“九五”国家重点电子出版物规划项目·计算机知识普及系列

SQL Server 2000 中文版系列丛书 1

Microsoft .net 丛书

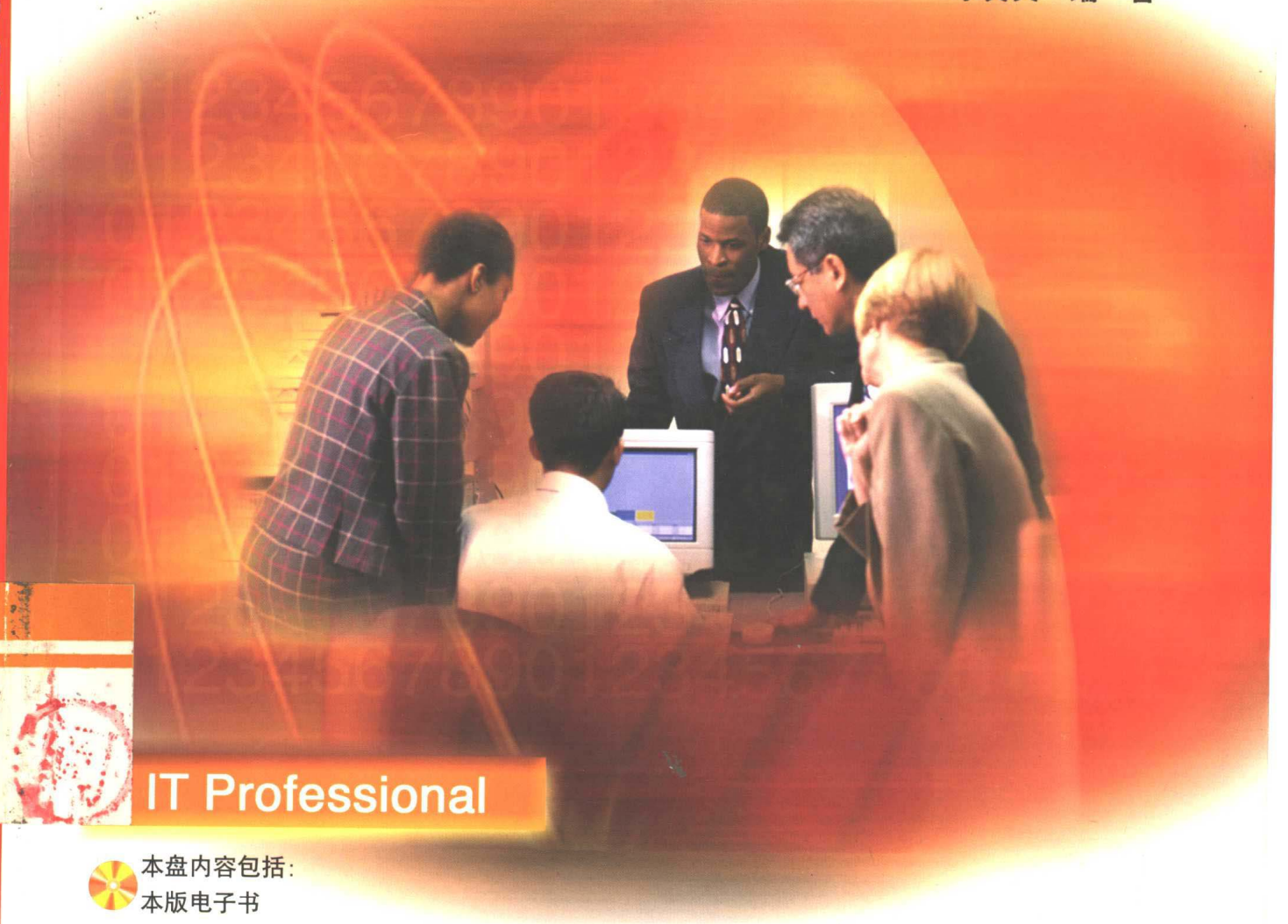
SQL Server 2000

Developer's Guide

开发人员指南

北京希望电子出版社 总策划

李真文 编 著



IT Professional

本盘内容包括：
本版电子书

北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

“九五”国家重点电子出版物规划项目·计算机知识普及系列

SQL Server 2000 中文版系列丛书 1

Microsoft .net 丛书

SQL Server 2000 Developer's Guide

开发人员指南

北京希望电子出版社 总策划

李真文 编 著

IT Professional

本盘内容包括:
本版电子书

北京希望电子出版社
Beijing Hope Electronic Press

内 容 简 介

本盘书是 SQL Server 2000 中文版系列从书之一，主要介绍了在 SQL Server 2000 下进行数据库开发的主要接口技术。本书分三部分，共计 12 章。第一部分，“Transact-SQL 编程”，主要讲解了 Transact-SQL 基本的 DDL 和 DML 语句。其中，第 1 章主要介绍 Transact-SQL 的 DDL 语言，第 2 章主要介绍 Transact-SQL 的 DML 语言。第二部分，“桌面应用开发”，主要讲解通过各种流行的工具、语言和接口访问和操作 SQL Server 数据库。其中，第 3 章综合介绍了 SQL Server 2000 与微软其他产品的集成；第 4 章讲了通过 Access 访问 SQL Server 的技术；第 5 章讲了通过 ODBC API 开发数据库应用程序的方法；第 6 章主要介绍了如何在 Visual Basic 中通过 ADO 接口访问 SQL Server 数据库。第 7 章讲了通过 SQL-DMO 对象进行数据库开发的方法；第 8 章讲了嵌入式 SQL 编程方法，这种方法由于具有很好的可移植性而得到了广泛的应用。第三部分，“Internet 应用开发”，主要讲解在 Web 页面上如何访问和操作 SQL Server 数据库。其中，第 9 章讲了如何用 Web 助理向导发布 SQL Server 数据库信息到 Web 页面上；第 10 章讲了如何在 ASP 页面中通过 ADO 接口访问 SQL Server；第 11 章讲了 XML 的基本知识；第 12 章讲述如何在 XML 页面中访问 SQL Server，这是 SQL Server 2000 的新特征，相信在以后这方面的功能将会更进一步增强，这只是 SQL Server 2000 迈向支持 XML 的序曲。

数据库开发方面的内容非常繁杂，本盘书本着简洁、实用的原则着重介绍了一些流行、实用的数据库编程技术。本书内容翔实、轮廓清晰，示例典型、易于理解且配合有大量的插图和列表，非常适合于具有基本 SQL Server 开发经验的数据库编程人员使用。也可作为高等院校相关专业师生的自学、教学参考书和社会相关领域培训班推荐教材。

本盘内容包括：配套电子书。

系 列 盘 书:	SQL Server 2000 中文版系列丛书（1）
盘 书 名:	SQL Server 2000 Developer's Guide 开发人员指南
总 策 划:	北京希望电子出版社
文本著作者:	李真文 编著
责 任 编 辑:	龙启铭
CD 制 作 者:	希望多媒体创作中心
CD 测 试 者:	希望多媒体测试部
出版、发行者:	北京希望电子出版社
地 址:	北京中关村大街 26 号 100080
	网址：www.bhp.com.cn E-mail: lwm@hope.com.cn
	电话：010-62562329,62541992,62637101,62637102,62633308,62633309
	（发行和技术支持）
	010-62613322-215（门市） 010-62531267（编辑部）
经 销:	各地新华书店、软件连锁店
排 版:	希望图书输出中心
CD 生 产 者:	中新联光盘有限责任公司
文本印刷者:	北京双青印刷厂
规 格 / 开 本:	787×1092 1/16 开本 23.5 印张 546 千字
版 次 / 印 次:	2001 年 1 月第 1 版 2001 年 1 月第 1 次印刷
印 数:	0001～5000 册
本 版 号	ISBN 7-900056-46-7/TP•45
定 价	39.00 元(1CD，含配套书)

前　言

在本套丛书中，数据库开发是指用各种语言和接口访问和操作 SQL Server 2000 数据库。如果你准备从事这方面的工作，或者对这个领域感兴趣，请阅读本书：本书就是为你而准备。

本书分三部分，共计 12 章。

第一部分，“Transact-SQL 编程”，主要讲解了 Transact-SQL 基本的 DDL 和 DML 语句。其中，第 1 章主要介绍 Transact-SQL 的 DDL 语言，第 2 章主要介绍 Transact-SQL 的 DML 语言。Transact-SQL 是 SQL Server 在标准 SQL 上的扩展，是 SQL Server 数据库管理和开发的基础。

第二部分，“桌面应用开发”，主要讲解通过各种流行的工具、语言和接口访问和操作 SQL Server 数据库。第 3 章综合介绍了 SQL Server 2000 与微软其他产品的集成，使读者对 SQL Server 在微软系列产品中的定位有更清晰的把握。第 4 章讲了通过 Access 访问 SQL Server 的技术：Access 作为前端开发工具，而 SQL Server 2000 作为后台数据库，是一种非常流行的 C/S 结构。第 5 章讲了通过 ODBC API 开发数据库应用程序的方法，ODBC 是当今广为流传的一种数据库编程接口，因此尽管它不是最新的技术，但是，它还被广泛采用，本书仍然将它作为重点内容进行介绍。第 6 章主要介绍了如何在 Visual Basic 中通过 ADO 接口访问 SQL Server 数据库。第 7 章讲了通过 SQL-DMO 对象进行数据库开发的方法，SQL-DMO 是一种非常重要的对象模型，其优势不在数据访问而在于管理。通过它可以实现定制的数据库管理功能。第 8 章讲了嵌入式 SQL 编程方法，这种方法由于具有很好的可移植性而得到了广泛的应用。

第三部分，“Internet 应用开发”，主要讲解在 Web 页面上如何访问和操作 SQL Server 数据库。第 9 章讲了如何用 Web 助理向导发布 SQL Server 数据库信息到 Web 页面上。当然，发布的页面是静态的 HTML 页面。第 10 章讲了如何在 ASP 页面中通过 ADO 接口访问 SQL Server，与通过 Web 助手向导发布的页面相比，ASP 页面是动态的页面，内容可以随 SQL Server 数据库的变化而即时进行更新。第 11 章讲了 XML 的基本知识，这是为第 12 章做准备。XML 是下一代 Web 页面描述语言，其最大特点是，其页面的描述形式和数据库有完全的对应关系，这使得 SQL Server 数据库更容易与 Web 集成。和第 12 章以及如何在 XML 页面中访问 SQL Server，这是 SQL Server 2000 的新特征，相信在以后这方面的功能将会更进一步增强，这只是 SQL Server 2000 迈向支持 XML 的序曲。

本书由希望图书创作室成员集体完成，由于时间紧迫，加之水平有限，错误和疏漏在所难免，恳请读者批评指正。

目 录

第一部分 Transact-SQL 编程

第二部分 桌面应用开发

第三部分 Internet 应用开发

第一部分　Transact-SQL 编程

本部分主要讨论 Microsoft SQL Server 的 Transact-SQL 程序设计技术。Transact-SQL 是结构化查询语言(SQL)的扩展，而 SQL 是关系数据库语言的工业标准。SQL 语言分两种：一是用于定义关系数据库的数据，叫做数据定义语言（DDL），二是用于操纵关系数据库中的数据，叫做数据操纵语言（DML）。本部分将分别讨论 DDL 和 DML 设计技术。

第 1 章　Transact-SQL DDL

在 Transact-SQL 语言中，DDL（数据定义语言）是基础，因为它负责各类数据库对象的创建和删除。本章主要介绍下列四个基本 DDL 语句：Create Database、Create Table、Create View 和 Create Index，以及删除这些语句创建的对象：Drop Database、Drop Table、Drop View 和 Drop Index。

1.1　创建数据库

在 SQL 术语中，数据库是一个容器，包含了相关的基表、视图、索引、存储过程和其他对象。在创建这些对象之前，必须有一个存储它们的数据库。在数据库中，对象被进一步组织为有一个所有者。对于某些类型的对象，例如表，只要属于不同的用户，就可以在同一个数据库中有同样的名称，然而，对于产品系统最好不要出现重复的名称。一般情况下，大多数产品对象由数据库所有者拥有。

为了创建数据库，用户必须是系统管理员或者被授权使用 Create Database 语句。Create Database 命令最简单的形式如下：

```
Create Database AppDta
```

这条语句创建 AppDta 数据库，并且把 SQL Server 的 model 数据库定义复制到新数据库中。也就是说，model 数据库中的每一个表、视图、存储过程等等的空拷贝都复制在新数据库中创建。SQL Server 为这个数据库创建两个 NT Server 文件：appdta.mdf 保存数据，appdta_log.ldf 保存事务日志的内容。这两个文件的默认初始大小分别设置为 model 数据库的主文件和日志文件的大小。如果需要，SQL Server 将自动扩展这些文件。

1.1.1　指定位置和大小

如果希望为数据库或事务日志指定一个或者多个特定文件，增加一个 On 子句，列出一个或者多个文件，并可为分配这个文件的空间(以 MB 为单位)指定一个可选值，例 1-1 说明了创建数据库的基本语句。

例 1-1

```
Create Database AppDta
   On  Primary
     ( Name          = AppDta1,
       Filename      = `c:\production\data\appdta1.mdf',
       Size          = 10MB,
       MaxSize       = 100MB,
       FileGrowth    = 10MB),
     ( Name          = AppDta2,
       Filename      = `c:\production\data\appdta1.ndf',
       Size          = 10MB,
```

```
    FileGrowth    = 10MB )
```

On 关键字之后的第一项应该指定可选的 Primary 关键字，表明这是一个主文件，它包含该数据库的系统表和初始化信息。指定的其他从属文件用于保存在用户表和其他对象中的数据。在这个示例中的参数如下：

- Name SQL Server 使用的逻辑名称
- Filename 完全限定的 NT Server 文件名
- Size 文件的初始大小，默认值是 model 数据库主文件(model.mdf)的大小
- MaxSize 最大的文件尺寸，默认值是占满整个空间
- FileGrowth 当需要时，SQL Server 扩展文件的量。默认值是 10%

一个 SQL Server 数据库可以超过一百万 TB 大小 1TB=1000MB。

为了提高性能和可恢复性，可以使用 Log On 子句来指定数据库的 SQL Server 将事务日志存储在一个与数据库对象不同的设备上，如例 1-2。

例 1-2

```
Create Database AppDta
   On Primary
     ( Name          = AppDta1,
       Filename      = ' C:\Production\date\appdta1.mdf' ,
       Size          = 10MB,
       MaxSize       = 100MB,
       FileGrowth    = 10MB),
     ( Name          = AppDta2,
       Filename      = 'C:\Production\date\appdta1.mdf',
       Size          = 10MB,
       FileGrowth    = 10MB)
   Log On
     ( Name          = AppDtaLog1,
       Filename      = 'd:\production\log\appdtalog1.ldf'',
       Size          = 10MB,
       MaxSize       = 100MB,
       FileGrowth    = 10MB )
```

这种方式，如果数据库所在的物理设备被破坏，日志还可以使用(如果该日志所在的设备没有被破坏)。使用一个以前的数据库备份和一个未破坏的日志的脱机拷贝，可以将数据库恢复到保存数据库恢复的设备失败时的状态。当指定明确的文件时，按照 SQL Server 的约定，使用.mdf 作为数据库主文件的扩展名，.ndf 作为数据库从属文件的扩展名，.ldf 作为事务日志文件的扩展名。

1.1.2 修改数据库

在数据库创建之后，可以使用 Alter Database 语句增加新文件、删除已有的文件或修改文件的设置。例 1-3 的语句可增加一个新文件：

例 1-3

```
Alter Database AppDta
   Add File
```

```
    ( Name          = AppDta3,
      Fileanme      = `c:\production\data\appdta3.ndf' ,
      Size          = 10MB,
      FileGrowth    = 10MB )
```

还有一个 Add Log File 子句，它与 Add File 子句有相同的格式。Remove File 子句只使用以前指定的逻辑文件名：

```
Alter Database AppDta
  Remove File AppDta2
```

Modify File 子句需要以前指定的逻辑文件名，并且可以带任何其他的参数，例如，参数 File Growth:

```
Alter Database AppDta
  Modify File
    ( Name    = AppDta1,
      FileGrowth=50MB )
```

1.1.3 定义文件组

数据库文件（不包括事务日志文件）可以组成文件组。最初创建一个数据库时，该数据库的默认文件组包含了主文件和没有明确分配给用户定义文件组的从属文件。在许多情况下，默认的文件组已经足够了。对于有些系统，在指定的设备上创建用户定义的文件组，可以提高数据库性能或可恢复性。因为可以指定表或索引所在的文件组，所以文件组提供了一种间接手段，可以把表和索引放在指定的设备上。另外，当使用一个包含了许多文件的文件组时，SQL Server 根据文件可用的自由空间，把文件组中的数据按比例散布在文件中。下面是一个创建文件组的示例：

```
Alter Database AppDta
  Add FileGrouup AppDtaGroup1
```

当创建一个文件组之后，可以使用 Alter Database 语句把新文件增加到该文件组中。对于那些增加到用户定义的文件组中的文件，一般应该指定一个与该数据库主文件不同的设备，如例 1-4。

例 1-4

```
Alter Database AppDta
   Add File
     ( Name          = AppDta2,
       Filename      = 'd:\production\data\apppdta2.ndf' ,
       Size          = 10MB,
       MaxSize       = 100MB,
       FileGrowth    = 10MB),
     ( Name          = AppDta3,
       Filename      = 'd:\production\data\appdta3.ndf' ,
       Size          = 10MB,
       MaxSize       = 100MB,
       FileGrowth    = 10MB)
   To FileGroup AppDtaGroup1
```

为了删除文件组和文件组所包含的文件，可以使用类似下面的语句：

```
Alter Database AppDta
  Remove File AppDta2
Alter Database AppDta
  Remove File AppDta3
Alter Database AppDta
  Remove FileGroup AppDtaGroup1
```

在删除文件或者文件组时，它们必须为空。

也可以使用 Alter Database 语句改变某个数据库的默认文件组，例如下面的语句：

```
Alter Database AppData
  Modify FileGroup AppDtaGroup1 Default
```

1.2 创 建 表

在关系数据库中，基表包含了实际的数据。在一个 SQL Server 数据库中，可以创建多达两万亿个表。为了用 SQL 创建表，输入一条 Create Table 语句，指定下述内容：

- 包含表的数据库
- 表的所有者
- 表名，在同一个数据库中和同一个所有者下，该表名必须与任何其它基表或视图不同
- 指定 1 到 1024 个列
- 主键约束(可选)
- 1 到 250 个 Unique 约束(可选)
- 1 到 253 个外键约束(可选)
- 1 个或者多个 Check 约束，限制插入表中的数据(可选)
- 存储表的文件组(可选)

1.2.1 创建表的一般要求

例 1-5 示意了一个比较复杂的 Customer 表的 Create Table 语句。

正如例 1-5 所示，Create Table 语句首先指定将要创建的表，然后列出列和约束的定义，之间由逗号分开，用一组括号括住。SQL 是一种自由格式的语言，一条语句可以放在多行上，在字和符号之间使用空格，以提高可读性。该示例说明代码样式：代码以每个列的定义开始，约束单独占一行，每个列的定义中相似的部分对齐。虽然这种列样式不是必须的，但是它比不对齐的文本流形式更容易理解。

SQL 关键字对大小写不敏感：Create Table，CREATE TABLE 和 CrEaTe TaBLE 都是正确的。然而，记住在安装 SQL Server 时选择的排列顺序决定标识符和字符串文字是否对大小写敏感。如果使用对大小写不敏感的排列顺序(默认的安装选项)安装 SQL Server，那么 SQL Server 认为表名 Customer，customer 和 CUSTOMER 是相同的，并且 'x' 等于 'x' 。如果使用对大小写敏感的排列顺序安装 SQL Server，那么 SQL Server 把表名 Customer，customer 和 CUSTOMER 作为不同的标识符，并且认为 'x' 不等于 'X' 。

注意：即使使用对大小写不敏感的排列顺序，系统表中的表名和其他对象名称也是以

输入它们的方式存储，例如Customer，customer 或 CUSTOMER。因此，如果在 AppDta 数据库中显示表的清单，那么例 1-5 所示的语句创建的表将被列为 Customer，使用对大小写不敏感的排列顺序，仍然可以在 SQL 语句中使用 CUSTOMER 或者 customer 引用该表。

例 1-5 中使用的表名是一种完全限定的名称，在未限定的表名（Customer）之前包括数据库名称(AppDta)和所有者名称（dbo）。使用小圆点"."作为限定名称的分隔符。

当创建数据库对象时，一般建议在代码中明确指定数据库名称和所有者名称，这样可以归类表的数据库和所有者，避免偶然在错误的数据库中或者使用错误的所有者创建表。正如本示例所示，为了创建一个由数据库所有者拥有的表，可以在限定的名称中使用 dbo 。如果省略了数据库名称，就会在当前数据库中创建表。当前数据库可以是分配给 SQL Server 帐户的数据库，或者是在 Use 语句中指定的数据库，例 1-5 是 Customer 基表的 Create Table 语句。

例 1-5

```
create Table AppDta.dbo.Customer
    ( CustId            Int             Not Null
                                        check(CustId>0),
      Name              Char(30)        Not Null
                                        check(Name<>''),
      ShipLine1         VarChar(100)    Not Null
                                        Default '',
      ShipLine2         VarChar(100)    Not Null
                                        Default '',
      ShipCity          Char    (30)    Not Null
                                        Default '',
      ShipState         Char    ( 2)    Not Null
                                        Default '',
      ShipPostalCode1 Char    (10)      Not Null
                                        Default '',
      ShipPostalCode2 Char    (10)      Not Null
                                        Default '',
      ShipCountry       Char    (30)    Not Null
                                        Default '',
      PhoneVoice        Char    (15)    Not Null
                                        Default '',
      PhoneFax          Char    (15)    Not Null
                                        Default '',
      Status            Char    ( 1)    Not Null
                                        Default '',
      CreditLimit       Money( 1)       Not Null
                                        Check (
                                        (CreditLimit Is Null) Or
                                        (CreditLimit >=0));
      EntryDateTime     DateTime        Not Null
                                        Default Current_Timestamp,
```

```
    RowTimeStamp        imeStamp            Not Null,
    Constraint          CustomerPk          Primary Key ( CustId),
    Constraint          CustomerStatus      Check ( ( Status<>' ')Or
                                            (CreditLimit Is Null) ) )
Use AppDta
```

所有者名称也可以省略，SQL Server 会用当前的用户名作为所有者，完全限定的表名和视图名必须是唯一的。

SQL 对象名称可以长达 128 个字符，包括字母、数字和下述特殊的符号：_(下划线)、#(井号)、$(美元符号)和@(at 号)。如果只使用字母(A-Z)开头并且只包含字母和数字(0-9)，对于跨系统或者跨国应用程序，可以避免可能出现的命名问题。

另外，Transact-SQL 有多个保留字，例如 Create，Table 和 Order，它们有特殊的含义。这些保留字列在 Microsoft SQL Server Transact-SQL 参考手册中的“保留关键字”话题中。如果希望使用这些保留字作为表名、列名或者其他 SQL 对象名称，那么当它们出现在 SQL 语句中时，必须用双引号括起这些名称。下面这个示例说明如何编写一个 SQL Server 语句，从表名为 Order 的表中检索行：

```
Select *
  From "Order"
  Where CustId = 499320
```

引号括起的名称也可以用于包含特殊字符的名称，如下示例：

```
Select *
  From" Order+Detail"
  Where CustId = 499320
```

一般应避免使用 SQL 的保留字或者特殊的字符作为 SQL 对象的名称。

注意： 必须使用 SQL Server 的 “quoted identifier” 用户选项才能使用引号标识符。Transact-SQL 还允许使用方括号作为关键字的分隔符，例如 [Order]。这种语法不需要引号标识符用户选项。

1.2.2 在指定文件组上创建表

正如在本章前面“定义文件组”一节中讨论的那样，可以定义文件组，它是一个或者多个存储应用程序数据的操作系统文件的集合。为了把一个新表放在用户定义的文件组上，可以在 Create Table 语句的末端增加一个 On 子句：

```
Create Table AppDta.dbo.Customer
   ( CustId        Int             Not Null
                                   Check ( CustId > 0 ),
   Name            Char( 30 )      Not Null
                                   Check ( Name <> ' ' )
...
Constraint CustomerStatus Check ( ( Status <. ' ' ) Or
                        ( CreditLimit Is Null ) ) )
On AppDtaGroup1
```

这个文件组必须已由 Alter Database 语句创建。

注意： SQL 还有一个 Create Schema 语句，用于将多个 Create Table, CreateView 和

Grant 语句组合到一个 SQL 语句中。当处理 Create Schema 语句时，SQL Server 按序连续创建对象，以便满足所有的逻辑依赖关系(例如，视图和外键)。虽然，一般由代码生成器使用 Create Schema 语句，但是该语句也可以用于手工创建有从属外键的两个或者多个表。

1.3 表 的 列

在 Create Table 语句中，在新表名之后，可以编码 1 到 1024 个列定义。行的最大长度是 8060 字节，包括内部数据所需的字节。实际中，每行最大的应用程序数据要低于 8060 字节。例如，有 10 个列声明为 Character 数据类型的表最多可以保存 8038 个应用数据字节。

每一个列定义指定列名和一种数据类型。有些数据类型有长度或者精度(数字总长)。另外，Decimal 和 Numeric 数据类型有小数(小数点右端的位数)。表 1-1 列出了 SQL 列的数据类型。

表 1-1 SQL 列的数据类型

数据类型	描述
字符和文本	
Char(*length*)	固定长度的字符串，长度从 1 到 8000（如果省略了长度，那么默认值是 1）
Character(*length*)	
NChar(*length*)	固定长度的 Unicode 字符串，长度从 1 到 4000(如果省略了长度，则默认长度是 1)
NCharacter(*length*)	
National Char(*length*)	
National Character(*length*)	
Char Varying(*max-length*)	变长度的字符串，最大长度从 1 到 8000。如果省略了 *max-length*,则默认长度是 1
Character Varying(*max-length*)	
VarChar(*max-length*)	
nChar Varying(*max-length*)	变长度的 Unicode 字符串，最大长度从 1 到 4000，如果省略了 max-length,则默认值是 1
nCharater Varying(*max-length*)	
nVarChar(*max-length*)	
National Char Varying(*max-length*)	
National Character Varying(*max-length*)	
National VarChar(*max-length*)	
Text	变长度字符数据，最多达到 2，147，483，647 字节。行中存储指向第一个数据页的指针，实际的文本是以 b-树数据页存储
nText	变长度的 Unicode 字符数据，最多可达 1， 073， 741， 823 个字符(或者 2 ，147， 483， 646 字节。行中存储指向第一个数据页的指针，实际的文本是以 b-树数据页存储
National Text	
Dec(*precision,scale*)	数值。precision 是位数，范围是 1 到 38。scale 是小数点右边的位数，范围是从 0 到 precision 指定的数字。可用 Decimal(p)表示 Decimal(p,0),也可以用 Decimal 自身表示 Decimal(18,0)，然而，Decimal 使用明确的 Precision，有助于提高文档的清晰度。
Decimal(*precision,scale*)	
Numeric(*precision,scale*)	

（续表）

数据类型	描述
	注意，对于 Decimal 列，SQL Server 的最大默认 precision 是 28。使用/p 命令行开关启动 SQL Server，可以设置该值为其他值
数字	
TinyInt	单字节、无符号、二进制整数，范围是 0 到 255
SmallInt	两字节二进制整数，范围是-32,768 到 32,767
IntInteger	四字节二进制整数，范围是-2,147, 483,648 到 2,147, 483, 647
BigInt	八字节二进制整数，范围是-1.8E19 到 1.8E19
Float(*mantissa-size-in-bits*)	浮点数。Mantissa-size-in-bits 是尾数的位数，范围是 1 到 53。1-24 指定单精度(4 字节)，25-53 指定双精度(8 字节)。 注意：可以使用 Float 本身表示 Float(53)
Real	等价于 Float(23)，Real 列有 7 位数精度
Double Precision	等价于 Float
货币	
SmallMoney	四字节数字，小数点右面有四位数字。范围是-214，748.3648 到 214，748.3647
Money	八字节数字，小数点右面有四位数字，范围是从-922，337，203，685，477.5808 到 922，337，203，685，477.5807
日期和时间	
SmallDateTime	四字节日期和时间，日期范围是 1-1-1900 到 6-6-2079。时间精度是自午夜开始的 1 分钟之内
DateTime	八字节日期和时间，日期范围是 1-1-1753 到 11-31-9999，时间精度：3.33 毫秒之内
字节、二进制和图像	
Bit	一位，数字 0 或者 1
Binary(*length*)	固定长度二进制数据，长度从 1 到 8000 字节（如果省略 Legth,默认值是 1）
BinaryVarying(*max-length*)	变长度二进制数据，最大长度从 1 到 8000 字节（如果省略 max-Legth 默认值是 1）
VarBinary(*max-length*)	
Image	变长度二进制数据，最长为 2,147,483,647 字节。行中存储指向第一个数据页的指针，实际的数字以 b-树数据页存储
系统类型	
SysName	等价于 nChar Varying(128)
UniqueIdentifier	唯一标识数字，存储为 16 字节的二进制值
TimeStamp	当插入或者修改行时，由 SQL Server 指定的唯一的、8 字节、时间序列值。注意：名称是 TimeStamp 而无数据类型的列是使用 TimeStamp 数据类型创建的
Cursor	Cursor 允许在存储过程中创建游标变量（游标允许一次一行地处理数据）。这个数据类型不能用作表中的列数据类型。
Sql_variant	可包含除 text, ntext, image, 和 timestamp 之外的其它任何数据类型
Table	可用来声明 T-SQL 语句的变量，也可作为自定义函数的返回值，但不能用来定义表的列。

在上述所有的数据类型中，text，ntext 和 image 是较为特殊的三个。由于 text，ntext 和 image 型的数据最大可达 2GB，所以在 7.0 及以前的版本中，用 text，ntext 和 image 定义的列，不能在每一行中直接存储数据，而只能在其中存储指针，由指针指向实际存储 text，ntext 和 image 型数据的页面。而在 SQL Server 2000 中，情况有所变化，用户可以用系统存储过程 sp_tableoption 的开关选项 text in row 来决定是否直接在行中存储数据。

当 text in row 被设置为 off 时，同 7.0 及以前的版本一样，只能在其中存储指针，指向实际的页面，而不能存数据。

当 text in row 被设置为 on 时，就可以在其中存储较小的 text，ntext 和 image 型数据，只有当数据超过了一行所允许的范围时，才存到单独的页面文件中去。

1.3.1 TimeStamp 列

无论何时插入或者修改一行数据时，都会自动修改用 TimeStamp 数据类型说明的列（或名为 TimeStamp 但没有数据类型的列）。SQL Server 设置 TimeStamp 列为下述值：保证该值在一个数据库中是唯一的，并且大于以前指定的值。该值的数据类型与 DateTime 数据类型不同。永远也不能直接设置 TimeStamp 列的值。

一个表只能有一个 TimeStamp 列。可以使用 TimeStamp 列的值决定自从上次检索之后是否有其他进程修改数据行。在应用程序中，检查 TimeStamp 列的值提供了一种有效的方法，可以用这种方法在防止与其他进程的修改冲突时，允许并行浏览和修改表。

1.3.2 Identity 列

SQL Server 允许标识表中一个整数(TinyInt，SmallInt，Integer，Decimal(p，0)或者 Numeric(p，0))列作为该表的 Identity 列。为此，可以在列的数据类型之后增加 Identity 关键字，如下示例:

```
...,
RowId Integer Identity,...
```

当插入一个新行时，SQL Server 自动为 identity 列分配一个数字。在默认情况下，对每一个新行，这个数字以 1 开始，增量是 1。通过在关键字 Identity 的后面指定一个或者两个值，可以设置起始值和增量值。

```
RowId Integer Identity(10),...Start at 10,increment by 1
RowId Integer Identity(100,2),...Start at 100,increment by 2
```

注意，Identity 列不允许空(参见下一节)并且不能保证唯一性，除非指定一个主键约束或者唯一约束，或者为该列创建一个唯一索引。

1.3.3 行全局唯一标识符列

SQL Server 还允许在表中标记一个 UniqueIdentifier 列作为一个行全局唯一标识符(GUID)列。为此，可以在列的数据类型之后增加 RowGuidCol 关键字，例如下面的例子：

```
...,
RowGuid UniqueIdentifier RowGuidCol,...
```

当插入一个新行时，RowGuidCol 列不能像一个 Identity 列那样自动修改。然而，可以将 NewId 函数指定为列的默认值，以达到类似的效果。