户自定义角色，规定该角色的权限，然后将需要该权限的人员加入其中即可。

使用企业管理器实现创建自定义数据库角色

操作步骤：

- (1) 展开需要定义角色的数据库，如 `sales`。
- (2) 右键单击“角色”节点，选择“新建数据库角色”。
- (3) 在“名称”框中输入新角色的名称，如“`admins`”，如图 11.14 所示。

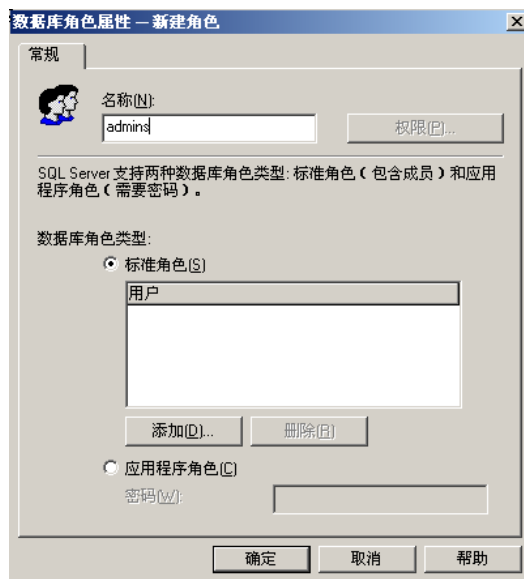


图 11.14 新建角色

(4) 选择“标准角色”，然后单击“确定”完成设置。

【例 11.13】假设 whb 是数据库 sales 的数据库用户，sales 中有员工表 Employees，如果要求 whb 只能查看 Employees 中的数据，不能做任何更改和删除，应如何设置？

分析：要实现上述目的，可对 SQL Server 中的 sales 数据库用户 whb 进行必要的数据库访问权限设置。

1、使用企业管理器管理数据库用户的权限

操作步骤：

在企业管理器中，展开要进行权限设置的数据库（如：Sales）。

选择“用户”，在右边列出的用户中选择要设置权限的用户（如：whb）。

右击该用户，选择“所有任务”->“管理权限”，弹出“数据库用户属性”对话框。

在“数据库用户属性”对话框中根据实际需要执行下列操作之一：

若要授予用户对某个数据对象的访问权限，可在对象列表中单击相应复选框，打上 ☒ 标记。（如：给表 sales 的 SELECT 打上 ☒ 标记）

若要禁止用户对某个数据对象的访问权限，可在对象列表中单击相应复选框两次，打上 ☒ 标记。

若要废除用户对某个数据库对象的访问权限，可以在对象列表中单击相应的复选框，以清除复选框。

(5) 单击“确定”或“应用”，使所作设置生效。

例如，要实现【例 11.13】的目标可用如下语句：

```
USE sales
GRANT SELECT ON Employees TO whb
DENY INSERT, UPDATE, DELETE ON Employees TO whb
```

【例 11.14】XX 学院有一群用户，他们对数据库 sales 中的员工表 Employees 有访问权，但要求他们只能查看 Employees 中的数据，不能做任何更改和删除，应如何设置？

分析：按照【例 11.13】的方法，给每个数据库用户设置权限可以解决本问题，但效率很差。

科学的方法是：新建一数据库角色，然后对此角色设置权限，把相关数据库用户添加到此角色中。新建数据库角色、向数据库角色中添加成员的方法前面已经学习，这里只要解决给数据库角色设置权限即可。

1、使用企业管理器管理数据库角色的权限

操作步骤：

- (1) 在企业管理器中，展开要进行权限设置的数据库（如：sales）。
- (2) 选择“角色”，在右边列出的角色中选择要设置权限的自定义角色（如：ck）。
- (3) 右击该角色，选择“属性”，然后单击“权限”，弹出“数据库角色属性”对话框。

(4) 在“数据库角色属性”对话框中根据实际需要单击相应的复选框，以便授予、废除或禁止该角色对某个数据对象的访问权限。

(5) 单击“确定”或“应用”，使所作设置生效。

【例 11.15】对于数据库 Sales，它的所有数据库用户对员工表 Employees 有访问权，但不能访问“地址”字段，而且所有用户都不能删除、修改和插入记录。

分析：可对所有用户正常设置其他权限，然后对“地址”字段设置为拒绝访问。

1、使用企业管理器管理数据库对象的权限

操作步骤：

- (1) 在企业管理器中选择要设置的数据库，如：Sales。
- (2) 根据实际需要，执行下列操作之一：
 - 若要设置表的访问权限，可以单击“表”；
 - 若要设置视图的访问权限，可以单击“视图”；
 - 若要设置存储过程的访问权限，可以单击“存储过程”。
- (3) 在右边窗口中右击要设置权限的数据对象（如：右击表 Employees），然后选择“所有任务”->“管理权限”。
- (4) 在弹出的“对象属性”对话框中可以看到此数据库中的所有用户和角色，根据实际需要，按前面所说方法设置权限，如图 11.17 所示。
- (5) 设置完用户的其他权限后单击“列”，弹出如图 11.18 所示“列权限”对话框。
- (6) 在图 11.18 中的 SELECT 列的需要禁止访问字段处（如：“地址”），选中方格，使其打上  标记，表示拒绝访问，然后按“确定”回到“对象属性”对话框。
- (7) 单击“确定”或“应用”，使所作设置生效。

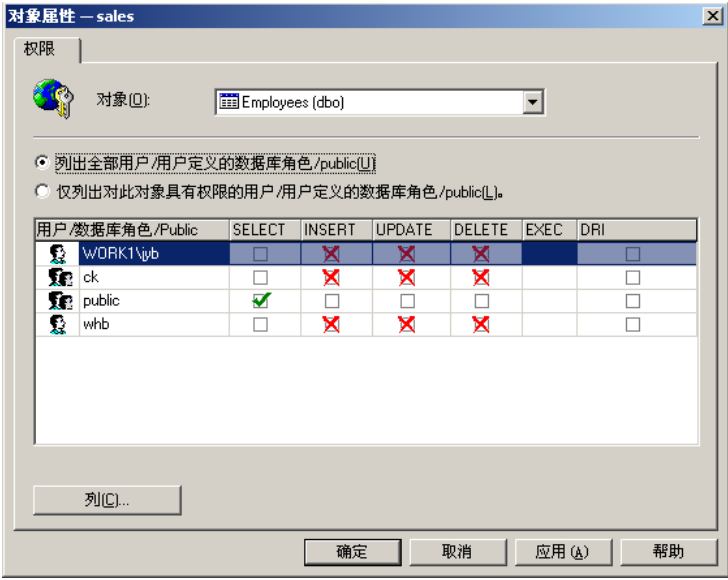


图 11.17 设置数据库对象的权限

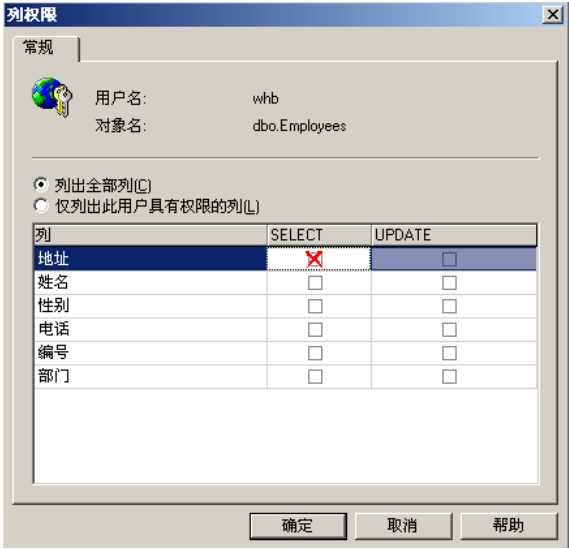


图 11.18 设置数据库对象列的权限

【例 11.16】假如在目录 C:\Program Files\Microsoft Office\Office\Samples 下有一 Access 数据库 Northwind.mdb，此数据库的表如图 11.24，现在要把“产品”表中的所有数据导入到 SQL Server 数据 Sales 中。

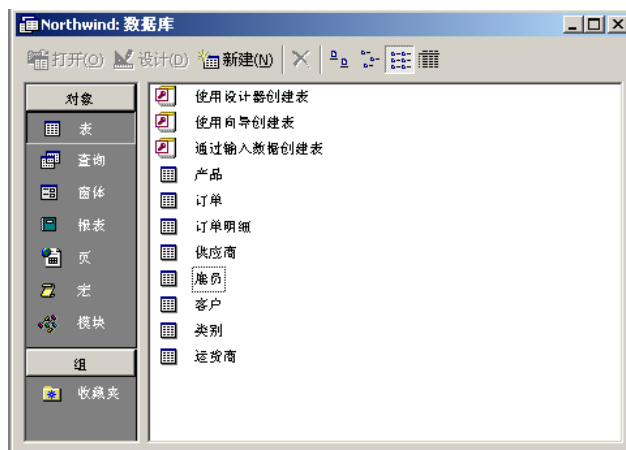


图 11.24 数据库 Northwind.mdb

操作步骤:

- (1) 启动数据导入导出工具。选择“开始->程序->Microsoft SQL Server->导入和导出数据”命令，然后单击“下一步”。
- (2) 当出现如图 11.19 所示的“选择数据源”对话框时，在“数据源”下拉列表中选择  Microsoft Access。
- (3) 单击“文件名”文本框右边的 ，然后选择要导入数据的 Access 数据库（如：选择 C:\Program Files\Microsoft Office\Office\Samples\Northwind.mdb），并单击“下一步”。

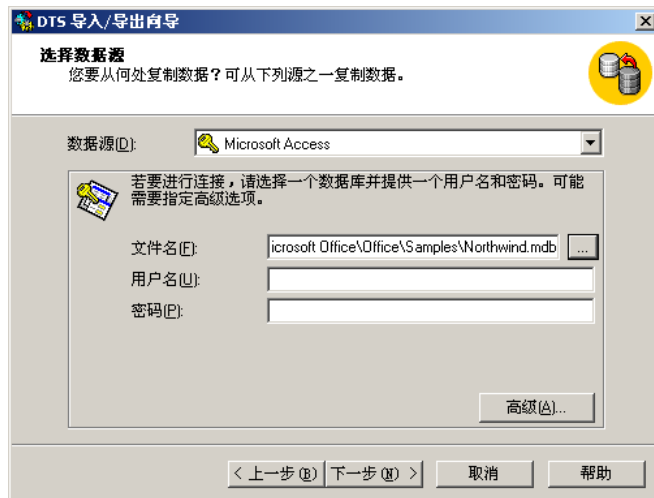


图 11.25 选择数据源

- (4) 当出现图 11.26 所示的对话框时，在“目的”下拉列表中选择“用于 SQL Server 的 Microsoft OLE DB 提供程序”或“用于 SQL Server 的 Microsoft ODBC 驱动程序”。
- (5) 在“服务器”框中输入或选择服务器名，并选择登录方式。
- (6) 单击“刷新”按钮，使所选服务器上的所有数据库出现在“数据库”下拉

列表中，然后选择要导入数据的目标 SQL Server 数据库（如：sales），单击“下一步”。

(7) 选择  从源数据库复制表和视图，然后单击“下一步”。

(8) 在弹出的如图 11.27 所示的选择来源表对话框时，选择需要的表或视图，并“源”一列方格中打上 ☒，然后单击“下一步”。

(9) 在“保存、调度和复制包”对话框中选择“立即执行”，然后按“下一步”。

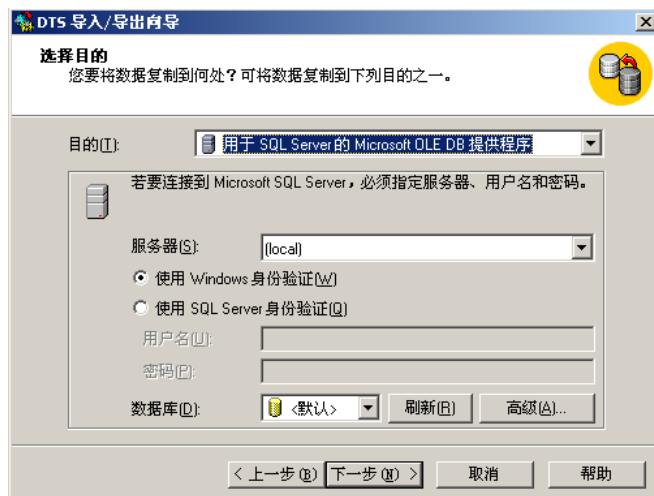


图 11.26 选择目标数据库



图 11.27 选择来源表

(10) 在“正在完成 DTS 导入/导出向导”对话框中单击“完成”。

(11) 当数据转换全部完成后，单击“确定”，然后单击“完成”。

当上述步骤成功完成后，在 sales 数据库中已导入表“产品”。

【例 11.17】首先在本地机器的一个磁盘（如：D 盘）内新建一个文件夹，命名为“数据库备份”，用来保存备份文件。本例通过对 Sales 数据库进行备份，讲解如何采

用向导来备份数据库。

操作步骤：

- (1) 在企业管理器中展开服务器组，然后选中 Sales 数据库所在的服务器。
- (2) 从“工具”菜单中选择“向导”命令，弹出如图 11.28 所示的“选择向导”对话框。



图 11.28 “选择向导”对话框

(3) 在“管理”节点中选择备份向导，单击“确定”按钮，弹出“创建数据库备份向导”的欢迎界面。

(4) 单击“下一步”按钮，弹出“创建数据库备份向导”的选择要备份数据库的界面。在数据库的下拉列表中选择 sales 数据库（如图 11.29）。



图 11.29 选择要备份的数据库

(5) 单击“下一步”弹出“创建数据库备份向导”的键入备份的名称和描述的界面。在名称下面的文本框中输入备份文件的名称，在描述下面的文本框中输入说明性文字。

(6) 单击“下一步”弹出“创建数据库备份向导”的选择数据库备份类型界面。这里面有三种类型供选择：数据库备份，即全库备份；差异数据库，即差异备份；事务日志，即事务日志备份。选择其中的一种备份类型（如图 11.30）。

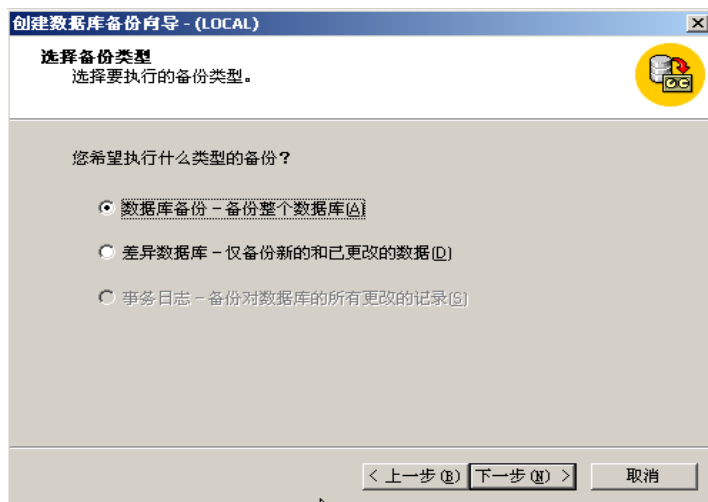


图 11.30 选择数据库备份类型

(7) 单击“下一步”弹出“创建数据库备份向导”的选择备份目的和操作的界面（如图 11.31）。在“选择备份设备”选项中选择“文件”，然后单击...按钮，弹出备份设备位置的界面（如图 11.32），选择数据库要备份到的位置 D 盘的“数据库备份”文件夹。在文件的名称后面文本框中可以对备份文件进行重命名。

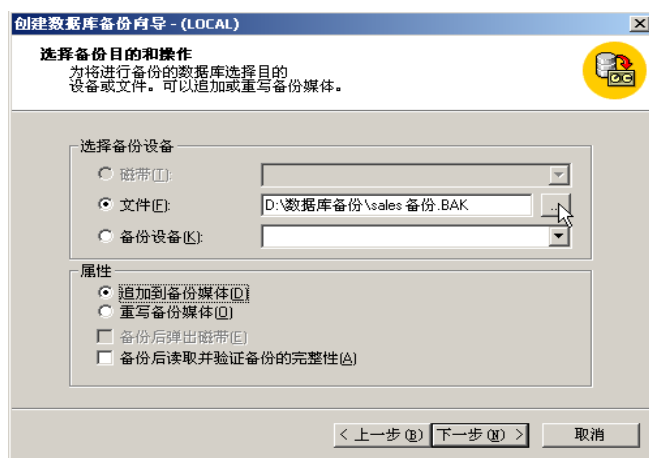


图 11.31 选择备份文件的备份位置

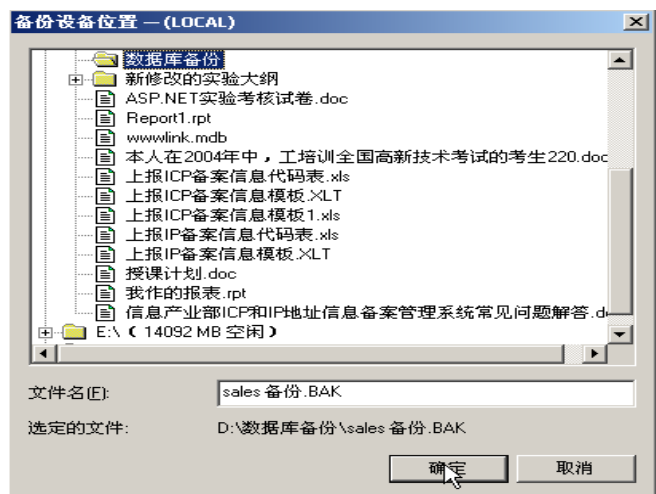


图 11.32 重命锁定在名文件名称和选择备份位置

(8) 单击“下一步”弹出“创建数据库备份向导”的备份验证和调度的界面（如图 11.33）。可对“检查媒体集”和“调度”进行设置。

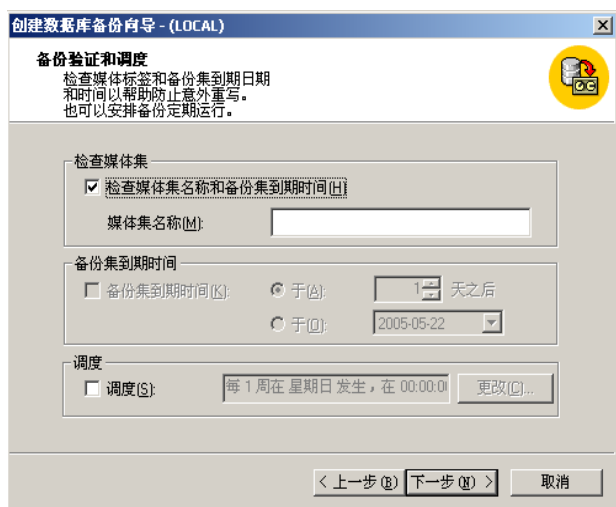


图 11.33 备份验证和调度

(9) 设置完毕之后单击“下一步”弹出“创建数据库备份向导”的最后界面，可以查看我们刚刚所做的各项步骤设置结果（如图 11.34）。然后单击“完成”按钮，进行数据库备份。

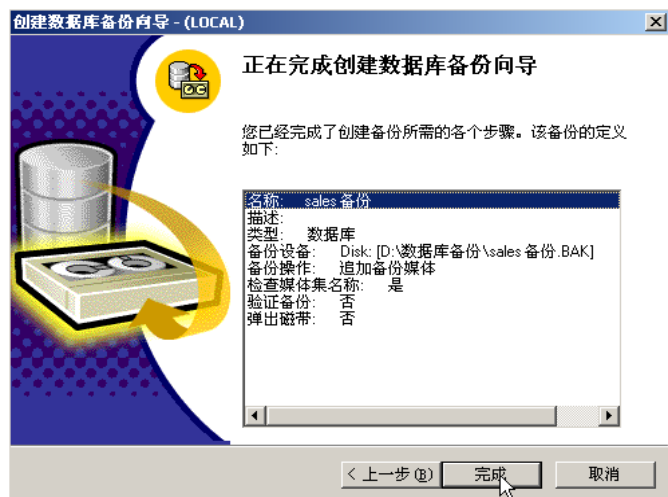


图 11.34 完成数据库备份设置操作

(10) 到“D:\数据库备份\”目录下就会看到已经存在“sales 备份.bak”这个文件。

【例 11.18】本例同样通过对 sales 数据库进行备份，备份位置为“D:\备份数据库\”，讲解如何采用企业管器备份数据库。

(1) 在企业管理器中展开服务器组，打开 sales 数据库所在的服务器，选中 sales 数据库。

(2) 选择“操作”菜单中的“所有任务”，在弹出的下级菜单中选择“备份数据库”命令（如图 11.35），弹出“SQL Server 备份—sales”对话框（如图 11.36）。也可以单击鼠标右键，在弹出的菜单中选择“所有任务”，然后在下级菜单中选择“备份数据库”命令。

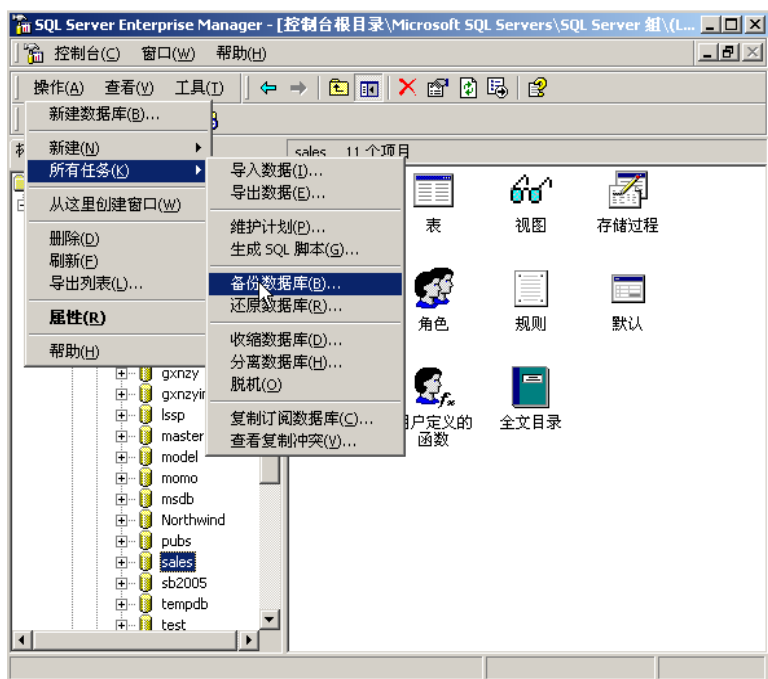


图 11.35 备份数据库

(3) 在“SQL Server 备份—sales”对话框的“常规”选项卡中确定保存的文件名称，这次将备份文件的名称命名为 sales1 备份,避免覆盖【例 12】中的备份文件。在“备份”区选择备份的类型（如果“事务日志”与“文件和文件组”选项为灰色，在选中 sales 数据库后单击鼠标右键，在菜单中选择“属性”命令，然后在“选项”选项卡“故障还原”的模型的下拉列表中选择“完全”，那么两项就会变为可操作项）；在“目的”区单击 **添加(A)...** 按钮选择备份的位置；在“重写”区选择备份方式；在“调度”区中指定备份日程。



图 11.36 “SQL Server 备份—sales”对话框

(4) 设置完毕后，单击“确定”按钮开始数据库备份。

(5) 到“D:\数据库备份\”目录下就会看到已经存在“sales1 备份.bak”这个文件。

【例 11.19】添加磁盘存储设备：下面的示例添加一个逻辑名为“mybackup”的磁盘备份设备，其物理名称为“D:\数据库备份\sales2 备份.bak”。

```
USE sales
```

```
EXEC sp_addumpdevice 'disk', 'mybackup', 'D:\数据库备份\sales2 备份.bak'
```

【例 11.20】添加网络磁盘备份设备：下面的示例显示一个远程磁盘备份设备。在其下启动 SQL Server 的名称必须对该远程文件拥有权限。

```
USE sales
```

```
EXEC sp_addumpdevice 'disk', 'networkdevice', '\\servername \sharename \path \filename.ext'
```

【例 11.21】备份整个 sales 数据库：

```
USE sales
```

```
EXEC sp_addumpdevice 'disk', 'mybackup', 'D:\数据库备份\sales2 备份.bak'
```

```
BACKUP DATABASE sales TO mybackup
```

【例 11.22】在上例的基础上进行差异备份

```
BACKUP DATABASE sales TO mybackup
```

```
WITH DIFFERENTIAL, NOINIT
```

【例 11.23】对 sales 数据库进行日志备份

```
BACKUP LOG sales TO mybackup WITH NOINIT
```

【例 11.24】将 sales 数据库的文件 sales_data 文件备份到本地磁盘设备 myfileback。

```
USE sales
```

```
EXEC sp_addumpdevice 'disk', 'myfilebackup ', 'D:\数据库备份\salesfile.bak'  
BACKUP DATABASE sales FILE='sales_data' TO myfilebackup
```

【例 11.25】前面章节中，我们已经对 **sales** 数据库进行了备份。那么我们首先在 **sales** 中添加一些新的内容（比如表或视图等），然后利用备份文件 **sales2 备份.bak** 进行恢复，查看操作效果。

操作步骤：

在企业管理器中选中要还原的 **sales** 数据库。

右击选中的数据库，在弹出的快捷菜单中选择”所有任务”->“还原数据库”（如图 11.37），弹出”还原数据库”的对话框（如图 11.38）。

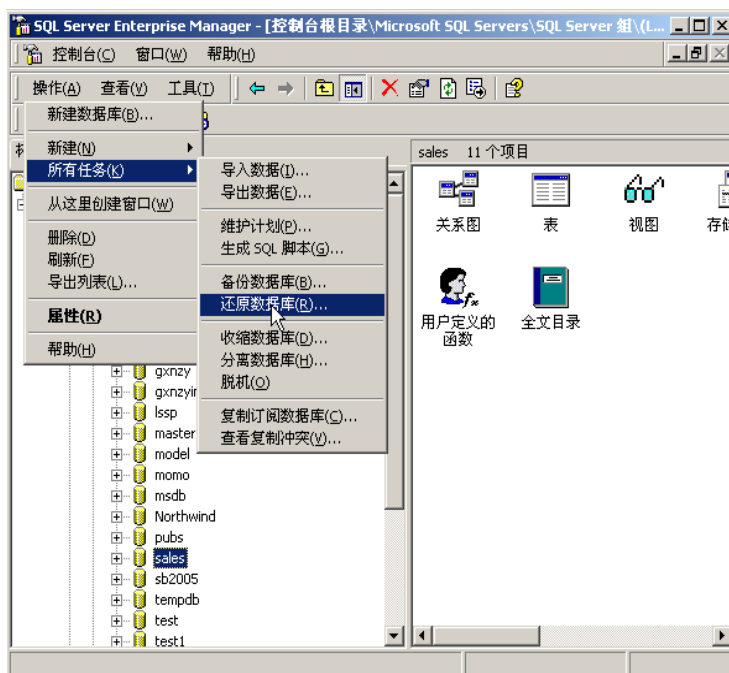


图 11.37 选择还原数据库

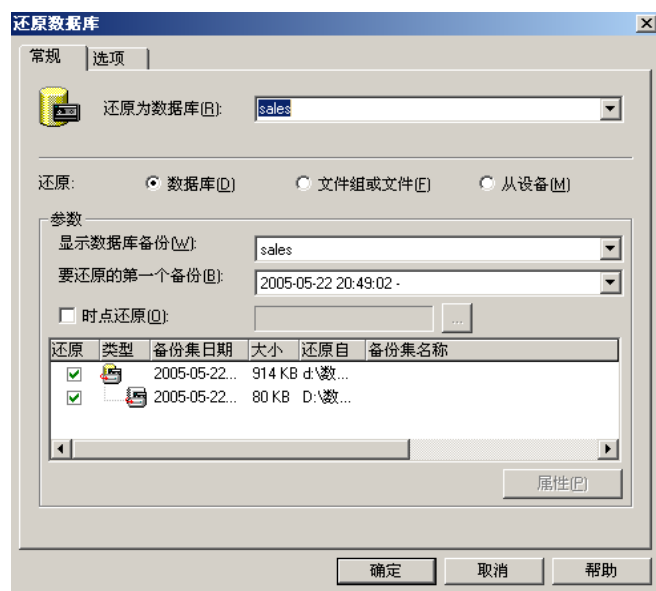


图 11.38 还原数据库

(2) 在“常规”选项卡中的“还原为数据库”区中选择 **sales**；在“还原”区中有三个选项可供选择，“数据库”、“文件组或文件”和“从设备”，本实验选择“数据库”选项；在“参数”区内，在“显示数据库备份”中确认是哪一个数据库的备份，在“要还原的第一备份”中选择哪一时间的备份，在最下的列表框中选择备份文件。

(3) 在“选项”选项卡（如图 11.39）中可以设置还原的选项。在“将数据库还原为”的列表中可以设置还原文件的路径和名称，默认状态下与原文件同路径同名称；在“恢复完成状态”选项中有三个可供选择项，本例中选择第一个单选按钮。

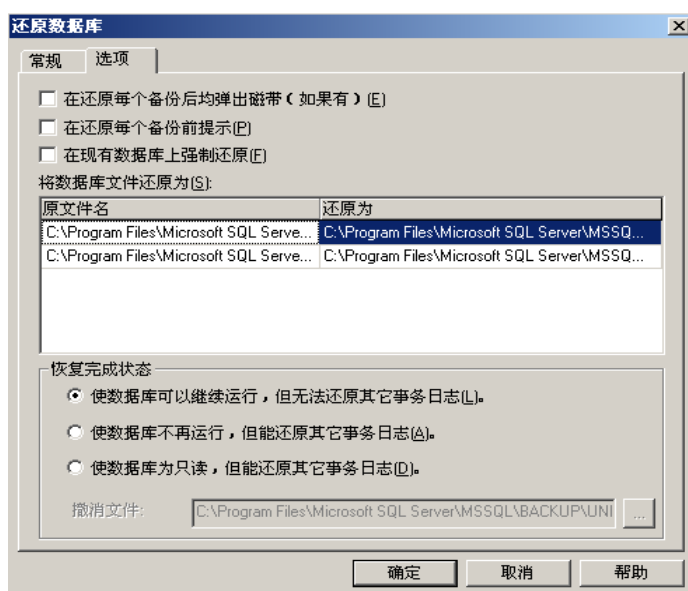


图 11.39 “还原数据库”对话框的”选项”选项卡

(4) 设置完毕后, 单击”确定”按钮, 执行数据库还原操作。单击 **sale** 数据库的“表”选项, 单击工具栏上的  刷新按钮, 发现所建的新内容已经被删除, 还原为备份的数据库内容。

【例 11.26】还原整个数据库

```
RESTORE DATABASE sales FROM mybackup
```

【例 11.27】还原完整数据库备份和差异备份: 下例还原完整数据库备份后还原差异备份。另外, 下例还说明如何还原媒体上的另一个备份集。差异备份追加到包含完整数据库备份的备份设备上。

```
RESTORE DATABASE sales FROM mybackup  
WITH NORECOVERY  
RESTORE DATABASE sales  
FROM mybackup_2  
WITH FILE = 2
```

第 12 章 数据库综合开发应用

12.3 ASP.NET 操作数据库

12.3.3 ASP.NET 程序设计

本示例所书写的程序代码部分采用 VB 的程序语法。

1. 创建 ASP.NET 动态页

运行 Dreamweaver MX, 单击“文件”→“新建”命令, 会弹出图 12-5 所示的“新建文件”对话框, 在“常规”选项卡的“类别”中选择“动态页”, 然后在右边的“动态页”中双击“ASP.NET VB”选项就会得到一个动态页面, 然后对它命名后保存到 C:\inetpub\wwwroot\路径下, 请注意 ASP.NET 文件的扩展名称为.aspx。

2. 导入名称空间

因为我们要在程序中操作数据库, 所以需要导入名称空间, 将它们写在

```
<%@ Page Language="VB" ContentType="text/html"  
ResponseEncoding="gb2312" %>的下面, 如图 12.6。
```

```
<%@Import Namespace="System.Data" %>
```

```
<%@Import Namespace="System.Data.SqlClient"%
```

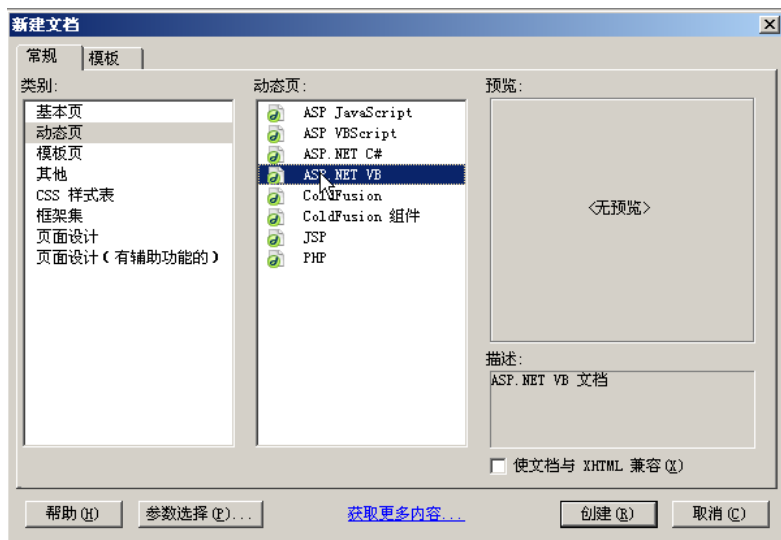


图 12.5 选择动态页 ASP.NET VB 选项

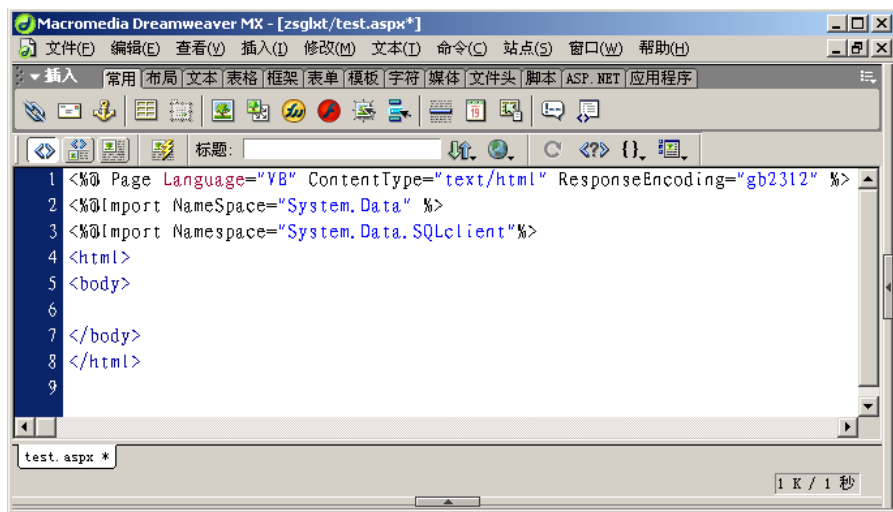


图 12.6 创建名称为 test 的动态页面

3.利用 Datagrid 控件显示 sales 库 Employees 表中的记录

- (1) 创建一个动态页面 browser.aspx 并保存到路径 C:\Inetpub\wwwroot\。
- (2) 在此页面书写添加一个 Datagrid 控件用来显示记录，下面是整个页面的程序代码（程序 12.1）。

程序 12.1 browser.aspx 显示 sales 库 Employees 表中的记录

```
<%@ Page Language="VB" ContentType="text/html" ResponseEncoding="gb2312" %>
```

```

<%@Import Namespace="System.Data" %>
<%@Import Namespace="System.Data.SqlClient"%>

<script runat="server">
'下面定义变量
dim sql as string
'下面定义读取数据库的对象，连接数据库
dim str as string="server=(local);uid=sa;pwd=;database=sales"
public myconn as sqlconnection=new sqlconnection(str)
public mydata as sqldataadapter
public myset as new dataset()
public mytable as new datatable()
'下面书写程序将数据库绑定到 Datagrid 控件，页面一运行就将数据库的记录显示出来
sub page_load(sender as object,e as eventargs)
if not ispostback then
    list()
end if
end sub
sub list()
    sql="select * from Employees"
    mydata=new sqldataadapter(sql,myconn)
    mydata.fill(myset,"Employees")
    mydatagrid.datasource=myset.tables("Employees")
    mydatagrid.databind()
end sub
</script>

<html>
<body>
<form runat="server">
<!-- 下面是定义 Datagrid 控件显示数据库记录 -->
<asp:datagrid ID="mydatagrid" runat="server" DataKeyField="编号"
AutoGenerateColumns="false" Width="100%">
<headerstyle HorizontalAlign="center"/>
<itemstyle HorizontalAlign="center"/>
<columns>
<asp:boundcolumn DataField="编号" HeaderText="编号"/>
<asp:boundcolumn DataField="姓名" HeaderText="姓名"/>
<asp:boundcolumn DataField="性别" HeaderText="性别"/>
<asp:boundcolumn DataField="部门" HeaderText="部门"/>
<asp:boundcolumn DataField="电话" HeaderText="电话"/>
<asp:boundcolumn DataField="地址" HeaderText="地址"/>
</columns>
</asp:datagrid>
</form>
</body>
</html>

```

(3) 测试运行：在浏览器的地址栏中输入 localhost/browser.aspx,运行结果如图 12.7。发现页面显示记录与 sales 数据库 Employees 表中的记录一致。

4.向数据库 sales 的 Employees 表中增加记录

(1) 创建动态页面 add.aspx，并保存到默认路径 C:\Inetpub\wwwroot\。

(2) 将第三部分的所有代码复制过来, 在其基础之上添加新的控件。通过“文本框”控件进行数据输入编号、姓名、电话和地址, 通过“单选按钮列表”控件进行选择性别, 通过“下拉列表”控件进行选择部门; 并在原程序基础上添加增加记录程序代码, 其中灰色部分是相对于第三部分新增的内容(程序 12.6)。

程序 12.2 add.aspx 增加 sales 库 Employees 表中的记录

```
<%@ Page Language="VB" ContentType="text/html" ResponseEncoding="gb2312" %>
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient"%>

<script runat="server">
'下面定义变量
dim sql as string
'下面定义读取数据库的对象, 连接数据库
dim str as string="server=(local);uid=sa;pwd=;database=sales"
public myconn as sqlconnection=new sqlconnection(str)
public mydata as sqldataadapter
public myset as new dataset()
public mytable as new datatable()
'下面书写程序将数据库绑定到 Datagrid 控件
sub page_load(sender as object,e as eventargs)
if not ispostback then
    list
end if
end sub
sub list()
    sql="select * from Employees"
    mydata=new sqldataadapter(sql,myconn)
    mydata.fill(myset,"Employees")
    mydatagrid.datasource=myset.tables("Employees")
    mydatagrid.databind()
end sub
'下面是向 sales 数据库 Employees 表增加数据记录, 比第三步中新增加的程序
sub mydatagrid_add (sender as object,e as eventargs)
    sql="insert into Employees(编号,姓名,性别,部门,电话,地址) values("
    sql &="'"& bh.text &";'"& xm.text &";'"& xb.selecteditem.value &""
    sql &=",'"& bm.selecteditem.value &";'"& dh.text &";'"& dz.text &";'"
    mydata=new sqldataadapter(sql,myconn)
    mydata.fill(myset,"tjsj")
    mydatagrid.datasource=myset.tables("tjsj")
    mydatagrid.databind()
    list()
end sub
</script>

<html>
<body>
<form runat="server">
<!-- 下面是定义 Datagrid 控件显示数据库记录 -->
<asp:datagrid ID="mydatagrid" runat="server" DataKeyField="编号"
AutoGenerateColumns="false" Width="100%">
<headerstyle HorizontalAlign="center"/>
<itemstyle HorizontalAlign="center"/>
```

```

<columns>
<asp:boundcolumn DataField="编号" HeaderText="编号"/>
<asp:boundcolumn DataField="姓名" HeaderText="姓名"/>
<asp:boundcolumn DataField="性别" HeaderText="性别"/>
<asp:boundcolumn DataField="部门" HeaderText="部门"/>
<asp:boundcolumn DataField="电话" HeaderText="电话"/>
<asp:boundcolumn DataField="地址" HeaderText="地址"/>
</columns>
</asp:datagrid>
<!--利用下面的控件添加记录，比三中新增加的控件 -->
编号: <asp:textbox ID="bh" runat="server" /><br>
姓名: <asp:textbox ID="xm" runat="server" /><br>
性别:
<asp:radiobuttonlist ID="xb" runat="server" RepeatDirection="Horizontal">
<asp:listitem Value="女"/>
<asp:listitem Value="男"/>
</asp:radiobuttonlist>
<br>
部门: <asp:dropdownlist ID="bm" runat="server">
<asp:listitem Value="销售科"/>
<asp:listitem Value="生产科"/>
<asp:listitem Value="后勤管理科"/>
<asp:listitem Value="仓管科"/>
<asp:listitem Value="会计科"/>
</asp:dropdownlist>
<br>
电话: <asp:textbox ID="dh" runat="server" />
<br>
地址: <asp:textbox ID="dz" runat="server" Columns="50"/>
<br>
<asp:button ID="tjxx" runat="server" Text="提交" OnClick=" mydatagrid_add " />
</form>
</body>
</html>

```

(3) 测试运行：在浏览器的地址栏中输入 localhost/add.aspx，然后输入或选择新添加的数据，单击“提交”按钮（操作过程和运行结果如图 12.8 和 12.9）。

5.修改 sales 数据库 Employees 表中的记录

(1) 创建动态页面 edit.aspx，并保存到路径 C:\inetpub\wwwroot\。

(2) 将第三部分的程序代码复制完全复制过来，在其基础上添加 Datagrid 编辑模板；添加修改 sales 数据库 Employees 表中记录的程序代码，其中灰色部分是相对于第三部分新增的内容（程序 12.3）。

程序 12.3 edit.aspx 修改 sales 库 Employees 表中的记录

```

<%@ Page Language="VB" ContentType="text/html" ResponseEncoding="gb2312" %>

<%@Import Namespace="System.Data" %>
<%@Import Namespace="System.Data.SqlClient"%>

```

```

<script runat="server">
'下面定义变量
dim sql as string
'下面定义读取数据库的对象，连接数据库
dim str as string="server=(local);uid=sa;pwd=;database=sales"
public myconn as sqlconnection=new sqlconnection(str)
public mydata as sqldataadapter
public myset as new dataset()
public mytable as new datatable()
'下面书写程序将数据库绑定到 Datagrid 控件
sub page_load(sender as object,e as eventargs)
if not ispostback then
    list
end if
end sub
sub list()
    sql="select * from Employees"
    mydata=new sqldataadapter(sql,myconn)
    mydata.fill(myset,"Employees")
    mydatagrid.datasource=myset.tables("Employees")
    mydatagrid.databind()
end sub
'下面是修改 sales 数据库 Employees 表中的记录
sub mydatagrid_edit(sender as object,e as datagridcommandeventargs)
mydatagrid.edittitemindex=cint(e.item.itemindex)
list()
end sub
sub mydatagrid_update(sender as object,e as datagridcommandeventargs)
dim bh,xm,xb,bm,dh,dz as textbox
bh=e.item.cells(0).controls(0)
xm=e.item.cells(1).controls(0)
xb=e.item.cells(2).controls(0)
bm=e.item.cells(3).controls(0)
dh=e.item.cells(4).controls(0)
dz=e.item.cells(5).controls(0)
sql="update Employees set 编号=" & bh.text & ",姓名=" & xm.text & ""
sql &=",性别=" & xb.text & ",部门=" & bm.text & ",电话=" & dh.text & ""
sql &=",地址=" & dz.text & ""
sql &=" where 编号=" & mydatagrid.datakeys(cint(e.item.itemindex))
mydata=new sqldataadapter(sql,myconn)
mydata.fill(myset,"xgjl")
mydatagrid.datasource=myset.tables("xgjl")
mydatagrid.databind()
mydatagrid.edittitemindex=-1
list()
end sub
sub mydatagrid_cancel(sender as object,e as datagridcommandeventargs)
mydatagrid.edittitemindex=-1
list()
end sub
</script>
<html>
<body>
<form runat="server">
<!-- 下面是定义 Datagrid 控件 -->
<asp:datagrid ID="mydatagrid" runat="server" DataKeyField="编号"
AutoGenerateColumns="false" Width="100%"

```

```

OnEditCommand="mydatagrid_edit"
OnUpdateCommand="mydatagrid_update"
OnCancelCommand="mydatagrid_cancel" >
<headerstyle HorizontalAlign="center"/>
<itemstyle HorizontalAlign="center"/>
<columns>
<asp:boundcolumn DataField="编号" HeaderText="编号"/>
<asp:boundcolumn DataField="姓名" HeaderText="姓名"/>
<asp:boundcolumn DataField="性别" HeaderText="性别"/>
<asp:boundcolumn DataField="部门" HeaderText="部门"/>
<asp:boundcolumn DataField="电话" HeaderText="电话"/>
<asp:boundcolumn DataField="地址" HeaderText="地址"/>
<asp:editcommandcolumn HeaderText="修改" EditText="修改" UpdateText="保存"
CancelText="取消"/>
</columns>
</asp:datagrid>
</form>
</body>
</html>

```

(3) 测试运行：在浏览器的地址栏中输入 localhost/edit.aspx，运行后单击某一记录后面“修改”列中的“修改”按钮，比如修改第五条记录中的“电话”为“(010) 65554444”，然后单击“保存”按钮观察效果（如图 12.10、图 12.11 和图 12.12）。

6. 删除 sales 数据库 Employees 表中的记录

(1) 创建动态页面 delete.aspx，并保存到路径 C:\inetpub\wwwroot\。

(2) 将第三部分的程序代码完全复制过来，在其基础上添加 Datagrid 的删除模板；添加程序删除 sales 数据库 Employees 表中记录的程序代码，其中灰色部分是相对于第三部分新增的内容（程序 12.4）。

程序 12.4 delete.aspx 删除 sales 库 Employees 表中的记录

```

<%@ Page Language="VB" ContentType="text/html" ResponseEncoding="gb2312" %>

<%@Import Namespace="System.Data" %>
<%@Import Namespace="System.Data.SqlClient"%>

<script runat="server">
'下面定义变量
dim sql as string
'下面定义读取数据库的对象，连接数据库
dim str as string="server=(local);uid=sa;pwd=;database=sales"
public myconn as sqlconnection=new sqlconnection(str)
public mydata as sqldataadapter
public myset as new dataset()
public mytable as new datatable()
'下面书写程序将数据库绑定到 Datagrid 控件

```

```

sub page_load(sender as object,e as eventargs)
if not ispostback then
list()
end if
end sub
sub list()
sql="select * from Employees"
mydata=new sqldataadapter(sql,myconn)
mydata.fill(myset,"Employees")
mydatagrid.datasource=myset.tables("Employees")
mydatagrid.databind()
end sub

```

'下面是删除记录里面数据的程序代码

```

sub mydatagrid_delete(sender as object,e as datagridcommandeventargs)
sql="delete from Employees where 编号="
sql &=mydatagrid.datakeys(cint(e.item.itemindex))
mydata=new sqldataadapter(sql,myconn)
mydata.fill(myset,"scjl")
mydatagrid.datasource=myset.tables("scjl")
mydatagrid.databind()
list()
end sub

```

</script>

<html>

<body>

<form runat="server">

<!-- 下面是定义 Datagrid 控件显示数据库记录 -->

<asp:datagrid ID="mydatagrid" runat="server" DataKeyField="编号"

AutoGenerateColumns="false" Width="100%"

OnDeleteCommand="mydatagrid_delete">

<headerstyle HorizontalAlign="center"/>

<itemstyle HorizontalAlign="center"/>

<columns>

<asp:boundcolumn DataField="编号" HeaderText="编号"/>

<asp:boundcolumn DataField="姓名" HeaderText="姓名"/>

<asp:boundcolumn DataField="性别" HeaderText="性别"/>

<asp:boundcolumn DataField="部门" HeaderText="部门"/>

<asp:boundcolumn DataField="电话" HeaderText="电话"/>

<asp:boundcolumn DataField="地址" HeaderText="地址"/>

<asp:buttoncolumn CommandName="delete" HeaderText="删除" Text="删除"/>

</columns>

</asp:datagrid>

</form>

</body>

</html>

(3) 测试运行：在浏览器的地址栏中输入 localhost/delete.aspx，运行后单击某一条记录后面“删除”列中的“删除”按钮，观察效果（如图 12.13 和图 12.14）。

7. 查询 sales 数据库 Employees 表中的记录

(1) 创建动态页面 search.aspx，并保存到路径 C:\inetpub\wwwroot\。

(2) 将第三步的程序代码复制完全复制过来，在其基础上添加一个文本框控件，一个按钮控件；添加程序代码，其中灰色部分是相对于第三部分新增的内容（程序 12.5）。

程序 12.5 search.aspx 查询 sales 库 Employees 表中的记录

```
<%@ Page Language="VB" ContentType="text/html" ResponseEncoding="gb2312" %>
<%@Import Namespace="System.Data" %>
<%@Import Namespace="System.Data.SqlClient"%>

<script runat="server">
'下面定义变量
dim sql as string
'下面定义读取数据库的对象，连接数据库
dim str as string="server=(local);uid=sa;pwd=;database=sales"
public myconn as sqlconnection=new sqlconnection(str)
public mydata as sqldataadapter
public myset as new dataset()
public mytable as new datatable()
'下面书写程序将数据库绑定到 Datagrid 控件
sub page_load(sender as object,e as eventargs)
if not ispostback then
list()
end if
end sub
sub list()
sql="select * from Employees"
mydata=new sqldataadapter(sql,myconn)
mydata.fill(myset,"Employees")
mydatagrid.datasource=myset.tables("Employees")
mydatagrid.databind()
end sub
'下面是查询 Employees 表中以字段姓名为关键字的记录
sub mydatagrid_search(sender as object,e as eventargs)
sql="select * from Employees where 姓名='"& trim(xm.text) &'"
mydata=new sqldataadapter(sql,myconn)
mydata.fill(myset,"xxcx")
mydatagrid.datasource=myset.tables("xxcx")
mydatagrid.databind()
end sub
</script>

<html>
<body>
<form runat="server">
<!--下面是查询记录时用到的控件-->
姓名: <asp:textbox ID="xm" runat="server" />
<asp:button ID="xxcx" runat="server" Text="查询">
```

```

OnClick="mydatagrid_search" />
<!-- 下面是定义 Datagrid 控件显示数据库记录 -->
<asp:datagrid ID="mydatagrid" runat="server" DataKeyField="编号"
AutoGenerateColumns="false" Width="100%">
<headerstyle HorizontalAlign="center"/>
<itemstyle HorizontalAlign="center"/>
<columns>
<asp:boundcolumn DataField="编号" HeaderText="编号"/>
<asp:boundcolumn DataField="姓名" HeaderText="姓名"/>
<asp:boundcolumn DataField="性别" HeaderText="性别"/>
<asp:boundcolumn DataField="部门" HeaderText="部门"/>
<asp:boundcolumn DataField="电话" HeaderText="电话"/>
<asp:boundcolumn DataField="地址" HeaderText="地址"/>
</columns>
</asp:datagrid>
</form>
</body>
</html>

```

(3) 测试运行：在浏览器的地址栏中输入 localhost/search.aspx，在姓名后面的文本框中输入测试想查询的姓名（如测试 3），单击“查询”按钮，观察效果（如图 12.15 和 12.16）。

8. 综合程序代码

(1) 下面综合了以上各项功能的完整程序代码（如程序 12.6）。

程序 12.6 综合各项功能的程序代码

```

<%@ Page Language="VB" ContentType="text/html" ResponseEncoding="gb2312" %>
<%@Import Namespace="System.Data" %>
<%@Import Namespace="System.Data.SqlClient"%>

<script runat="server">
'下面定义变量
dim sql as string
'下面定义读取数据库的对象，连接数据库
dim str as string="server=(local);uid=sa;pwd=;database=sales"
public myconn as sqlconnection=new sqlconnection(str)
public mydata as sqldataadapter
public myset as new dataset()
public mytable as new datatable()

'下面书写程序将数据库绑定到 Datagrid 控件
sub page_load(sender as object,e as eventargs)
if not ispostback then
    list()
end if
end sub
sub list()
    sql="select * from Employees"
    mydata=new sqldataadapter(sql,myconn)
    mydata.fill(myset,"Employees")

```

```

        mydatagrid.datasource=myset.tables("Employees")
        mydatagrid.databind()
    end sub
'下面是向 sales 数据库 Employees 表增加数据记录
sub mydatagrid_add(sender as object,e as eventargs)
    sql="insert into Employees(编号,姓名,性别,部门,电话,地址) values("
    sql &="'"& bh.text &"','"& xm.text &"','"& xb.selecteditem.value &"'"
    sql &="','"& bm.selecteditem.value &"','"& dh.text &"','"& dz.text &"'"
    mydata=new sqldataadapter(sql,myconn)
    mydata.fill(myset,"tjsj")
    mydatagrid.datasource=myset.tables("tjsj")
    mydatagrid.databind()
    list()
end sub
'下面是修改 sales 数据库 Employees 表中的记录
sub mydatagrid_edit(sender as object,e as datagridcommandeventargs)
    mydatagrid.edititemindex=cint(e.item.itemindex)
    list()
end sub
sub mydatagrid_update(sender as object,e as datagridcommandeventargs)
    dim bh,xm,xb,bm,dh,dz as textbox
    bh=e.item.cells(0).controls(0)
    xm=e.item.cells(1).controls(0)
    xb=e.item.cells(2).controls(0)
    bm=e.item.cells(3).controls(0)
    dh=e.item.cells(4).controls(0)
    dz=e.item.cells(5).controls(0)
    sql="update Employees set 编号='"& bh.text &"',姓名='"& xm.text &"'"
    sql &="',性别='"& xb.text &"',部门='"& bm.text &"',电话='"& dh.text &"'"
    sql &="',地址='"& dz.text &"'"
    sql &=" where 编号='"&mydatagrid.datakeys(cint(e.item.itemindex))
    mydata=new sqldataadapter(sql,myconn)
    mydata.fill(myset,"xgjl")
    mydatagrid.datasource=myset.tables("xgjl")
    mydatagrid.databind()
    mydatagrid.edititemindex=-1
    list()
end sub
sub mydatagrid_cancel(sender as object,e as datagridcommandeventargs)
    mydatagrid.edititemindex=-1
    list()
end sub
'下面是删除记录里面数据的程序代码
sub mydatagrid_delete(sender as object,e as datagridcommandeventargs)
    sql="delete from Employees where 编号="
    sql &="'"&mydatagrid.datakeys(cint(e.item.itemindex))
    mydata=new sqldataadapter(sql,myconn)
    mydata.fill(myset,"scjl")
    mydatagrid.datasource=myset.tables("scjl")
    mydatagrid.databind()
    list()
end sub
'下面是查询 Employees 表中以字段姓名为关键字的记录
sub mydatagrid_search(sender as object,e as eventargs)
    sql="select * from Employees where 姓名='"& trim(xm1.text) &"'"
    mydata=new sqldataadapter(sql,myconn)
    mydata.fill(myset,"xxcx")
    mydatagrid.datasource=myset.tables("xxcx")

```

```

        mydatagrid.databind()
    end sub
</script>

<html>
<body>
<form runat="server">
<!-- 下面是查询记录时用到的控件-->
姓名: <asp:textbox ID="xm1" runat="server" />
<asp:button ID="xxcx" runat="server" Text="查询" OnClick="mydatagrid_search" />
<!-- 下面是定义 Datagrid 控件显示数据库记录 -->
<asp:datagrid ID="mydatagrid" runat="server" DataKeyField="编号"
AutoGenerateColumns="false" Width="100%"
OnEditCommand="mydatagrid_edit"
OnUpdateCommand="mydatagrid_update"
OnCancelCommand="mydatagrid_cancel"
OnDeleteCommand="mydatagrid_delete">
<headerstyle HorizontalAlign="center"/>
<itemstyle HorizontalAlign="center"/>
<columns>
<asp:boundcolumn DataField="编号" HeaderText="编号"/>
<asp:boundcolumn DataField="姓名" HeaderText="姓名"/>
<asp:boundcolumn DataField="性别" HeaderText="性别"/>
<asp:boundcolumn DataField="部门" HeaderText="部门"/>
<asp:boundcolumn DataField="电话" HeaderText="电话"/>
<asp:boundcolumn DataField="地址" HeaderText="地址"/>
<asp:editcommandcolumn HeaderText="修改" EditText="修改" UpdateText="保存"
CancelText="取消"/>
<asp:buttoncolumn CommandName="delete" HeaderText="删除" Text="删除"/>
</columns>
</asp:datagrid>
<!-- 利用下面的控件添加记录-->
编号: <asp:textbox ID="bh" runat="server" /><br>
姓名: <asp:textbox ID="xm" runat="server" /><br>
性别:
<asp:radiobuttonlist ID="xb" runat="server" RepeatDirection="Horizontal">
<asp:listitem Value="女"/>
<asp:listitem Value="男"/>
</asp:radiobuttonlist>
<br>
部门: <asp:dropdownlist ID="bm" runat="server">
<asp:listitem Value="销售科"/>
<asp:listitem Value="生产科"/>
<asp:listitem Value="后勤管理科"/>
<asp:listitem Value="仓管科"/>
<asp:listitem Value="会计科"/>
</asp:dropdownlist>
<br>
电话: <asp:textbox ID="dh" runat="server" />
<br>
地址: <asp:textbox ID="dz" runat="server" Columns="50"/>
<br>
<asp:button ID="tjxx" runat="server" Text="提交" OnClick=" mydatagrid_add " />
</form>
</body>
</html>

```

(2) 测试运行：在浏览器的地址栏里面输入 localhost/allprogram.aspx，运行结果如图 12.17。

12.4.2 VB.NET 程序设计

1.主窗体设计

(1) 从“工具箱”向”主窗体”添加六个标签控件  Label，在”属性窗口”中将标签的 Text 值分别改为“编号:”、“姓名:”、“性别:”、“部门:”、“电话:”和“地址 ”。

(2) 在上述各标签后添加控件，并设置各控件属性，如表 12.2 所示。

表 12.2 添加控件

标签	控件	属性设置
编号	 TextBox	Text 属性值清空，Name 属性值改为“bh”
姓名	 TextBox	Text 属性值清空，Name 属性值改为“xm”
性别	 RadioButton	Text 属性值改为“女”，Name 属性值改为“nv”
性别	 RadioButton	Text 属性值改为“男”，Name 属性值改为“nan”
部门	 ComboBox	Text 属性值改为“[请选择部门]”，Name 属性值改为“bm”； 单击 Items 后面的“(Collection)”弹出”字符串集合编辑器”面板，输入如图 12.21 的选项
电话	 TextBox	Text 属性值清空，Name 属性值改为“dh”
地址	 TextBox	Text 属性值清空，Name 属性值改为“dz”

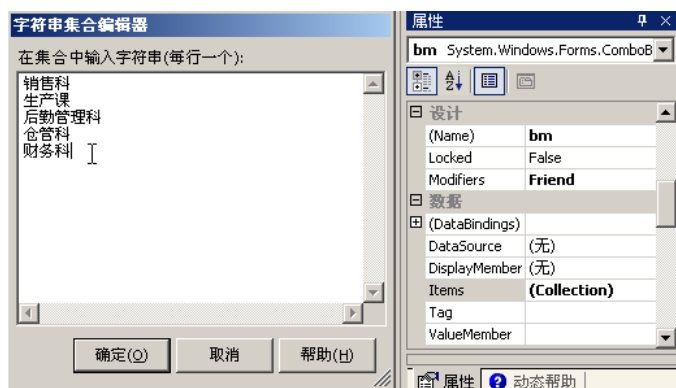


图 12.21 添加 ComboBox 里面的选项

(3) 添加 Datagrid 控件， DataGrid，用来显示数据库 sales 中 Employees 表的数据记录。在“属性窗口”中将 Name 属性值改为“mydatagrid”。

(4) 添加 6 个按钮控件  Button。“增加”按钮的属性中 Name 的属性值为“add1”，Text 的属性值为“增加”。“保存”按钮的属性中 Name 的属性值为“save1”，Text 的属性值为“保存”。“取消”按钮的属性中 Name 的属性值为“cancel1”，Text 的属性值为“取消”。“修改”按钮的属性中 Name 的属性值为“edit1”，Text 的属性值为“修改”。“删除”按钮的属性中 Name 的属性值为“delete1”，Text 的属性值为“删除”。“退出”按钮的属性中 Name 的属性值为“exit1”，Text 的属性值为“退出”。设计完的主窗体样式如图 12.22。

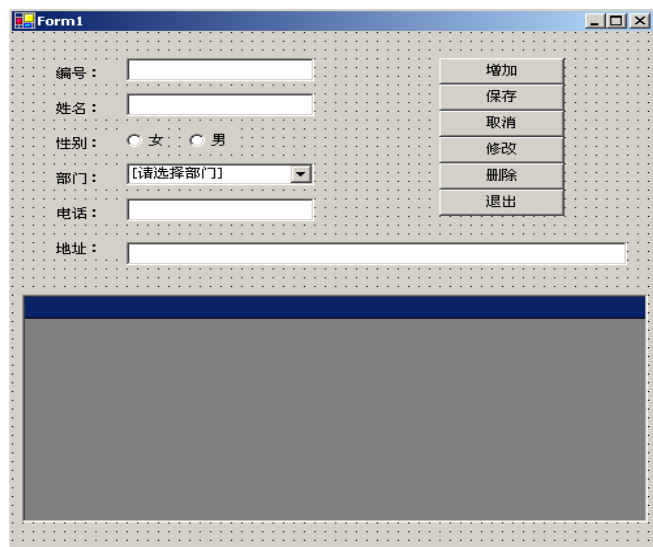


图 12.22 主窗体

2.导入名称空间

因为我们要在程序中操作数据库，所以需要导入名称空间。

(1) 双击主窗体，进入代码设计界面。

(2) 导入名称空间：在界面最上部导入名称空间（如图 12.23）。

```
Imports System.Data
```

```
Imports System.Data.SqlClient
```

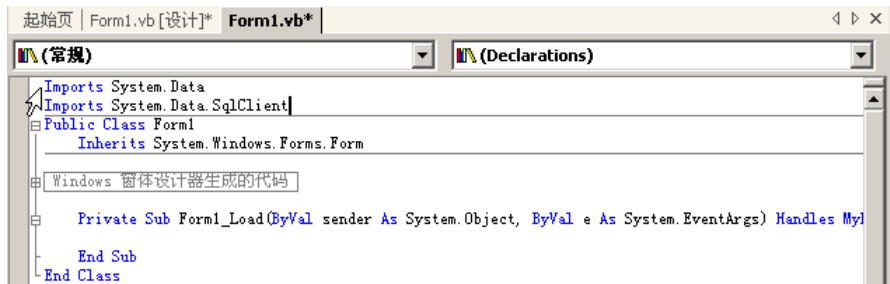


图 12.23 导入名称空间

3.程序设计

(1) 窗体初始化程序设计

本程序主要实现将 Datagrid 控件和数据库进行绑定，当项目一启动的时候，用 Datagrid 控件来显示 sales 数据库中 Employees 表中的记录。并将“保存”按钮“取消”按钮设为不可操作，因为只有单击了“增加”按钮后，才可以对“保存”按钮和“取消”按钮进行操作（程序 12.7）。按键盘上“F5”键启动程序，运行结果如图 12.24。

程序 12.7 窗体初始化程序代码

```
Imports System.Data
```

```
Imports System.Data.SqlClient
```

```
Public Class Form1
```

```
Inherits System.Windows.Forms.Form
```

```
Windows 窗体设计器生成的代码
```

```
'调用过程list()实现窗体初始化
```

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
```

```
MyBase.Load
```

```
list() '调用list()过程
```

```
End Sub
```

```
'下面的程序代码实现界面初始化
```

```
Private Sub list()
```

```

'下面是datagrid与数据库绑定程序，首先是建立数据库连接，然后定义各操作数据库的对象
Dim myconn As SqlConnection = New SqlConnection("server=(local);uid=sa;pwd=;database=sales")
Dim mydata As SqlDataAdapter
Dim myset As New DataSet
Dim mytable As New DataTable
Dim sql As String
sql = "select * from employees order by 编号"
mydata = New SqlDataAdapter(sql, myconn)
mydata.Fill(myset, "employees")
mydatagrid.DataSource = myset.Tables("employees")
'下面是按钮初始操作设置
save1.Enabled = False      '设保存按钮为不可操作，呈灰色
cancel1.Enabled = False    '设取消按钮为不可操作，呈灰色
add1.Enabled = True        '增加按钮为可操作
edit1.Enabled = True       '修改按钮为可操作
delete1.Enabled = True     '删除按钮为可操作
End Sub

```

编号	姓名	性别	部门	电话	地址
0101	张颖	女	销售科	(010) 65559	复兴门 245
0102	王伟	男	销售科	(010) 65559	罗马花园 85
0201	李芳	女	生产科	(010) 65553	芍药园小区
0301	赵军	男	后勤管理科	(010) 65558	前门大街 75

图12.24 窗体初始化运行结果

(2) 单击 Datagrid 一行所触发事件过程 mydatagrid_Click 程序设计
 在“类”下拉列表中选择 mydatagrid，在“方法名称”下拉列表中选择 Click 方法
 (如图 12.24)。然后在所形成的事件里面书写代码。



图 12.25 选择方法

本步骤要实现如下功能,当单击 **Datagrid** 中的某一记录左边时,触发一个事件过程,将本行的字段值提取出来显示在上面的文本框、单选按钮和下拉列表里面,然后可以修改或删除该记录(程序 12-8)。按键盘上“F5”键启动程序,操作结果如图 12.26。

程序12-8 单击datagrid控件的记录所触发的事件过程程序代码

```
Private Sub mydatagrid_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles_ mydatagrid.Click
    bh.Enabled = False ' 设置编号的文本框为不可操作项, 在修改记录时此字段不可修改
    Dim i As Integer
    ' 下面是进行数据库连接, 并定义操作数据库的各个对象
    Dim myconn As SqlConnection = New SqlConnection ("server=(local); uid=sa; pwd=;
database=sales")
    Dim mydata As SqlDataAdapter
    Dim myset As New DataSet
    Dim mytable As New DataTable
    Dim sql As String = "select * from employees"
    mydata = New SqlDataAdapter(sql, myconn)
    mydata.Fill(myset, "employees2")
    mytable = myset.Tables("employees2")
    For i = 0 To mytable.Rows.Count - 1
        If mydatagrid.IsSelected(i) Then
            bh.Text = mytable.Rows(i).Item(0)
            xm.Text = mytable.Rows(i).Item(1)
            If mytable.Rows(i).Item(2) = "女" Then
                nv.Checked = True
            ElseIf mytable.Rows(i).Item(2) = "男" Then
                nan.Checked = True
            End If
            bm.Text = mytable.Rows(i).Item(3)
            dh.Text = mytable.Rows(i).Item(4)
            dz.Text = mytable.Rows(i).Item(5)
        End If
    Next
End Sub
```

(3) 增加数据库记录

①单击“增加”按钮触发事件过程 **add1_Click**。当单击“增加”按钮后,“增加”按钮、“修改”按钮、“删除”按钮呈灰色不可操作,“保存”按钮和“取消”按钮变

成可操作（程序 12.9）；同时编号、姓名、电话和地址后面的文本框被清空，性别默认为“女”，部门的值默认为“[请选择部门]”，然后就可以在编号、姓名、电话和地址后面的文本框里面输入内容，选择性别，选择部门。按键盘上“F5”键启动程序，操作结果如图 12.7。

程序12.9 单击”增加”按钮所触发的事件过程程序代码

```
Private Sub add1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles add1.Click
    bh.Text = "" ' 编号后的文本框清空
    bh.Enabled = True ' 编号为可操作项
    xm.Text = "" ' 姓名后的文本框清空
    nv.Checked = True ' 设置单选按钮女为默认选项
    nan.Checked = False ' 设置单选按钮男为非选项
    bm.Text = "[请选择部门]" ' 部门后的下拉列表初值为“[请选择部门]”
    dh.Text = "" ' 电话后的文本框清空
    dz.Text = "" ' 地址后的文本框清空
    save1.Enabled = True ' 保存按钮为可操作
    cancell.Enabled = True ' 取消按钮为可操作
    add1.Enabled = False ' 增加按钮为不可操作，呈灰色
    edit1.Enabled = False ' 编辑按钮为不可操作，呈灰色
    delet1.Enabled = False ' 删除按钮为不可操作，呈灰色
End Sub
```

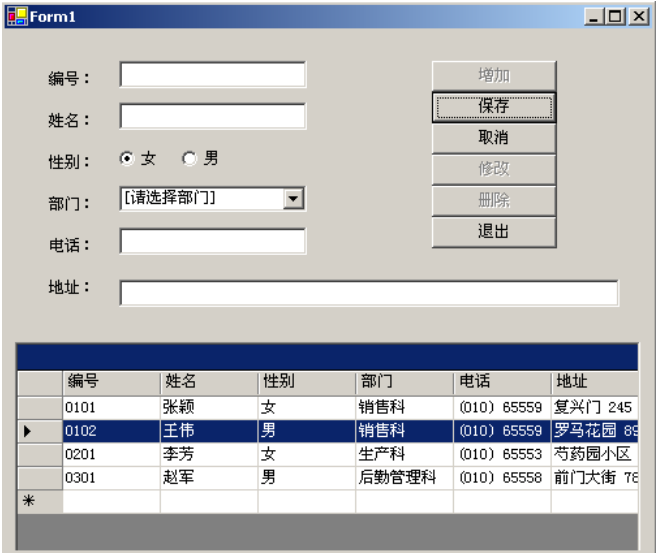


图 12.27 单击”增加”按钮的运行结果

②单击“保存”按钮触发事件过程 save1_Click。当单击“保存”按钮时，会弹出一个对话框，假如选择“是”的话，输入的编号、姓名、电话、地址，选择的性别和部门数据将被保存到 sales 数据库的 Employees 表中，同时“保存”按钮和“取消”按钮为不可操作，呈灰色；假如选择“否”，则回到原状态（程序 12.10）。按键盘上“F5”键启动程序，运行结果如图 12.28 和图 12.29。

程序12.10 单击“保存”按钮所触发事件过程的程序代码

```
Private Sub save1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
save1.Click
If bh.Text = "" Or xm.Text = "" Or dh.Text = "" Or dz.Text = "" Then
MsgBox("编号、姓名、电话和地址均不能为空！")
Else
Dim yn As Integer
yn = MsgBox("是否保存新增加的记录？", MsgBoxStyle.YesNo, "增加记录")
If yn = 6 Then
Dim myconn As SqlConnection = New SqlConnection("server=(local);uid=sa;pwd=;database=sales")
Dim mydata As SqlDataAdapter
Dim myset As New DataSet
Dim mytable As New DataTable
Dim sql, str As String
If nv.Checked Then
str = "女"
ElseIf nan.Checked Then
str = "男"
End If
sql = "insert into employees(编号,姓名,性别,部门,电话,地址) values("
sql &= "" & bh.Text & "," & xm.Text & "," & str & "," & bm.Text & ""
sql &= "," & dh.Text & "," & dz.Text & ")"
mydata = New SqlDataAdapter(sql, myconn)
mydata.Fill(myset, "save")
mydatagrid.DataSource = myset.Tables("save")
bh.Enabled = True
list()
ElseIf yn = 7 Then
bh.Enabled = True
list()
End If
End If
End Sub
```

③单击“取消”按钮触发事件 cancel1_Click，调用过程 list()，进行窗体初始化（程序 12.11）。

程序12.11 单击“取消”按钮触发的事件过程程序代码

```
Private Sub cancel1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles  
    cancel1.Click  
    list() '单击取消按钮调用list()过程  
End Sub
```

（4）修改数据库记录

当单击 datagrid 控件某一行时，该记录的数据将被传递到上面的文本框、单选按钮和下拉列表里面，然后可以进行修改里面的数据。修改完毕，单击“修改”按钮将新数据保存到 sales 数据库的 Employees 表中（程序 12.12）。按键盘上“F5”键启动程序，程序运行结果如图 12.30、图 12.31 和图 12.32。比如单击编号为“0401”这条记录，然后将“电话”修改为“(010) 65554444”，然后单击“修改”按钮。

程序12.12 单击“修改”按钮触发的程序代码

```
Private Sub edit1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles  
    edit1.Click  
If xm.Text = "" Or dh.Text = "" Or dz.Text = "" Then  
    MsgBox("姓名、电话和地址均不能为空！")  
Else  
    Dim yn As Integer  
    yn = MsgBox("是否更新记录？", MsgBoxStyle.YesNo, "更新记录")  
    If yn = 6 Then  
        Dim myconn As SqlConnection = New SqlConnection("server=(local);uid=sa;pwd=;database=sales")  
        Dim mydata As SqlDataAdapter  
        Dim myset As New DataSet  
        Dim mytable As New DataTable  
        Dim sql, str As String  
        If nv.Checked Then  
            str = "女"  
        ElseIf nan.Checked Then  
            str = "男"  
        End If  
        sql = "update employees set 姓名='" & xm.Text & "',性别='" & str & ""  
        sql &= ",部门='" & bm.Text & "',电话='" & dh.Text & "',地址='" & dz.Text & ""  
        sql &= " where 编号='" & bh.Text & ""
```

```

mydata = New SqlDataAdapter(sql, myconn)
mydata.Fill(myset, "update")
mydatagrid.DataSource = myset.Tables("update")
list()
bh.Enabled = True
ElseIf yn = 7 Then
    list()
    bh.Enabled = True
End If
End If
End Sub

```

(5) 删除数据库记录

当单击 **datagrid** 控件某一行时，该记录的数据将被传递到上面的文本框、单选按钮和下拉列表里面，然后单击“删除”按钮将删除 **sales** 数据库的 **Employees** 表中的该记录（程序 12.13）。按键盘上“F5”键启动程序，程序运行结果如图 12.33 和图 12.34。比如单击编号为“0401”的记录，然后单击“删除”按钮，就会发现少了这条记录。

程序12.13 单击“删除”按钮删除数据库记录触发事件程序代码

```

Private Sub delete1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    delete1.Click
    Dim yn As Integer
    yn = MsgBox("是否删除记录？", MsgBoxStyle.YesNo, "删除记录")
    If yn = 6 Then
        Dim myconn As SqlConnection = New &_
        SqlConnection("server=(local);uid=sa;pwd=;database=sales")
        Dim mydata As SqlDataAdapter
        Dim myset As New DataSet
        Dim mytable As New DataTable
        Dim sql As String
        sql = "delete from employees where 编号='" & bh.Text & "'"
        mydata = New SqlDataAdapter(sql, myconn)
        mydata.Fill(myset, "delete")
        mydatagrid.DataSource = myset.Tables("delete")
        list()
        bh.Enabled = True
    ElseIf yn = 7 Then
        list()
        bh.Enabled = True
    End If
End Sub

```

```
End If
End Sub
```

（6）退出窗体

当单击“退出”按钮后，将退出该窗体（程序 12.14）。

程序12.14 单击“退出”按钮触发事件程序代码

```
Private Sub exit1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    exit1.Click
    Close() '单击退出按钮退出窗体
End Sub
```

（7）综合程序代码

以下为本示例的完整程序代码，实现了以上各项功能（程序 12.15）。

程序 12.15 完整程序代码

```
Imports System.Data

Imports System.Data.SqlClient
Public Class Form1
Inherits System.Windows.Forms.Form

Windows 窗体设计器生成的代码

'调用过程list()实现窗体初始化
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    MyBase.Load
    list()
End Sub
'下面的程序代码实现界面初始化
Private Sub list()
    'datagrid与数据库绑定
    Dim myconn As SqlConnection = New SqlConnection("server=(local);uid=sa;pwd=;database=sales")
    Dim mydata As SqlDataAdapter
    Dim myset As New DataSet
    Dim mytable As New DataTable
    Dim sql As String
    sql = "select * from employees order by 编号"
```

```

mydata = New SqlDataAdapter(sql, myconn)
mydata.Fill(myset, "employees")
mydatagrid.DataSource = myset.Tables("employees")
'按钮操作设置
save1.Enabled = False '设保存按钮为不可操作
cancel1.Enabled = False '设取消按钮为不可操作
add1.Enabled = True '增加按钮为可操作
edit1.Enabled = True '修改按钮为可操作
delete1.Enabled = True '删除按钮为可操作
End Sub

Private Sub add1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles dd1.Click
    bh.Text = ""
    xm.Text = ""
    nv.Checked = True
    nan.Checked = False
    bm.Text = "[请选择部门]"
    dh.Text = ""
    dz.Text = ""
    save1.Enabled = True
    cancel1.Enabled = True
    add1.Enabled = False
    edit1.Enabled = False
    delete1.Enabled = False
End Sub

Private Sub exit1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    exit1.Click
    Close()
End Sub

Private Sub cancel1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    cancel1.Click
    list()
End Sub

Private Sub save1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    save1.Click
    If bh.Text = "" Or xm.Text = "" Or dh.Text = "" Or dz.Text = "" Then
        MsgBox("编号、姓名、电话和地址均不能为空！")
    Else

```

```

Dim yn As Integer
yn = MsgBox("是否保存新增加的记录？ ", MsgBoxStyle.YesNo, "增加记录")
If yn = 6 Then
    Dim myconn As SqlConnection = New
    SqlConnection("server=(local);uid=sa;pwd=;database=sales")
    Dim mydata As SqlDataAdapter
    Dim myset As New DataSet
    Dim mytable As New DataTable
    Dim sql, str As String
    If nv.Checked Then
        str = "女"
    ElseIf nan.Checked Then
        str = "男"
    End If
    sql = "insert into employees(编号,姓名,性别,部门,电话,地址) values("
    sql &= """" & bh.Text & "," & xm.Text & "," & str & "," & bm.Text & """"
    sql &= "," & dh.Text & "," & dz.Text & ")"
    mydata = New SqlDataAdapter(sql, myconn)
    mydata.Fill(myset, "save")
    mydatagrid.DataSource = myset.Tables("save")
    bh.Enabled = True
    list()
ElseIf yn = 7 Then
    bh.Enabled = True
    list()
End If
End If
End Sub

Private Sub edit1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    edit1.Click
    If xm.Text = "" Or dh.Text = "" Or dz.Text = "" Then
        MsgBox("姓名、电话和地址均不能为空！ ")
    Else
        Dim yn As Integer
        yn = MsgBox("是否更新记录？ ", MsgBoxStyle.YesNo, "更新记录")
        If yn = 6 Then
            Dim myconn As SqlConnection = New
            SqlConnection("server=(local);uid=sa;pwd=;database=sales")

```

```

Dim mydata As SqlDataAdapter
Dim myset As New DataSet
Dim mytable As New DataTable
Dim sql, str As String
If nv.Checked Then
    str = "女"
ElseIf nan.Checked Then
    str = "男"
End If
sql = "update employees set 姓名=" & xm.Text & ",性别=" & str & ""
sql &= ",部门=" & bm.Text & ",电话=" & dh.Text & ",地址=" & dz.Text & ""
sql &= " where 编号=" & bh.Text & ""
mydata = New SqlDataAdapter(sql, myconn)
mydata.Fill(myset, "update")
mydatagrid.DataSource = myset.Tables("update")
list()
bh.Enabled = True
ElseIf yn = 7 Then
    list()
    bh.Enabled = True
End If
End If
End Sub
Private Sub delete1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    delete1.Click
Dim yn As Integer
yn = MsgBox("是否删除记录？", MsgBoxStyle.YesNo, "删除记录")
If yn = 6 Then
    Dim myconn As SqlConnection = New
    SqlConnection("server=(local);uid=sa;pwd=;database=sales")
    Dim mydata As SqlDataAdapter
    Dim myset As New DataSet
    Dim mytable As New DataTable
    Dim sql As String
    sql = "delete from employees where 编号=" & bh.Text & ""
    mydata = New SqlDataAdapter(sql, myconn)
    mydata.Fill(myset, "delete")
    mydatagrid.DataSource = myset.Tables("delete")
    list()

```

```

        bh.Enabled = True
    ElseIf yn = 7 Then
        list()
        bh.Enabled = True
    End If
End Sub

Private Sub mydatagrid_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles
    mydatagrid.Click
    bh.Enabled = False '设置编号的文本框为不可操作项，在修改记录时此字段不可修改
    Dim i As Integer
    Dim myconn As SqlConnection = New SqlConnection("server=(local);uid=sa;pwd=;database=sales")
    Dim mydata As SqlDataAdapter
    Dim myset As New DataSet
    Dim mytable As New DataTable
    Dim sql As String = "select * from employees"
    mydata = New SqlDataAdapter(sql, myconn)
    mydata.Fill(myset, "employees2")
    mytable = myset.Tables("employees2")
    For i = 0 To mytable.Rows.Count - 1
        If mydatagrid.IsSelected(i) Then
            bh.Text = mytable.Rows(i).Item(0)
            xm.Text = mytable.Rows(i).Item(1)
            If mytable.Rows(i).Item(2) = "女" Then
                nv.Checked = True
            ElseIf mytable.Rows(i).Item(2) = "男" Then
                nan.Checked = True
            End If
            bm.Text = mytable.Rows(i).Item(3)
            dh.Text = mytable.Rows(i).Item(4)
            dz.Text = mytable.Rows(i).Item(5)
        End If
    Next
End Sub
End Class

```

4.测试运行