第 6 章　SQL Server 数据库系统

本章学习目标

本章主要介绍 SQL Server 的使用和开发。通过本章学习，读者应该掌握以下内容:

- 了解 SQL Server
- SQL Server 管理工具使用方法
- SQL Server 中数据库的创建与维护
- 了解 Transact-SQL 语言
- 熟悉 SQL Server 中存储过程的使用
- 熟悉 SQL Server 中触发器的使用
- SQL Server 数据导入、导出
- SQL Server 的应用系统开发环境

6.1　SQL Server 管理工具

SQL Server 是 Microsoft 公司在原来和 Sybase 公司合作的基础上打包出的一款面向高端的数据库系统，定位于 Internet 背景下的基于 Windows 的数据库的应用。经过不断的更新换代，已发展到了 SQL Server 2008，具有性能高、功能强、安全性好、易操作、易维护等特点，为用户的 Web 应用提供了一款完善的数据管理和数据分析解决方案。

为方便应用，各种数据库都提供管理工具，采用可视化方式提供服务，一般建立数据库、建立表、建立视图、基本查询、存储过程、触发器、简单报表等操作都可以利用其管理工具完成。

SQL Server 2000 的管理工具包括查询分析器、导入和导出数据、服务管理器、服务器网络实用工具、客户端网络实用工具、企业管理器、事件探查器等。SQL Server 2005 的管理工具将企业管理器、查询分析器、数据转换、报表服务等合并为管理控制台 Management Studio，又加入了分析服务、配置工具、性能工具。考虑到易学易入门，本节仍借助 SQL Server 2000 的管理工具进行介绍，如图 6.1 所示。

6.1.1　服务管理器

在安装 SQL Server 后，一般在计算机开机后会自动打开 SQL Server 数据库，用户自己建立的应用系统可以直接对 SQL Server 数据库进行操作。可以应用服务管理器（如图 6.2 所示）实现暂停、停止、再启动数据库等操作，其服务包括：distributed transaction coordinator（分布式的事务处理）、Microsoft Search（搜索）、SQL Server（SQL 服务器）、SQL Server agent（SQL 服务代理）等。如果不要求开机时自动启动 SQL Server，可将“当启动 OS 时自动启动服务”复选框取消勾选。服务器上的箭头为绿色时表示服务器启动，红色表示停止。

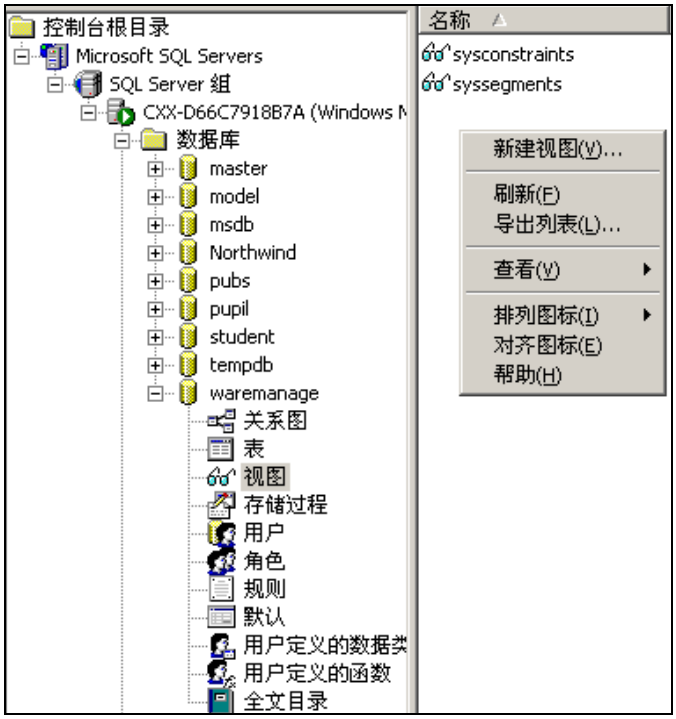

图 6.1　SQL Server 2000 的管理工具组成

图 6.2　服务管理器

6.1.2　建立数据库、表、索引的操作

企业管理器以树形结构显示并管理所有 SQL Server 对象，包括数据库、表、视图、存储过程、规则、创建与管理用户账号与角色、用户定义的数据类型与函数、数据转换、服务器启停与配置、备份与恢复、数据库日志、检查数据库一致性、数据统计、安全性管理、分布式事务处理、全文检索等内容。可以利用它进行建库、建表、建立视图、建立存储过程、建立触发器等常规操作。其中有关服务器的操作见图 6.3。在 SQL Server 组下选择服务器并右击，从弹出菜单中选择有关菜单项可以进行服务器注册、属性（内存配置、处理器配置、安全性身份验

证方法、启动账号、连接特性、服务器设置（语言、行为、SQL 邮件、年份范围）、数据库设置（重建索引、备份与还原期限、故障还原间隔、数据库默认位置））、新建（作业、操作员、数据库）、所有任务（管理 SQL Server 消息、数据导入与导出等）操作，也可以管理具体服务器的断开与停止。

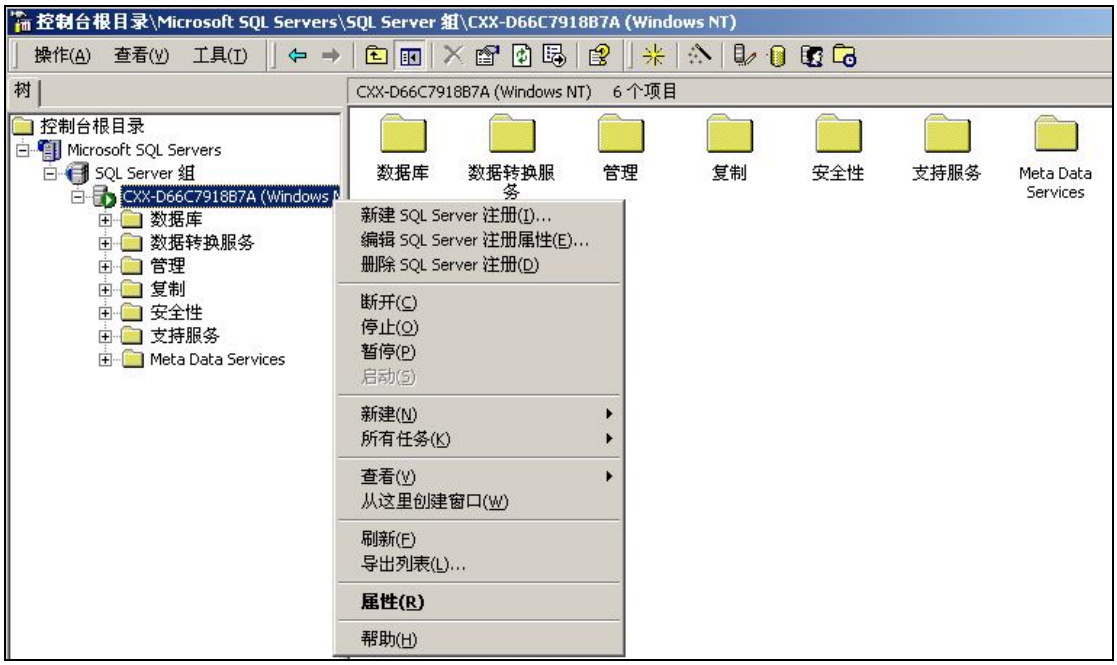

图 6.3　企业管理器中有关服务器的操作

1. 建立数据库

SQL Server 规定只有系统管理员、数据库建立者和他们所授权的用户才能建立数据库。数据库建立者称为数据库的所有者（DBO），他具有对该数据库及其对象管理的权限。

如果要建数据库，在选服务器后，右点击“数据库”，出现数据库管理界面，如图 6.4 所示。

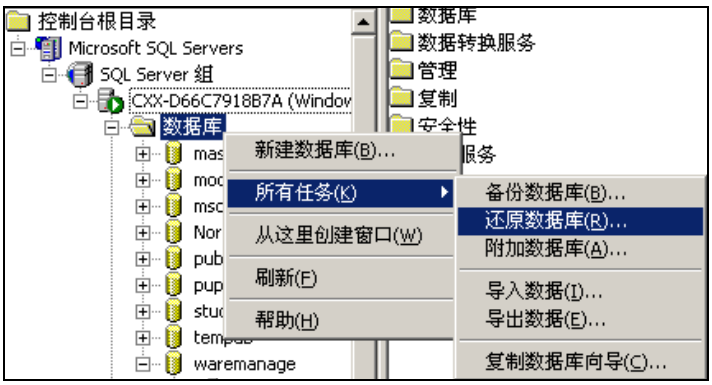

图 6.4　数据库管理界面

在该界面中单击“新建数据库”，进入数据库属性界面，如图 6.5 所示。

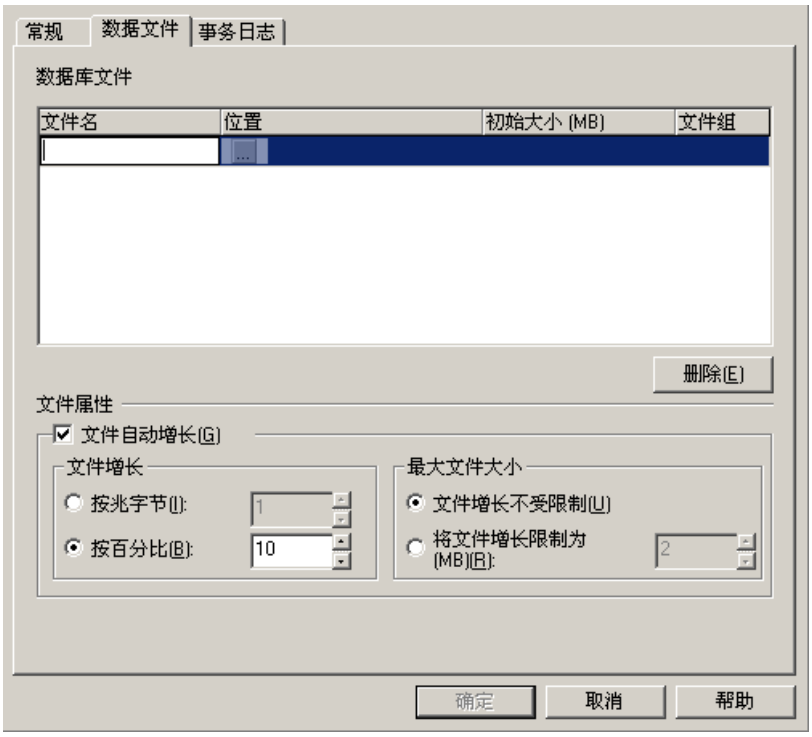

图 6.5 数据库属性界面——新建数据库

在“常规”选项卡中输入数据库名称，名字必须满足 SQL Server 标识符的命名规则：标识符只能由不超过 128 个字符构成，这些字符只能是字母、数字及符号：_、#、@、$。再在“数据文件”选项卡中设置有关数据库文件的属性，包括数据库文件在操作系统中的目录位置、初始大小、所属文件组、文件增长属性等。

如果建立数据库名为 waremanage，会产生：主数据文件 waremanage_Data.MDF 与日志文件 waremanage_Data.LDF 两个文件（见第 1 章相关内容）。除了这两个文件外，如果主数据文件不能存储全部数据信息，还可以有多个次数据文件，扩展名为 NDF。

加“_Data”的文件名为物理文件名，将来在程序中（如 Transact_SQL）这些文件名还有一个逻辑文件名，例如对 waremanage 数据库，主数据文件名为 waremanage.mdf，日志文件名为 waremanagelog.ldf。为了方便管理，SQL Server 将多个数据文件（不包括日志文件）组合在一起，称为数据文件组，它控制每个数据文件的存放位置，由于它的作用，可以将属于一个文件组的文件分布在不同硬盘上，从而减轻单个硬盘驱动器承受的负载，提高工作效率，也便于系统的扩展。

在建立数据库之后，系统自动将该数据库的名称、顺序号、建立日期等信息存储在 SQL Server 的系统数据库 master 的 sysdatabases 表中。

数据库中可以包含表、视图、存储过程，可以规定用户、角色、规则（见图 6.6）。表即关系，在 SQL Server 中表分为永久表与临时表，包含在数据库中的表为永久表，临时表只存在于内存中，表名以#号打头，当用户退出系统时会自动删除。

2. 建立数据表

SQL Server 一个数据库中可以创建约 20 亿个表，每个表可含有 1024 个列（字段），每一

列中数据均为相同数据类型。利用企业管理器建表的操作方法是打开相应数据库（例如waremanage），右击“表”，在弹出菜单中选择“新建表”（见图 6.6），进入新表定义对话框，如图 6.7 所示。

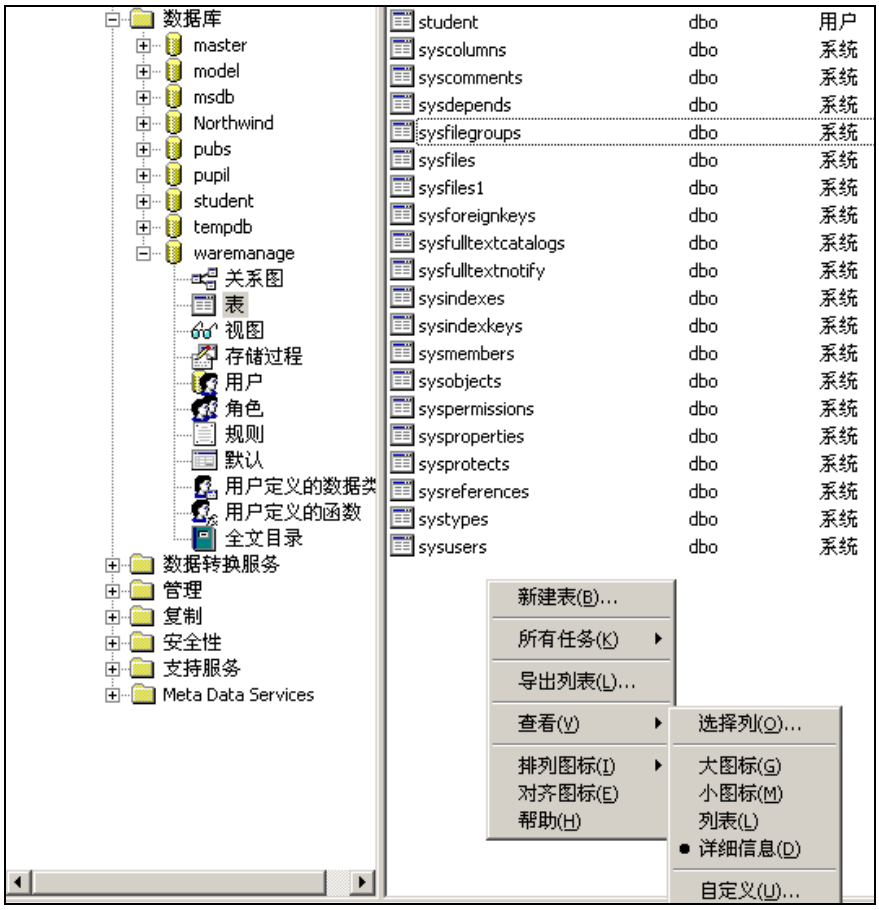

图 6.6　利用企业管理器建立数据表

在新表定义对话框中可以依次输入列名（字段名）、数据类型、长度、是否允许空值。长度指字段宽度，一般除第 1 章表 1.1 中所列宽度前加*的那些类型外的字段宽度均为定值，无须用户设置。空值（NULL）指未定义的不确定的值，是否允许空值中打勾的列在未来输入数据时，可以不填入具体值。一般关键字字段不允许空值。另外注意，如果设置为允许空值，在查询与修改操作时要作特殊处理，因此尽量不要允许空值。

对每一个字段可以进行描述，其内容包括：默认值，指输入时未具体指定数据值，但又不允许空值时自动填入的值；精度与小数位数，对某些允许进行算术运算的二进制数据需要考虑设置。标识，用于设定自动编号的标识列，当用户向表中每增加一新记录时，系统会为该记录生成一个唯一的数值并填入该列，如果选中该复选框，其下“标识种子”、“标识递增量”将自动设置为 1，用户可以修改这两个数据；“是 RowGuid”：如果在该复选框中打勾，将来每录入一条新记录时会在该记录的 uniqueidentifier 类型字段中自动生成一个 16 位的全局唯一标识符（GUID）。

可以为表设定关键字，操作方法是当输入列名后，右击该列名，在弹出菜单中包括设置主键、插入列、删除列、索引/键、CHECK 约束等菜单项。选择“设置主键”，将来表中该字段将不允许出现空值与重复值。

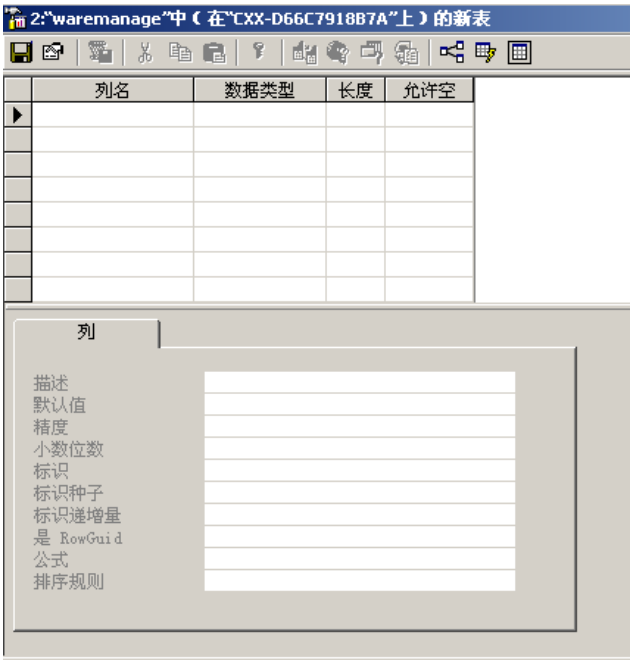

图 6.7　新表定义对话框

3. 修改表结构与数据维护

在建立数据表后，若需要修改表结构，例如增加字段、修改字段名、修改字段属性、修改关于列的描述及建立索引等，可以右击表名，在弹出菜单（见图 6.8）中选择“设计表”，进入表结构修改界面。

如果作数据录入、修改、删除操作，可在如图 6.8 所示弹出菜单中选择“打开表→返回所有行”。

图 6.8　进入表结构修改界面与数据录入、修改、删除界面

4. 建立索引

为了加快查询速度，可以给表建立索引，一个表可以建多个索引。在右击列名后弹出的菜单中选择“索引”，出现如图 6.9 所示对话框。

图 6.9　建立索引

SQL Server 索引分为主索引、唯一索引、普通索引、聚集索引四类。唯一索引要求为之创建索引的列的值在表中具有唯一性，不能有重复值，将来在输入与修改数据时，如果新数据与表中该列已经有的数据相同，操作将不能成功。普通索引则无唯一性要求。

如果设置了关键字，将自动建立主索引（又称为主键索引），主索引是唯一标识记录的特殊的唯一索引。建立主索引也即建立了 PRIMARY 约束，它是数据库进行实体完整性保护的依据。

聚集索引指对数据表中数据按该索引排序后重新存盘，将来表中数据在维护过程中保持该索引所涉及列的数据排序特性不变。对于一个表只可能有一个聚集索引，如果设计聚集索引，按该索引操作时效率较高，但数据维护效率较低。如果在表上尚未创建聚集索引，且在创建 PRIMARY 约束时未指定非聚集索引，PRIMARY 约束会自动创建聚集索引。

在图 6.9 界面中先输入索引名，索引名是今后在程序中调用时使用的名字，当设定后，如果需要修改，下一次进入时，可以先在“选定的索引”下拉列表框中按名字选择欲修改的索引。再选择列名，并选择“升序”或“降序”。每一索引可以选择多列，如果选择多列，称为复合索引，将来数据处理时会先按第一列顺序处理，在第一列的值相同的情况下再按第二列顺序处理，依此类推，最多可指定 16 列。

如果选中“创建 UNIQUE”，则建立唯一索引；如果不选中“创建 UNIQUE”，则建立普通索引。如果选中“创建为 CLUSTERED”，则建立聚集索引。如果建立 PRIMARY 约束，或选中“创建 UNIQUE”的同时选择“约束”，将同时建立同名索引。

创建索引时，可以指定一个**填充因子**，以便在索引项间留出额外的间隙和保留一定百分比的空间，使将来表的数据存储容量进行扩充时对索引文件的维护有较高效率。填充因子的值是从 0～100 的百分比数值，指定为创建索引后的填充比例。值为 100 时表示填满，所留出的存储空间量最小。只有当不会对数据进行更改时才会使用 100%。值越小则对索引文件的维护效率越高，但索引需要更多的存储空间。

在图 6.9 界面中，在“选定的索引”下拉列表框中按名字选择索引，单击“删除”按钮可以删除索引。

6.1.3 建立视图的操作

1. 建立视图

视图是为不同用户提供的不同窗口，可以如同表一样对视图进行操作：查询表中的数据，在一定条件下对表进行数据录入、修改、删除等维护操作。

可以在企业管理器支持下建立视图，右击某数据库下“视图”，在弹出菜单中选择“新建视图”（如图 6.10 所示），将出现如图 6.11 所示“新视图”对话框。

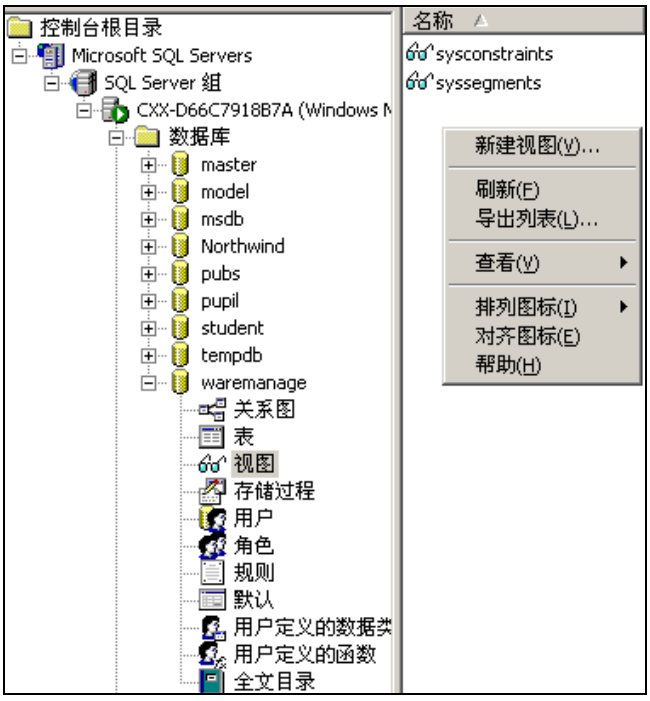

图 6.10 利用企业管理器建立视图

在该对话框中，单击工具条中“添加表”按钮，选择视图所基于的基表，再选择将来该视图中包括的数据列，允许使用别名，使视图中的列名与原基表中的列名不相同，在输出列如果不打勾，则不作为 SELECT 后面输出的内容（实现关系投影操作），可以规定按哪些列排序

及各是升序还是降序。

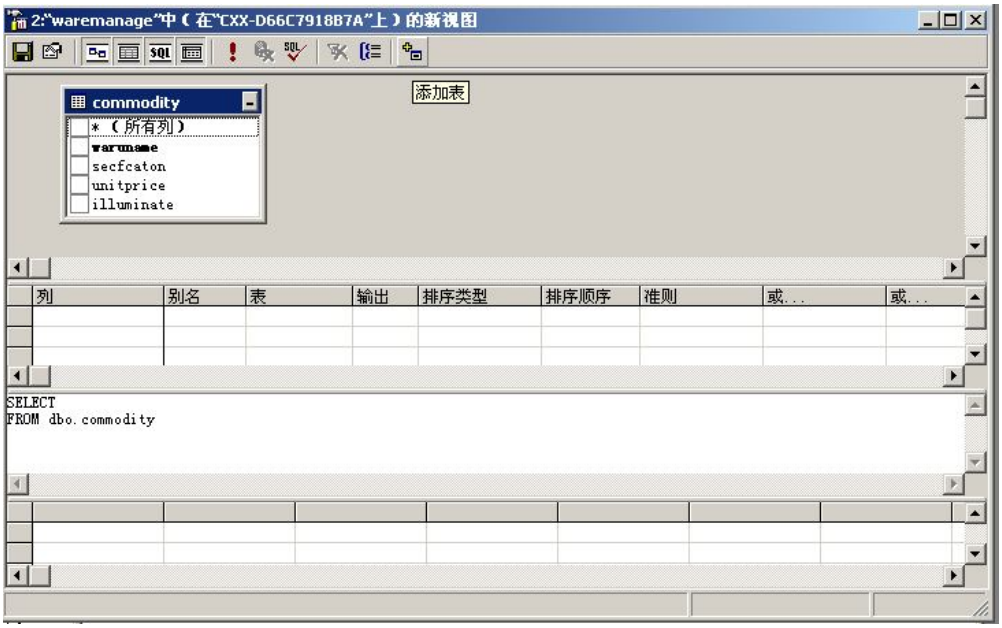

图 6.11　“新视图”对话框

可以规定输出的选择条件（实现关系选择操作）：如果选择条件是多条件的逻辑组合，且条件间是 OR 关系，要求将各条件分别写到“准则”与各个“或”列下；如果是 AND 关系，则要再起一行，将两条件分别写到两行的“准则”下面。

列下允许使用表达式，如采用聚集函数（SUM、MAX、MIN、AVG、COUNT），允许规定按什么分组。

可以选择多个基表，基表间如果有同名字段，会自动建立联系（见图 6.12）。

双击两个基表之间的连线，会弹出属性对话框，如图 6.13 所示，可以改变列间的关系符。

另外，在列间关系下面允许增加两个选择：第 1 个表的所有行与第 2 个表的所有行。如果加选第 1 个表的所有行，表示进行左外连接，输出内容除符合连接条件的那些记录之外，还要加上第 1 个表未列进输出记录的那些记录，这些记录在第 2 个表的各列位置中填 NULL 或按默认值填入；如果加选第 2 个表的所有行，表示进行右外连接，输出内容除符合连接条件的那些记录之外，还要加上第 2 个表未列进输出记录的那些记录，这些记录在第 1 个表的各列位置中填 NULL 或按默认值填入；如果两个复选框都选中，表示进行全连接，输出内容除符合连接条件的那些记录之外，还要加上两个表各自未列进输出记录的那些记录，这些记录在不匹配的另一个表的各列位置中填 NULL 或按默认值填入。

关闭时，给定视图名称，例如保持默认的名字 VIEW1，视图设计完毕。

2. 使用视图

如果视图中涉及仅 1 个基表，且输出全是字段内容，没有表达式，没有聚集函数，则对视图的操作等同于对表操作，向视图录入数据将直接录入到表中，对视图中数据的修改与删除，将直接修改与删除表中的数据。如果是不满足上述条件的视图，只有查询操作与表的操作一致。

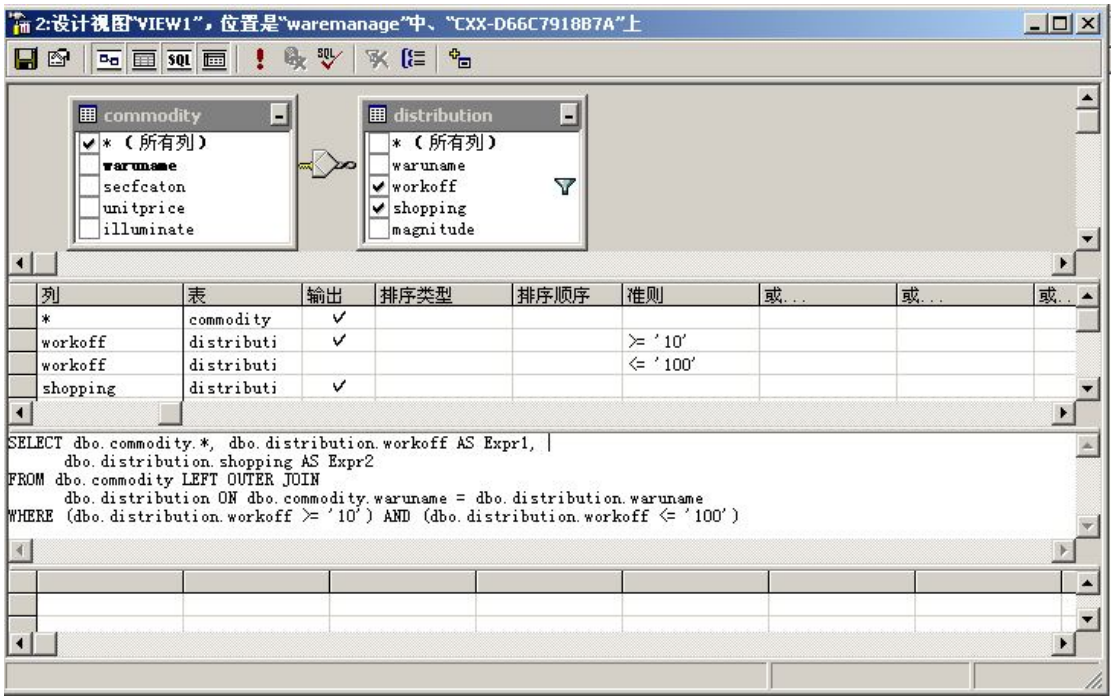

图 6.12　选择一到多个基表建立相应连接

图 6.13　选择联接关系符与实现外联接

例如上述视图 VIEW1 只能用于查询，不能用于数据维护：

```
SELECT * FROM VIEW1
```

将实际完成 commodity 与 distribution 的左连接，在两个表的连接集里按条件选择记录并显示这些记录中原 commodity 表的所有数据及 distribution 表的 workoff、shopping 两个字段的数据。

又例如，如果从 commodity 表中选择 WareName、Specification、unitprice 三个字段，分别更名为 spmc、gg、danw，建立视图 VIEW2。这样的视图中涉及的只有 1 个基表的内容，且输出的全是字段内容，对其中数据的维护操作实际上是对数据表中数据的维护操作：

```
INSERT INTO VIEW2 (spmc,gg) VALUES ('pencil', 'BBB')
```

意义是向 commodity 中录入一条新记录，其商品名称为 pencil，规格为'BBB'。

由此可见，视图可以简化程序编写的操作，使程序结构简单、清晰。尤其是给不同操作人员以不同视图，这是实现安全性控制的重要方法之一。

6.1.4　数据完整性保护

1. 实体完整性保护的实现

如果在表中定义了关键字（主键），就实际定义了 PRIMARY 约束（又称主键约束），自动建立主索引。当录入数据或修改数据值时，将自动检查主键的值，如果为空值，或与已经录入的主键值重复，将拒绝存盘。

2. 参照完整性保护的实现

在图 6.9 界面中选择“关系”选项卡，可以设置表与表之间的关系，如图 6.14 所示。

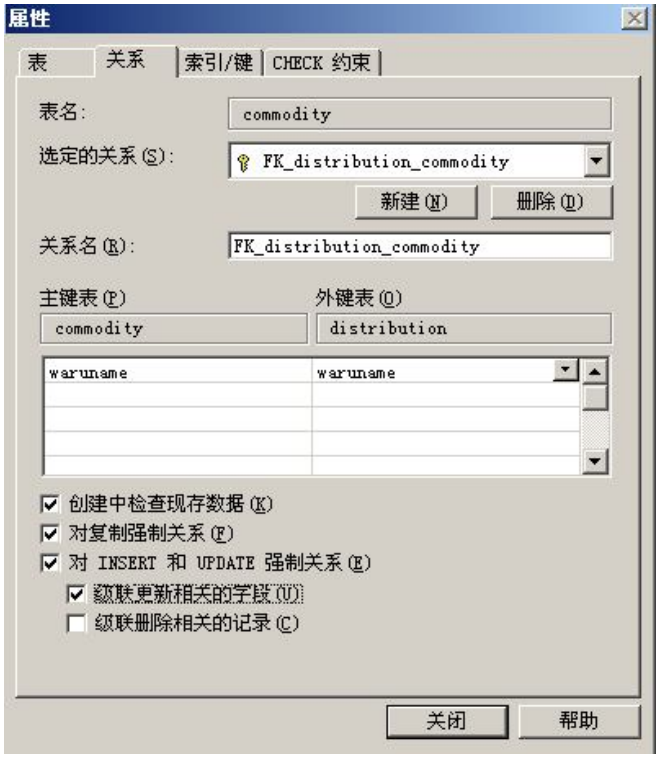

图 6.14　建立表之间关系或 FOREIGN 约束

单击“新建”按钮，再选择主表（主键表）的名字与子表（外键表）的名字，分别选择两表中相关联的字段（主键与外键），就建立两表的 FOREIGN 约束（又称外键约束）。要求子表中外键的值在主表中必须存在，称为参照完整性。

操作时可以选择“级联更新相关的字段”，在数据录入或更新操作时，如果数据不满足上述参照完整性约束要求，将不能录入或自动修改子表外键的值，使之符合参考完整性要求。还可以选择“级联删除相关的记录”，在删除主表数据时，将同时删除子表中外键值与其主键中相关联字段值相同的所有记录。

对于上述操作有三个复选框可选：①创建中检查现存数据，要求在建立该关系同时检查已经存放在表中的数据，如果有违反该关系的，必须先修改表中的数据之后才能再重新建立该约束；②对复制强制关系，指在复制数据时检查该约束关系，只有满足该约束的数据才允许复制；③对 INSERT 和 UPDATE 强制关系，指在录入新数据或修改表中数据时，只承认满足该约束关系的数据。

3. 用户定义完整性保护的实现

在图 6.9 界面中选择“CHECK 约束”选项卡，可以定义关于数据范围的约束（又称检查约束），如图 6.15 所示。

图 6.15　对列设置数据正确性约束

为尽量保证数据在录入、修改、导入等操作中的正确性，可以设置保证数据正确性的约束条件，使数据只能在一定范围内才能存进数据库。

该约束起作用的范围同样可作三方面考虑：①创建中检查现存数据；②对复制强制约束；③对 INSERT 和 UPDATE 强制约束。

6.1.5　备份与恢复

1. 备份与恢复的概念

当操作不当或出现意外事故（停电、设备故障）时，数据可能丢失或被破坏。为了保证数据的正确性，有必要经常将数据库中的数据保存到其它其他地方（其它其他文件夹、其它其他盘、其它其他计算机中），称为备份。一旦出现意外，可以利用备份文件恢复原来数据库中的备份前的内容，称为恢复。SQL Server 支持 4 种基本备份：数据库备份、事务日志备份、差

异备份、文件和文件组备份。

数据库备份是指对数据库的完全备份，包括用户表、系统表、索引、视图、存储过程、事务日志等所有数据与数据库对象。事务日志备份则是从上次备份事务日志之后对数据库执行的所有事务的记录，事务记录只记录数据库改变情况，所需要的时间与存储空间都比较少。差异备份只备份从上次差异备份之后更改了的数据，需要的存储空间更少。文件和文件组备份将数据库所涉及文件与文件组全部备份，需要时间很短，但是可能需要较多的存储空间。要根据需要选择备份内容，一般日常多做差异备份，定期进行全备份与文件备份。

2. 创建备份设备

数据库备份前，先要选择存放备份数据的备份设备，操作为：进入企业管理器、展开服务器组、展开服务器，如图 6.3 所示。展开“管理”文件夹，选择备份，在弹出菜单中选择“新建备份设备”，弹出新设备对话框，如图 6.16 所示。在“名称”文本框中输入操作者定义的设备名称；在“文件名”文本框中输入备份设备的文件名称，可以单击“…”按钮，从“备份设备位置”对话框中选择备份设备的物理文件名称。最后单击“确定”按钮完成备份设备的创建。

图 6.16　新设备对话框

3. 备份

右击要备份的数据库的名字，在“所有任务”下面选择“备份数据库”（也可以展开“管理”文件夹，选择备份，在弹出菜单中选择“备份数据库”），将弹出 SQL Server 备份对话框（如图 6.17 所示），可以重选数据库，定义备份文件的名称（例如 master 备份），如果需要，可以在“描述”对话框中进一步对备份文件进行说明。

在“目的”选项组中通过“添加”、“删除”等按钮选择备份文件或备份设备。

如果选择的是用户自己的数据库，在“备份”选项组中可以选择备份内容：数据库－完全备份、数据库－差异备份、事务日志备份、文件和文件组备份。

在“重写”选项组中可以选择备份方式，如果选择“追加到媒体”，表示将备份内容写到当前备份设备原有内容的后面。如果选择“重写现有媒体”，表示将备份内容写到当前备份设备内，复盖当前备份设备中的内容。

如果希望按照一定周期对数据库进行备份，可以选中“调度”复选框，再单击其后“…”按钮，打开编辑调度对话框，安排每次备份数据库的执行时间。如果不选中“调度”，单击“确

定”按钮，将执行备份操作。

图 6.17　SQL Server 备份数据库的操作界面

4. 恢复

恢复是加载备份并应用事务日志重建数据库的过程，当执行恢复操作时，会将备份文件中的数据复制到数据库中，并回滚所有未完成的事务，以保证数据库中数据的一致性。

操作方法：右击“数据库”或右击具体数据库的名字，在弹出菜单中选择“所有任务”，再选择“还原数据库”，如图 6.18 所示。

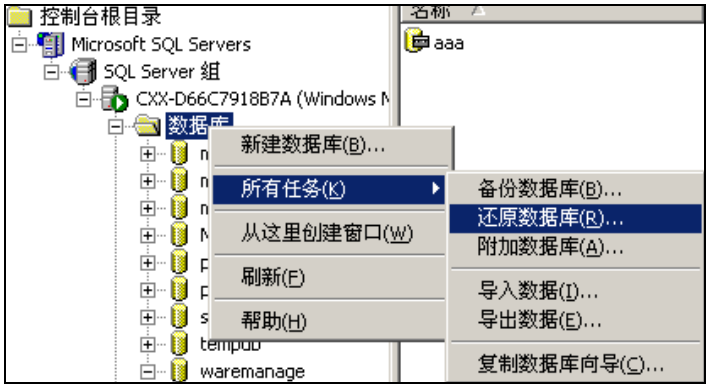

图 6.18　恢复数据库的操作过程

在“还原数据库”对话框中选定欲还原的数据库名，选择是从数据库备份、文件组或文件备份还是从备份设备中还原，在“参数”选项组中，选择备份文件的名字，如果在一个备份文件中保存了多个备份，再选择从其中的哪一个备份恢复（默认的是最新的备份），如图 6.19 所示。

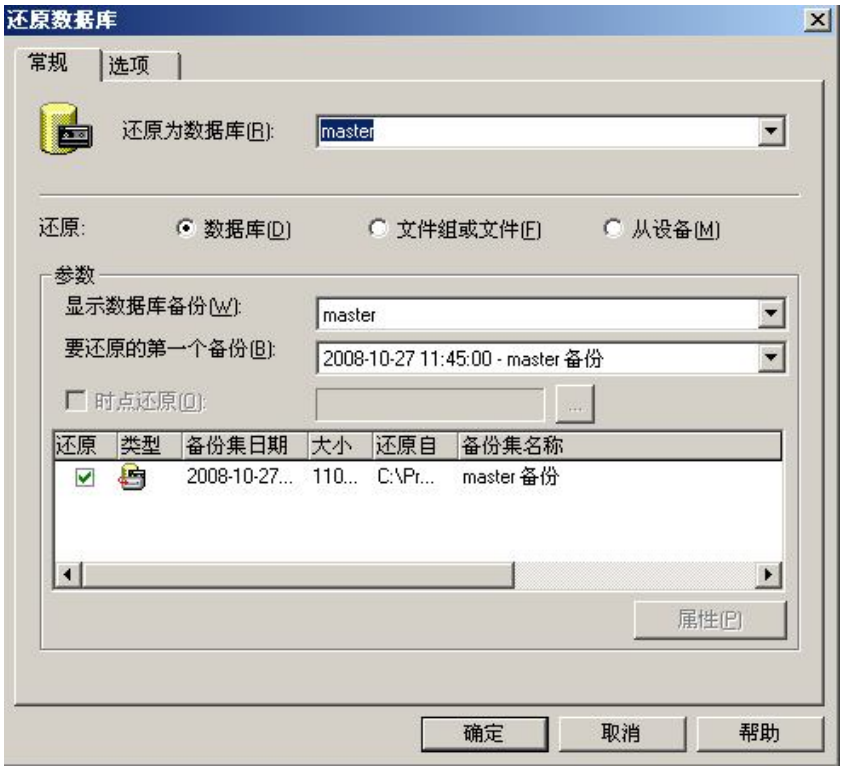

图 6.19　“还原数据库”对话框

可以转到“选项”选项卡（如图 6.20 所示）。在“将数据库文件还原为”列表框中给出了要恢复的数据库原文件名和将要恢复的文件名。在“恢复完成状态”选项组中可以选择数据库恢复完毕后数据库的状态。

在设置完毕后单击“确定”按钮，将自动完成数据库恢复工作。

6.1.6　数据库安全性管理

数据库的目标是让数据最大限度地为各用户共享，尽可能为更多的应用服务。这就要求数据按权限使用，即不同用户可能对数据结构操作的权限不同，如数据的录入权限不同、修改权限不同、删除权限不同、查询权限不同等，数据库要确保只有具有相应权限的用户才能进行相关的操作，防止越权使用。要具有这样的性能，数据库一是要能记录不同用户对有关目标所具有的权限，二是要能在不同情况下识别操作者、检查他所具有的权限、控制他只能进行他有权进行的操作。为此，数据库都应具有安全性管理功能。

数据库的安全性管理功能一般包括两方面内容：①用户能否登录及如何登录的管理；②用户能够操作哪些对象与执行哪些操作的管理。

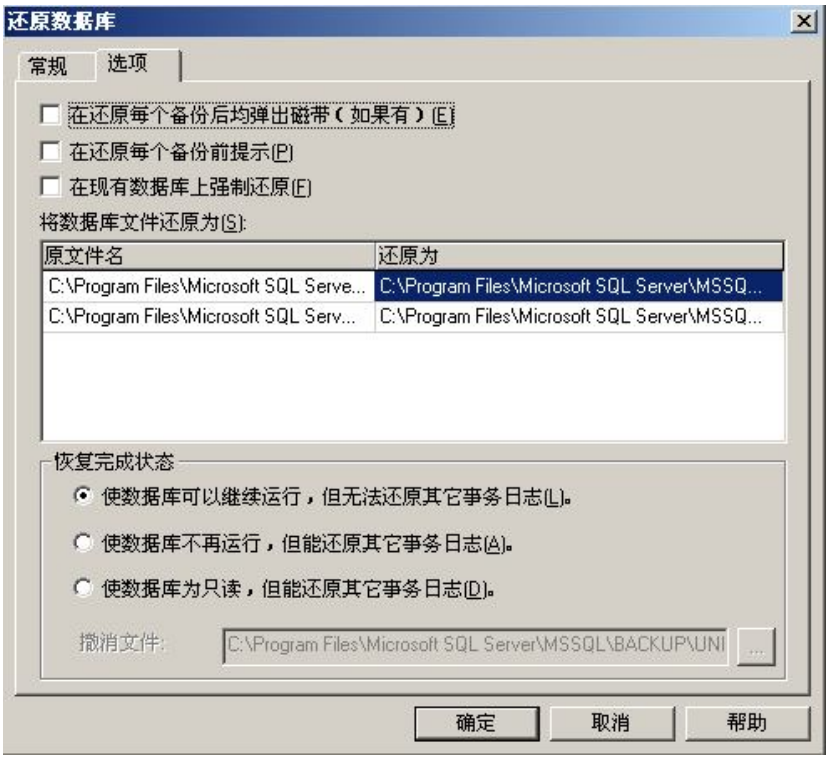

图 6.20 “选项”选项卡

SQL Server 的安全性管理是建立在身份验证和访问许可的机制上的。身份验证要求首先建立用户表，用户通过注册将自己的信息（包括自己选择的密码）存入表内，系统管理员要将其所具有的权限信息（访问许可）也存入表内，在进入系统前先要执行登录操作，报告自己的账号与密码，系统检查其输入内容是否与用户表中保存的数据一致，确定用户的合法性、确定他所具有的权限，控制其可以进行的操作。

1. 身份验证

SQL Server 身份验证有两种模式：Windows 身份验证模式与混合身份验证模式。前者只要用户能登录 Windows NT/2000/2003 操作系统，即具有 Windows 用户账号，就视同 SQL Server 身份验证通过。这种方法集成了 Windows NT/2000/2003 的安全系统的功能，例如密码加密、审核、密码过期、最短密码长度、身份验证（包括多次登录申请无效后锁定账户等）功能。而混合身份验证使用户得以使用 Windows 身份验证或使用 SQL Server 身份验证实现与 SQL Server 连接。在 SQL Server 身份验证下，用户在连接 SQL Server 时必须提供登录名和密码，SQL Server 在系统表 syslogin 中检测输入的账户名和密码，只有找到相匹配的，才能进入系统。

SQL Server 身份验证模式通过 SQL Server 属性对话框设置。方法是：展开服务器组→右击需要修改验证模式的“服务器”名→选择属性（R）→系统弹出 SQL Server 属性对话框→进入“安全性”选项卡，如图 6.21 所示。

2. 登录账号的管理

在 SQL Server 中登录账号与用户账号是两个不同的概念，一个登录账号总是与一个或多个数据库用户账号对应。用户登录后将具有进入 SQL Server 的权限，在进一步有了用户账号

后，才具有访问数据库的权限。

图 6.21　SQL Server 属性对话框

创建登录账号的操作如图 6.22 所示：展开服务器组→展开服务器→选择“安全性” →登录→新建登录→弹出 SQL Server 登录属性－新建登录对话框，如图 6.23 所示。

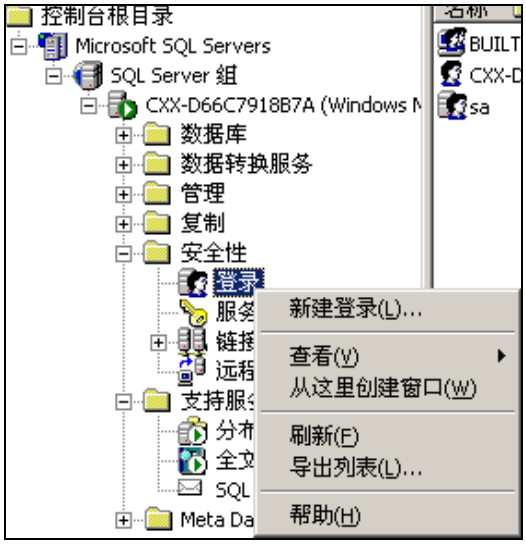

图 6.22　创建登录账号的操作

图 6.23 “新建登录”对话框

在“常规”选项卡中先输入用户登录账号，再确定身份验证模式，如果选择 SQL Server 身份验证，将要求输入密码。之后选择该登录账号有权操作的数据库与语言。

3. 创建用户账号

创建用户账号的操作方法是：展开服务器组→展开服务器→展开数据库→右击某具体数据库→新建→数据库用户（如图 6.24 所示）→进入“新建用户”对话框。

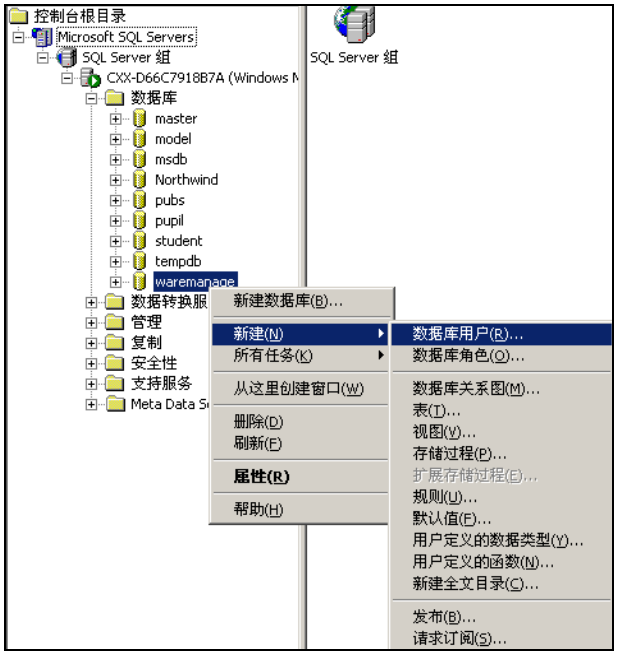

图 6.24 创建用户账号的操作

在“登录名”列表框中选择于“新建登录”对话框中输入的某登录账户，再输入用户名，选择数据库角色中所允许的操作权限，单击“确定”按钮，如图 6.25 所示。

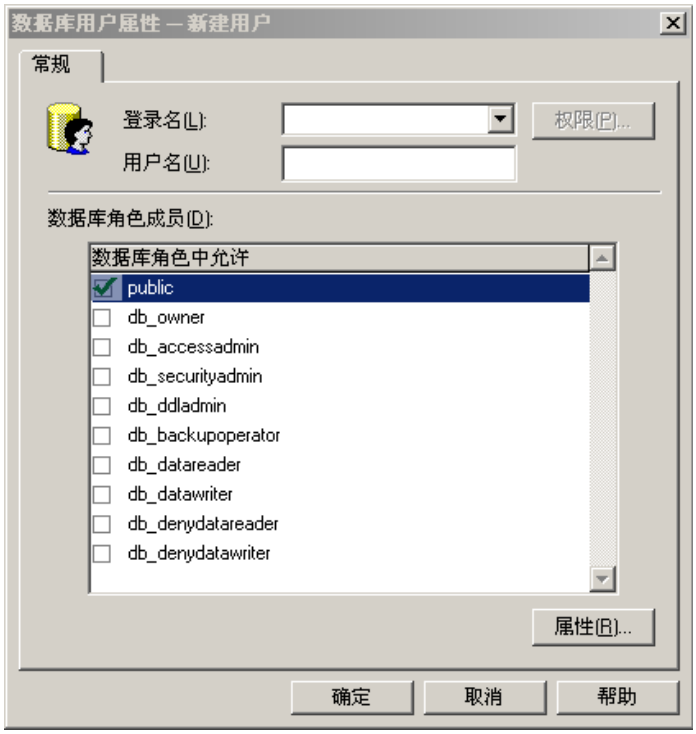

图 6.25　“新建用户”对话框

4. 服务器角色

当几个用户工作相类似时，为简化管理与方便操作，可以将他们集中到一个称为角色的单元中，按角色分配权限，对一个角色的权限设置可以用到多个用户的管理中。

SQL Server 规定了两种角色类型：服务器角色与数据库角色。系统创建了 8 个服务器角色，如表 6.1 所示。

表 6.1　服务器角色及其权限

服务器角色	操作权限
sysadmin	在 SQL Server 中各种活动
securityadmin	管理服务器登录
serveradmin	配置服务器范围
setupadmin	添加和删除链接数据库并执行某些系统存储过程
processadmin	管理在 SQL Server 中运行的进程
diskadmin	管理磁盘文件
dbcreator	创建和改变数据库
bulkadmin	执行 BULK INSERT 语句

指定用户角色的方法：在“新建登录”对话框中选择“服务器角色”选项卡→双击某角色，弹出“服务器角色属性”对话框，如图 6.26 所示。单击“属性”按钮，在弹出的“添加成员”对话框中选择用户名，并单击“确定”按钮。如果选择“权限”选项卡，可以查看该服务器角色所具有的权限情况，如图 6.27 所示。

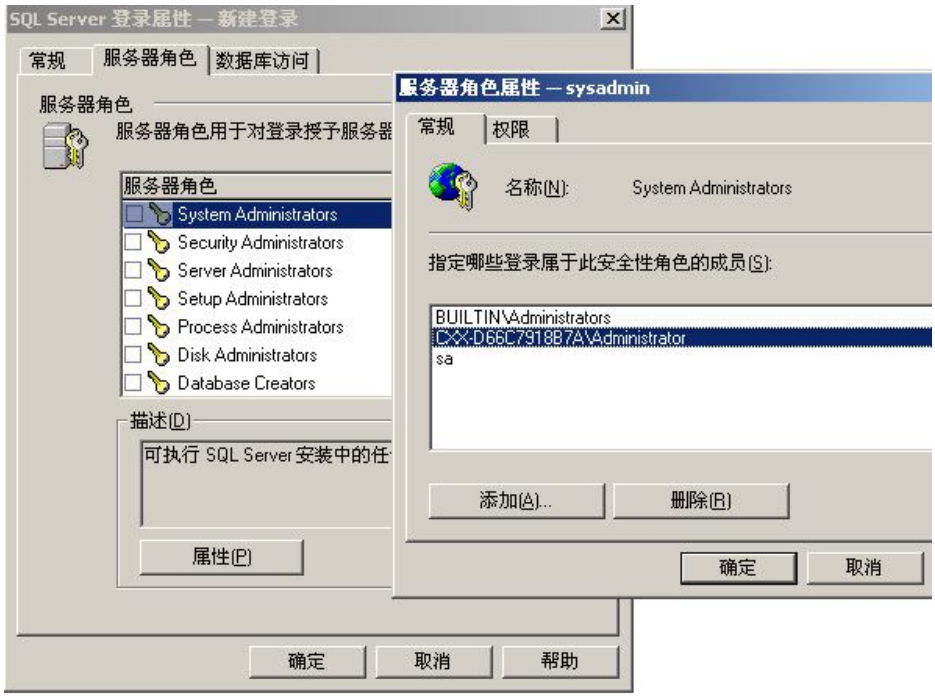

图 6.26　为用户授予服务器角色

图 6.27　查看该服务器角色所具有的权限

5. 数据库角色

系统管理员可以将用户加入到用户内部数据库角色中，从而能在数据库级别上进行操作。

SQL Server 提供了两种数据库角色类型：预定义的数据库角色和用户自定义的数据库角色。预定义的数据库角色有规定的权限，只要给某角色赋予某种预定义的数据库角色，该角色就具有预先规定的那些权限。

如果要将登录用户添加到固定数据库角色成员中，可以展开数据库文件夹，展开用户准备授权的数据库，右击“角色”，选择“新建数据库角色”，在弹出的数据库角色属性对话框中进行操作。

6. 权限管理

用户或角色还需要进一步被授予某些创建或操作权限才能对数据表、视图、存储过程进行具体的操作。

创建权限包括：①创建数据库权限：CREATE DATABASE；②创建表：CREATE TABLE；③创建视图：CREATE VIEW；④创建规则：CREATE RULE；⑤创建缺省：CREATE DEFAULT；⑥创建存储过程：CREATE PROCEDURE；⑦备份数据库：BACKUP DATABASE；⑧备份事务日志：BACKUP LOG。

操作权限包括：①对表或视图录入操作权限：INSERT；②对表或视图或列修改操作权限：UPDATE；③从表或视图删除操作权限：DELETE；④对表或视图或列查询操作权限：SELECT；⑤对表转授权操作权限：REFERENCE；⑥对存储过程的执行权限：EXECUTE。

利用企业管理器授权的方法：展开服务器组→展开服务器→展开数据库→展开某具体数据库→用户→双击某具体用户→进入“数据库用户属性”对话框，如图 6.28 所示。

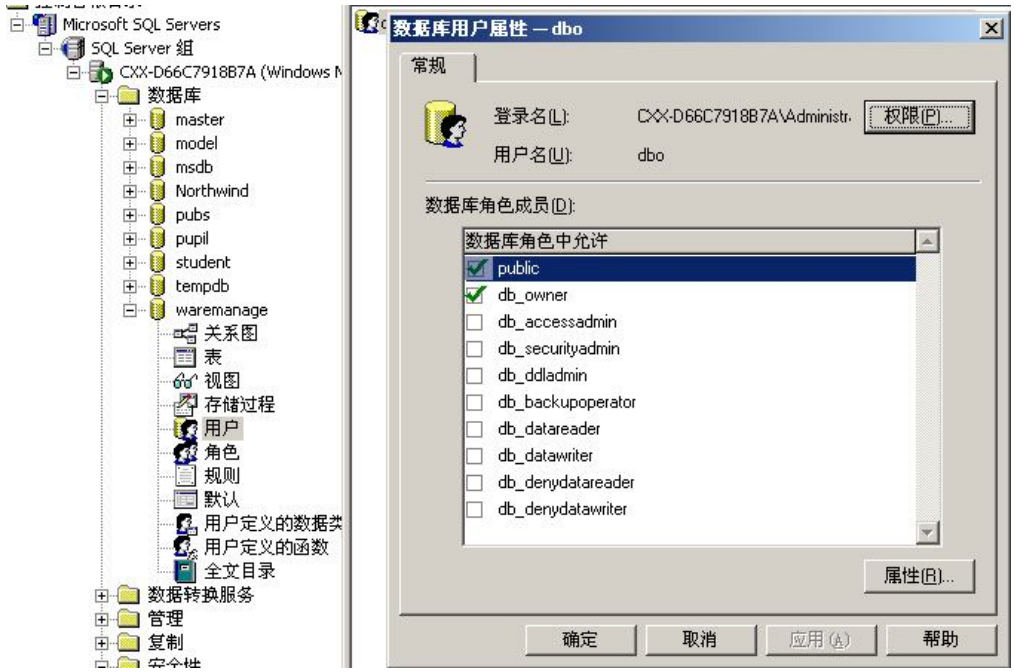

图 6.28　“数据库用户属性”对话框

在“数据库用户属性”对话框中单击“权限”按钮，弹出“数据库用户属性”对话框的“权限”选项卡，如图 6.29 所示。先选择视图、表或列等对象，再授予有关权限。

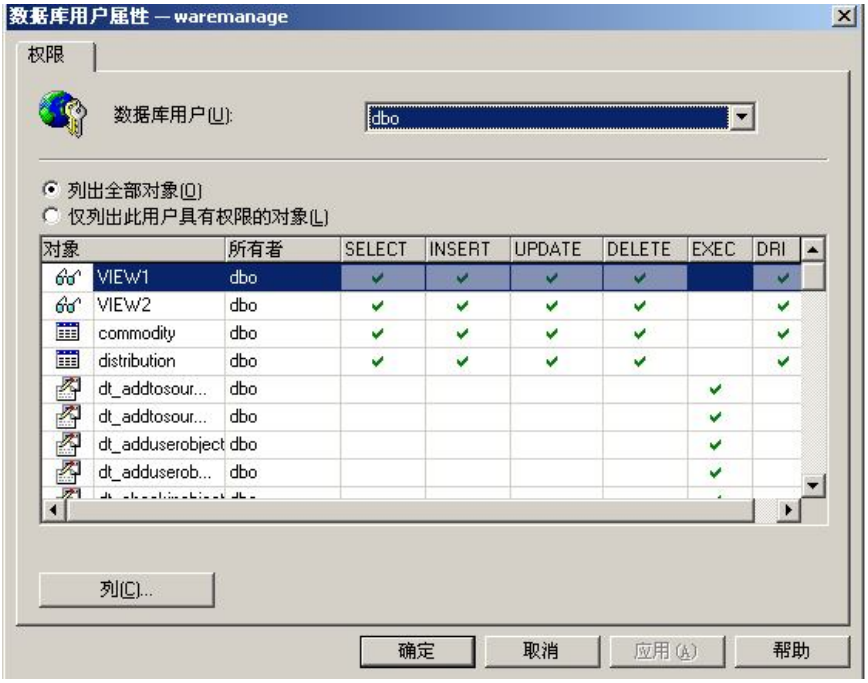

图 6.29　权限选项卡

6.1.7　查询分析器

除了利用企业管理器对数据库进行操作外，还可以利用 SQL 语言的命令完成以上各项操作。SQL Server 中的 SQL 语言是 Transact-SQL 语言，利用 Transact-SQL 语言操作的界面之一是查询分析器。

当选用查询分析器后首先进入“连接到 SQL Server”对话框，如图 6.30 所示。

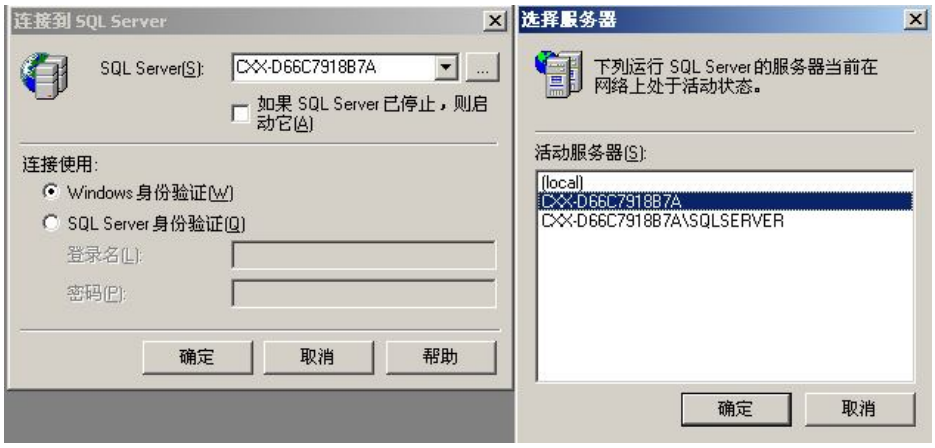

图 6.30　“连接到 SQL Server”对话框

选择“…”按钮可以选择服务器，再选择身份验证方式之后进入查询分析器操作界面。

如果涉及建表、数据维护、查询等操作时首先要选择数据库，在主菜单中选择“查询”，再选择“更改数据库”（见图 6.31），进入更改数据库的界面，如图 6.32 所示。

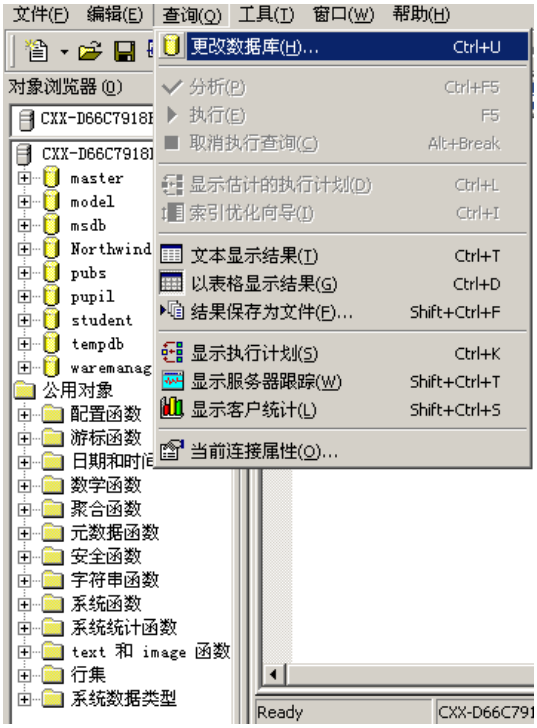

图 6.31　更改数据库的操作

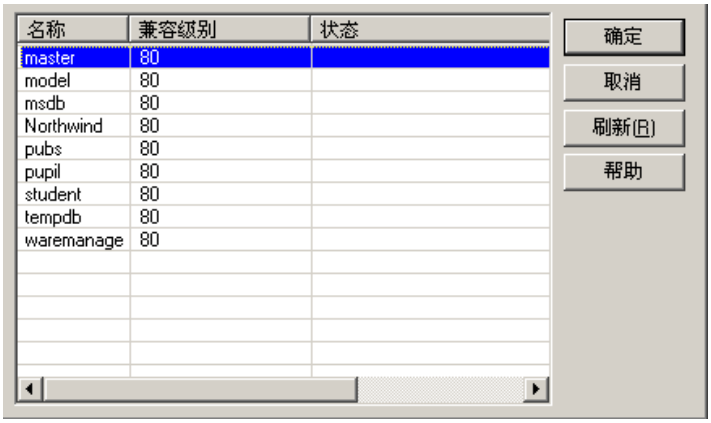

图 6.32　更改数据库的界面

选定数据库后，就可以在查询分析器的编辑框中输入 Transact-SQL 语句，如图 6.33 所示。

在查询分析器的工具栏中用▶标志“执行”按钮，单击该按钮即可执行所输入的 Transact-SQL 语句，如图 6.34 所示。

图 6.33 查询分析器的编辑框

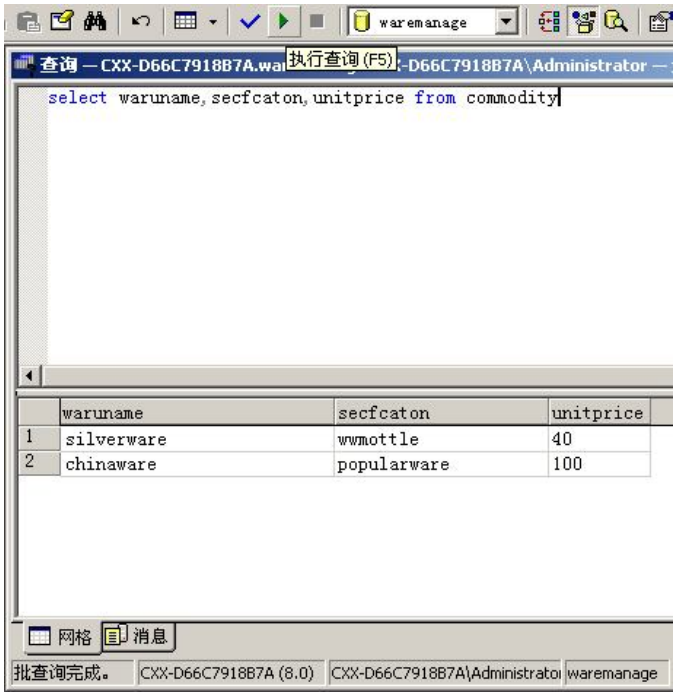

图 6.34 执行 Transact-SQL 语句

6.2 Transact-SQL 语言简介

SQL Server 中的 SQL 语言是 Transact-SQL 语言。它包括以下主要组成部分：

- 数据定义语言（Data Definition Language，DDL）
- 数据操纵语言（Data Manipulation Language，DML）
- 数据控制语言（Data Control Language，DCL）
- 系统存储过程（System Stored Procedure）
- 一些附加的语言元素

本节主要介绍 Transact-SQL 语言比标准 SQL 语言扩展或加强的部分内容。

6.2.1 数据定义语言（DDL）

数据定义语言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，包括 CREATE、ALTER 和 DROP 等语句。在 SQL Server 中，数据库对象包括表、视图、触发器、存储过程、规则、缺省、用户自定义的数据类型等。这些对象的创建、修改和删除等都可以通

过使用 CREATE、ALTER、DROP 等语句来完成。

1. 创建数据库的命令

命令格式：CREATE DATABASE <数据库名> [ON [PRIMARY] [<文件 1>[，<文件 2>……]][，<文件组和文件>]] [LOG ON <文件>] [COLLATE <排序规则名称>] [FOR LOAD | FOR ATTACH]

说明：

（1）PRIMARY 表示定义数据主文件，其后是关于数据文件的描述。如果没有 PRIMARY 字样，其中第一个文件为主文件。

（2）关于“文件”的格式：NAME= <文件逻辑名称>，FILENAME=<操作系统中的文件名称>，SIZE=<文件大小>，MAXSIZE=<文件最大尺寸>，FILEGROWTH=<每次文件添加空间的大小>。

其中“文件逻辑名称”指将来在指令中使用的名称，“操作系统中的文件名称”指存盘路径与文件名称，路径应当是 SQL Server 实例中的目录，不应是压缩文件系统中的目录。

大小与尺寸的默认单位为 MB。每次需要增加空间时添加空间的大小可以用默认单位 MB，也可以用百分比：%，默认为 10%。

文件可以是多个文件，对每一个文件的描述用括号括起，彼此间用逗号分隔。

（3）关于“文件组和文件”的意义：定义中用 PRIMARY 定义主文件组，可以再增加用户定义文件组及文件组中的文件。格式为：FILEGROUP <文件组名> <文件 1>[,<文件 2>……]。

（4）LOG 子句中定义的文件指日志文件名。

（5）COLLATE 指定默认的排序规则，排序规则可以是 Windows 排序规则，也可以是 SQL Server 排序规则。如果没有指定排序规则，则以 SQL Server 实例的排序规则为排序规则。

（6）FOR LOAD 指可以从备份数据库中加载。FOR ATTACH 是将已脱机的数据库重新联机。

【例 6.1】创建数据库：waremanage，数据文件初始大小为 1M，最大为 10M，如果需要增加空间，每次增加 1M。逻辑文件同样设置。

```
CREATE DATABASE waremanage ON(NAME= 'waremanage_data',FILENAME='c:\program files\Microsoft sql server\data\ waremanage_data.mdf',SIZE=1,MAXSIZE=10,FILEGROWTH=1) LOG ON (NAME='waremanage_log',FILENAME='waremanage_data.ldf',SIZE=1,MAXSIZE=10,FILEGROWTH=1)
```

2. 创建数据库表

创建表的语句格式类似于标准 SQL 语言中创建表的语句格式，但涉及数据库、约束条件、索引等描述后，表达更深入。

命令格式：CREATE TABLE <表名说明>({列定义或列计算式} [CHECK 子句] [ON { <文件组名> | DEFAULT }] [TEXTIMAGE_ON { <文件组名> | DEFAULT }])

说明：

（1）关于“表名说明”有三种格式：①直接用表名，表示在当前数据库下建表。②<数据库名>.<表名>，只有数据库属主有权操作；③<数据库名>.<属主>.<表名>。

（2）列定义的一般格式是：〈列名〉〈数据类型〉[〈宽度〉] [NOT [NULL]] [CONSTRAINT 子句] [UNIQUE 子句] [PRIMARY 子句] [FOREIGN 子句] [DEFAULT 子句]

1）CONSTRAINT 子句是可选关键字，表示 PRIMARY KEY、NOT NULL、UNIQUE、

FOREIGN KEY 或 CHECK 约束定义的开始，同时定义索引名称。格式为：CONSTRAINT <索引名>。

2）UNIQUE 表示唯一性，表示该列不允许有重复值。它和主键不同的是，主键除不允许有重复值外，还不允许有空值。

3）PRIMARY 子句说明该列为主键。格式：PRIMARY KEY CLUSTERED。

4）DEFAULT 子句定义默认值，如果该列不允许空值，而在录入时又未说明该列的值，则自动用默认值填充；否则填入 NULL。

5）FOREIGN 子句定义外键，其格式为：FOREIGN KEY <外键名> REFERENCES <主表名称>（主键名称）。其中外键名可以是多个列的列名，用逗号分隔，但其数量与类型必须与其后主表中说明的相关列的列名与类型一一对应。

（3）{}表示其中内容为多个类似结构的集合，例如{列定义}表示多个列定义的集合，列定义之间用逗号分隔。

（4）列计算式指有的列的数据可以由另外一些列的数据根据具体的公式计算得到，称为派生数据。在列定义中，这样的列可以用如下格式定义：

<列名> AS <计算列值的表达式>

计算列不能作为 INSERT 或 UPDATE 语句的目标，也不能作为 DEFAULT 和 FOREIGN KEY 的约束定义。

（5）CHECK 子句为表级约束，说明列自定义约束条件。其格式为：CHECK (<约束条件表达式>)。

其中约束条件表达式如下所示：

1）<字段名> IN (<值表>)。

2）<字段名> <关系符><数据值>。

3）<字段名> LIKE <匹配表达式>。

约束条件表达式可以由多个表达式组成，彼此间用 AND、OR 连接。

（6）ON { <文件组名> | DEFAULT }定义存储表的文件组，该文件组必须在数据库中存在。如果使用“DEFAULT”或没有该子句，表示存储在默认文件组中。

（7）TEXTIMAGE_ON { <文件组名> | DEFAULT } 定义 text、ntext、image 等类数据所存储的文件组名称。

【例 6.2】在数据库 waremanage 中创建关于出版物的表 publishers(pub_id, pub_name, author ,unitprice, unit)。

在选择数据库 waremanage 之后在查询分析器中可以输入如下语句建表：

CREATE TABLE publishers (pub_id char(4) NOT NULL CONSTRAINT UPKCL_pubind PRIMARY KEY CLUSTERED CHECK (pub_id IN ('0389', '0736', '0877', '1622', '1756') OR pub_id LIKE '20[0-9][0-9]'),pub_name varchar(40) NULL, author varchar(20) NULL, unitprice int NULL, unit varchar(30) NULL DEFAULT('USA'))

说明：

（1）NOT NULL 表示在输入数据时不允许空值，NULL 表示在输入数据时允许空值。

（2）CONSTRAINT UPKCL_pubind PRIMARY KEY CLUSTERED 建立主键约束，同时建立主键索引，索引名为 UPKCL_pubind。CLUSTERED 表示索引类型为聚集索引。

（3）CHECK (pub_id IN ('0389', '0736', '0877', '1622', '1756') OR pub_id LIKE '20[0-9][0-9]') 子句建立用户定义约束，将建立如下表达式：([pub_id] = '1756' or ([pub_id] = '1622' or ([pub_id] = '0877' or ([pub_id] = '0736' or [pub_id] = '0389'))) or [pub_id] like '20[0-9][0-9]')

like '20[0-9][0-9]')表示出版物编号头两位为“20”，后两位的每一位由 0 到 9 之间的数字构成。

3. 修改数据表与删除数据表

修改数据表与删除数据表的语句和标准 SQL 语言中修改数据表与删除数据表的语句基本相同，但修改数据表的添加新列、修改列属性等语句中可包括添加约束、修改约束的内容。

【例 6.3】为 publishers 的 unitprice 创建一个名为 CK_publishers 的约束，要求控制 unitprice 的值在 10～1000 之间。

```
ALTER TABLE publishers ADD CONSTRAINT CK_publishers CHECK (unitprice>=10 AND unitprice<=1000)
```

4. 建立视图

语句格式：CREATE VIEW [<数据库名>.][<属主名>.] <视图名>[({<列名>})] [WITH <视图属性>] AS <子查询> [WITH CHECK OPTION]

说明：

（1）{<列名>}定义视图中的列名，可以与表中列名不相同，列名间用逗号分隔。

（2）视图属性包括①ENCRYPTION：表示加密包含 CREATE VIEW 语句文本的系统表列；②SCHEMABINDING：将视图绑定在架构上，要求<SELECT 子句>包含表、视图等两部分的名称；③VIEW_METADATA：表示在某些查询中可以返回元数据信息。

（3）子查询与 WITH CHECK OPTION 见 4.7 节相关内容。

5. 修改视图

语句格式：ALTER VIEW [<数据库名>.][<属主名>.] <视图名>[({<列名>})] [WITH <视图属性>] AS <子查询> [WITH CHECK OPTION]

不难发现，修改视图语句的参数和建立视图的语句参数是相同的。

【例 6.4】修改原有表 publication 的视图 VIEW_pub 的内容，要求输出 pub_id、pub_name 与 unitprice，显示条件是 unitprice 大于 100。

ALTER VIEW waremanage. VIEW_pub AS SELECT pub_id, pub_name, unitprice FROM publishers WHERE unitprice>100

6. 建立索引

语句格式：CREATE [UNIQUE] [CLUSTERED | NONCLUSTERED] INDEX <索引名称> ON {<表名> | <视图名>} ({<列名>[ASC | DESC]}) [{WITH <索引选项>}] [ON <文件组>]

说明：

（1）UNIQUE 选项表示创建唯一索引。

（2）CLUSTERED | NONCLUSTERED 选项表示创建聚集或非聚集索引。如果是聚集索引，则数据表中数据物理存储时按该索引规定的顺序排列。

（3）索引选项如 IGNORE_DUP_KEY 表示向属于聚集索引的列插入重复值时将发出警告并拒绝执行。又如 DROP_EXISTING 表示除去并重建先前存在的聚集或非聚集索引。

【例 6.5】为表 publishers 建立关于 pub_id 的聚集索引。

CREATE UNIQUE CLUSTERED INDEX pub_pub_id (pub_id ASC) WITH IGNORE_DUP_KEY

6.2.2 数据操纵语言（DML）

数据操纵语言是指用来查询、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。

1. 插入数据语句

语句基本结构：INSERT INTO ({<表名> | <视图名>} {[<列名>]}) VALUES ({DEFAULT | NULL | <表达式>})

说明：

（1）{[<列名>]}指多个列的名字，彼此用逗号分隔。如果该项省略，表示对表的所有字段。

（2）{DEFAULT | NULL | <表达式>}表示多个“DEFAULT | NULL | <表达式>”的集合，彼此用逗号分隔，其数量与顺序要和{[<列名>]}一致，如果语句中省略{[<列名>]}，那么要和表定义的结构中列的数量与顺序一致。

“DEFAULT | NULL | <表达式>”表示可以是“DEFAULT”，或者是 NULL，或者是<表达式>。“DEFAULT”表示按表定义中关于列默认值的定义填入。NULL 表示填入“空”，只有表定义中允许为空值的列允许填入。<表达式>指当前可以计算得到具体数据的计算式或函数式。

2. 修改数据语句

语句基本结构：UPDATE {<表名> | <视图名>} SET {列名=<表达式> | DEFAULT | NULL} [FROM {<表名>}] [WHERE <条件表达式>]

【例 6.6】 如果有成绩表 1（学号，姓名，总分）和成绩表 2（学号，姓名，C 语言分数），要求更新成绩 1 表中的总分，等于相同学号记录的原总分加 C 语言分数。

UPDATE 成绩表 1 SET 成绩表 1.总分=成绩表 1.总分+ 成绩表 2.C 语言分数 WHERE 成绩表 1.学号=成绩表 2.学号

3. 查询语句

语句基本结构：

SELECT {<表达式>} [INTO <新表名>] FROM {<表名>} [WHERE <选择条件表达式>] [GROUP BY {<分组字段名称>}] [HAVING <分组条件表达式>] [ORDER BY {<排序字段> [ASC | DESC]}] [COMPUTE <聚集函数>] [FOR < BROWSE 或 XML 选项>]

说明：

（1）INTO <新表名>子句将查询结果传入一个新数据表中，将产生一个新数据表。

（2）COMPUTE <聚集函数>的结构为：

COMPUTE { { AVG | COUNT | MAX | MIN | STDEV | STDEVP | VAR | VARP | SUM } (<表达式>) } [BY <表达式>]

其功能是生成统计数据作为附加的汇总列出现在结果集的最后。当与 BY 一起使用时，可以在结果集内生成分类汇总。在同一查询内可同时指定 COMPUTE BY 和 COMPUTE。

其中各聚集函数的意义：AVG 表示数字表达式中所有值的平均值；COUNT 表示记录条数；MAX 表示表达式中的最大值；MIN 表示表达式中的最小值；STDEV 表示表达式中所有值的统计标准偏差；STDEVP 表示表达式中所有值的填充统计标准偏差；SUM 表示数字表达

式中所有值的和；VAR 表示表达式中所有值的统计方差；VARP 表示表达式中所有值的填充统计方差。

BY <表达式>中的<表达式>通常是列名，表示分类汇总的分组字段。

（3）FOR < BROWSE 或 XML 选项>子句格式为：

FOR { BROWSE | XML { RAW | AUTO | EXPLICIT } [, XMLDATA] [, ELEMENTS] [, BINARY BASE64]}

说明：

1）FOR BROWSE 指定当查看 DB-Library 浏览模式游标中的数据时允许更新。如果表包含时间戳列（用 timestamp 数据类型定义的列），表有唯一索引且 FOR BROWSE 选项在 SELECT 语句的最后发送到 SQL Server，则可以在应用程序中浏览该表。

2）FOR XML 指定查询结果将作为 XML 文档返回。必须指定下列 XML 模式之一：RAW、AUTO、EXPLICIT。其中，RAW：获得查询结果并将结果集内的各行转换为 XML 元素，用一般标识符 <row /> 作为元素标记。AUTO：以简单的嵌套 XML 树返回查询结果。在 FROM 子句内，每个在 SELECT 子句中至少有一列被列出的表都表示为一个 XML 元素。SELECT 子句中列出的列映射到适当的元素特性。EXPLICIT：指定显式定义所得到的 XML 树的形状。使用此种模式，要求以一种特定的方式编写查询，以便显式指定有关期望的嵌套的附加信息。

3）XMLDATA 表示返回架构，但不将根元素添加到结果中。如果指定了 XMLDATA，它将被追加到文档上。

4）ELEMENTS 表示指定列作为子元素返回。否则，列将映射到 XML 特性。

5）BINARY BASE64 表示指定查询返回二进制 base64 编码格式的二进制数据。使用 RAW 和 EXPLICIT 模式检索二进制数据时，必须指定该选项。这是 AUTO 模式中的默认值。

6）使用 UNION 运算符可以将两个或更多查询的结果组合为单个结果集，该结果集包含联合查询中的所有查询的全部行，实现求关系并集的操作。使用 UNION 组合两个查询的结果集的两个基本规则是：①所有查询中的列数和列的顺序必须相同；②数据类型必须兼容。

语句结构：<查询语句 1> UNION [ALL] <查询语句 2> [UNION [ALL] ……]

其中 ALL 表示在组合过程中不删除重复行。

【例 6.7】 如果有表 titles，包括 title、type、price、advance、ytd_sales 等字段，显示含有本年度截止到现在的当前销售额的行，然后按 type 分类以递减顺序计算书籍的平均价格和预付款总额，最后计算全部书籍的平均价格和预付款总额。

```
SELECT CAST(title AS char(20)) AS title, type, price, advance FROM titles WHERE ytd_sales IS NOT NULL ORDER BY type DESC COMPUTE AVG(price), SUM(advance) BY type COMPUTE SUM(price), SUM(advance)
```

6.2.3　数据控制语言（DCL）

数据控制语言（DCL）是用来设置或者更改数据库用户或角色权限的语句，包括 GRANT、DENY、REVOKE 等语句，在默认状态下，只有 sysadmin、dbcreator、db_owner 或 db_securityadmin 等角色的成员才有权利执行数据控制语言。

1. GRANT 语句

GRANT 语句是授权语句，它可以把语句权限或者对象权限授予给其他用户和角色。

（1）授予语句权限的语句。

语句格式：GRANT {ALL | <语句>} TO <用户名>

说明：“语句”包括：CREATE DATABASE、CREATE DEFAULT、CREATE FUNCTION、CREATE PROCEDURE、CREATE RULE、CREATE TABLE、CREATE VIEW、BACKUP DATABASE、BACKUP LOG。

（2）授予对象权限的语句。

语句格式：GRANT { ALL [PRIVILEGES] | <对象权限>}{[(<列名>) ON {<表名> | <视图名>} | ON {<表名> | <视图名>} [(<列名>)] | ON { <存储过程> | <扩展程序>} | ON { <用户定义函数>}] TO <用户名> [WITH GRANT OPTION] [AS {<组> | <角色>}]

说明：

1）“对象权限”是当前授予的对象权限。当在表、表值函数或视图上授予对象权限时，权限列表可以包括这些权限中的一个或多个：SELECT、INSERT、DELETE、REFERENCES 或 UPDATE。列列表可以与 SELECT 和 UPDATE 权限一起提供。如果列列表未与 SELECT 和 UPDATE 权限一起提供，那么该权限应用于表、视图或表值函数中的所有列。在存储过程上授予的对象权限只可以包括 EXECUTE。

2）WITH GRANT OPTION 表示给予用户将指定的对象权限授予~~其它~~其他安全账户的能力。

3）AS {<组> | <角色>}指当前数据库中有执行 GRANT 语句权力的安全账户的可选名。当对象上的权限被授予一个组或角色时使用 AS，对象权限需要进一步授予不是组或角色的成员的用户。因为只有用户（而不是组或角色）可执行 GRANT 语句，组或角色的特定成员授予组或角色权力之下的对象的权限。

2. DENY 语句

DENY 语句用于拒绝给当前数据库内的用户或者角色授予权限，并防止用户或角色通过其组或角色成员继承权限。

（1）否定语句权限的语句。

语句格式：DENY { ALL | <语句>} TO <用户名>

（2）否定对象权限的语句。

语句格式：DENY { ALL [PRIVILEGES] | <对象权限>} { [(<列名>)] ON {<表名> | <视图名>} | ON {<表名> | <视图名>} [(<列名>)] | ON { <存储过程> | <扩展程序>} | ON {<用户定义函数>} } TO <用户名> [CASCADE]

3. REVOKE 语句

REVOKE 语句是与 GRANT 语句相反的语句，它能够将以前对当前数据库内的用户或者角色上授予或拒绝的权限删除，但是该语句并不影响用户或者角色从其他角色中作为成员继承过来的权限。

（1）收回语句权限的语法的语句。

语句格式：REVOKE { ALL | <语句> } FROM <用户名>

（2）收回对象权限的语法的语句。

语句格式：REVOKE [GRANT OPTION FOR] { ALL [PRIVILEGES] | <对象权限>} { [(<列名>)] } ON {<表名> | <视图名>} | ON {<表名> | <视图名>} [(<列名>)] | ON {<存储过程> | <扩展程序>} | ON { <用户定义函数>} { TO | FROM }<用户名> [CASCADE] [AS {<组> | <角色>}]

6.2.4　其它其他语言元素

在 Transact-SQL 语言中嵌入了其他程序设计语言的元素，用其到存储过程与触发器之中。

1. 注释

注释是程序代码中不执行的文本字符串（也称为注解）。在 SQL Server 中，可以使用两种类型的注释字符：一种是 ANSI 标准的注释符“--”，它用于单行注释；另一种是与 C 语言相同的程序注释符号，即“/* */”。

2. 变量

变量是一种语言中必不可少的组成部分。Transact-SQL 语言中有两种形式的变量，一种是用户自己定义的局部变量，另一种是系统提供的全局变量。

（1）局部变量。局部变量是一个能够拥有特定数据类型的对象，它的作用范围仅限制在程序内部。局部变量可以作为计数器来计算循环执行的次数，或是控制循环执行的次数。另外，利用局部变量还可以保存数据值，以供控制流语句测试以及保存由存储过程返回的数据值。局部变量被引用时要在其名称前加上标志“@”，而且必须先用 DECLARE 命令定义后才可以使用。

（2）全局变量。全局变量是 SQL Server 系统内部使用的变量，其作用范围并不仅仅局限于某一程序，而是任何程序均可以随时调用。全局变量通常存储一些 SQL Server 的配置设定值和统计数据。用户可以在程序中用全局变量来测试系统的设定值或者是 Transact-SQL 命令执行后的状态值。

使用全局变量时应该注意以下几点：

1）全局变量不是由用户的程序定义的，它们是在服务器级定义的。

2）用户只能使用预先定义的全局变量。

3）引用全局变量时，必须以标记符“@@”开头。

4）局部变量的名称不能与全局变量的名称相同，否则会在应用程序中出现不可预测的结果。

3. 运算符

运算符是一些符号，它们能够用来执行算术运算、字符串连接、赋值以及在字段、常量和变量之间进行比较。在 SQL Server 中，运算符主要有以下六大类：算术运算符、赋值运算符、位运算符、比较运算符、逻辑运算符以及字符串串联运算符。

（1）算术运算符。算术运算符可以在两个表达式上执行数学运算，这两个表达式可以是数字数据类型分类的任何数据类型。算术运算符包括：加（+）、减（-）、乘（*）、除（/）和取模（%）。

（2）赋值运算符。Transact-SQL 中只有一个赋值运算符，即等号（=）。赋值运算符使我们能够将数据值指派给特定的对象。另外，还可以使用赋值运算符在列标题和为列定义值的表达式之间建立关系。

（3）位运算符。位运算符使我们能够在整型数据或者二进制数据（image 数据类型除外）之间执行位操作。此外，在位运算符左右两侧的操作数不能同时是二进制数据。位运算符包括：^、&、|。

（4）比较运算符。比较运算符用于比较两个表达式的大小或是否相同，其比较的结果是布尔值，即 TRUE（表示表达式的结果为真）、FALSE（表示表达式的结果为假）以及 UNKNOWN。除了 text、ntext 或 image 数据类型的表达式外，比较运算符可以用于所有的表达式。比较运算符包括：=、>、<、>=、<=、<>、!=、!>、!<。

（5）逻辑运算符。逻辑运算符可以把多个逻辑表达式连接起来。逻辑运算符包括 AND、OR 和 NOT 等运算符。逻辑运算符和比较运算符一样，返回带有 TRUE 或 FALSE 值的布尔数据类型。逻辑运算符包括：AND、OR、NOT。

（6）字符串串联运算符。字符串串联运算符允许通过加号（+）进行字符串串联，这个加号即被称为字符串串联运算符。例如对于语句 SELECT 'abc'+'def'，其结果为 abcdef。

运算符的优先等级从高到低如下所示。

括号：（）

乘、除、求模运算符：*、/、%

加减运算符：+、－

比较运算符：=、>、<、>=、<=、<>、!=、!>、!<

位运算符：^、&、|;

逻辑运算符：NOT、AND、OR

4. 函数

在 Transact-SQL 语言中，函数被用来执行一些特殊的运算以支持 SQL Server 的标准命令。Transact-SQL 编程语言提供了三种函数：

（1）行集函数：行集函数可以在 Transact-SQL 语句中当作表引用。

（2）聚集函数：聚集函数用于对一组值执行计算并返回一个单一的值。

（3）标量函数：标量函数用于对传递给它的一个或者多个参数值进行处理和计算，并返回一个单一的值。

SQL Server 中最常用的几种函数：

（1）字符串函数。字符串函数可以对二进制数据、字符串和表达式执行不同的运算，大多数字符串函数只能用于 char 和 varchar 数据类型以及明确转换成 char 和 varchar 的数据类型，少数几个字符串函数也可以用于 binary 和 varbinary 数据类型。此外，某些字符串函数还能够处理 text、ntext、image 数据类型的数据。

字符串函数的分类：

1）基本字符串函数：

UPPER(<字符串>)：将串中小写字符变大写字符。

LOWER(<字符串>)：将串中大写字符变小写字符。

SPACE(<整数>)：产生“整数”个空格。

REPLICATE(<字符串>,<整数>)：将字符串重复“整数”次。

STUFF(<字符串 1>,<数字>,<整数>,<字符串 2>)：将字符串 1 中从“数字”开始的整数个字符用字符串 2 代替。

REVERSE(<字符串表达式>)：反转字符串表达式。

LTRIM(<字符串>)：删除字符串前面空格。

RTRIM(<字符串>)：删除字符串后面空格。

2）字符串查找函数：

CHARINDEX(<字符串 1>,<字符串 2>)：在字符串 2 中搜索字符串 1 的起始位置。

PATINDEX('%<字串>%',<字符串>)：在字符串中搜索字串出现的起始位置。

3）长度和分析函数：

SUBSTRING(<字符串 1>,<数字>,<整数>)：从字符串 1 中取从“数字”开始的长度为“整数”的字符串。

RIGHT(<字符串>,<整数>)：从字符串右边取长度为“整数”的字符串。

LEFT(<字符串>,<整数>)：从字符串左边取长度为“整数”的字符串。

4）转换函数：

ASCII(<字符>)：求“字符”的 ASCII 码。

CHAR(<整数>)：求 ASCII 码等于“整数”的字符。

STR(<数值表达式>[,<整数>[,<小数位>]])：将数值表达式的值变成长度等于“整数”、小数位位数等于“小数位”的字符串。

（2）日期和时间函数。日期和时间函数用于对日期和时间数据进行各种不同的处理和运算，并返回一个字符串、数字值或日期和时间值。

DATEADD(<参数>,<数字>,<日期>)：按参数指定部分计算日期与数字之和并返回。参数可以为：Year、quarter、Month、dayofyear、Day、Week、Hour、minute、second、millisecond。

DATEDIFF(<参数>,<日期 1>,<日期 2>)：按参数指定部分计算日期与日期之差。

DATENAME(<参数>,<日期>)：按参数指定部分返回日期相应部分的字符串。

DATEPART(<参数>,<日期>)：按参数指定部分返回日期相应部分的整数值。

GETDATE(<日期>)：返回系统日期。

DAY(<日期>)：返回日期表达式的日期数据。

MONTH(<日期>)：返回日期表达式的月份数据。

YEAR(<日期>)：返回日期表达式的年份数据。

（3）数学函数。数学函数用于对数字表达式进行数学运算并返回运算结果。数学函数可以对 SQL Server 提供的数字数据（decimal、integer、float、real、money、smallmoney、smallint 和 tinyint）进行处理。

数学函数有：

ABS(n)：求 n 的绝对值。

CEILING(n)：求大于等于 n 的最小整数。

DEGREES(n)：求弧度 n 的度数值。

FLOOR(n)：求小于等于 n 的最大整数。

POWER(n,m)：求 n 的 m 次方。

RADIANS(n)：求度数 n 的弧度值。

SIGN(n)：求 n 的符号，分别用 1、-1、0 表示正数、负数、0。

EXP(n)：求 n 的指数值。

LOG(n)：求 n 的对数值。

LOG10(n)：求 n 的以 10 为底的对数值。

SQUARE(n)：求 n 的平方。

SQRT(n)：求 n 的平方根。

SIN(n)：求 n 的正弦值。

COS(n)：求 n 的余弦值。

TAN(n)：求 n 的正切值。

PI()：返回 π 的值。

RAND()：产生随机数。

MOD(m,n)：求 m 除以 n 的余数。

ROUND(n,m)：对 n 作四舍五入处理，保留 m 位。

（4）转换函数。一般情况下，SQL Server 会自动处理某些数据类型的转换。例如，如果比较 char 和 datetime 表达式、smallint 和 int 表达式、或不同长度的 char 表达式，SQL Server 可以将它们自动转换，这种转换被称为隐性转换。但是，无法由 SQL Server 自动转换的或者是 SQL Server 自动转换的结果不符合预期结果的，就需要使用转换函数做显式转换。

CONVERT(<转换后数据类型>[(<长度>)],<表达式>[,<转换样式>])：将表达式的值转换为“转换后数据类型”指定的数据类型，“长度”表示转换后的数据长度，“转换样式”只在表达式为日期时间类型且欲转换为字符类型时使用，给出转换成字符类型的样式。

转换后数据类型用 SQL Server 数据类型描述符表示。

CAST(<表达式>,<转换后数据类型>)：将表达式的值转换为“转换后数据类型”指定的数据类型。

（5）系统函数。系统函数用于返回有关 SQL Server 系统、用户、数据库和数据库对象的信息。系统函数可以让用户在得到信息后，使用条件语句，根据返回的信息进行不同的操作。与~~其它~~其他函数一样，可以在 SELECT 语句的 SELECT 和 WHERE 子句以及表达式中使用系统函数。

DB_ID(<名称>)：返回数据库 ID 号。

DB_NAME(<ID 号>)：返回数据库名称。

HOST_ID(<名称>)：返回主机 ID 号。

HOST_NAME(<ID 号>)：返回主机名称。

OBJECT_ID(<名称>)：返回指定对象 ID 号。

OBJECT_NAME(<ID 号>)：返回指定对象名称。

SUSER_ID(<名称>)：返回指定登录 ID 号。

SUSER_NAME(<ID 号>)：返回指定登录的名称。

USER_ID(<名称>)：返回指定用户 ID 号。

USER_NAME(<ID 号>)：返回指定用户名称。

COL_NAME(<表号>,<列号>)：返回列名。

COL_LENGTH(<表名>,<列名>)：返回列定义长度。

DATALENGTH(<表达式>)：返回表达式占用的字节长度。

（6）聚集函数。聚集函数可以返回整个或者几个列或者一个列的汇总数据，它常用来计算

SELECT 语句查询的统计值。聚集函数经常与 SELECT 语句的 GROUP BY 子句一同使用。

聚集函数包括：AVG、COUNT、MAX、MIN、SUM。

5. 流程控制语句

流程控制语句是指那些用来控制程序执行和流程分支的命令，在 SQL Server 中，流程控制语句主要用来控制 SQL 语句、语句块或者存储过程的执行流程。

（1）BEGIN…END 语句。BEGIN…END 语句能够将多个 Transact-SQL 语句组合成一个语句块，并将它们视为一个单元处理。在条件语句和循环语句等控制流程语句中，当符合特定条件便要执行两个或者多个语句时，就需要使用 BEGIN…END 语句

语句格式：

BEGIN { <执行语句> | <语句块> } END

（2）IF…ELSE 语句。IF…ELSE 语句是条件判断语句，其中，ELSE 子句是可选的，最简单的 IF 语句没有 ELSE 子句部分。IF…ELSE 语句用来判断当某一条件成立时执行某段程序，条件不成立时执行另一段程序。SQL Server 允许嵌套使用 IF…ELSE 语句，而且嵌套层数没有限制。

语句格式：

IF <条件表达式> {<执行语句> | <语句块>} [ELSE {<执行语句> | <语句块> }]

（3）CASE 语句。CASE 语句可以计算多个条件式，并将其中一个符合条件的结果表达式返回。CASE 语句按照使用形式的不同，可以分为简单 CASE 语句和搜索 CASE 语句。

语句格式：

```
CASE <表达式>
   WHEN <值 1> THEN <表达式 1>
   ……
   WHEN <值 n> THEN <表达式 n>
    [ELSE <表达式 n+1 >]
END
```

意义：如果“表达式”的值等于值 1，则返回表达式 1 的值且跳出 CASE 语句，否则继续检查“表达式”的值是否等于值 2，……。如果与值 n 也不相等且有 ELSE 子句，则返回表达式 n+1 的值。

CASE 函数语句还有一种格式：

```
CASE
   WHEN <条件表达式 1> THEN <表达式 1>
   ……
   WHEN <条件表达式 n> THEN <表达式 n>
    [ELSE <表达式 n+1> ]
END
```

意义：如果“条件表达式 1”的值为真，则返回表达式 1 的值且跳出 CASE 语句，否则继续检查“条件表达式 2”的值是否为真，……。如果所有条件表达式的值都不为真且有 ELSE 子句，则返回表达式 n+1 的值。

（4）WHILE…CONTINUE…BREAK 语句。WHILE…CONTINUE…BREAK 语句用于设

置重复执行 SQL 语句或语句块的条件。只要指定的条件为真，就重复执行语句。其中，CONTINUE 语句可以使程序跳过 CONTINUE 语句后面的语句，回到 WHILE 循环的第一行命令。BREAK 语句则使程序完全跳出循环，结束 WHILE 语句的执行。

语句格式：

WHILE 条件表达式{ <执行语句 1> | <语句块 1> }[BREAK] [CONTINUE] { <执行语句 2> | <语句块 2> }

意义：只要“条件表达式”值为真，就重复执行<执行语句 1>或<语句块 1>。在<执行语句 1> | <语句块 1>中还应当有条件语句，控制是否执行 BREAK 或 CONTINUE，如果执行 BREAK，则停止循环。如果执行 CONTINUE，则停止本次循环，转至<执行语句 1> | <语句块 1>的首条语句继续执行。

（5）GOTO 语句。GOTO 语句可以使程序直接跳到指定的标有标识符的位置处继续执行，而位于 GOTO 语句和标识符之间的程序将不会被执行。GOTO 语句和标识符可以用在语句块、批处理和存储过程中，标识符可以为数字与字符的组合，但必须以“：”结尾。

语句结构：

GOTO <标识符>

……

<标识符>：

【例 6.8】利用 GOTO 语句求出从 1 加到 5 的总和。

```
declare @sum int, @count int
select @sum=0, @count=1
label_1:
select @sum=@sum+@count
select @count=@count+1
if @count<=5
goto label_1
select @count @sum
```

（6）WAITFOR 语句。WAITFOR 语句用于暂时停止执行 SQL 语句、语句块或者存储过程等，直到所设定的时间已过或者所设定的时间已到才继续执行。

语句格式：

WAITFOR { DELAY '<时间间隔>' | TIME '<时间>' }

其中，DELAY 用于指定时间间隔，TIME 用于指定某一时刻，其数据类型为 datetime，格式为'hh:mm:ss'。

（7）RETURN 语句。RETURN 语句用于无条件地终止一个查询、存储过程或者批处理，此时位于 RETURN 语句之后的程序将不会被执行。

语句格式：

RETURN [<整数>]

其中，参数“整数”为返回的整型值。存储过程可以给调用过程或应用程序返回整型值。

6.3　SQL Server 中的存储过程

在大型数据库系统中，存储过程和触发器具有很重要的作用。无论是存储过程还是触发器，都是 SQL 语句和流程控制语句的集合。就本质而言，触发器也是一种存储过程。存储过程在运算时生成执行方式，所以以后对其再运行时执行速度很快。SQL Server 不仅提供了用户自定义存储过程的功能，而且提供了许多可作为工具使用的系统存储过程。

6.3.1　存储过程的概念

存储过程（Stored Procedure）是一组为了完成特定功能的 Transact-SQL 语句集，经编译后存储在数据库中。用户通过存储过程的名字并给出参数（如果该存储过程带有参数）来执行它。

在 SQL Server 系列版本中存储过程分为两类：系统提供的存储过程和用户自定义的存储过程。系统存储过程主要存储在 master 数据库中，并以 sp_为前缀，它从系统表中获取信息，为系统管理员管理 SQL Server 提供支持。通过系统存储过程，SQL Server 中的许多管理性或信息性的活动（如了解数据库对象、数据库信息）都可以被顺利有效地完成。系统存储过程可以在~~其它~~其他数据库中被调用，在调用时不必在存储过程名前加上数据库名。而且当创建一个新数据库时，一些系统存储过程会在新数据库中被自动创建。

用户自定义存储过程是由用户创建并能完成某一特定功能（如查询用户所需数据信息）的存储过程。本节主要介绍用户自定义存储过程。

6.3.2　存储过程的优点

当利用 SQL Server 创建一个应用程序时，Transact-SQL 是一种主要的编程语言。若运用 Transact-SQL 来进行编程，有两种方法。

（1）在本地存储包含 Transact-SQL 语句的应用程序，通过 Transact-SQL 语句向 SQL Server 发送命令来进行数据处理。

（2）把部分用 Transact-SQL 编写的程序作为存储过程存储在 SQL Server 中，另外创建应用程序调用这些存储过程对数据进行处理，包括对数据库的操作，不同参数驱使存储过程向调用者返回不同格式或内容的结果集。同时，返回状态值给调用者，指明调用是成功或是失败。

存储过程可以嵌套，允许在一个存储过程中调用另一存储过程。

我们通常更偏爱于使用第二种方法，即在 SQL Server 中使用存储过程而不是在客户计算机上调用 Transact-SQL 编写的一段程序，原因在于存储过程具有以下优点：

1. 存储过程实现模块化程序设计

存储过程在被创建以后可以在程序中被多次调用，而不必重新编写该存储过程的 SQL 语句。而且数据库专业人员可随时对存储过程进行修改，但对应用程序源代码毫无影响（因为应用程序源代码只包含存储过程的调用语句），从而极大地提高了程序的可移植性。

2. 存储过程能够实现较快的执行速度

如果某一操作包含大量的 Transact-SQL 代码或被多次执行，那么存储过程要比批处理的执行速度快很多。因为存储过程是预编译的，在首次运行一个存储过程时，查询优化器对其进行分析、优化，并给出最终被存在系统表中的执行计划。而批处理的 Transact-SQL 语句在每

次运行时都要进行编译和优化，速度相对要慢一些。

3. 存储过程能够减少网络流量

对于同一个针对数据库对象的操作（如查询、修改），如果这一操作所涉及到的Transact-SQL 语句被组织成一个存储过程，那么当在客户计算机上调用该存储过程时，网络中传送的只是该调用语句，而不是多条 SQL 语句，从而减少网络流量，降低网络负载。

4. 存储过程可被作为一种安全机制使用

系统管理员通过对存储过程的执行权限进行限制，从而实现对相应数据访问权限的限制，避免非授权用户对数据的访问，保证数据的安全。

注意：存储过程虽然既有参数又有返回值，但是它与函数不同。存储过程的返回值只是指明执行是否成功，并且它不能像函数那样被直接调用，在调用存储过程时，在存储过程名字前一定要有 EXEC 保留字。

在 SQL Server 中，创建一个存储过程有两种方法：

（1）使用 Transact-SQL 命令 Create Procedure。

（2）使用图形化管理工具 Enterprise Manager。

用 Transact-SQL 创建存储过程是一种较为快速的方法，但对于初学者，使用 Enterprise Manager 更易理解，更为简单。

当创建存储过程时，需要确定存储过程的三个组成部分：

（1）所有的输入参数以及传给调用者的输出参数。

（2）被执行的针对数据库的操作语句，包括调用~~其它~~其他存储过程的语句。

（3）返回给调用者的状态值，以指明调用是成功还是失败。

6.3.3 使用 Transact-SQL 命令创建存储过程

在创建存储过程之前，应该考虑到以下几个方面：

（1）在一个批处理中，Create Procedure 语句不能与~~其它~~其他 SQL 语句合并在一起。

（2）数据库所有者具有默认的创建存储过程的权限，它可把该权限传递给~~其它~~其他的用户。

（3）存储过程作为数据库对象，其命名必须符合命名规则。

（4）只能在当前数据库中创建属于当前数据库的存储过程。

用 Create Procedure 创建存储过程的语句格式为：

CREATE PROC[EDURE] <存储过程名> [;<分组数字>]

[{ @<参数> <参数数据类型>} [VARYING] [= <参数默认值>] [OUTPUT]] [,...n]

[WITH {RECOMPILE | ENCRYPTION | RECOMPILE, ENCRYPTION}] [FOR REPLICATION]

AS <SQL 语句> [...n]

说明：

（1）存储过程的名称必须符合标识符规则，且对于数据库及其所有者必须唯一。要创建局部临时过程，可以在<存储过程名>前面加一个编号符（#<存储过程名>），要创建全局临时过程，可以在<存储过程名>前面加两个编号符（##<存储过程名>）。完整的名称（包括#或##）不能超过 128 个字符。指定过程所有者的名称是可选的。

（2）过程中的参数可以有一个或多个。用户必须在执行过程时提供每个所声明参数的值（除非定义了该参数的默认值）。存储过程最多可以有 2100 个参数。

要求使用@符号作为第一个字符来指定参数名称。参数名称必须符合标识符的规则。每个过程的参数仅用于该过程本身；相同的参数名称可以用在~~其它~~其他过程中。默认情况下，参数只能代替常量，而不能用于代替表名、列名或~~其它~~其他数据库对象的名称。

（3）参数的数据类型可以是所有数据类型（包括 text、ntext 和 image）。不过，cursor 数据类型只能用于 OUTPUT 参数。如果指定的数据类型为 cursor，也必须同时指定 VARYING 和 OUTPUT 关键字。VARYING 指定由 OUTPUT 参数支持的结果集，仅应用于游标型参数。

（4）如果定义了参数的缺省值，那么即使不给出参数值，该存储过程仍能被调用。缺省值必须是常数，或者是空值。

（5）OUTPUT。表明该参数是一个返回参数。用 OUTPUT 参数可以向调用者返回信息。Text 类型参数不能用作 OUTPUT 参数。

（6）RECOMPILE 指明 SQL Server 不保存该存储过程的执行计划，该存储过程每执行一次都要重新编译。

（7）ENCRYPTION 表明 SQL Server 加密了 syscomments 表，该表的 text 字段是包含有 Create procedure 语句的存储过程文本，使用该关键字无法通过查看 syscomments 表来查看存储过程内容。

（8）FOR REPLICATION 选项指明了为复制创建的存储过程不能在订购服务器上执行，只有在创建过滤存储过程时（仅当进行数据复制时过滤存储过程才被执行），才使用该选项。FOR REPLICATION 与 WITH RECOMPILE 选项是互不兼容的。

（9）AS 指明该存储过程将要执行的动作。

（10）<SQL 语句>是任何数量和类型包含在存储过程中的 SQL 语句。[...n]表示 1 到多个。

一个存储过程的最大尺寸为 128M，用户定义的存储过程必须创建在当前数据库中。下面通过示例来详细介绍如何创建包含有各种保留字的存储过程。

【例 6.9】 创建数据库 waremanege 的存储过程：返回关于出版物的表 publishers 中所有作者以及他们的文章和说明。

在查询分析器中输入以下代码：

```
USE waremanage
GO
CREATE PROCEDURE pub_infor
AS
SELECT pub_name,author,unit FROM publishers
GO
```

单击工具条中“执行查询”按钮，将生成存储过程：pub_infor。

【例 6.10】在存储过程中使用参数 int_unitprice，将来在调用时只要给出 int_unitprice 的值，就能显示所有单价大于等于该值的出版物名称及其作者和说明。

```
USE waremanage
GO
CREATE PROCEDURE pub_infor @int_unitprice INT
AS
```

```
SELECT pub_name,author,unit FROM publishers WHERE unitprice>=@int_unitprice
GO
```

然后在查询分析器中调用已创建的存储过程 pub_infor，求显示所有单价大于等于 200 的记录的相关信息：首先选择“查询”，选择“更改数据库”，选择数据库，再输入如下语句：

```
declare @int_unitprice int
EXEC pub_infor @int_unitprice = 200
```

再单击“执行”按钮，将看见执行结果。

6.3.4 使用企业管理器创建存储过程

使用企业管理器创建存储过程的步骤如下：

（1）启动企业管理器，选择要使用的服务器。

（2）展开要创建存储过程的数据库，在左窗格中单击存储过程文件夹，右窗格中将显示该数据库的所有存储过程，如图 6.35 所示。右击存储过程文件夹，在弹出菜单中选择新建存储过程，此时打开“存储过程属性”对话框（见图 6.36）。

图 6.35 企业管理器中显示的存储过程详细信息

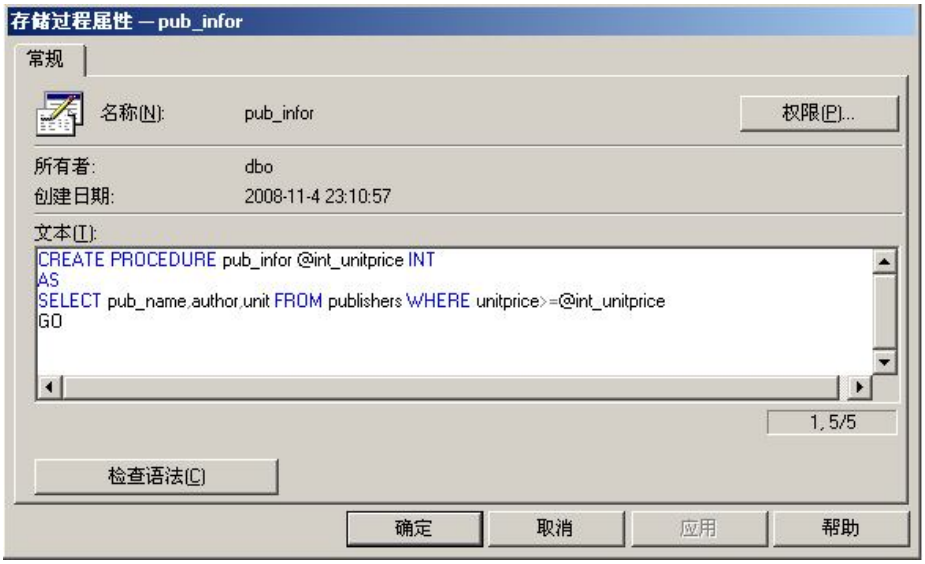

图 6.36　“存储过程属性”对话框

（3）在“文本”列表框中预置了一条默认语句，用户可将之换为自己的代码，例如：

```
CREATE PROCEDURE pub_infor @int_unitprice INT
AS
SELECT pub_name,author,unit FROM publishers WHERE unitprice>=@int_unitprice
GO
```

如图 6.35 所示。

（4）单击“检查语法”按钮，检查语法是否正确。

（5）单击“确定”按钮，保存。

（6）在右窗格中，右击该存储过程，在弹出菜单中选择所有任务，选择管理权限，设置权限，如图 6.37 所示。

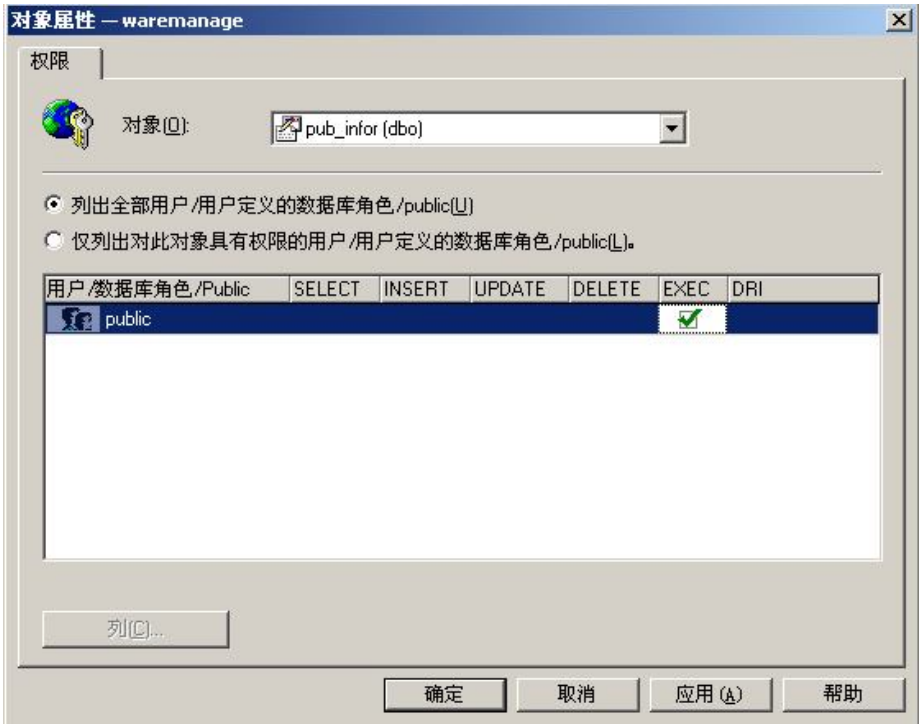

图 6.37 设置存储过程权限对话框

如果双击存储过程名称，可以重新调出存储过程属性对话框。

6.3.5 重新命名存储过程

修改存储过程的名字使用系统存储过程 sp_rename。其命令格式为：

sp_rename <原存储过程名>, <新存储过程名>

示例：将存储过程 reptq1 修改为 newproc：sp_rename reptq1, newproc

另外，通过 Enterprise Manager 也可修改存储过程的名字，其操作过程与 Windows 下修改文件名字的操作类似。即首先选中需修改名字的存储过程，然后右击，在弹出菜单中选取重命名选项，最后输入新存储过程的名字。

6.3.6 删除存储过程

删除存储过程使用 drop 命令，drop 命令可将一个或多个存储过程或者存储过程组从当前数据库中删除。其语法规则为：

DROP PROCEDURE {<存储过程名>} [,…n]

6.3.7 执行存储过程

执行已创建的存储过程使用 EXECUTE 命令，其语法如下：

[EXECUTE]

[@<整型变量>=]

{<存储过程名>[;<分组数字>] | @<变量名>}

[[@<参数>=] {<值> | @<返回参数值> [OUTPUT] | [DEFAULT] [,…n]
[WITH RECOMPILE]

说明：

（1）整型变量用来存储存储过程向调用者返回的值。

（2）<变量名>是用来代表存储过程名字的变量。

其它其他参数和保留字的含义与 CREATE PROCEDURE 中介绍的一样。

6.3.8　系统存储过程

系统存储过程就是系统创建的存储过程，目的在于能够方便地从系统表中查询信息或完成与更新数据库表相关的管理任务或其它其他的系统管理任务。系统过程以“sp_”开头，在 Master 数据库中创建并保存在该数据库中，为数据库管理者所有。一些系统过程只能由系统管理员使用，而有些系统过程通过授权可以被其它其他用户所使用。

系统存储过程主要包括以下几类（见表 6.2）。

表 6.2　系统存储过程分类

分类	描述
Active Directory 过程	用于在 Microsoft Windows 2000 Active Directory 中注册 SQL Server 实例和 SQL Server 数据库
目录过程	执行 ODBC 数据字典功能，并隔离 ODBC 应用程序，使之不受基础系统表更改的影响
游标过程	执行游标变量功能
数据库维护计划过程	用于设置确保数据库性能所需的核心维护任务
分布式查询过程	用于执行和管理分布式查询
全文检索过程	用于执行和查询全文索引
日志传送过程	用于配置和管理日志传送
OLE 自动化过程	允许在标准 Transact-SQL 批处理中使用标准 OLE 自动化对象
复制过程	用于管理复制

续表

分类	描述
安全过程	用于管理安全性
SQL 邮件过程	用于从 SQL Server 内执行电子邮件操作
SQL 事件探查器过程	SQL 事件探查器用于监视性能和活动
SQL Server 代理程序过程	SQL Server 代理程序用于管理调度的活动和事件驱动活动
系统过程	用于 SQL Server 的常规维护
Web 助手过程	由 Web 助手使用
XML 过程	用于可扩展标记语言（XML）的文本管理
常规扩展过程	提供从 SQL Server 到外部程序的接口，以便进行各种维护活动

说明：除非特别指明，所有系统存储过程返回 0 值表示成功，返回非零值则表示失败。具体的 SQL Server 系统存储过程在此不做详细说明。

6.4 SQL Server 中的触发器

上面我们介绍了一般意义的存储过程，即用户自定义的存储过程和系统存储过程。本节将介绍一种特殊的存储过程，即触发器。在本节中我们将对触发器的概念、作用以及使用方法作详尽介绍，使读者了解如何定义触发器，创建和使用各种触发器。

6.4.1 触发器的概念及作用

一般存储过程是通过存储过程名字被程序调用而执行的，而触发器是在对数据库中数据进行维护操作事件时被执行。当对某一表进行如 UPDATE、INSERT、DELETE 这些操作时，SQL Server 就会自动执行触发器所定义的 SQL 语句，保证对数据的处理必须符合数据库所定义的规则，例如实现由主键和外键所要求的参照完整性和数据的一致性。除此之外，触发器的功能还可以有：

（1）强化约束（Enforce restriction）。触发器能够实现比 CHECK 语句更为复杂的约束。

（2）跟踪变化（Auditing changes）。触发器可以侦测数据库内的操作，从而不允许数据库中未经许可的更新和变化。

（3）级联运行（Cascaded operation）。触发器可以级联影响数据库的各项操作。例如，某个表上的触发器中包含有对另一个表的数据操作（如删除，更新，插入），而该操作又导致该表上触发器被触发。

（4）调用存储过程（Stored procedure invocation）。数据库更新时触发器可以调用一个或多个存储过程，甚至可以通过外部过程的调用而在 DBMS（数据库管理系统）本身之外进行操作。

由此可见，触发器可以解决高级形式的业务规则或复杂行为限制以及实现定制记录等一系列问题。例如，触发器能够找出某一表在数据修改前后状态发生的差异，并根据这种差异执行一定的处理。一个表的同一类型（INSERT、UPDATE、DELETE）的多个触发器能够对同一数据操作采取多种不同的处理。

当运行触发器时，系统处理的大部分时间花费在参照~~其它~~其他表的相关处理上，这些表可能既不在内存中也不在数据库设备上，与总是位于内存中的删除表和插入表的操作不一样，因此这些“~~其它~~其他表”的位置将决定操作所需的时间。

6.4.2 触发器的种类

SQL Server 支持两种类型的触发器：AFTER 触发器和 INSTEAD OF 触发器。其中 AFTER 触发器为 SQL Server 老版本中的触发器。该类型触发器只有在执行对表的某一操作（INSERT、UPDATE、DELETE）之后，触发器才被触发；可以使用系统过程 sp_settriggerorder 定义哪一个触发器先触发，哪一个后触发。

既可在表上定义，也可在视图上定义；对同一操作只能定义一个 INSTEAD OF 触发器。当为表或视图定义了针对某一操作（INSERT、DELETE、UPDATE）的 INSTEAD OF 类型触

发器且被执行时，尽管触发器被触发，但相应的操作并不被执行，运行的仅是触发器 SQL 语句本身。

6.4.3　创建触发器

可以用 SQL Server 管理工具 Enterprise Manager 和 Transact_SQL 两种方法创建触发器。

1. 用管理工具 Enterprise Manger 创建触发器

利用企业管理器创建触发器的操作步骤如下：

（1）选择服务器并展开。

（2）选择并展开数据库，选择表。

（3）右击选定的表，在弹出菜单中选择“所有任务”，单击“管理触发器”。进入“触发器属性”对话框，在“文本”框中预写入了触发器的格式：

```
CREATE TRIGGER [TRIGGER NAME] ON [dbo].[publishers]
FOR INSERT, UPDATE, DELETE
AS
```

如图 6.38 所示。

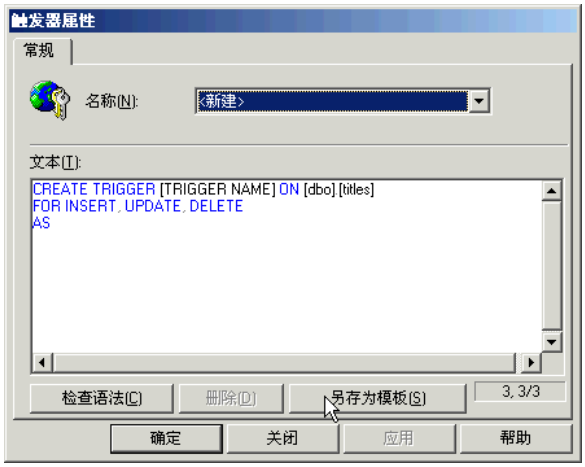

图 6.38　输入触发器 SQL 语句对话框

（4）在“名称”框中选择“新建”，在“文本”框中修改触发器文本。

（5）单击“检查语法”按钮检查语句是否正确。

（6）单击“应用”按钮，在“名称”下拉列表中会出现新创建的触发器名字。

（7）单击“确定”按钮完成创建。

2. 用 CREATE TRIGGER 命令创建触发器

命令格式：

```
CREATE TRIGGER <触发器名>
ON {<表名> | <视图名>} [WITH ENCRYPTION]
{
    { {FOR | AFTER | INSTEAD OF} {[INSERT] [ , ] [UPDATE] [ , ] [DELETE] }
```

```
        [WITH APPEND]
        [NOT FOR REPLICATION]
        AS
        [ {IF UPDATE (<列名>) [{AND | OR} UPDATE ({<列名>}) ]
            | IF (COLUMNS_UPDATED() {<位逻辑运算符>} <整型位掩码>)
                {<比较操作符>} {<被更新的列的位掩码>}
        } ]
        {<SQL 语句>}
    }
}
```

说明：

（1）WITH ENCRYPTION 表示对包含有 CREATE TRIGGER 文本的 syscomments 表进行加密。

（2）AFTER 表示只有在执行了指定的操作（INSERT、DELETE、UPDATE）之后触发器才被激活并执行触发器中的 SQL 语句。若使用关键字 FOR，则表示为 AFTER 触发器，该类型触发器仅能在表上创建。

（3）[DELETE][,][INSERT][,][UPDATE]关键字用来指明哪种数据操作将激活触发器。至少要指明一个选项，在触发器的定义中三者的顺序不受限制，且各选项要用逗号隔开。

（4）WITH APPEND 表明增加另外一个已存在的某一类型触发器。只有在兼容性水平（指某一数据库行为与以前版本的 SQL Server 兼容程度）不大于 65 时才使用该选项。

（5）NOT FOR REPLICATION 表明当复制处理修改与触发器相关联的表时，触发器不能被执行。

（6）AS 是触发器将要执行的动作。

（7）SQL 语句是包含在触发器中的条件语句或处理语句。触发器的条件语句定义了另外的标准来决定将被执行的 INSERT、DELETE、UPDATE 语句是否激活触发器。

（8）IF UPDATE 用来测定对某一确定列是插入操作还是更新操作，但不与删除操作用在一起。

（9）IF (COLUMNS_UPDATED())仅在 INSERT 和 UPDATE 类型的触发器中使用，用来检查所涉及的列是被更新还是被插入。

（10）位逻辑运算符用在比较中。

（11）整型位掩码用于那些被更新或插入的列。例如，如果表 T 包括 C1，C2，C3，C4，C5 五列。为了确定是否只有 C2 列被修改，可用 2 来做位掩码，如果想确定是否 C1，C2，C3，C4 都被修改，可用 14 来做位掩码。

（12）比较操作符：用“=”表示检查在整型位掩码中定义的所有列是否都被更新，用“>”表示检查在整型位掩码中定义的那些列是否被更新。

【例 6.11】当有人试图在 publishers 表中添加或更改数据时，要求向客户端显示一条消息，求设计带有提醒消息的触发器。

```
USE waremanage
GO
```

```
CREATE TRIGGER reminder
ON publishers
FOR INSERT, UPDATE
AS RAISERROR (50009, 16, 10)
GO
```

说明：RAISERROR 是返回用户定义的错误信息的语句，50009 是 sysmessages 中的用户定义消息，16 是用户定义的与消息关联的严重级别，10 是为从 1 到 127 间的一个整数，表示有关错误调用状态的信息。

【例 6.12】如果数据库 pubs 中有 employee 和 jobs 两个表，求创建一个触发器，当插入或更新雇员工作级别（job_lvls）时，该触发器检查指定雇员的工作级别（由此决定薪水）是否处于该工作定义的范围内。若要获得适当的范围，必须引用 jobs 表。

```
USE pubs
IF EXISTS (SELECT name FROM sysobjects
     WHERE name = 'employee_insupd' AND type = 'TR')
   DROP TRIGGER employee_insupd
GO
CREATE TRIGGER employee_insupd
ON employee
FOR INSERT, UPDATE
AS
DECLARE @min_lvl tinyint,
   @max_lvl tinyint,
   @emp_lvl tinyint,
   @job_id smallint
SELECT @min_lvl = min_lvl,
   @max_lvl = max_lvl,
   @emp_lvl = i.job_lvl,
   @job_id = i.job_id
FROM employee e INNER JOIN inserted i ON e.emp_id = i.emp_id
   JOIN jobs j ON j.job_id = i.job_id
IF (@job_id = 1) and (@emp_lvl <> 10)
BEGIN
   RAISERROR ('Job id 1 expects the default level of 10.', 16, 1)
   ROLLBACK TRANSACTION
END
ELSE
IF NOT (@emp_lvl BETWEEN @min_lvl AND @max_lvl)
BEGIN
   RAISERROR ('The level for job_id:%d should be between %d and %d.',
      16, 1, @job_id, @min_lvl, @max_lvl)
   ROLLBACK TRANSACTION
END
```

3. 删除触发器

命令格式：DROP TRIGGER <触发器名>

【例 6.13】删除触发器 employee_insupd。

```
USE pubs
IF EXISTS (SELECT name FROM sysobjects
WHERE name = 'employee_insupd' AND type = 'TR')
DROP TRIGGER employee_insupd
GO
```

4. 创建触发器时需要注意的问题

（1）建立触发器语句必须是批处理的第一条语句。

（2）表的所有者具有创建触发器的权限，表的所有者不能把该权限转给~~其它~~其他用户。

（3）触发器是数据库对象，所以其命名必须符合命名规则。

（4）尽管在触发器的 SQL 语句中可以涉及~~其它~~其他数据库中的对象，但是，触发器只能创建在当前数据库中。

（5）虽然触发器可以基于视图或临时表，但不能在视图或临时表上创建触发器，而只能在基表或创建视图的表上创建触发器。

（6）由触发器的机制决定，一个触发器只能对应一个表。

（7）尽管 TRUNCATE TABLE 语句如同没有 WHERE 从句的 DELETE 语句，但是由于 TRUNCATE TABLE 语句没有被记入日志，所以该语句不能触发 DELETE 型触发器。

（8）WRITETEXT 语句不能触发 INSERT 或 UPDATE 型的触发器。

当创建一个触发器时，必须指定触发器的名字，在哪一个表上定义触发器，激活触发 器的修改语句，如 INSERT、DELETE、UPDATE。当然两个或三个不同的修改语句也可以触发同一个触发器，如 INSERT 和 UPDATE 语句都能激活同一个触发器。

6.4.4 触发器的原理

每个触发器有两个特殊的表：插入表 inserted 和删除表 deleted。这两个表是逻辑表，并且这两个表是由系统管理的，存储在内存中，不是存储在数据库中，因此不允许用户直接对其修改。这两个表的结构总是与被该触发器作用的表相同，且动态驻留在内存中，当触发器工作完成，这两个表也被删除。这两个表主要保存因用户操作而被影响到的原数据值或新数据值。它们是只读的，用户不能向这两个表写入内容，但可以引用表中的数据。例如可用如下语句查看 deleted 表中的信息：

```
select * from deleted
```

1. 插入表的功能

对一个定义了插入类型触发器的表来讲，一旦对该表执行了插入操作，那么对向该表插入的所有行来说，都有一个相应的副本存放到插入表中。即插入表就是用来存储向原表插入的内容。

2. 删除表的功能

对一个定义了删除类型触发器的表来讲，一旦对该表执行了删除操作，则将所有的删除行存放至删除表中。这样做的目的是，一旦触发器遇到了强迫它中止的语句被执行时，删除的那些行可以从删除表中得以恢复。

需要强调的是，更新操作包括两部分，即先将更新的内容去掉，然后将新值插入。因此对一个定义了更新类型触发器的表来讲，当报告更新操作时，在删除表中存放了旧值，再在插

入表中存放新值。

由于触发器仅当被定义的操作被执行时才被激活，即仅当在执行插入、删除和更新操作时，触发器才执行。每条 SQL 语句仅能激活触发器一次，可能存在一条语句影响多条记录的情况，可以使用变量@rowcount。该变量存储了一条 SQL 语句执行后所影响的记录数，可以使用该值对触发器的 SQL 语句执行后所影响的记录求合计值。一般来说，首先要用 IF 语句测试@@rowcount 的值以确定后面的语句是否执行。

6.4.5　INSTEAD OF 触发器

前面已经提到，SQL Server 支持 AFTER 和 INSTEAD OF 两种类型的触发器。其中 INSTEAD OF 触发器是 SQL Server 新添加的功能。其主要优点是使不可被修改的视图能够支持修改。

INSTEAD OF 触发器被用于更新那些没有办法通过正常方式更新的视图。例如，通常不能在一个基于连接的视图上进行删除操作。然而，可以编写一个 INSTEAD OF Delete 触发器来实现删除。上述触发器可以访问那些如果视图是一个真正的表时已经被删除的数据行，将被删除的行存储在一个名为 deleted 的工作表中，就像 AFTER 触发器一样。相似地，在 Update INSTEAD OF 触发器或者 Insert INSTEAD OF 触发器中，你可以访问 inserted 表中的新行。

不能在带有 WITH CHECK OPTION 定义的视图中创建 INSTEAD OF 触发器。

【例 6.14】求创建一个德国客户表和一个墨西哥客户表。设计各自视图与放置在视图上的 INSTEAD OF 触发器，把更新操作重新定向到相应的基表上。

（1）创建两个包含客户数据的表：

```
select * INTO CustomersGer FROM Customers Where Customers.Country = 'Germany'
select * INTO CustomersMex FROM Customers Where Customers.Country = 'Mexico'
GO
```

（2）在该数据上创建视图：

```
Create VIEW CustomersView AS Select * FROM CustomersGer
UNION
Select * FROM CustomersMex
GO
```

（3）创建一个在上述视图上的 INSTEAD OF 触发器：

```
Create TRIGGER Customers_Update2
ON CustomersView
INSTEAD OF Update AS
DECLARE @Country nvarchar(15)
SET @Country = (Select Country FROM Inserted)
IF @Country = 'Germany'
BEGIN
Update CustomersGer
SET CustomersGer.Phone = Inserted.Phone
FROM CustomersGer JOIN Inserted
ON CustomersGer.CustomerID = Inserted.CustomerID
END
ELSE
```

```
IF @Country = 'Mexico'
BEGIN
Update CustomersMex
SET CustomersMex.Phone = Inserted.Phone
FROM CustomersMex JOIN Inserted
ON CustomersMex.CustomerID = Inserted.CustomerID
END
```

（4）通过更新视图，测试触发器：

```
Update CustomersView SET Phone = ' 030-007xxxx'
Where CustomerID = 'ALFKI'
Select CustomerID, Phone FROM CustomersView
Where CustomerID = 'ALFKI'
Select CustomerID, Phone FROM CustomersGer
Where CustomerID = 'ALFKI'
```

可以发现，这时发生的插入是对 CustomersGer 表的插入而不是对视图的插入。

6.4.6 触发器的应用

以上部分我们讨论了触发器的优缺点、工作原理以及创建触发器的具体方法。下面我们阐述各种不同触发器的应用。

1. INSERT 触发器

在触发器中如果带有 FOR INSERT 子句，属于 INSERT 类触发器。INSERT 触发器在向数据库插入数据之后触发，执行触发器所定义的操作，可以对新插入数据进行检查，拒绝某些数据的录入。在插入记录的同时，还将生成 inserted 表，记录副本将插入该表。

【例 6.15】假设在数据库 waremanage 中有表 commodity，求建立触发器 commodityinsert，检查新插入的每条记录，删除其中 unitprice>2000 的记录。

```
CREATE TRIGGER commodityinsert
ON commodity
FOR INSERT
AS
   DELETE FROM commodity WHERE unitprice>2000
```

2. DELETE 触发器的应用

在触发器中如果带有 FOR DELETE 子句，属于 DELETE 类触发器。它在数据库删除数据之后触发，执行触发器所定义的操作，可以对被删除数据的相关数据进行检查并执行同步的操作。

【例 6.16】对定义了删除型触发器的 commodity 表进行删除操作，首先检查要删除几行，如果将删除多行则返回错误信息。

```
CREATE TRIGGER commoditydelete
    ON commodity FOR DELETE
AS
IF @@rowcount = 0
    RETURN
IF @@rowcount > 1
BEGIN
    ROLLBACK TRANSACTION
```

```
        RAISERROR("You Can Only Delete Information At One Time", 16, 1)
    END
    RETURN
```

3. UPDATE 触发器

在触发器中如果带有 FOR UPDATE 子句，属于更新类触发器。它在修改数据之后触发，可以对被修改的数据进行检查。

【例 6.17】求建立触发器 commodityupdate，检查所修改的数据，如果改后 unitprice 比原来低，则恢复原来数据。

由于更新操作包括两部分：先将需更新的内容从表中删除掉，然后插入新值。因此，更新型触发器同时涉及到了删除表，将产生删除表 deteted，其中将保存删除前的数据值。可以根据更新后表中数据与 deleted 中保存的数据进行比较，判别改后 unitprice 是否比原来低，并设法恢复原来数值。要注意，判别时根据表中的关键字（本例中是商品名称 WareName）建立联系。

```
CREATE TRIGGER commodityupdate
    ON commodity FOR UPDATE
AS
    UPDATE commodity SET commodity. unitprice=deleted.unitprice
        FROM commodity,deleted
        WHERE commodity.WareName =deleted.WareName AND commodity. Unitprice
<deleted.unitprice
```

4. 嵌套触发器

当某一触发器执行时同时触发另外一个触发器称之为触发器嵌套，例如，在执行某修改数据的过程中，让一个触发器修改某个已经有~~其它~~其他触发器的表时就可能会有触发器嵌套问题。如果不需要嵌套触发器，可以通过 sp_configure 选项来进行设置。在 SQL Server 中触发器能够嵌套至 32 层。

可使用嵌套触发器执行如保存前一触发器所影响记录的一个备份等这一类工作。

【例 6.18】在 commodity 上另建一个触发器 commodityupdate1，保存由 commodityupdate 触发器所删除的 commodity 的记录的备份。将被删除的数据保存到另一个单独创建的名为 del_save 表中。

```
CREATE TRIGGER commodityupdate1
    ON commodity FOR UPDATE
AS
INSERT del_save SELECT * FROM deleted
```

5. 递归触发器

在触发器的应用中，常会遇到这种情况，即被触发的触发器试图更新与其相关联的原始的目标表，从而使触发器被无限循环地触发。对于该种情况，不同的数据库产品提供了不同的解决方案。有些 DBMS 对一个触发器的执行过程采取的动作强加了限制，有些 DBMS 提供了内嵌功能，允许一个触发器主体触发内嵌的触发器；另一些 DBMS 提供了一种系统设置，控制是否允许串联的触发器处理；还有一些 DBMS 对可能触发的嵌套触发器级别的数目进行限制。在 SQL Server 中，这种能触发自身的触发器被称为递归触发器。对它的控制是通过限制可能触发的嵌套触发器级别的数目，另外，通过是否允许触发嵌套触发器也能实现

对它的控制。

触发器不会以递归方式自行调用，除非设置了 RECURSIVE_TRIGGERS 数据库选项。有两种不同的递归方式：

（1）直接递归。触发器激发并执行一个操作，而该操作又使同一个触发器再次激发。例如，一应用程序更新了表 T3，从而引发触发器 Trig3。Trig3 再次更新表 T3，使触发器 Trig3 再次被引发。

（2）间接递归。触发器激发并执行一个操作，而该操作又使另一个表中的某个触发器激发。第二个触发器使原始表得到更新，从而再次引发第一个触发器。例如，一应用程序更新了表 T1，并引发触发器 Trig1。Trig1 更新表 T2，从而使触发器 Trig2 被引发。Trig2 转而更新表 T1，从而使 Trig1 再次被引发。

当将 RECURSIVE_TRIGGERS 数据库选项设置为 OFF 时，仅防止直接递归。若要禁用间接递归，需将 nested triggers 服务器选项设置为 0。

6.4.7 管理触发器

如果要显示作用于表上的触发器究竟对表有哪些操作，必须查看触发器信息。在 SQL Server 中，有多种方法查看触发器信息。在本节中将介绍两种常用的方法：①通过 Enterprise Manager；②通过系统存储过程 sp_help、sp_helptext 和 sp_depends。

1. 使用 Enterprise Manager 显示触发器信息

使用 Enterprise Manager 显示触发器信息的操作步骤如下：

（1）运行 Enterprise Manager，登录到指定的服务器。

（2）选择数据库和表，如图 6.39 所示。

（3）从“操作”菜单中选择“所有任务”，再选择“管理触发器”，如图 6.40 所示打开“触发器属性”对话框。

2. 使用系统存储过程查看触发器

系统存储过程 sp_help、sp_helptext 和 sp_depends 分别提供有关触发器的不同信息。

（1）sp_help。使用 sp_help 系统过程的命令格式是：

sp_help '<触发器名字>'

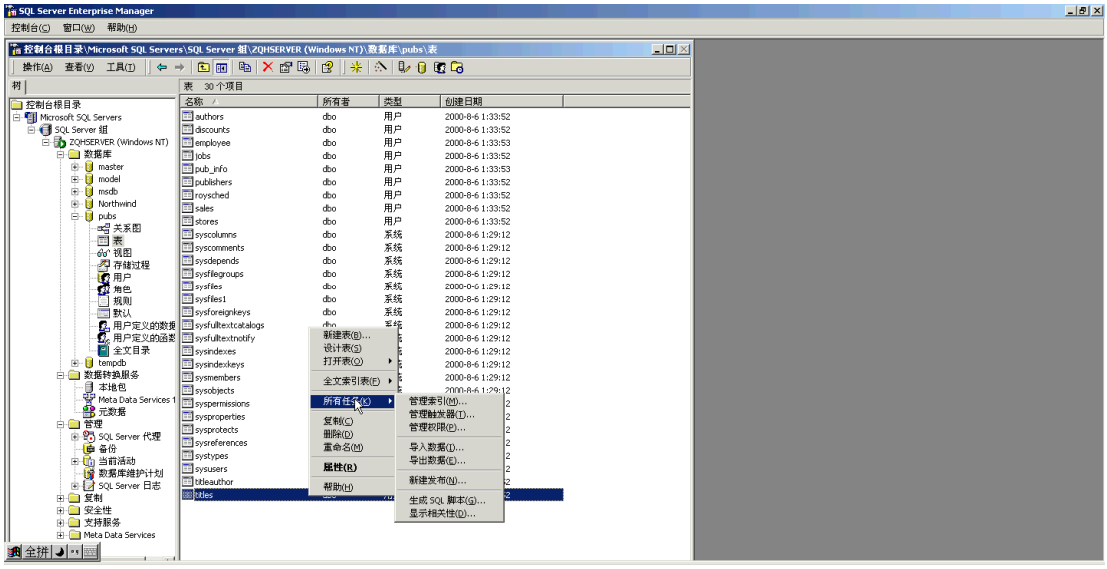

图 6.39　利用 Enterprise Manager 显示触发器信息

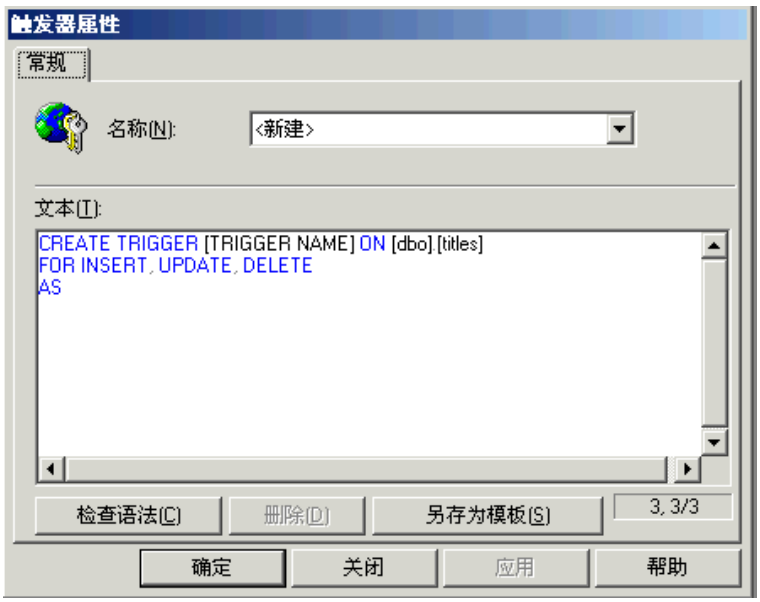

图 6.40　“触发器属性”对话框

通过该系统过程，可以了解触发器的一般信息，如触发器的名字、属性、类型、创建时间等。

示例：执行命令 sp_help reptq1，输出结果为：

```
Name      Owner     Type              Created_datetime
-------------------------------------------------------------------------------
reptq1    dbo       stored procedure  2000-08-06 01:33:58.763
```

（2）sp_helptext。通过 sp_helptext 能够查看触发器的正文信息，其语法格式为：

sp_helptext '<触发器名>'

示例：执行命令 sp_helptext reptq1，输出结果为：

```
CREATE PROCEDURE reptq1 AS
select pub_id, title_id, price, pubdate
from titles
where price is NOT NULL
order by pub_id
COMPUTE avg(price) BY pub_id
```

（3）sp_depends。通过 sp_depends 能够查看指定触发器所引用的表或指定的表涉及到的所有触发器，其语法格式如下：

sp_depends '<触发器名字>'

sp_depends '<表名>'

注意：用户必须在当前数据库中查看触发器的信息，而且被查看的触发器必须已经被创建。

3. 修改删除触发器

通过 Enterprise Manager 和系统过程或 Transact_SQL 命令，可以修改触发器的名字和正文。

（1）使用 sp_rename 命令修改触发器的名字。

其语法格式为：

sp_rename <原名>,<新名>

（2）通过 Enterprise Manager 修改触发器正文。通过 Enterprise Manager 修改触发器正文的操作步骤与查看触发器信息一样。修改完触发器后要使用 Check Syntax 选项对语句进行检查。

（3）删除触发器。用户在使用完触发器后可以将其删除，只有触发器属主才有权删除触发器。可以用系统命令 DROP TRIGGER 删除指定的触发器。

语句格式：

DROP TRIGGER <触发器名字>

删除触发器所在的表时，SQL Server 将自动删除与该表相关的触发器。

*6.5 SQL Server 中的数据导入和导出

当我们建立一个数据库，且想将分散在各处的不同类型的数据库分类汇总在这个新建的数据库中、或者想进行数据检验、净化和转换时，需要有从其他数据库采集数据转录到新数据库中的功能，称为导入。将现有数据库中的数据以其他数据库或应用程序能接受的形式输出出来，称为导出。导入、导出是在不同系统间建立联系并实现数据更大规模共享与建立更大规模应用系统的十分重要的功能。SQL Server 为我们提供了强大、丰富的数据导入、导出功能，在导入、导出的同时还可以对数据进行灵活的处理。

有多种方式导入、导出数据：①使用 Transact-SQL 对数据进行处理；②使用数据转换服务（DTS）对数据进行处理。还有一些其他方法，这些方法各有其特点，下面介绍 Transact-SQL 方法与 DTS 方法。

6.5.1　使用 Transact-SQL 进行数据导入、导出

Transact-SQL 方法就是通过 SQL 语句方式将相同或不同类型的数据库中的数据导入、导出的方法。如果是在不同的 SQL Server 数据库之间进行数据导入、导出，可使用 SELECT INTO FROM 和 INSERT INTO 语句，前者将查询结果生成 SQL Server 数据库中另外一个新表，实现导出；后者实现导入。

SELECT INTO FROM 语句格式为：SELECT * INTO table2 FROM table1

由该语句生成的新表 table2 的结构和源数据表 table1 的结构相同。

INSERT INTO 语句格式为：INSERT INTO table2 SELECT * FROM table3

要求基于源数据表 table3 的查询语句输出格式和目的数据表 table2 的结构相同。该语句将查询结果添加到 table2 原来数据的后面，可以实现两表数据的合并。

当在异构数据库之间进行数据导入、导出时，首先要解决的是如何打开非 SQL Server 数据库的问题。

在 SQL Server 中提供了两个函数可以根据各种类型数据库的 OLE DB Provider 打开并操作这些数据库，这两个函数是 OPENDATASOURCE 和 OPENROWSET。

1. 使用 OPENDATASOURCE

OPENDATASOURCE 的参数有两个，分别是 OLE DB Provider 和连接字符串。使用 OPENDATASOURCE 相当于引用数据库或者是服务（对于 SQL Server、Oracle 等数据库来说）。要想引用其中的数据表或视图，必须在 OPENDATASOURCE(...)后进行引用。

例如在 SQL Server 中通过 OPENDATASOURCE 查询 Access 数据库 abc.mdb 中的 table1 表中的数据：

```
SELECT * FROM OPENDATASOURCE('Microsoft.Jet.OLEDB.4.0',
'Provider=Microsoft.Jet.OLEDB.4.0;
Data Source=abc.mdb;
Persist Security Info=False')...table1
```

这段程序要求在 C:\WINNT\SYSTEM32\中存在数据库文件 abc.mdb，且其中存在数据表 table1。

2. 使用 OPENROWSET

函数 OPENROWSET 相当于一个记录集，可以将它直接当成一个表或视图使用。

例如在 SQL Server 中通过 OPENROWSET 查询 Access 数据库 abc.mdb 中的 table1 表：

```
SELECT * FROM OPENROWSET('Microsoft.Jet.OLEDB.4.0',
 'abc.mdb';'admin';'','SELECT * FROM table1')
```

OPENDATASOURCE 只能打开相应数据库中的表或视图，如果需要过滤的话，只能在 SQL Server 中进行处理。而 OPENROWSET 可以在打开数据库的同时对其进行过滤，如上面的例子，在 OPENROWSET 中可以使用 SELECT * FROM table1 对 abc.mdb 中的数据表进行查询，而 OPENDATASOURCE 只能引用 table1，而无法查询 table1。因此，OPENROWSET 比较 OPENDATASOURCE 更加灵活。

6.5.2　使用命令行 bcp 导入、导出数据

很多大型的系统不仅仅提供了友好的图形用户界面，同时也提供了命令行方式对系统进

行控制。在 SQL Server 中除了可以使用 SQL 语句对数据进行操作外，还可以使用一个命令行工具 bcp 对数据进行同样的操作。

bcp 是基于 DB-Library 客户端库的工具。它的功能十分强大，能够以并行方式将数据从多个客户端大量复制到单个表中，从而大大提高了装载效率。但在执行并行操作时要注意的是只有使用基于 ODBC 或 SQL OLE DB 的 API 的应用程序才可以执行将数据并行装载到单个表中的操作。

bcp 可以将 SQL Server 中的数据导出到任何 OLE DB 所支持的数据库，例如下面的语句是将 authors 表导出到 excel 文件中：

```
bcp pubs.dbo.authors out c:\temp1.xls -c -q -S"GNETDATA/GNETDATA" -U"sa" -P"password"
```

bcp 不仅能够通过命令行执行，同时也可以通过 SQL 执行，这需要一个系统存储过程 xp_cmdshell 来实现，如上面的命令可改写为如下形式：

```
EXEC master xp_cmdshell 'bcp pubs.dbo.authors out c:\temp1.xls -c -q -S"GNETDATA/GNETDATA" -U"sa" -P"password"'
```

6.5.3 使用数据转换服务（DTS）导入、导出数据

DTS 是 SQL Server 中导入、导出数据的核心，它除了具有 SQL 和命令行工具 bcp 相应的功能外，还可以灵活地通过 VBScript、JScript 等脚本语言对数据进行检验、净化和转换。

SQL Server 为 DTS 提供了图形用户界面，用户可以使用图形界面将 SQL Server 数据库数据导出到 Excel 工作表、文本文件、~~其它~~其他 ODBC 数据源、Visual FoxPro 数据库、Access 数据库中去。

导出操作方法是：右击“数据库”→所有任务→导出数据→单击“刷新”按钮并选择欲导出的数据库，选择服务器与身份验证方式→选择导出目标源的驱动程序（Excel 工作表、文本文件、~~其它~~其他 ODBC 数据源、Visual FoxPro 数据库、Access 数据库等）→输入导出的文件名（包括路径），如果是输出到其他数据库，还要根据需要输入系统名称、用户名称、密码等→导出细节：是导出表或视图还是根据查询语句的输出导出→具体选择表或写出查询语句之类的语句→可以利用转换按钮对列进行转换、选择是否重建同名数据表→如果需要，保存 DTS 包，完成导出操作。

【例 6.19】将数据库 Waremanage 中的表 Commodity 的内容导出到 Excel 表中。

利用企业管理器操作如下：

（1）右击“数据库”文件夹，选择“所有任务”中的“导出数据”。

（2）在“DTS 导入/导出向导”对话框中单击“下一步”，在数据源下拉列表框中选择“用于 SQL Server 的 Microsoft OLE DB 提供程序”，在“数据库”下拉列表框中选择“Waremanage”。

（3）单击“下一步”按钮，在“目的”下拉列表框中选择“Microsoft Excel 97-2000”，在“文件名”栏中输入 Excel 文件名，例如：c: Commodity.xls。

（4）单击“下一步”按钮，选择“从源数据库复制表和视图”。

（5）单击“下一步”按钮，在“源”列选中 Commodity；在“目的”列中输入 Commodity。

（6）单击“下一步”按钮，出现“保存、调度和复制包”对话框。单击“下一步”按钮，直到最终完成。

用户也可以使用图形界面导入数据，并对数据进行相应的处理。导入操作方法是：右击

“数据库”→所有任务→导入数据→选择数据源驱动程序，输入导入数据的文件名称→单击“刷新”按钮，选择导入数据源，选择数据库→指定导入的是表或视图或复制查询结果→具体选择表或视图→如果需要，保存 DTS 包，完成导入操作。

另外，DTS 还以 com 组件的形式提供编程接口，也就是说任何支持 com 组件的开发工具都可以利用 com 组件使用 DTS 所提供的功能。DTS 在 SQL Server 中可以保存为不同的形式，可以是包的形式，也可以保存成一些语言的源程序文件，这样只要在该语言环境中编译便可以使用 DTS com 组件了。

上述的三种数据导入、导出方法各有其利弊，它们之间的相互比较如表 6.3 如示。

表 6.3　三种数据导入、导出方法的比较

	Transact-SQL	命令行工具 bcp	DTS
图形管理界面	无	无	有
操作难易程度	容易	较复杂	复杂
编程接口	SQL	系统存储过程或通过 shell 调用	com 组件
数据处理	只能对数据进行简单过滤	只能对数据进行简单过滤	可以利用脚本对数据进行复杂处理

如果是 SQL Server 数据库之间的导入、导出，使用 Transact-SQL 方式时速度将非常快，但是使用 OPENDATASOURCE 和 OPENROWSET 方法利用 OLE DB Provider 打开并操作数据库时速度会慢一些。

如果不需要对数据进行验证等操作的话，使用 bcp 命令方式还是非常快的，这是因为它的内部使用 C 接口的 DB-library，使其在操作数据库时速度有很大的提升。

由于 DTS 方式整合了 Microsoft Universal Data Access 技术与 Microsoft ActiveX 技术，因此不仅可以灵活地处理数据，而且在数据导入、导出时的效率非常高。因此，它是最受推荐的方式。

6.5.4　如何选择具体的数据导入、导出方法

SQL Server 提供了丰富的数据导入、导出方法，给我们提供了更多的选择，但是又会给我们带来一个问题：如何根据具体情况选择合适的数据导入、导出方法呢？以下提出一些建议。

如果是在 SQL Server 数据库之间进行数据导入、导出，并且不需要对数据进行复杂的检验，最好使用 Transact-SQL 方法进行处理，因为在 SQL Server 数据库之间进行数据操作时，SQL 是非常快的。如果要进行复杂的操作，如数据检验、转换等操作时，最好还是使用 DTS 进行处理，因为 DTS 不光导入、导出数据效率高，而且能够对数据进行深度控制。但是 DTS 的编程接口是基于 com 的，这个接口十分复杂，因此，使用程序调用 DTS 也会变得很复杂，当数据量不是很大，并且想将数据导入、导出功能加入到程序中，而且没有复杂的数据处理功能时，可以使用 OPENDATASOURCE 或 OPENROWSET 进行处理。

bcp 命令并不太适合通过程序来调用，如果需要使用批量的方式导入、导出数据，可以通过批处理文件调用 bcp 命令，这样做既不需要编写大量的程序，也无需在企业管理器中通过各

种操作界面的切换来进行数据导入、导出。它比较适合在客户端未安装企业管理器或使用 SQL Server Express 时对数据进行快速导入、导出的场合。

*6.6 SQL Server 应用系统开发环境

6.6.1 SQL Server 应用系统的两种系统结构

SQL Server 应用系统一般要利用其他的开发语言建立，系统结构大体有 Client/Server 即客户机/服务器模式（C/S）与 Browse/Server 即浏览器/服务器模式（B/S）两种。管理信息系统最早采用集中管理和使用模式，20 世纪 90 年代后多基于 C/S 模式开发，目前普遍转向 B/S 模式。但是，事物总是螺旋式上升的，未来必将向 C/S 模式结合 B/S 模式开发方向发展。

B/S 模式用户工作界面是通过 WWW 浏览器来实现的，极少一部分事务逻辑在前端（Browser）实现，主要事务逻辑都在服务器端（Server）实现，形成所谓三层结构。这样就大大简化了客户端电脑载荷，减轻了系统维护与升级的成本和工作量，降低了用户的总体成本（TCO）。以目前的技术看，局域网建立B/S结构的网络应用，并基于 Internet/Intranet模式开发数据库应用，相对易于把握、成本也较低。它是一次性到位的开发，能实现不同的人员，从不同的地点，以不同的接入方式（比如 LAN、WAN、Internet/Intranet 等）访问和操作共同的数据库；它能有效地保护数据平台和管理访问权限，服务器数据库也很安全。目前用得比较普遍的 B/S 模式系统开发语言有 JSP、ASP（ASP.NET）、PHP、HTML 等。

C/S 模式将多机共享数据集中保存在一个中央计算机中，用户可在本地机中建立自己的客户端软件及客户端数据库系统。这个数据库系统包括对共享数据的复制品，包括以视图形式提供的对远程数据操作的全局数据库模式的子集，用户通过本地的客户端软件通过网络访问位于服务器上的数据库，对数据进行处理，同时还要对最终的输出进行控制。数据处理在客户机与服务器双方进行，客户端应用程序建立对远程数据的连接，在本地建立虚表（以远程视图形式），也可建立部分实表，查询并从远程取出数据传送到客户方，在客户机中处理完毕再写回并修改远程服务器中数据。这种方式客户机分担了程序服务器的部分工作，减轻了远程服务器的压力，但网络通信量较大，客户端完成的功能较为复杂。数据库服务器负责数据的存储及管理；客户机向服务器请求数据服务，再做必要的处理，并以图形界面呈现数据，用户在客户端进行录入、修改、删除、查询、计算、打印等操作。在 C/S 结构中，数据库服务器应能发挥积极主动的作用，例如在查询时，当客户端将查询指令透过网络传送至数据库服务器时，后者并不把全表数据传至客户端，而是先行对数据进行过滤查询处理，再将查询结果传到客户端，因而降低了网络的负荷。目前用得比较普遍的 C/S 模式系统开发语言有 C++、Java、PB、VB 等。

从以上描述可以看到，客户端应用程序需要透过网络与服务端联系，其间网络连接十分重要。为了满足各种不同的需要，目前有多种连接客户和服务器的标准接口和软件。

6.6.2 ODBC

ODBC（Open Database Connectivity）是微软定义的一种开放式的数据库连接技术。为异种数据库的访问提供了统一的接口。它基于 SQL(Structured Query Language)，并把它作为访问数据库的标准，它为应用程序提供了一套数据库调用接口函数和基于动态链接库的运行支持环

境，使开发数据库应用程序时，可以使用标准的 ODBC 函数和 SQL 语句，提供了最大限度的相互可操作性:一个应用程序可以通过一组通用的代码访问不同的数据库管理系统，可以为不同的数据库提供相应的驱动程序，提供统一接口，使得应用程序具有良好的适应性与可移植性，是一种公认的关系数据源的接口界面。

ODBC 体系结构分为应用程序、驱动管理程序、驱动程序与数据源四层。

ODBC 应用程序不能直接存取数据库，它必须通过 Windows 环境下的一个应用程序“ODBC Drive Manager”调用相应的 ODBC 驱动程序，由 ODBC 驱动程序实现对数据源的操作。操作结果也要通过 ODBC 驱动程序返回给应用程序。ODBC 驱动程序是由所有支持 ODBC 的数据库各自配置的，不同数据库的 ODBC 驱动程序不相同，它们连接各自数据库系统中代表目标数据库的“数据源”，再操纵目标数据库。由于以上的机制，就使得各种不同的语言能通过 ODBC 访问不同的数据库。

ODBC 应用程序的任务包括：

（1）连接数据源。

（2）向数据源发送 SQL 语句。

（3）为 SQL 语句执行结果分配存储空间。

（4）读取结果。

（5）提交处理结果。

（6）请求提交和回滚事务。

（7）处理出错。

（8）在处理完毕后断开与数据源的连接。

应用 ODBC 开发应用系统首先要建立数据源，之后建立程序与数据源的联系。

1. 建立数据源

为了使用的方便，并不要求用户知道 ODBC 驱动程序及数据库的细节，在使用 ODBC 之前，要求先创建一个“数据源（Data Source）”，将它作为用户访问数据库的桥梁。

以 Windows 2000 为例，创建数据源的步骤为：

（1）打开控制面板，选择“管理工具”，双击“数据源（ODBC）”，将可见到 ODBC 数据源管理器，如图 6.41 所示。

图 6.41　ODBC 数据源管理器

ODBC 数据源分为三种：用户数据源、系统数据源和文件数据源。选择“用户 DSN”选项卡（见图 6.41）。单击“添加”按钮，新建数据源。

（2）选择 ODBC 驱动程序，选择“SQL Server”，单击“完成”按钮（见图 6.42）。

图 6.42　添加用户数据源

（3）在“建立新的数据源到 SQL Server”对话框中，输入将来引用该数据源的名字，选择需要连接的服务器 CXX-D66C7918B7A（见图 6.43）。

图 6.43　选择服务器

（4）再选择需要连接的数据库 waremanage（见图 6.44）。

图 6.44　选择数据库

（5）单击“完成”按钮（见图 6.45）后就可以“确定”新的数据源建立了。如果按以上定义的情况，将来应用程序就可以调用“sqlserver1”连接到服务器 CXX-D66C7918B7A，访问其中数据库 waremanage 中的数据了。

2. 基于数据源访问数据库的程序设计

一般程序访问数据库的步骤至少有如下三步：

（1）产生连接句柄（连接字串）。

（2）建立对数据源的连接。

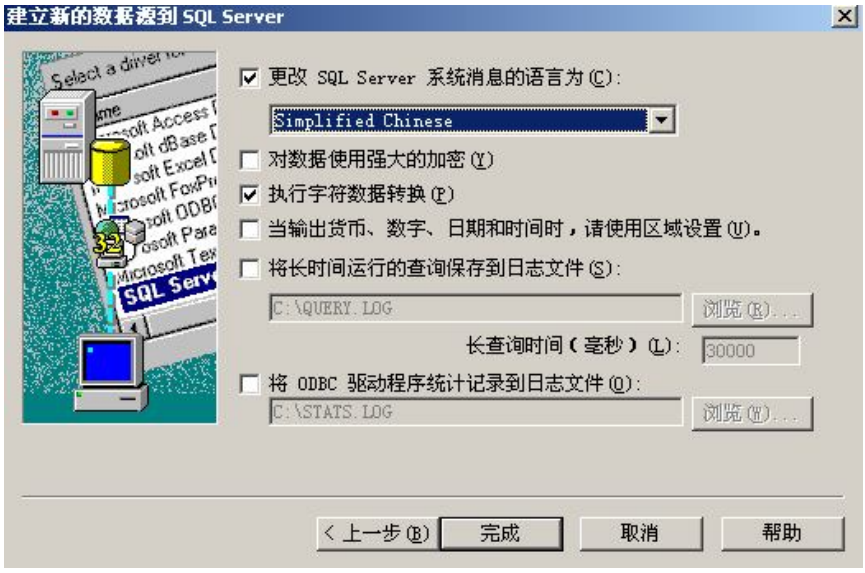

图 6.45　建立数据源成功完成

（3）远程执行 SQL 语句，对服务器中数据库进行操作。

具体实现的语句很多，以下举几个例子说明。

在 VB 中，用 ODBC API 编程，表 6.4 列出了常用的函数以及它们的功能。

表 6.4 常用函数

函数	功能
SQLALLocEnv	初始化 ODBC 环境，返回环境句柄
SQLALLocConnect	为连接句柄分配内存并返回连接句柄
SQLConnect	连接一个 SQL 数据源
SQLDriverConnect	连接一个 SQL 数据源，允许驱动器向用户询问信息
SQLALLocStmt	为语句句柄分配内存并返回语句句柄
SQLExecDirect	把 SQL 语句送到服务器
SQLFetchAdvances	到结果集的下一行（或第一行）
SQLGetData	从结果集的特定的一列取回数据
SQLFreeStmt	释放与语句句柄相关的资源
SQLDisconnect	切断连接
SQLFreeConnect	释放与连接句柄相关的资源
SQLFreeEnv	释放与环境句柄相关的资源

下面的代码使用上函数先登录到一个服务器数据库，并为随后的工作设置了语句句柄。

（1）应用程序调用 SQLAllocEnv 函数初始化 ODBC 接口，获取 ODBC 环境句柄。ODBC 环境句柄是~~其它~~其他所有 ODBC 资源句柄的父句柄。接着，与 ODBC 数据源建立连接，由下列两个步骤组成：①调用 SQLAllocConnect 函数获取连接句柄；②调用 SQLConnect 函数建立连接。之后对数据库操作，包括：①调用 SQLAllocStmt 函数获取语句句柄；②使用 SQLExecDirect 执行 SQL 语句。例如：

```
Dim rc As Integer                        'ODBC 函数的返回码
Dim henv As Long                         'ODBC 环境句柄
rc = SQLAllocEnv(henv)                   '获取 ODBC 环境句柄
Dim hdbc As Long                         '连接句柄
rc = SQLAllocConnect(henv, hdbc)         '获取连接句柄
Dim DSN As String, UID As String, PWD As String
DSN = " sqlserver1"                      'ODBC 数据源名称
UID = "dbo"                              '用户账号
PWD = ""                                 '用户口令
rc = SQLConnect(hdbc, DSN, Len(DSN), UID,Len(UID), PWD, Len(PWD))  '建立连接
Dim hstmt As Long
rc = SQLAllocStmt(hdbc, hstmt)           '获取语句句柄
Dim SQLstmt As String
rc=SQLExecDirect(hstmt,"SELECT * FROM commodity ", Len(SQLstmt)) '执行 SQL 语句
```

（2）在 VFP 中，如果建立了数据源 sqlserver1，可以按如下步骤访问 SQL Server 的数据库：①在 VFP 中需要预先建立数据库，例如“数据库 1”；②建立对数据库的连接，为连接定义一个名字；③依据数据源名、属主名、上一步定义的连接名生成连接句柄；④依赖连接句柄

或连接执行对远程数据库的操作，例如执行 SQL 语句、建立远程视图等；⑤对上面的操作返回数据进行处理；⑥释放连接（句柄）。

实现语句如下：

```
CREATE CONNECTION "conne" DATASOURCE sqlserver1 &&建立对数据源的连接
hstmt=SQLCONNECT("sqlserver1","dbo","conne")      &&产生连接句柄，dbo 为属主名
SQLEXEC(hstmt,"select * from commodity","as1") &&根据连接句柄执行对远程的 SQL 语句，将语句结果输入到临时表 AS1 中。以下可以选择 AS1 再操作：
SELECT * FROM AS1
```

（3）在 VFP 中，还可以使用如下语句访问 SQL Server 的数据库、建立远程视图并录入数据。

```
CREATE CONNECTION "conne" DATASOURCE sqlserver1   &&建立对数据源的连接
CREATE SQL VIEW VIEW1 REMOTE CONNECTION conne AS SELECT * FROM commodity          &&基于连接建立对 commodity 表的远程视图
INSERT INTO view1 VALUES ("塑料板","100*200*6",50," ")          &&录入一组数据
=CURSORSETPROP("WhereType",3,"view1")    &&更新主键和所有修改了的字段内容
=CURSORSETPROP("FetchMemo",.T.,"view1")          &&可以更新备注字段
=CURSORSETPROP("SENDUPDATES",.T.,"view1")       &&执行 SQL 更新查询以更新远程表
=TABLEUPDATE(.t.,.t.,"view1")             &&确定是否实施所有对表或临时表的更改
```

6.6.3　ADO

ADO（ActiveX Data Object）是一种对象模型，是最新的数据库访问模式，其特点是简单易用，已经成为当前数据库开发的主流。

ADO 涉及的数据存储有 DSN（数据源名称）、ODBC（开放式数据连接）以及 OLE DB 三种方式。

ADO 的对象层次结构大体上分为以下 7 层：

（1） Connection 对象：是最基本的对象，提供与数据库的连接。它指定 OLE DB 提供者、连接到数据存储的安全细节和~~其它~~其他涉及数据存储的细节。

（2）Command 对象：包含关于某个命令，例如查询字符串、参数定义等的信息，是对数据存储执行命令的对象，其主要作用是向 SQL 语句、存储过程传递参数，实现对数据库的操作。

（3）Error 对象：包含数据提供程序出错时的扩展信息。

（4）Recordset 对象：用来存储数据操作返回的记录集。此对象和 Connection 对象是所有对象中使用最普遍的两个对象。

（5）Field 对象：包含 Recordset 对象中数据的某单个列的信息。

（6）Parameter 对象：包含参数化的 Command 对象的某个参数的信息。该 Command 对象有一个包含其所有 Parameter 对象的 Parameters 集合。Parameter 对象可以提供输入、输出参数。

（7）Property 对象：包含属性、名字、类型及值的信息，为某个 ADO 对象提供程序定义的特征。

使用 ADO 访问 SQL Server 数据库的方法：

（1）可以借助 ODBC 程序，ADO 设计为一种极简单的格式，通过 ODBC 的方法同数据库接口。可以使用任何一种 ODBC 数据源，不仅适合于 SQL Server、Oracle、Access 等数据库应用程序，也适合于 Excel 表格、文本文件、图形文件和无格式的数据文件，是一个便于使用的应用程序层接口。

（2）借助 SQL Server 的 OLE DB 提供者。ADO 是为 Microsoft 最新和最强大的数据访问范例 OLE DB 而设计的，OLE DB 为任何数据源提供了高性能的访问，这些数据源包括关系和非关系数据库、电子邮件和文件系统、文本和图形、自定义业务对象等。ADO 在关键的 Internet 方案中使用最少的网络流量，并且在前端和数据源之间使用最少的层数，所有这些都是为了提供轻量、高性能的接口。

6.6.4 JDBC

JDBC 是 Sun 公司制定的 Java 数据库连接（Java DataBase Connectivity）技术。由一组用 Java 语言编写的类和接口组成，是从 Java 应用程序连接 DBMS 的标准方式。JDBC 既是 Java 编程人员的 API，也是实现数据库连接的服务提供者的接口模型。作为 API，JDBC 提供 Java 应用程序与各种数据库交互的标准接口；作为服务提供者的接口模型，JDBC 提供了数据库厂家和第三方中间件厂家实现数据库交互的标准接口方式，JDBC 利用现有的 SQL 标准，可以和 ODBC 之类~~其它~~的数据库连接标准相互桥接。其使用接口简单，支持高性能实现，它在 Web 和 Internet 应用程序中的作用和 ODBC 在 Windows 系列平台中的作用类似。

由于 ODBC 使用 C 语言接口，不适于直接在 Java 中使用。Java 要使用 ODBC，需要在 JDBC 的帮助下以 JDBC-ODBC 桥的形式使用。先要按前面介绍的方法建立数据源，再使用 JDBC。

JDBC 的任务是完成同一个数据库的连接；向数据库发送 SQL 语句；返回数据库处理结果。应用程序通过 JDBC 驱动管理器驱动 JDBC-ODBC 桥或者驱动供应商提供的 JDBC 驱动程序，实现对数据库的访问。为使用 JDBC，Java 提供了一系列重要的类：

（1）DriverManager：加载驱动程序，管理应用程序和驱动程序的连接。

（2）Connection：应用程序和数据库之间的连接。

（3）Driver：将 API 的调用映射到对数据库的操作。

（4）Statement：执行查询和更新。

（5）Metadata：返回有关数据、数据库和驱动程序的信息。

（6）ResultSet：执行查询结果后的结果集。

如果使用 JBuilder 开发，首先查看 JBuilder 开发平台可以使用的数据库驱动程序的情况。打开 JBuilder 10，单击 Tools 中的 Database Pilot，单击 View，选择最后一项——options，单击 File→New 菜单，选择 NEW URL，查看一下，在对话框的 Driver 列表中，列出了许多数据库连接驱动，其中红色的表示这一连接驱动目前还不能使用，黑色表示可以使用。如果发现 com.microsoft.jdbc.sqlserver.SQLServerDriver 为红色，还不能使用，表示需要安装 JDBC 驱动程序。JDBC 驱动程序包括 4 种类型：

（1）将 JDBC 转化为 ODBC 驱动，利用 JDBC-ODBC 桥和 ODBC 驱动访问数据库。

（2）将用户的调用转化为对客户端相应的 API 调用，要求数据库在本地安装数据库客户端。

（3）驱动程序和一个中间层通信，由中间层实现对数据库的访问，这一方式比较适合于

异构数据库应用系统。

（4）驱动程序直接将用户请求转化为对数据库的协议请求，直接和数据库通信。

Java.sql.DriverManager 类是 JDBC API 用户直接装入与管理 JDBC 驱动器的主要接口。DriverManage 可以通过设置系统属性 jdbc.drivers 隐式装入驱动器类，也可以用 class.forName()方法显示装入驱动器类。DriverManager 还可以生成新连接，与所装入驱动器的池相关联。

要隐式装入驱动类，只要设置系统属性 jdbc.drivers，指定驱动器类名。例如按 java.util.Properies 对象可读格式在文件中所提供的一组 JDBC 驱动器名，经选择后装入，对每个驱动器调用 Class.forName()方法。有些驱动器的构造函数用于驱动器注册，要在代码中硬编码驱动器名。

例如，安装与数据库 SQL Server 2000 相连的驱动程序 Microsoft SQL Server 2000 Driver for JDBC 之后，查看目录中是否有 msbase.jar、mssqlserver.jar 和 msutil.jar 三个文件，复制这三个文件到 JBuilder 10 的 lib 根下。在 JBuilder 中，单击 Tools→configure Libraries 菜单项，弹出 Library setting 对话框，单击 New 按钮，弹出 New Library Wizard（新建连接数据库向导）。在 Name 中输入 com.microsoft.jdbc.sqlserver.SQLServerDriver，在 Location 下拉列表框中选择 Jbuilder，再单击 Add，在弹出的 Select one or more libraries 对话框中，选择新增加的 com.microsoft.jdbc.sqlserver.SQLServerDriver 项，单击“确定”按钮。单击 Project→Default Project Properties 菜单项，选择 Paths 下的 Driver 按钮，弹出 select one or more libraries，选择 com.microsoft.jdbc.sqlserver.SQLServerDriver 项，然后单击“确定”按钮。单击 Tools→Exterprise setup，在弹出对话框中，按 Libraries 下的 Add 按钮，添加设置的路径于其中，单击 OK。重新启动后，单击 Tools→Database Pilot，打开窗口，单击 View→Options 进入 Driver 页面，单击 Add，可以发现增加的项在其中，选中它，单击“确定”按钮。重复上述步骤，可见 com.microsoft.jdbc.sqlserver.SQLServerDriver 已变黑，表示可以选定使用，在 URL 文本框中输入：jdbc:Microsoft:sqlserver://localhost:1433;DatabaseName=hmrtixing，配置完成。输入用户名和密码，连接成功。

如果在程序中建立连接，在安装 JDBC 驱动后要配置 classpath 路径，以 Windows XP 为例设置如下：

```
classpath=,;c:\Program Files\Microsoft SQL Server 2000 Driver for JDBC\lib\msbase.jar; c:\Program Files\Microsoft SQL Server 2000 Driver for JDBC\lib\mssqlserver.jar; c:\Program Files\Microsoft SQL Server 2000 Driver for JDBC\lib\msutil.jar
```

先编写连接数据库的类，之后对数据库进行操作。下面是一个示例程序，对数据库 Waremanage 中表 Commodity 上的 WareName 与 Specification 字段进行操作。

```
String name=request.getParameter("name");
if (name==null)
    {
     name="";
     }
    byte byte1[]=name.getBytes("ISO-8859-1");
    name=new String(byte1);
    Connection con=null;
    Statement sql=null;
```

```
        ResultSet rs=null;
        try{
             Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver");
             }
    catch(ClassNotFoundException e) {}
          try{
             con=DriverManager.getConnection("jdbc:microsoft:sqlserver:
//127.0.0.1:1433;DatabaseName=Waremanage;SelectMethod=direct","dbo","");
//数据库名 Waremanage，用户名 dbo，密码为空
             stm=con.createStatement();
             String condition="SELECT * FROM Commodity";
             rs=stm.executeQuery(condition);
             out.print("<Table Border='1' bordercolor='#C60001' bgcolor='#FFFFFF'>");
             out.println("<TR>");
             out.println("<TD width='45' align='center' size='4'>"+"WareName");
             out.println("<TD width='55' align='center' size='4'>"+"Specification");
             out.println("</TR>");
             while(rs.next())
             {
               out.println("<TR>");
               out.println("<TD align='center'>"+rs.getString(1)+"</TD>");
               out.println("<TD align='center'>"+rs.getString(2)+"</TD>");
               out.println("</TR>");
              }
             out.println("</Table>");
             con.close();
             }
             catch(Exception e){
                    out.println(e.toString());
                   }
             %>
```

再介绍一个基于 JSP 通过 JDBC 操作数据库的示例程序。该示例程序读取数据库 Waremanage 中表 Commodity 中的所有数据，并将 WareName 与 Specification 字段对应的数据显示在 JSP 页面上。其完整的程序如下。

```
    <%@ page language="java" contentType="text/html; charset=GB2312"
        pageEncoding="GB2312"%>
    <%@ page import="java.sql.*"%>
    <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
    <html>
    <head>
    <meta http-equiv="Content-Type" content="text/html; charset=GB2312">
    <title>Insert title here</title>
    </head>
    <body>
    <%
```

```
        Connection con = null;//声明连接对象
        Statement st = null;//声明语句对象
        ResultSet rs = null;//声明结果集对象
        try {
    /*
    *根据驱动程序的名称查找对应的类。在此我们使用微软的 JDBC 的 SQL Server 驱动程序。
    */
            Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver");
        } catch (ClassNotFoundException e) {
            //如果查找不到对应的类，输出异常。
            e.printStackTrace();
        }
        try {
    /*
    *创建与数据库的连接，如果成功就可以操作该数据库。此数据库名为
    *Waremanage，用户名为 dbo，密码为空。
    */
            con = DriverManager
            .getConnection(

    "jdbc:microsoft:sqlserver://127.0.0.1:1433;DatabaseName=Waremanage;SelectM
ethod=direct",
                    "dbo", "");
            //通过连接创建语句对象，语句对象可以执行 SQL 语句
            st = con.createStatement();



    /*
    *创建标准的 SQL 语句，在此我们创建了一个读取 Commodity 表中的所有数据的 SQL 语句。
    */
            String condition = "SELECT * FROM Commodity";
    //通过语句对象执行该 SQL 语句，从数据库中读取 Commodity 表中的所有数据。
            rs = st.executeQuery(condition);
    /*
    *将读取的数据输出到 JSP 页面中。在此我们只将 Commodity 表中的 WareName
    *字段和 Specification 字段的数据输出到 JSP 页面。
    *由于 WareName 字段和 Specification 字段分别是 Commodity 表中的第 1 个字段和第 2 个字段，
所以通过语句 rs.getString(1) 和 rs.getString(2) 就可以依次读取 WareName 字段和
Specification 字段对应的数据。
    */
            out
            .print("<Table Border='1' bordercolor='#C60001' bgcolor='#FFFFFF'>");
            out.println("<TR>");
            out.println("<TD width='45' align='center' size='4'>"
            + "WareName");
            out.println("<TD width='55' align='center' size='4'>"
```

```
            + "Specification");
            out.println("</TR>");
            while (rs.next()) {
                out.println("<TR>");
//通过 rs.getString(1)读取 WareName 字段的数据，并显示在 JSP 页面上。
                out.println("<TD align='center'>" + rs.getString(1)
                + "</TD>");
//通过 rs.getString(2)读取 Specification 字段的数据，并显示在 JSP 页面上。
                out.println("<TD align='center'>" + rs.getString(2)
                + "</TD>");
                out.println("</TR>");
            }
            out.println("</Table>");
        } catch (Exception e) {
            out.println(e.toString());
        } finally {
//操作完毕后释放资源，包括结果集对象，语句对象和连接对象。
            try {
                if (rs != null) {
                   rs.close();
                }
                if (st != null) {
                    st.close();
                }
                if (con != null) {
                   con.close();
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
%>
</body>
</html>
```

该程序通过 JDBC 操作数据库的基本步骤为：

（1）创建与数据库的连接对象。

（2）通过连接对象创建语句对象。

（3）通过语句对象执行 SQL 语句来操作数据库，并获得结果集对象。

（4）处理结果集对象。

（5）释放资源，包括结果集对象，语句对象和连接对象。

本章小结

本章对 SQL Server 的操作和开发进行了介绍，主要包括：

（1)结合 SQL Server 2000 介绍了管理工具的使用方法，重点介绍了利用企业管理器建库、建表、建立索引、建视图、数据录入等维护操作、实现数据完整性控制、安全性管理等内容；介绍了查询分析器使用方法。

各种数据库都提供了管理工具，这些管理工具有良好的图形界面、向导式操作方法，可以帮助了解相关数据库的基本功能与基本操作，用于建立库、表、数据维护与查询及其他基本应用，帮助进行数据库管理的工作。熟悉这类管理工具无疑是数据库入门的好方法。

（2）介绍 SQL Server 中的 Transact-SQL 语言。Transact-SQL 基于标准 SQL 语言，同时又为了应用需求而扩展了 SQL 语言。许多数据库都对标准 SQL 语言进行了不同程度的扩展。学习 Transact-SQL 可以帮助我们进一步深化对 SQL 语言的学习，同时，在面对一种新的数据库时，也能借助学习 Transact-SQL 的经验，快速地学习与适应其新的理论与技术。

（3)介绍了 SQL Server 中的存储过程和触发器。存储过程和触发器在数据库开发过程中，在对数据库的维护和管理等任务中以及在维护数据库参照完整性等方面具有不可替代的作用。因此无论对于开发人员，还是对于数据库管理人员来说，熟练地使用存储过程，尤其是系统存储过程，深刻地理解有关存储过程和触发器的各个方面问题是极为必要的。在本章中通过实例，展示了有关存储过程和触发器的各种问题：

1）存储过程、触发器的概念、作用和优点。

2）创建、删除、查看、修改存储过程、触发器的方法。

3）存储过程，触发器的应用。

4）创建、使用存储过程和触发器的过程中应注意的若干问题。

（4）介绍了 SQL Server 应用系统开发环境。虽然 SQL Server 提供了大量的工具，但要开发实用的管理信息系统还得依靠其他高级语言。为此，需要有与其他高级语言的接口，使得其他语言能够对数据库中的数据进行操作。为了操作方便、规范及更好的共享性与可扩展性，微软提出了开放式数据库连接（ODBC）接口标准，只要是基于这一规范的数据库开发，不同语言开发应用系统都将只面对一个虚拟的“数据源”，屏蔽了接口的差异。基于同样的原因，SUN 公司提出了一个与数据库连接的 API 标准－JDBC，不同的是，它只是单一地和 Java 语言的数据库接口。本章介绍了 ODBC 与 JDBC 的基本知识。

习题六

1．说明服务管理器的作用及如何操作。

2．简述利用企业管理器可以做哪些工作？

3．简述利用企业管理器建库、建表、建立视图、建立索引的方法。

4．什么是聚集索引？什么是填充因子？

5．SQL Server 是如何实施数据完整性保护的？

6．简述 SQL Server 保证数据安全的方法，如何操作？

7．说明 SQL Server 备份与恢复的机制与操作方法。

8．说明存储过程的意义是什么，简述建立存储过程的方法。

9．SQL Server 有哪些触发器，各自意义是什么？

10．简述查询分析器的启动过程。如何编辑、执行、保存 Transact-SQL 命令？

11．理解 SQL Server 中数据文件与日志文件的作用。

12．在 SQL Server 中用 Transact-SQL 命令建立数据库：Student，其中包括“学生”、“课程”、“成绩”三个数据表，学生表中有字段：学号、姓名、性别、出生日期、所在学院、专业、班级、履历、相片等。课程表中有字段：课程号、课程名称、主教材名称、教师姓名、学时数、学分等。成绩表中有字段：学号、课程号、分数等。各字段数据类型自行设计，其中班级包括入学 4 位年份与 2 位序号。

13．求在“学生”表中用 Transact-SQL 命令添加字段：出生年份，要求限制输入数据在 1970～1990 之间。

14．基于 12 题数据结构写出以下语句：

（1）查询计算机学院 06 级学生男生名单。

（2）打印计算机学院专业与班级清单。

（3）输出计算机学院全体学生姓名、所在学院、专业、班级、课程名称、分数。对于什么课都没选的学生也要显示一条记录。

（4）求选了课程号为 C001 的课程的学生姓名、性别、所在学院、专业、班级、分数，保存到表“选课”中。

（5）求没有选课程号为 C001 的课程的学生姓名、所在学院、专业、班级、分数。

（6）求以课程名称、平均分、最高分、最低分为名显示每门课的情况。

（7）求选了课程号为 C001 的课程且其成绩比计算机学院学生该课平均成绩高的学生姓名、所在学院、专业、班级、分数。

（8）求所有课程成绩均优良的学生名单。

15．设计一个存储过程，当输入一大一小两个 50～100 之间的数时输出分数在这两数之间的所有学生的学院、专业、班级、课程名称、分数记录，要求附加按学院、专业、班级、课程名称的分数平均数。

16．SQL Server 的安全级别分为哪两个层次，各通过哪些设置实施？

17．角色的用途是什么？服务器角色与数据库角色有何不同？

18．要求用 SQL 语句实现：

（1）为用户 Wang 创建一个登录账户，默认数据库为 Student。

（2）将 Wang 的密码由“wang”改为“1a2b3c”

（3）删除一个使用 SQL Server 身份验证的登录账户 Ling。

19．怎样用 Transact-SQL 命令进行备份与恢复。

20．什么是 ODBC 与 JDBC？说明数据源的意义及建立步骤。