
学校的理想装备
电子图书·学校专集
校园网上的最佳资源
SQL Server 7
参考手册（二）
BOOK
内部资料 非卖品

返回总目录

目　　录

第二部分　安装

第 11 章安装 SQL Server

本章将介绍怎样安装 SQL Server 7。如果你现有一更早版本的 SQL Server 数据库，参阅 13 章“升级 SQL Server”。通过插入 SQL Server CD-ROM 的驱动器可以获得安装程序的第一个屏幕如图 11.1 所示。Setup.exe 程序将自动运行，并且出现一个 Windows 图形用户界面屏幕，开始安装 SQL Server 7。如果 Setup.exe 程序不自动地运行或你正在一网络驱动器上访问它，从适当的目录中运行 Setup.exe 程序。

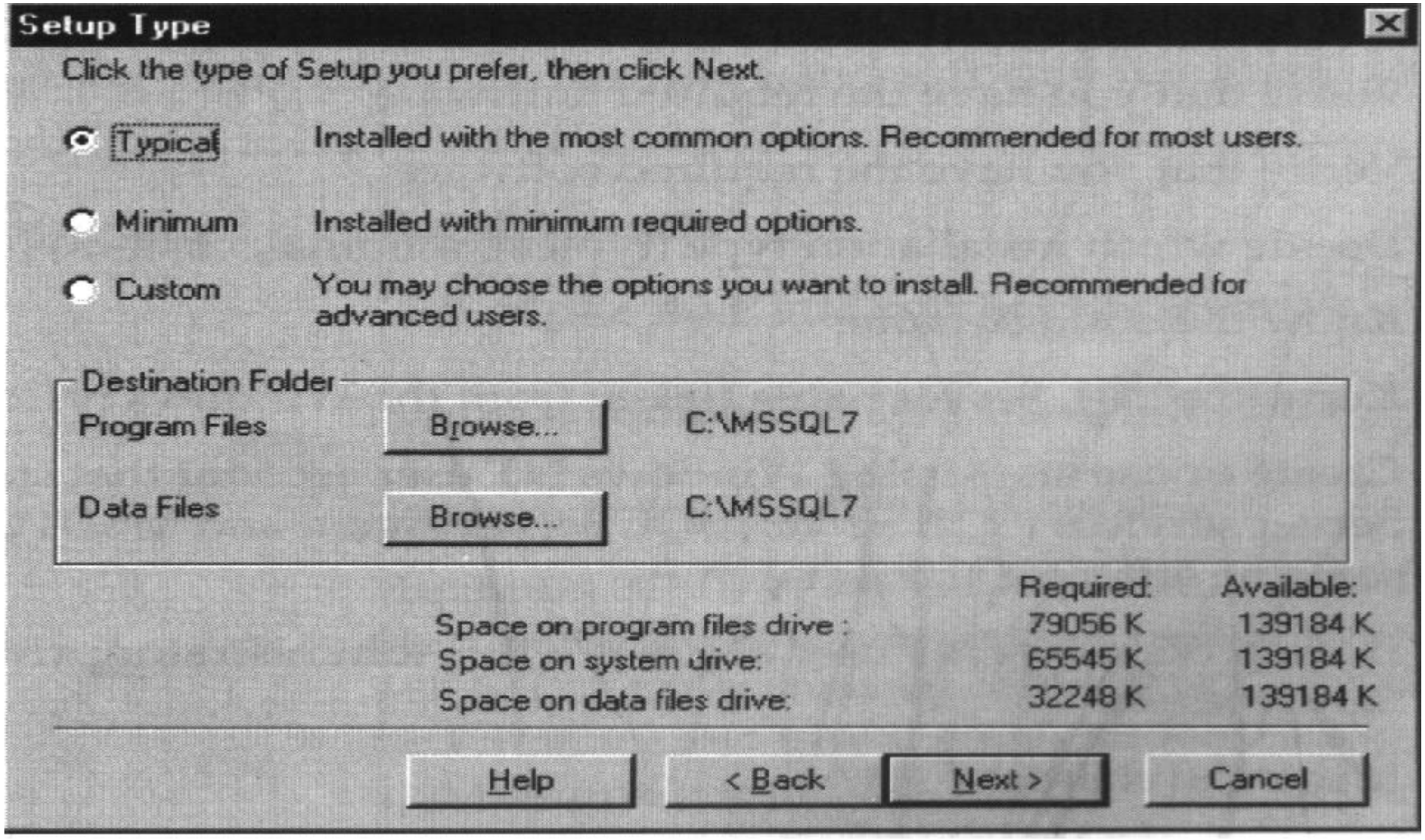

图 11.1SQL Server 的安装

SQL 服务器提供三种安装类型：

· 典型
· 最小
· 定制

以下这些指令用来选择安装类型。读这些指令弄清楚哪种安装类型对你的情况是最好的。

如果适合下列条件，执行典型安装：

· No existing data to convert
· Network Protocols are Named Pipes,TCP／IP Sockets,and Multi-Protocol
· Defeult character set,1252-ISO Character Set
· Default sort order,dictionary order,case-insensitive
· Default Unicode collation,general,case insensitive
· You wish to install all Client Management Tools
· You wish to install Online Books
· You wish to keep the default program file location of ＼MSSQL7
· You wish to keep the default data file location of ＼MSSQL7
· 148 MB disk space is available

如果以下条件成立，执行最小安装：

· 有 112MB 的最小磁盘使用空间
· 不需要在线帮助

如果需要将 SQL Server 6.x 转换成 SQL Server 7(参见 13 章“升级 SQL

Server”)或者需要更改下列安装选项中任一项，可以使用定制安装：

- Network protocols
- Character set
- Sort order
- Unicode collation
- Server components to install
- Installation of online documentation
- Program file location
- Data file location
- Logon account for SQL Server
- logon account for SQL Server Agent
- Autostart of SQL Server

注意： 使用 SQL Server 7 安装光盘。

在选定安装种类后，考虑安装前的准备。

11.1 安装前的准备

在安装象 SQL Server 这样的应用程序之前，尽可能地多做些准备总是好的。首先，确定在运行 SQL Server 时计算机写高速缓冲磁盘控制器是禁止使用的。如果做不到这一点，在你最不想要的时候会导致数据库错误百出。有一些驱动

器，其高速缓冲对 SQL Server 是安全的。和硬件供应商讨论并解释：SQL Server 是一个数据库系统，该系统依赖于恢复和不能有脏页损失的向前写机制。越来越多的高速缓冲器与电池备份和其它机制的到来确保了这一点。

接下来是启动安装程序前必须执行的物理任务：

- 确定你有足够的磁盘空间安装 SQL Server 7
- 备份任何现有的 SQL Server 6.x 数据库
- 在运行 SQL Server 安装程序之前，关闭任何在服务器上运行的其它应用或小程序在启动安装程序之前，也需要收集信息并做一些判断。审阅安装选项，使用下面提供的安装前准备列表写下你的选择，以便当运行安装程序时，知道该选择什么。下面的安装前的列表能帮助汇编安装 SQL Server 7 所需的信息：
- 确定有所需要的硬件
- 确定有所需要的软件
- 确定所用的安装类型(典型，最小，定制)
- 知道域名
- 知道 SQL Server 名(从 Windows NT 计算机名取)
- 创建或使用现有的 Windows NT 用户帐号启动 SQL Server 服务(SQL Server，SQL Server Agent 和 MSDTC 能使用相同的或不同的帐号)
- 有一个具有运行 SQL Server 程序特权的 NT 管理员帐户
- 知道字符集
- 知道排列顺序

- 选择默认 Unicode Collation 或知道使用哪个
- 选择一个 SQL Server 验证方式(NT 验证或混合式)
- 知道主要的 SQL Server 用户名
- 知道公司名称
- 知道序列号
- 知道打算使用哪个网络协议
- 决定是否安装 Online Books
- 知道 SQL Server 数据文件的放置位置
- 知道放置程序文件的位置
- 决定是否使用自动起动 SQL Server

如果对安装前准备列表中的选项项目有不解之处，下一节“安装选项和要求”中的内容能帮助你做出决定。

11.2 安装选项和要求

接下来这节提供关于安装前准备组件和要求的详细数据。

11.2.1 设备需要

在安装 SQL Server 7 之前的准备工作中既需要软件也需要硬件。下列表格

给出硬件和软件的要求。

表 11.1 硬件要求

硬件组件	要求
计算机	DEC Alpha AXP 和兼容系统 Intel 和兼容机系统 486／33MHz 或更高奔腾或 PRO 处理器
内存	最小内存不能少于 32MB。安装内存和你能负担的或你的机器将支持的一样多。正确地调整内存大小将极大地增强性能并能给出足够的内存支持 SQL Server 使用的应用程序。当你有大的索引表格时，如果能从内存检索，系统将比页面到磁盘的索引执行的更好。如果没有大量的数据，可以安装较小的内存
光盘驱动器	标准的 CD-ROM 驱动器
硬盘最小空间安装	80MB
硬盘典型空间安装	185MB
硬盘定制空间安装	如果从 SQL Server 6.x 升级，将除了需要 185MB 以外还需要多出 SQL Server 6.x 非系统数据库所需空间的 1.5 倍。这将在第 13 章“升级的 SQL Server”中介绍
NT 支持网络适配器	仅当你打算在一网络上使用 SQL Server 时需要。对一个独立安装的 SQL Server 来说，不需要网络适配器

表 11.2 软件要求

软件组件	要求
操作系统	Windows NT Server 4.0 或工作站、Service Pack 4 以及任何随后的版本
客户机	Windows 95 ／ 98,Windows NT Workstation,UNIX,Apple Macintosh 和 OS／2 小商业服务器
Internet	Internet Explorer 4.01
网络软件	如果正在使用 Banyan VINES 或 AppleTalk ADSP，将需要附加的网络软件

11.3 缺省域

如果打算使用网络，对可信任的连接网络用户，SQL Server 关心的是缺省 Windows 域的名字。当 SQL Server 正在增加一个安全性帐户，它将以缺省域名称命名。可以使用微软 Windows 控制面板访问 Network 图标查看域名。

尝试不在主域控制器上或备份域控制器上安装 SQL Server。这些 NT 机器正忙于帐户数据库、认证和 NT 复制。相反，在一个域的成员服务器上安装 SQL Server。当安装 Windows NT 时，能指定一个服务器成员。

11.3.1SQL Server 名称

SQL 服务器名的第一个字符可以是字母或是没有嵌入空格的下划线。记住该名称实际上是计算机名，该计算机名是当微软 Windows 操作系统装到计算机上时选择的。你能容易地更改名称，到 Windows Control Panel,在 Network 图标上单击 Identification 标签和 Change 按钮。为了使这种改变生效，必须在改名后重新启动计算机。

11.3.2SQL Server 和 SQL Server Agent 程序和 MSDTC 登录帐号

如果在安装期间选择一个用户帐号域(为分配处理和复制与其它计算机相互作用所用)代替一个当地的系统帐户，一个 Windows 用户帐号必须在运行 SQL Server 安装程序之前建立。用 Windows NT User Manager 创建该帐号。

注意：该 User Manager 能由按下 Start Menu 按钮激发，选择 Programs,Administrator Tools(Common),User Manager for Domains,New User。

选择 Password Never Expires 和 User Can Not Change Password 复选框，以避免口令在用户 NT 级别而非 SQL Server 启动帐目级别被更改。如果用户试着更改口令，将引起 SQL Server 不能启动。清除其它。图 11.2 给出该屏幕。该新建的帐户必须是该管理员组的一成员。

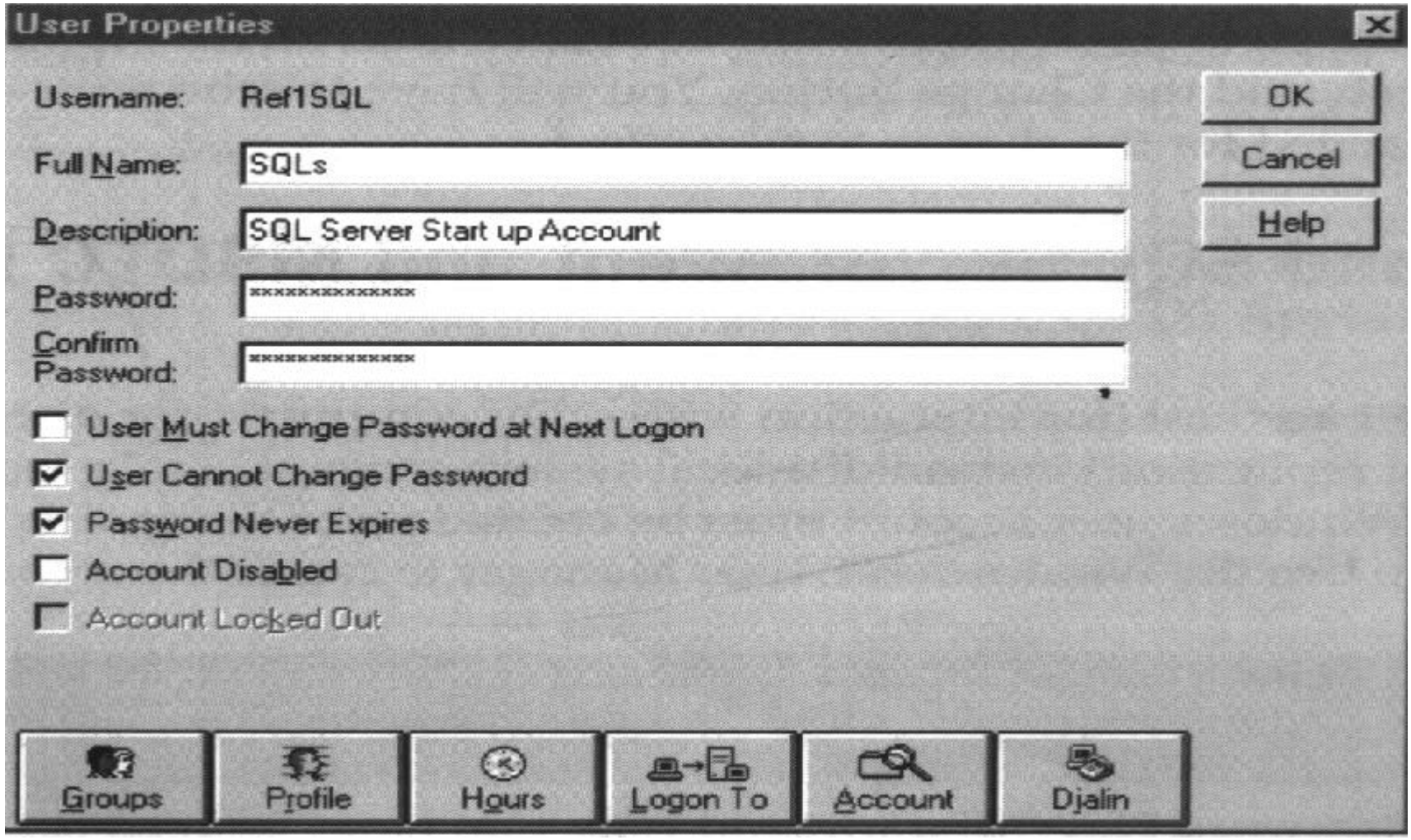

图 11.2 域用户管理器、新用户

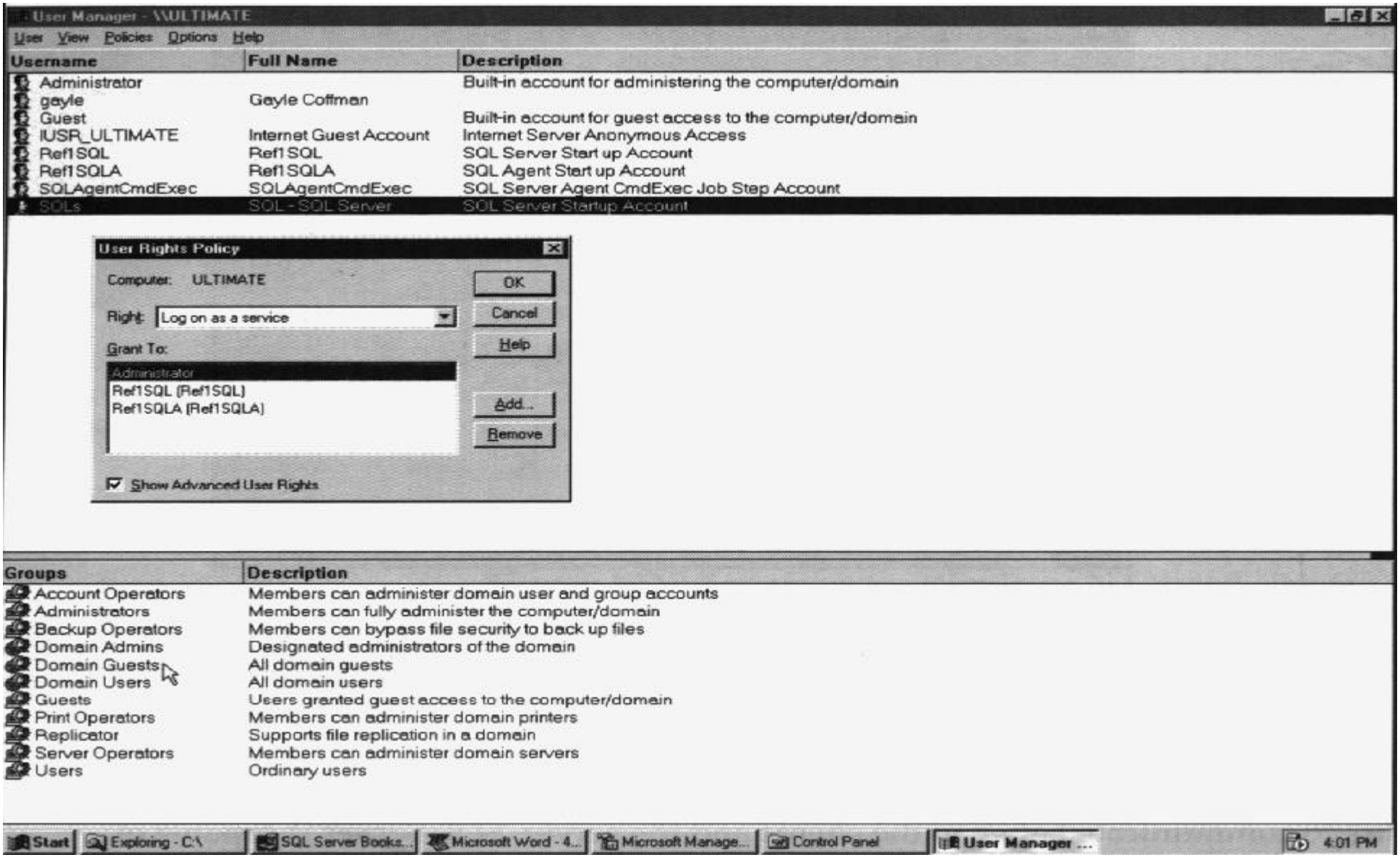
User Manager - \\ULTIMATE
User View Policies Options Help
Username
Full Name
Description
Administrator
Built-in account for administering the computer/domain
gayle
Gayle Coffman
Guest
Built-in account for guest access to the computer/domain
IUSR_ULTIMATE
Internet Guest Account
Internet Server Anonymous Access
Ref1SQL
Ref1SQL
SQL Server Start up Account
Ref1SQLA
Ref1SQLA
SQL Agent Start up Account
SQLAgentCmdExec
SQLAgentCmdExec
SQL Server Agent CmdExec Job Step Account
SQLs
SQL - SQL Server
SQL Server Startup Account
User Rights Policy
Computer: ULTIMATE
OK
Right: Log on as a service
Cancel
Help
Grant To:
Administrator
Ref1SQL (Ref1SQL)
Ref1SQLA (Ref1SQLA)
Add...
Remove
Show Advanced User Rights
Groups
Description
Account Operators
Members can administer domain user and group accounts
Administrators
Members can fully administer the computer/domain
Backup Operators
Members can bypass file security to back up files
Domain Admins
Designated administrators of the domain
Domain Guests
All domain guests
Domain Users
All domain users
Guests
Users granted guest access to the computer/domain
Print Operators
Members can administer domain printers
Replicator
Supports file replication in a domain
Server Operators
Members can administer domain servers
Users
Ordinary users
Start
Exploring - C:\
SQL Server Books...
Microsoft Word - 4...
Microsoft Manage...
Control Panel
User Manager ...
4:01 PM

图 11.3 域用户管理器

在 New User 上用 Group 按钮来将该帐户放入 Administrator 或 Domain Admin 组中。当用户帐户被赋于管理员特权时，它必须作为一服务权力被赋于 Logon。在你选择了用户后，就选 Policy。在命令条 User Manager 主菜单中，选择 User Rights。在这个屏幕中，单击 Advanced Rights 复选框。如果该复选框不被选择，那么作为服务器权力的 Logon 就不会在下拉列表框中出现。

为用户帐户选择 Logon 作为服务权力。

注意 ：图 11.3 这个屏幕可以用 Microsoft Windows NT Start Menu | Programs | Administrator Tools(Connon)User Manager for Domains 来访问，然后选择 User | Polity | User Rithts 复选 Advanced Reights 复选框，从其下的列表中选择“ Logon as a service”。

可以分别为 SQL Server、SQL Server Agent 程序和 MSDTC 设置帐户，也可以选择相同的帐目。在 SQL Server、SQL Server Agent 和 MSDTC 安装程序中，选择 Use a Donain User Account。

11.3.3 具有运行 SQL Server 安装的管理员权力的 Windows NT 用户帐户

当正在安装 SQL Server 时，在一个用户帐户下登录该系统，该用户帐户是 NT 管理员组或域管理组的成员。用 NT 用户管理器来校验用来运行安装的帐户是具有系统管理员权力的帐户。

11.3.4 用户名、公司和序列号

在安装过程中，必需输入 SQL Server 用户主要负责人的名称。为了支持目的，公司和序列号也应输入。能在该方框上发现系列号，在 SQL Server 安装 CD 已捆梆好了。

11.3.5 将现有的 SQL Server 数据转换到 SQL Server 7

转换现有的 SQL Server 6.x 数据的过程是一个选项，该选项在执行常规安装时是存在的。这个过程将在第 13 章“升级 SQL Server”中详细介绍。然而对这个讨论的目的来说，我们还没选择“Yes run the SQL Server Upgrade Wizard”的单选按钮，如图 11.4 所示。

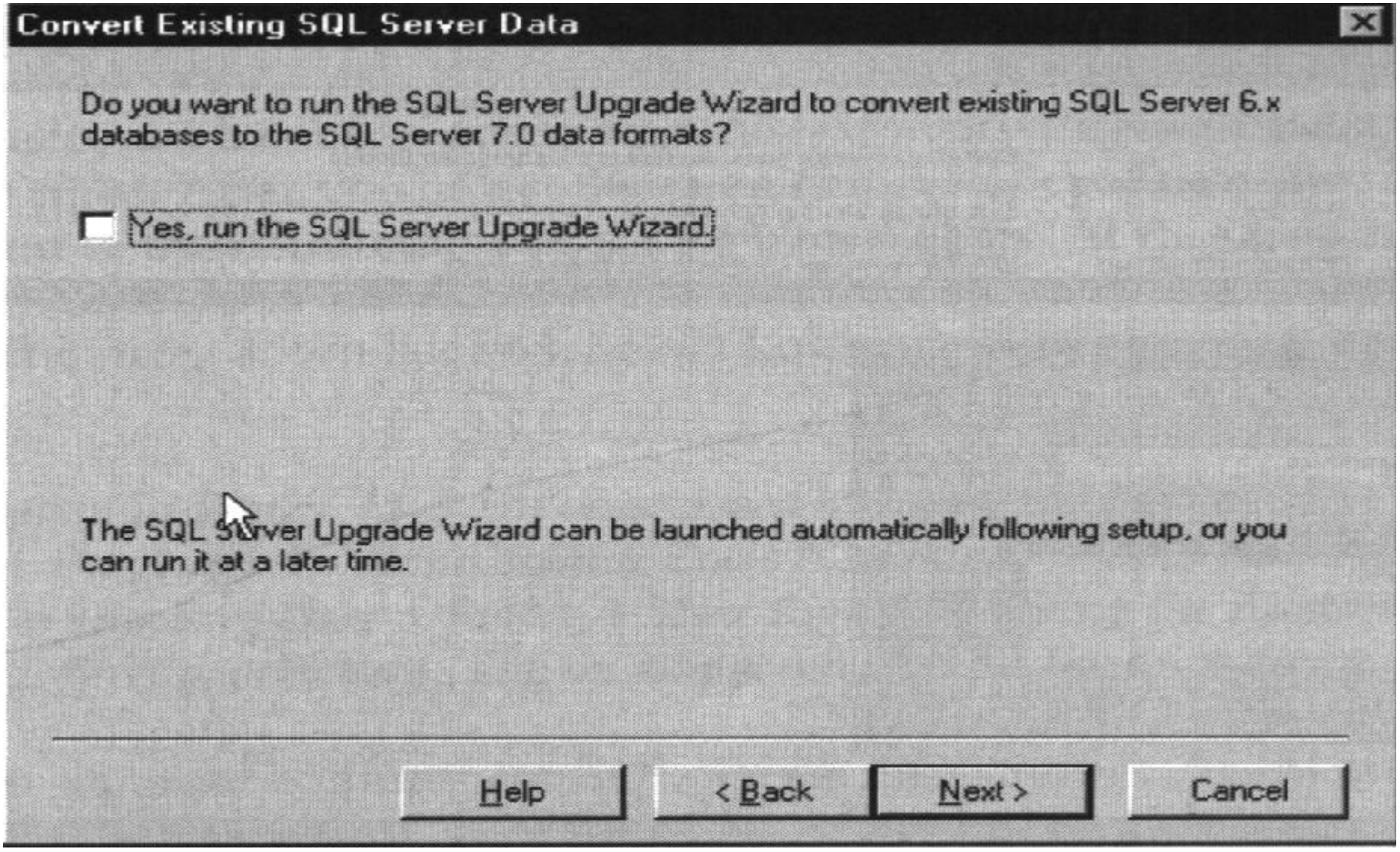

图 11.4SQL Server **安装、转换现有的数据**

11.3.6 网络协议

Network Protocol 安装选项允许选择用来在 Client 委托方和服务器之间交流的网络库。SQL Server 能够同时监视多个网络 Protocol，因此，可以在表 11.3 中选择多个网络协议。

表 11.3 可用的网络协议

网络协议	描述
Named Pipes	Named Pipes 是在 Windows NT 上运行的 SQL Server 的默认值，SQL Server 在标准的管道上监听＼＼.＼pipe＼sql＼query 为命令的管道连接
TCP／IP Sockets	TCP／IP Sockets 是 Windows 95 的默认值。使用 Windows Sockets，需要输入端口数，该默认值是 1433，SQL Server 的 Internet Assigned Number Authority(IANA)插口数。如果你使用微软代理服务器将 SQL Server 安装到代理服务器上监听越过 TCP／IP 插口，参阅第 9 章“数据加密”这节
Multiprotocol 网络库	Multiprotocol 网络库使用 Windows NT 远程过程调用设备。Windows NT Authentication 在 TCP／IP Windows 插口，NWLink IPX／SPX 和 Named Pipes 协议之上移动。Multiprotocol 支持加密，见第 9 章

续表

NWLink IPX／SPX	这个网络库已被用于 Novell IPX 委托方了。你将被访问到 Novell Bindery 服务器名(SQL Server 的计算机名是默认名)在 Novell 网络上注册 SQL Server
AppleTalk ADSP	苹果 Macintosh-based 客户机使用这个连接 SQL Server。当安装这个网络库的时候，你将访问到 AppleTalk 服务对象的名子。如果你选择，你可以输入 SQLServer 计算机名
Banyan VINES	Banyan VINES 支持 Windows NT-based 客户机，并且服务器仅在 Intel 平台上对 SQL Server 是有用的。你将被问到 StreetTalk 服务名,该名称可用被叫做 MSERVICE 的用 Vines 软件程序创建。服务名@group@org 是输入 StreetTalk 服务名的格式

可用于 SQL Server 的网络协议如图 11.5 所示。

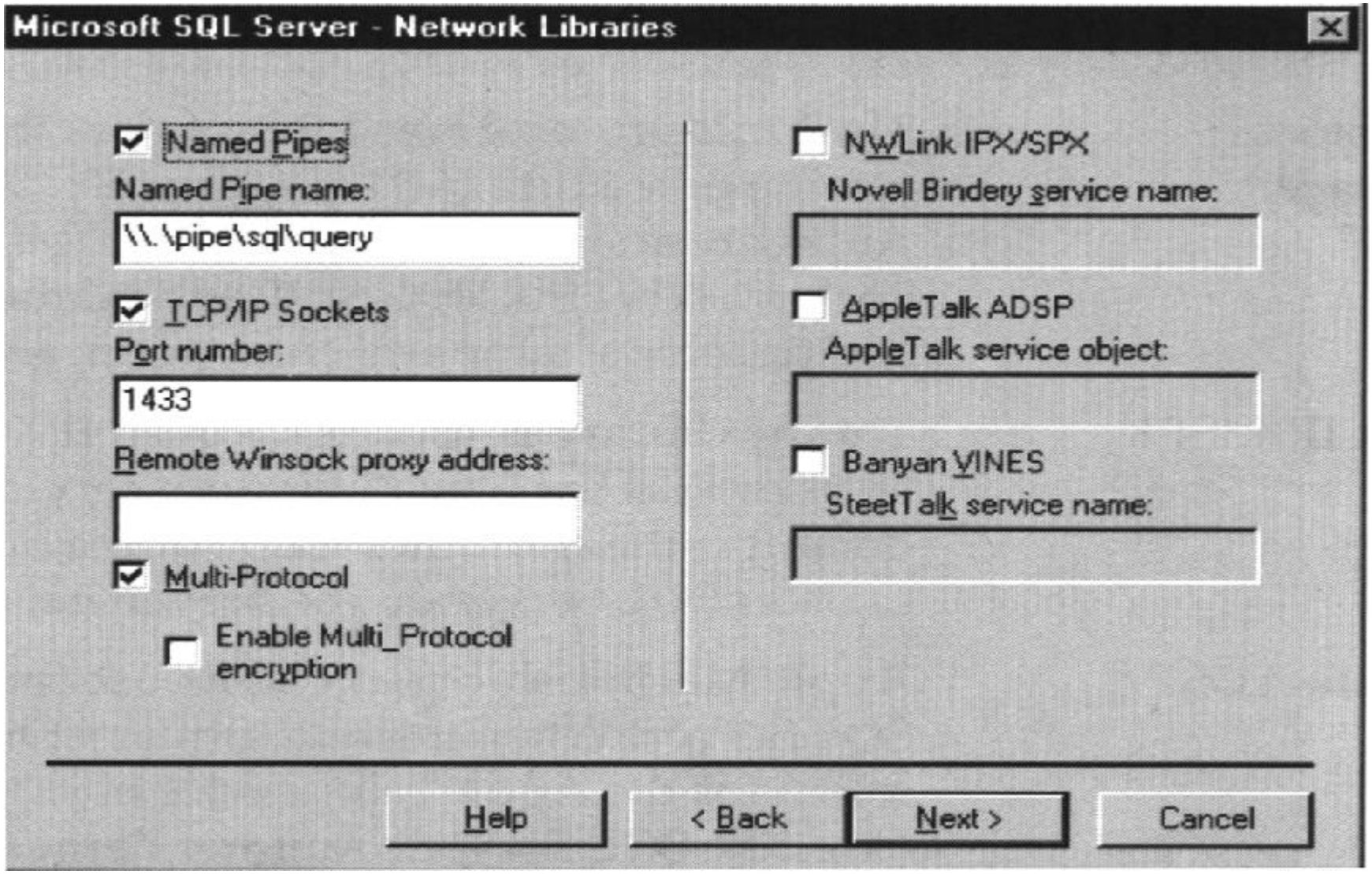

图 11.5SQL Server 安装及网络协议

11.3.7 字符集

在安装过程中为应用程序选择正确的字符是重要的。该默认字符集是代码页 1252(ISO Character Set)。一个字符集是由 256 个大写字母和小写字母、数字以及符号组成，最后 128 个字符可区分各个字符集。这些字符是 SQL Server 能够储存到数据库中的。不同的语言使用不同的最后 128 个字符集。为了使 SQL Server 储存这些不同语言的特殊字符，该字符集对该语言来说必须是正确的一个。表 11.4 描写了一些字符集。

表 11.4SQL Server 7 字符集

SQL 服务器字符集	描述
代码页 1252	ISO 字符集——SQL Server 默认字符集。该字符集也叫 ANSI 字符集
代码页 850	多种语言字符集——该字符集为欧洲、北美和南美语言。这有利于使用扩展字符的微软 DOS 的各种应用
代码页 437	美国英语——包括不常在数据库中使用的图形，用 1252 页代码(默认值)代替除美式英语以外的其它语言，可得到更多的兼容性
代码页 932	日本语
代码页 936	简化的汉语
代码页 949	南朝鲜语
代码页 950	传统的汉语
代码页 1250	中欧
代码页 1251	西里尔

续表

代码页 1253	希腊语
代码页 1254	土耳其语
代码页 1255	希伯来语
代码页 1256	阿伯位数字
代码页 1257	波罗的海语

这里有一个叫做 Unicode 数据类型的新特征，即允许列从多个字符集中存储数据。如果进入一种状态，在这里必须从其它语言中向 SQL Server 数据库输入数据，这是很方便的。

11.3.8 排列顺序

在安装过程中，为指定的应用程序做出正确的排列顺序是很重要的。默认的排列顺序是词典排列顺序,忽略大小写。当带有 GROUP BY,ORDER BY 和 DISTINCT 子句的语句使用时，排列顺序指定怎样存储数据。对每一个字符集可从中选择一个不同的排列顺序集。例.如，你能为代码页 1252 的默认字符集从中选择排列顺序，ISO Character Set 是：

- 字典顺序，不区分大小写
- 二进制顺序
- 字典顺序，不区分大小写
- 字典顺序，不区分大小写，大写字母优先

- 字典顺序，不区分大小写，不分重音
- 丹麦／挪威字典顺序，不区分大小写，大写字母优先
- 冰岛字典顺序，不区分大小写，大写字母优先
- 瑞典／芬兰(标准)字典顺序，不区分大小写，大写字母优先
- 瑞典／芬兰(语音)字典顺序，不区分大小写，大写字母优先

11.3.9Unicode 排序

如果在数据库中有 Unicode 数据类型，Unicode 排序指定其决定怎样排序 Unicode 列。这与规则的排序顺序不同。安装程序将提供默认 Unicode 排序，该 Unicode 排序取决于所选择的字符集和排序顺序。如果选择——除默认以外的 Unicode 排序就可知道正在执行什么。你只有选择默认 Unicode 排序，SQL Server 6.x 才能转换成 SQL Server 7，如图 11.6 所示。

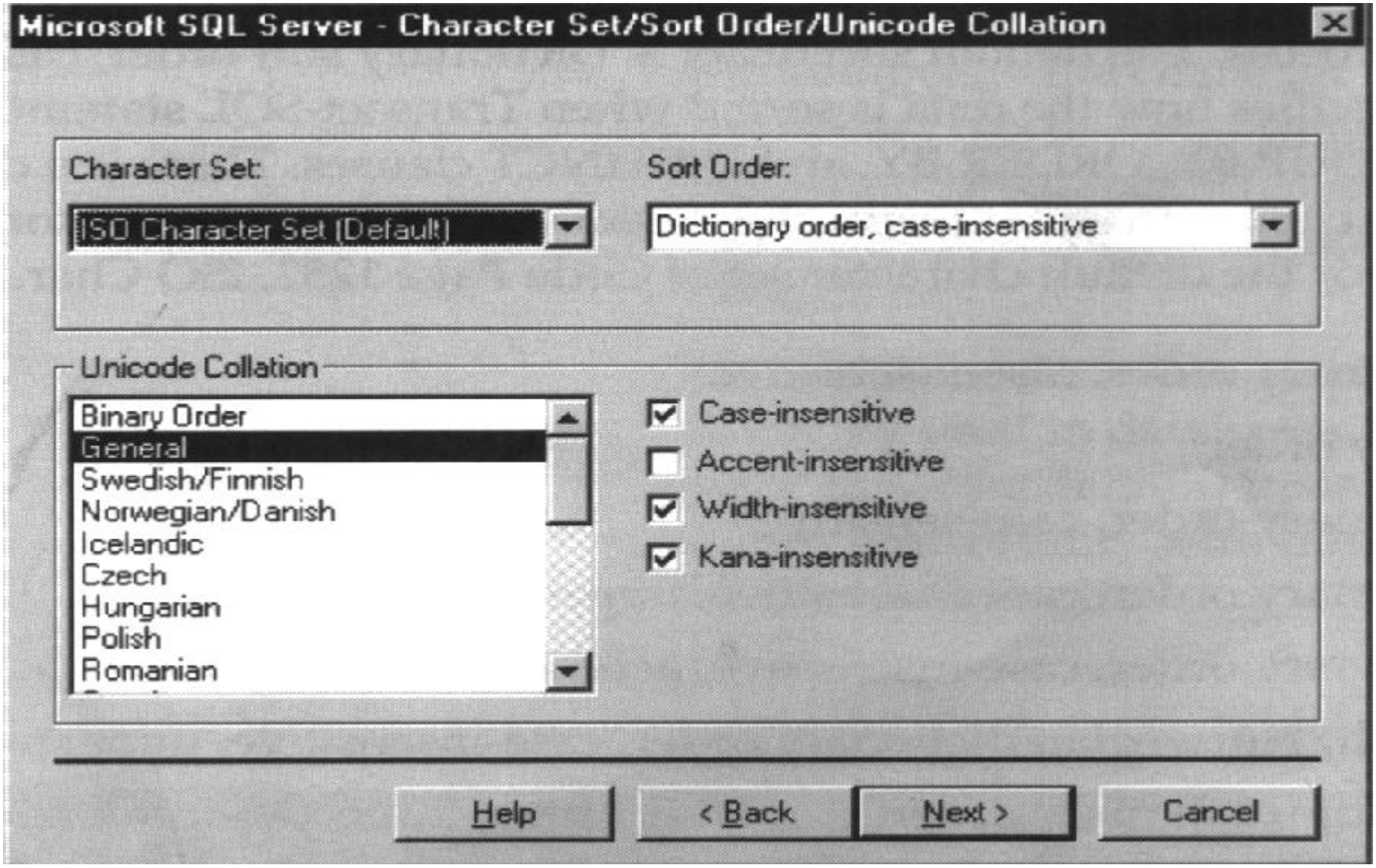

图 11.6SQL Server **安装及** Unicode **排序**

11.3.10 管理工具

有几种实用程序，总是由安装程序安装的：

- BCP
- ISQL
- OSQL

在安装安装程序时，也能安装其它一些可选的实用程序。Select Components 屏幕，给出选择安装服务器组件(服务器升级工具、复制支持或全文搜索)，管理工具(企业管理器，Profiler，查询分析器，DTC 客户支持，复制冲突解决)，客户连接性，在线说明和开发工具(标题和库，虚拟的设备界面)。

11.3.11 验证模式

SQL Server 7 验证模式是：

验证模式	允许的登录类型
Windows NT Authentication	Windows NT logins
Mixed Authentication	SQL Server and Windows NT logins

如果不能肯定使用哪一种，参考第 5 章“与 Windows NT 安全性集成”，可为你的选择作解释。在你已经安装和注册 SQL Server 7 以后，你可以选择。使用企业管理器登录和选择服务器，右击并且选择 Properties，然后选择 Security

标签(参见图 11.7)。

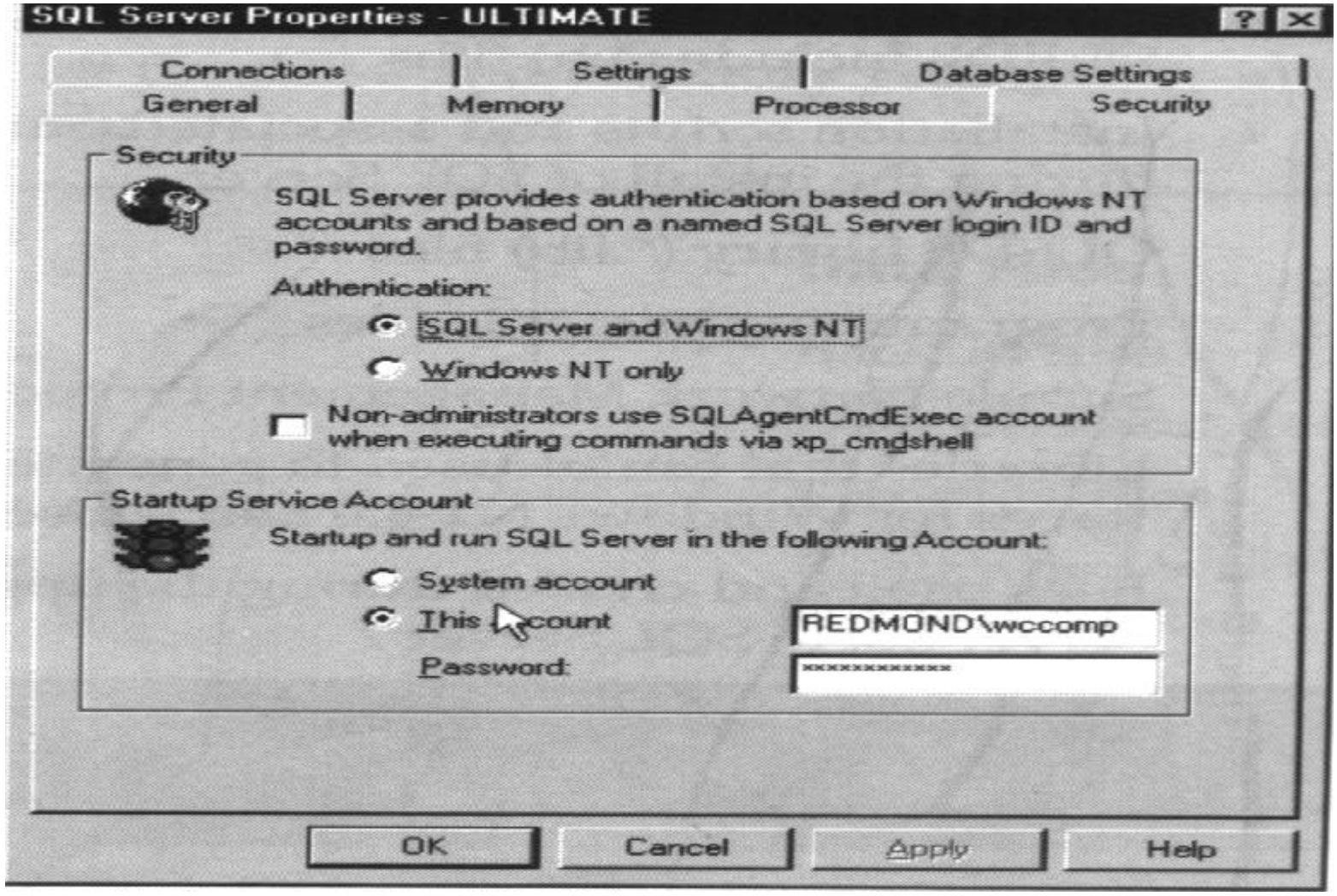

图 11.7 企业管理器及安全性

11.3.12 程序文件的位置

Windows NT 和 Windows 95 安装的默认位置是 C:＼MSSQL7，然而，可以将该安装位置改变到另一个驱动器和目录名。你必须遵守“8 位.3 位”命名常规替换 MSSQL7 的目录名。表 11.5 中所列的子目录就是在安装过程中创建的。在运行安装程序时，在指定的驱动器和目录下包含有该程序文件。

11.3.13 管理工具文件的位置

该默认位置是 C:／MSSQL7 用于 Windows NT 和 Windows 95 的安装。表 11.6 中所列的子目录就是在安装时创建的，在运行安装程序时，在指定的驱动器和目录下包含有实用程序文件。

表 11.5 子目录包含的程序文件

子目录	程序文件
＼Bin	客户可执行的文件，Windows DLL 文件
＼Binn	客户和服务器可执行文件，DLLS 和在线 Help 文件
＼Books	SQL Server Books Online 文件
＼Charsets	字符集和排列顺序文件
＼Include	OLEDB 包括(*.h)文件

续表

＼Install	在 SQL Server 的安装过程中创建的安装脚本和相关的输出文件
＼Lib	OLEDB 库(*.Lib)文件
＼Samples	程序设计例子文件
＼Snmp	简单的 Network Management Protocol(SNMP)文件
＼Symbols	能被用来为 Windows NT 4.0 和随后版本产生堆栈备份和软迹的库
＼Upgrade	在 SQL Server 从 6.x 升级到 7 过程中，使用和创建的文件

表 11.6 包含工具文件的子目录

子目录	实用程序文件
＼Binn	客户可执行文件
＼Books	SQL Server Books Online 文件
＼HTML	HELP 文件
＼Include	OLEDB 包含(*.h)文件
＼Lib	OLEDB 库(*.lib)文件
＼Samples	程序设计例子文件
＼Upgrade	用于升级的文件
＼System32 directory for Windows	DB-Library 和 Net-Library DLLs

11.3.14 数据文件的位置

该默认位置是：C:＼MSSQL 7 用于安装 Windows NT 和 Windows 95，然而可将安装的位置改变到另一个驱动器和目录名。必须遵守“8 位.3 位”的命名规则，为用来代替 MSSQL 7 的目录名命名。表 11.7 所示例子目录就是在安装的时候创建的，在运行安装时，在指定的驱动器和目录下包含有该数据文件。

表 11.7 包含数据文件的子目录

子目录	数据文件
＼Backup	备份文件
＼Data	数据文件和数据库日志文件
＼Log	错误日志文件
＼Repldata	用于复制文件的复制工作目录

11.3.15 自动起动 SQL Server 和 SQL Server Agent 程序

最初的安装程序决定是否想要自动启动 SQL Server 服务和 SQL Server Agent 服务，当 NT 被启动的时候，默认值是该服务不自动启动。如果有一生产系统，可能想要服务自动地开始。这将阻止 SQL Server 在 NT 重新启动时变成不能进入。在安装程序完成之后，可以改变该服务的自动启动性能，如果以后的时间

里，决定不执行自动启动的话。可用 Enterprise Manager 在以后改变这一特性，选择你的服务器，然后右击，从快捷菜单中选择 Properties。

11.4 服务器标准

为在 NT 服务器上安装 SQL Server,建立一套安装标准是一个好的想法。如果你有一个大公司，具有不同规格的服务器，标准化服务器规格并为每种规格建立标准。确定在每一个分区上将有哪种类型的硬件更新。弄明白程序文件和数据文件在哪里，当在不同的机器上安装 SQL Server 时，不超过标准的范围。

11.5 删除 SQL Server

如果需要从计算机中删除 SQL Server，使用程序的 Remove SQL Server 选项。在这个程序中有复选框，能复选将要移去的程序文件和数据文件。可以移去程序文件或数据库；你也可以在命令提示符下发出命令删除 SQL Server。

C:＼Mssql7＼Binn＼Setup／t RemoveAll = WARN

11.6 使用自动化安装的维护安装标准

通过创建为每个安装选项设置的初始化文件来实现自动化安装。可以从命令提示符中启动 Setup.exe，并在另一台机器上运行 Use SQL Server Agent 来在规定的时间执行安装。要创建该文件，安装 SQL Server 作为希望看到的设置选项，从 C:MSSQL7＼INSTALL 目录中寻找 SQLInstS.ini。当需要和更改标准的时候，可以使用该文件作为一出发点并对其进行编辑。对其进行编辑，并输入口令，作为运行 SQL Server 和 SQL Server Agent 程序服务的帐号。仅仅使用本地安装选项，Remote Installation 选项是无法实现自动安装的。要创建一个安装初始化文件,可以使用任何文本编辑器创建一文件,以 C:MSSQL7＼INSTALL 目录下 SQLInstS.ini 相同的格式输入安装信息。以.ini 扩展名保存文件，当没有内部错误检查时，确保其有正确的值。也能创建一定制脚本在安装期间运行它。路径／文件名必须在安装初始化文件时引用。运行一自动化安装的命令行语法是：

```
Setup／t IniFilePath=setupinitializationfilename
```

安装初始化文件在使用前后都必须进行编辑,因为它包含有 SQLServerAgent 服务启动帐号的口令，并且不想这个信息进入操作系统文件。如果运行下列命令行语句，将看见安装参数的说明：

```
Setup ／?
```

11.7 远 程 安 装

当安装 SQL Server 时，你可以本地安装，也可以远程安装。如果想将 SQL Server 安装在一台机器上而不是正在运行安装程序的计算机上，选择 Remote 选项，必须能在网络上到达远程机器并能输入远程机器的名称以及安装 SQL Server 的驱动器盘。

第 12 章安装 SQL Mail

SQL Mail 是使用邮件应用程序界面的邮件服务，该应用程序界面能够使用 SQL Server 收发电子邮件消息。SQL Mail 使用扩展的存储过程，该过程是由 Microsoft SQL Server 安装程序带来的。

12.1Windows NT 消息服务

该 NT 服务器具有内在的消息服务功能，与 SQL Server 一起工作并能收发电子邮件。SQL Mail 常常与 Microsoft Outlook、Microsoft Exchange、Microsoft Windows NT Mail 或其它 MAPI 一道用来收发电子邮件。当安装 Windows NT 时，也许没有安装 NT 消息。如果原先没有安装的话，当需要安装 SQL Mail 时，必须首先安装 NT Messaging Service 步骤是：

1. 以在本地机上正在运行 SQL Server 服务的帐号登录 NT。

2. 在 Inbox 图标上双击，消息出现，问是否安装 Windows NT Messaging，这表示是否安装。

3. 需要访问 NT 安装 CD ROM 驱动器和 I386 目录。如果系统提示改写新文

件，回答否定。

为了使 SQL Mail 工作，MAPI 界面需要在 Windows NT 服务器上安装一客户邮件程序。Microsoft Outlook 或 Microsoft Exchange 与 SQL Mail 一道工作并能安装在 Widnows NT 服务器上创建一邮件客户。

12.2SQL Mail 的安装

为了使 SQL Mail 在 Microsoft SQL Server 上成功地运行，必须在服务器上安装一客户邮件应用程序。另外，位于另一个服务器上的 Mail 邮局必须是可用的。最后必须在 SQL Server 启动帐号下，启动 Mail 客户。这意味着 SQL Server 必须设置使用邮件帐号，该帐号既与 MSSQL Server 启动帐号相同又是一个 MSSQL Server 启动帐号所拥有的 Exchange 邮箱。

12.3SQL Mail 先决条件

在开始配置和安装 SQL Mail 之前，了解 SQL Mail 的先决条件是有用的。

条件 1： 一个安装在 Microsoft SQL Server 计算机上的邮件客户程序。

条件 2： 一个 SQL Server 启动帐号所具有的 Exchange 邮箱。

条件 3： 一个具有 SQL Server 服务启动帐号功能的 NT 帐号，该帐号拥有

交换邮箱。

条件 4： Exchange 邮件服务器的名称。

12.3.1NT 帐号

将需要一个 NT 帐号，该帐号拥有 Exchange 交换邮箱并能被用作 SQL Server 启动帐号。在输入下列命令提示符后可以运行服务管理器来观察 SQL Server 启动帐号：

srvmgr＼＼servername

当服务管理器启动时，选择 Computer／Services 并且从如图 12.1 服务列表中选择 MSSQL Server 服务。按下 Starup 按钮可看见如图 12.2 所示的用来启动 MSSQL Server 的帐号。使该 Windows NT 帐号成为邮箱的所有者。

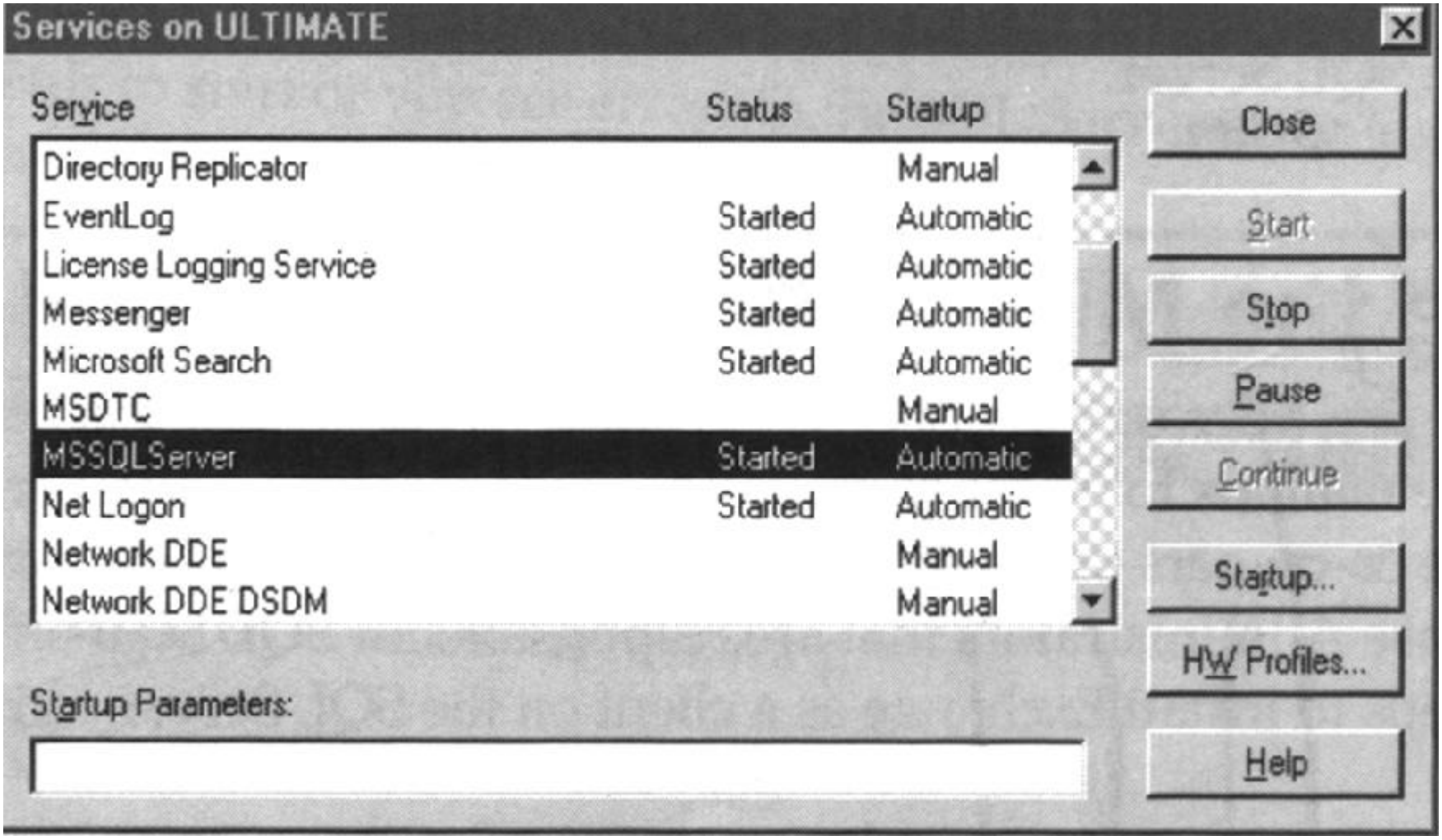
Services on ULTIMATE
Service
Status
Startup
Directory Replicator
Manual
EventLog
Started
Automatic
License Logging Service
Started
Automatic
Messenger
Started
Automatic
Microsoft Search
Started
Automatic
MSDTC
Manual
MSSQLServer
Started
Automatic
Net Logon
Started
Automatic
Network DDE
Manual
Network DDE DSDM
Manual
Startup Parameters:
Close
Start
Stop
Pause
Continue
Startup...
HW Profiles...
Help

图 12.1 服务列表

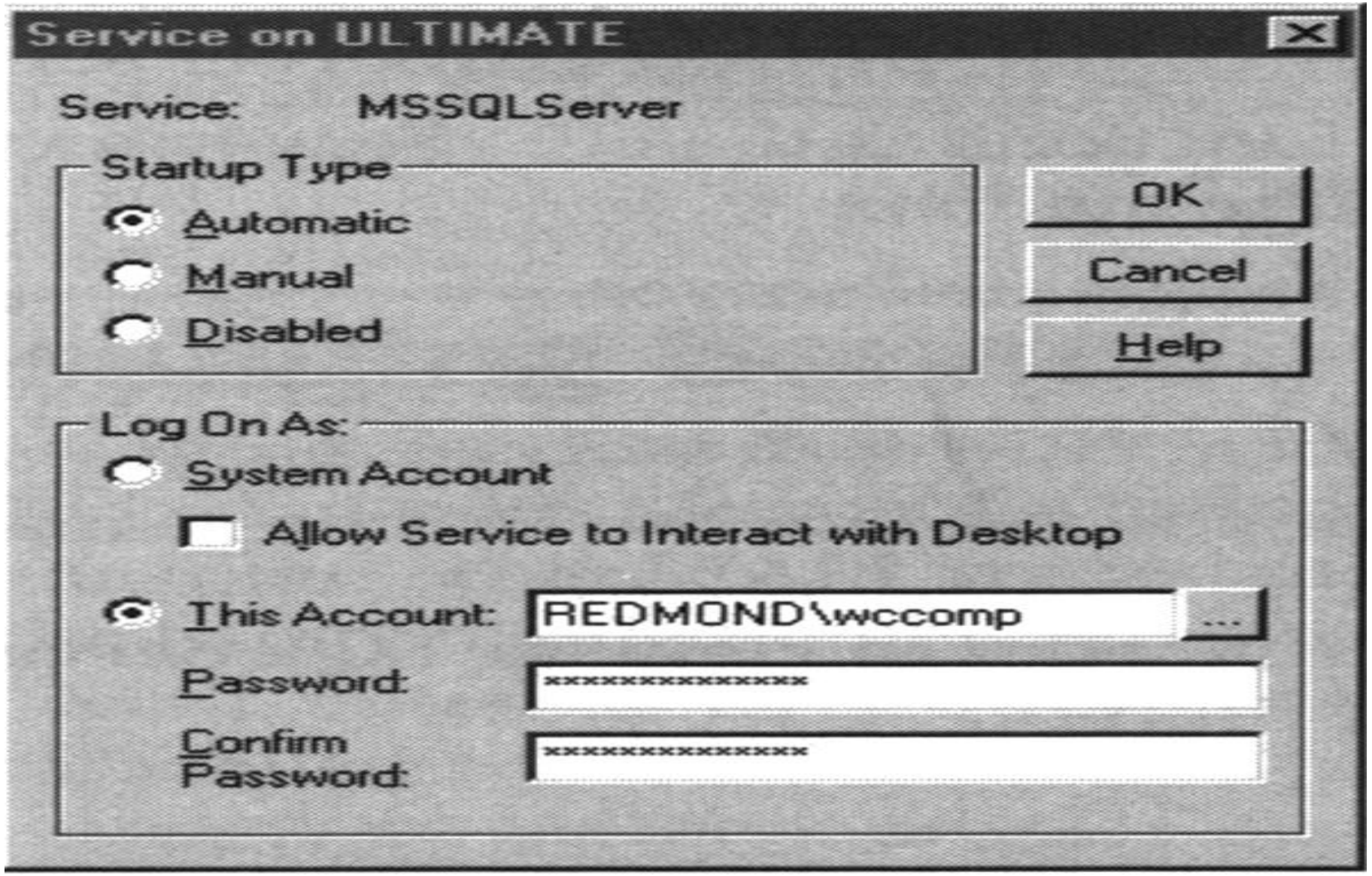

图 12.2MSSQLServer **启动** Windows NT **帐号**

12.3.2Exchange 邮箱

让我们在 SQL Server 计算机上使用 Microsoft Exchange 邮件程序并由邮件管理员程序启动，使邮箱所有者成为能启动 SQL Server 的相同的 NT 帐号。对邮箱的共有者来说，这是可能的。邮箱的用户也能被设置成在 SQL Server 上决定 SQL Mail 的数据库管理器。

12.4 安装 Microsoft Exchange 邮件客户

如果选择使用 Microsoft Exchange 作为邮件的客户，将可能想要邮件的拥有者是启动 SQL Server 的相同的 NT 帐号(它对有邮箱的共有者是可能的并且邮箱的用户能被设置是数据库管理员)，以下是在 SQL Server 计算机上安装作为客户的 Exchange 的步骤：

1. 在本地 SQL Server 计算机上登录 NT 作为启动 SQL Server 服务的帐号。
2. 通过运行邮件程序 setup.exe，在 SQL Server 上安装 Exchange Mail Client。
3. 选择典型安装模式并按下该按钮。
4. 当问到名称时，输入该 Exchange 邮箱的名称。该邮箱由 SQL Server 服务所拥有，启动 Windows NT Acoount。

5．完成 Exchange 客户安装。

6．开始双击该图标，该图标是由邮件客户的安装程序在该台面上创立的或者是从 NT 菜单中选择的。

7．删除非 Microsoft Exchange Server 的任何服务。

8．继续通过该屏幕在远程使用(选择不)和个人地址。

9．当提示在 NT 启动时允许自动启动 Exchange 时选择 Do Not Add。

10．完成安装。

11．开始 Exchange。选择 Tools／Options 和 Send 标签。清除 Sent Items Folder 复选框中该
项目的拷贝副本。这可以使你避免在文件夹满时不得不清除该收送项目。

12．回到 Inbox 并将邮件送去校验安装。

13．选择 File／Exit 并注销该 Exchange 客户程序。

14．使用企业管理器设置属性。

15．检查 Auto Start Mail Client 复选框；在图 12.3 所示的 Profile 名区域输入邮箱名称。

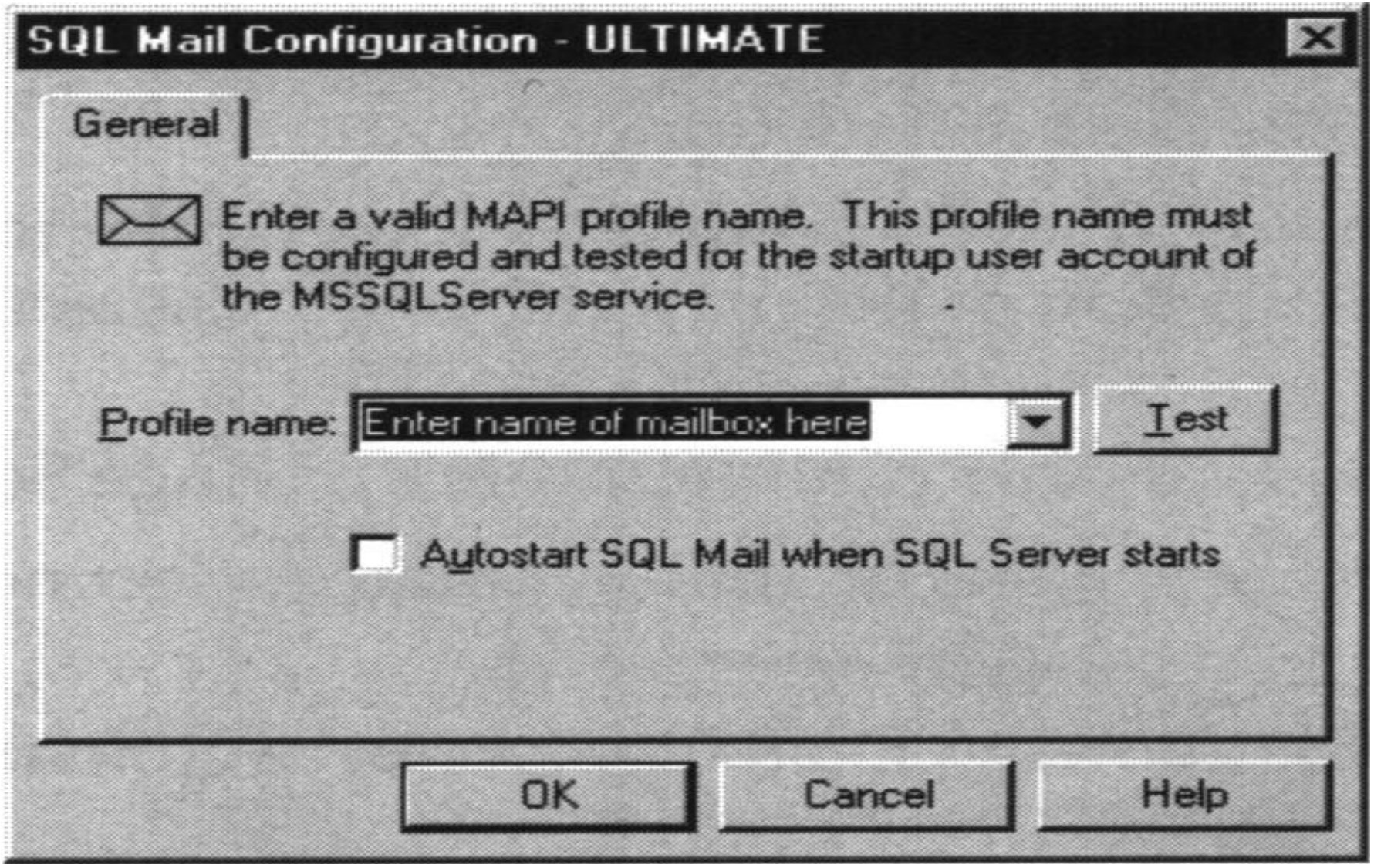

图 12.3SQL Mail 属性窗口

16. 停机，然后重新启动 SQL Server。

17. 在启动 SQL Mail时，检查 SQL Server的错误日志。

如果 SQL Server 启动失败，或者在 SQL Server 错误日志上发现错误，下一节介绍如何解决这个问题。

12.5 解决 SQL Mail 启动失败的技巧

如果 SQL Server没有启动，核对下列各项：

1. 在 SQL Server错误日志中的 MAPI 错误。

(1) SQL Server是否运行在拥有邮箱的 NT 帐号下？

(2) SQL Server 邮件登录的安装程序对 SQL Server 和 SQL Server Agent 是正确的吗？

(3) 运行在 SQL Server上交换是在 SQL Server服务启动帐号下启动的吗？

2. 如果 SQL Mail停止应答，循环 SQL Server。

3. 通过在桌面上双击 inbox 图标，保证 Mail Messaging 服务程序是安装在 Widnows NT 上。如果问是否想安装该服务程序，说明 Windows NT Messaging 是没有安装。因此，着手进行安装。

12.6SQL Mail 扩展存储过程

扩展存储过程是被用在 SQL Mail 包含 XP 启动邮件，XP 终止邮件，XP 发送邮件，XP 读邮件，XP 分理邮件，XP 查找下一个信息和 XP 删除邮件。只有系统管理员能够执行 SQL Mail 扩展存储过程；然而系统管理员允许其他用户执行扩展存储过程。关于以上所列的 SQL Mail 扩展存储过程的使用的更多的信息参阅第 23 章“系统存储过程”。

12.6.1SQL Mail 的启动和终止

有两种不同的方法来启动和终止 SQL Mail。可以使用企业管理器或执行为这个目的提供的 SQL Server 的扩展存储过程。也能在企业管理器 Tools 菜单中选择 Query Analyzer 并在 Query Analyzer 窗口中使用 XP 启动邮件或 XP 终止邮件扩展存储过程。

在 Query Analyzer 窗口中，下列命令可以启动 SQL Mail：

```
xp_startmail
GO
```

命令在 Query Analyzer 窗口中可以终止 SQL Mail。

```
xp_stopmail
GO
```

使用企业管理器，扩展该服务器并且在 SQL Mail 上右击快捷菜单将允许选择启动或终止 SQL Mail。

12.6.2 从 SQL Server 中发送邮件

可以从 SQL Mail 中发送一条简单的消息、一查询结果集或一个附件。使用位于企业管理器的 Tools 菜单中的 Query Analyzer 窗口中发出下列命令：

```
xp_sendmail email_name,' Any message you wish to send '
```

Mail Sent 消息将出现，并在电子邮件名称输入盒中出现该电子邮件。
关于更多的有关使用 SQL Mail 扩展存储过程的信息参阅第 23 章“系统存储过程”。SQL Mail 也具有邮寄查询结果的能力。

12.7SQL Mail 和 SQL Server Agent

当发生报警时或当一预定任务完成或失败时，如果想使用 SQL Mail 与 SQL Server Agent 一起工作，以便通过电子邮件通知数据库管理员，必须为 SQL Server Agent 设置 SQL 邮件的配置。为了完成这个任务，在企业管理器中扩展该服务器，并选择 SQL Server Agent。在 SQL Server 上右击，然后选择 Properties(参见图 12.4)。在 General 标签中选择 SQL Mail 按钮，输入 NT 帐号名称，该名称与 MSSQL Server 服务启动帐号相同。如果正在使用带有 SQL Server 的 SQL Mail，你可以设置 SQL Mail 自动启动，每当 SQL Server Agent 服务被启动的时候——甚至当 SQL Server Agent 停止，然后重新启动时 SQL Mail

便自动启动。消息能被发送到扩展存储过程能被调用的任何地方，例如：从一个触发器或存储过程调用，或从脚本文件以内一次调用。当性能阈值达到的时候，SQL Mail也能与SQL Server Profiler一起用于发送邮件。

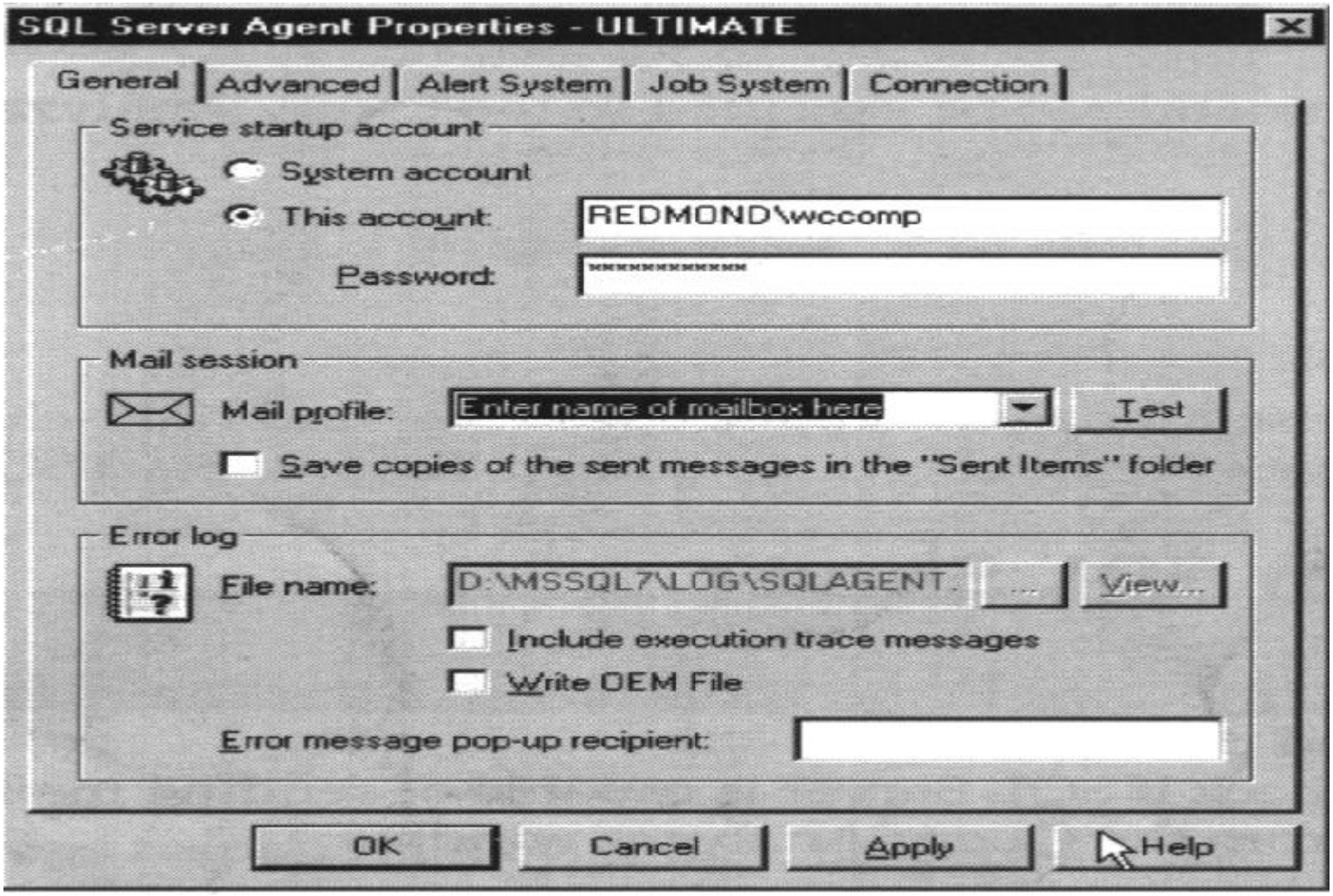

图 12.4SQL Server Agent在企业管理器的属性

该 SQL Server Agent(先前所知的 SQL Executive)已经被提高到能充分利用安装在 SQL Server 上的 SQL Mail 了。参阅第 4 章“SQL Server Agent”可以获得关于如何同时使用 SQL ServerAgent 和 SQL Mail 的详细数据，在一报警或作业失败的事件中通知用户。

第 13 章升级 SQL Server

有人可能认为，升级到 Microsoft SQL Server 7 一定是个复杂而困难的过程。其实，开发应用程序并在 7.0 版上运行是个很简单的过程，看上去及运行过程与在以前版上几乎相同，因为在缺省设置时与用户当前的以前版本是兼容的。然而，新版本需要的磁盘空间要大一倍，以便继续保留 6.x 版本。这样可在多出的空间进行开发和充分利用 7.0 版的新功能。

升级后也可以把兼容级别改变到 60(V6.0)、65(V6.5)或 70(V7.0)的功能，仍可获得 7.0 版的功能，例如，可获得最低成本的锁定策略(运行时根据从 Query Optimizer 的输入的行、页或表)。需要注意的是，在升级后，model 系统数据库的兼容级别为 70。这就是说，如果不执行 sp_dbcmptlevel 数据库存储过程，并把 model 系统数据库的兼容级别设置成 65，用 7.0 版本也可创建所有新的数据库。

SQL Server 的升级过程的主要特性之一，是两种版本可并行存放在一台计算机上。升级过程的每一步都设计成保护 6.x 数据库。然而，安装 Microsoft SQL Server 7 将替换原系统中的某些通用库文件，而卸载 7.0 版本时又不能自动返回到 SQL Server 6.x 或 Windows NT Server 安装时的原始状态。

安装时可能遇到问题的是这类应用程序，它们是直接参照系统表的应用程

序和使用 SQL DMO 的管理类工具应用程序。由于 SQL DMO 的原因，SQL Executive 中的任务、复制和许多配置设置都发生了变化。如果用户的应用程序正在动态地改变配置设置，或正在使用这些特性，可能需要改变自己的代码。在 master 系统数据库中创建的用户对象可保证 7.0 版的性能，因为 master 系统数据库必须在功能上正确地保持 SQL Server 7.0 的兼容级别上。在该 master 系统数据库中不能改变这种兼容级别。

Microsoft 公司对这种转换过程经过长期艰苦的思考后，决定必须改变这种体系结构，以便在质量、性能、可量测性和灵活性方面，适应下一个 20 年，所以用户数据库从 V6.x 到 7 的转换需要改变磁盘。用户的对象可以进行编译，数据也可以从 6.x 的磁盘格式移到 7 磁盘格式，因为该数据页的尺寸已经从 6.x 版的 2k 改为 7.0 版的 8k。

13.1 何 时 升 级

最好是一得到新的服务软件包和新软本就尽快升级。如果用户有经销商以外开发的应用程序，并且该经销商迟迟才转向新服务软件包或新版本，那么升级时往往会出现问题。应鼓励经销商及时升级。如果用户认为不能升级自己的生产系统，那么可升级自己的测试系统，甚至可升级台式计算机，这样自己可保持处在新的技术前沿。Microsoft SQL Server 数据库技术正在飞速发展，这些改进正是 Microsoft SQL Server 所要进行的，这样可激励产品年复一年地使

用下去，但是，如果用户在面临版本的全面升级，现时就有许多东西要学习。

13.2 升 级 计 划

升级计划过程中最重要的一点就是不要感到惊讶。要计划好培训时间和练习地点，以使自己熟悉升级过程，确定自己的应用程序是否存在问题，如参照现在还没有表的存储过程、在 7 版中改作了改变的代码参照系统表和对自己的应用程序非常特别的其他任何问题。自己要准备足够的时间来辨别和确定可能会发生的任何问题。虽然大多数应用程序至今还未发现任何问题，但是还是可与 Microsoft 技术支持联系以解决任何问题。Microsoft 在其 1K Challenge 转换程序中已经转换了 1000 多个数据库，并已为可能出现的许多问题做了预测和编码。SQL DMO、SQL Executive 中的任务、复制和许多配置设置都已做了改变，所以最好是建立测试系统，如有可能，可在非生产性机器上通过升级来运行。但对于大型系统而言因为硬件冲突，这可能无法实现。在这种情况下，任何时候都可以用原来的 6.x 环境放弃这一过程。实际上，甚至可在过程中间也可以停下来，并在断开的位置重新启动该过程。

在开始升级之前用户可能想要做以下一些事情:。

- 备份 SQL Server 6.x 环境。
- 保证 tempdb 系统数据库在 6.x 版中至少为 25MB；
- 保证 SQL Server 6.x 版具有足够的内存以有效地运行 DBCC，这样可加

速这一过程(7版的内存大小是动态的)；

· 保证在自己的用户数据库中的所有数据库用户都在 master 系统数据库中登录 SQL Server，并更正“孤立”的任何问题(sysusers 系统表，与 suid 连接的 syslogins 系统表)。还需检查 sysalternates 系统表输入项。

· 保证 MSSQLServer 服务启动帐号是 Domain 用户帐号，而不是供两种 SQL Server 版本共享的 LocalSystem 帐号。

· 保证 SQLExecutive 和 SQLServerAgent 服务启动帐号是域用户帐号，而不是 LocalSystem 帐号。

· 保证至少有 200MB 的自由磁盘空间，以执行 Microsoft SQL Server 7 的安装和升级，并有足够的自由空间拷贝升级选择数据库中的对象和数据。详细情况请参阅本章“并行(单机)”和“机到机(双机)”部分。

· 如果用户有最新版的 6.5 Service Pack 软件包的拷贝，应安装最新版，不要安装拷贝，并使 logreader 保持空。

· 保证在最小模式下运行 Windows NT Server 4.0，Service Pack 4。

· 如果自己的数据库正在相互参照对象，可同时升级所有相关数据库。

· 保证对每个登录的 ID，随着正在升级的数据库或在 7.0 版中已经有的缺省数据库，来升级缺省数据库。

· 在开始升级过程之前，关闭所有应用程序，停止运行 SQL Server 服务程序。

13.3 升 级 之 前

在升级之前，应先阅读下面这部分内容，这些内容可应用到当前正在运行的版本上。

13.3.1 从 4.2.1 版升级

如果需要从 Microsoft SQL Server 4.2.1 版升级，在升级到 7.0 版之前必须首先升级到 6.5 版和最新的服务包软件。也可能需要把 NT Server 4.0 升级到可得到的最新的服务包软件。

13.3.2 从 6.0 版升级

最好是先把 Microsoft SQL Server 6.0 安装程序升级到 Microsoft SQL Server 6.5 和最新的服务包。然而，也可能从 Microsoft SQL Server 6.0 直接升级到 7.0。用 6.0 中可正常工作的 GRANT／REVOKES 方法可能会发生异常。正如用户所知，7.0 版中有一个 sp_dbcmptlever 60 选项，目的就是保持与 6.0 版的兼容性。

13.3.3 从 6.5 版升级

在从 Microsoft SQL Server 6.5 升级到 7.0 版时，在开始升级之前必须应用最新的 SQL Server 服务包。

13.4 版本升级实用程序

采用 Version Upgrade Utility 实用程序，即可进行并行升级，又可进行双机升级。Utility 实用程序是 SQL Server 7 的安装选项，可在 NT Menu(菜单)上的 Microsoft SQL Server Program 程序组中进行选择。这种类型的实用程序通常是不提供的，但由于大约 80%的 SQL Server 是为 7 版编写的，并且许多关键组件已经改变，所以该实用程序对于执行把 6.5 数据库转换到 7.0 数据库这种复杂过程，是十分必要的。Version Upgrade Utility 非常有用，特别是考虑到整个数据库以及所有相关数据都必须写成 Microsoft SQL Server 7 的磁盘格式。

所需要的磁盘空间量影响将要使用的升级情况。另外，如果使用复制，就必须使用并行升级。无论什么情况，从 6.x 转换到 7.0 版，都至少需要两倍当前数据库所占的空间。许多公司的预算都将受到这种磁盘空缩需求的影响，除非对数据库技术早已进行了培训，在培训时就希望过升级相关的空间需要。

在升级时，用户可选择升级选项，如图 13.1 所示。Export 和 Import 选项，以及数据转换方法等都可以选择。

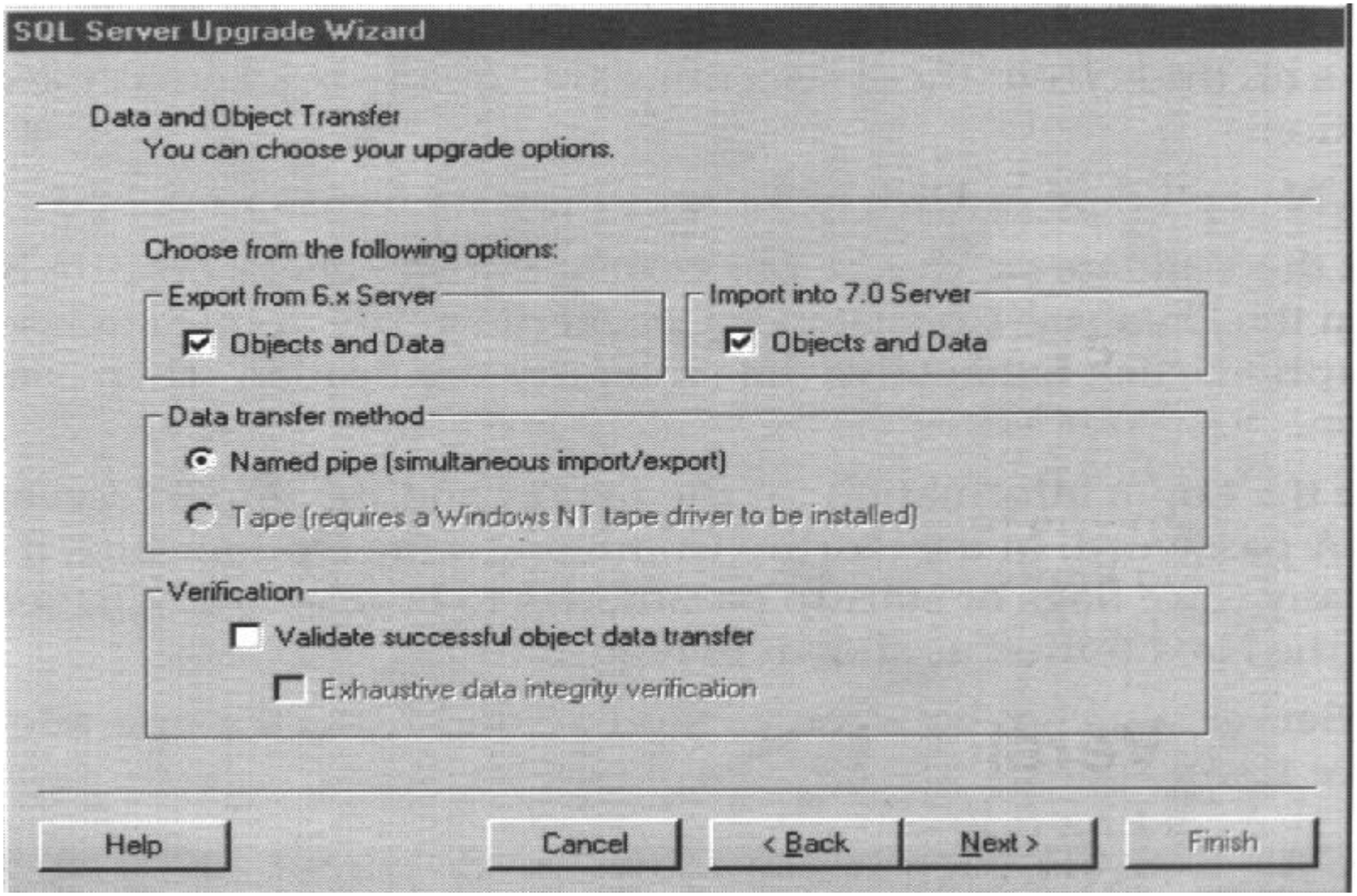

图 13.1SQL Server Upgrade Wizard | Data and Object Transfer

13.4.1 并行升级(单机)

并行升级是指在已经有 Microsoft SQL Server 6.x 版的计算机上安装 Microsoft SQL Server 7.0 版。数据传输方法为下列方法之一:

- 磁盘对磁盘命名的管道连接
- 磁带驱动器备份
- 网络共享

磁盘对磁盘命令管道连接是最快的方法,可用于计算机上磁盘空间足够的情况,可直接通过命名管道连接传输数据。如果要使用磁盘对磁盘命名管道连接,则可为该数据库和相关日志文件准备 1.5 倍的磁盘空间大小。

如果磁盘空间不足,对于磁带驱动器或网络共享可使用 Version Upgarde Wizard(版本升级向导)输出 6.x 版日志(Catalog)、数据、对象和数据库。Version Upgrade Wizard 删去 SQL Server 6.x 驱动程序,重新使用磁盘空间。然后再次运行 Version Upgrade Wizard,输入并升级从磁带驱动器或网络共享输出的数据、对象和数据库。

进行并行升级时应按以下步骤执行:

1. 从 SQL Server 7 程序组中选择 Version Upgrade Wizard。
2. 在向导屏幕上单击 Next 按钮。
3. 从屏幕上的 Export From 6.x 和 Import To 7.0 块上选择 Objects 和 Data 复选框(这些选择已标记为缺省选择)。
4. 单击 Named Pipe 或 Disk 或 Tape 作为数据传输方法。

5. 单击 Validate successful object data transfer 复选框。如果感到需要运行 Database Consistency Checker，以验证数据库没有任何损坏，可单击 Exhaustive data integrity verification 复选框，然后再单击 Next 按钮。

6. 输入 Export 和 Import 服务器的 Login 信息，包括 SA 口令。在 Startup Command-Line Options 框中，如果适用，可键入用于 SQL Server 6.x 的任何跟踪标志或启动参数，然后单击 Next 按钮，如图 13.2 所示。

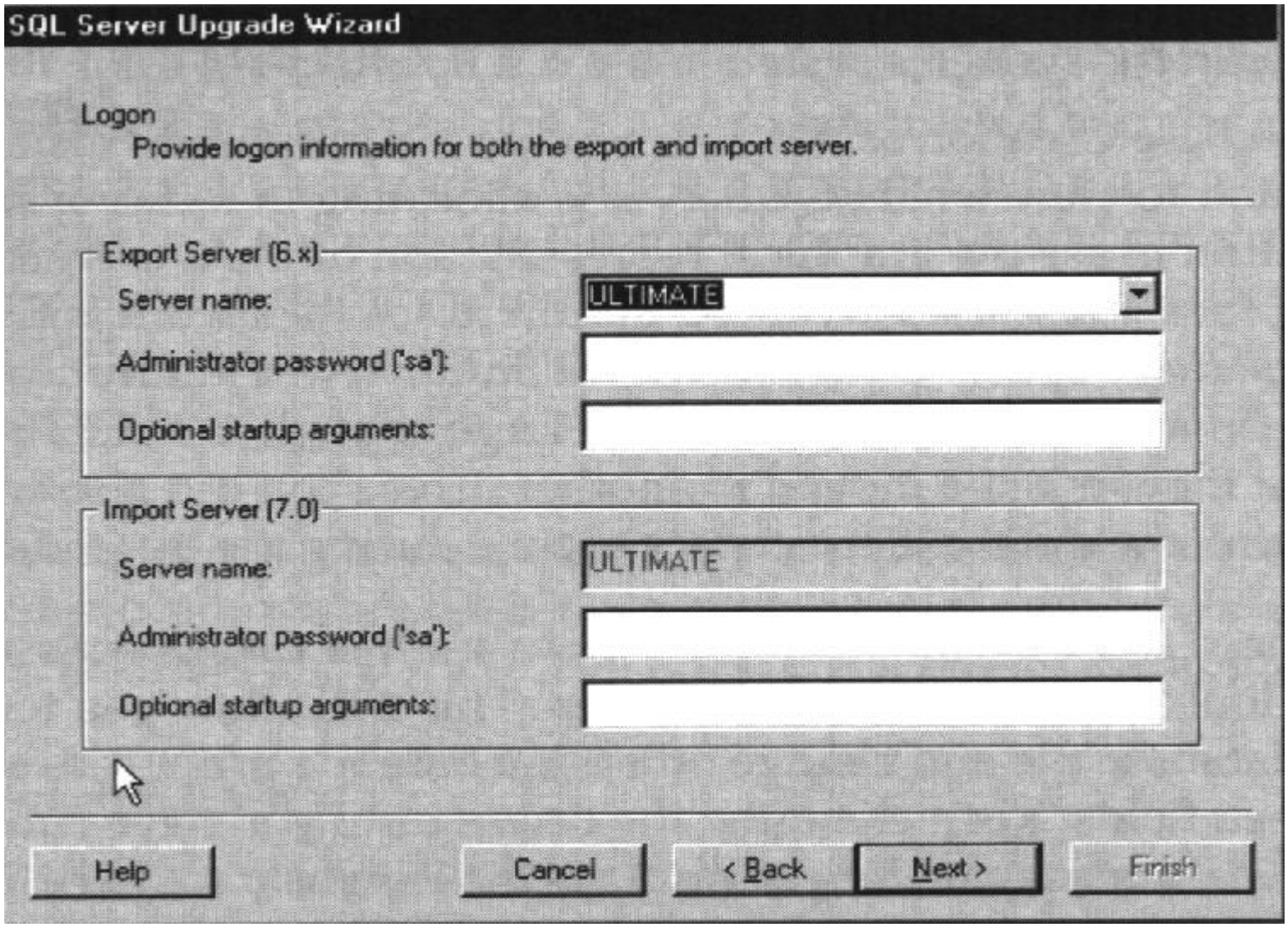

图 13.2 为输入和输出服务器提供登录信息

7. 6.5 版和 7.0 版之间的 SQL Server 开关(切换)可作为升级的继续(如图 13.3 所示)

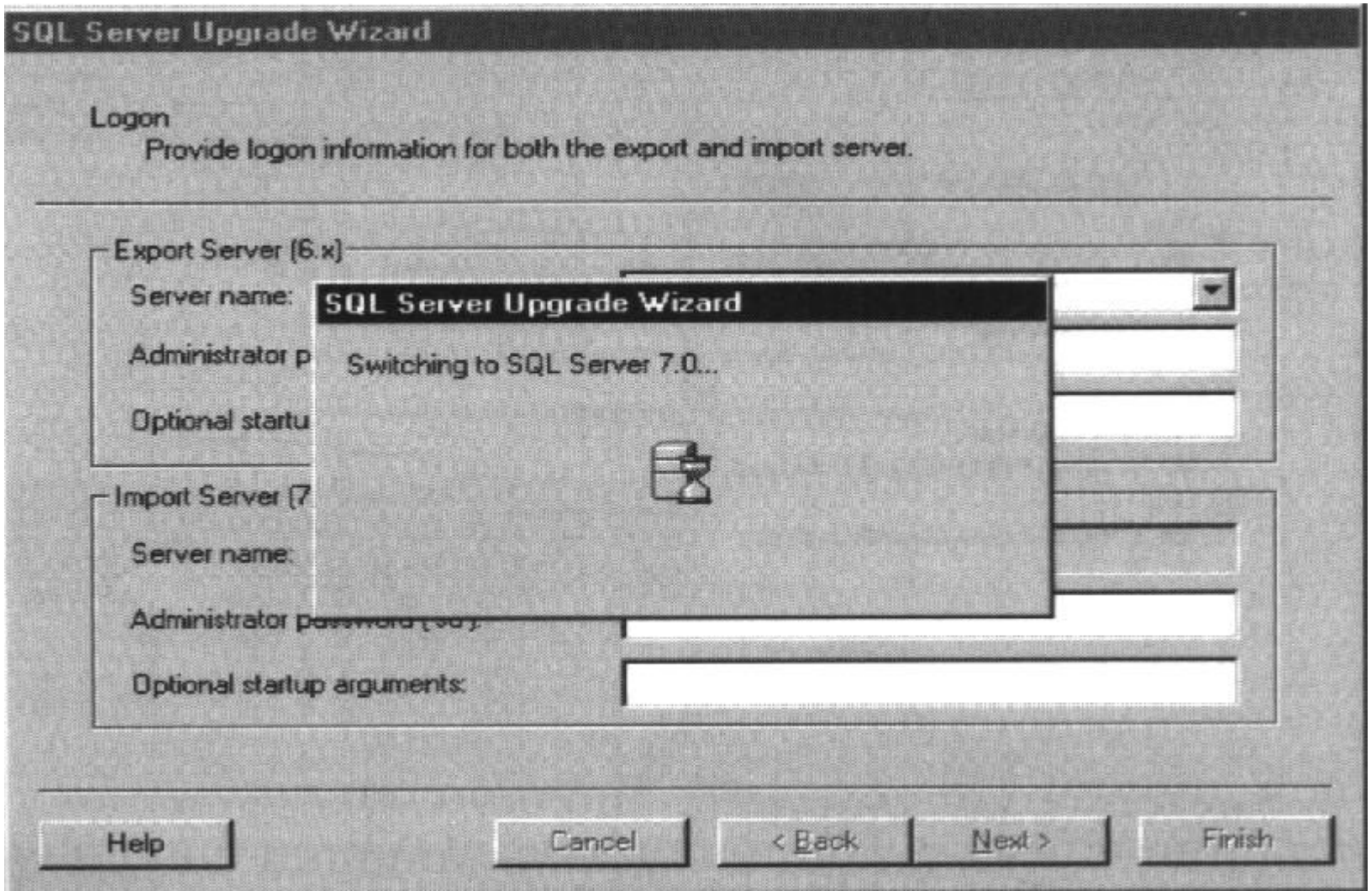

图 13.3 升级过程中两个版本之间的 SQL Server 切换(开关)

8. Code Page Selection 屏幕显示提示输入代码页(用于某些国际用户),但大多数安装都可采用缺省安装。

9. 排除用户不想升级到 7.0 版的数据库(注意:如有可能，最好还是一次将它们升级。否则，syslogins 系统表中的登录的缺省数据库可能受到影响)。然后单击 Next 按钮。

10. Database Creation 屏幕显示推荐的 7.0 版数据库的布局，并可观察 SQL Server 在空间各项目中分配的内容。按下 Edit 按钮查看 7.0 新版数据库的图形说明和记录文件。按下 Advanced 按钮可查看可用驱动器空间的布局和摘要信息。在图 13.4 所示的 Drive Summary 下，靠近驱动器 C:有一个图标，指示要容纳所有数据库文件空间不够大。必在升级进程中还可以查看 SQL Server 将减小数据库以节省空间。

11. 如果因为空间不足需要把数据库文件移到另一个驱动器，可双击相应的数据库文件。这样系统允许编辑数据库文件的安装位置，通过编辑该屏幕区段来改变其他数据库文件的特性。一直保持改变驱动器，直到空间足够，Drive Summary 不再显示 C:旁边的图标。

12. 在为数据库文件确定了最佳位置后，按下 Accept 按钮。

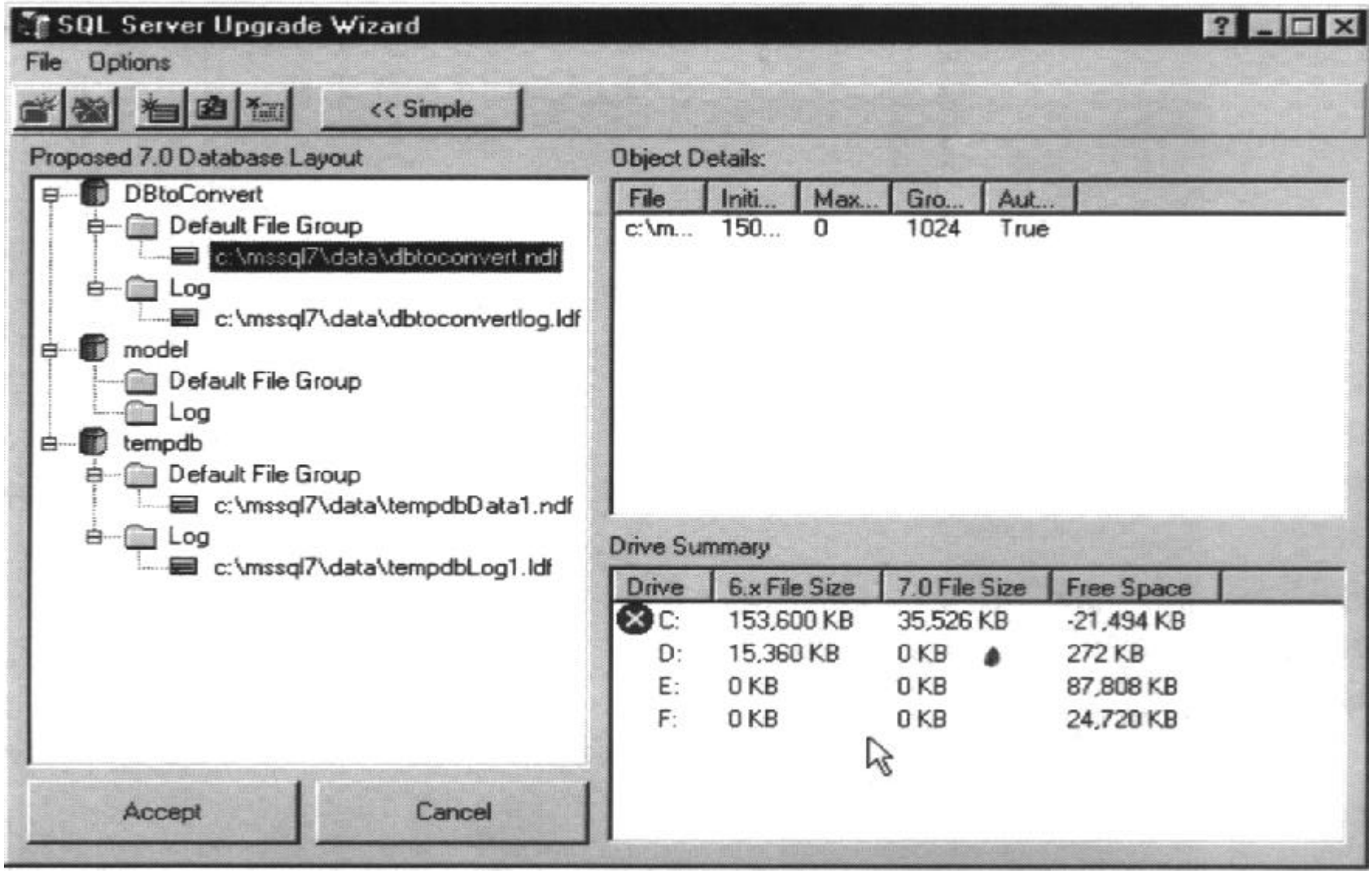

图 13.4Version Upgrade Utility | Database Creation | Edit | Advanced | Drive Summary **选项**

13. 选择自己想要传输的 System Objects 类型。

14. 对于 Quoted Identifier，如果不能确定用户的对象是用 QUOTED_IDENTIFIERS OFF 还是用 ON 创建的，可选择 Mixed。注意如果因为升级期间 Quoted_Identifiers 出现错误而导致用户对象创建失败，在升级后可正确设置选项重新执行适当的文件(在升级目录中执行.prc、.viw 和其他文件)。检验 C:＼MSSQL7＼Upgrade 目录中的文件(或安装 SQL Server 7 的其他目录中的文件)查看升级过程中由 SQL Server 执行过的 Transact-SQL。

15. 对于 ANSI Nulls，如果在创建存储过程时未使用 ANSI 空位，可选择缺省设置 OFF，如果使用了 ANSI 空位，则选择 ON。在升级前最后屏幕显示如图 13.5 所示。

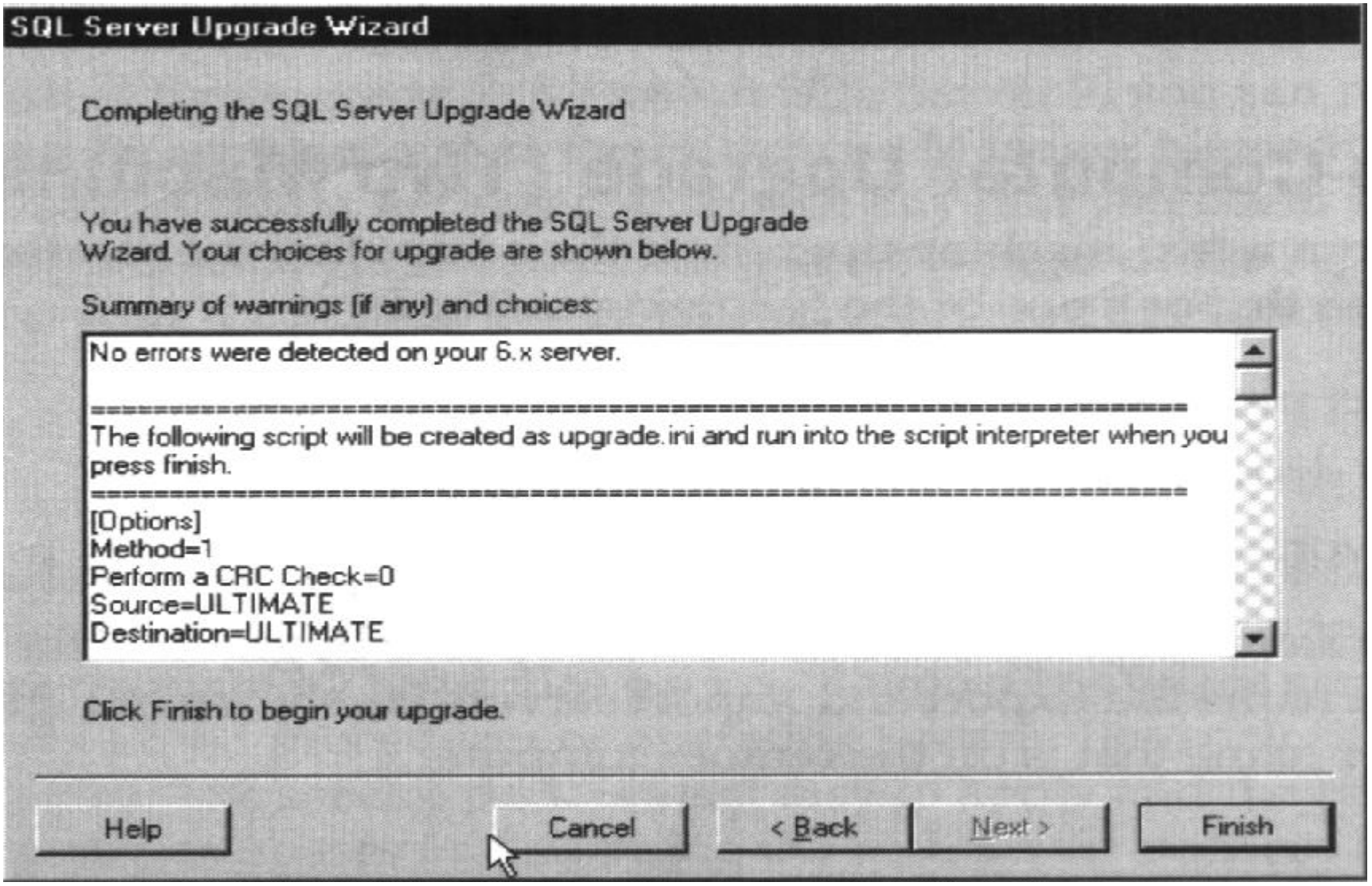

图 13.5 升级开始前 SQL Server Upgrade Wizard 的最终屏幕显示

16. 单击 Finish 按钮。

17. 升级开始，随着升级到 Microsoft SQL Server 7，屏幕上显示进程报告(如图 13.6 所示)。

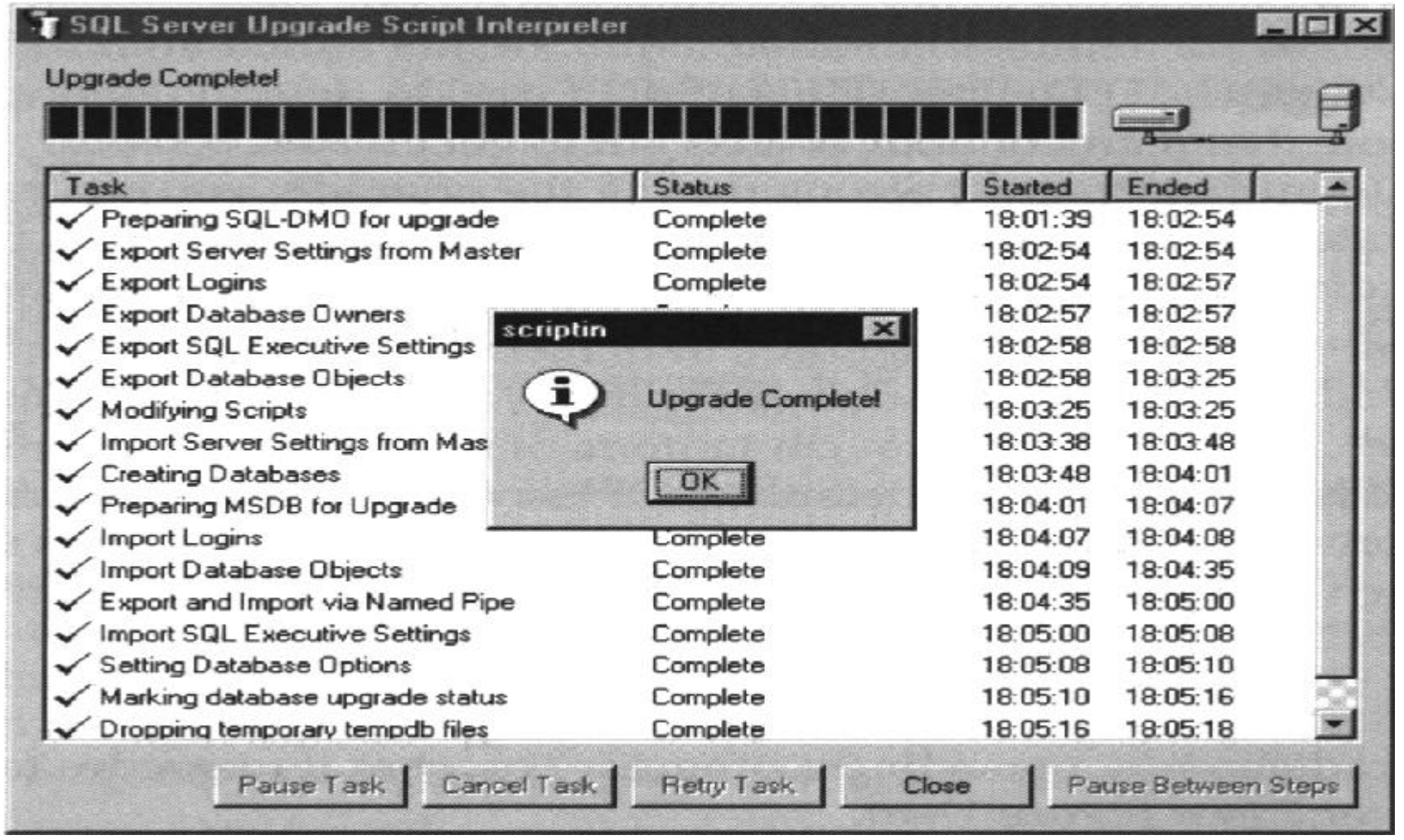

图 13.6Version Upgrade Utility|Upgrade Completion 选项

13.4.2 计算机对计算机升级(双机)

如果计算机通过网络相连，可以实施双机升级。数据移动方法为下列方法之一：

- 直接管道线
- 磁带驱动器备份
- 网络共享

升级的步骤与前一节中并行升级介绍的步骤相同，只是要命名 Export 和 Import 服务器时(如图 13.2 所示)，应指明位于远程计算机上的 SQL Server 名称。

13.5 复制和升级

用户将能够升级自己的复制服务器，甚至服务器版本级别不同也可运行复制。然而，如果用户的复制拓扑包括 6.x 和 7.0 SQL Server，以便复制服务器能够在安装在相应版本上，那么必须应用某些规则。

- 如果正在从 7.0 服务器发布到 6.x Subscriber，就必须运行脚本(C:\MSSQL7\Install\Replp70.sql)，然后在 6.x Subscriber 上执行该脚本安装的存储过程 sp_addpublisher70。

· 在升级 Publisher 之前，必须先升级 distribution 数据库。如果 7.0 分布器和 6.5 Publisher 位于不同的计算机上，就可以使用它们。

· 可以把订阅物从 7.0 Publisher 扩展到(push) SQL Server 6.x 上，但不能卸除它们。

· 7.0 安装可以在不同代码页的服务器之间进行复制，但在 6.x 版中这是不可能的。

考虑到安装规则，如果用户仍需要包括 6.x SQL Server 可以选择如何设置自己的复制拓扑。

如果按并行升级法进行升级，事务复制应该正确转换。检查第 9 章“SQL Server 7 的强化复制”的复制功能，除了 6.x 版的 Log Reader Agent 和 Distribution Agent 之外，还有两个新的 Replication Agents:Snapshot Agent 和 Merge Agent。

13.6 升级后的工作

在结束向 Microsoft SQL Server 7 的升级后，可注册服务器，并／或在 Enterprise Manager 层次上添加新的服务器组。用户可能会想到要检查自己的配置设置，并改变自以为必要的任何东西。要执行任何升级后变化，可使用 Enterprise Manager 或 SQL Server 安装程序。在确认升级已经成功才可退出 6.x SQL 安装程序。

13.6.1 服务包

Service pack(服务包)是缺陷修复集，这种修复随对缺陷的认识已经积累了一段时期。重要的是在系统变得可用时，通过应用服务包来保持系统的当前状态。服务包一般用于代替系统数据库对象，如存储过程，或代替打开的系统文件，如 SQL Server 可执行程序文件。服务包的应用非常方便，在运行应用服务包的程序之前，使用步骤通常由备份数据库和停止 SQL Server 服务两部分组成。readme.txt 文件位于服务包内，通过各个步骤对用户进行指导，使用户了解服务包正在修复的内容。

13.6.2Hotfix

hotfix 类似于服务包，但通常是针对特殊安装时已经发现的特殊缺陷的。hotfix 是一个临时修复文件，可把它发送给顾客，直到包含缺陷修复的服务包被代替。hotfix 应用十分方便，相当于数据库对象的快速替换或操作系统文件。它包含一个带有说明的 readme.txt 文件。更重要的是，系统通知它关于最新得到的 hotfix，以便决定用户是否需要把它应用到 SQL Server。例如，如果 hotfix 用于复制，而自己又不使用复制，那么可以选择等待，直到下一个服务包。

升级并不是一个困难的任务，一般会为用户提供新的功能和关于数据库的平滑性能模型。为保持系统处于良好的工作顺序，用最佳可用技术运行，升级是最佳途径之一，无论何时变得可用，都应该安装。

第三部分　管理

第 14 章数据库管理

14.1 管理工具集

管理工具集在 Microsoft SQL Server 7.0 中已经作了扩展，详情已在第 7 章作过介绍。数据库管理员可用的 7.0 工具清单如下：

- SQL Server Query Analyzer
- Data Transformation Package
- SQL Server Profiler
- Index Tuning Wizard
- Replay SQL Server
- Database Maintenance Plans
- Web Assistant
- SQL Mail
- Enterprise Manager 内置的 Visual Database Tools
- Replication Monitor

关于命令行类型的实用程序，请参阅第 8 章，如块拷贝(BCP)和 OSQL，新的 ISQL 型的命令行实用程序，该实用程序使用 ODBC 与服务器进行通讯。

14.2 系统存储过程

编写 SQL Server 系统存储过程的目的，是提供可执行所有数据库管理任务的功能，同时防止用户直接访问系统表。通过使用 Enterprise Manager,扩展服务器和 master 数据库，并双击存储过程，可以查看系统存储过程清单。

选择特定系统存储过程并双击，可查看选中系统存储过程的 Transact-SQL 代码。其他系统存储程序位于 MSDB 系统数据库中，可以使用同样的方法访问，但要扩展 MSDB 数据库，而不是 master 数据库。如果其他复制系统存储过程已经标记为复制，那么该过程就位于复制分布数据库或用户数据库中。

通过功能区域可在下列目录中组合系统存储过程：

- Catalog(目录) 数据词典功能
- Cursors(游标) 游标功能
- Distributed Queries(分布式查询) 分布式查询功能
- SQL Server Agent(SQL Server 代理) 任务管理功能
- Replication(复制) 复制功能
- Security(安全性) 安全性管理功能
- System(系统) SQL Server 管理功能

- Web Assistant(Web 助手） Web 功能
- Extended Procedures(扩展过程） SQL Server 与外部程序的接口
- OLE Automation(OLE 自动化） 在 Transact-SQL 内的 OLE 自动化

在第 23 章中列出和描述了系统存储过程。

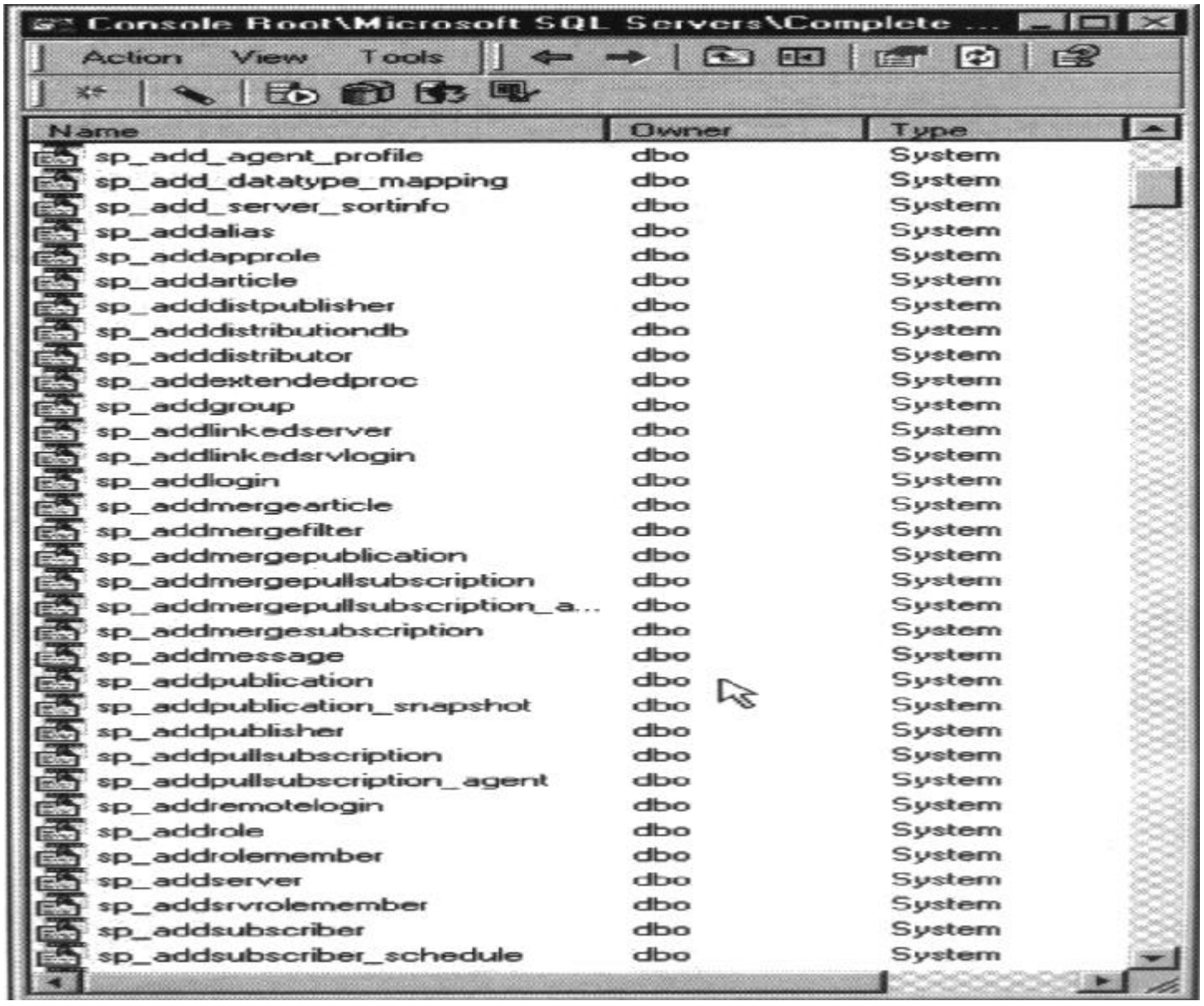

图 14.1 位于 master 系统数据库中的系统存储过程

14.3 系统表

系统表的功能是用于 Microsoft SQL Server 的目录或数据词典。关于 SQL Server、每个数据库和其他数据库对象的信息都存放在系统表中。

确定是否在不使用系统存储过程时进行特定升级的配置选项称为 allow updates。该选项缺省时应设置为关(0)，此时可防止特定升级到系统表。记住，在 allow updates 选项设置为开(1)时创建的任何存储过程，可继续能够修改系统表。在把 allow updates 设置为开或真(1)之前，最好是以单用户模式启动 SQL Server。

特定修改到系统表的语法是：

```
sp_configure ' Allow Updates ', 1
GO
RECONFIGURE WITH OVERRIDE
GO
```

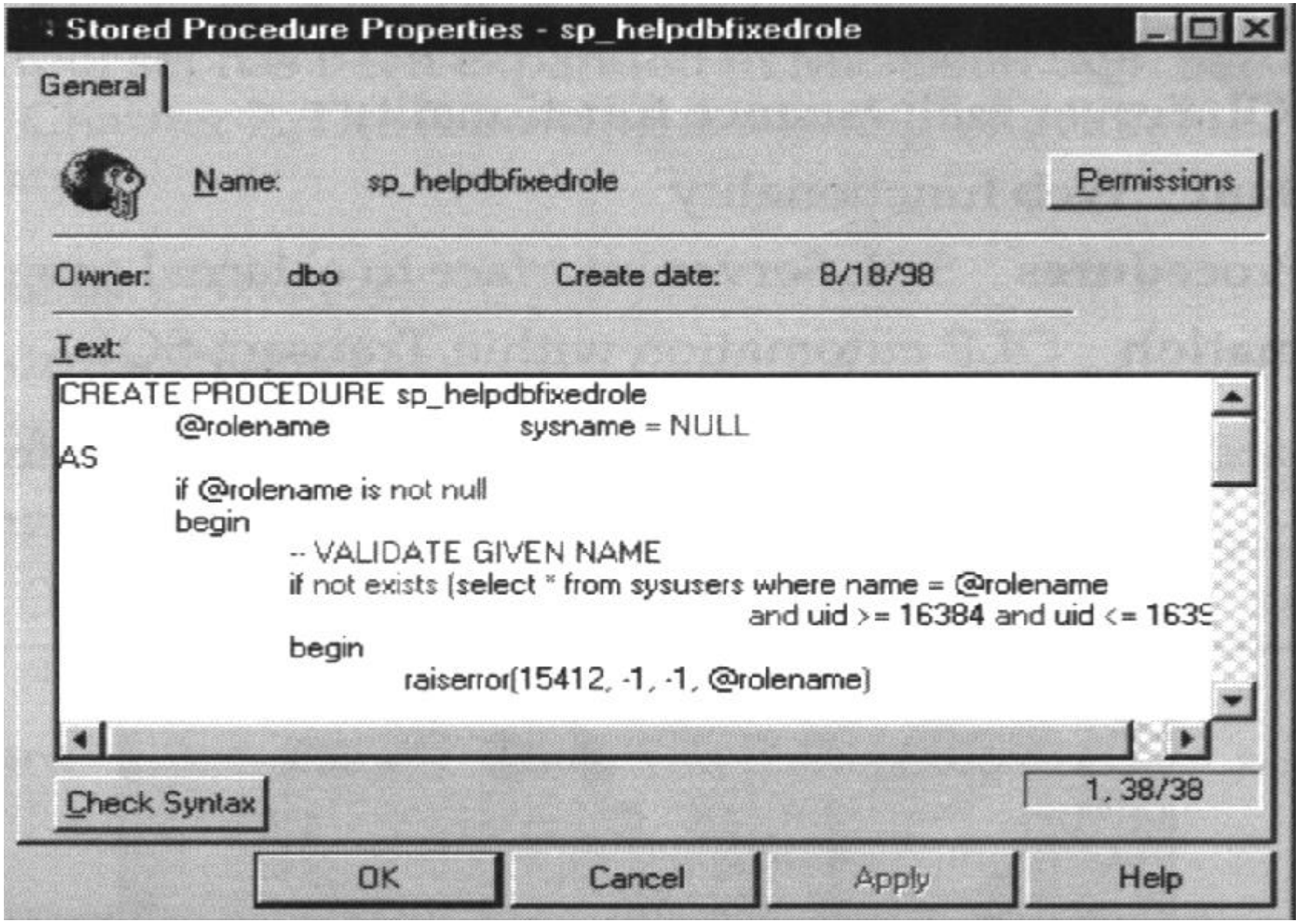

图 14.2 用于系统存储过程的 TransactSQL 代码

用户可运行 Enterprise Manager，从 Tools Menu 上选择 Query Analyzer，然而运行 sp_configure 系统存储过程把 allow updates 选项设置成真(1)，如图 14.3 所示。

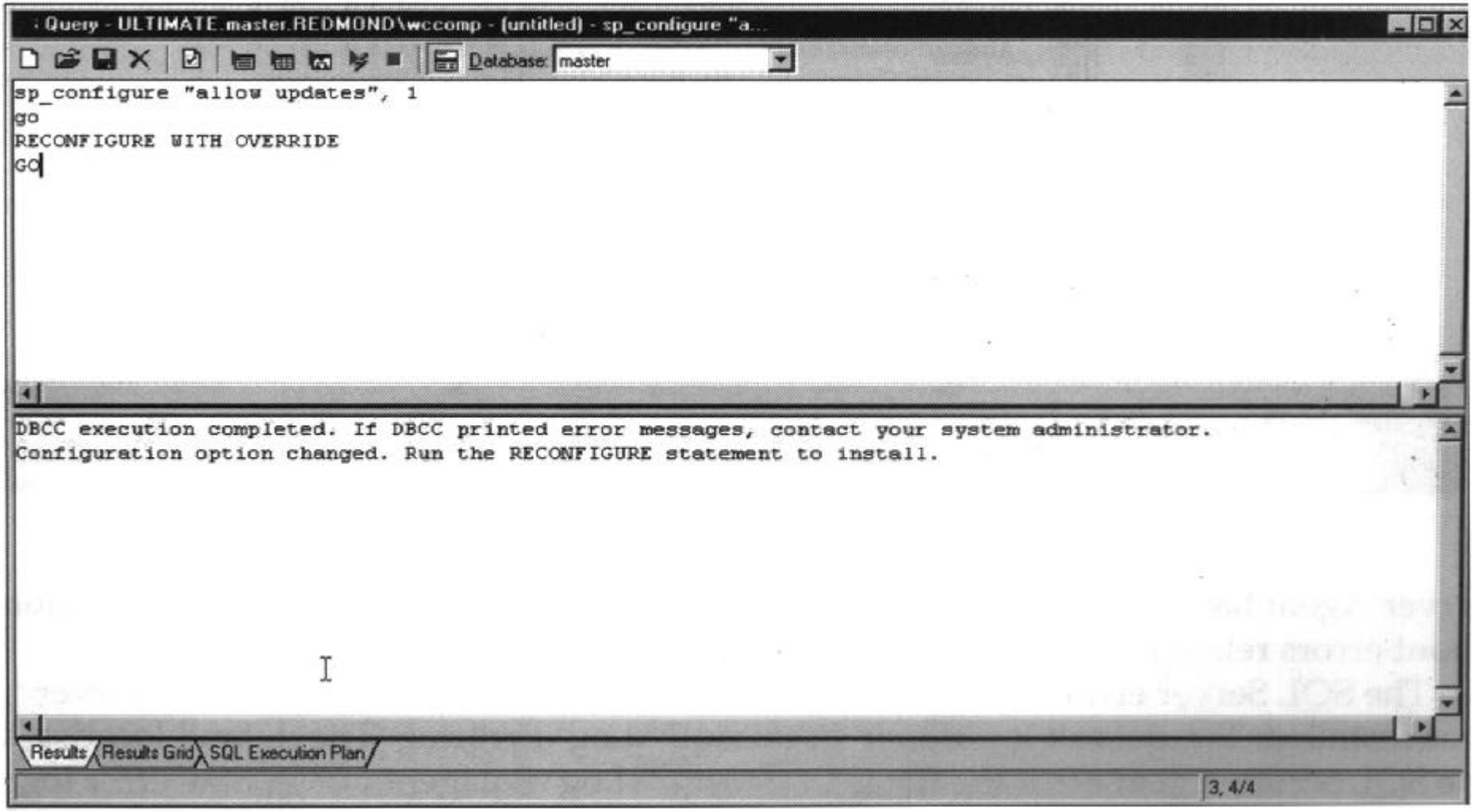

图 14.3 允许修改到系统表

Microsoft SQL Server 中的每个数据库都包含某种系统表。master 系统数据库包含大多数系统表(如图 14.4 所示)，但每个数据库都包含一系列本身的系统表，这取决于数据库的功能。MSDB 系统数据库包含具有作业编序功能的 SQL Server Agent 的系统表。distribution 系统数据库和标记为复制的用户数据库包含关于复制的系统表。所有系统数据库和用户数据库都包含 master 数据库中的系统表子集，这种系统表子集包含关于每个数据库中包含的对象的信息。在第 22 章“系统表”中列出并描述了系统表。

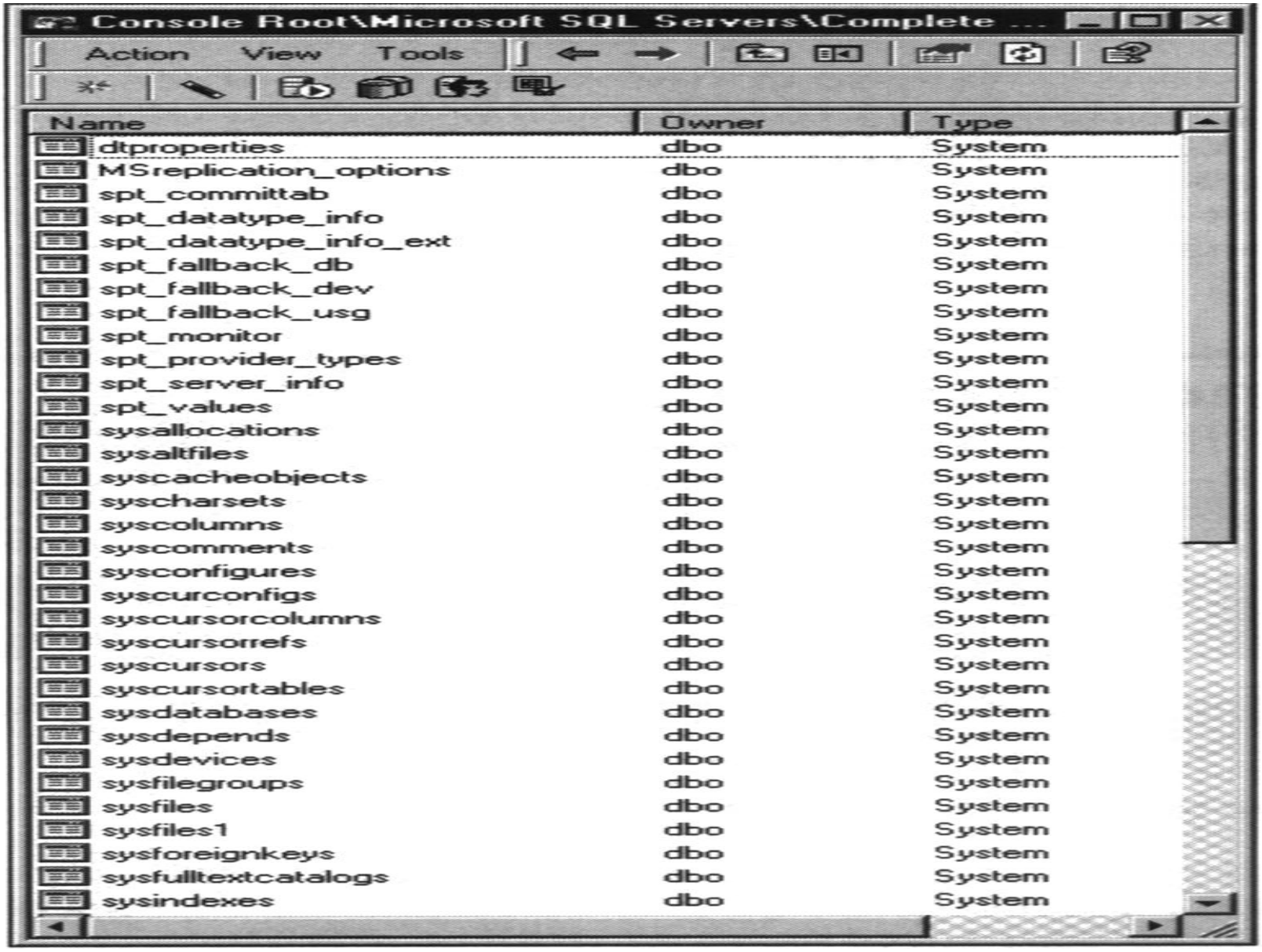

图 14.4 主系统数据库中的系统表

14.4SQL 错误和错误日志

SQL 可能发生错误。如果出现这种情况，并且安全性足够高，则这些错误被写入 SQL Server 错误日志。错误日志是确定 SQL Server 的运行良好状态的重要方法，通过研究在此报告的任何错误也是很重要的。Microsoft SQL Server 和 SQL Server Agent 都有错误日志文件。SQL Server Agent 错误日志包含了关于来自 SQL Server Agent 服务的作业和任务的错误信息。

SQL Server 错误日志在安装 SQL Server 时位于所选择的位置。缺省位置是 C:＼Mssql7＼Log＼Errorlog。SQL Server Agent 错误日志文件的缺省位置是 C:＼Mssql7＼Log＼Sqlagent.out。每次启动 SQL Server 时都要创建新的错误日志文件。六代错误日志文件进行累加，扩展名为整数的错误文件 errorlog.1、errorlog.2、errorlog.3 等直到 errorlog.6 都存放在安装时选中的或缺省的位置，如果不再改变的话。观察错误日志至少有两种方法：

- SQL Server Enterprise Manager
- 用文本编辑器打开该文件

要用 Enterprise Manager 观察 SQL Server 的错误日志，在选择了服务器后可扩展 Enterprise Manager 体系结构树中的 ErrorLogs 入口，如图 14.5 所示。

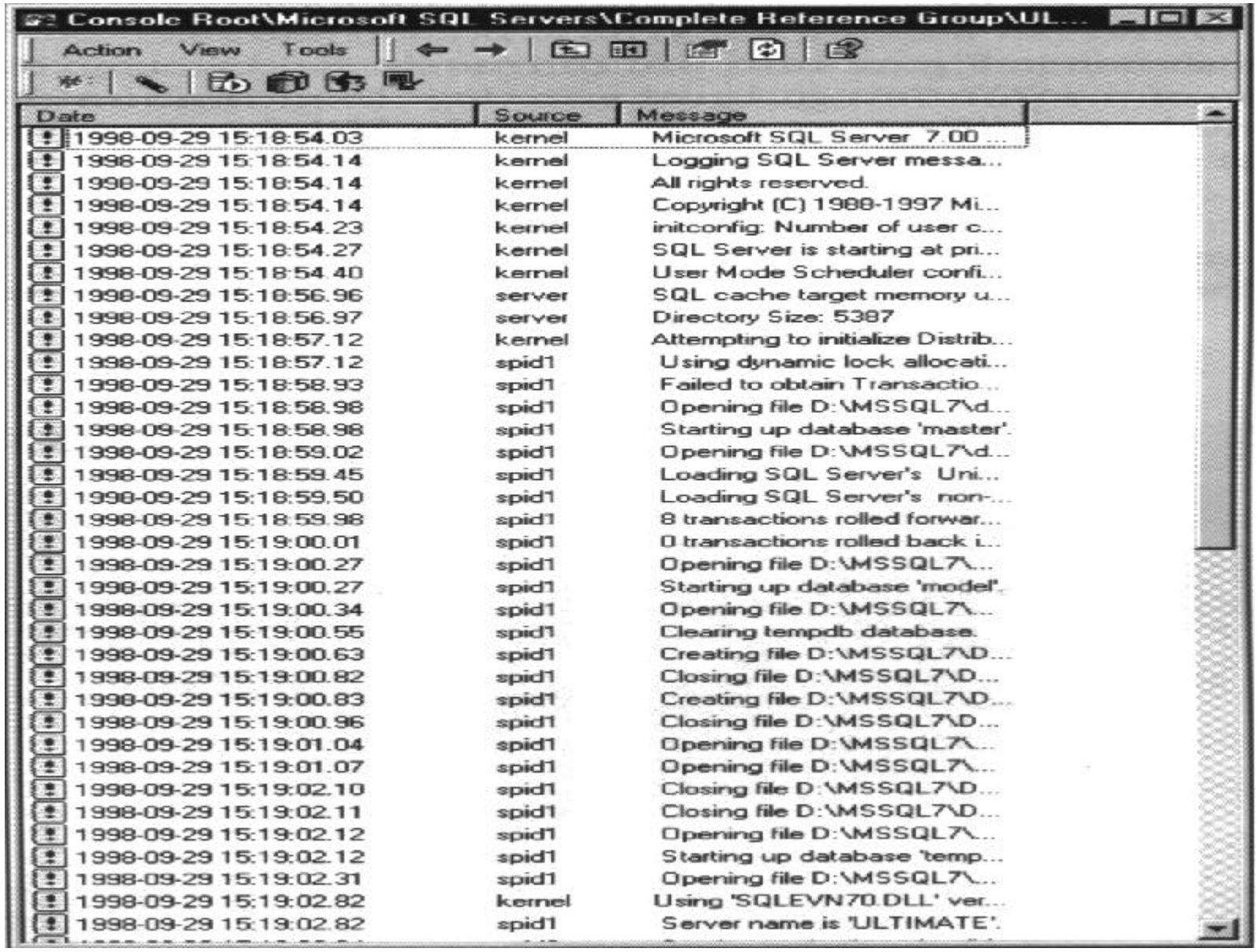

图 14.5SQL Server **错误日志**

使用 Enterprise Manager,选择了服务器，然后右击 Enterprise Manager 体系结构树中的 SQL Server Agent 项，可观察 SQL Server Agent 错误日志，如图 14.6 所示。

通过右击 Enterprise Manager 体系结构树中的 SQL Server Agent 项，然后再从快捷菜单中选择 Properties，用户可以访问屏幕，设置 SQL Server Agent 错误日志的选项。可以设置的选项如下：

- SQL Server Agent 错误日志的名称和位置
- 错误日志中是否包含跟踪执行消息
- 用户想通过每次发生 SQL Server Agent 错误时在其计算机上的网络发送而接受弹出式消息的人的 NT 登录字段

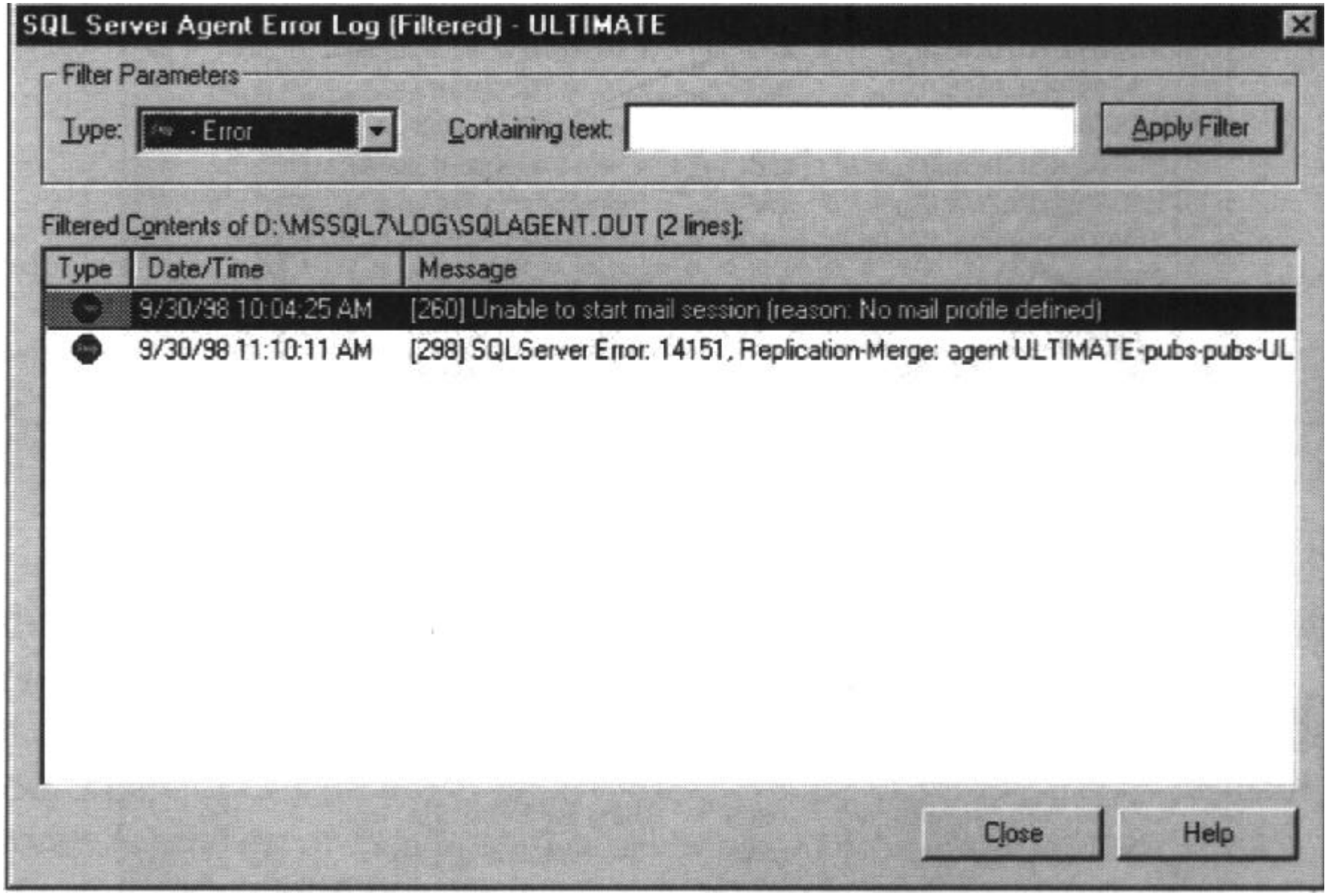

图 14.6 样本 SQL Server Agent 错误日志 (SQLAGENT.OUT)

在如图 14.7 所示的屏幕上，可以看到能改变 SQL Server Agent 错误日志的名称和位置的字段。允许在 SQL Server Agent 错误日志中包括执行跟踪消息，也有一个复选框。记住，如果服务器必须把许多项都写到错误日志中，将影响整体性能。错误日志在某些环境下具有以惊图 14.7 在 SQL Server Agent Properties 对话框中设置错误日志选项人速率增大的能力，在选择此选项时需要引起注意。这并非只是一个不检查和忘记检查的问题。如果检查包含执行跟踪消息选项，可监测 SQL Server Agent 错误日志文件，以确保硬盘有足够的空间来支持该文件的膨胀。如果决定通过网络发送来发送弹出式消息，则接收方一般是计算机操作员或数据库管理员，他们担负着在与 SQL Server Agent 服务有关的工作流中解决错误的责任。

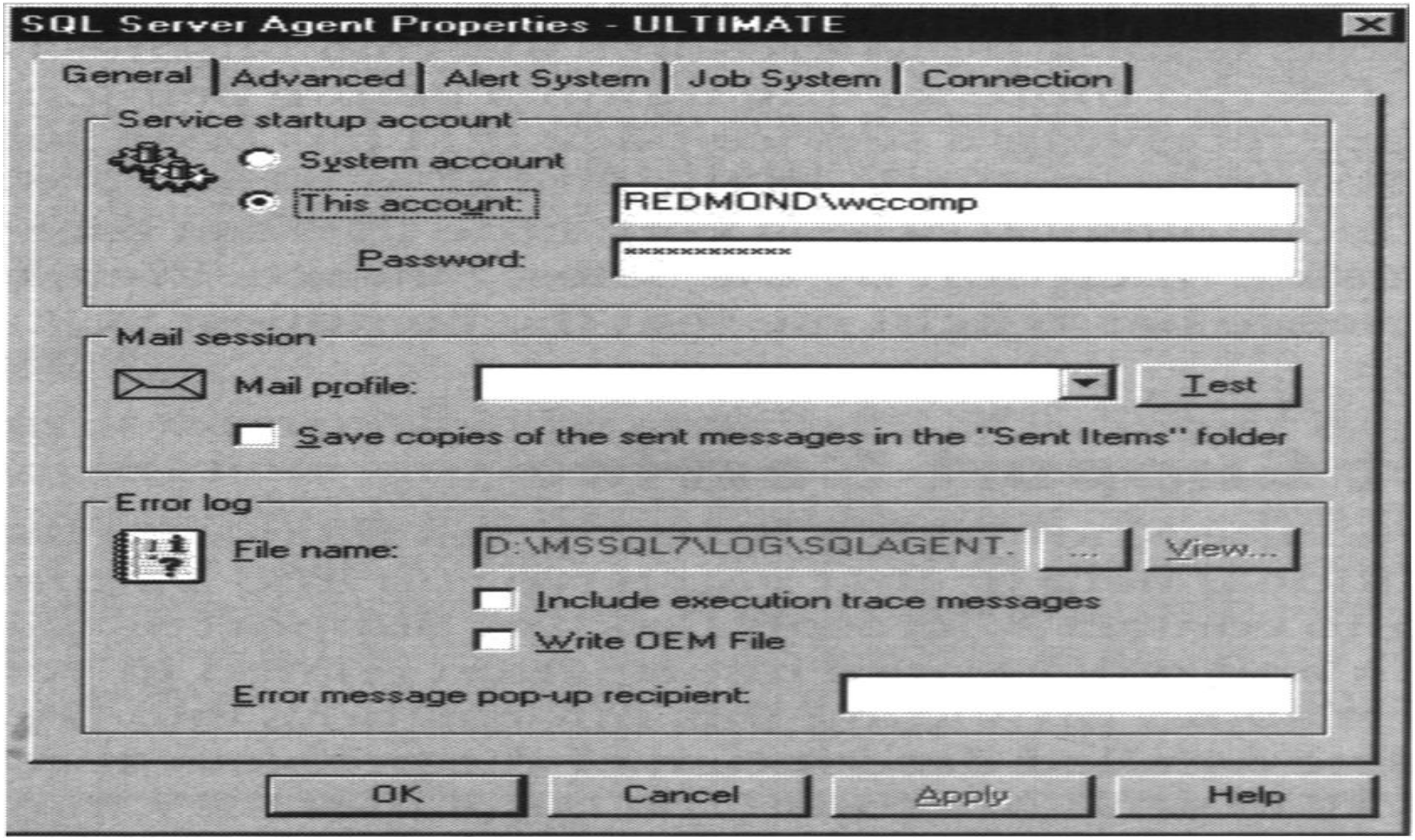

图 14.7 在 SQL Server Agent properties 对话框中设置错误日志选项

14.4.1 监测错误日志

通过监测 Microsoft SQL Server 错误日志来确定 SQL Server 上正在发生什么事件，这是一个好主意。与 SQL Server 和 SQL Server Agent 有关的事件都被写入各自的错误日志中，同时也写入可使用 NT Event Viewer 来观察的 Windows NT 应用程序日志中。如果用户碰到问题，不仅要检查 SQL Server 错误日志或 SQL Server Agent 错误日志，而且还要检查 NT 应用程序错误日志。由 SQL Server 和 NT 操作系统生成的所有信息的汇集，查看这个或那个日志更能对问题给出完整的标识。两种日志对所有记载的事件都打上时间戳记。

14.4.2TechNet 和 Knowledge Base Articles

如果用户真想能够有效地管理自己的 SQL Server 错误日志，Micrsooft TechNet Monthly Issues 是一种必不可少的资源。TechNet(Technical Information Network，技术信息网)是包含 Microsoft 产品信息的活页笔记本，全部装有 CD-ROM，其中包括 Microsoft SQL Server。技术笔记、情况研究、资源组件(kit)、产品情况以及 Microsoft Knowledge Base 都给出更新的信息，可使用户以可能的最有效的方法支持 Microsoft SQL Server。用户不必依靠自己的雇员来提醒自己访问这些信息。TechNet 适用于一年几百美元的普通人员，对于了解最新的 Microsoft 产品，并给出在自己的 SQL Server 上发生的错误的

详细、特别、历史的信息，花每个便士都是值得的。

包含在 TechNet CD-ROM 中的 Knowledge Base 包含了关于该错误的信息，在许多情况下，可为解决关于该错误的问题提供了准备工作。用户所经历的许多错误在世界各地的其他 SQL Server 上都发生过，并已向 Microsoft 公司作了报告。然后都用 Knowledge Basearticles 把错误及解决方案进行归档，对经历过类似问题其他用户也是可用的。更重要的是，在 Knowledge Base 中查找错误并按照在文章中找到的推荐方法解决。

大多数错误都是在后来的服务包中更正的，这就是把 SQL Server 升级到最新可用的服务包为什么重要的原因。服务包都由 Microsoft 内部商务应用程序进行过严格的测试，在投放市场前在内部发行以进行更多的测试。用户尽管放心，你所接到的服务包已经在大小和封装技术方面，经受了最严格商务应用程序的经验。在 Microsoft 内部逃避由 Sales、Finance、Web 和 SAP 应用程序对服务包的严格锤炼，几乎是不可能的。笔者的经验说明最健康的 SQL Server 是用新升级版升级过的 SQL Server。

14.4.3Microsoft 产品支持服务

用户一旦回顾起关于错误的 Knowledge Base 文章，并决定需要附加的帮助，或已经进入还未有人报告的状态，需要与 Microsoft Product Support Service(PSS)联系。PSS Engineer 将为您提供帮助，要求您提供关于运行环境的信息。与您合作并提醒您提供信息，请设法向 PSS Engineer 提供问题和状况

的详细完整的说明。要确保您有办法向 Microsoft 发送大文件，这些问题可能需要对上述问题进行调试。还有可能要求提供错误记录和数据库文件，以便比较并向 Microsoft 提供解决该问题需要的途径。这将大大有利于及时找到问题的解决方法。

14.4.4SQL Server 对发生错误的策略

SQL Server 7.0 版的策略是杀死有问题并在错误日志中记录了错误的进程 ID。这样可防止引起可能触发 SQL Server 缺陷或中断过程处理规则的过程可能造成的损失。更重要的是要紧紧跟踪 TechNet 和 Knowledge Base 文章，以找到关于该问题的信息。如果用户相信自己已经遇到新的未报告的缺陷，应该向 Microsoft 报告。

14.4.5 研究错误

当在错误日志中看到错误时，可用 TechNet 的 Knowledge Base 搜寻错误号。当通过查询收到 Knowledge Base 文章列表时，应阅读全部 Knowledge Base 文章，并决定那些接近自己的状态。要注意在其中有错误报告的版本，并与当前版本进行比较。然后按相应条款中建议的方法去处理。

如果未找到该错误，或需要附加的帮助，可打电话给 Microsoft Product

Suppart Services 报告该错误，随后即可得到帮助。有些错误在 Microsoft 内部的 Knowledge Base 文章中，而有关信息尚未放入到 TechNet 中。

14.4.6 辨别致命错误

用户通过阅读 Knowledge Base 文章和寻找安全级别，可判断问题的严重程度。在错误日志中一旦发现错误就设法这样做。这就是为什么错误日志中一写入错误,就要尽快设置自己的 SQL Server,以使用 SQL Server Agent 向 SQL Server 操作员或数据库管理员写信或发 e-mail。

14.4.7 周期运行的 SQL Server

如果周期运行(停机和开机)SQL Server，有些可能会造成损坏的错误可以得以避免，用户可能很想立即知道自己是否属于这种情况，以便可防止退出和启动 SQL Server 可能造成进一步损失时做出自己的选择。Knowledge Base 允许用户了解自己是否有需要周期运行 SQL Server 的错误。

在 tempdb 系统数据库内多次会发生问题，tempdb 系统数据库是当停止和启动 SQL Server 时初始化的工作数据库。如果 tempdb 系统数据库报告发生故障，此时是周期运行 SQL Server 的最佳时间。

14.5 驱动器管理

对于安装有关键商务应用程序的大型 SQL Server，需要了解磁盘驱动器的详细情况。测试执行配置就是在硬件层次上运行 RAID。费用较低的配置是在操作系统层次上运行 RAID。

14.5.1 廉价磁盘的冗余阵列

如有可能，在硬件层次上最好是运行 Redundant Array of Inexpensive Disks(廉价磁盘冗余阵列，RAID)，RAID 是磁盘拆卸的方法，这种方法可通过提供部分冗余信息，在 RAID 的 1～5 层次上提供故障允许值。数据分散在多个磁盘上，此时可同时发生 I／O。RAID-0 不提供故障值；RAID-5 可提供最大故障允许值。最好是将 RAID0,1 和 5 的层次与 Microsoft SQL Server 联合使用。

14.5.1.1 数据库文件：RAID-0 或 RAID-5

使用 SQL Server 的 RAID 技术可提高性能，执行 RAID 的方法可直接影响 SQL Server 的性能。把数据库文件放到 RAID-0 上，可实现最佳性能(也称为基于硬件的并行集，因为数据分割的方法和在一个阵列上通过整个磁盘放置)。然而，

如果一个磁盘失效，整个 RAID-0 并行集都变得不可访问，无法调整失效磁盘上的数据。

如果系统随时由于种种原因而不能关闭，但仍可涉及到性能，可使用 RAID-5 提供冗余，因为数据和奇偶性总是存放在不同的磁盘上。RAID-5 类似于 RAID-0，但包括故障允许值的冗余，此时奇偶性可通过所有磁盘写入。使用在硬件层次上为数据库文件执行的 RAID-5，可节省处理器计算奇偶性的工作量，因此可提高处理器的性能。

14.5.1.2 事务日志文件：RAID-1(磁盘镜像)

最好是在硬件层次上对包含数据库事务记录文件的驱动器使用执行的 RAID-1(磁盘镜像)，因为 RAID-1 的冗余最佳。RAID-1 使用镜像集提供磁盘的冗余拷贝，而且也是阅读数据的优秀执行器；然而，它将降低磁盘的写入速度。如果有关键应用程序，在决定如何管理磁盘驱动器时，必须冒着可能提供所希望的性能的风险。

14.5.2 并行集

记住，RAID 的奇偶并行集可在 NT 操作系统层次(软件层次)以及硬件层次上执行。然后假设处理器循环运行，处理器要求执行奇偶性检查，如果用户涉及

到成本比处理器循环的使用更高，那么在 NT 操作系统层次上执行 RAID 就比较便宜，并且仍可提高性能。NT Windows RAID 并行集对于热衷于改进性能和降低成本的人是一种良好的解决方案。

14.6 移动数据库

当面临把数据库移到另一个服务器的任务时，须考虑以下许多问题：

- 两个服务器上的排列顺序和字符集都相同吗
- 两个服务器上的 SQL Server Authentication Modes 相同吗？
- 设置了 Trace Flags 吗？
- master 系统数据库中有用户的存储过程吗？
- NT Registry 中的 ANSI to OEM 选项是打开还是关闭的？
- 两个服务器的内存和版本兼容吗？
- 两个服务器上的 SQL Server 登录和远程登录都相同吗？
- sysservers 表相同吗？
- 目录服务器上有特别的警告吗？
- 目标服务器上安装了要运行的作业或任务吗？
- 两个服务器上的 NT 组相同吗？
- 数据库选项相同吗？
- 在调度时间上支持事务日志吗？

如果要把数据库从一个服务器上移到另一个服务器，必须回答和验证上述所有问题。数据库必须先移到某个环境中，该环境应与源服务器相兼容和相似，以便二者具有相同的格式，移动之前表现的特性应相同。

14.7 重建 SQL Server

在必须重建整个服务器、硬件和全部设置的情况下，如果需要移到一个较大的服务器上，或经历过硬件的灾难性失败，为确保具有重建的脚本，每天都应该执行下列任务。设置重建脚本需要使用 SQL Server Agent 设置每天要运行的下列任务。这样就会有重建 SQL Server 的基本文件。

· 每天运行系统存储过程 sp_configure、sp_helpdevice、sp_helpntgroup，并把运行结果保存在不同于以此定位 master 数据库的磁盘驱动器中。
· 备份全部数据库。
· 使用 BCP 实用程序把 master 数据库系统表拷贝成一个文本文件。
· 以书面文件的形式制定在驱动器上位置的标准。
· 以书面文件的形式，制定安装 NT 和 SQL Server 的标准。

一旦有了这些基本文件，就可以按以下步骤重建服务器：

1. 采购硬件。
2. 根据标准安装 NT(这些标准应书面记下来)。
3. 根据标准安装 SQL Server(这些标准应书面记下来)。

4．不要存储 master 系统数据库，以使 master 系统数据库中的任何操作都必须重建。用户要了解 master 数据库的内容，通常情况下，在 master 系统数据库中不创建用户存储的程序。
5．在新的服务器上重建 NT 组。
6．通过检查文本文件的内容重建 master 系统表，该文本文件已经被读出，并且重建了 SQL Server 的值。在 6.5 版中，笔者常用 BCP 备份系统表，由于从 6.5 版升级到 7.0 版系统表发生了巨大变化，外来的关键关系可能不同，所以此时再这样做必须加倍谨慎。
7．恢复自己的数据库。
8．用自己的应用程序和用户测试其配置，看看这些用户是否仍能登录，能否运行该应用程序。
9．准备排除任何有问题的部分。

当上述处理过程无论何原因变得必要时，这是重建服务器的一般步骤。运行系统与测试附属在每个数据库的应用程序同时进行。

14.8 管理关键任务的应用程序

如果数据库包含对用户的业务非常关键的信息，那么用户可能希望采用下列某些方法，以便获关于自己的服务器的容错、冗余和故障率等对策：

· 使用事务日志备份，每周至少运行一次整个数据库备份。

· 在日常安排的基础上运行 Database Consistency Check 实用程序(对系统数据库每天运行一次，对用户数据库至少每周运行一次)。
· 使用 SQL Server Agent 把严重的(严重等级 21-25)SQL Server 错误打印出报告，以便对致命的错误能立即作出反应。
· 使用包含数据库文件的驱动器的硬件等级 RAID-5。
· 使用包含了事务日志文件的驱动器的硬件等级 RAID-1(监测)。
· 对故障排出对策考虑使用 Microsoft Transaction Server(MTS)。
· 对故障排出对策考虑使用 NT Clustering。
· 在不同地点都拥有备份服务器，并复制到该备份服务器。

真诚地备份数据库，并使用 SQL Server Agent 对严重错误进行报警是特别重要的。立即作出反应可把损失减小到最低程度，长期坚持执行这种过程会使用户对解决错误变得轻车熟路。

当使用第四章介绍的 SQL Server Agent 时，这一过程执行起来显示得非常方便。随着时间的延长，用户将会对所遇到的应用程序之间、版本之间各种不同的错误变得熟能生巧。

14.9 管理特大型数据库

特大型数据库(超过数千兆字节)的出现向数据库管理员起了挑战，因为日常的维护任务(如备份)和数据库定期检查就要花费大量时间。Microsoft SQL

Server 7.0 通过改进历史上对大型数据花费大量时间开发的实用程序，可直接解决这些问题。DBCC 的运行速度奇迹般大大提高，在不影响服务器上其他程序运行，或不降低性能的前提下即可进行数据库备份。现在备份工作只占不到 5% 的 CPU 资源，实际其他数据库用户甚至感觉不到备份的运行。

如果采用 Very Large Database(VLDB)，要确保另有一台服务器用作备份服务器，在备份服务器上可运行数据库的常规检查和进行测试。如果硬件遇到灾难性的故障，通过备份也可以提供一个新起点，这对于从当前服务器取出数据永远都是有必要的。

14.10 启动和终止 SQL Server

Microsoft SQL Server 既可以自动启动，也可以手动启动，所有启动方法都可使 SQL Server 起到 NT 的作用，只有在命令行提供下运行带有-C 参数的 sqlservr.exe 时除外。如果选择自动启动 SQL Server，在安装 SQL Server 或在安装 SQL Server 之后使用 Enterprise Manager 时，均可以指定自动启动，通过右击 Enterprise Manager 分层树中选中的服务并选择 Properties 即实现自动启动。如果在 NT 启动时复选了 Auto Start SQL Server 复选框，在下次启动 Windows NT 时 SQL Server 就会自动启动。

使用 Enterprise Manager 和选中的 Start 或 Stop 后，启动和中止 SQL Server 是非常方便的，在此还可以暂停 SQL Server 的运行。这意味着不允许生产新的

登录，但允许已经存在的登录继续运行。如果 SQL Server 已经从带有-C 选项的命令被启动，那么就无法使用上述方法。

手工启动和终止 SQL Server 的另一种方法，以及类似于 SQL Server Agent 和 Distributed Transaction Coordinator 的相关服务，就是使用 SQL Server Service Manager。从 Windows NT Start 菜单中选择 SQL Server Service Manager 便可实现上述功能。双击绿色灯泡可启动 SQL Server，双击黄色灯泡可暂停 SQL Server 的运行，双击红色灯泡可中断 SQL Server 的运行。从屏幕上的下拉列表框中可以选择该项功能。

启动和终止 SQL Server 的第三种方法是，从 Control Panel 上选择 Services 可观察运行在计算机上的服务列表，然后选择 MSSQL Server，并选择 Start 或 Stop 按钮即可。用 Control Panel 法也可以启动和终止 SQL Server Agnet 服务和 Microsoft Data Transaction Coordinator(MSDTC)服务。

启动和终止 SQL Server 还有另一种方法，就是从命令行提示符上直接实现。当用这种方法启动 SQL Server 时，SQL Server 的表现形式很不相同，它允许传递参数，以防止 SQL Server 的服务运行，并可以单用户模式或最低配置启动 SQL Server。下一节概括了这些参数，并提示如何从命令行提示符启动 SQL Server。

14.10.1 从命令行提示符启动 SQL Server

从 NT 启动菜单选择 Run 并键入以下三个命令，可以从命令提示符下启动

Microsoft SQL Server：

命令的语法为

```
net start mssqlserver
```

或

```
sqlservr
```

或

```
net start  SQLServerAgent
```

缺省的自变量为：

· -dmaster_file_path 该自变量代表缺省安装的 master 数据库文件的路径 C:＼Mssql7＼Data＼Master.mdf。如不指定此自变量，可使用 NT Registry 中的值。

· -eerror_log_path 该自变量代表缺省安装的错误日志文件的路径 C:＼Mssql7＼Log＼Errorlog。如不指定此自变量，可使用 NT Registry 中的值。

· -lmaster_log_path 该自变量代表缺省安装的 master 数据库日志文件的路径 C:＼Mssql7＼Data＼Mastlog.ldf。

其他启动变量为：

· -c 该变量可启动 SQL Server 不能作为 Wndows NT 服务而运行的 SQL Server。所有的系统信息都显示在启动 SQL Server 的窗口中，此时用户不能使用 SQL Server Enterprise Manager、SQL Server Service Manager、Control Panel 中的 Services 应用程序或任意网络命令来暂停或终止 SQL Server 的运行。另

外，在注销之前无法终止 SQL Server。

· -f 该变量可用不执行 CHECKPOINT 的最低配置启动 SQL Server。如果该服务器因为此配置选项设置太高而不能启动，则可以这样启动。该自变量对于系统表选项可打开 allow updates 选项，并可以单用户模式启动 SQL Server，所以，用户必须记住要终止 SQL Server Agent 的运行，或占用用户的唯一的连接。对内存的用法、用户的 连接、开放的数据库、锁闭和打开的对象、高速缓冲区的语言信息、同步 I／O、总体过程和高速缓存的改变，都可设置到最低值。另外，过程高速缓存可设置到 50%，使远程访问和向前读均失效，启动存储程序被忽略。改变用户需要改变的东西，重新以正常方式启动 SQL Server。

· -m 该自变量可以单用户模式启动 SQL Server，终止 SQL Server Agent 或占用用户的唯一的连接，只有单一用户可以连接。可打开 allow updates 选项，意味着如果需要升级系统表，并且当 allow updates 对系统表为真时不想别人登录到 SQL Server。

· -n 用该变量启动 SQL Server 可指定 Windows NT 事件日志不用于记录 SQL Server 事件。与-e 变量一起使用，否则 SQL Server 事件也不记入 SQL Server 错误日志。

· -pprecision_level 该变量可用来确定十进制数据和公制数据类型的最大精度等级。SQL Server 的缺省值为 28，如果不用-p 自变量提供精度等级，该系统可使用的最高精度等级为 38。

· -sregistry_key 使用该变量可以通过使用来自存储在 registry_key(Server Subkey)下的可替换的启动参数来启动 SQL Server,registry_key 可多次作为以

前定义的启动配置。

- ／Ttrace_number 该大写字母变量以指定的跟踪标志启动 SQL Server。
- -x 该自变量不保持可改进性能的 CPU 时间和高速缓存影响的比率统计。

终止 SQL Server 运行的另一种方法是，在 Transact-SQL 对话期间使用 SHUTDOWN 命令。

14.10.2SHUTDOWN 和 SHUTDOWN WITH NOWAIT

通过运行来自 OSQL 或 Query Analyzer 的 SHUTDOWN Transact-SQL 语句，或对于可发出 Transact-SQL 语句再次启动生效，并可逐点检查每个数据库。因为所有上述这样功能均允许结束，所以在 SQL Server 时，恢复时间缩短。使用 SHUTDOWN WITH NOWAIT 不会结束任何操作，并且 SQL Server 的运行可立即终止，但恢复时间延长。

如果用户时间充足，终止 SQL Server 的最佳方法是使用 SHUTDOWN 语句，因为所有登录均告失效，任何正在执行的 Transact-SQL 语句都允许结束，每个数据库均可执行 CHECKPOINT 语句。

14.11 归 档 处 理

数据库管理的另一个重要主题是，通过把旧的数据经过归档处理从数据库

中清除，来管理磁盘空间。定期清除和归档数据库中的历史数据就像在管理数据库时启动和终止数据库一样重要，并且这样可改善性能。

更为重要的是，通过归档功能建立自己的应用程序，并从开始就引入到设计中。当数据库中充满了不得不到下周才清除的无用数据时再开始考虑归档已为时已晚。归档任务计划和设计，尤其是如果这些数据为金融数据和税法管理数据，还必须保留一定时间。这并非是一个容易解决的问题，毫无计划或预见地去完成这一艰巨的任务可能会造成更大的问题，使数据库管理员和整个公司穷途未路。数据库结构及其中所包含的关系可能会超时变化，不足以把数据放在文本文件中。用户必须考虑到其中的关系，采取合适的始终需要恢复数据和了解数据的良好策略。遗憾的是，由于应用程序在建立时受时间限制，在产生应用程序崩溃时，归档常常只考虑到最不重要的事情。实际上，在应用程序的设计和实施中，维护数据是最重要的因素之一。

第 15 章 数据库的预防性维护

Microsoft SQL Server 7 通过提供 Maintenance Plan Wizard，使数据库的预防性维护进行起来十分方便，Maintenance Plan Wizard 贯穿创建维护计划的各个步骤。该向导使用 SQL Server Agent 的作业方案功能，并允许用户重新组织数据和索引页，在常规规划的基础上更新统计。Maintenance Plan Wizard 还允许用户管理关于数据库的策略测试，并确定数据库备份计划和数据页的自由空间百分比，可以将结果报告生成为一个文件或一个 Web 页。所有这些功能都可引入到包含 Database Maintenance Plan Wizard 的几个屏幕视图上。这对数据库管理员是一个十分惊人的技术进步，因为它执行起来简单，可减少用 Transact-SQL 对此功能进行编码的工作量。数据库维护计划可能无法回答用户的所有要求，以及在 SQL Server Agent 作业中使用 sqlmaint 实用程序或改变在此提供的功能性的多项选择。关于 sqlmain 实用程序已在第 8 章“实用程序改变”中作过讨论。

15.1 更 新 统 计

UPDATE STATISTICS 是可添加到 Database Maintenance Plan 的 Transact-SQL 语句。如果系统表中的行数变化惊人，那么运行该语句可以更新系统表中数值分布的统计结果。

基于成本的查询优选器使用上述统计来估算使用索引的成本，并设法以最低成本使用该索引。当向系统表增加行数或从中删去行数时，或当列中的值的分布发生变化时，或当在制订查询计划的查询优选器没有最新的统计信息时，都必须更新统计。这样可能会使优选器对所使用的索引做出较少优化的决定。如果统计信息指示为 0 行，甚至可能会对包含上百万行的表格进行扫描。图 15.1 表示 Database Maintenance Plan Wizard。利用 Database Maintenance Plan Wizard，用户可以对额外大型表格指明应采样的行数或表格的百分数。如果输入的值太小，SQL Server 将根据表中的行数更正输入的数值，并使用最低的数值数使采样具有代表性和可用性。在发出 UPDATE STATISTICS 语句时如果不使用 FULLSCAN 自变量，SQL Server 将对表格进行采样，以收集存储在系统表中的统计信息，查询优选器需要系统表制订查询计划。这对大型表格可以节省时间，因为只扫描样本行便可收集统计信息，并不需扫描整个表。

运行更新统计是忽略数据库维护任务最普遍的方式之一，如果所需要的统计在制订查询计划时不能更新，运行更新统计可能会影响 SQL Server 查询性能。UPDATE STATISTICS 语句的语法和自变量可在第 21 章找到。

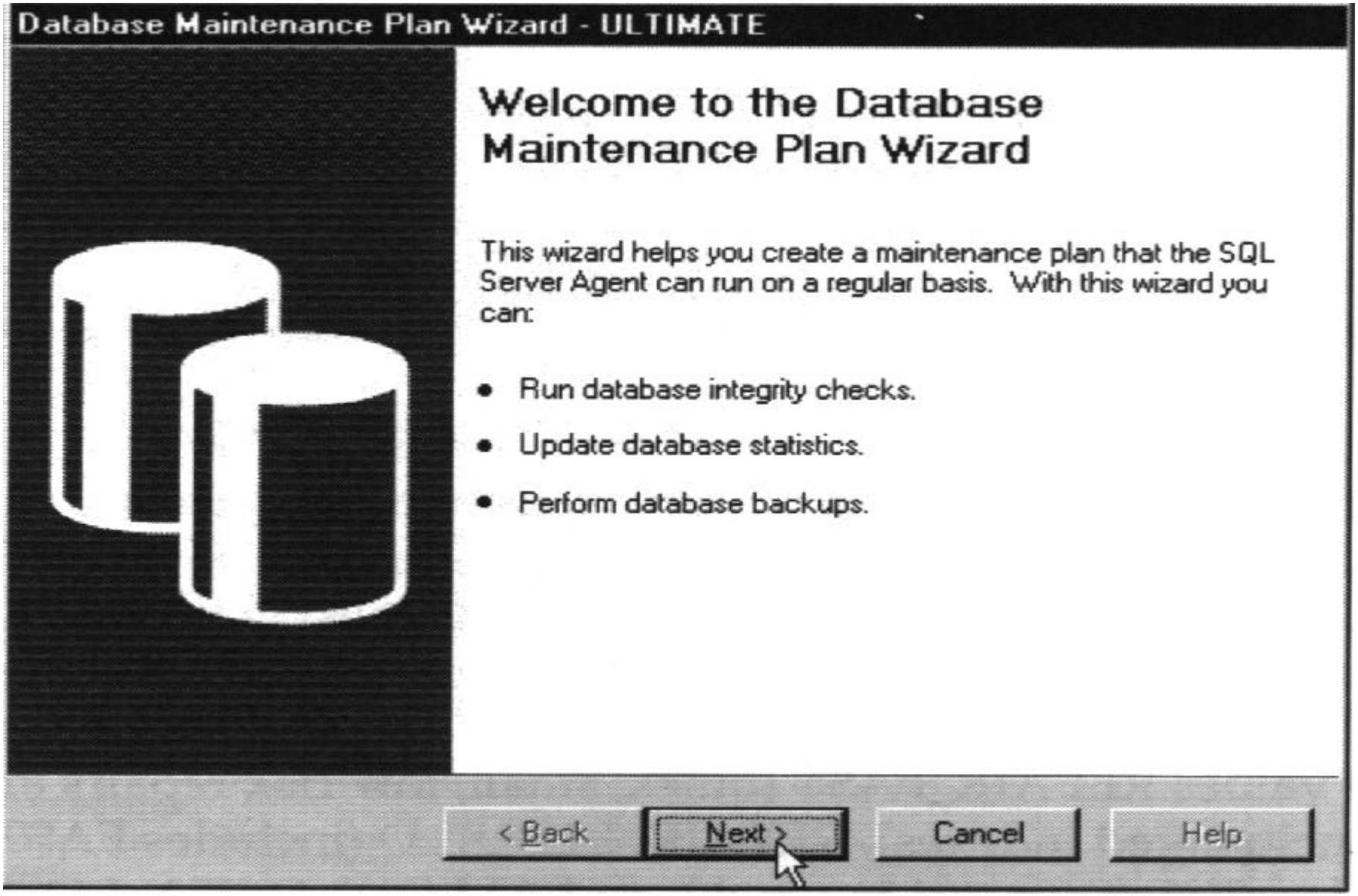

图 15.1Database Maintenance Plan Wizard

15.2 数据库一致性检查器

正如通过使用定期维护的 CHKDSK 实用程序运行检查硬盘很有必要一样，通过运行作为常规维护计划的一部分的 Database Consistency Checker(DBCC)来检查数据库，并验证其逻辑和物理一致性也是很有必要的。在备份数据库之前，最好是先运行一下 DBCC CHECKCATALOG 和 DBCC CHECKDB，这样可以确保不会用可能包含错误的备份覆盖最后的最佳备份，然后再检查 DBCC 输出文件，解决可能出现的任何错误，最后再备份数据库。并计划在用户不在系统中时运行 DBCC。如果用户的数据库包含大型表格，或数据库大于 5GB，在存储数据库之后用户可能想运行 DBCC 备份服务器，或使用可略去非成簇索引的扫描的 NO_INDEX 选项。但用户仍应检查自己的非成簇索引，但也许检查频度小于数据页和成簇索引。关于 DBCC 情况的详细信息，请参阅第 8 章。

DBCC CHECKDB 检查数据库中每个对象的完整性，并修复发现的任何错误。如果运行 DBCC CHECKDB，就一定不要运行 DBCC CHECKALLOG 或 DBCC CHECKTABLE。每个表的索引和数据页都要进行检查，以便更正页链接和一致性指针。然后再进一步检查索引，确保输入项分类正确，每页的偏移量和数据都是合理的。对于所有页都要检查每个表的文本尺寸、ntext 和图像，以及页分配。

DBCC CHECKCATALOG 用于检查系统表看是否一致性，每次预定在数据库上运行 DBCC 时,都应该运行 DBCC CHECKCATALOG,这样可用 Systypes 检查 Syscolumns 的完整性，用 Syscolumns 检查 Sysobjects 的完整性。syscolumns 中的每个数

据类型在 systypes 中必须有一行，而每个表和视图在 syscolumns 中也至少必须有一列。

在 Microsoft SQL Server 7 中，DBCC 的表现形式变化比较显著，用户不必再运行 DBCC DBREPAIR 来删除标记为有疑问的数据库，用 DROP DATABASE 就可实现上述操作。另外，应该使用 DBCC SHRINKDATABASE，而不使用 DBCC SHRINKDB，因为系统不再支持 DBCC SHRINKDB。

如果发现错误，DBCC 也具有修复数据库的能力。DBCC CHECKTABLE、DBCC CHECKDB 和 DBCC CHECKALLOC 都有修复选项，该选项的值可以为 REPAIR_FAST、REPAIR_REBUILD 和 ALOW_DATA_LOSS，其中 REPAIR_FAST 表示可进行快速小型低风险的修复，如非聚簇索引中的额外空间；REPAIR_REBUILD 表示，包括 FAST 快速修复再加上较长时间的修复，仍是低风险的；ALLOW_DATA_LOSS 表示，包括 REPAIR_REBUILD 修复，再加上更正分配错误、更正行或页错误、甚至包括删去已损坏的文本对象等。

DBCC 表现行为的另一个变化是，在反向兼容性上都支持 DBCC NEWALLOC 和 DBCC ROWLOCK。在 Microsoft SQL Server 7 中，DBCC NEWALLOC 已被 DBCC CHECKALLOC 所代替，用户可以删除所有 DBCC ROWLOCK 语句，因为在 Microsoft SQL Server 7 中，行的层次锁定已成为标准特性。

Microsoft SQL Server 7 的新索引结构要求 SQL Server 在聚簇索引减低时应重建对表格的全部非聚簇索引。减低成簇索引时可使用 DBCC REINDEX，不要使用 DROP INDEX。最后，希望看到用户在以前版本中创建的任何 DBCC 输出发生的变化。关于 Microsoft SQL Server 7 的 DBCC CHECKDB 输出，如图 15.2 所

示。

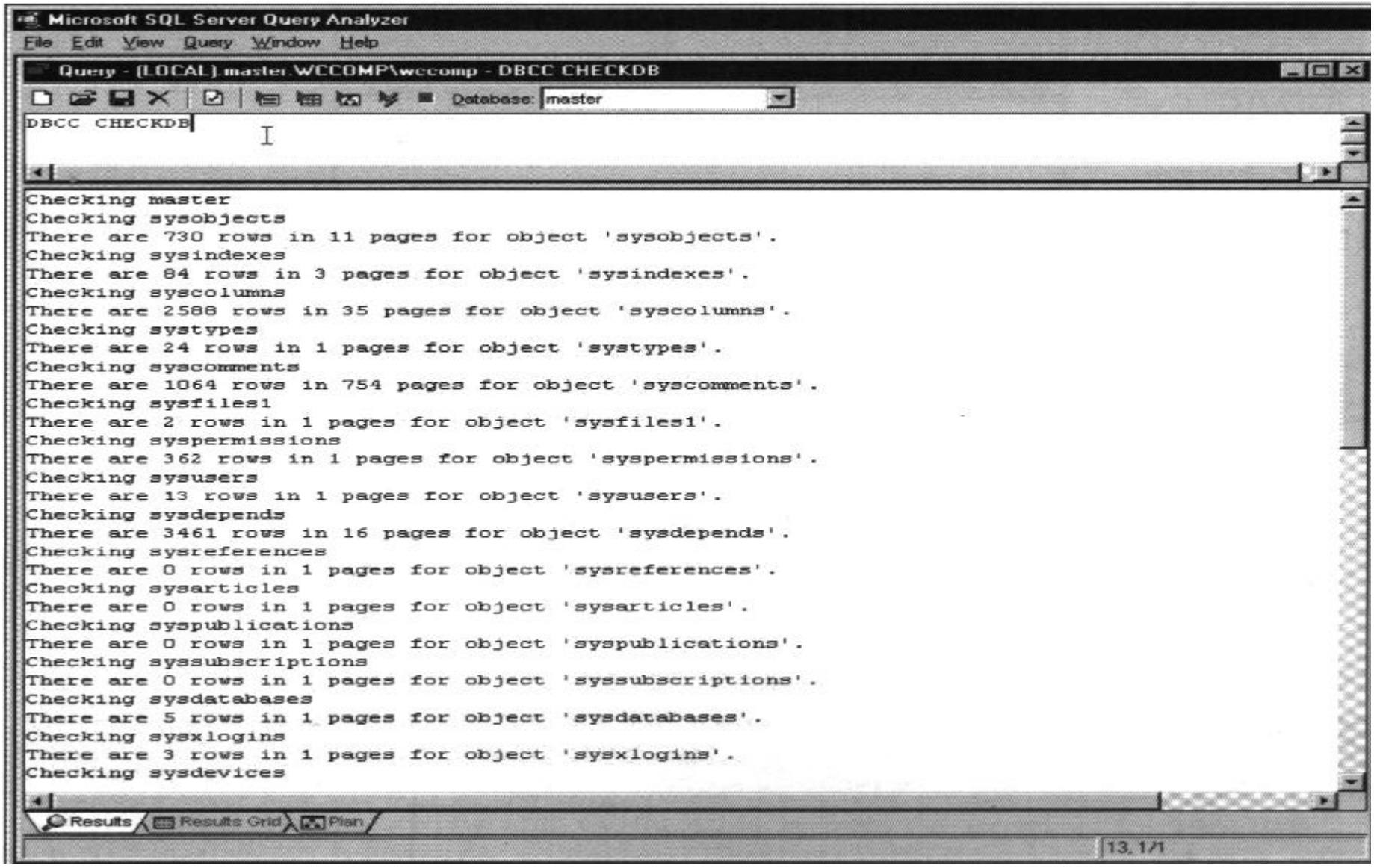

图 15.2DBCC CHECKDB **的输出**

15.2.1 DBCC 错误的修复技术

首先要向 Knowledge Base 咨询研究错误。第 14 章介绍了 TechNet 和 Knowledge Base Articles，其中包括了该主题的详细信息。另外，请参阅图 15.3，该图表示 Microsoft TechNet 中的一系列关于 Knowledge Base 的文章。

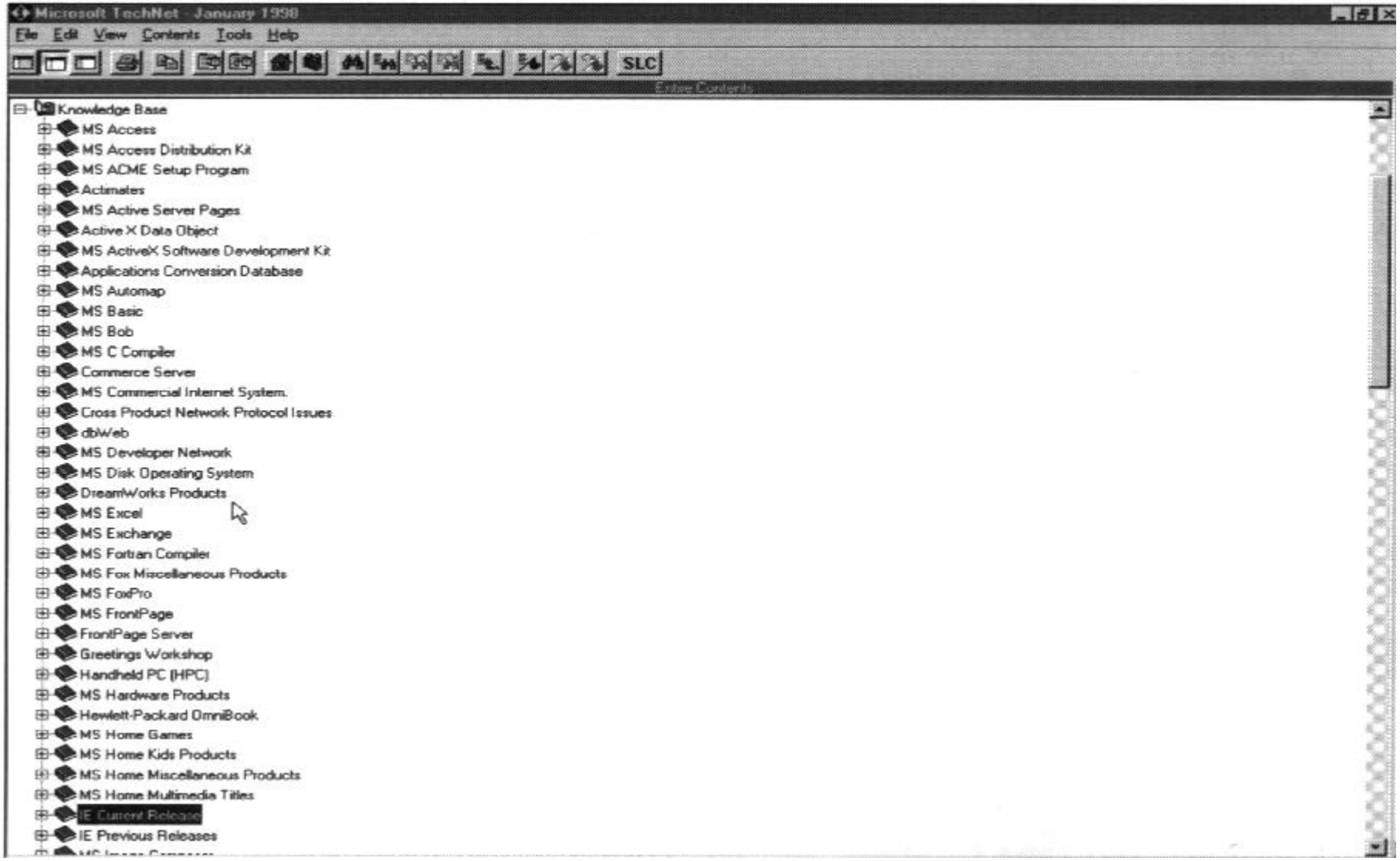

图 15.3Microsoft TechNet**中的** Knowledge Base **文章**

如果对错误的研究表明某索引已经受到损坏，修复对减低和重建该索引是

比较简单的。如果对错误的研究表明表格已受到损坏，那么经常使用 SELECT INTO 把该数据移到另一个表中可挽救该数据。旧的表格可以放弃，然后把新的表格改名为旧的表格名。除了研究任何相关的对象外，如存储过程和应该在放弃该索引之前 Scriptout()的许可外，还可以重建索引。

设法使用 BCP 实用程序把数据输出到某个操作系统文件也是很有用的。如果 DBCC 能够识别受损数据的指定行或列，最好的方法也许是删除该行，或更新该列，不要废弃整个表格。如果用户请教了 Knowledge Base，认识到自己需要附加帮助，可打电话给 Microsoft Technical Support，他们将帮助更正错误。图 15.4 是对 2546 号错误的查找以及关于该错误的 Knowledge Base 文章的结果列表。这些信息帮助用户排除在 DBCC 输出文件中可能看到的错误。

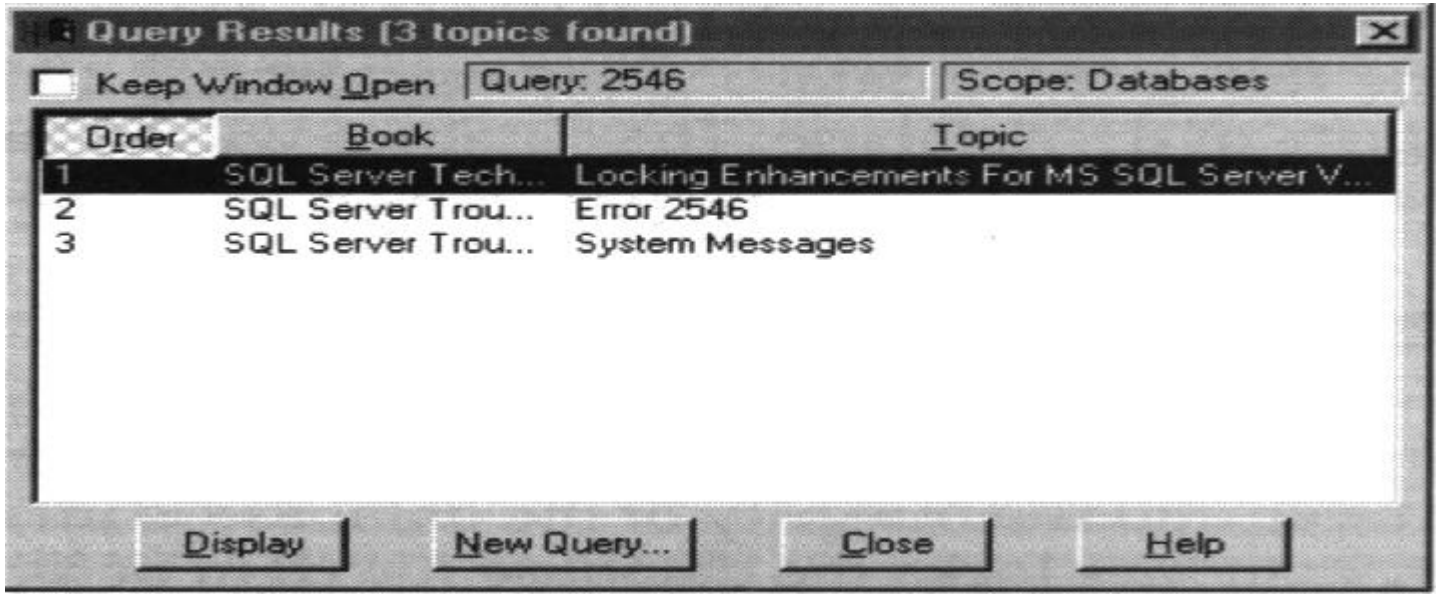

图 15.4 查找 Knowledge Base 文章

15.2.2DBCC UPDATEUSAGE

DBCC UPDATEUSAGE是可在运行Transact-SQL语句的任何地方都能够运行的Transact-SQL语句。该语句可处理要评价的数据库、表格或索引，并可为表格和聚簇索引更正Sysindexes表中的任何误差。应该运行DBCC UPDATEUSAGE的最普通的特征是，当存储过程sp_spaceused的系统返回用户所考虑的问题时的误差信息。在索引已经减小后这种情况可能会发生。用户也可以使用带sp_spaceused的自变量updateusage，该自变量在把使用信息的空间释放之前将运行DBCC UPDATEUSAGE，然而，要了解这种操作在大型数据库上要占用较多的时间。

15.2.3DBCC OPENTRAN

DBCC OPENTRAN可用来检测运行时间较长的事务，该事务可防止事务日志被截断。该自变量指示最旧分布式和非分布式的标记为复制的事务。开放式事务可防止事务日志免遭被清除，使日志保持完整，甚至使用DUMP TRANSACTION语句也不能清除该日志。当这种情况发生时，必须先了解哪个事务是有问题的，以便对问题进行诊断和修复，所以可不必使用该自变量，并使SQL Server恢复到正确的功能状态。长时间运行查询和开放事务的原因可能是缺乏优秀的编程技术员，使SQL Server存在缺陷。在发出COMMIT TRANSACTION之前，SQL Server不会清除开放的事务，如用未定的COMMIT TRANSACTION语句发布BEGIN TRANSACTION的事务。如果记录中存在开放的事务，DBCC OPENTRAN就显示系统

进程的识别号(SPID)，以便可以“清除”此进程。

如果使用 WITH TABLERESULTS 自变量，可把 DBCC OPENTRAN 语句的输出累加到一个表中。如果存在 SQL Server 缺陷或相反需要排除这种情况，分析此表可提供关于问题事务的信息，并可修复编码问题，或找到工作区。

15.3 可疑的数据库

如果 SQL Server 检测到危机数据库完整性的事件，它便把数据库的状态改为“可疑的”。此时该数据库变成不可用的，数据库内包含的数据是不可访问的。

15.3.1 恢复可疑数据库

有时复位在 Sysdatabases 系统表中的可疑状态，取消数据库的可疑标记不失为明智之举。在某些情况下，数据库是可恢复的，数据库的完整性完好如初。例如，如果在 SQL Server 启动之前，操作系统文件的备份锁住了数据库文件，SQL Server 可把关于该文件的数据库状态改为“可疑的”，因为它不能打开数据库文件。一旦数据库文件解除锁定，启动 SQL Server 可打开该文件，但数据库仍标记为可疑状态，在该状态消失之前不能访问。

这种情况的另一个例子是，SQL Server 不能完成数据库的恢复，因为该数据库或记录已经完全充满数据。使用下列代码清单可以解除可疑的数据库：

```
EXEC SP_CONFIGURE ALLOW UPDATES ' ,1
RECONFIGURE WITH OVERRIDE
BEGIN TRAN
UPDATE sysdatabases
SET status = status - 256
WHERE name = databasename
AND status & 256=256
ROLLBACK TRAN
EXEC SP_CONFIGURE ' ALLOW UPDATES ' ,0
RECONFIGURE WITH OVERRIDE
```

使用 ROLLBACK 语句可检查更新的行是否多于一行。如果只有一行被更新，可再次运行 UPDATE 语句，并使用 COMMIT TRAN 语句代替 ROLLBACK TRAN 语句，这样可确认对数据库的改变。如果受影响的行多于一行，就应保证执行 ROLLBACK TRAN 语句，并修复此处的子语句，因为一个数据库只应有一个特征名。

15.3.2 用 RESTORE 语句恢复可疑数据库

用户可以使用 RESTORE 语句尝试恢复可疑数据库。RESTORE 语句的 FROM 子语句通常指定正在用来恢复备份设备。这里介绍一种含蓄的巧妙用法。如果在发出 RESTORE 语句时还未使用 FROM 子语句，恢复开始，但不是还原。当在解除数据的可疑性和修复它之后需要恢复数据库时，可以使用此法，也可以使用这

一技术恢复由 NORECOVERY 选项存储的数据库。当移到后备服务器时也可以使用这一技术。

15.4 遵 守 标 准

预防性维护的一个更为重要的方面是，全面强化服务器的标准集。该标准集可以包括下列内容：

- 目录结构(数据库、记录、操作系统、程序、Internet 目录、SQL 备份以及应用程序相关文件的存放位置)
- 磁盘驱动器 RAID 层失灵及冗余
- DBCC 和 BACKUP 维护程序的频率
- 在 DBCC 输出文件中发现错误时的过程
- 有系统管理员(SA)的组
- 有数据库拥有者(DBO)访问的组
- 用户创建对象的命名习惯
- 字符集和分类顺序
- 客户访问
- 在整个 NT 组和域帐户中的控制中心区
- 安装的通信协议
- 安全模型

- 产品中使用的 SQL Server 和 NT 的版本
- 口令改变时间间隔
- 内存
- IIS 标准(FTP 安装标准、主目录、目录许可、虚拟根特许、安装的服务、成员服务器相对于主服务器或备份域服务器)

标准化可简化数据库的维护过程，并且有助于提高效率和组织操作。标准应做成文档，放到网络上共享。这样可节省时间，使数据库维护时间降到最低。

15.5 备 份 方 法

当事务恢复防护措施不能抵卸防护挑战的情况发生时，备份可为数据防护提供附加级别。维护当前备份文件提供第二级数据防护，是数据库维护责任的重要组成部分。

15.5.1 数据库备份

数据库备份有两种不同的基本类型，数据库备份和事务日志备份。
如果只选择进行数据库备份，则可以恢复到数据库备份的最后时刻。如果既进行数据库备份，又进行事务日志备份，则可把数据恢复到出现失灵前的时刻。后一种方法的防护性更强一些。恢复方法就是存储最新的数据库备份，然后再

存储每个事务日志备份，直到接近出现数据损坏或丢失的时刻。利用 Database Maintenance Plan Wizard 可制定数据库备份的方案(请参见图 15.5)。用 SQL Server Agent 可编制事务日志备份方案，本章在“事务日志备份”一节将对此进行详述。

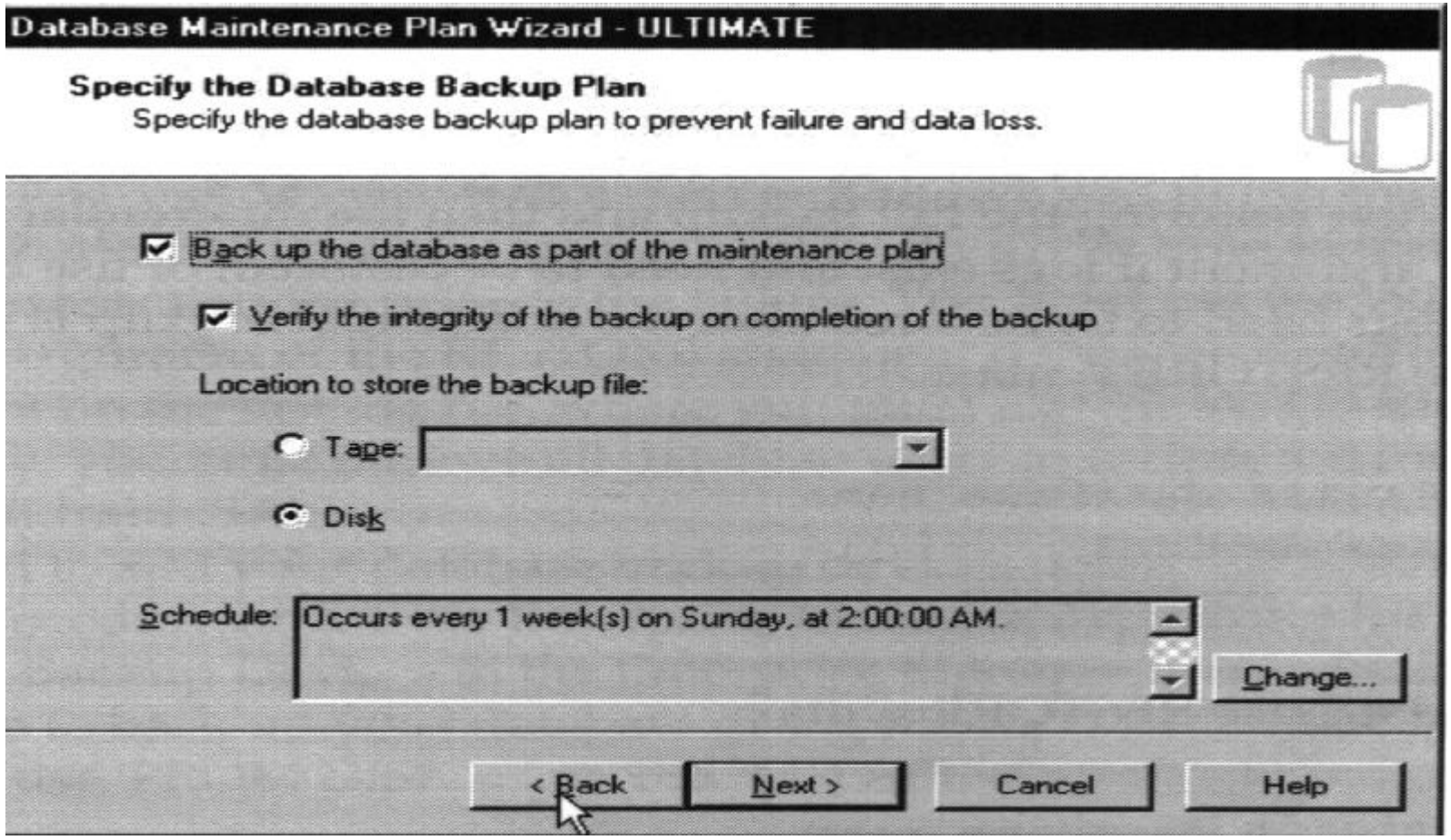

图 15.5 用 Database Maintenance Plan Wizard 制定数据库备份方案

15.5.2 差分数据库备份

差分备份是 BACKUP DATABASE 语句的变量，如果使用该变量，则只能备份从最后数据库完整备份以来发生变化的部分。这样通常使备份文件较小，对于大型数据库可节省时间和空间。使用差分备份的典型方法是，每周做一次完整备份，每天增补备份。当然，这些备份间隔可根据需要进行调整。

也可以把数据库、差分和日志备份结合起来，这样可减少恢复花费的时间。为此，应恢复最后数据库的完整备份、恢复差分备份和恢复差分备份之后创建的所有日志备份，创建的文件顺序一定要正确。

使用完整数据库备份和差分备份数据库时可利用下列语法：

```
BACKUP DATABASE database_name
TO database_device1
WITH INIT
GO
```

差分数据库备份，随后又执行下列语法：

```
BACKUP DATABASE database_name
TO database_devicel
WITH DIFFERENTIAL
```

如果后来遇到问题需要恢复，首先恢复完整备份，然后再恢复差分备份，这样可利用下面的语法恢复数据库。在恢复了差分备份后，如果存在日志需要恢复，可使用 NORECOVERY 变量，如果不需要恢复日志，可使用 RECOVERY 变量。

完整数据库的 RESTORE 语法为：

```
RESTORE DATABASE database_name
FROM database_device1
WITH FILE = 1, NORECOVERY
```

差分数据库的 RESTORE 语法为：

```
RESTORE DATABASE database_name
FROM database_device1
WITH FILE = 2, NORECOVERY
```

如果没有要恢复的事务日志备份，在有附加事务日志要恢复(用 RESTORE 或 RECOVERY)时，可使用 NORECOVERY。使用下一节概述的语法，对于事务日志备份可恢复任何可以有的事务日志。

如果未使用设备备份数据库或日志文件，但使用了磁盘或磁带文件，就必须使用 FROM=或 FROM TAPE=语法指明把数据恢复到什么介质上。关于 BACKUP 和 RESTORE 语句的完整和详细语法请参阅第 21 章。

15.5.3 事务日志备份

通过 Database Maintenance Plan 或使用 SQL Server Agent，可以运行事务日志备份。对于事务日志备份，即可以使用 Database Maintenance Plan Wizard，也可以输入 Transact-SQL 语法，只需先在 Enterprise Manager 中右

击 SQL Server Agent 并选择 New，然后从快捷菜单上选择 Job 即可。如果希望保存事务日志供以后恢复，应确保 trunc.log on chkpt 数据库选项为假。通过使用 Microsoft SQL Server Agent 的自动化作业编制功能，并选择 New Job 屏幕视图内的 Schedules Panel，可以制订要出现的事务日志备份。应设法每 10 到 30 分钟备份一次事务日志，并根据新事务的编号和希望的恢复量调整备份时间间隔。如果事务日志备份作业步骤失败，也可以对作业步骤设置警告，或向数据库管理员发送 e-mail。图 15.6 所示为 New Job Step 屏幕画面。注意 New Job Properties 窗口中的标签标明了 Schedules 和 Notifications。单击这些标签可输出管理作业的方案，设置作业失败或成功事件的注意事项。

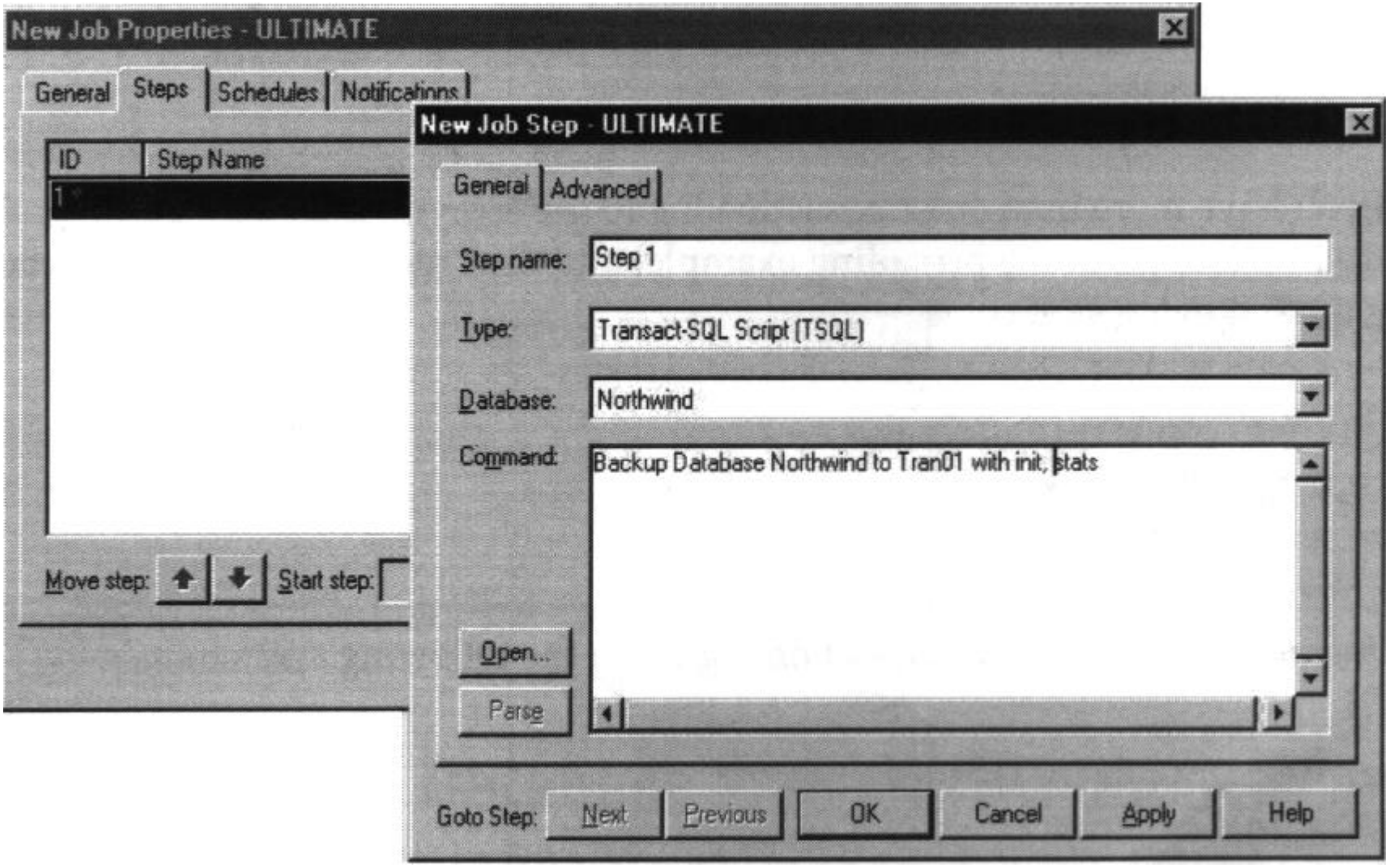

图 15.6 用于事务日志备份的 New Job Properties 窗口

如果每半小时备份一次记录，并且在时间间隔备份失败，Microsoft SQL Server 7 允许用户使用 RESTORELOG STOPAT="Mar 7, 1998 12:00AM"变量，

在任意指定时间只能恢复在备份失败前的部分事务日志。如果恢复的日志文件多于一个,除了最后一个日志文件外,必须使用 RESTORE 语句的 WIHT NORECOVERY 变量，对最后一个日志文件必须使用 WIHT RECOVERY 变量。最后，必须删除所有数据库用户以恢复数据库，并与 Sysadmin(SA)联系。

使用存储过程的 sp_addumpdevice 系统可添加数据库备份设备，或使用 Enterprise Manager扩充服务器,并在 Enterprise Manager分层树中右击 Backup Devices。如果没有使用设备备份了数据库或记录文件，并采用了磁盘文件或磁带文件，就必须使用'FROM DISK='或'FROM TAPE='语法指明恢复数据的位置。关于 BACKUP 和 RESTORE 语句的完整和详细语法，请参阅第 21 章。

要备份事务日志，可使用以下语法：

```
BACKUP LOG database_name
TO LOG_device1
WITH INIT
```

用完整数据库备份和事务日志进行恢复时，所用的语法取决于是只恢复一个事务日志，还是恢复多个事务日志。

如果只恢复一个事务日志，首先应使用完整数据库 RESTORE 语法：

```
RESTORE DATABASE database_name
FROM database_device1
```

然后只对一个事务日志使用事务日志 RESTORE 语法：

```
RESTORE LOG database_name
FROM log_device1
WITH RECOVERY
```

要恢复多个事务日志，必须首先进行完整数据库恢复(如前例)，然后再使用以下语法恢复第一个事务日志：

```
RESTORE LOG database_name
FROM log_device1
WITH NORECOVERY
```

然后再使用以下语法恢复第二个事务日志：

```
RESTORE LOG database_name
FROM log_device2
WITH RECOVERY
```

就会明白，在多个事务日志的情况下对第一个文件使用 WIHT NORESCOVERY，但要恢复最后一个事务日志，应使用 WIHT RECOVERY。

对于使用事务备份要权衡利憋。在失败时能够将其恢复，但更要涉及到 BACKUP 和 RESTORE 程序。记住，作为维护作业步骤的组成部分，每次进行完整数据库备份后，必须从操作系统中清除并移去事务日志文件，因为此时 SQL Server 不能自动删除它们。关于 BACKUP 和 RESTORE 语句的完整和详细语法，请参阅第 25 章。

15.5.4 使用多个备份设备进行备份

如果数据库非常大，如果通过网络复制备份文件，或如果想加快 BACKUP 和 RESTORE 的操作速度，可以考虑使用多台备份设备进行多卷备份(拆分备份)。

这样可充分利用 SQL Server 的能力通过 IO 并行备份到多台设备上，并可减少数据库备份或恢复所占用的时间。如果使用磁带介质并使用多台备份设备，应事先安装多个磁带驱动器。使用存储过程的 sp_addumpdevice 系统，可添加数据库备份设备，或使用 Enterprise Manager 扩充服务器，并在 Enterprise Manager 分层树中右击 Backup Device。

向多台设备备份数据库的语法是：

```
BACKUP DATABASE database_name
TO database_device1, database_device2, database_device3,...
```

从多台设备恢复(RESTORE)数据库的语法是：

```
RESTORE DATABASE database_name
FROM database_device1, database_device2, database_device3,...
```

BACKUP 和 RESTORE 语句的完整和详细语法，请参阅第 25 章。

15.5.5 备份单个数据库文件或文件组

如果要完整备份数据库，可以恢复单个文件和文件组，不只是可以备份单个文件或文件组。这表明，如果数据库的已知部分丢失，并知道其上保留的文件或文件组，可以只恢复单个文件或文件组，不恢复整个数据库。在多文件数据库遇到磁盘发生故障时，对特大型数据库(VLDBs)非常有用。在恢复文件后要保持与数据库其余部分的一致性，在最后一次完整备份开始时，必须恢复事务日志。如果使用这一技术，并且数据已经被修改，就必须备份事务日志并恢复，

以使数据库保持一致状态。这一技术不能用于增量备份。

当数据库非常大，而备份窗口又很小时，可采用这些类型的恢复技术。使用单个文件和文件组备份，一夜可备份一半文件，再一夜备份另一半文件。如果选择使用这种方法进行备份，还必须维持事务日志备份，以便使恢复的文件或文件组与数据库的其余部分具有相同的功能。

如果一个表格横跨多个文件或文件组，SQL Server 不能恢复只包含部分表格的文件组。另外，如果一个索引也横跨多个文件或文件组，必须同时恢复包含该索引的所有文件和文件组。

如果通过数据库处理文件和文件组，可考虑用文件和文件组的备份和恢复规则设计数据库的物理文件结构。

要备份单个文件或文件组，可使用以下语法：

```
BACKUP DATABASE database_name
file_or_filegroup , ...
TO backup_file , ...
WITH INIT
```

要恢复单个文件或文件组，可使用以下语法：

```
RESTORE DATABASE database_name
file_or_filegroup , ...
FROM backup_file, ...
```

保持数据库的一致性状态，还必须从最后的数据库或文件／文件组备份恢复事务日志文件，也可以从完整数据库备份恢复单个文件或文件组，不必总是处理单个文件或文件组。关于 BACKUP 和 RESTORE 语句的完整详细的语法，请参

阅第 21 章。

15.5.6 备份／恢复磁带重启动能力

BACKUP 和 RESTORE 语句的 RESTART 变量可提供重启动 BACKUP 和 RESTORE 操作的能力，只要该能力不被损坏。这种技术包括重复使用相同的 BACKUP 或 RESTORE 语句，添加 RESTART 变量。然而，这种技术有些重要的缺陷：它只能用于磁带介质的备份，以及横跨多卷磁带的备份。如果在磁带备份或恢复的中途出现意外情况，这个变量可节省时间，必要时可重新启动。

15.5.7 备用服务器

备用服务器是卸防数据丢失的另一种描述。它是完全不同的一套硬件，具有与主服务器严格一致的配置，最初通过恢复完整数据库备份保持运行，然后在事先制订的周期基础上恢复事务日志,以与主服务器保持同步(这是 Microsoft SQL Server Agent 的另一项任务)。在事务日志未运行时，可使用服务器支持特定的查询，或运行 DBCC 检查。

在发生失败时，恢复和让用户访问数据所占用的时间将会显著增加，尤其是对于大型数据库更是如此。此时必须采用某些机制用自编的应用程序代码切换到备用服务器，因为服务器名和 IP 地址都不相同。Microsoft Cluster Server

使用 SQL Server 为这种失败提供支持。

如果正在使用备用服务器，可使用 BACKUP LOG 语句的 WITH NOTRUNCATE 变量保存完整的日志，这样可允许恢复到失败前的时刻。将此事务日志恢复到服务器，但应知道任何事务在排除失败时都会涉及到。当主服务器可用时，必须把数据库恢复到主服务器上。

记住，事务日志备份得越频繁，把事务恢复到备用服务器所用的时间就越少。可使用 RESTORE 语句的 NORECOVERY 变量恢复数据库，使用 RESTORE 语句的 STANDBY 选项可把每个事务日志恢复到备用服务器，使用 RECOVERY 选项恢复来自主服务器的最终事务记录。RESTORE 语句的 NORECOVERY 和 STANDBY 选项的功能是不用检验点，所以没必要在备用服务器上的数据库中设置 no chkpt.on recovery 数据库点。

在本章中，介绍了数据库预防性维护和数据保护的概念。Microsoft SQL Server Agent 和 Enterprise Manager 可用来以自动的、调度的方式执行数据库维护任务。充分利用 Maintenance Plan 的新功能和新的作业步骤以及 SQL Server Agent 的调度功能，可节省时间。

第 16 章链接服务器和远程服务器

链接服务器和远程服务器二者都使用 Microsoft 分布式协调器(Microsoft Distributed Transaction Coordinator，MS DTC)，只是所用的方法不同而已。如果支持 OLE DB 分布式事务接口，链接服务器就使用 OLE DB 提供者和数据源，反之就调用远程服务器上的存储过程，该远程服务器使用 ODBC CALL 消除顺序或 Transact-SQL EXECUTE 语法。在远程服务器的两种机理中，因为速度更快，从而更喜欢使用 ODBC CALL 消除顺序。

本章将讨论 Microsoft Distributed Transaction Coordinator，包括通过使用分布式事务和两阶段提交，如何执行向链接服务器和远程服务器的升级。

16.1Microsoft 分布式事务协调器

Microsoft 分布式事务协调器(MS DTC)是 Microsoft SQL Server 在安装时增加到 Control Panel 的一个服务项目，它可以在跨越基于 Microsoft Windows 的网络系统中协调服务，并可升级多个 SQL Server 或 OLE DB 数据源上的数据。

MS DTC 的优点是，在事务包含来自不同 SQL Server 或 OLE DB 数据源(如果

支持 OLE DB 分布式事务接口)的许多数据源时，它可以保证事务的完整性。另外，在远程 SQL Server 上还可以调用存储过程。这项服务可以保证包含的所有数据源都能得到修改，或保证如果发生错误或整个事务的整体行不完整，其中的任何一个数据源都不能修改。

修改通过网络连接的不同的数据源意味着任何事情都可能出错。网络可能关闭、计算机也可能关机、数据库或应用程序都可能发生错误等。如果出现这些情况，应用程序必须能够保持事务完好的整体性。应用程序编程人员可能不想只为修改一个数据源而动用资金，并不因为一次失败就修改到另一个数据源。程序员不仅必须回答会计师和产品经理提出的问题，而且还要证明利用程序本身从数据整体性的丢失进行恢复是非常困难的。

如果编写应用程序采用 Transact-SQL、C 或 C＋＋,使用 DB-Library 或 ODBC，并需要在不同的数据源上的一个事务中修改数据，MS DTC 可以使事情得到简化。Visual Basic 应用程序也通过调用 Transact-SQL 存储过程，使用 MS DTC，Transact-SQL 存储过程的目的就是针对 MS DTC 接口技术的。

Transact-SQL 应用程序可以使用 Transact-SQL BEGIN DISTRIBUTED TRANSACTION 语句，或可以使 SQL Server 自动控制 MS DTC 事务。如果可以调用其他某些 SQL Server 上的远程存储过程，SQL Server 和 MS DTC 可以一起工作，自动协调引用在该事务中连续调用存储过程的所有远程服务器的参与。

如果一个远程存储过程调用了其他远程存储过程，所有引用的 SQL Server 可自动落在分布式事务的保护之中。不论远程存储过程是否使用 Transact-SQL BEGIN DISTRIBUTED TRANSACTION 语句，这种情况都将会发生。如果发生错误，

并且不能实现连续调用 SQL Server，在所有参与的 SQL Server 上整个事务都将重新运行。

即使 SQL Server 和 MS DTC 可以自动协调用 Transact-SQL 编写的应用程序的分布式事务，Transact-SQL 编程语言也不能直接访问 MS DTC 事务对象。

对于对 DTC 事务对象的高级控制，应用程序需要使用 C 或 C＋＋，以便可以选择事务协调程序，事务选项对象可以设置事务的超时值，可以给出对事务的描述。另外，通过使用异步 Commit 或 Abort 调用，使用 C 或 C＋＋允许应用程序避免阻碍调用线程。

在 NT Start Menu 上的 Microsoft SQL Server 程序组内部提供了 MS DTC Administrative Console，或在作为图形用户界面的 Enterprise Manager 中可以调用它。该控制台允许观察分布式事务的统计结果以及它的状态，控制台提供了停止和启动这项服务的能力。用户可以观察当前正在运行的所有事务、停止活动的事务或强迫事务提交或重新运行。甚至可以将事务从协调 MS DTC 服务器的事务日志中删除。MS DTC Administrative Console 的面板如图 16.1 所示。

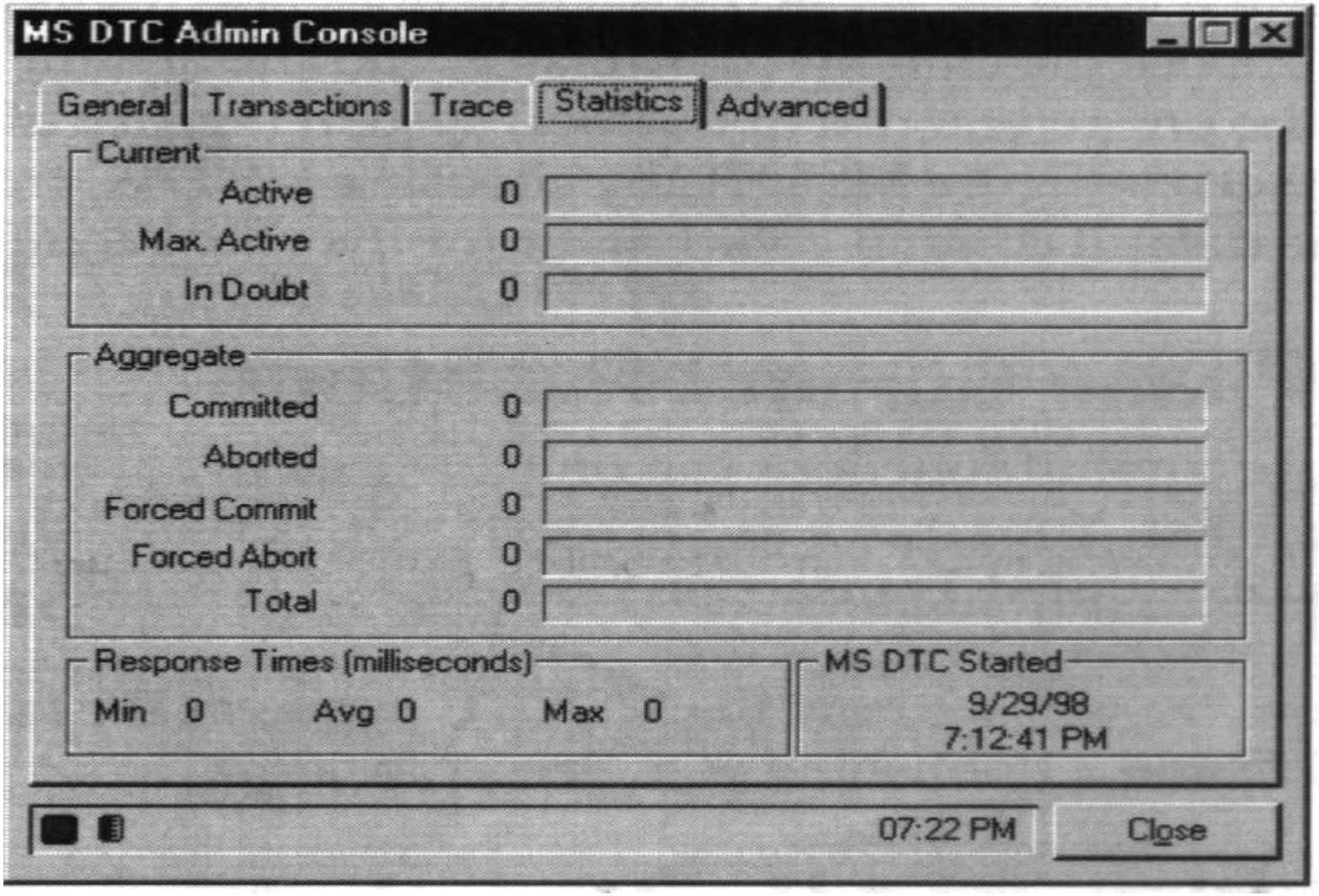

图 16.1MS DTC Administrative Console

MS DTC Administrative Console 中的 Trace 面板显示事件记录和跟踪信息，并具有通过严重等级过滤的能力。

MS DTC Statistics 面板显示活动事务的当前数量、并行事务的最大数量、已提交、终止、可疑的数量，以及已从管理上解决的事务。另外，MS DTC Administrative Console 面板上还显示在 MS DTC 已处理事务的总数、事务响应时间和对话开始的日期和时间。

MS DTC Advanced 配置面板允许用户改变 MS DTC 的刷新频率和检查点以及刷新计时器。它还允许用户指定显示的规定时间的事务和 MS DTC 记录文件的位置。

在下一节中，将介绍 Microsoft Distributed Transaction Coordinator 是如何管理两阶段提交的。

16.2 两阶段提交

两阶段提交是一个协议，当事务中包含的数据源多于一个时，它可以产生事务的细节和持续性。假设可以响应指导在网络上安装其他 MS DTC，用其本地 SQL Server 引导两阶段提交协议，则可通过用一次 MS DTC 安装的每次 MS DTC 安装对两阶段提交进行协调。

提交协调程序或上一级节点调用主 MS DTC 安装，它负责保证事务在包含的所有服务器上提交或终止，并通知应用程序该事务已经提交或需重新运行。提交协调程序总是把 MS DTC 安装在该事务发源的位置。

在客户计算机上安装了 MS DTC 实用程序后，通过应用程序可以选择提交协

调程序。如果该事务不是发源于客户机，该应用程序将不选择提交协调程序，并且缺省的提交协调程序将是在客户计算机的本地系统注册表中引用的提交协调程序。然后从 NT Control Panel 上选择 MS DTC 对 MS DTC 进行配置，并为客户机选择缺省的提交协调程序。

16.2.1DB-Library 两阶段提交功能

DB-Library 两阶段提交功能只能用于 SQL Server 6.5 和更早版本开发的应用程序。向 Microsoft SQL Server 7 的迁移应该包括计划使用 Microsoft 分布式协调器(MS DTC)，不要使用 DB-Lib 两阶段提交功能。

在下一节中，将介绍在远程存储过程内，MS DTC 在 Subscribing 和 Publishing SQL Server 之间如何管理两阶段提交。

16.2.2MS DTC、复制和更新 Subscriber

Microsoft 分布式事务协调器(MS DTC)在 SQL Server 远程存储过程调用内部，在 Subscriber 和 Publisher 之间使用 Transact-SQL BEGIN DISTRIBUTED TRANSACTION 语句来管理两阶段提交。无论 BEGIN DISTRIBUTED TRANSACTION 语句是否在远程存储过程中直接编程，使用 SQL Server 远程过程，都可自动支持 SQL Server 和 MS DTC 提交或重新运行所有引用 SQL Server 上事务的能力。

16.2.3 更新 Subscriber 和 Publisher 的冲突检测

在从 Subscribers 提交的事务中，对 Publisher 的冲突检测可使用两种方法：

- 时间戳冲突检测
- 行比较冲突检测

在已发布表中有一栏带有时间戳日期类型时，可使用时间戳进行冲突检测。Subscriber 把时间戳传递到 Publisher，将其值与该行的 Publisher 上的当前时间戳进行比较。如果这些值相同，就意味着 Publisher 上的行在复制到 Subscriber 后没有变化，该事务可以提交。

当已发布表中没有时间戳栏时，可使用行比较冲突检测。该行中所有 Publisher 栏的值都与更新的 Subscriber 进行比较。

16.2.4 对时间戳冲突检测的回环检测

事务一旦从更新的 Subscriber 在 Publisher 上进行提交，就没有必要使用复制法把变化发送到 Subscriber。只有当该列中有时间戳时，SQL Server 才使用回环检测。如果希望充分利用这一功效，可把时间戳栏添加到表格中。

16.3 分布式事务

要启动分布式事务，可使用 BEGIN DISTRIBUTED TRANSACTION 语句。要求远程服务器的链接服务器或远程服务器的分布式查询将由提交协调程序发布。这种控制服务器可自动调用 MS DTC，以便在分布式事务中支持链接服务器和远程服务器。当发出 COMMIT 或 ROLLBACK 语句时，提交协调程序，也称为控制 SQL Server 将调用 MS DTC，通过提交或重新运行来管理两阶段提交处理。

如果对链接服务器的分布式查询在本地事务内发出，并且支持 OLE DB 分布式事务接口，则该事务自动变成分布式事务。如果数据源不支持 OLE DB 分布式事务接口，则只有 SELECT 只读语句才是有效的。使用 Transact-SQL COMMIT TRANSACTION、COMMIT WORK、ROLLBACK TRANSACTION 或 ROLLBACK WORK 语句，可以完成分布式事务的自动调用，以管理该事务。

16.3.1 对话级 REMOTE_PROC_TRANSACTIONS 选项

REMOTE_PROC_TRANSACTIONS 选项可以通过使用 Transact-SQL 语句打开或关闭。当这种对话级的选项为 OFF 时，对远程存储过程调用或链接服务器分布式查询不会自动用本地事务发起 MS DTC 事务。设置这种对话级选项的 Transact-SQL 语法为：

```
set remote_proc_transactions { ON|OFF}
```

如果 REMOTE_PROC_TRANSACTIONS 设置为 OFF，若本地事务已经重新运行，用本地事务发出的远程存储过程调用就不能重新运行。当在远程服务器上的执行过程已经结束了远程存储过程时，就提交远程服务器的变化。

16.3.2REMOTE PROC TRANS 服务器配置参数

REMOTE PROC TRANS 是通过使用 sp_configure 语句控制的服务器配置。该服务器配置参数可设置为上一节介绍的 REMOTE_PROC_TRANSACTIONS 对话级选项的缺省设置。

REMOTE PROC TRANS 与其中所进行的对话级选项是非常相同的，但是它能使隐含的 MS DTC 事务对整个 SQL Server 系统有效或无效，不只是一次对话。下面的语法表示如何配置 SQL Server REMOTE PROC TRANS 服务器的配置参数：

```
sp_configure ' remote proc trans ', 1
reconfigure with override
```

用 REMOTE PROC TRANS 服务器配置参数时，1 表示 ON，0 表示 OFF。如果 REMOTE PROC TRANS 设置为 OFF，若本地事务已经重新运行，用本地事务发出的远程存储过程调用就不能重新运行。当在远程服务器上的执行过程已经结束了远程存储过程时，就提交对远程服务器的变化。

16.3.3MS DTC 和 COM

MS DTC 是 COM 体系结构的组成部分，而事务则是分布式应用程序的必要组成部分。事务的概念是用这样的方法与 COM 结合的，即自动进行安装、管理和使用。OLE 事务接口可通过使用 MS DTC 来执行。通过增加资源管理器，而不是其他 SQL Server，可以预计未来的 MS DTC 的功能性和应用范围将不断增长。

16.4 链接服务器

链接服务器是由 SQL Server 存储过程 sp_addlinkedserver 定义的虚拟服务器，用它可为执行分布式的不同查询，针对 OLE DB 数据源创建链接服务器。增加链接服务器之后，在 Transact-SQL 语句中可使用参考 linked_server_name.catalog.schema.object_name，以便参考链接服务器中的数据对象。
linked_server 是容纳数据对象的链接服务器的名称，catalog 是数据对象的目录名称，schema 是数据对象的结构名称，而 object_name 是数据对象的名称。这些名称联合起来可识别特定的数据对象。

使用下列系统存储过程在链接服务器上返回元数据：

sp_linkedserverssp_primarykeys

sp_catalogssp_indexes
sp_column_privilegessp_table_privileges
sp_columns_exsp_tables_ex
sp_foreignkeys

关于系统存储过程的详细信息，请参阅第 26 章。

当前的 SQL Server 登录时，Microsoft SQL Server 使用映射变换连接到系统存储过程。当前 SQL Server 登录对远程 SQL Server 登录映射,类似于 Remote Procedure Call (RPC)映射，只是使用的语法不同。同时，使用链接服务器和分布式查询，可在源服务器上完成映射，在处于 RPC 映射的情况下不接受服务器。

系统存储过程 sp_droplinkedsrvlogin 和 sp_addlinkedsrvlogin 可直接创建映射，并可创建许可所有用户都映射到链接服务器上的相应用户登录的缺省映射。当使用系统存储过程 sp_addlinkedserver 增加链接服务器时，可以创建上述映射。第 23 章详细介绍了关于 sp_addlinkedserver 等系统存储过程的信息。

16.5 如何定义远程服务器

为简化某个组织内各 SQL Server 之间的连接，SQL Server 可增加远程服务器。这样，SQL Server 可以从其他 SQL Server 验证连接端。网络中每个单个

的 SQL Server 通过列出 SQL Server,都可以控制对它的访问,在它的 sysservers 系统表中可接受连接。

使用下列语法可以定义远程服务器。该语法也可以用于定义本地服务器的名称。

```
sp_addserver ' remote_server_name'
```

该存储过程在远程服务器上执行，并把该远程服务器增加到 master 系统数据库中的 sysservers 系统表中。

16.5.1 调用远程存储过程

调用远程存储过程的原理是由 ODBC CALL 消除顺序和 Transact-SQL EXECUTE 语句组成的。ODBC CALL 消除顺序是最佳的执行机制。

使用 ODBC CALL 意味着使用应用程序可检索存储过程的返回代码。SQL Server ODBC 驱动程序使用经优化的协议，在 SQL Server 之间发送远程过程调用，省去了许多处理和语法分析过程，提高了运行性能。

16.5.2 远程配置选项

要用 Enterprise Manager 查看和设置远程服务器配置选项，可右击服务器，选择 Properties，然后再选择 Connections 面板，如图 16.2 所示。通过配置，

可以改变通过 RPC 远程连接的其他 SQL Server 以及在从查询返回前等待的时间秒数。缺省等待时间为 0 秒，表示在 SQL Server 重新启动前一直处于等待状态。在此之前可以终止运行。

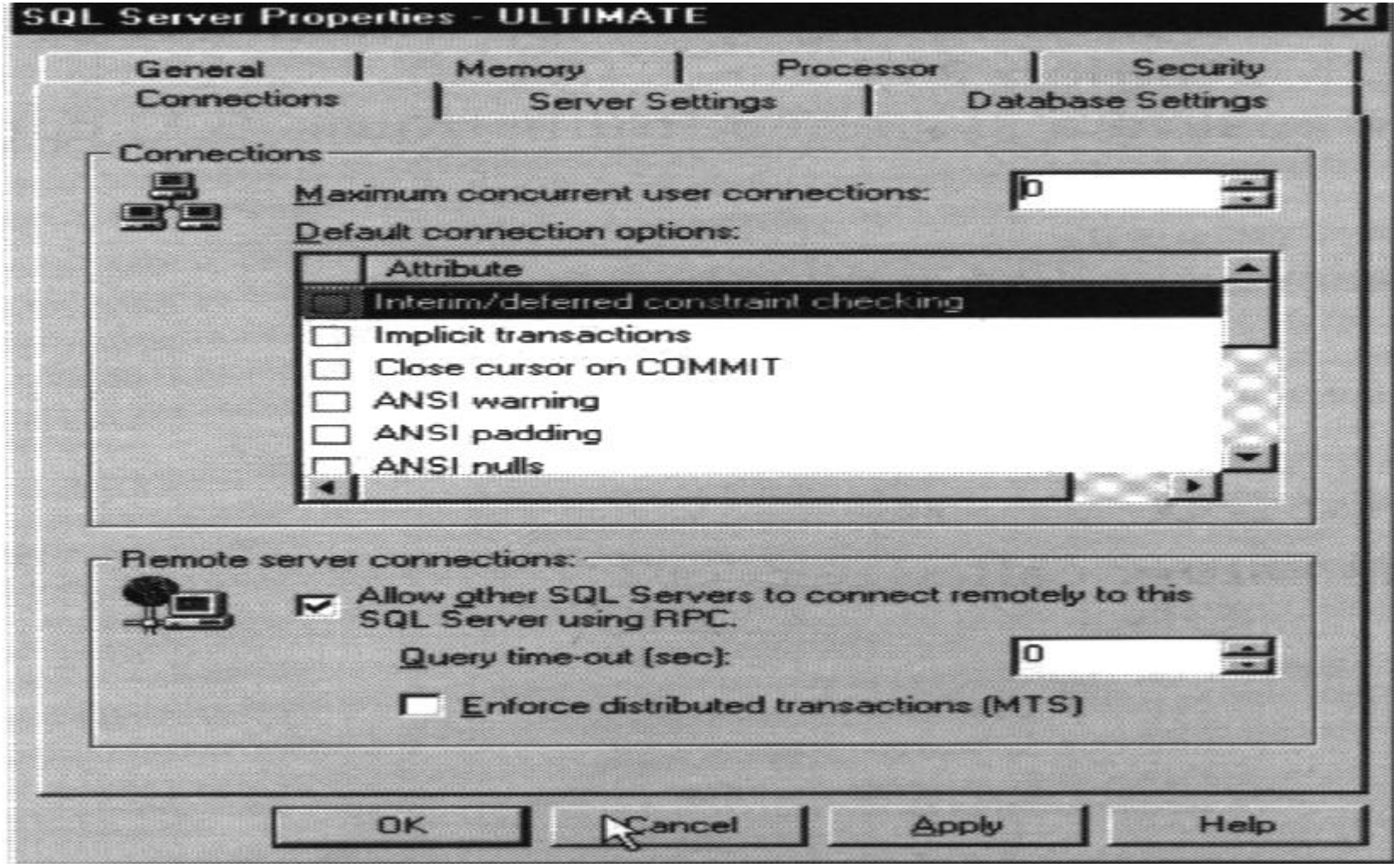

图 16.2 远程配置选项

用户也可以改变设置，强迫分布式事务用于 Microsoft Transaction Server(MTS)，并设置并行运行的最大用户。最大用户数是介于 0 到 256 之间的一个数值，缺省设置为 10，用它可以确定许可的并行连接的远程服务器的最大数量。在设置远程服务器最大数值之前，在 SQL Server 上必须许可远程服务器的连接。改变这些配置选项需要停止和启动 SQL Server。

必须在两个服务器上都改变配置选项，以使设置生效。

16.6 使用远程登录访问远程服务器

要使用远程登录访问远程服务器，必须把直接映射设置成 SQL Server 记录的专用用户 ID 和密码。然而，要映射所有用户是非常容易的，仅通过使用系统存储过程的 remoteserver 语句 sp_addremotelogin 即可完成。

使用下面的语法可直接添加远程记录：

```
sp_addremotelogin { 'remoteserver' } [, 'login' [, 'remote_name']]
```

使用这些参数如何确定登录映射的范围呢？只使用 remoteserver 参数表示，远程服务器上的所有用户都被映射到本地服务器上已有的同名登录上。本地登录与远程服务器上的登录相对应。

如果已知 remoteserver 和登录参数，远程服务器上的所有用户都被映射到给定的本地登录上。当远程服务器上的用户连接本地 SQL Server 时，它们按给定的登录进行连接。

如果除了其他两个参数外还给定远程服务器参数，那么只有在远程服务器参数中已注册的用户才能映射到本地服务器上的给定登录中。当远程服务器上的远程名称连接到本地 SQL Server 时，它们才能按给定的登录进行连接。

远程登录的缺省状态是 NT Authentication(受托安全)。如果不想进行密码检查，可使用 sp_remoteoption 系统存储过程改变密码的状态。关于系统存储过程 sp_remoteoption 的详细内容，以及如何使用，可参阅第 23 章中的 sp_remoteoption 系统存储过程。

总之，向远程服务器增加系统管理员(SA)连接的步骤如下：

在调用 Server1 的本地服务器上执行以下程序：

```
sp_addlinked server   server2
GO
```

在调用 Server2 的远程服务器上执行以下程序：

```
sp_addlinked server Server1
GO
sp_add remotelogin Server1, sa, sa
GO
sp_serveroption Server1, 'data access', TRUE
GO
```

注意：在用 7 以前的版本增加服务器时，可使用 sp_addserver 和 sp_configure 'remote access'。

使用下列语法可避开输入密码：

```
sp_serveroption Server1, sa, sa, trusted, TRUE
```

在本章中，介绍了链接服务器和远程服务器，以及它们使用 Microsoft Distributed Transaction Coordinator (MS DTC)执行两阶段提交的方法。在下一章，将介绍 SQL Server Profiler，研究查找并修复性能问题的方法。

第 17 章监测服务器性能的工具

监测 Microsoft SQL Server 对于确保避免出现性能瓶颈是非常重要的。可能出现问题的三个关键方面是性能、处理量和一致性。如果用户的硬件设备配置不正确或申请错误，查询速度就会降低，每秒钟执行的事务数也达不到期望的数值，而且所有共享用户将会造成锁定和性能降低。如果在以上三个关键方面中任一方面遇到问题，均可以改变结构选项以适应服务器，如果应用问题是由故障所致，则可以识别这种故障和修改程序。

有两个图形工具可以提供密切监视影响服务器性能活动的能力，这就是 SQL Server Profiler 和 SQL Server Performance Monitor。对这两个工具使用得越熟练，发现并解决问题的速度就会越快。

SQL Server Profile 目前已经取代了原来的 SQL Trace，前者能够提供在服务器上活动的实时数据。通过把这些数据保存到表格或文件中，可以将它们收集起来以供分析，建立有关 SQL Server 性能的原始资料。

对于这两种工具，SQL Server Profile 可提供有关数据库性能方面的详细信息，而 SQL Server Performance Monitor 则具有调度工作和超过性能阈值时发出警告的功能。

17.1SQL Server Profiler

SQL Server Profiler 是一个为系统管理员设计的应用程序调试工具,可为 Transact-SQL 语句和链接提供视图，能够识别查询、插入、数据更新及其它发生在 SQL Sever 上的事件。该工具还能监视远程访问。

注意：要执行 SQL Server Profiler,可从 NT Start 按钮下的 Microsoft SQL Server 菜单中，或从 Enterprise Manager 中 Tools Menu 中选择 SQL Server Profiler。

17.1.1 SQL Server Profile Queue

SQL Server Profiler 使用排序法保存收集到的事件,在 SQL Server Profiler 下从 View 菜单中选择 Options，将缓冲区显示器的项目数改变到控制台，如图 17.1 所示。

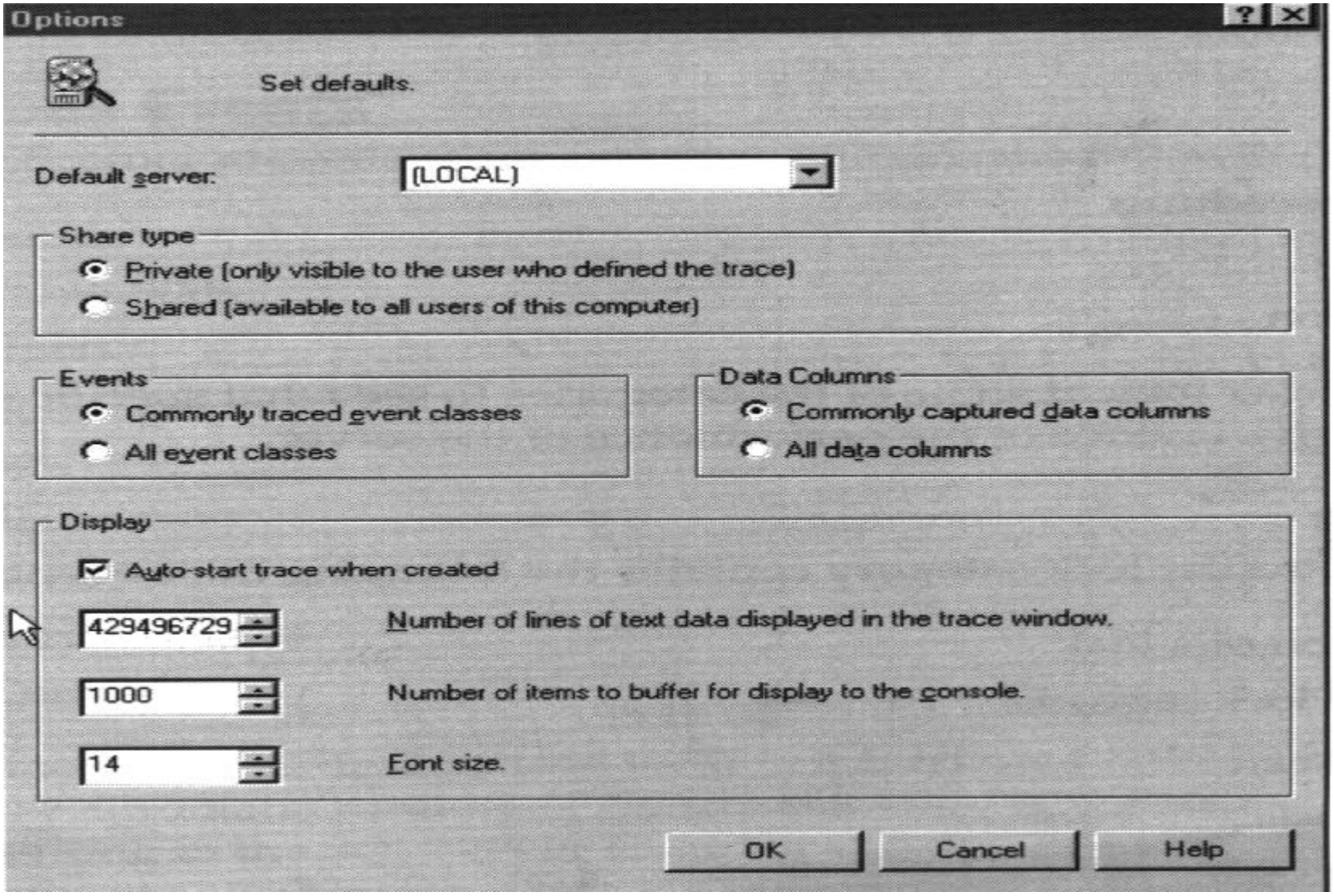

图 17.1SQL Server Profiler 的 Queue 选项

排序的作用与任何缓冲区的作用一样：即先进先出。每个轨迹都是由不同

的、称作发生器的计算机代码模型产生的，每个发生器对应一个事件种类：都是把事件按时间顺序发送到队列中。SQL Server 根据无空间就暂停发生器来控制队列。若当发生器能写入队列之前暂停时间消失，那么 SQL Server 就暂停所有的发生器，这叫做自动停止状态。在自动停止状态中，发生器仍继续收集事件数据，但不写入队列中。

要想提高线程优先权以便从缓冲区移走更多的事件和脱离自动停止状态，可打开 File 菜单，单击 Properties，从 General Panel 中选择 Trace Name，然后单击 SQL Server 下拉列表框旁边的按钮，就会看到服务器队列设置，这样就能改变这些设置，提高线性程先权，如图 17.2 所示。

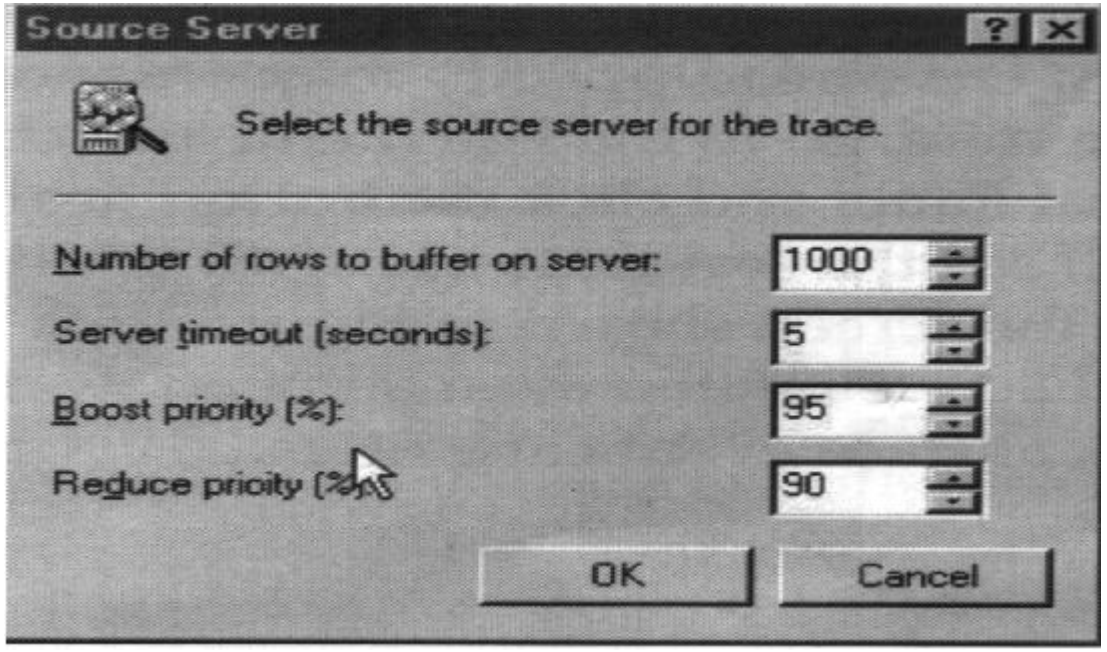

图 17.2SQL Server Profiler Trace Name 属性

由 SQL Server Profiler 跟踪的相关事件可形成不同的种类，下一节将介绍跟踪事件的类别。

17.1.2 跟踪事件的类别

跟踪事件可按下面的类别进行分组：

- Lock
- Misc
- Objects
- SQL Operators
- Scan
- Session
- Stored Procedures
- TSQL
- Transaction

下面进一步看介绍其中某些种类中可跟踪哪些特殊事件，以了解服务器的性能。

17.1.3 锁定事件

SQL Server Profiler的锁定类别包含以下跟踪事件：

- 获得锁定
- 取消锁定请求
- 锁死
- 解除锁定
- 超时锁定

当某个请求还没有正确写入时,在 Microsoft SQL Server 中通常会发生锁定问题，遗憾的是，几乎每个请求在初期都会经历锁定问题。要尽量多获取产生问题原因的信息,可以使用 SQL Server Profiler，以便更容易、更快捷和更准确地确定锁定问题中包含的代码。

17.1.4 杂项事件

SQL Server Profiler包含以下跟踪事件：

- Attention
- Auto-UpdateStats
- CursorOpen
- ErrorLog

- EventLog
- HashBail
- LoginFailed
- Recompile(NoHints)
- ServiceControl

杂项事件描述包括中断或破坏连接；有关更新的统计量自动运行的事件；SQL Server 指针打开事件；写到 SQL Server 的错误日志和事件日志的事件；转到另一个计划的杂乱操作；不成功的联机请求；存贮过程和起动系统未给出最佳线索就重新编译；以及暂停或重新启动 SQL Server。

17.1.5 对象事件

SQL Server Profiler对象类别包含以下跟踪事件：

- Object:Closed
- Object:Created
- Object:Deleted
- Object:Opened

对象的事件描述包括当打开的对象已被 SELECT,INSERT 或 DELETE 状态关闭；当某个对象已被创建、删除或存取时；当某个 SQL SELECT,INSERT 或 DELETE 已经发生时；二重性的程度；及当一个表格或索引扫描开始和结束时。

17.1.6 会话事件

SQL Server Profiler会话类别包括以下跟踪事件：

- Connect
- Disconnect
- ExistingConnection

会话事件的描述包括所有连接事件，所有未连接事件，以及在跟踪之前使用者登录的活动。

17.1.7 存储过程事件

SQL Server Profiler被存储过程的类别包括以下跟踪事件：

- SP:CacheHit
- SP:CacheInsert
- SP:CacheMiss
- SP:CacheRemove
- SP:Completed
- SP:ExecContext
- SP:Recompile
- SP:Starting

- SP:StmtStarting

存贮过程跟踪事件包括在过程存储器中发现存储过程;将一个存储过程移到或插入过程存储器中;已完成的存储过程;执行过程的局部状态;重编译或开始存储过程;及包含在一存储过程中的语句的开始。

17.1.8TSQL 事件

SQL Server Profiler TSQL 类别包括以下跟踪事件:

- RPC:Completed
- RPC:Starting
- SQL:BatchCompleted
- SQL:BatchStarting
- SQL:StmtStarting

TSQL 跟踪事件包括已完成的远程过程呼叫;已经开始的远程过程呼叫;已完成的用 Transact-SQL 语句写的程序组; Transact-SQL 语句写的程序的开始执行时间;及每一条 Transact-SQL 语句的开始执行时间。

17.1.9 事务事件

SQL Server Profile Transaction 类别包含以下跟踪事件:

- DTCTransaction
- SQLTransaction

执行跟踪事件包括 MS DTC 在数据库和开始之间的双向委托处理;委托;保存点;及重新运行处理.

如上所述，SQL Server Profiler 允许跟踪很多重要事件，当你的服务器遇到问题时，别忘了使用这个工具。下一节包含了如何去建立一个跟踪。

17.1.10 创建跟踪

要建立一个 SQL Server 跟踪,需从 Microsoft SQL Server 程序组中选择 SQL Server Profiler,当 Profiler 运行时,从 File 菜单中选择 New,然后选择 Trace。在 General 面板上的 Trace Name 对话框中为跟踪键入一个名字，从下拉列表框中选择 SQL Server 并决定是否将捕获的输出结果放到一个文件中还是一个表中，然后输入地址。单击 Events 面板，选择希望跟踪的事件，图 17.3 显示的是 Events 面板。选完跟踪事件后，选择 Data Columns 面板，再选择希望包括在跟踪输出中的任何附加列，最后在 Filters 面板中选择不希望看到的所有事件。

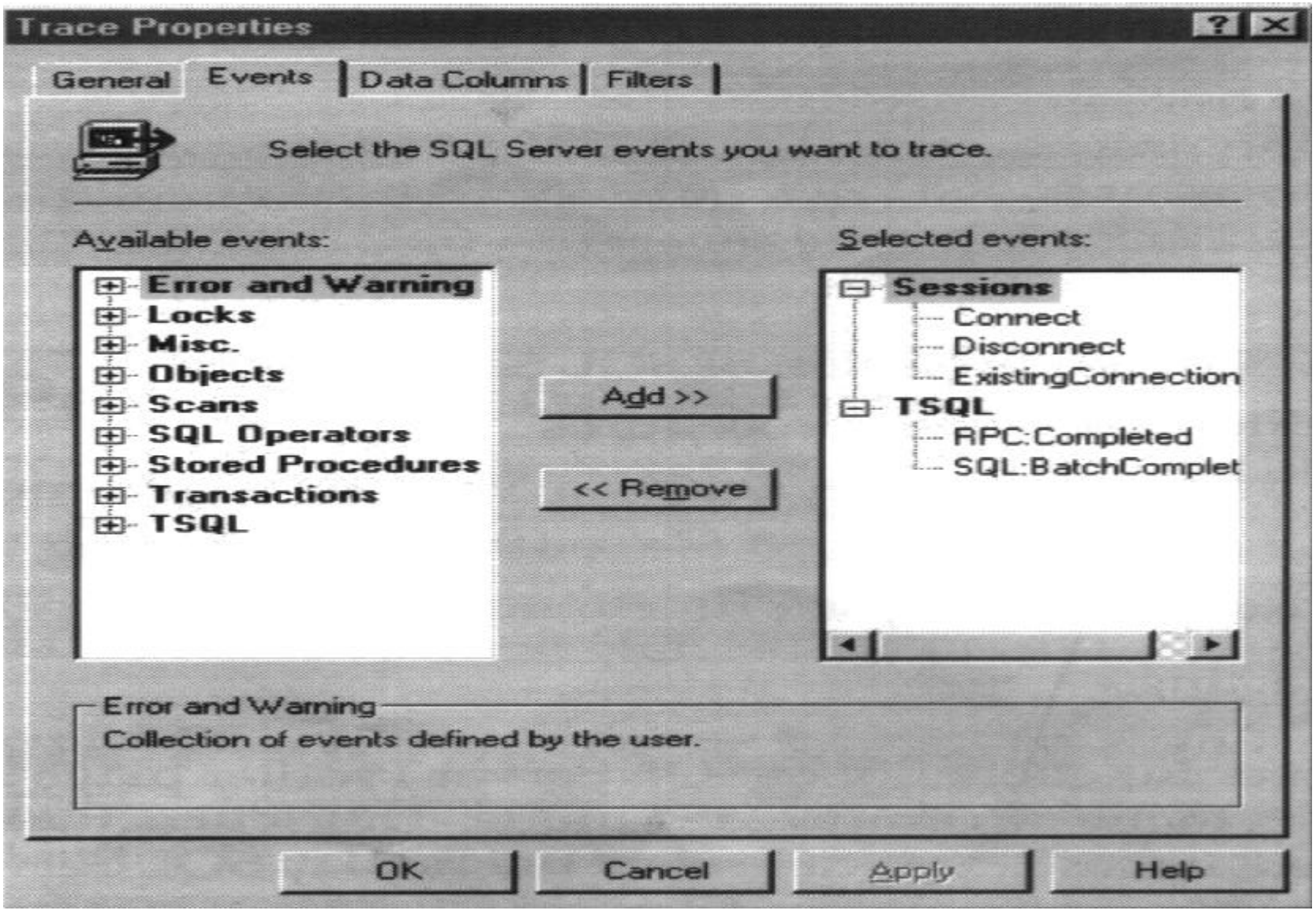

图 17.3SQL Server Profiler Trace Events 面板

选择 OK 按钮后跟踪立刻开始，将结果送到所选择的目的地。若前面没选择

文件或表，结果就传到屏幕上。

17.1.11 使用已有的跟踪

若启动一个已经创建的 SQL Server Profiler 跟踪，需从 Microsoft SQL Server 程序组中选择 SQL Server Profiler。当 Profiler 运行时,从 File 菜单中选择 Run Traces ,然后从已存在的跟踪的名单中选择想启动的跟踪。

从 File 菜单中也可以停止、暂停和删除跟踪。跟踪必须在停止后才能被删除。在关闭 SQL Server Profiler 窗口之前，所有活动的跟踪都必须被关闭。

在已经选择了 Run Traces 后，可以从 File 菜单中选择 Properties 来改变一个活动的跟踪，这个跟踪将会出现在屏幕上或输出到一个表或文件中，再选择 General、Events、Data Columns 或 Filters 面板以改变相应的性能。

若已有多个活动的跟踪而想清理跟踪窗口以看到其它的跟踪，从 File 菜单中选择 Clear Trace Window。

17.1.12 跟踪输出文件

若已在 General 面板中选择了把数据保存到一个跟踪文件中,如图 17.4 所示,那么跟踪输出将被保存在一个文件中。这种扩展名为 .scu 的文件有一些重要作用：

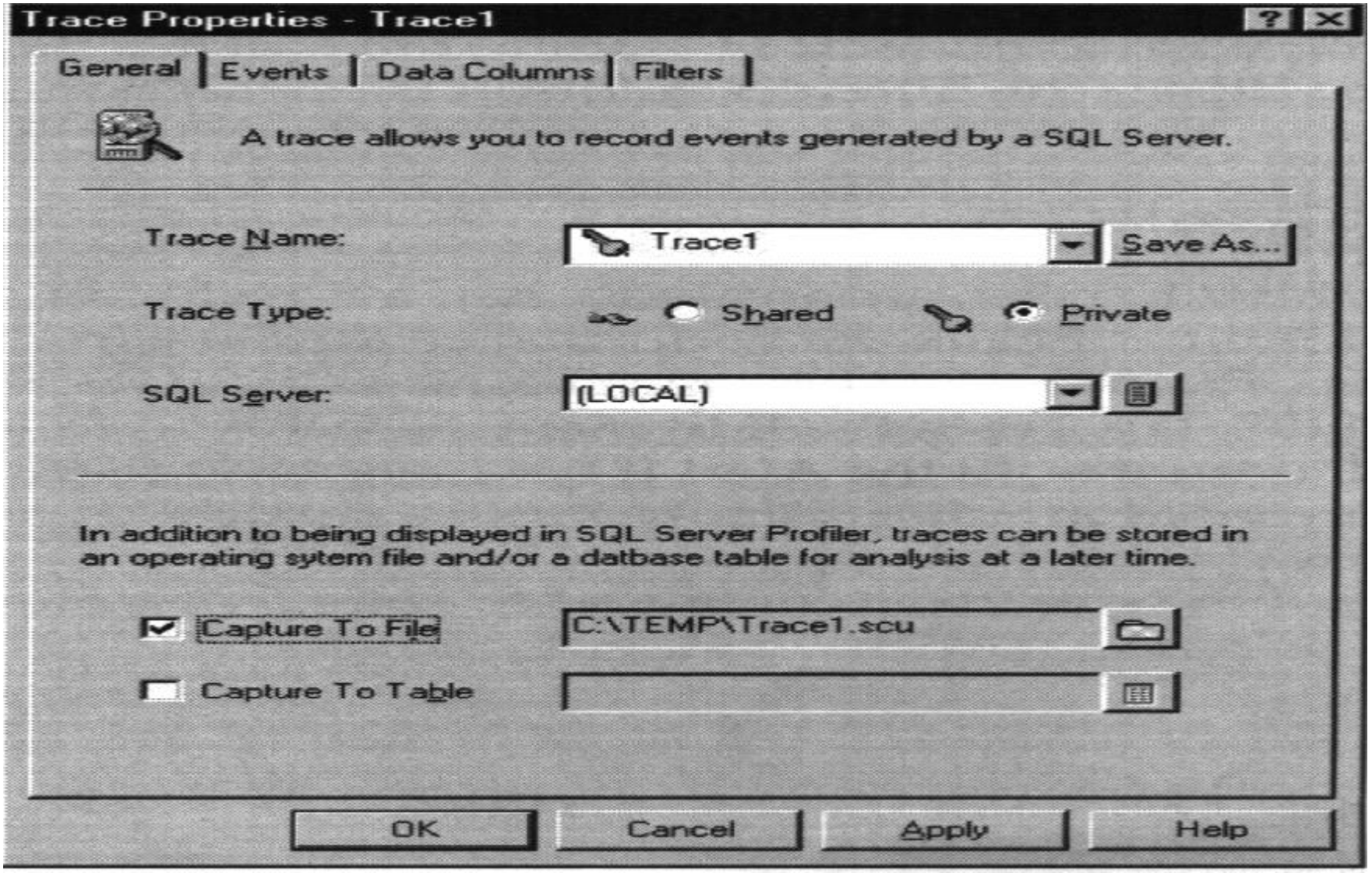

图 17.4SQL Server Profiler General 面板

- **捕获解决 SQL Server 错误的事件**

- 调试扩展存储过程
- 建立一段时间后的服务器事件的度量标准
- 辅助建立 Index Tuning Wizard 的工作文件
- 在 SQL Server Profiler 中输入到 Replay SQL Server 功能

17.1.13 重现 SQL Server

SQL Server Profiler 中的 Replay SQL Server 函数功能以实时的或压缩模式读出跟踪的捕获数据文件,这将实际上是在数据库文件应用 Transact SQL。可从 SQL Server Profiler 菜单中选择 Replay SQL Server 函数,有关 Replay SQL Server 的更多信息请看第 7 章。

17.2SQL Server 性能监视器

SQL Server Performance Monitor 也提供了监视服务器性能和活动的功能,其功能是在选择的计数器上设置阈值,当达到阈值时产生警告。

SQL Performance Monitor 在安装 SQL Server 中产生的,可作为 Microsoft SQL Server 程序组中的选择项。

17.2.1 启动 SQL Server 性能监视器

从 NT Start 菜单中选择 SQL Server 程序组，然后再选择 SQL Server Perpormance Monitor。SQL Server Perporfance Monitor 具有创建、观察和保存功能的图表。用户也可以选择对象和计数器并把它们添加到图表中，不同颜色的图形代表不同的计数器。要使图表较小并且可读，可创建不同的图表来监视不同的统计类型。

从 SQL Server Performance Monitor File 菜单中选择 New Chart，然后选择 Edit 菜单和 Add to Chart，此时有许多计数器可供选择。通过把计数器添加到图标中可选择希望监视的计数器。这一任务完成后，监视不同类型统计的图表中就会出现不同的颜色。

17.2.2 生成警告

本节解释当在 SQL Server Performance Monitor 中超出阈值时如何生成警告信息。从 Microsoft SQL Server 程序组中选择 Performance Monitor，再从 Performance Monitor View 菜单中选择 Alert，出现 Alert Log 屏幕后，从 Edit 菜单中选择 Add to Alert，然后选择感兴趣的对象和计数器。在 Alert If 框中选择 Over 或 Under 单选按钮，并输入一个阈值，最后在 Alert 框中的 Run Program 中输入 sqlalrtr70 命令。

Sqlalrtr70 命令格式如下：

```
sqlalrtr70 -E error_number [-S server_name] [-P password]
[-D database_name] [-V severity] [-T]
```

语法的前半部，-E 项后面的错误号是多少，作用在服务器上的错误号就是多少。Sqlalrtr70 中的参数-E、-S、-P、-V 和-T 是区分大小写的，所以要确保按大写方式输入这些参数。也可以选择 First Time 或 Error Time 运行警告。

到了这一步还没有完成。要想从 SQL Server Performance Monitor 中的阈值产生警告，需建立两种不同的警告，除了前面描述的 SQL Server Performance Monitor 警告外，还须建立一个 SQL Server 警告。

要建立 SQL Server 警告，首先要使用 Enterprise Manager 建立一个用户定义的事件错误信息以扩展到服务器上，然后从 Enterprise Manager 的分层树中选择 SQL Server Agent 和 Alerts，在 Result 面板中双击警告，或右击它再选择 Properties，单击浏览按钮（...）出现 Manage Server Messages 对话框，如图 17.5 所示。

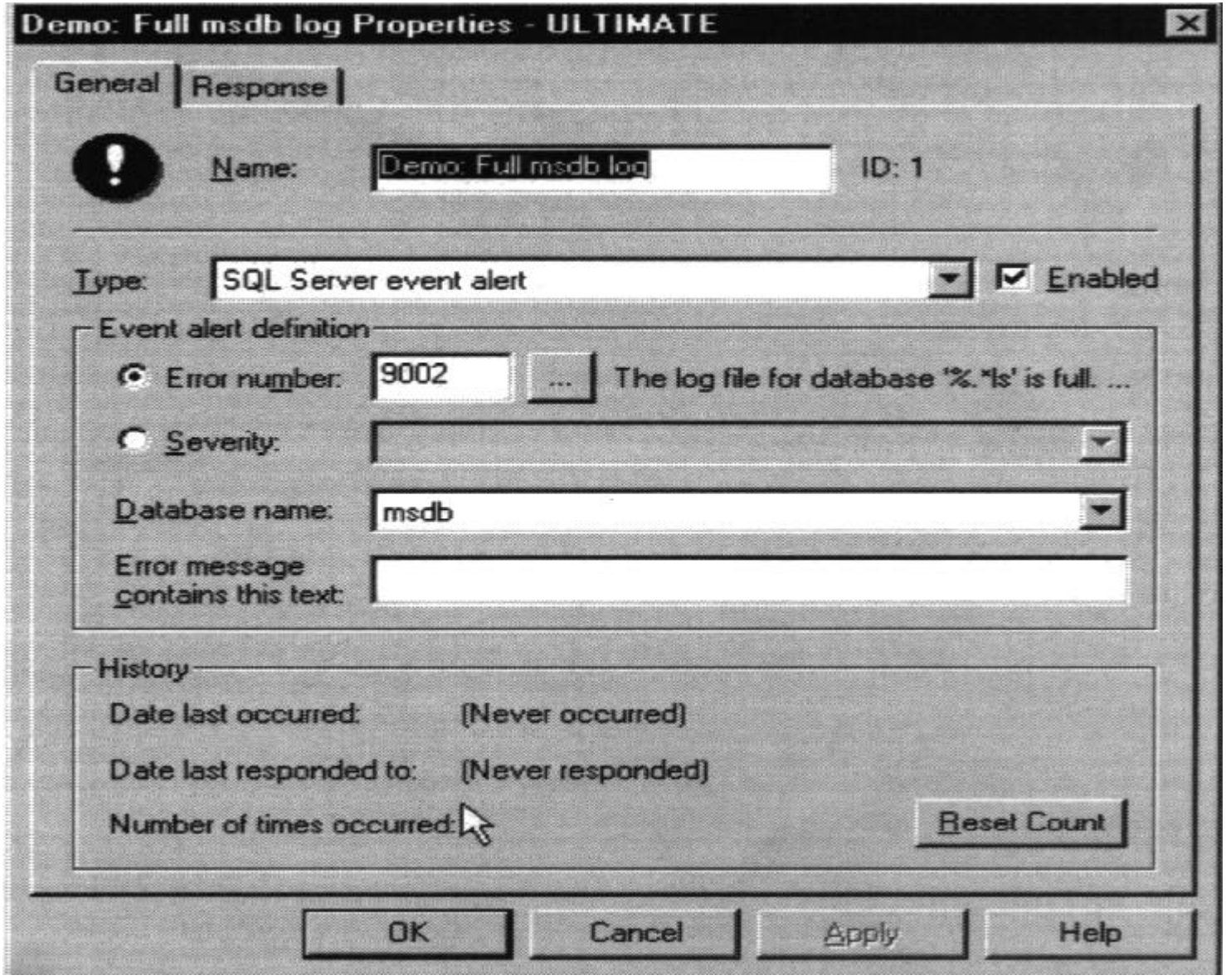

图 17.5 创建用户定义的错误消息

单击 New 创建一个用户定义消息，当 Performance Monitor 产生错误时，可使用这一消息产生一个 SQL Server 警告，输入与在 Performance Monitor 中的 Run Program on Alert 框中的 sqlalrtr70 命令中要使用的错误号相同的数，用户定义消息中的错误号值是从 50001 开始，SQL Server 保留小于 50001 的数值。

最后一步是建立一新的 SQL Server 警告，当错误产生时这一警告会响。要用用户定义的错误号建立一个警告，需在 Enterprise Manager 的分层树中扩展到服务器，再扩展到 SQL Server Agent，然后右击 Alerts，选择 New Alert。为警告输入一个名字，然后选中 Error Number 按钮，单击浏览按钮（...）选择刚建立的那个错误号，也即在 sqlalrtr70 命令中出现的值。警告也可由缺省值产生。若想让这警告只在一个数据库中起作用，必须从这屏幕上选择这个数据库，如图 17.6 所示。

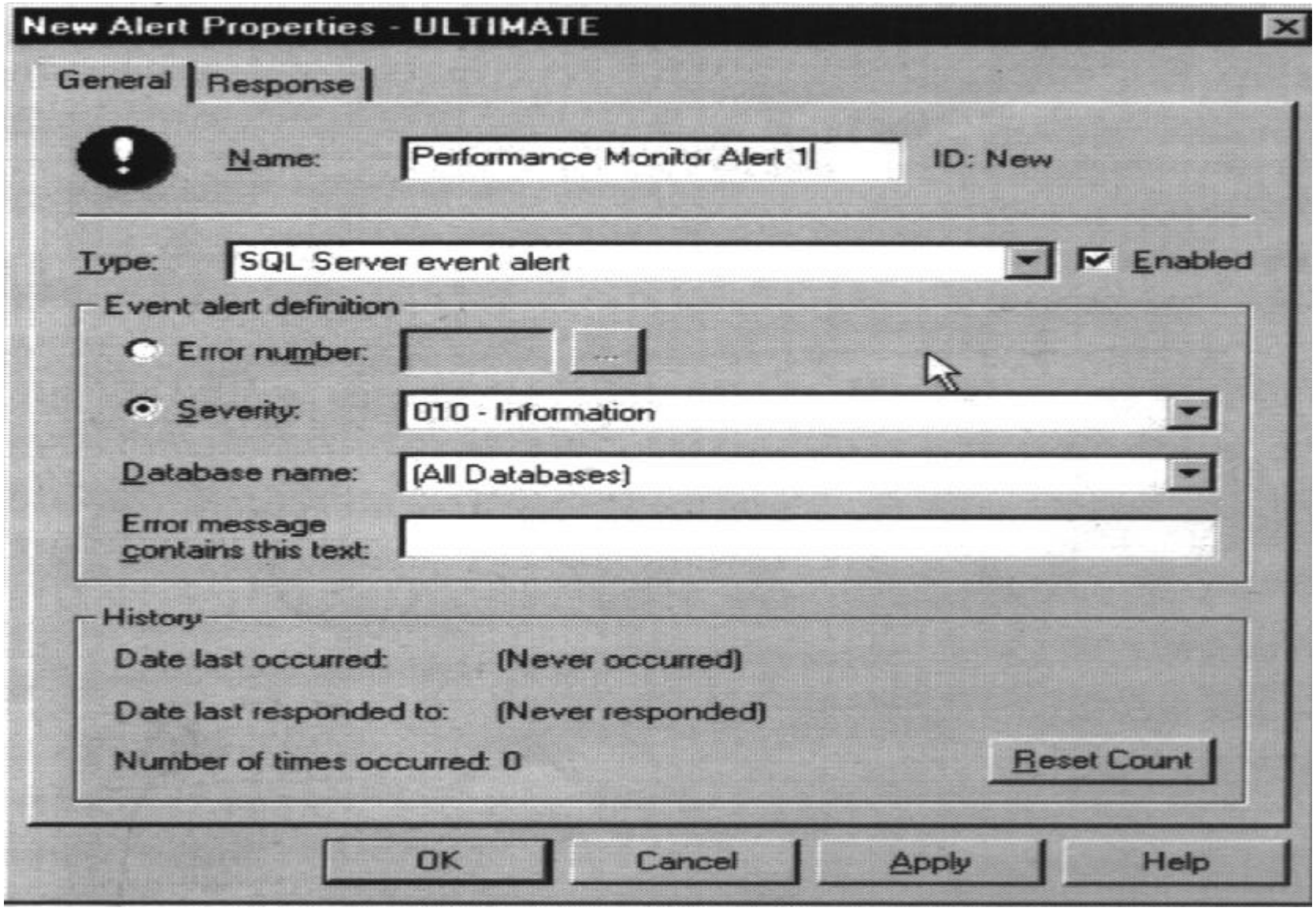

图 17.6SQL Server Agent— — New Alert 属性

选择 Response 面板，指出当警告发出响声时通知谁、通知什么和如何通知。

SQL Server Performance Monitor和SQL Server Agent服务使用必须为警告发出而运行。

当超过SQL Server Performance Monitor的阈值时，SQL Server Performance Monitor的警告就产生了，它也启动了sqlalrtr70应用，这一应用与SQL Server Agent、isql应用及SQL Server Performance Monitor都有联系，isql应用连着SQL Server并运行RAISERROR WITH LOG Transact-SQL语句，这将把事件写到Windows NT应用记录中。当建立新的SQL Server警告时，SQL Server Agent会注意到这一事件，SQL Server警告发出响声并通知在Response面板中选择的人。

17.2.3 对象和计数器

SQL Server Performance Monitor从一个属性对象中聚集起来的性能计数器中收集信息，SQL Server计数器对监视器的好处在于：

- Access Methods Object-Page Splits／sec由于索引页太满引起。
- Access Methods Object-Table Lock Escalations／sec每秒对表格的锁定强度所扩大的次数。
- Access Methods Object-Extents Allocated／sec每秒由存储数据的数据库对象所分配的范围。
- Access Methods Object-Extents deallocated／sec每秒钟由数据库对象存储的数据未分配的范围。

· Access Methods Object-Worktables Created／sec 在前一秒建立的工作台的个数。
· Buffer Manager Object-Buffer Cache Hit Ratio 在内存中的缓冲区中发现的而不是从磁盘上读一个请求的次数的比率。这一比率应趋近于 100%，数值越高，服务器的性能越好。
· Buffer Manager Object-Checkpoint Writes／sec 通过检查点每秒钟写到磁盘上的脏页。
· Buffer Manager Object-Page Reads／sec 每秒钟物理页的读数和通过所有数据库每秒钟。
物理页的总读数。如果这个值高的话，在索引、查询效率和数据库设计方面的申请会受到检验。
· Buffer Manager Object-Readahead Pages／sec 每秒物理 I／O 请求的数目，需要提前读和提前得到这一页。
· Cache Manager Object-Cache Hit Ratio 在内存的数据区中发现的而不是从磁盘上读一个请求的次数的比率。这一比率应趋近于 100%，数值越高，服务器的性能越好。
· Databases Object-Transactions／sec 每秒钟执行 Transact-SQL 命令批的数目。每秒执行速度高表明有好的信息通过量。
· Databases Object-Active Transaction 作用于数据库上的有效的执行数目。
· Databases Object-Log Growths 日志必须要增加尺寸的次数。
· Latches Object-Average Latch Wait Time 当计数器需等待时以毫秒计的

等待时间(锁存器是新的版本 7 锁定范例的组成部分)。
· Locks Object-Number of Deadlocks／sec 每秒钟死锁的次数。
· Replication Logreader Object-Delivery Latency 日志阅读器的延迟秒数。
· Replication Logreader Object-Delivery Rate 每秒插入的日志阅读器命令的数目。
· Replication Merge Object-Conflicts 计算更新用户中出现的冲突。该值大于零表示失败，必须控制该冲突。
· Replication Snapshot Object-Bulk Copy Rate 每秒块拷贝的行数。
· General Statistics Objects-Logins／sec 每秒注册数。

17.2.4 性能因素

SQLServer Performance Monitor 可能会降低服务器的性能，这取决于它运行时间的长短、它正在监测的内容以及所选用的硬件平台、计数器的数码和更新间隔。

17.2.5 需要连接到 SQLServer 的许可

要确认运行 SQL Server Performance Monitor 的 Windows 帐户已经访问连接到 SQL Server，并成为 SQL Server Public 角色的成员。关于 Windows NT

第四部分　开发——编程语言

第 18 章开 发 话 题

在本章中，主要介绍与 SQL Server 7 应用开发相关的各种话题。在此版本中，行级锁定和死锁的可避免性、锁定的增强和更优化的查询功能，将直接为开发人员带来利益。另外也可以将 SQL Server 与其他可以用来调试存储过程的 Microsoft 开发工具，如 Visual C++和 Visual Basic 结合使用。用户可以使用 Samples 目录下的样本程序。SQL Server 7 即安装在这个 Samples 目录下。

本章第一节中将讨论一项非常实用的功能——创建数据库图表。数据库图表是数据库的映像。它对于那些希望继承系统和在完成开发过程后需要进行数据维护的用户是非常重要的。

18.1 数据库图表

Microsoft Visual Studio 中的 Visual Database Tools 能用来创建数据库图表。Database Designer 有一个特殊的用户接口。用此接口，不仅可以改变 Microsoft SQL Server 数据库结构，也可以改变 Oracle 数据库结构。对数据库管理员而言，在准备通过由 Database Designer 创建的脚本之前，使用“What

if”设计方法，可以不影响当前的设计，数据库对象的关系、索引、约束和修改，以及数据库定义语言的改变，都不必逐行编写 Transact-SQL 代码，即可实现。

Visual Database Tools 正在融入 Enterprise Manager 之中。随着时间的推移，这种融合将越来越多。在没有数据库图表的情况下去设计一个复杂的数据库是非常困难的。因此，要保持复杂数据库图表的先进性是十分重要的。

18.2 数据完整性

加强数据库的数据完整性有许多迫切的原因。Microsoft SQL 为这种完整性提供了许多途径，以保证数据质量。数据库的数据完整性包括以下四个方面：

- 实体完整性
- 域完整性
- 参照完整性
- 用户定义完整性

下面将分几部分对上述几方面分别进行解释，并阐述其实现的方法。采用这些方法实现数据库完整性，不仅能够保证数据质量，而且能够避免由于缺乏数据库完整性而导致的费时费力的数据审查和随之而来的数据清理问题。

18.2.1 实体完整性

必须确保表中的每一行都是独立的。每一行都能够被设置指针，并且能够在不干扰其他行的情况下被访问，这些是很重要的。独立形式的实体完整性是通过建立主关键字约束，唯一索引、唯一约束或 IDENTITY 特征来实现的。

主关键字是表中的一列或多列，其所具有的值对表格中的其他各行而言必须是唯一的。每个表格只能有一个主关键字，并且主关键字不允许存在空值。主关键字被创建后，SQL Server 通过以主关键字建立唯一索引来实现表格的唯一性。通过给表格设定主关键字，不仅实现了表格中行的独立，而且访问设有主关键字的表格，其数据的检索也更加快捷。

创建一个 identity 列是一种更为复杂、更为先进的强化独立性的方式。主关键字列常常用 IDENTITY 属性来定义，这样做的好处是:当用户插用行时，SQL Server 会同时自动给被插入行的每列赋值，用户不必知道插入行的每个值，也无须存储。当索引是整数时，数据的访问更加快捷，因而功能也相应增强。

在使用 IDENTITY 属性的情况下插入行，而插入失败，则插入行的值不会在表格中出现，下一行的值仍将递增。例如，插入行 10，而由于某种原因使插入失败，数字 10 将不会在表中出现;下一次插入成功时，行 9 后出现的将是行 11。然而，通过 SET IDENTITY_INSERT ON 命令，可以控制表格中的数值，人为地给插入的行赋一个整数值(该整数值须在表格中未被使用过)。可以使用 SET IDENTITY_INSERT OFF 命令来结束操作，使用 DBCC CHECKIDENT 命令来检验和改变 IDENTITY 值。

每个表格只允许有一个主关键字，而用户有显时需要非关键字段中的唯一值。此种情况下，可以使用 UNIQUE 约束来给非主关键字段建立唯一索引。如果用户插入一个已在表格中存在的行，SQL Server 将给予错误提示，告知用户一个相同的行业已存在。以此防止相同行的插入。如果用户想采用缺省方式，则主关键字约束可以使用 CLUSTERED 索引，UNIQUE 约束可以使用 NONCLUSTERED 索引。NONCLUSTERED 索引现使用聚簇关键字，而不再使用行标识符。因此当主关键字很长时，用户可以指定 NONCLOSTERED 索引。

18.2.2 域完整性

域完整性指:对一个给定的列而言，只有一定范围内的数值是有效的。通过规定数据类型、使用 CHECK 约束和规则、使用 DEFAULT 值、NOT NULL 定义以及 FOREIGN KEY 约束，可以限定有效值的范围。

数据类型用于定义可以放置于列、局部变量和存储过程参数中的数据的类型。用户可以直接使用 SQL Server 提供的数据类型，亦可以自己定义数据类型。数据类型对填入某列的数据加以限制。例如，字符型数据不能插入到以数字型数据创建的列中。CHECK 约束对插入某列的数值或数值的格式加以限制，例如电话号码和邮政编码。表格中的一列可以有多个 CHECK 约束。用户亦可随时给新创建的或业已存在的表格增加限制。

在 SQL Server 7 中，RULES 被用以反向兼容，它与 CHECK 约束起相同作用，区别在于每一列只能有一条规则。SQL Server 7 中，一般使用 CHECK 约束而不

使用 RULES，因为 CHECK 约束在表格创建或变更同时自然产生，而 RULES 则需要单独创建。

有时，用户可能不知道确切的数字;或者数据中带有空值，而表格中的该列不允许使用空值，或用户本身不希望出现空值。在这些情况下，可以使用缺省值。缺省值可以在创建或变更表格时用 DEFAULT 关键字来定义。例如，用户想对某列中的数值取平均值，此时，如列中出现空值，则该命令不能被执行。此种情况下，可以给每行中的每一列空值上赋一缺省值 0。这样，平均运算即有效，且包含了表格中每一行的数字。

在创建表格的同时定义某一列为 NOT NULL，能够防止该列被赋以 NULL 值。这是确保数据完整性，特别是域的完整性的另一种方式。列中的 NULL 意味着该值是未知的。NULL 的运行方式不同于 0 值或空值。因为它们不能相互比较，而且也不相等。在使用 Transact-SQL 命令时，NULL 将引起许多麻烦。因为在遇到 NULL 值时，每条命令都将不能正常运行。

FOREIGN KEY 约束与主关键字或唯一约束并同使用，可以确保被插入列中的值也被包含在另一个表中的主关键字之中。它不仅被用以确保域的完整性，而且对下文将讲述的参照完整性也很重要。

18.2.3 参照完整性

使用参照完整性意味着要维持两个表格之间的关系，并且数据库中不允许孤立行的存在。也就是说，SQL Server 不允许用户在子表中添加父表中不具有

的行;同时，如果一行在其他表格中有子行时，也不允许对该父行进行删改。

SQL Server 通过上文讲述的 FOREIGN KEY 约束和 CHECK 约束来实现参照完整性。

如果在标准的 SQL Server 确保完整性的工具中，没有能够满足需求的，用户也可以以 CONSTRAINTS、TRIGGRES 和 STORED PROCEDURES 形式定义用户定义的事务规则。这些形式将在下一节中介绍。

18.2.4 用户自定义完整性

用户自定义完整性是指用户可以为某一个表格或某一列设定自己的特殊的事务规则。触发器和存储过程用户可以以更为复杂的方式实现这一操作。使用 Transact-SQL 语言中的流程控制语言(如 IF 语句)可以控制插入表格中或某一列中的元素。这是一种较为复杂的方式。书写存储过程和触发器的方法将在第 19 章中阐述。

18.3 设计数据库

设计一个数据库比我们所想象的要简单。注意不要将这一过程复杂化。设计数据库有几条实用的规则，在用户自己设计过几个数据库后，用户即可以凭直觉遵循这些规则。设计得再完善的数据库也有其自身的缺陷，因为它们的设

计者——人本身便是有缺陷的。我们不可能预见到明年人们将以何种方式去访问数据，也很难预测商业需求的变化。所以，只需大胆着手，做出我们最好的设计就行了。关系型数据库的一大优点是它易于修改，这一点区别于以往的网状或层次数据库。

建立数据库的三大要素是：

- 实体
- 属性
- 关系

实体可以是一个人、一个地方、一件物品、一个抽象的事物或其他任何可以包含一定信息的用名词表示的东西。这样的例子很多，每一个例子都用一个独立的关键字表示。这些名词都对应着表格，并且用单数名词表示。例如，应说“客户”，而非“客户们”。另外，常会用到一个“相关实体”的概念，它指实体以常见的关系组合在一起。

一个实体可能包含多个属性，其中包括可能作为主关键字的属性。含有空值、遗漏值或者 NULL 值的属性不可能作为主关键字，而被称为备用关键字。

属性是每一个实体的各种事实和特征。每个实体都包含着作为主关键字特征的属性。其他类型的关键字包括候选关键字、备用关键字、外部关键字和代理关键字。其中代理关键字是具有单一属性的关键字，是大型、组合型(含多个列)的关键字。代理关键字使得 Transact-SQL 程序更加简便易行，因为用户在查询时，不必逐一在编码中写出每一个列。

关系是数据库设计的核心，它包含着事务规则。关系是指两个且仅两个实

体之间的关联。它是以相关实体的组合形式表现出来的。在了解关系之前，需要先了解基数的概念。基数是指:父表中的多少实例与子表中的多少实例相关联。包括一对多、多对多;标识性一对多、标识性多对多;以及回归性一对多、回归性多对多。

多对多关系亦称非确定关系。非确定关系必须被分解。关系亦可看作是实体，因而关系的分解可以通过表格的连接来实现。

当父关键字是子表关键字的一部分时，即产生标识性关系。当父关键字不是子表关键字的一部分时，非标识性关系产生。

回归关系是一种非标识性关系。在这种关系中，父表与子表的实例相同。

18.3.1 数据库设计的方法

数据库设计可以通过三种方式完成:

- 自上而下
- 自里而外
- 自下而上

自上而下的设计方法即按事件的时间发展顺序设计表格。例如，按照需求的收集、逻辑模式设计以及物理设计的顺序。这种方法在需求定义时期使用。而在外部环境快速发展的今天，这种费时费力的方法已不再适用了。有些时候，我们需要就手头现有的材料进行设计。这些材料可能仅仅是一个有限的调试，或一个不能过多改变其界面的遗留系统。这时，就要采用其他的方法。

如果需要安装一个现存的软件包，则自上而下的方法比较适用。如果需要修改或扩展现存的系统，则自内而外的方法比较适用。

在下一节中，我们将讨论一个比较有争议的话题:规范化。正确的事务规则的实现和数据库的性能比起一个完善的、规范化了的数据库重要得多。切勿为了追求数据库规范化而忽视了其运行和功能。

18.3.2 规范化

数据库将如何运行，是由数据库、表格和表格间关系的逻辑设计决定的。规范化技术即将数据划分为不同表格的规则。适当的规范化有利于数据库的运行。要注意，规范化的程序越高，连接越复杂。而过度复杂的关系连接需要大量的表格，这将对数据库运行起消极作用。系统的规范化有五种范式，但大多数系统的规范化只进行到第三范式。

采用自上而下或自内而外的设计方法，可以将一个大的表格规范为几个较小的表格。如果用户是通过存储过程来访问数据库的，则用户可以使用视图，而不必使用应用程序。

达到第三范式的规范化规则如下:

1. 表格应具有一个主关键字，以唯一地标识表格中的每一行。即要达到第一范式，需要给每一个表格创设一个唯一的主关键字。

2. 达到第二范式，需将已满足第一范式的数据库分解，将其中多余的数据属性移入另一个单独的表格。

3．要达到第三范式，要清除表格中不依赖主关键字的列。第三范式的表格应仅仅具有一种类型的具体的数据。

规范化即用多个较小的表格来代替一个大的、包罗万象的表格。过大的表格不利于数据库的运行。注意，允许空值存在的列要尽可能少。不是十分必要，在列中不要采用空值。

过分的规范化会导致大量表格间过分复杂的连接;而规范化程度不足，又会使数据库只包含少数几个包含过多列的大表格。上述两种情况都会降低数据库的运行速度，因此要尽可能寻找两者之间的平衡点。

18.3.3 归纳

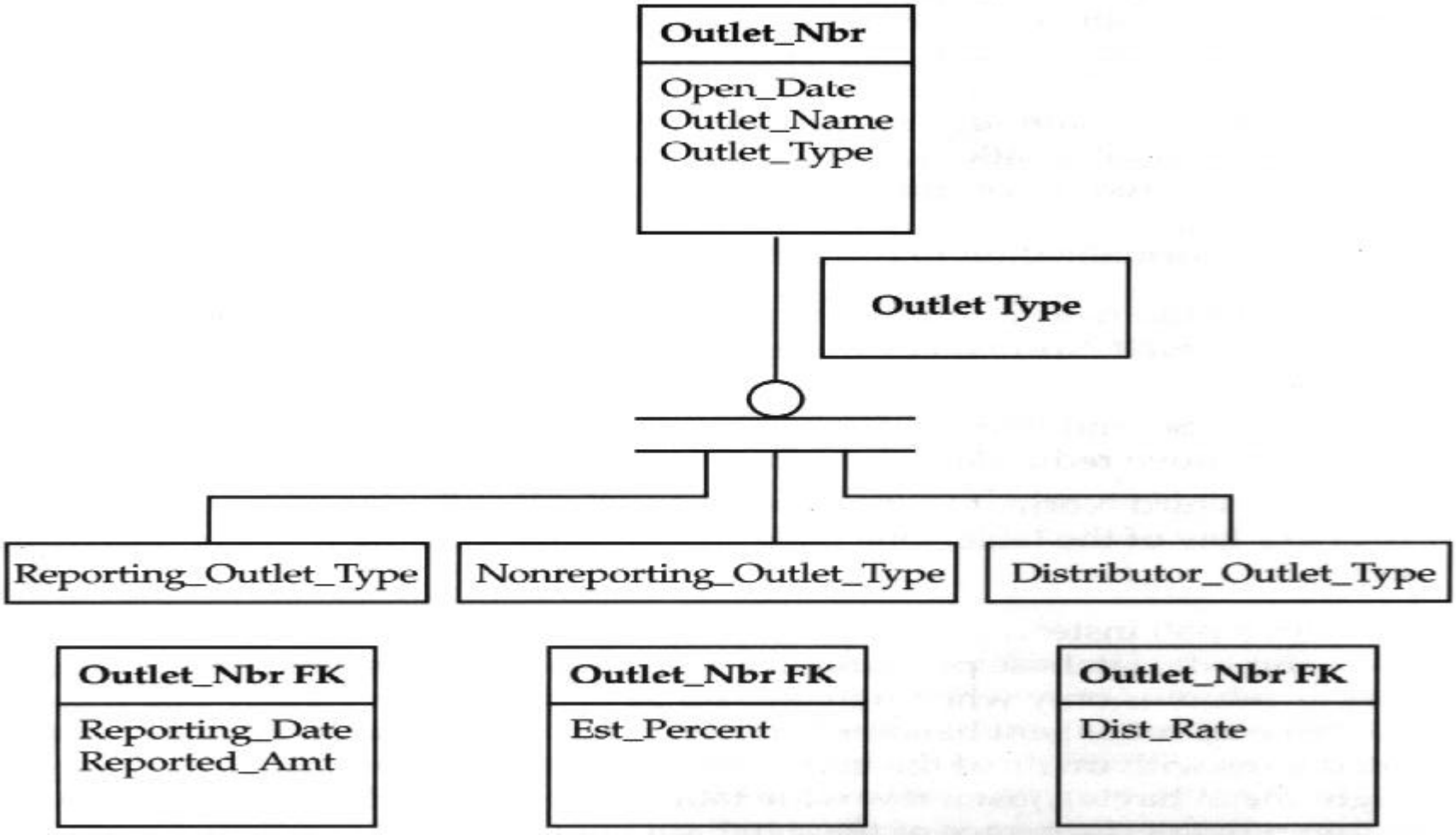

图 18.1 归纳：“类型 OR 类型”

人们总是会自然地将事物分类。在进行数据库设计时更是如此。一般认为，事物是可以划分为各种类型的，这个分类的过程被称为“归纳”。在设计数据库时，应该具有这种归纳的意识，并运用这种意识去设计一个包含有主关键字、某种类型的列和由所有类型的列所共同具有的属性组成的表格。图 18.1 给出了几个独立的子表格，这几个子表格具有相同的主关键字(这个主关键字在父表中，是一个外部关键字)和不同的属性。

如果将采用“这一类型或那一类型”的归纳思路，则图 18.1 所示的设计方式无疑是正确的。然而如果采用“这一类型和那一类型”的归纳思路，则设计的结果会有所不同。如图 18.2 所示，这时，各种类型在父表中位于同一列，而不是每种类型各占一列。

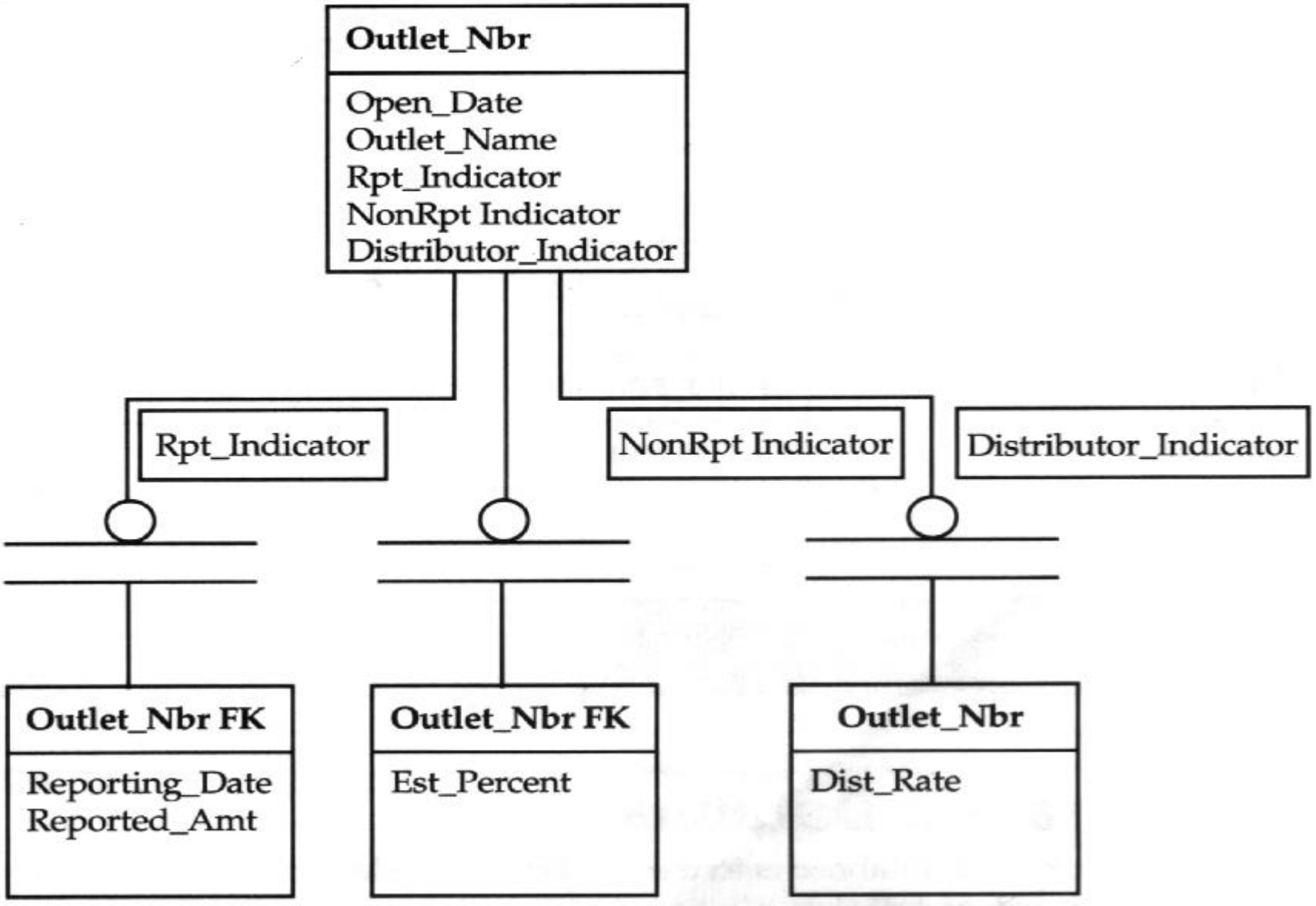

图 18.2 归纳：“类型 AND 类型”

然而，归纳还有第三种情况，即：“这一类型或那一类型”和“这一类型和那一类型”同时存在。如图 18.3 所示，在这种综合性的情况下，outlet 可能是一个 reporting oulet 或 nonreporting outlet 和一个 distributor outlet。

接下来我们研究如何设计数据库，让我们来看一下，Enterprise Manager 将给数据库设计带来多大的方便。

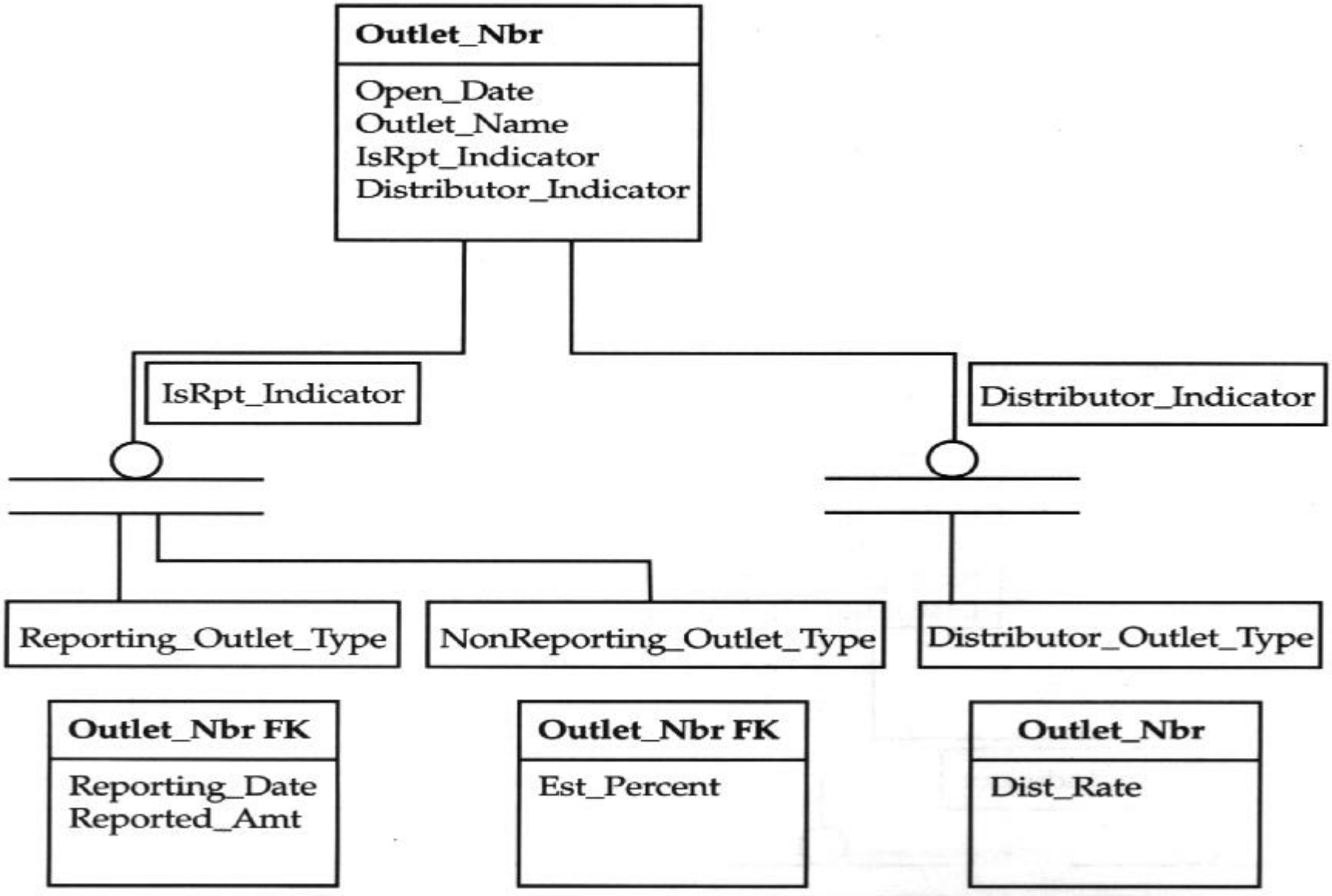

图 18.3 归纳：“类型 OR 类型”和“类型 AND 类型”

18.4 如何创建数据库

创建数据库最为简单的方法是使用 Enterprise Manager，运行 Create Database Wizard。方法如下：

1. 从 Windows start 菜单中的 SQL Server 程序组中选择 SQL Server Enterprise Manager。

2. 展开 Server Group，然后展开 Enterprise Manager 层次树中的 SQL Server。

3. 从 Microsoft Management Console 中的 Help Menu 中，选择 Wizards，就会看到 wizards 清单(如图 18.4 所示)。可以从该清单中进行选择。

4. 双击 Create Database Wizard 选项，并回答向导所提出的问题，按照缺省值或输入用户个人的参数。

也可以使用 CREATE DATABASE Transact-SQL 语句。本书第 21 章将详细介绍 CREATE DATABASETransact-SQL 语句。

图 18.4 选择向导对话框

18.5 特大型数据库(VLDB)

SQL Server 的许多方面在处理特大型数据库时会发生一些变化，以适应特大型数据库的特殊需要。

这种变化无疑使得 SQL Server 在备份、在增量备份具有更大优势，并且使得 SQL Server 能够只对一个文件或文件组进行修改，而不必变动整个数据库。SQL Server 的这种优势和能力对特大型数据的管理是非常有利的。如果用户要将一个特大型数据库备份到一张磁盘上，就可以从重新起动备份文件或修复两者之中进行选择。

Database Consistency Checker 已经完成了一项重要功能的提高工作。对特大型数据库的管理正是这一改变的主要目的。SQL Server 7 的运用对大型数据库的管理有益无害。

18.6 编 码 技 巧

在编码过程中，有一条技巧是应当掌握的，即在使用 SQL Server 时，应充分发挥存储过程的作用，而不应在应用码中使用 Transact-SQL。

使用存储过程的好处很多，其中包括:它们为模块化程序设计提供关键字。因而我们可以建立存储过程，将其存储在数据库中的 syscomments 系统表中，

在使用时调用它。对存储过程进行修改时，不必重新编辑应用码，也不必在成千条编码中搜寻那些因数据库设计的变化而需改动的 Transact-SQL 编码。

Transact-SQL 码已经过了分析和优化，因而运行十分快捷。如果有人在最近时间内调用存储程序，则查询规划会被保存在内存中。另一方面，那些来自于被嵌入应用码的客户机的 Transact-SQL，每次通过网络传送到 SQL Server 中使用时，都要经过编译和优化。存储过程减少了网络信息量。除此之外，存储过程可以作为安全性设计的一部分。因为客户可以不必使用特殊查询 Transact-SQL 语句即可运行存储过程。

18.7 索引

索引的建立加速了对数据库的数据查询。索引可以只含有一列，也可以是多列的组合体。SQL Server 索引分两大类：

- 聚簇索引
- 非聚簇索引

聚簇索引是指：按照索引中所包含的列排序，并将经过排序的数据安排好的顺序存入表格之中。因为行是经过排序的，包含这样的行的实体只能有一个。所以，每个表格只能有一个聚簇索引。数据行是索引的一部分。除了那些很小的表格外，每个表格都应该有一个聚簇索引，而且最好是以主关键字建立的聚

簇索引。

而在非聚簇索引中，数据行不是索引的一部分。非聚簇索引中，表格中的每一行数据都放置了指针。非聚簇索引的效率不及聚簇索引，但是它非常适合作表格浏览，以及在表格中查询满足某些条件的行。

索引可以按以下两种方式定义：

- 唯一索引
- 非唯一索引

唯一索引是指索引关键字段的各行无相同数值。而非唯一索引相反，它允许索引关键字段各行有相同的值。

下列语句可以用来创建索引：

- CREATE INDEX 语句
- CREATE TABLE 语句，PRIMARY KEY 子句
- CREATE TABLE 语句，UNIQUE 子句
- CREATE TABLE 语句，PRIMARY KEY 子句
- CREATE TABLE 语句，UNIQUE 子句

如果数据已经被排序，可以使用 CREATE INDEX 中的 SORTED_DATA 或 SORTED_DATA_REORG 选择。

这种选择可以提高建立聚簇索引的速度。填充度决定着在非聚簇索引的索引页上，或聚类索引的数据页上，每一个索引项目之间的间距。如果数据库需要经常增删数据，则建立索引时就使用低填充度;如果数据库是只读性的，则建立索引时就使入高填充度。填充度并非永久不变的。索引会长期地保持一定的

密度，并在索引重新建立之前保持不变。

以下是关于索引的几点注意事项：

· 较小的索引比含有多个列的较大索引运行速度快。而整数或 IDENTITY 关键字是最快的。

· 如果一个索引的第一列未在 WHERE 子句中被引用，则 SQL Server 将不能使用这个多列的复合性索引。

· 作为索引关键字的列中最好无重复值。否则运行结果只能是表格浏览。

· 研究用户查询记录的方式，选择效率较高的聚簇索引的最佳位置和用途。尽可能不使用非聚簇索引。

索引调整是增强数据库功能的主要途径。Microsoft SQL Server 在 Query Analyzer 中提供了 Index Tuning Wizard。它是一个与 Enterprise Manager 中的 Tools 菜单选择下的 isql／w 相似的窗口。Index Tuning Wizard 与 SQL Server Profiler 结合使用，可以用来分析用户运用 SQL Server Profiler 工具捕捉并运用的工作负荷。Index Tuning Wizard 可以针对不同数据库向用户推荐最佳的索引排列方法。在第 7 章中，我们对 Index Tuning Wizard 已作过详细说明。

18.8 分区

大型表格存在着运行不便的问题。将一个数据库划分为几个较小的表格，则能够缩短查询时间。而且，较小的表格索引的重建也更快捷。

如果表格不易被分解，则可以将不同的表格分别置于不同的磁盘轨道上。这样做的目的在于，同时运用多个驱动器同时读取资料。SQL Server 中有一个新的文件组功能。使用这个功能可以将多个表格进行物理划分，分置于不同驱动器之中。之后，通过多个驱动器，运用磁盘阵列将表格分解。

18.8.1 水平分区

水平分区指将一个大表格分解成几个具有相同结构的小表格。水平分区可以依据月份、字母顺序或其他任何数据的自然因素。划分表格后，数据查询所需进入的表格越少越好。这样可以避免分解表格的过分联合，使分解后的表格不致回复成为一个大表。

18.8.2 垂直分区

垂直分区是通过规范化和行的分割实现的。行的分割是将一个大表格划分成几个含较少列的小表格。分解后的两个表格的行相连接，即得到与分解并的大表格相同的行。

垂直分区同样可以减少查询所需访问的数据。如果一个大表格中只有少数几列是常用的，就可以将那些不常用的列置于另一个表格之中。划分表格可以使大表格的运行变得快捷。

18.9 并发性

并发性是指，在多用户同时访问数据时，通过锁定保持事务完整性的功能。为大量并发用户开发的系统，最好能在一段时间内保持事务。排他型锁可以起到这一作用，它可以阻止其他事务读取表中的数据，直到当前的事务被确认或滚回时为止。

排他型在事务被确认或被滚回之前一直起作用。事务阻隔标准装置可以确定 SELECT 语句能否获取排他型锁。有些情况下，锁定是不必要的。例如:开发人员需要输入数据，或在某一事务处理中需浏览更多的数据，或者试图在某一事务的处理中进入更多的非必要进入的数据。这些情况下，就应选用可读的较低阻隔标准。保持事务的完整性和允许更多开发用户进入数据行，两者都是很重要的。阻隔标准就是为了协调两者的矛盾而设置的。标准越高，则允许存在的阻隔越多，限制锁定越多。阻隔标准越低，则相反。

阻隔标准由低到高排列如下所示：

- READ UNCOMMITTED
- READ COMMITTED
- REPEATABLE READ
- SERIALIZABLE

在第 21 章中，有对 SET TRANSACTION ISOLATION LEVEL Transact-SQL 语句的详细介绍和用法讲解。如果一个数据在使用的同时难以变更，则应指出在

使用游标时，于何处设置游标阻隔标准更有意义。

在隐含事务模式下处理某一事务时，COMMIT 或 ROLLBACK 后，将产生新的事务，而用户可能觉察不到这一新事务的产生。很多情况下，用户都意识不到自己正在要求正在处理某一事务的其他用户输入数据。为避免隐式事务模式可能带来的不便，可以在 COMMIT 或 ROLLBACK 之后，在编码中使用 Transact-SQL 语句，将隐式事务模式设置到 OFF 状态。下列任何一个 Transact-SQL 语句都可以起到这一作用:ALTER TABLE、FETCH、REVOKE、CREATE、GRANT、SELECT、DELETE、INSERT、TRUNCATE、TABLE、DROP、OPEN、UPDATE 等。而 COMMIT 或 ROLLBACK 可以关闭事务。

本章中，讨论了有关开发的话题。下一章中，将讨论如何书写 Transact-SQL 查询、存储过程以及触发器。

第 19 章 Transact-SQL 的使用

本章将讨论如何使用 Transact-SQL。它即是一条非常简单的也是非常有用的查询语句。并且 Transact-SQL 的查询书写也很简单。使用 SQL Server Query Analyzer 即可以键入简单的查询命令并能得到结果集。从 Microsoft SQL Server 程序组或 Enterprise Manager 中的 Tools 菜单中可以进入 Quary Analyzer。要了解更多的有关 SQL Server 7 的安装问题，请参照本书第 11 章。

在对 SQL Server 有了初步了解之后，即可着手编写较复杂的查询语句，接下来的几节中，将讨论与 Transact-SQL 相关的如下主题：

- 编写查询语句
- 编写存储过程
- 编写触发器
- 快捷、大量的输入／输出
- 查询处理器
- 多索引查询
- SHOWPLAN
- 表格连接
- Tempdb

- 游标
- 重新编译存储过程
- 异构型分布式查询

SQL Server 所安装的目录下提供了可供使用的范本程序。下文将首先讨论第一个主题——编写查询语句。

19.1 编写查询语句

Transact-SQL SELECT 语句连带一个 WHERE 子句，就构成一个最简单、最有效的查询语句。下面是一个使用 SELECT 语句的简单例子：

```
SELECT*
FROM MyAccountTable
WHERE Past_Due_Amt > 100000
```

在本例中，*表示“取 MyAccountTable 中所有的列和列中的数值”。

通过这个简单的查询，可以得到许多相关的重要信息。而从所得到的信息中，又可以得到更为重要的、进一步的信息。如本例查询所得的结果集中，有一个 Customer ID 列。我们可以得到该列的值，也可以运用以下这个类似的查询语句，查找到 CustomerName(客户姓名)和电话号码：

```
SELECT*
FROM CustomerTable
```

```
WHERE CustomerID = 3876098
```

本例中的这个命令很可能帮助用户直接找到客户姓名和电话号码。即使不能，它也会给用户提供一些信息，提示客户电话号码所在的表格。根据提示，可以使用 SELECT 语句继续查询。

使用下述语句，可以看到数据库中所有用户定义表格的名称，并查询其内容。

```
SELECT name
FROM sysobjects
WHERE type = 'U'
ORDER BY name
```

在本例中，在 sysobjects 系统表中，列 type 中是 U 的对象是一个用户定义的表格。那些具有相同名称的列，且该列的值相同的表格是相互连接的，这种连接叫做关系。

查询是 Transact-SQL 中最有趣味的部分。它的形式可以是最为简单的，但却非常有效，而且，在实际使用中，查询也可以变得相当复杂。因此，无论对于初学者还是专家，它都同样具有挑战性和趣味性。

下一节中，将讨论如何编写存储过程。

19.2 编写存储过程

像查询一样，最简单的存储过程非常容易书写，但也是非常有效的。存储过程实质就是经过编辑，并被存入数据库之中的 Transact-SQL 语句。它的运行速度非常快，因为在调用存储程序时，语句编辑和分析过程并不出现。

存储过程的另一个优势是:它们集中存储在一个中心部分，而不是分散在整个应用程序的编码之中。当用户的事务规则变更时，只需在一个地方寻找 Transact-SQL 编码并修改它即可。存储过程是一种数据库对象。

下面是一个简单的存储过程的例子:

```
CREATE PROCEDURE MajorPastDue
AS
SELECT*
FROM MyAccountTable
WHERE Past_Due_Amt > 100000
GO
```

在例子中，在上节讲过的 SELECT 语句中增加了一个 CREATE PROCDVRE AS 语句。CREATE PROCDVRE 语句可以将一系列 Transact-SQL 语句作为一个数据库对象来创建，并将它存储在数据库之中。

下一个例子将展示如何调用上例所创建的存储过程:

```
EXECUTE MajorPastDue
```

运用本例中提供的语句，可以调用上例所创建的存储过程，得到 SELECT 语句运行所得的结果集。

下面，我们研究略为复杂的存储过程。用户可以使用局部变量和参数将信息传递到存储过程中，也可以从一个存储过程中调用其他存储过程。使用 OUTPUT

选择，可以将变量从一个存储过程传送到另一个正在被调用的存储过程中。下面是一个较为复杂的存储过程的例子：

```
CREATE PROCEDURE ProcWithParm
@customer_id int
AS
SELECT Customer_Name
FROM Customer
WHERE Customer_ID = @customer_id
```

在本例中，客户的姓名被反馈回来，客户 ID 被传送到存储过程之中。

下面一个例子将展示如何调用一个带有参数的存储过程。

```
EXECUTE ProcWithParm 10034
```

在本例中,上例所创建的名为 ProcwithParm 的存储过程被调用,客户 ID10034 作为一个自变量或参数被传送到存储过程之中。

下面一个例子中的存储过程更长、更复杂。它需要与游标共同运行。并且，数据库的每个表格均需运行 UPDATE STATISTICS 命令。

```
/*************************************
Description: UpdStatsAll is a stored procedure that uses a cursor to run
        Update Statistics on every table in the database.
        Execute this stored procedure in a user database.
*************************************/
```

```
IF EXISTS (SELECT * FROM sysobjects
     WHERE id = object_id ( ' dbo.UpdStatsAll ' )
     ANDtype = ' P ' )
BEGIN



END
GO
     PRINT ' Create Procedure UpdStatsAll'
GO
     CREATE PROCEDURE UpdStatsAll
AS
     DECLARE @table_name varchar(128),
                    @table_name_msg varchar(95),
                    @errmsg varchar(85)
     SELECT ' Starting Update Statistics ' ,getdate()
     DECLARE cursor1 CURSOR
     FOR SELECT name
     FROM sysobjects
     WHERE type = ' U '
     ORDER BY name
OPEN cursor1
```

```
FETCH NEXT FROM cursor1
INTO @table_name
WHILE(@@FETCH_status<> -1)
BEGIN
     IF (@@FETCH_status<> -2)
     BEGIN
          SELECT @table_name_msg =  ' System is running Update Statistics on  ' +
 RTRIM(@table_name)
     PRINT @ table_name_msg
     EXEC( ' UPDATE STATISTICS ' + @table_name)
END
ELSE
     BEGIN
     SELECT @errmsg =  ' FETCH error has occurred. '
     GOTO err_rtn
END
     FETCH NEXT FROM cursor1
     INTO @table_name
END
SELECT  ' Finished with Update Statistics  ' ,  getdate()
DEALLOCATE cursor1
RETURN
```

```
err_rtn:
    DEALLOCATE cursor1
    RAISERROR 500000 @errmsg
    RETURN - 100
GO
```

19.3 编写触发器

触发器实质上是一种特殊的存储过程。当表中运行 INSERT、UPDATE、DELETE 命令时，触发器会自动被激活，发挥一些特殊作用。下述例子将展示当表中运行 INSERT、UPDATE 或 DELETE 命令时，如何发送一封电子邮件。

```
IF EXISTS (SELECT name FROM sysobjects
WHERE name = ' CustChgTr '  AND type = ' TR ' )
DROP TRIGGER CustChgTr
GO
CREATE TRIGGER CustChgTr
ON Customer
FOR INSERT, UPDATE, DELETE
```

```
AS
EXEC master..xp_sendmail ' CUSTSERV ' ,
  ' The data in the customer table has changed.'
GO
```

本例中，当表格 Customer 中的数据变更时，电子邮件被发送到别名为 CUSTSERV 的表格 Customer Service 中去。

触发器的另一个重要作用是建立起了虚拟表格的概念。虚拟表格是一个带有触发器的表格的拷贝。当该表格增删数据时，表格的触发器可以按编定的程序被激活。下例展示表格被插入数据时，触发器是如何被激活的：

```
IF EXISTS (SELECT name FROM sysobjects
WHERE name = ' reminder'  AND type = ' TR ' )
DROP TRIGGER reminder
GO
CREATE TRIGGER MyTrigger
ON MyTable
FOR INSERT, UPDATE
AS
DECLARE @MyFlag tinyint
SELECT @MyFlag = d.Disc_Type
FROM Artist a INNER JOIN inserted i ON a.art_id = i.art_ID
JOIN Discipline d ON d.disc_id = i.disc_id
IF (@MyFlag = 0 )
```

```
BEGIN
RAISERROR ('This artist does not have a discipline.',16,-1)
ROLLBACK TRANSACTION
END
```

在本例中，插入数据的表格被查看后，触发器被激活。本例同样适用于运行 UPDATE 和 DELETE 命令的表格。

当一个触发器中的同一个表格第二次更改时，触发器不能激活它自己，除非用户设置了 Resursive Triggers(重复触发器)选择。使用 sp_dboption 语句可以设置这种数据库选择。重复触发器功能是 SQL Server 7 新增设的功能。

19.4 快捷、大型的输入／输出

SQL Server 7 版本增设了快捷、大型输入／输出功能，这是该版本在技术上的主要进步。它将 SQL Server 带入 21 世纪。Microsoft 在设计中强调可扩展性和运行性能——除了大力增加 I／O 水平外，还有什么更能实现这一目标呢？此版本中，输入／输出的页数大大增加，并实现了并行 I／O。

19.4.1 大型 I／O

8KB 的 I／O 量，在页数上是 2KB 的 4 倍，所占盘区也是 2KB 的 4 倍;同样，64KB 的页数和所占盘区也是 16KB 的 4 倍。

19.4.2 快捷的 I／O

快捷的 I／O 被概念化，使得查询处理器成为数据 pump 的功能得以实现。运用快捷的 I／O，我们可以处理大型 I／O，和大量的超前读内容(为实现查询目的，超前读的页数已被存放在高速缓冲存储器中)，也可以进行物理行序的数据浏览，并从文件上读取并行 I／O。其中超前阅读既适用于聚簇索引，也适用于非聚簇索引。超前阅读由查询处理器驱动，对随后所需的页的内容加以提示。

19.5 多索引查询

在 SQL Server 7 版本中，通过索引交叉进行查询。每个表格可以使用多个索引，这些被同时使用的索引中写有共同的行指示器，因而 SQL Server 可以将这些索引进行连接，在短时间内反馈所需的结果集，这样就增强了查询功能。

19.6 查询处理器

SQL Server 7 版本中，查询处理器的运行与以前版本有所不同。在此版本中，它用以支持特大型数据库和复杂的查询。本版本中，增加了新的运行技巧，如:散连，散合计连接以及合并连接技巧。这些技巧与传统的嵌套循环连接相比，使用更方便、效果更好，并能适应较大型数据库的要求。

多索引的索引交叉和联合能够在检索之前对数据进行过滤。表格的所有索引可以被同时更新，各种约束也将在查询程序设计中被加以考虑。

成本模式的改进和编译时间的增强使得查询规则有了大幅度的发展。下一节中，将讲解 SHOWPLAN 功能，看一看查询处理器是如何运行的。

19.7SHOWPLAN

SQL Server 7 将 SHOWPLAN 置于 Query Analyzer 工具中一个显著的位置，以突出它的功能。在 Query Analyzer 窗口中键入一个查询命令，选择 SQL Server Query Analyzer 中的 Execution Plan 标签，调出 SHOWPLAN，就能看到查询处理器是如何进行数据查询的。

在 SQL Server 7 中，一条查询命令可以在 32 个表格中被执行。内部工作

表格也不再局限在 16 个。如果希望信息以文本形式而非图表形式反馈，也可以使用 Transact-SQL 语句来运行 SHOWPLAN 来实现。但是，如果设置了 SHOWPLAN_TEXT 或 SHOWPLAN_ALL 状态，则不能再进行查询，相反地，用户将看到查询执行规化是如何被查询优化器格式化的。

下面的 SHOWPLAN 命令范例展示了如何运行查询命令，以得到文本形式的数据：

```
SET SHOWPLAN_TEXT ON
```

本例中，Transact-SQL 语句被执行后，即可看到数据以可读文本形式输出。下面是一个类似的例子：

```
SET SHOWPLAN_ALL ON
```

19.7.1SHOWPLAN 树

SHOWPLAN_TEXT 和 SHOWPLAN_ALL 都是展示查询执行规化的 Transact-SQL 语句。当 SHOWPLAN_TEXT 或 SHOWPLAN_ALL 语句被设置在 ON 状态时，查询不能被执行。此时查询优化器反馈回来的，是详尽的查询执行规化。

如果输入的是一个存储过程或 Transact-SQL 语句，它会将自己作为树根建

立，而被该存储过程调用的语句则作为树枝。

如果输入的是一个 Data Manipulation Language(DML)语句，例如 SELECT、INSERT、DELETE 或 UPDATE，则这些语句将作为树根，并可以带有两个分枝:执行规划和触发器。

SHOWPLAN 将条件语句，如 IF…ELSE 划分为三个分枝。IF…ELSE 语句作为树根，if 条件句作为它的子树分枝，then 和 else 条件句点是作为一个程序块被使用;WHILE 和 DO-UNTIL 语句有相似的三枝规化。

查询引擎，如:表格的浏览、连接、集合等执行的操作作为树的枝，每个分枝都包含一定的信息，例如:相关的代数运算符和这些代表运算符的算法(如无效数据连接、合并连接、嵌入循环等)。这些运算所占的字节数可以被显示出来，优化器将选择占用字节最少的一种方案。

OPEN CURSOR 语句作为树根，并带有其他语句作为树枝。

下一节中，将讨论有关表格连接的问题。

19.8 表 格 连 接

表格连接的目的是对两个或两个以上表格中相关联的数据进行检索。它是通过比较两个或两个以上表格中的数据，并用表格中相匹配的行组成一个新的表格来完成的。那些符合连接条件的相匹配的行，是通过在每个表格的两列之间使用逻辑运算符，如<，=或>来进行选择的。

在表格连接时需要使用 WHERE 子句。WHERE 子句包含着一个连接符号，例如等号。并且，在 WHERE 子句中通常要给出两个被连接的表格的名称。在使用等号的情况下，则反馈回来的列每一行的值都相等。如果连接中未使用 WHERE 子句，则会产生笛卡尔乘积。也就是说，参加连接的所有表格的所有行的任何一种可能的组合方式都出现。此种情况下，反馈回来的行数多于用户查询所需的结果。也可以在 WHERE 子句中使用关系运算符来连接表格。使用 ANSI 样式语句连接表格时，有一种新的功能，使用户可以使用嵌入的外连接，或嵌于一个外连接中的内连接。

连接的类型包括：

内连接　　是一种常用的，使用比较运算符(例如,=)的连接方式。其结果是，在两个被连接的表格中，只有那些符合比较条件的行被显示出来。

外连接　　包括左外连接、右外连接和全(外)连接，稍后将分别对它们作详细说明。

交叉连接　　交叉连接产生两个表格的各组的交叉乘积。反馈回来的结果与无 WHERE 子句的传统非 ANSI 样式相同。

外连接的方式包括：

左外连接　　使用左外连接时，JOIN 子句中第一个表格的所有行均被反馈回来，而第二个表格中那些没有与第一个表格相匹配的值的行，则被赋 NULLS。

右外连接　　使用右外连接时，JOIN 子句中第二个表格的所有行均被反馈回来，而第一个表格中那些没有与第二个表格相匹配的值的行，则被赋 NULLS。

全(外)连接　　使用全(外)连接时，第一个或第二个表格中那些不符合选择条件的行均被选出来，对应表格的列中被赋 NULL 值。

内连接的关系运算符包括：

=	等于
>	大于
>=	大于或等于
<	小于
<=	小于或等于
< >	不等于(ANSI 标准)
! >	不大于
! <	不小于
!=	不等于
LIKE	匹配
NOT	不
OR	或者

可以使用 substring 函数，通过比较第一个字符、列的长度来连接文本和图像列。这也是连接此数据类型的列的唯一方法。查询处理器的新的连接技术将在下文中介绍:合并连接和散列连接。

19.8.1 合并连接

合并连接是一种新的连接方式。当查询优化器收到一个对有序排列的数据进行连接的命令时，查询优化器将选择合并连接方式。如果两个表格中的数据都是按连接关键字排序的，或要求按照聚簇关键字排序，在这两种情况下，等值连接是最为高效的。

合并连接的运行方式如下。从外表中取一行，从具有相同关键字的内表中寻找匹配行。如找到，则将该行反馈并在内表中循环，如未找到，则在外表中循环。

散列连接是 7 版本中查询优化器中另一个新的内容。

19.8.2 散列连接

当表格无索引，或无需按序输出数据，且连接为等值连接时，可以采用一种新的、无需按序输入数据的连接方式——散列连接。查询命令被格式化，并且表格索引尚未建立时，它被用以进行特定查询。

散列连接连接方法如下;读取较小表格中的数据，散列关键字值，将关键字和行 ID 都置于散列存储桶中，之后循环，直至较小表格的末行。接下来，读取较大表格的数据，hash 关键字，看在 hash 存储桶中是否有与之相同的值。如果有，将关键字及行 ID 反馈回来，之后循环，直至外表末行。

查询优化器所具有的最后一种连接类型是嵌套循环连接。

19.8.3 嵌套循环连接

嵌套循环连接是在以往的 SQL Server 版本中已存在的传统的连接方式。当一个表格的输入数据少，而另一个较大时，这种方式常被采用。它是一种不等值连接。在合并和散列不能适用的情况下，亦可采用此方式。

嵌套循环连接的操作过程如下:从外表中与内表格各取一行，将其进行比较。如其相匹配，则将结果反馈回来;并在内表中依次循环这一过程。内表循环结束后，返回第一行，继续在外表中循环。

19.9Tempdb

在 7 版本中，如果需要占用更大的内存空间，则 tempdb 系统数据库会自动扩展。而在下一次 SQL Server 启动时，SQL Server 又将 tempdb 系统数据库恢复成原来的大小。tempdb 是数据库工作区。当运行特定查询，或年末、月末操作过程中 tempdb 需求量大而所需持续时间较短时，这种新的功能的使用将带来很大的便利。

19.10 游标

确切地说，游标应被称为卷动器，因为它实际在做数据卷动的工作。游标在数据库中卷动数据，使上行、下行、相关或绝对行成为当前行。游标的整个语法框架是类似的，有几种不同的类型。使用游标的最简单的程序如下例所示。它的功能是比较容易理解的:卷动那些执行 SELECT 语句所反馈回来的数据。

```
DECLARE cursor1 CURSOR
FOR SELECT name
FROM sysobjects
WHERE type = 'U'
ORDER BY name
OPEN cursor1
FETCH NEXT FROM cursor1
INTO @table_name
WHILE (@@FETCH_status <> -1)
BEGIN
IF (@@FETCH_status <> -2)
BEGIN
SELECT @table_name_msg = " System is running Update Stat

istics
```

```
ON " +
 RTRIM(@table_name)
PRINT @table_name_msg
EXEC ("UPDATE STATISTICS"+ @table_name)
END
ELSE
BEGIN
SELECT @errmsg = ' FETCH error has occurred.'
GOTO err_rtn
END
FETCH NEXT FROM cursor1
INTO @table_name
END
SELECT "Finished with Update Statistics"   ,getdate()
DEALLOCATE cursor1
RETURN
err_rtn:
    DEALLOCATE cursor1
    RAISERROR 500000 @errmsg
    RETURN -100
    GO
```

在本例中，执行 SELECT FROM sysobjects 命令后，反馈回一系列数据。通过在一个循环中依次移动游标，数据被卷动。通过卷动数据，使其逐行显示，就可以对数据进行某些处理。在本例中，是获取用户定义表格的名称，之后对该表运行 UPDATE STATISTICS 命令。SQL Server 7 版本中，有一种新的方法，可以对每个表格通过使用 Database Maintance Plan 在循环的基础上运行 UPDATE STATISTICS 命令。

在 7 版本中，有关游标的某些细节有所变化。本版本中，游标可以是全局的，也可以是局部的。局部游标只在一定范围内有效，在范围外将被取消。Transact-SQL 中还有一个新的游标，它是由存储过程创建的，并可以在整个运行过程中传递。游标还有许多类型，详细情况请参看本书第 21 章。

19.11 重新编译存储过程

有时，需要对数据库进行某些处理，而存储过程在设计中可能未将这些处理考虑在内。例如:需要追加索引或被编入索引的列中的数据需要重新分布，这时，存储过程就需要重新编辑，以适应新的需求。但是，并不是总是需要进行存储过程的重编译。SQL Server 被重新启动或存储过程再次被调用之后，或者当存储过程中所使用的表格被更改时，存储过程会自动进行重新编译。

19.12 异构型分布式查询

SQL Server 7 支持异构型分布式查询。本版本中，Transact-SQL 查询使用来自于 OLE DB 数据源中的数据，而不使用 Open Data Services 来书写服务器应用程序，以之作为与其他数据库系统的连接点。

本章中，介绍了 SQL Server7 的 Transact-SQL 使用的主要部分。有关 Transact-SQL 语言的语法结构，请参看本书第 21 章。

第 20 章数据仓库和数据市场

数据仓库概念的出现是为了解决在线分析处理(OLAP)时发生的问题，在线分析处理包括与在线事务处理(OLTP)发生干涉的决策支持活动。

在同时运行某个 OLAP 查询时，增加和改变数据行的大量并行用户之间可能会引起争议，例如经历无法接受 OLAP 性能的用户和报告说 OLAP 查询运行时间太长的 OLAP 用户。按规则编排的基础提供数据库拷贝可以有所帮助，但是随着数据库的不断膨胀，往往有必要重新设计数据库，有必要为加载和查询大量数据的数据仓库进行特别设计。为此，可为 OLTP 或可操作类型的活动特意设计另一个数据库或一组数据库，并且另一个数据库或一组数据库可以是数据仓库的一个子集，该数据仓库用于报告部门层次的活动。当包含需要摘要的大量数据时，数据仓库显得特别有用。

OLTP 系统和 OLAP 系统之间的区别在于它们存储数据的方法不同。在 OLTP 系统中，数据是以高标准方式按详细层次存储的。而在 OLAP 系统中，数据仓库中的数据是以非标准摘要方式存储的，为的是提高 OLTP 决策支持处理的查询性能。

数据仓库可提供摘要性的历史只读数据，以便简化决策支持活动，通常可强化来自不同可操作数据源的数据。在对数据进行一致性分析时，数据仓库还

可提供跨组织的标准化，当把数据仓库放到一个位置时，对它的组织只包含关键的性能指示器。对于运行与执行报告无关的事务，在低层次上需要的外部信息被留下。

对信息进行概述，长时期限制数据仓库，目的都是为了提供历史分析和趋势报告。在数据仓库中，数据一般不作改变，除非因为发生错误而必须进行重新处理。发生的唯一操作是加载和报告数据。数据仓库使用星型模式(star schemas)来改进响应时间。

20.1 星 型 模 式

星型模式对于主题领域使用一种“事实”表和描述该事实表的许多尺寸表。为提高性能，可对存储在事实表中的信息进行概述。空间上的关键字是可用作外来关键字的唯一标识符，外来关键字可把事实和尺寸表联结起来。当数据仓库数据库在使用星型模式时，它可以包含长整型窄行事实表，以及小型宽行尺寸表。查询可检索小型尺寸表中的尺寸关键字，以嵌进主事实表，减少磁盘扫描量。由于数据在加载时就已经进行概括，所以查询速度加快。

20.1.1 事 实 表

事实表是与事件相关的实体，其中包含每个时期和每个地理上的或历史上

的分组的数据集合。销售、事务、灾难、计算机崩溃、需要服务、书籍出版、房屋结构和历史事件等都有资格作为事件，都可根据业务变换无穷。OLAP 以其集合的形式需要历史的和高层次的观察，以检测趋势和分析商业实践的成功，并支持由主管人员做出的商业决策。

事实表可以包含成千上万行历史只读数据。事实表应该只包含数字型数据，而非带有整数外来关键字的字符型数据，这些数字型数据再转换成尺寸表，以履行它们作为星型模式中心部分的作用。如果中心事实表变得太大，有可能会变成性能瓶颈，应该按自然分割的形式通过将它分割成多个逻辑表进行分段，如按日期分段。

20.1.2 尺寸表

尺寸表是一种非常小的表(与数百万或数十亿行相比只有数百或数千行)，并引用存储在事实表中的数据，事实表中含有描述型、名称型和其他字符型数据。把小型尺寸表中的字符型数据放入巨型数字型事实表的一侧，可允许查询嵌进小型尺寸表，对大型事实表拾取索引关键字和扫描更小型的数据集。首先对事实表进行处理结果的清楚描述或命名结果，使在 tempdb 系统表中带有切实可行需求的巨型数据区域的查询扫描速度大幅度降低。当包含超大量的数据时，带有围绕概括事实表的决策表的星型模式的决策，在管理层次上一直保持磁盘扫描和 tempdb 用法。

而事实表中的数据在出现错误之前不会发生变化，尺寸表中的数据设计成

可说明改变的数值。决策表有一个整数主关键字，也称为尺寸关键字，由占优势的字符型数据组成。所有标准化的规则在设计尺寸表时均被中断，目标是把数据拆散成单一表。使用的技巧是要知道如何访问数据，并用尽可能最为便利和最为快捷的方法设计访问方法，以消除与数据间接参考的表联结。多头联结是这种尺寸的数据库的大敌，必须通过拆散尺寸表进行设计。换句话说，如果普通使用的查询需要联结四个表，可为该查询设计一个特殊的尺寸表。如果区段存储在另一个表中(例如销售厂家表)，则可删除该表，把它放到自己的尺寸表中，并把 districk_id 的尺寸关键字添加到事实表中。用日期或周期数据也可以做到这一点。财政年、周期、星期、月和年都是日期型尺寸标的实例。这种方法可大幅度提高性能。

通过确定核心商业事件可仔细慎重地逼近事实表和尺寸表，在核心商业事件中，数据仓库将集中创建事实表。然后再确定分析事件的方法，创建尺寸表。

20.2 数 据 市 场

数据市场是数据仓库的一个较小分支，它位于自身的数据库中，并为商业的特殊焦点领域所使用。数据市场包含的数据可以处于细节层次和／或概述层次。由于数据量较小，所以数据市场查询起来即容易又快捷。

设计数据仓库及其相应的数据市场有多种不同的方法，唯一的限制性因素就是开发自己的想象和自己的商业需求。从数据市场可以创建数据仓库，从数

据仓库也可以创建数据市场，使用下拉式的、中间弹出式的和底部弹出式的方法均可。其中最重要的操作是正确定义关键字、基本原理和底层商业事件，以及对周边事件的分析。开始建立数据仓库时不必完全重做所有事情，通过识别关键商业事件开始，如销售，并从一个关键事件开始建立自己的数据仓库，提供分析和趋势报告，这些趋势报告可提供商业内部的决定性市场需要的关键信息。罗马不是一天建成的，数据仓库也不是一天可以建成的，但是从数据仓库获得的利益足以保证沿着正确方向前进的步伐。

20.3Microsoft OLAP Server

Microsoft OLAP Server 是一个允许分析大量数据的中层数据库。该产品支持在线分析处理(OLAP)应用程序，并可做出报告、数据模型和决策支持。

Microsoft Desktop Data Cube Service 可提供客户应用程序对 OLAP 数据的客户机访问。OLAP 技术设计目的是对可能需要充足时间才能回答的问题提供答案。OLAP 技术可对需要立刻回答的问题提供答案，因为这些问题要进行一致性查询和有效地设计数据仓库。OLAP 服务器把数据预处理成“立体数据”，概括成如时间和地理这样的尺寸。Microsoft OLAP Server 支持三种技术，即多维型的(MOLAP)、关系型的(ROLAP)和混合型的(HOLAP)数据库,均可用于存储 OLAP 数据。

OLAP 解决方案可用于大型商业的报告和分析需求。到现在这些解决方案一

般还是非常昂贵的。Microsoft OLAP Server 和 Desktop Data Service 都有图形用户接口和向导，以帮助从已有数据仓库快速和廉价地创建 OLAP 数据库。

20.3.1 数据爆炸

当合计所有可能的数据集合时，以及结果行大于事实表中已有的行时，就会发生数据爆炸。Microsoft OLAP Server 可以自动选择所有可能集合的子集，并在需要时计算来自子集的集合的容量。

Aggregation Design Wizard 可提供操纵磁盘存储需求和合计集合量之间平衡的功能。Microsoft OLAP Server 可以跨几个服务器分段立体数据和存储数据，通过支持完整 MOLAP 实现、完整 ROLAP 实现和 HOLAP 解决方案，Microsoft OLAP Server 具有数据模型的灵活性，为 OLAP 数据库设计者提供的数据模型选择最接近符合组织的需要。

20.3.2Desktop Data Cube Service——灵活的解决方案

在客户机上可以运行 Microsoft Desktop Data Cube Service，可以使用 Microsoft Visual Basic 或其他语言开发用户应用程序，这些用户应用程序可充分利用来自 Microsoft OLAP Server 的数据，或来自使用 Microsoft OLE DB 的关系型数据库的数据。这样可允许 Microsoft OLAP Server 具有使用相同立

体数据的多台客户机。这样可对表现形式和分析提供灵活的解决方案，这种分析允许用户在不连接到 Microsoft OLAP Server 的情况下分析数据。

Microsoft ActiveX 控件和 Microsoft Office 实现都通过提供类似的终端图形用户接口来创建。其他软件供应商也提供开放接口，以开发第三方应用程序。

20.3.3 多维数据库立体数据

立体数据是多维数据库中的主要对象。依据如何分析和带有普通分级空间的数据，每个立体数据都包含一组尺寸。尺寸的度量是依据尺寸位于立体数据中的定量数据进行的。一个立体数据可以容纳大量的集合。

20.3.4Microsft OLAP Manager

Microsft OLAP Manager 是连接在 Microsoft Management 控制台中的。在该工具内部，用户可以创建新的数据库，在 ODBC Data Source Administrator 中设置 System 数据源名称 DSN 之后可以定义数据源。

通过右击新创建数据库下面的 Public Dimensions 文件夹，并从快捷菜单中选择 New Dimension，可创建新的尺寸。Dimension Wizard 允许用户贯穿创建新尺寸的每个步骤。Dimension Wizard 的最后一步进入 Dimension Editor，在此可以创建尺寸的层次。

当完成尺寸的创建时，通过右击数据库下面的 Cubes 文件夹，并从快捷菜单上选择 New Cube，可以建立立体数据。Cube Wizard 允许用户贯穿创建立体数据的各个步骤。Cube Wizard 的最后一步进入 Cube 编辑器，在此可以添加更多的公共尺寸、私人尺寸和选择立体数据的量度。

至此已准备好确定集合，通过右击分层树形结构中的新立体数据处理该立体数据，然后从快捷菜单从上选择 Process。Aggregations Wizard 允许用户贯穿设计集合的各个步骤，Aggregations Wizard 的最后一步将处理该立体数据，并显示计算的进程。

20.3.5 服务器分区

分区可把一个逻辑 OLAP 数据库放到不同的物理存储器中。对于每个 OLAP 数据库，缺省时可创建单一分区。把数据库分割成多个分区，就是用户如何允许不同的存储模式充分利用 Microsoft 的 OLAP Server 支持的 MOLAP、ROLAP 或 HOLAP 特性。

单一服务器分区是在单一服务器上，用在此可调谐集合的单一 DSN 创建的多个分区。在带有小比例集合的 ROLAP 中，可以存储极少使用的历史数据，以节省磁盘空间，在带有高比例集合的 MOLAP 中，可以存储当前数据，以提高性能。

多服务器分区可在多个服务器上创建，它并行使用多个 DSN 处理。不同的数据可以有相同的数据库结构，也许可以按年或地理类别进行分区。

在 Microsoft Management Console 的分层树形结构中，可选择立体数据下面的 Partitions 文件夹，并从快捷菜单上选择 New Partition。Partition Wizard 允许用户贯穿创建分区的各个步骤。

用户可以使用 ADO 的扩展 ADO MD 作为多维数据访问的程序对象模型。ADO MD 是以多维数据库方式使用的独立语言的数据访问接口。用户可以使用 Microsoft Visual Basic，以 ADO MD 作为对多维存储器中数据的数据访问接口。

在本章中，简单介绍了数据仓库领域和新的 OLAP Server。在下一章中，将介绍 Transact-SQL 编程语言的语法。