



轻松搞定 SQL Server 2000 程序设计

作者: **Rebecca M. Riordan**

译者: 良辰信息工作室

出版日期: 2001/04/09

书号: 957-2085-45-X

致谢

我要感谢 **Microsoft Press** 编辑团队的支持、他们的专业技术等等诸如此类的帮忙。依照时间前后的排列顺序，我依序要感谢：**Ben Ryan, David Clark, Alice Turner, Julie Xiao, Denise Bankaitis, Dail Magee, Jr., and Becka McKay。**

本书简介

欢迎阅读《轻松搞定 Microsoft SQL Server 2000 程序设计》。在本书中，首先将教您如何使用 SQL Server 2000 Enterprise Manager 来建立和管理数据库和数据库对象。然后会把重点放在 Query Analyzer 上，接着您就可以学习藉由使用指令码、触发程序、函数以及预存程序来建立一些便利性的程序。

SQL Server 2000 是一套功能强大、复杂的关系型数据库引擎。为了不让本书变得太厚，笔者撰写本书时采取了一些假设以及限制，笔者假设您对于关系型数据库如何设计已经有了具体的了解，您不需要有任何建置过数据库的经验，只要您能够按照本书中所有的课程实际操作即可。

在撰写书籍的时候总是会有一些功能强大的部份会被删除，虽然这是一个很自然的现象，但笔者为这个内文增减的问题困扰了好久。不过，SQL Server 2000 的强大功能部份，不是一本入门的书可以一次说完的。

本书的内容可以给您一些使用此产品的一些概念。而在彻底探索本书之前，您可能同时需要更了解其它一些相关的课题，例如 OLAP 或电子商务（e-commerce）的应用。

找出您自己最好的起始点

Microsoft SQL Server 2000 是一套客户端/伺服器端的关系型数据库管理系统，它可以用来设计高效率的在线交易处理（online transaction processing, OLTP）、资料仓储（data warehousing）

以及电子商务的应用等。而本书将帮助您快速了解如何使用 **SQL Server 2000** 个人版、标准版以及企业版等来建立、维护数据库以及一些由 **SQL Server 2000** 所提供的好用工具。

重要

本书虽然是使用 **SQL Server 2000** 来设计的，但是 **SQL Server 2000** 并没有包含在本书所附赠的光盘片中，您必须要购买 **SQL Server 2000** 的软件，并且在您开始学习之前必须先安装 **SQL Server** 。

本书是针对初学者学习关系型数据库所设计的，只要您有其它数据库管理系统的经验，例如 **Microsoft Access** 等，或是您想要升级到操作 **SQL Server 2000**，都可以阅读本书。您可以依照下列表格来找出阅读本书最好的起始点。

假如您是	依照下列步骤
学习关系型数据库系统新手	<div>1. 安装本章所介绍的范例档案。</div> <div>2. 藉由学习第一章至第五章来熟悉 SQL Server 的环境。</div> <div>3. 完整的学习第二篇至第四篇的课程。</div> <div>4. 完整的学习第五篇的课程。</div>

之前使用其它的数据库产品	<ol style="list-style-type: none"> 1. 安装本章所介绍的范例档案。 2. 藉由学习第一章至第五章来熟悉 SQL Server 的环境。 3. 完整学习其它您想要学习的课程。
将早期的 Microsoft SQL Server 升级	<ol style="list-style-type: none"> 1. 学习本章所介绍的〈SQL Server 2000 的新功能〉一节的说明。 2. 安装本章所介绍的范例档案。 3. 完整学习您所需要的课题，此表格可以帮您快速了解您的需求，而索引可以帮您找到一些特殊的课题。
将这本书当作是参考书籍	<ol style="list-style-type: none"> 1. 您可以使用索引来了解一些特殊的课题。使用此表格来找出一般性的课题。 2. 您可以阅读每一章的后面所附的〈本章总结〉来了解每一章的重点。

SQL Server 2000 的新功能

下表为 Microsoft SQL Server 2000 一些主要的新功能，以及本书中哪些章节中有说明这些新的功能，当然您也可以使用索引来找出您想要的特殊信息。

要学习如何	请参考
-------	-----

在 Enterprise Manager 中新增数据行的描述	第五章
在有主索引键的数据表进行串联更新	第七章
在检视表上建立索引	第十六章
在关联性中建置串联删除	第十九章
宣告 Table 的数据型别，也就是变量可以用来储存数据表型态的值	第二十五章
建立一个 INSTEAD OF 触发程序	第二十九章
建立使用者自订函数	第三十章

使用随书光盘

本书所附的光盘中包含了本书所有的练习题所需的档案。您可以使用这些档案，并且您不需要浪费时间来建立数据库以及输入范例数据，您只要全神贯注在如何学习 **Microsoft SQL Server 2000** 即可。使用这些范例档案以及书中所提供的按部就班步骤，您将可学习到如何执行，并且容易地取得相关的新技术。

重要

在您开始使用随书光盘之前，您必须要先确定您所使用的 **Microsoft SQL Server 2000** 的版本是否和本书所使用的版本相符合。本书中所使用的 **Microsoft SQL Server 2000** 是可以安装在 **Microsoft Windows NT** 以及 **Microsoft Windows 2000** 操作系统的版本。

系统需求

为了可以安装并且执行 **Microsoft SQL Server 2000**，您的计算机必须符合下列的需求：

- **Intel**（或兼容）**Pentium** 处理器，并且至少要在 **166 MHz** 以上。
- 至少要 **64 MB** 的内存，如果是企业版时建议使用 **128 MB** 的内存。
- 至少要 **95 MB** 的硬盘空间；如果使用一般安装以安装 **SQL Server** 的数据库完整的组件则需要 **250 MB**。
- 至少要 **50 MB** 的硬盘空间；如果要安装 **SQL Server** 分析服务则需要 **130 MB**。
- **80 MB** 的硬盘空间安装 **English Query**。

- 44 MB 的硬盘空间只安装 Desktop Engine。
- 至少要 800×600 的屏幕分辨率。
- Microsoft 鼠标或兼容性的鼠标，以及光驱。
- Microsoft Internet Explorer 5.0 或以上的版本。

安装范例档案

要将范例档案安装在您的计算机上，您必须要有大约 6.21 MB 的硬盘空间，请您按照下列步骤

将范例档案安装在您的计算机上，这样您就可以开始学习了：

重要

要使用这些范例档案，您必须要有足够的 SQL Server 执行个体的安全性权限。假如您没有登入

SQL Server 的名称和密码或因其它问题而导致无法继续执行时，您可以询问您的系统管理者。

1. 将光盘片放入您的光驱中（假如菜单没有出现时，请在光盘片的根目录中直接按两下 **StartCD.exe**）。
 2. 按一下 **Install Practice Files** 选项，然后按照屏幕上所显示的步骤来进行。
 3. 当您安装完毕之后，在您的硬盘内会出现一个 **SQL 2000 Step by Step** 的档案目录，并且有一些范例档案存放在此档案目录内。
 4. 启动 SQL Server Enterprise Manager。
 5. 展开左侧的 **Microsoft SQL Servers**，再展开 **SQL Server 群组**。
-

重要

您可以向您的系统管理者询问您将要使用哪一个服务器来作为练习之用。

6. 在 **数据库** 数据夹上按右键，并在快捷菜单中选取 **所有工作** 项目内的 **附加数据库** 项目。

此时，SQL Server 会显示一个 **附加数据库** 的对话框。

7. 按一下 **浏览** 按钮。

此时 SQL Server 会显示一个 **浏览现有的档案** 对话框。

8. 选取 **SQL 2000 Step by Step** 数据夹，然后选取 **Aromatherapy.mdf** 文件名称，

接着再按一下 **确定** 按钮。

9. 按一下 **确定** 按钮。

此时 SQL Server 会显示一个 **已经成功地附加数据库** 对话框。

除了安装这些范例档案之外，安装程序会在您的桌面上建立一个 **Micorosft**

Press World Wide Web 网站的快捷方式。假如您的计算机可以连接至因特网时，

您就可以在此快捷方式上按两下以联机至 **Microsoft Press** 的网站中，您也可以

直接联机至 <http://mspress.microsoft.com/>。

使用范例档案

我们所提供的范例数据库在第一篇至第四篇中所有的章节都会使用到。而在第五篇的章节中也有

其它的范例档案。在每一章节中我们都会告诉各位如何使用这些档案，您可以藉由使用这些档案

来达到学习的目的地。

反安装范例档案

您现在可以按照下列步骤所示，将所安装的范例档案自您的计算机中移除。请您注意，当您要将这些范例档案移除安装之前，您必须将 SQL Server 中的 Aromatherapy 数据库先进行卸离的动作。

1. 按一下 [开始/设定/控制台](#) 。

重要

在 Windows 2000 操作系统中已经将 [控制台](#) 项目直接放置在 [开始](#) 菜单内。假如您的计算机也是如此设定时，[控制台](#) 内的项目将会出现在 [设定](#) 菜单内的子菜单中。

-
2. 按两下 [新增/移除程序](#) 图标。
 3. 自列表选取 [SQL 2000 Step by Step](#) 项目，然后按一下 [新增/移除](#) 按钮（在 Windows 2000 操作系统中，请按一下 [变更/移除](#) 按钮）。

此时会出现一段确认讯息。

4. 按一下 [是](#) 。

此时范例档案将会被反安装。

5. 按一下 **确定** 按钮，以便将 **新增/移除程序** 对话框关闭。
6. 假如有需要，请将 **控制台** 窗口关闭。

问题的修正、建议以及求助

为了确定本书的内容正确无误，**Microsoft Press** 提供了关于问题的修正以及提供这本书额外的讯息来让您知道，您可以拜访下列的网址：

<http://www.microsoft.com/taiwan/mspress>

<http://www.ibookpress.com>

我们殷切的盼望您拜访我们的网站！！

第一篇 使用 SQL Server

1. SQL Server 2000 的环境

- . 认识 Enterprise Manager
- . 控制 SQL Server
- . **Enterprise Manager** 主控台树状目录
- . 系统数据库
- . 连接至现存的数据库中
- . 本章总结

2. 管理 SQL Server

- . 备份及还原数据库
- . 本章总结

3. SQL Server 2000 的安全性

- . 认识安全性模式
- . 使用者登入账户
- . 数据库层级的安全性
- . 本章总结

1. SQL Server 2000 的环境

在本章中，您将学习到：

- 启动 Enterprise Manager。
- 注册服务器。
- 启动和停止服务器。
- 在 Enterprise Manager 主控台树状目录之下显示对象。
- 结束 Enterprise Manager。

Microsoft SQL Server 2000 已经被设计成一个在执行效能、可靠度、品质等方面有显著增益的数据库，它也提供了一些不同的应用范围，这其中包括在线交易处理（online transaction processing, OLTP）、资料仓储（data warehousing）以及电子商务（e-commerce）等。为了可以支持这些功能，SQL Server 提供了一些工具，这其中包括命令列提示工具，例如 `bcp.exe`，它可以用在 SQL Server 和操作系统档案之间大量复制数据；以及 Enterprise Manager，它是一个很复杂的图形化工具，可以用来管理多个数据库以及 SQL Server 本身。

本书大部份的章节中将使用图形化工具 Enterprise Manager，在第一章中我们将学习在此环境中应该如何工作。

认识 Enterprise Manager

SQL Server Enterprise Manager 是用来管理 SQL Server 2000 的主要管理工具,使用 Enterprise

Manager 的图形化接口,您可以:

- 定义一个 SQL Server 执行个体的群组,并且将独立的服务器注册到群组中。
 - 针对每一个注册的服务器设定所有的选项。
 - 建立和管理所有的 SQL Server 数据库、对象、登入账号、使用者以及每一个已经注册的服务器权限。
 - 在每一个已注册的服务器中定义,并且执行所有的 SQL Server 管理工作。
 - 检视数据表的内容,并且使用 [查询设计师](#) (Query Designer) 来进行检视。
 - 设计并测试 SQL 陈述式,并且藉由 SQL Server Query Analyzer 来进行编写指令码的工作。
 - 藉由各种精灵来完成工作。
-

Microsoft Management Console

Microsoft Windows 中包含一种称为 Microsoft Management Console (MMC) 的工具，它提供了管理服务器应用程序的标准管理接口，称为 **主控台** (console)。大多数服务器主控台的型态是属于嵌入式管理单元 (snap-in)，而您用来管理 SQL Server 2000 的 SQL Server Enterprise Manager 就是属于嵌入式管理单元。关于 MMC 更多的信息，您可以藉由按一下 **开始** 菜单来选取 **说明**，然后在 **搜寻** 标签页中输入『Microsoft Management Console』。

启动 Enterprise Manager

在您开始执行管理工作之前您必须要先执行 SQL Server Enterprise Manager。

启动 Enterprise Manager

1. 在 Microsoft Windows 的工作列上按一下 **开始** 按钮。
2. 指向 **程序集/ Microsoft SQL Server** 数据夹。

Microsoft SQL Server 数据夹清单会出现一个图标。

3. 选取 **Enterprise Manager** 图示。

此时 Enterprise Manager 窗口将会显示，如图 1-1 所示：

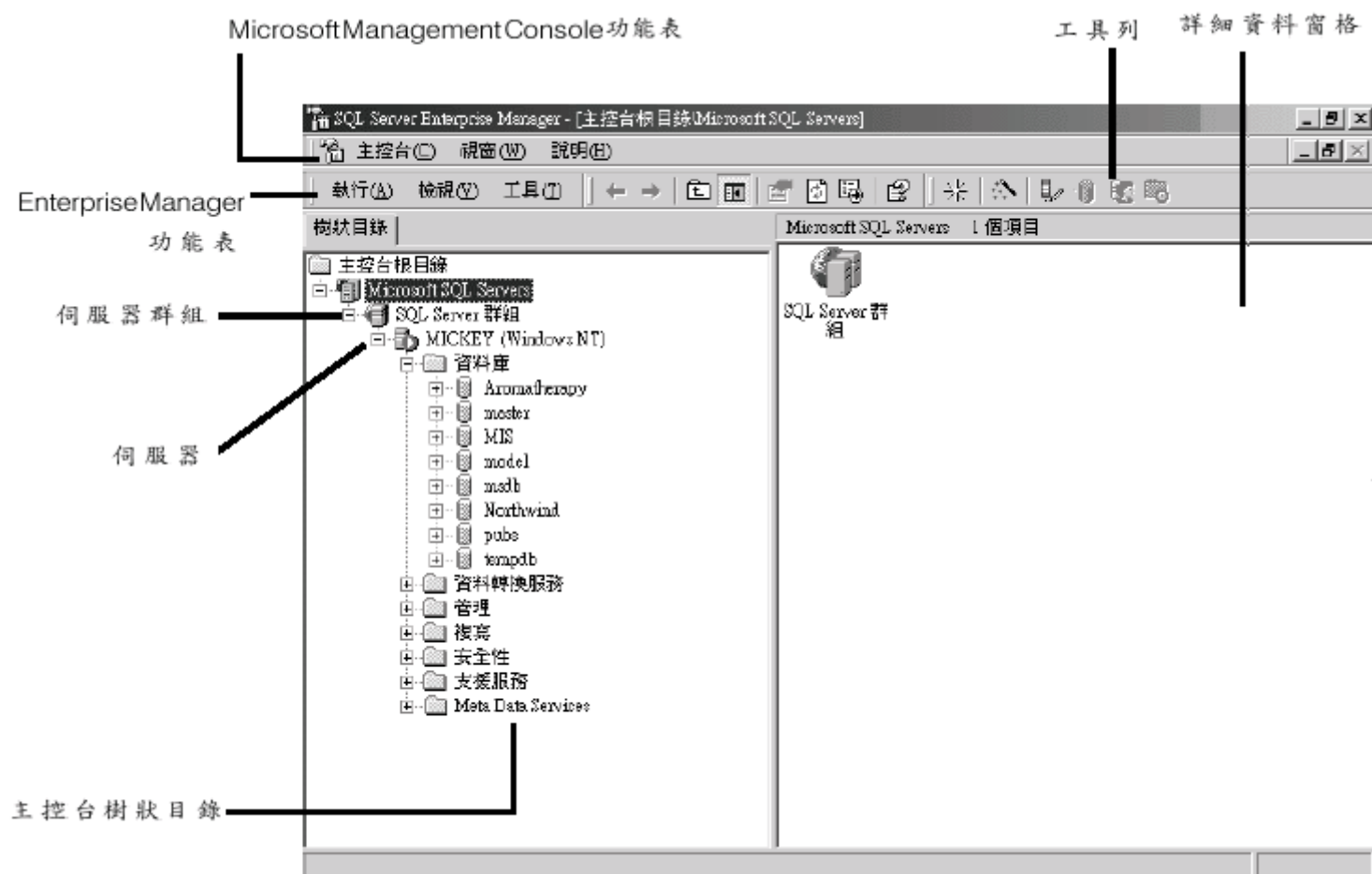




图 1-1 Enterprise Manager 窗口

Enterprise Manager 提供您管理 SQL Server 所需要的工具，以便让您建立和维护数据库。

Enterprise Manager 窗口包含二个窗格：左边窗格的 [主控制台树状目录](#)，以及右边的 [详细数据](#) 窗格。在主控制台树状窗格内的项目是以阶层式的方式来进行排列，就像 Microsoft Windows 的档案总管显示的方式一样，您可以藉由按一下展开图标来展开子项目，或者是藉由按一下折迭图标将这些子项目折迭起来。

展开图示： 

折迭图示： 

重要

当您启动 Enterprise Manager 之后会在主控制台树状目录之下来显示对象，其所显示出来的对象可能不会和本书所提供的范例一样，举例来说，书中的服务器名称是 MICKEY，而您的服务器名称应该不会和本书一样。

控制 SQL Server

在您可以使用 **SQL Server Enterprise Manager** 来建立新的数据库或者是存取已经存在的数据库之前，您必须先确认此服务器的执行个体已经连到 **Enterprise Manager** 中，并且确定此服务器正在执行当中以及已经连上您要进行工作的数据库中。

SQL Server 安全性模式

SQL Server 提供二种不同登入的安全性模式：**Windows 验证**及 **SQL Server 验证**。**Windows 验证**的方式是 **Microsoft** 建议采用的，它可以让 **Windows 2000** 和 **Microsoft Windows NT** 的使用者使用他自己的操作系统的使用者名称和密码来进行登入的工作。当您使用 **SQL Server** 的验证方式时，**SQL Server** 本身会控制使用者验证以及当他们要连接至数据库中时，其使用者必须提供一个登入的名称以及密码。

本书是采用 **Windows** 的验证方式，假如您的服务器是使用 **SQL Server** 验证方式时，当您要注册此服务器或连接至数据库时，您将需要提供您的登入账号及密码。假如发生这种情形时，您必须输入由系统管理者所提供的登入名称及密码的数据，然后按一下 **确定** 按钮。

我们将在第 3 章中说明 **SQL Server 2000** 安全性的相关课题。

服务器的注册

当您第一次执行 **Enterprise Manager** 时，它会自动将所有 **SQL Server** 正在执行的执行个体进行注册，假如您安装新的 **SQL Server** 的执行个体或者是您想要透过网络的存取来连接至 **SQL Server** 的执行个体时，您必须要将它进行注册。

重要

在您开始将 **SQL Server** 的执行个体进行注册之前，您需要此服务器的名称，以及登入该服务器的安全性方式、名称和密码。假如您不知道相关的资料，您可以询问您的系统管理者。

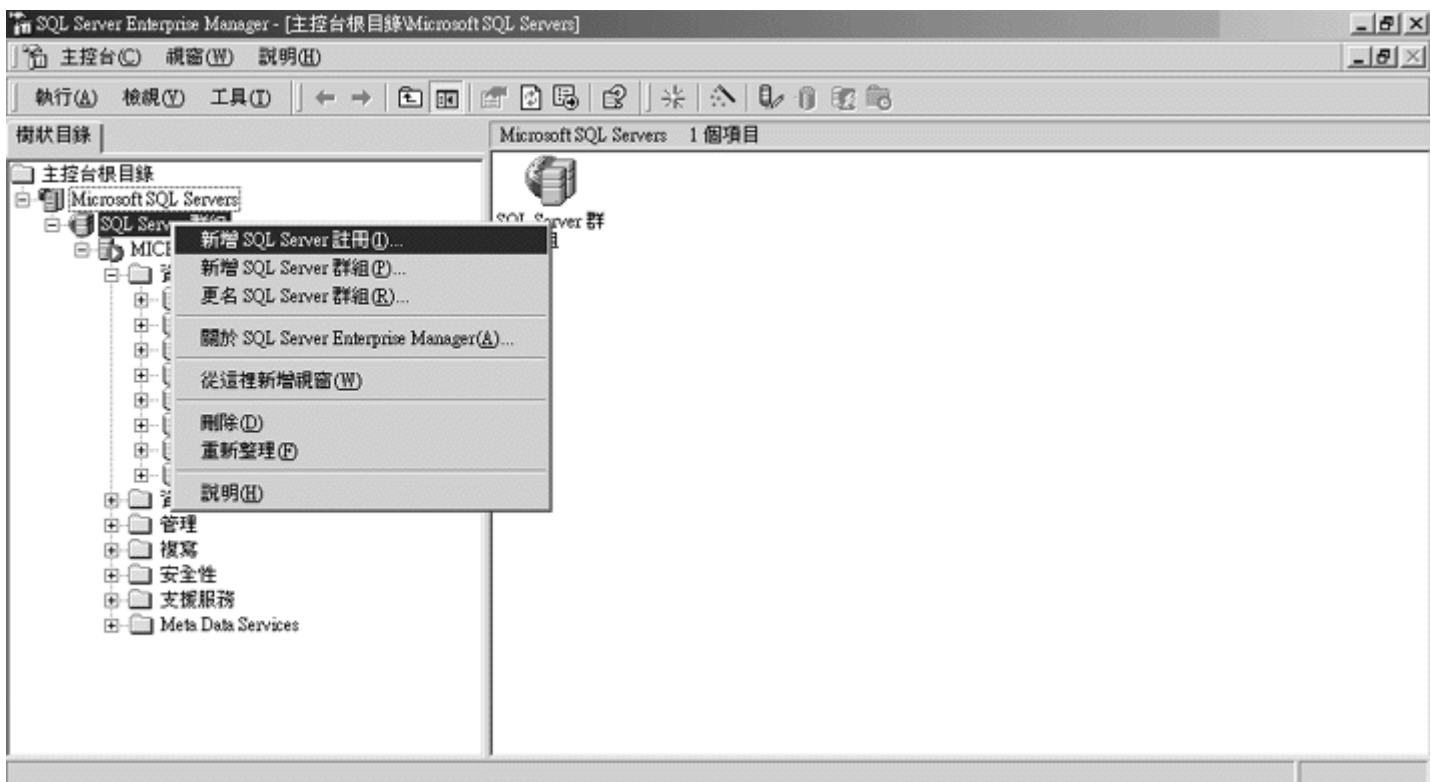
注册服务器

重要

如果在主控台树状目录之下的服务器名称已经显示出来时，那就表示它已经藉由 **Enterprise Manager** 注册完毕，那您就不用再执行这个步骤了，而直接跳至下一节〈启动和停止服务器〉。

1. 在控制台树状目录之内的 SQL Server 群组上按右键。

此时会显示快捷菜单。



2. 在快捷菜单中选取 **新增 SQL server 注册** 项目。

此时会在屏幕上出现一个注册 SQL Server 精灵的欢迎使用画面。



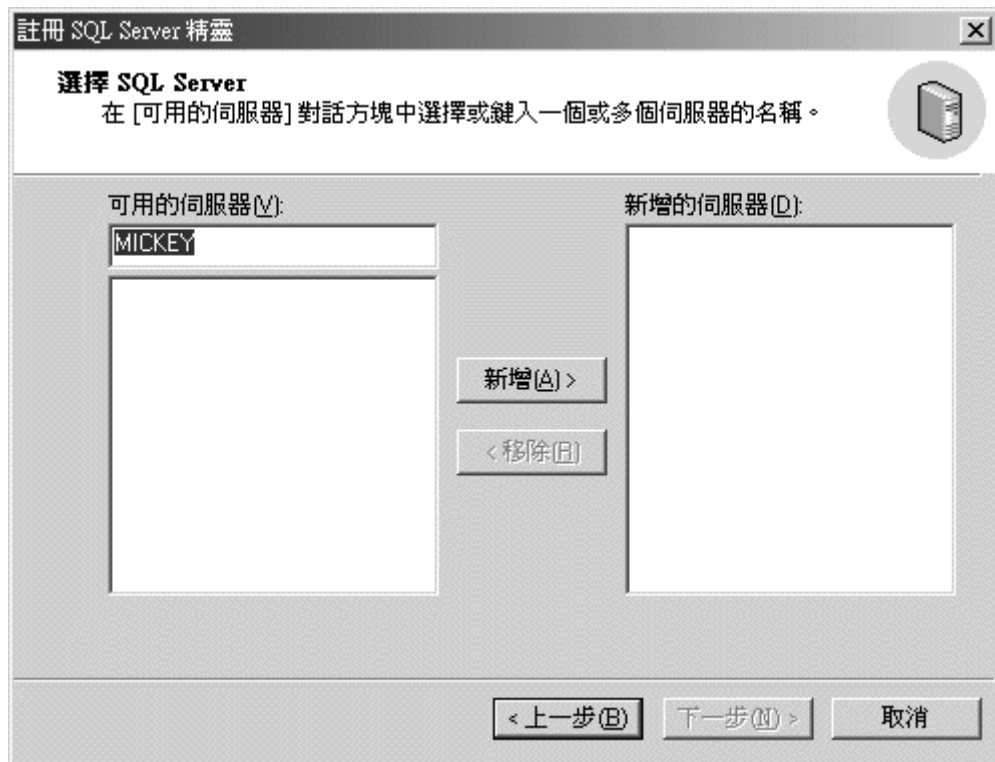
提示

您也可以在主控台树状目录内的服务器名称上按右键，自快捷菜单中选取 **新增 SQL**

Server 注册 项目。

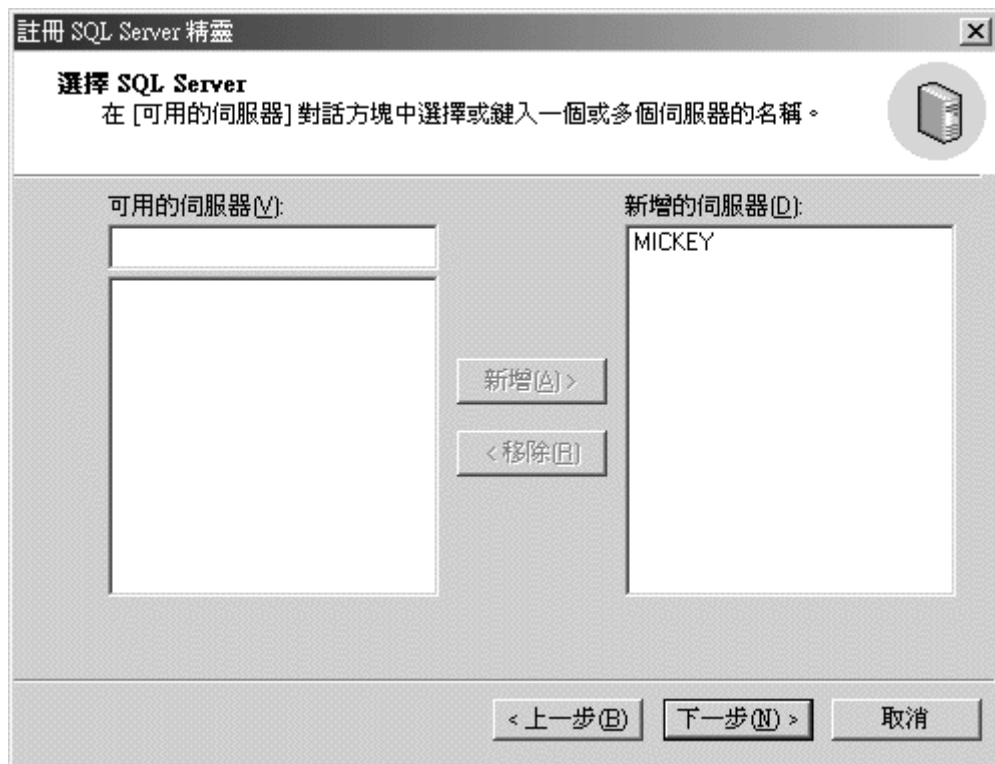
3. 按一下 **下一步** 按钮。

注册 **SQL Server 精灵** 会显示第一页的欢迎画面，并显示可用的服务器列表。



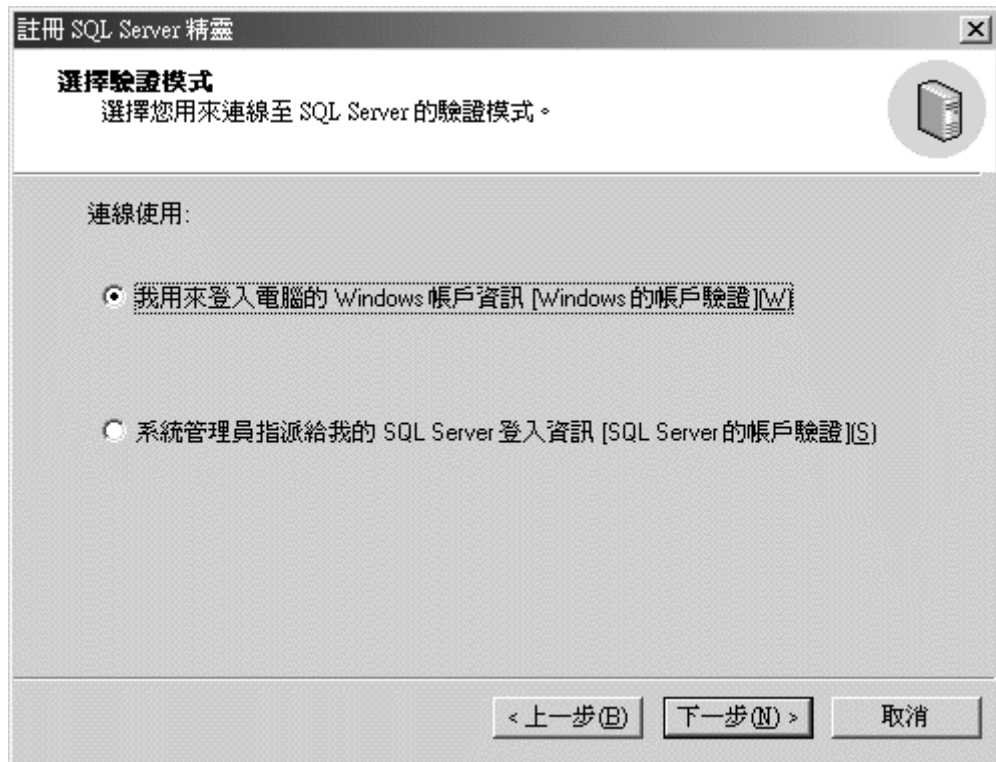
4. 假如您想要注册的服务器出现在列表内，请在列表选取，并且按一下 **新增** 按钮。假如服务器没有出现在列表内，将服务器的名称输入到文字区块中，然后按一下 **新增** 按钮。

此精灵会把您所选取的服务器加入到新增的服务器列表中。



5. 按一下 **下一步** 按钮。

此精灵会询问您联机至 SQL Server 的验证方式。



6. 假如您的系统管理者已经告诉您要使用 SQL Server 的验证方式时，请选取 [系统管理员指派给我的 SQL Server 登入信息 \[SQL Server 的账户验证\]](#) 项目。
7. 按一下 [下一步](#) 按钮。

假如您是选取 SQL Server 的验证方式，请输入登入名称及密码数据。输入完毕之后，请按 [下一步](#) 按钮。

此精灵会显示一个请您指定要进行注册的 SQL Server 群组的画面。



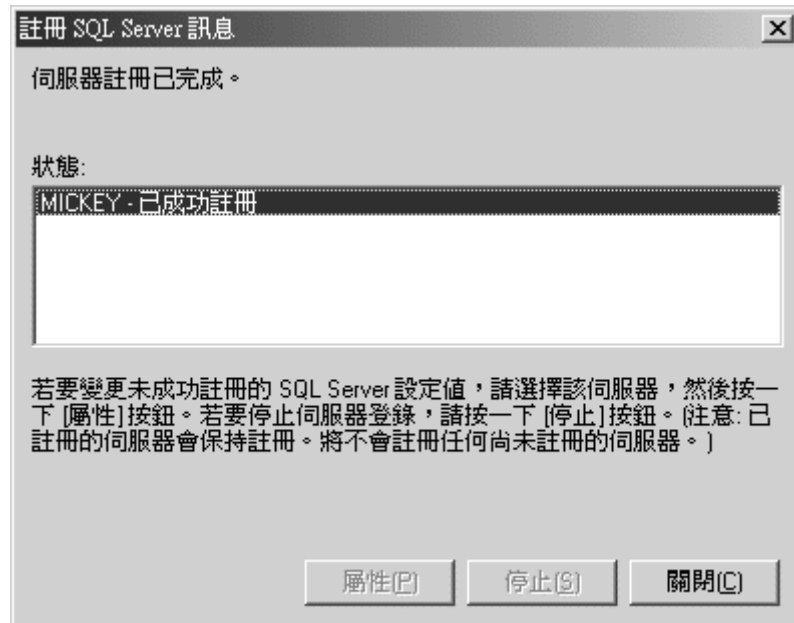
8. 按一下 **下一步** 按钮以便将您所选取的服务器加到预设的群组中。

此精灵会显示一个您要加入的服务器名称是否正确的画面。



9. 按一下 **完成** 按钮。

此精灵会显示工作已经执行完成并且服务器注册已经成功的画面。



10. 按一下 [关闭](#) 按钮。

此注册 SQL Server 精灵窗口将会关闭，并且主控台树状目录会重新显示数据。

提示

您只需要将 SQL Server 的执行个体注册一次即可，当您下次启动此程序时，Enterprise Manager 会记得此服务器已经注册过了。

启动和停止服务器

在您和 SQL Server 的执行个体进行连结时，此服务必须要启动，您可以藉由在 Enterprise Manager 的主控台树状目录之下的服务器图标来确认此服务是否正在执行中。表 1-1 为每一个图示所代表的意义。

图标符号	意义
	服务器目前正在执行中。
	服务器目前暂停使用。
	服务器目前停止使用。

表 1-1 这些服务器图示是代表 SQL Server 目前正在执行的状态

启动服务器

- 在服务器名称上按一下右钮，并在快捷菜单中选取 启动 项目。

暂停服务器

- 在服务器名称上按一下右钮，并在快捷菜单中选取 暂停 项目。

停止服务器

- 在服务器名称上按一下右钮，并在快捷菜单中选取 **停止** 项目。

重要

请确定本课程的服务器已经在执行中。

Enterprise Manager 主控台树状目录

Enterprise Manager 的主控台树状目录窗口内会显示所有的 SQL Server 对象。

重要

Enterprise Manager 的主控台树状目录内所显示的对象可能和您的系统所提供的对象内容不

太一样，但是您不需要担心这些问题，因为您的系统管理者知道这其中的差异性。

当您进行本书中的课程范例时，您将会使用主控台树状目录内大多数的项目。为了能够让您了解，

表 1-2 为主控制台树状目录内每一个数据夹内所包含的主要对象的说明。

图标符 号	资料夹	说明
	服务器群组	可将一个或多个服务器分组，这样比较易于管理。
	服务器	是 SQL Server 执行个体，是使用 Enterprise Manager 注册。
	数据库	集合数据表和其它的对象，是储存结构化数据的集合。
	数据转换服务	是一套图形化工具及针对管理 SQL Server 的可程序化对象，以便让数据可以执行转换和合并。
	管理	是一套图形化工具及针对管理 SQL Server 的可程序化对象。
	复写	是一套图形化工具及针对管理 SQL Server 的可程序化对象，以便让数据和数据库对象进行复制并且分散至另一个服务器中。
	安全性	是一套图形化工具及针对管理 SQL Server 的可程序化对象，以便控制存取 SQL Server 的使用者。
	支持服务	各种管理 SQL Server 的工具。
	Meta Data	维护数据库的中继数据（关于数据的数据）的工具。

	Services	
--	----------	--

表 1-2 Enterprise Manager 主控台树状目录内的主要对象

系统数据库

Enterprise Manager 的主控台树状目录内通常会包含四个由 SQL Server 本身所使用的数据库。

这些数据库会被当作系统数据库。表 1-3 中为这些数据库在 SQL Server 内所扮演的角色说明。

重要

因为系统数据库的完整性会影响到 SQL Server 是否操作成功，所以您必须要使用 Enterprise Manager 的工具或程序化界面（SQL-DMO）来修改 master 和 msdb 数据库的内容。

系统数据库	说明
master	master 数据库记录了管理 SQL Server 系统所需的所有信息，包含所有的登入使用者、系统中定义的数据库和服务器的处理程序。储存在 master 数据库中的数据表称为 system catalog 。

model	model 数据库即为数据库模板，于建立新数据库时使用。
	预设的 model 数据库包含建立 database catalog 的数据表，这些数据表用于让 SQL Server 定义使用者数据库的其它对象。
msdb	msdb 数据库是用来记录 SQL Server 代理程序服务项目，如排程与工作项目、警示和作业等等，并且会有历程纪录。警示为使用者定义响应给 SQL Server 的事件；作业是由 SQL Server 代理程序执行的一系列动作。
tempdb	tempdb 包含了所有的暂存数据表与暂存的预存程序。它也可填满任何其它的暂时储存需求。当 SQL Server 结束执行时，其内容会自动地移除。事实上，任何一个暂存对象都可以由使用者来建立。

表 1-3 SQL Server 的系统数据库

提示

任何您在 **model** 数据库内所加入的对象，都会自动地加入到您所新建立的数据库中，举例来说，当您想要在所有新建立的数据库中有某种特定的使用者或数据库设定选项时，这个功能就非常地好用。

连接至现存的数据库中

既然服务器已经在 **Enterprise Manager** 中注册过并且也已经启动时，您就可以建立新的数据库或连接到一个已存在的数据库中。

重要

假如您尚未安装本书所提供的范例数据库时，您可以先跳至〈本书简介〉的 [〈安装及使用范例档案〉](#) 一节中来安装范例数据库，然后待安装完毕之后再回到本课程中来继续学习。

数据库对象

SQL Server 数据库是属于 **SQL Server** 主控台树状目录内许多项目的一部份，如图 1-2 所示。这些数据库包含各种对象，它可以用来定义储存在数据库内的数据以及数据存取的方式。

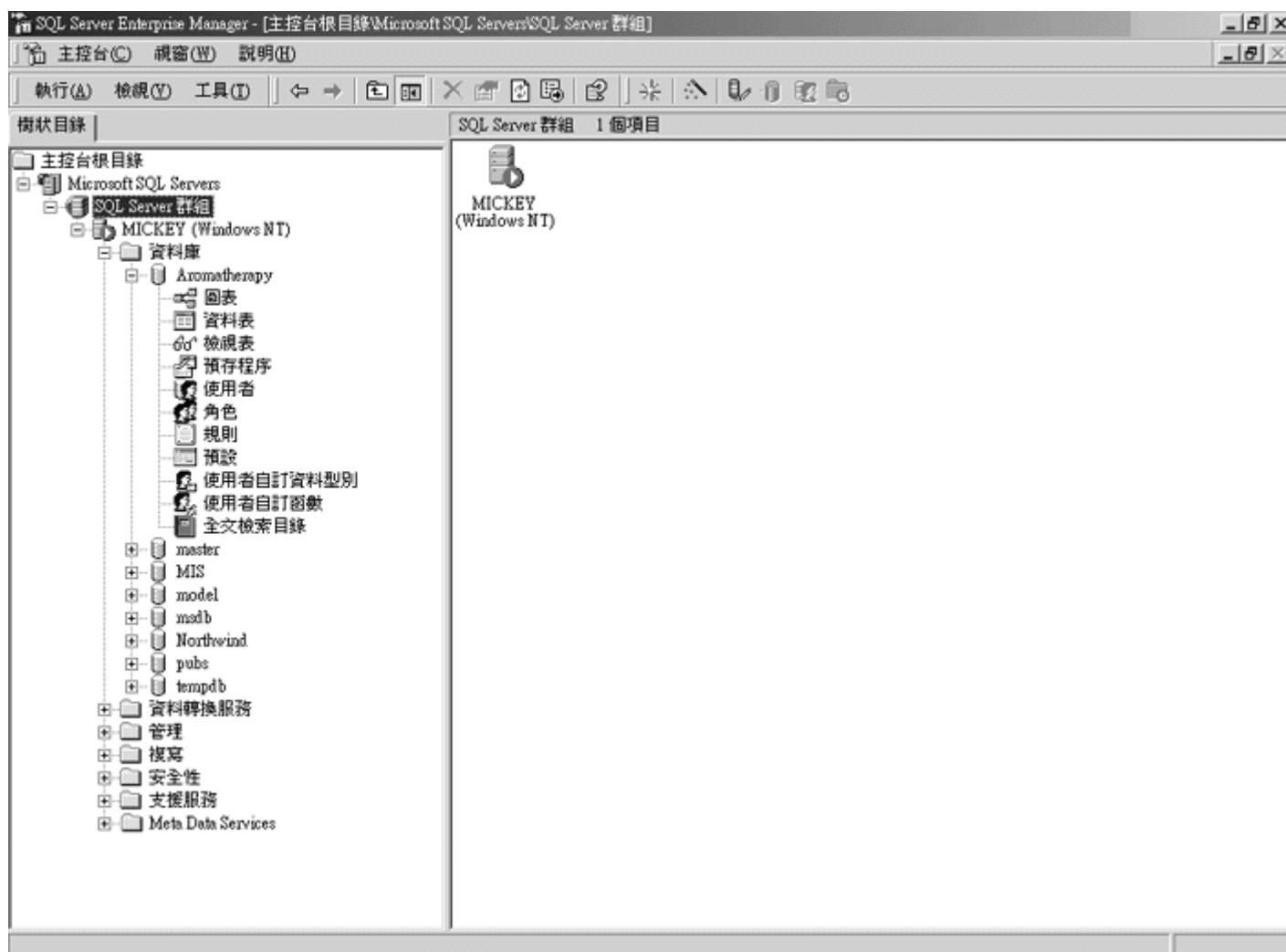




图 1-2 在 SQL Server 主控台树状目录之内的数据库对象

我们将在稍后的课程中了解这些对象的用法，您可以先参考表 1-4 的说明。

图标 号	物件	说明
	图表	在数据库中将数据表以图形化方式呈现。
	资料表	将信息组织化，分成数据行和数据列的方式储存。

	检视表	虚拟数据表，是另一种检索数据库信息的方法。
	预存程序	T-SQL 命令的集合，并且以批次方式执行。
	使用者	基于安全性的目的，系统用以辨识每一个独立使用者的方法。
	角色	基于安全性的目的，将权限以群组的方式分派。
	规则	数据库对象的一种，用以系结数据行或使用者定义的数据型别，以指定数据如何输入数据行。
	预设	假如使用者没有输入值时，系统会自动指派值。
	使用者自订数据型别	一个由使用者所自订定义的数据型别。
	使用者自订函数	T-SQL 命令的集合，可以接收参数和传回值。

表 1-4 在 SQL Server 数据库中所有包含的对象

结束 Enterper Manager

当您要结束 Enterprise Manager 时，您必须在关机之前先结束程序的执行。

结束 Enterprise Manager

- 在主控台菜单中，选取结束。假如出现一个另存新文件的对话框，请选取 [是](#)。

此时 Enterprise Manager 将会关闭。

假如您要继续下一个课程

- 再次启动 Enterprise Manager 并且到第 2 章继续学习的课程。

假如您现在要停止执行 Enterprise Manager

- 假如您已经离开 Enterprise Manager，您就已经停止执行了。

本章总结

要执行的工作	执行	按钮
启动 Enterprise Manager	在工作列上按一下 开始 按钮，指向 程序集/Microsoft SQL Server/Enterprise Manager 。	

注册服务器	在主控台树状目录中的服务器或服务器群组上按右钮，在快捷菜单中选取 新增 SQL Server 注册 。	
启动服务器	在服务器上按右钮，在快捷菜单中选取 启动 。	
暂停执行服务器	在服务器上按右钮，在快捷菜单中选取 暂停 。	
停止执行服务器	在服务器上按右钮，在快捷菜单中选取 停止 。	
在对象树状目录中展开一个项目	在项目的左边展开图标按一下。	
在对象树状目录中折叠一个项目	在项目的左边折叠图标按一下。	
离开 Enterprise Manager	在 主控台 菜单中选取 结束 。	

2. 管理 SQL Server

在本章中，您将学习到：

- 如何备份数据库。
- 如何还原数据库。
- 如何使用数据库维护计划精灵来建立维护计划工作。

Microsoft SQL Server 就像您的爱车一样都需要定期进行保养，好将车况保持在最佳状态，因此 Microsoft SQL Server 提供了一些不错的工具，让您需要执行一些定期性的工作时可以使用，好将 Microsoft SQL Server 保持在最佳的执行状态。

SQL Server 所提供的定期保养工具的操作方式比起更换您爱车的机油还要简单得多，一般人都能轻易的上手操作。

身为决策者的系统管理人员，所要处理的事情通常非常的复杂，SQL Server 了解身为系统管理人员的辛苦，便提供了 Enterprise Manager 来简化复杂的工作。在本课程中，您将学习到如何容易地进行保护、备份数据库，以及如何使用数据维护计划精灵来管理您的 SQL Server。

备份及还原数据库

当下列情况发生时，是需要第一优先处理的：当计算机的硬件发生故障时、软件发生一些不明状况或当机时，数据将发生遗失的现象。举例来说，当计算机正在处理某些与硬件相关的程序而发生硬件故障时，其程序因无法侦测硬件发生故障而一直在执行，将会导致程序一直执行或当机的现象。假如您有一些必须和硬件结合的程序要执行时，数据保护的课题就相当重要了，一般最常见的保护措施如：将重要的数据复制一份，然后将拷贝版本的数据储存在您认为安全的地方，这种将数据复制一份，再将拷贝版本的数据储存在安全的地方的动作就称之为「备份」。假如您的数据发生重大毁损（如磁盘毁坏、停电）时，您就可以将之前储存在安全地方的拷贝版数据再复制回来使用，这种动作就称之为「还原」。

将数据库进行备份

SQL Server 2000 提供了数种方法，让您将数据库进行备份。其中最简单的方法就是将整个数据库进行备份，也就是 **完整备份**（Full Backup）。所谓的完整备份是指将整个数据库进行备份，而此种备份数据库的方式是既安全又有效的备份方法。

提示

当您想要将数据库进行备份时，SQL Server 提供了一个进阶的功能，您可以备份其它使用者正在使用的数据库，而不需强迫使用者中断 SQL Server 的联机。但是为了确保数据库备份的完整及正确，有些状况之下还是不能进行备份，这些状况包括：使用者正在更改数据库结构，例如建立、删除档案或建立索引等状况。

SQL Server 也提供了另一种备份方式，那就是 **差异备份**（Differential Backup）。所谓的差异备份，它只备份自上一次进行完整备份之后所改变的数据。差异备份的好处是在于它只备份更改过的数据，因此所要备份的数据量也会比完整备份的数据量还要小，当然其执行的速度也会比完整备份的速度还要快。

提示

假如您想要将已变更过的资料进行备份，您可以使用差异备份。利用此特性，您可以将差异备份设定在每天进行，而备份速度慢的完整备份就可以设定一星期进行一次备份即可。

SQL Server 提供的第三种备份方式是 **交易记录文件备份**（Transaction Log Backup）。所谓的交易记录备份是指它会将自上次备份后的所有数据库的交易记录文件进行备份。

提示

交易记录文件备份可以让您将数据库还原至某一个时间点，这种方式非常好用。举例来说，当一位操作者将错误的数据输入到数据库内时，您可以使用交易记录将错误的数据还原至发生错误之前的正确数据。

数据库发生错误时，可以使用交易记录文件来进行还原，但是当 **SQL Server** 发生错误时，该如何拯救呢？当 **SQL Server** 发生错误时，**SQL Server** 也会自动使用交易记录文件进行数据库的还原，并且您也可以使用完整备份或差异备份将您的数据进行整合的工作。使用交易记录文件备份的好处是所产生出来的备份文件容量比您使用完整备份或差异备份所备份出的档案容量还要小得多。

提示

在某些情形之下，交易记录文件备份的档案容量会比数据库备份的档案容量还要大。举例来说，假如有一些经常被更改内容的交易记录文件要进行交易记录文件备份时，您可以选择要使用完整备份或交易记录文件备份的方式来进行备份。

使用数据库备份精灵来备份数据库

重要

假如您还没有安装本书所附的 **Aromatherapy** 数据库范例时，您可以回到前面的〈本书简介〉中来安装 **Aromatherapy** 数据库范例。

1. 在主控台的树状目录之中，选取 **Aromatherapy** 数据库。

SQL Server 会在 [详细资料窗格](#) 中显示数据库的对象。



2. 在 Enterprise Manager 的工具列中按一下 [執行精靈](#) 按钮，此时 SQL Server 会显

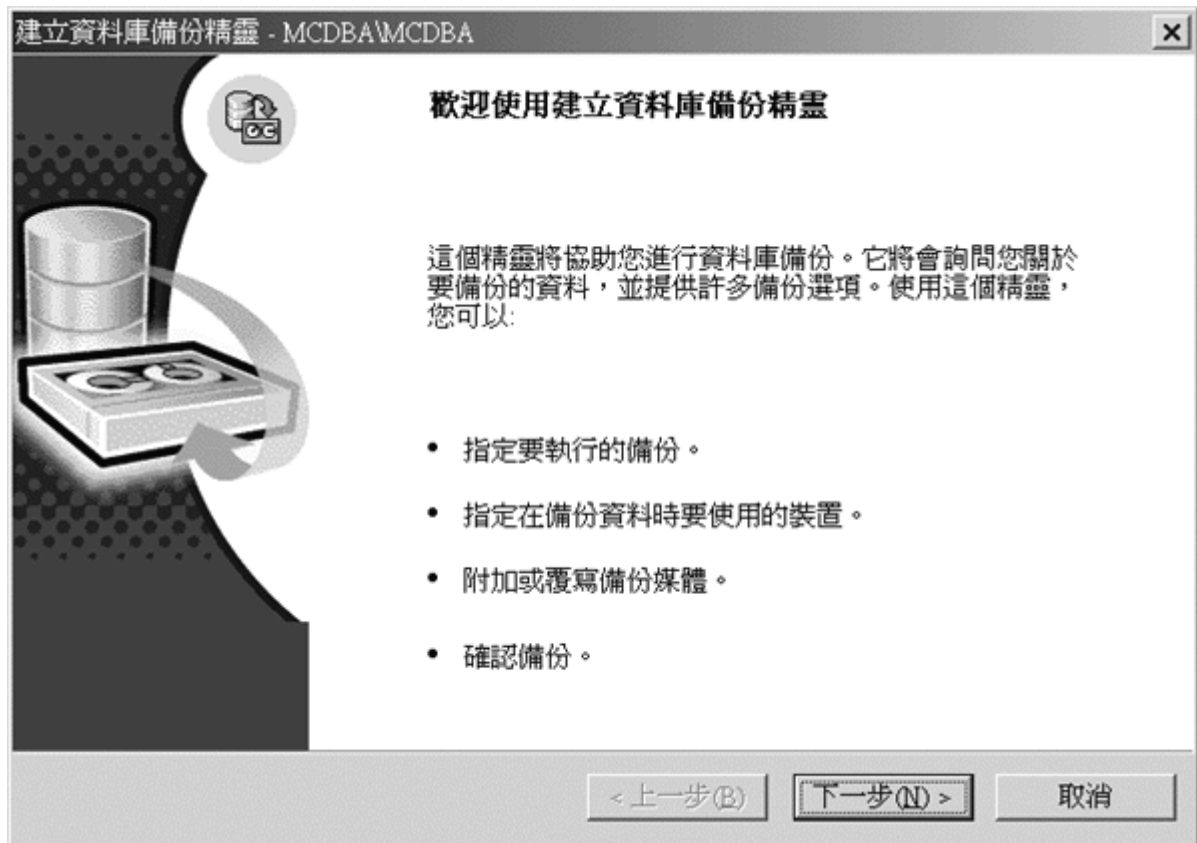
示 [选择精灵](#) 的对话框。



执行精灵按钮



3. 在 [管理](#) 項目中選取 [備份精靈](#)，此時 SQL Server 會顯示 [建立数据库备份精灵](#)。



4. 按一下 **下一步** 按钮。

此时备份精灵会显示您要选择哪一个数据库来进行备份的画面。



5. 请您确定 Aromatherapy 数据库已经在下拉式方块中被选取，然后接着请按一

下 **下一步** 按钮。

此时备份精灵会显示询问您备份的名称以及描述内容的窗口。

建立資料庫備份精靈 - MCDBA\MCDBA

輸入備份的名稱及描述
輸入備份的名稱及描述。

名稱(A):
Aromatherapy 備份

描述(S):

< 上一步(B) 下一步(N) > 取消

6. 請您在描述文字方塊中输入『Lesson 2 backup』。

建立資料庫備份精靈 - MCDBA\MCDBA

輸入備份的名稱及描述
輸入備份的名稱及描述。

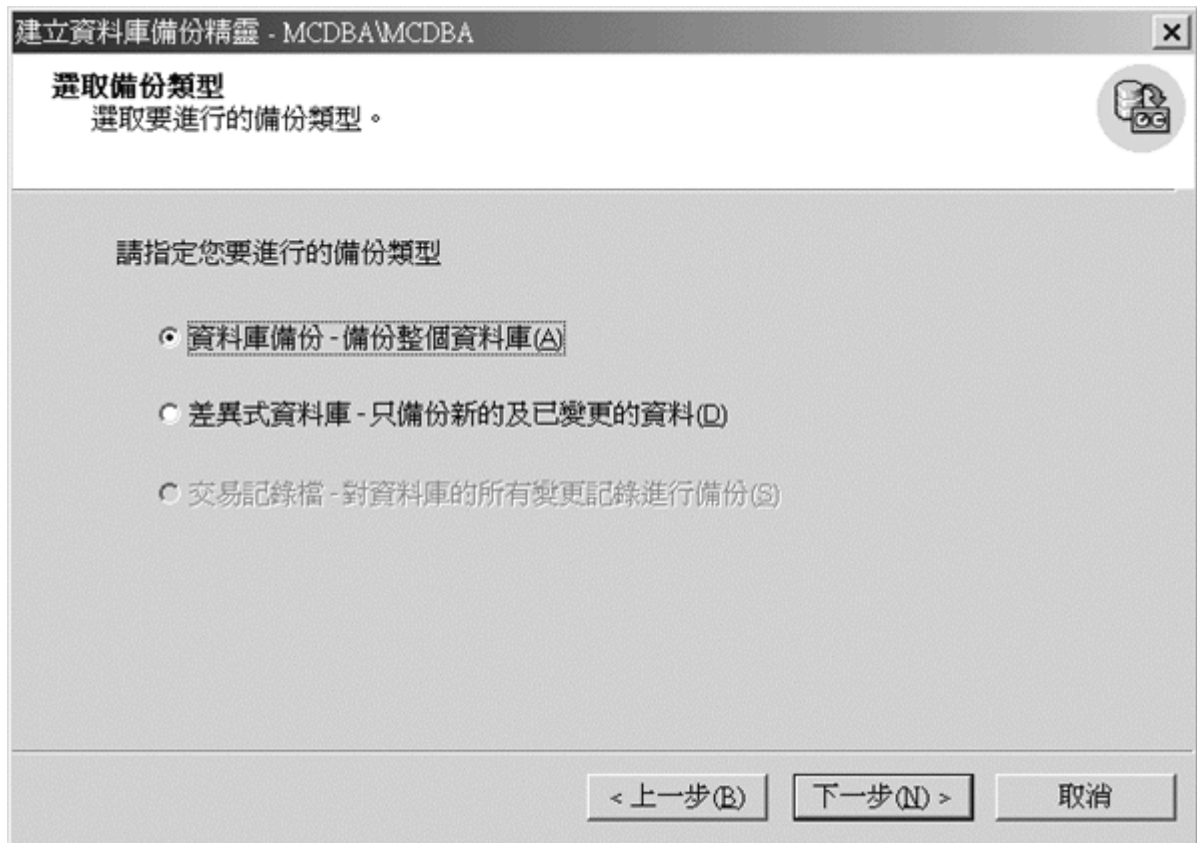
名稱(A):
Aromatherapy 備份

描述(S):
Lesson 2 backup

< 上一步(B) 下一步(N) > 取消

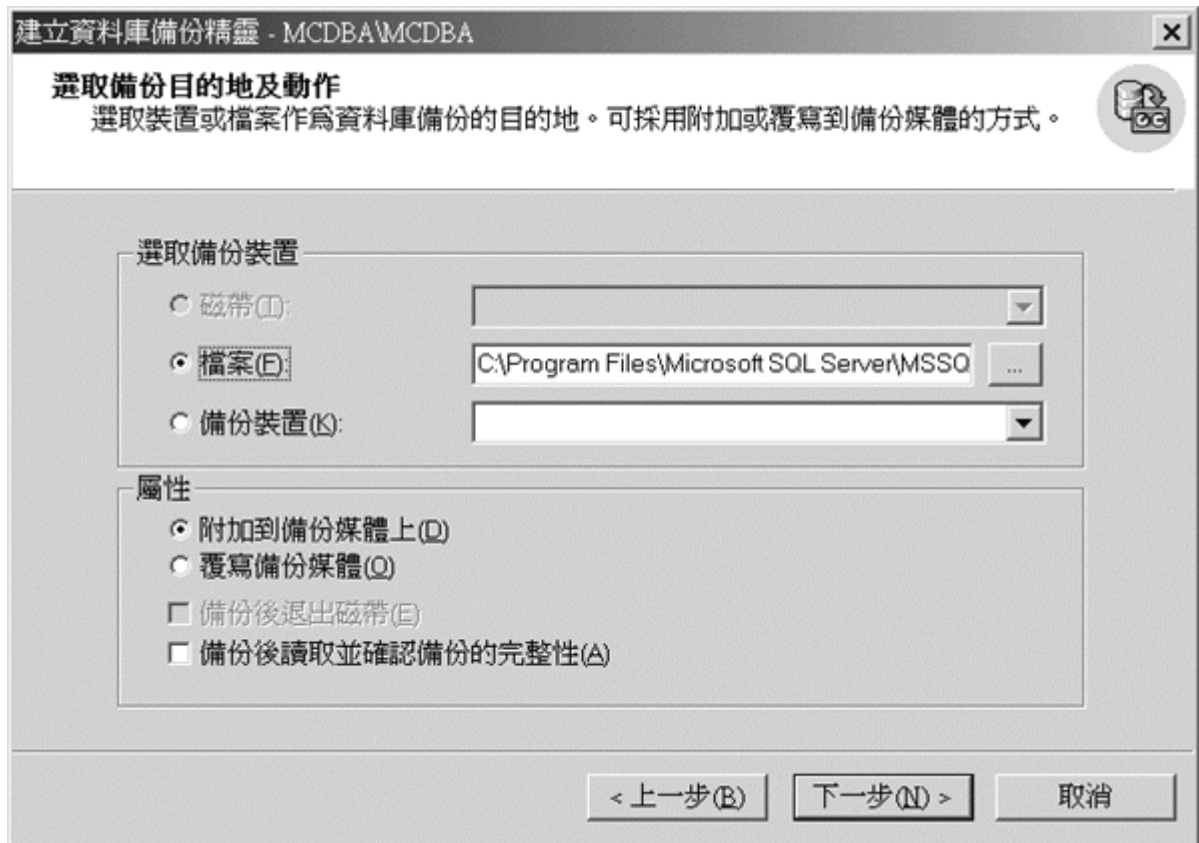
7. 按一下 [下一步](#) 按钮。

此时备份精灵会显示一个询问您想要执行的备份型态的画面：[完整数据库备份](#)、[差异数据库备份](#) 或 [交易记录文件备份](#)。



8. 在此次练习当中，我们选择使用 [数据库备份-备份整个数据库](#) 来进行数据库的备份。选择完毕之后，请按一下 [下一步](#) 按钮。

此时备份精灵会询问您要选择哪一个档案或装置来作为备份文件的备份位置。



9. 按一下 [浏览](#) 按钮。



浏览按钮

此时备份精灵会显示一个 [备份装置位置](#) 对话框。



10. 找到您安装本章练习档案的数据夹。

重要

一般来说，当您要储存备份的数据时，您必须要确定将备份的数据与在线数据库分开储存于不同的磁盘装置，假如您将备份数据与在线数据库储存在同一个档案

装置，则当此储存位置发生毁损时，仍然无法还原数据库，那是因为您将备份数据库与在线数据库储存在同一个位置所造成的。

-
11. 按一下 **确定** 按钮以回到备份精灵窗口中。在 **属性** 区域中，您可以选取附加或覆写备份媒体。在本练习中，我们选择 **附加到备份媒体上** 项目。

12. 按一下 **下一步** 按钮。

此时备份精灵会显示一个询问您选择备份排程确认的画面。

建立資料庫備份精靈 - MCDBA\MCDBA

備份確認與排程
檢查媒體標籤及備份的到期日與時間，可以協助預防意外的覆寫。您也可以設定排程來定期執行備份。

進行檢查媒體集

☒ 檢查媒體集名稱及備份集到期日期(H)

媒體集名稱(M):

備份集到期

☐ 備份集逾期時間(K):

☒ 之後(A): 日

☐ 於(Q):

進行排程

☐ 排程(S):

< 上一步(B) 下一步(N) > 取消

13. 在本练习中，我们不需要 SQL Server 确认这个备份，因此我们将 [检查媒体集名称及备份集到期日期](#) 项目取消选取。

建立資料庫備份精靈 - MCDBA\MCDBA

備份確認與排程

檢查媒體標籤及備份的到期日與時間，可以協助預防意外的覆寫。您也可以設定排程來定期執行備份。

進行檢查媒體集

☐ 檢查媒體集名稱及備份集到期日期(H)

媒體集名稱(M):

備份集到期

☐ 備份集逾期時間(S):

☒ 之後(A): 日

☐ 於(Q):

進行排程

☐ 排程(S):

變更(C)...

< 上一步(B) 下一步(N) > 取消

14. 按一下 [下一步](#) 按钮。

此时备份精灵会询问您是否已经确定您所选择的项目。

提示

SQL Server 会在 [媒体集](#)（media set）中组织备份媒体。而媒体集可以是独立的磁盘档案或者是备份磁带的集合。



15. 按一下 **完成** 按钮。当备份精灵开始执行备份动作时，此时建立数据库备份精灵会显示一个备份进度的对话框，并且当备份工作处理完毕之后，会在此对话框内显示 **备份已经成功** 的讯息。



提示

您也可以藉由在 Enterprise Manager 的 [工具](#) 菜单中选取 [备份数据库](#) 项目来进行数据库备份。如果您是选择这个选项来进行数据库的备份动作，它只会显示一个单独的对话框，除此之外其所有的操作方式都是一样的。

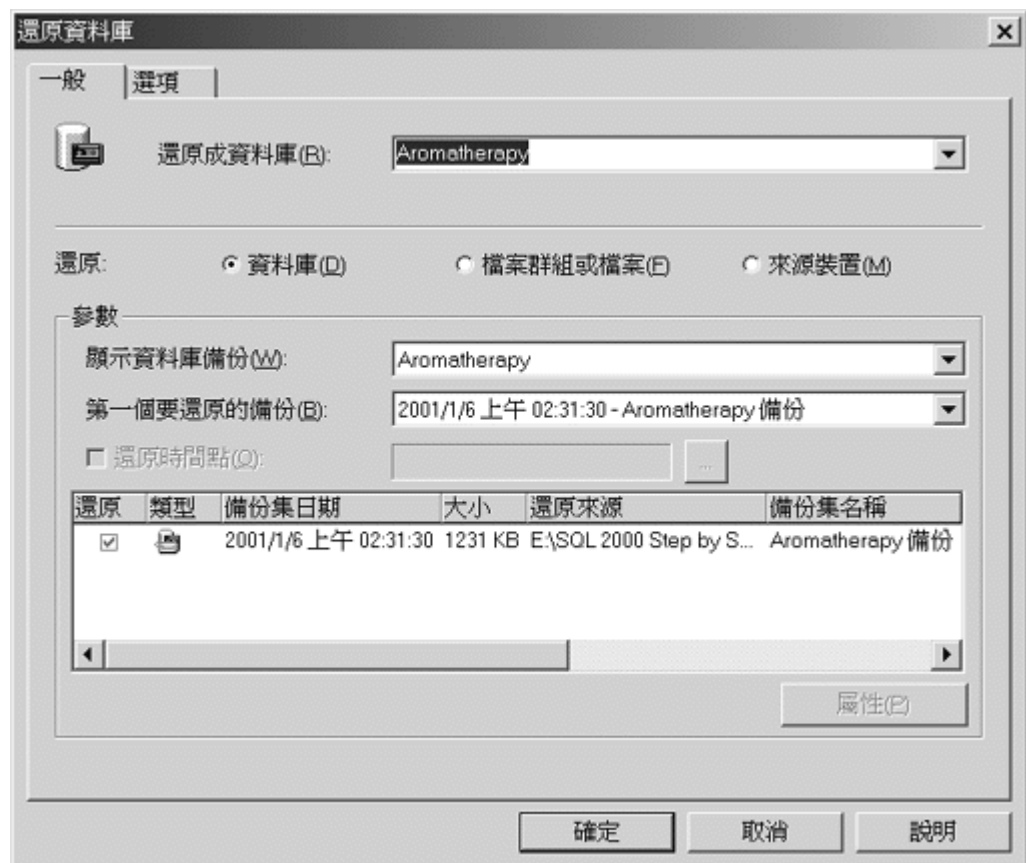
将数据库进行还原

将数据库进行还原，并不像将数据库备份一样，您无法随时随地将数据库进行还原，因为要将数据库进行还原必须是数据库发生异常现象时，但 SQL Server 所提供的 Enterprise Manager 可以让您很容易地将已经备份的数据库还原。

还原数据库

1. 在控制台树状目录中，在 **Aromatherapy** 数据库名称上按一下鼠标右键，并指向 **所有工作** 项目，然后再选择 **还原数据库** 项目。

此时 SQL Server 会显示还原数据库的对话框。



2. 按一下 **确定** 按钮。

当系统正在执行数据库还原动作时，**SQL Server** 会显示一个还原进度的对话框，并且当还原工作处理完毕之后，会在此对话框内显示已经还原成功的讯息。



使用数据库维护计划精灵

在前面的章节中，您已经学习到如何备份以及还原数据库，数据库的备份是一项重要而且必须定期执行的工作，但是一般系统管理人员会很容易忘记将数据库进行备份，等到数据库发生错误时，才为没有定期将数据库备份而懊恼，正所谓「千金难买数据库」，当然我们非常不希望有这种情形发生，但应该如何才不会忘记将数据库备份呢？**SQL Server 2000** 提供一种既好用又可以设定排程的工作来进行数据库的备份，那就是使用 [数据库维护计划精灵](#)。

[数据库维护计划精灵](#) 可以让您排定何时执行数据库备份的工作。每当您所设定的时间一到，它会自动地执行备份数据库的动作，并且更新数据库的统计信息。

SQL Server 会维护关于数据表中的统计信息，而此种信息可以提供 **SQL Server** 知道该用哪种有效率的方式将数据还原。当 **SQL Server** 开始进行还原数据库的动作时，它会将已更改过的数

据放弃不用，而将新的资料新增进来。SQL Server 会周期性地将数据的统计信息自动地更新，

您可以在某些特定情况之下使用数据库维护计划精灵将数据的统计信息更新。

另外，您也可以使用 [数据库维护计划精灵](#) 来安排 SQL Server 执行定期性的检查工作。您设定

检查的工作可以是比较低阶的使用者及系统数据表格的完整检查。最后，您可以使用 [数据库维](#)

[护计划精灵](#) 来设定 [记录传送](#)（Log Shipping）的工作。所谓的 [记录传送](#) 是指自动将交易记录

文件定时的传送到另一台服务器上，使两台实体上分离的数据库可以达到同步化的结果。

译注

记录传送的功能可以让您以轻松方式自动化的建立一部 [待命服务器](#)（Standby Server），当主

要的实际生产服务器发生时，待命服务器可以马上就接替其工作，让您的系统随时处于正常

运作的状态。

单一的数据库维护计划是由许多的工作组合而成。另外，您也可以得到数据库维护计划精灵所产

生的工作报告，而这份报告您可以把它当成一个文本文件或 HTML 文件储存在您所指定的档案

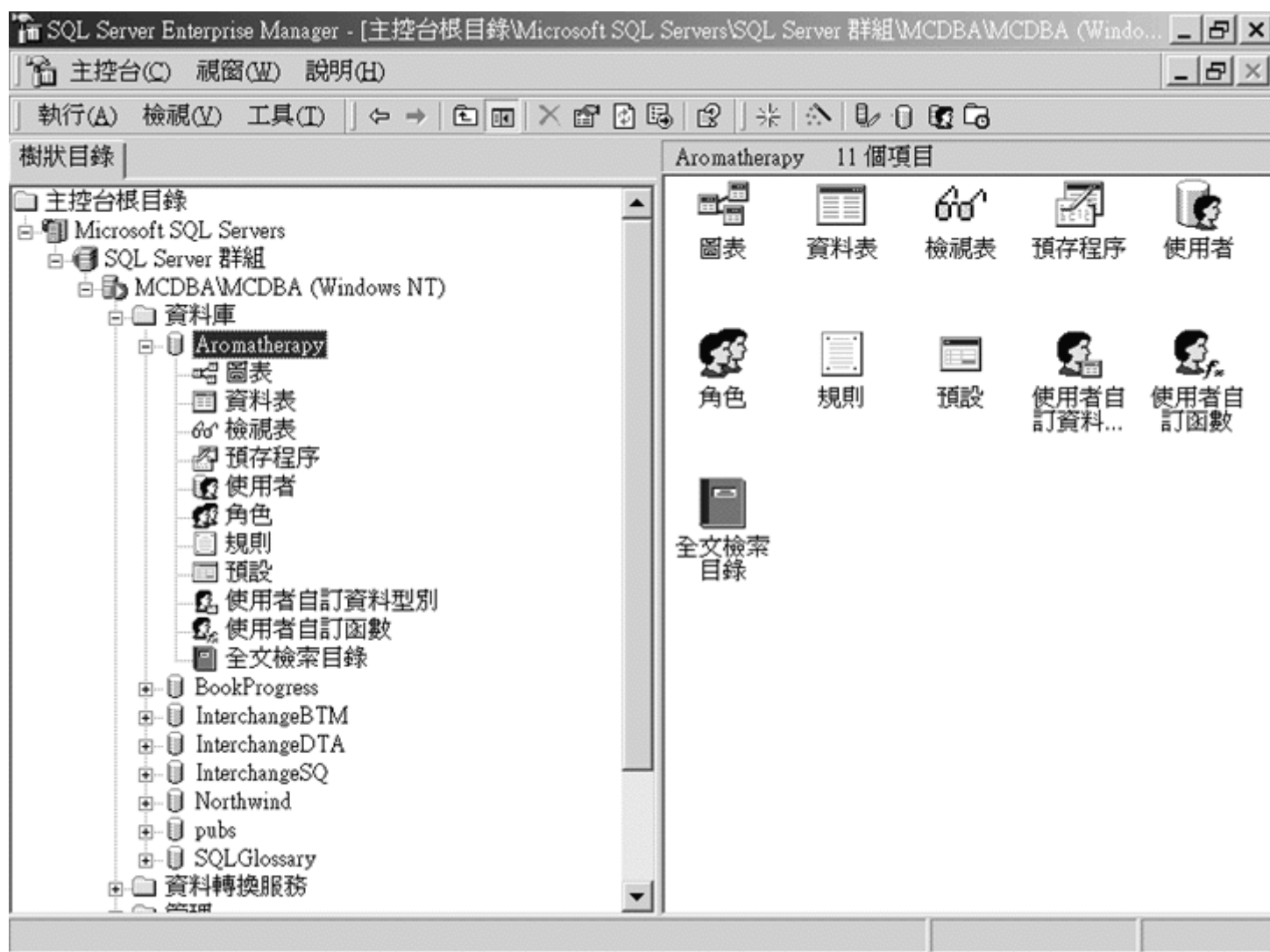
位置内，或者您也可以将它以电子邮件方式传送给其它需要这份报告的相关操作人员。在下面练

习中，我们将简单设定一个定期性的备份工作。

建立每个月备份数据库的维护计划

1. 在主控台的树状目录之中，选取 **Aromatherapy** 数据库。

SQL Server 会在 [详细资料窗格](#) 中显示其数据库的对象。

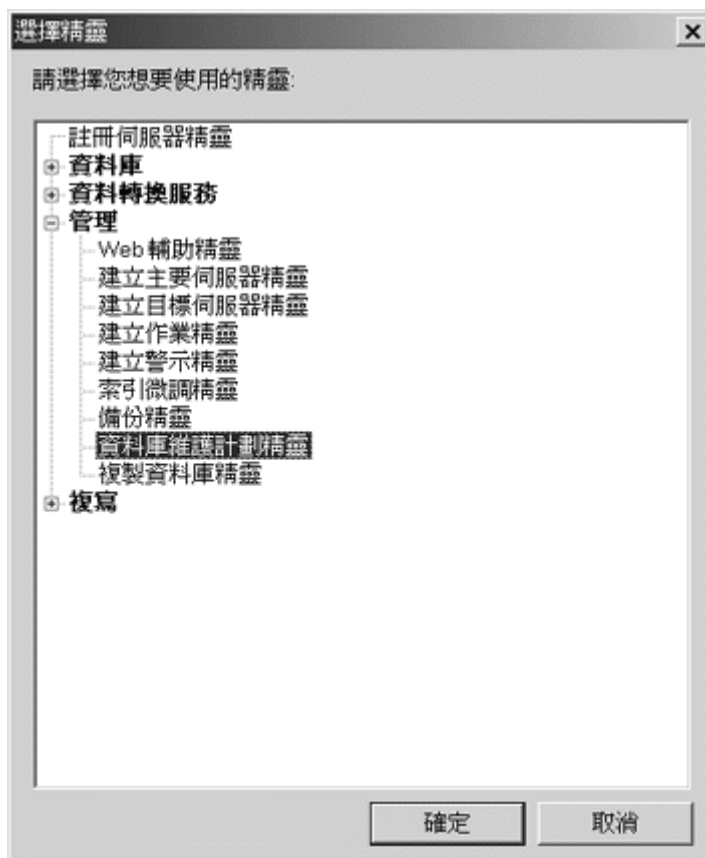


2. 在 Enterprise Manager 的工具列中按一下 [執行精靈](#) 按钮。

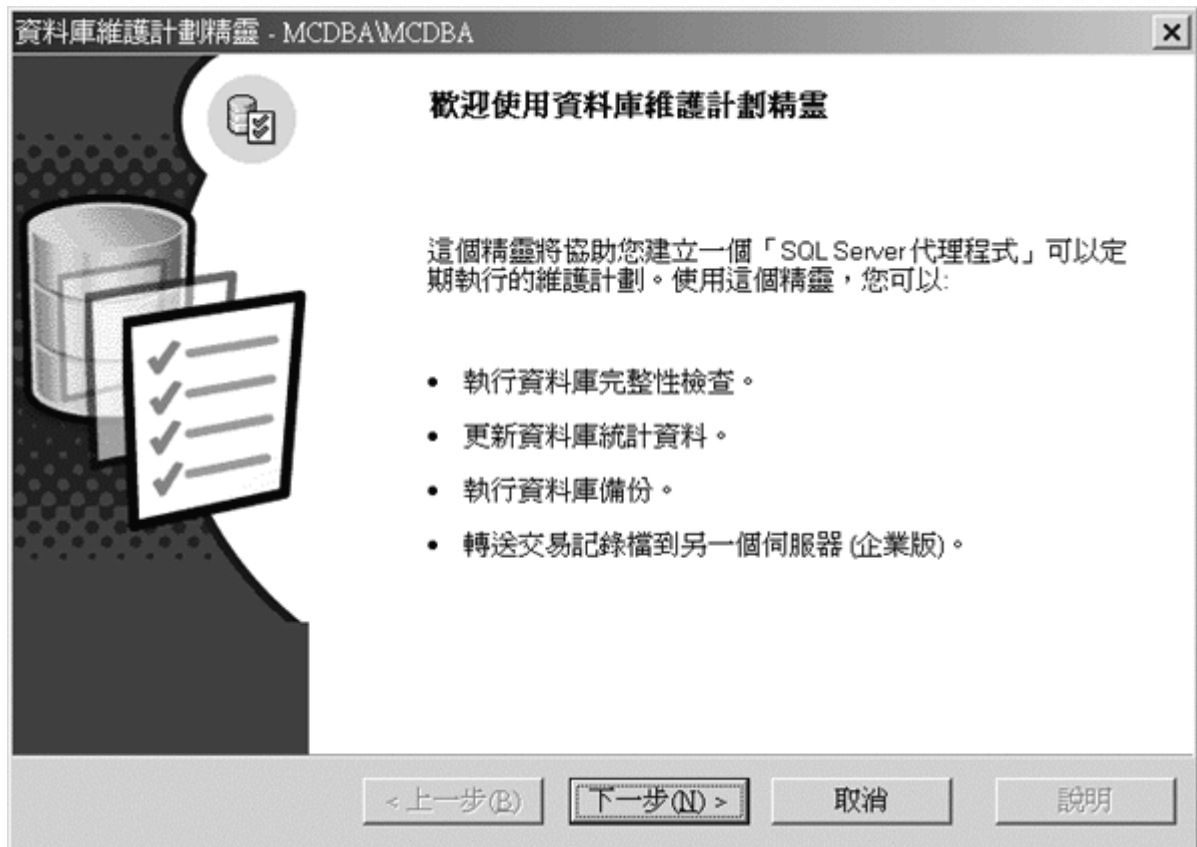


执行精灵按钮

此时 SQL Server 会显示 [选择精灵](#) 的对话框。



3. 在 [管理](#) 项目中选取 [数据库维护计划精灵](#)，此时 SQL Server 会显示 [数据库维护计划精灵](#)。



4. 按一下 **下一步** 按钮。

此时精灵会询问您要选择哪一个数据库来作为建立数据维护计划的数据库。当一开始启动此精灵时，我们就已经在主控台树状目录之下选取 **Aromatherapy** 数据库，而此数据库会成为此精灵要进行维护的预设数据库。如果您临时反悔不要使用 **Aromatherapy** 数据库为要维护的数据库时，您可以在此对话框中选取您要进入维护的数据库名称。



5. 按一下 [下一步](#) 按钮。

此时精灵会显示一个询问您要以哪一种最佳化方式来进更新数据的画面。

資料庫維護計劃精靈 - MCDBA\MCDBA

更新資料最佳化資訊
當資料及索引頁被填滿時，更新需要更多的時間。請重新組織您的資料及索引頁以改善效能。

☐ **重新組織資料及索引頁(R)**
☐ 使用原有的可用空間來重組頁面(E)
☒ 改變每頁可用空間百分比為(B): 10

☐ **更新查詢最佳化所使用的統計資料。取樣百分比為資料庫的(Q):** 10 %

☐ **從資料庫檔案中移除尚未使用的空間(M)**
當資料庫成長超過(W): 50 MB
壓縮後保持的可用空間量。空間百分比為資料空間的(A): 10 %

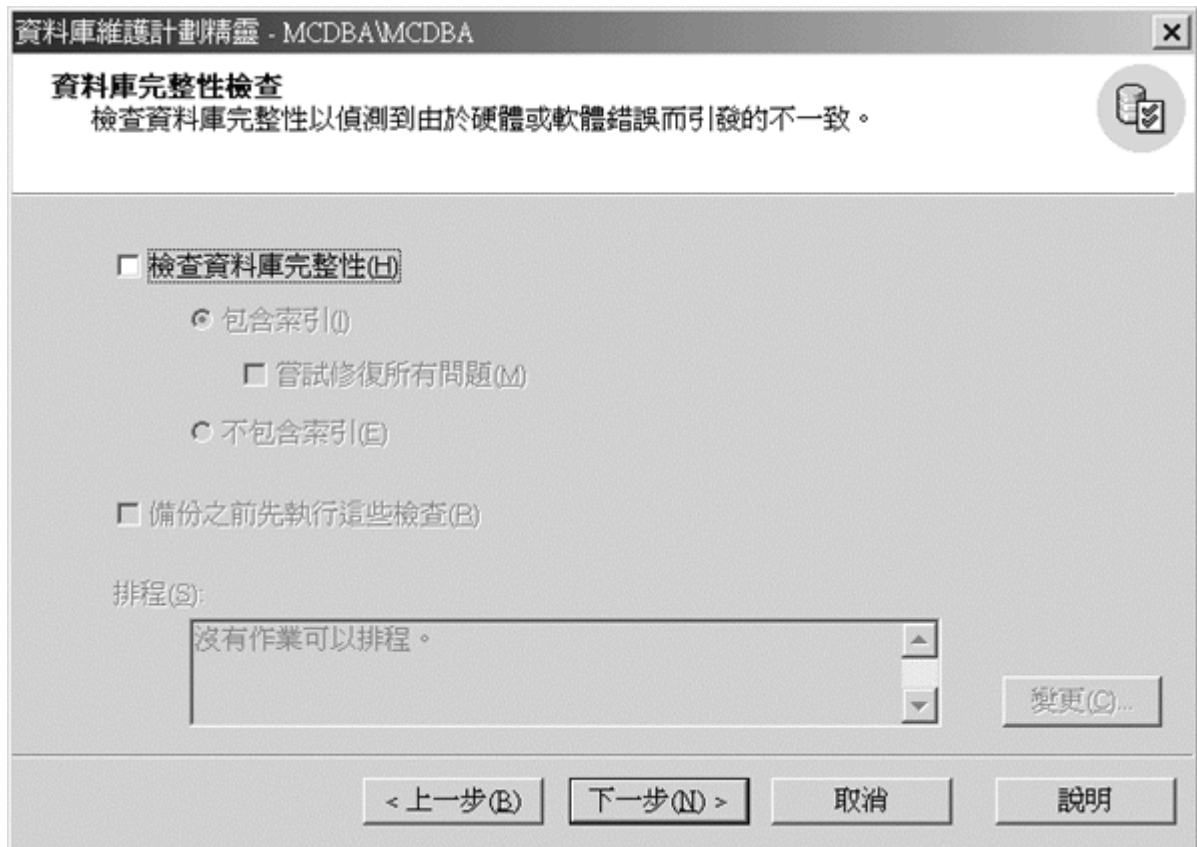
排程(S): 沒有作業可以排程。

變更(C)...

< 上一步(B) 下一步(N) > 取消 說明

6. 按一下 **下一步** 按钮。

此时精灵会显示一个询问您是否要执行完整性的检查工作的画面。



7. 按一下 [下一步](#) 按钮。

此时精灵会显示一个询问您要指定数据库备份的计划数据的画面。在本练习中，

我们将使用所有的默认值来作为我们的选项值，只有更改备份排程而已。

資料庫維護計劃精靈 - MCDBA\MCDBA

指定資料庫備份計劃
指定資料庫備份計劃來防止因系統錯誤而造成的資料遺失。

☒ 將資料庫備份視為維護計劃的一部份(A)

☒ 備份完成後確認備份的完整性(V)

儲存備份檔案的位置:

☐ 磁帶(P):

☒ 磁碟(K)

排程(S):

每 1 週的 星期日 發生於 上午 02:00:00。

變更(C)...

< 上一步(B) 下一步(N) > 取消 說明

8. 按一下 [變更](#) 按钮，以便更改备份的排程数据。

此时此精灵会显示一个 [编辑重复执行作业排程](#) 的对话框。

編輯重複執行作業排程 [X]

作業名稱: (新作業) ☒ 啟動排程(B)

發生頻率

☐ 每天(D)
☒ 每週(W)
☐ 每月(M)

每週

每隔(V) 週的:

<input type="checkbox"/> 星期一	<input type="checkbox"/> 星期二	<input type="checkbox"/> 星期三	<input type="checkbox"/> 星期四
<input type="checkbox"/> 星期五	<input type="checkbox"/> 星期六	<input checked="" type="checkbox"/> 星期日	

每日頻率

☒ 執行一次於(U):
☐ 重複執行於每(B):

開始時間(O):
 結束時間(N):

持續時間

開始日期(S):
☐ 結束日期(E):
☒ 沒有結束日期(A)

9. 在 [發生頻率](#) 區域中選擇 [每月](#)。此精靈會依據您所選擇的項目來更改其對話框的

選項內容。

10. 選擇每一個月的第一個星期日来作為備份的排程。

編輯重複執行作業排程

作業名稱: (新作業) ☒ 啓動排程(B)

發生頻率

☐ 每天(D)
☐ 每週(W)
☒ 每月(M)

每月

☐ 於(Y) 1 日 (每 1 個月)
☒ 於(D) 第一 星期日 (每 1 個月)

每日頻率

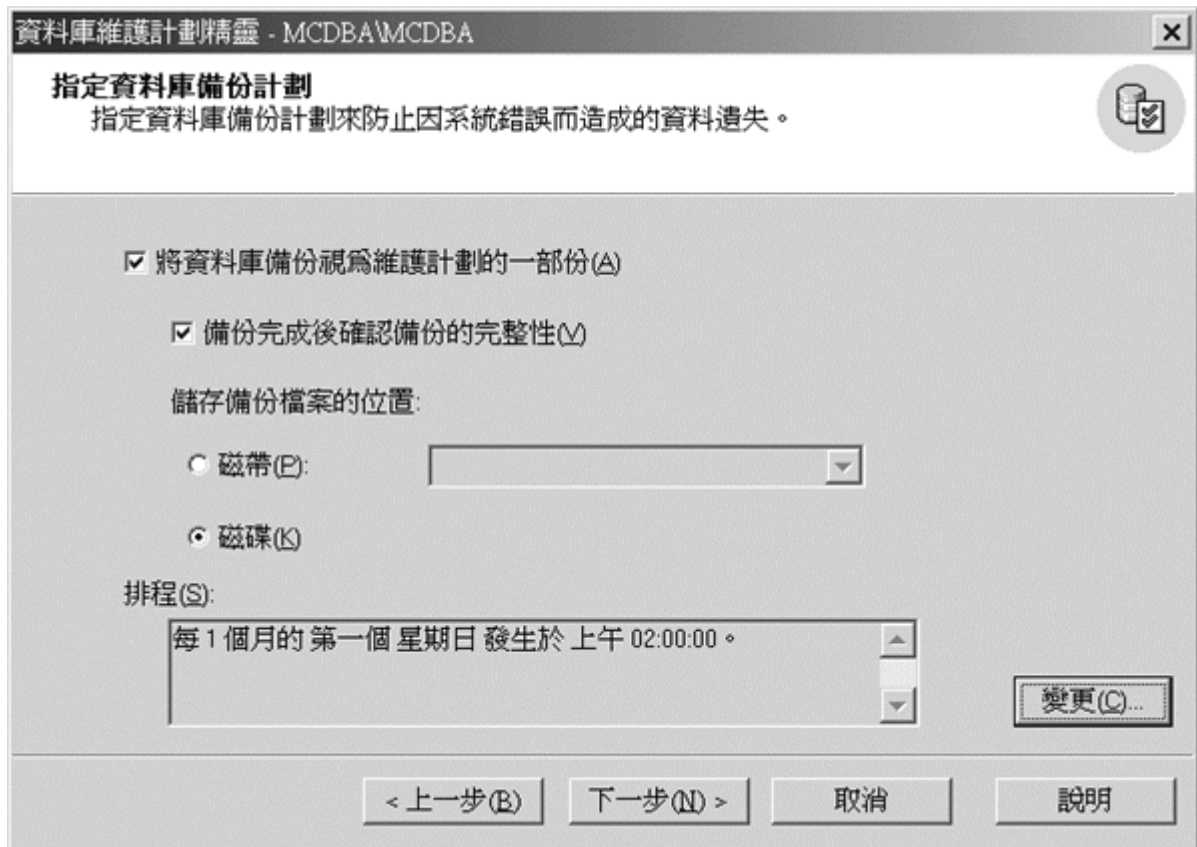
☒ 執行一次於(U): 上午 02:00:00
☐ 重複執行於每(B): 1 小時 開始時間(I): 上午 02:00:00
結束時間(N): 下午 11:59:59

持續時間

開始日期(S): 2001/ 1/ 6 ☐ 結束日期(E): 2001/ 1/ 6
☒ 沒有結束日期(A)

確定 取消 說明

11. 按一下 [確定](#) 按钮，以便回到 [数据库维护计划精灵](#) 窗口中。

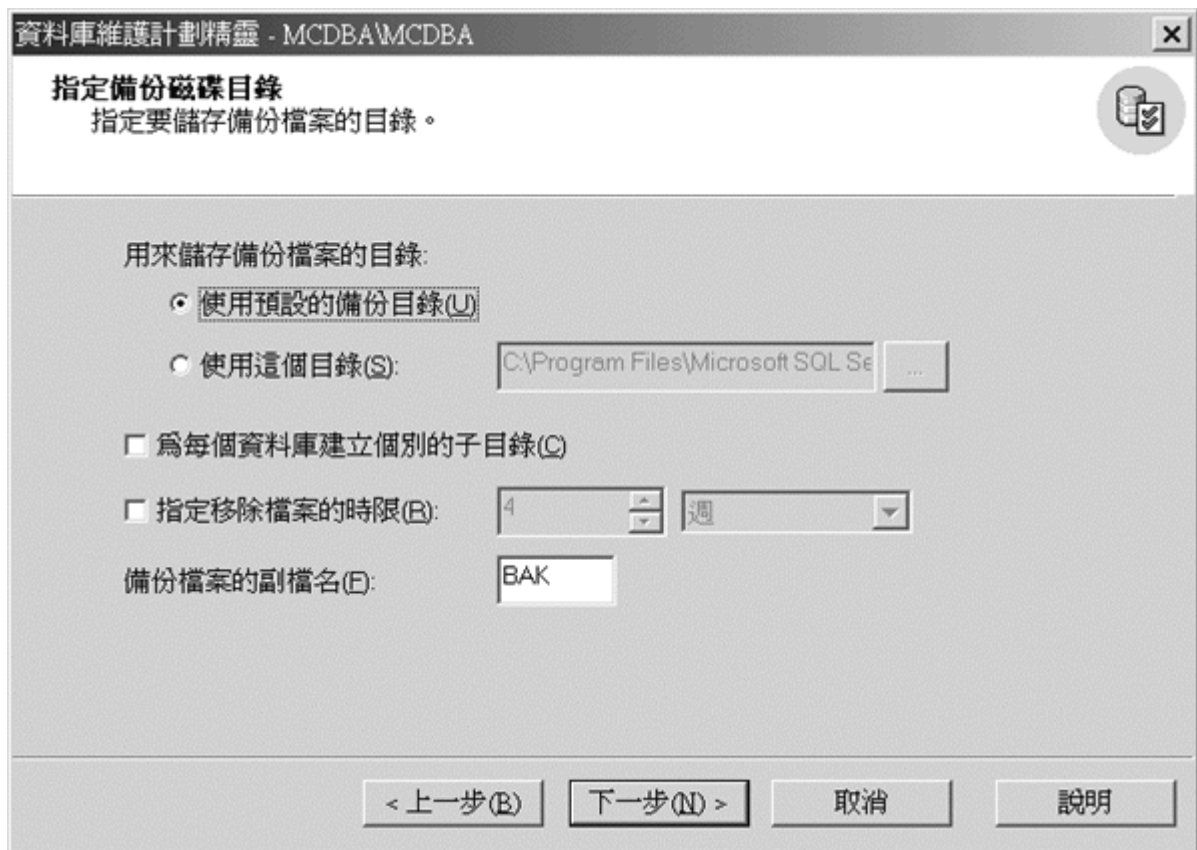


12. 按一下 [下一步](#) 按钮。

此时精灵会显示一个备份磁盘目录的画面。

提示

假如您是使用 [数据库维护计划精灵](#) 来规划其它维护的工作，并且没有选择相关备份项目时，此画面是不会显示的。

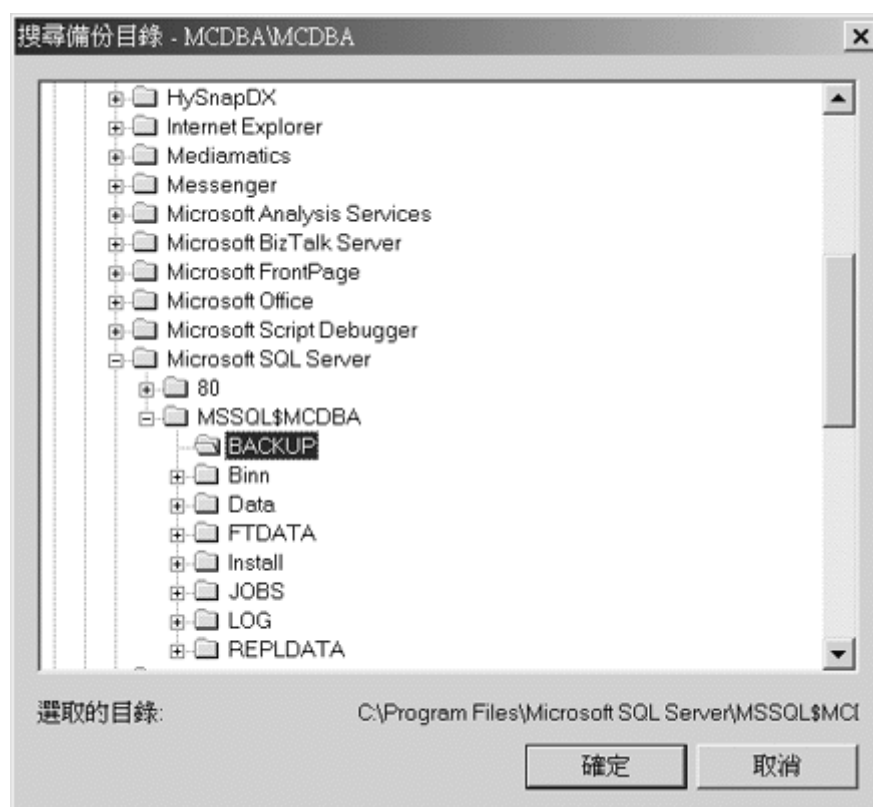


13. 选取 [使用这个目录](#) 项目，然后按一下 [浏览](#) 按钮。



浏览按钮

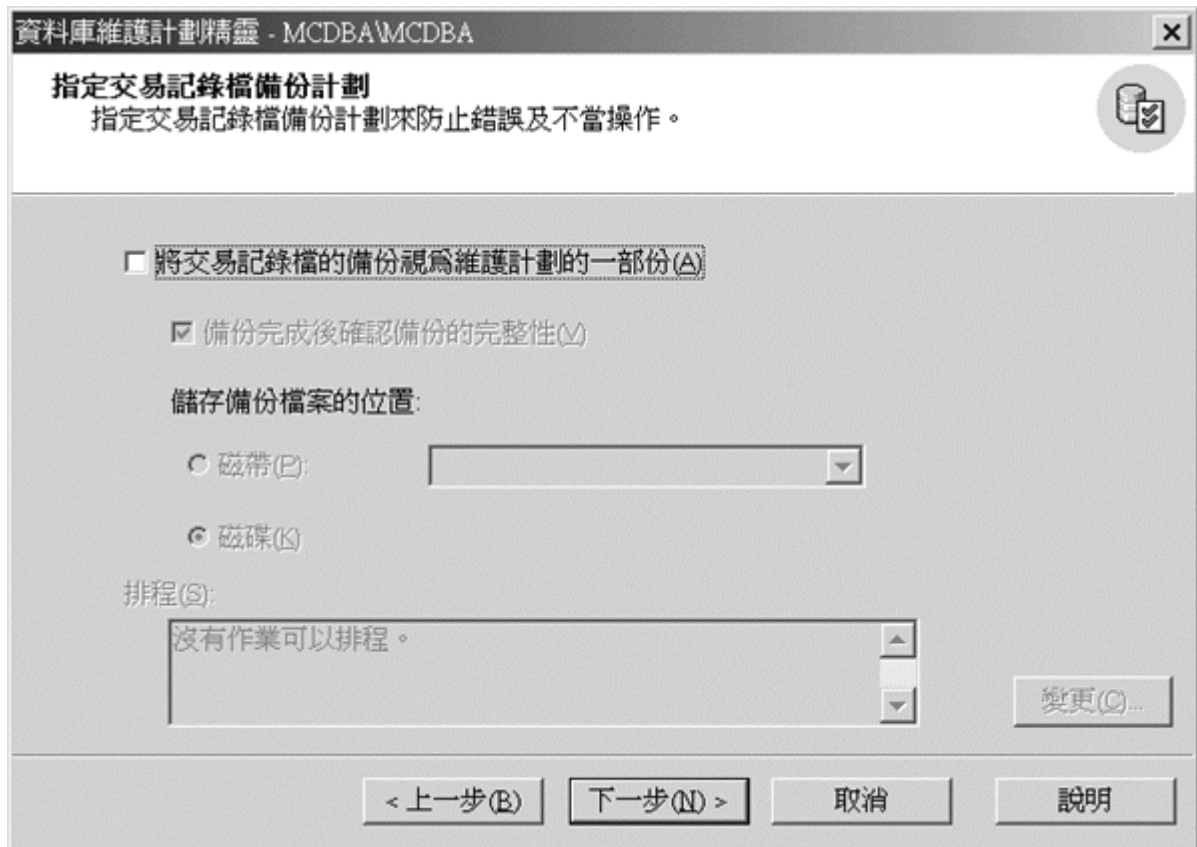
此时精灵会显示一个 **搜寻备份目录** 的对话框。



14. 选取您要备份文件的储存位置，然后再按一下 **确定** 按钮。

15. 按一下 **下一步** 按钮。

此时精灵会显示一个询问您在此备份计划中，要将交易记录文件备份在哪个位置的画面。



提示

假如您只是要执行交易记录文件备份（而不是使用完整备份）时，您将会使用到这个画面，而不需要使用到另一个包括数据库维护计划的画面。

16. 按一下 **下一步** 按钮。

此时精灵会询问您当计划执行完毕之后是否要产生一份报告。

資料庫維護計劃精靈 - MCDBA\MCDBA

產生報告
指定要儲存維護計劃所產生的報告存放目錄。

☐ 將報告寫入文字檔，目錄在(R): C:\Program Files\Microsoft SQL Servi ...

☐ 指定刪除文字報告檔的時限(D): 4 週

☐ 傳送電子郵件報告給操作員(S): Jentai Chen ...

新增操作員(W)...

< 上一步(B) 下一步(N) > 取消 說明

17. 按一下 **下一步** 按钮。

此时精灵会询问您是否要将维护历程记录储存在本机服务器。在本练习中，我们

采用预设的设定值。

資料庫維護計劃精靈 - MCDBA\MCDBA

維護計劃歷程記錄
指定您要儲存維護計劃資料錄的方式。

本機伺服器

☒ 將歷程記錄寫入伺服器上的 msdb.dbo.sysdbmaintplan_history 資料表(H)

☒ 限制此計劃中資料表的資料列數為(M): 1000 個資料列

遠端伺服器

將歷程記錄寫入遠端伺服器上的 msdb.dbo.sysdbmaintplan_history 資料表中。使用 Windows 的帳戶驗證登入遠端伺服器。

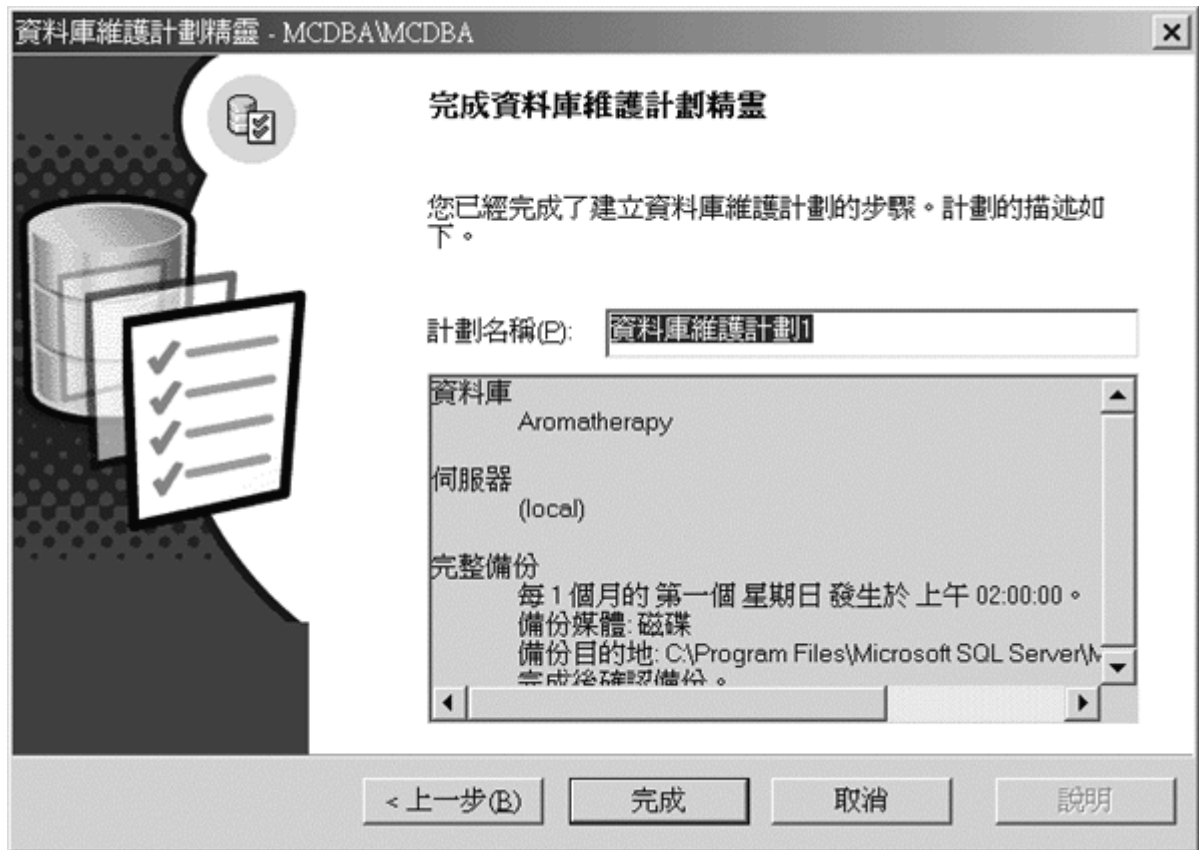
☐ 將歷程記錄寫入伺服器(W):

☒ 限制此計劃中資料表的資料列數為(L): 10000 個資料列

< 上一步(B) 下一步(N) > 取消 說明

18. 按一下 [下一步](#) 按钮。

[数据库维护计划精灵](#) 会显示您所设定的项目。



19. 按一下 **取消** 按钮。

重要

一般来说，我们都是按一下 **完成** 按钮以便执行维护计划，但是在练习阶段我们不需要执行这个动作。

提示

维护计划可以在主控台的树状目录之中加以编辑，您可以在 [详细数据窗格](#) 内按下维护计划的图示，以便进行浏览以及编辑维护计划。

本章总结

要执行的工作	执行的步骤	按钮
备份数据库	在主控台的树状目录下选取数据库，并且在 Enterprise Manager 工具列中按一下 执行精灵 按钮，以便显示 选择精灵 对话框。在 管理 区域中选择 备份精灵 ，并且按照此精灵的提示来执行。	
还原数据库	在主控台树状目录中的数据库名称上按一下鼠标右钮，指向 所有工作 ，然后再选择 还原数据库 。	
建立一个维护计划	在主控台的树状目录下选取数据库，并且在 Enterprise Manager 工具列中按一下 执行精灵 按钮，以便显示 选择精灵 对话框。在管理区域中选择 数据库维护计划精灵 ，并且按照此精灵的提示来执行。	

3. SQL Server 2000 的安全性

在本章中，您将学习到：

- 建立 Microsoft SQL Server 登入账户。
- 建立 Microsoft Windows 登入账户。
- 移除登入账户。
- 建立数据库的使用者。
- 移除数据库的使用者。
- 建立数据库的角色。
- 将使用者指派至数据库角色中。
- 自数据库角色中移除使用者。

- 移除数据库角色。

维护数据库最重要的部份在于建置数据的安全性。您必须要确认每一个人都是可以存取每一个人应该存取的数据。在本课程中，我们将说明 SQL Server 2000 是如何管理安全性的问题，以及学习如何建立以及如何指派安全性的基本权限。

认识安全性模式

当 SQL Server 的执行个体(instance)接受一个进行连接的要求时，它会验证登入 ID(Login ID)。

登入 ID 是一个账号识别项，它可以用来控制使用 SQL Server 2000 的权利。SQL Server 会检查此登入 ID 是否合法，然后决定此登入 ID 是否已经授权可以执行他所要求的事情。这个处理的过程就是大家都知道的 **验证**（authentication）。

SQL Server 2000 支持的验证方式有二种，您可以选择其中一种方式来进行验证：它可以藉由 Microsoft Windows NT 或 Windows 2000 的安全性来确认此登入 ID 是否正确，或者它自己本身来执行验证工作。

Windows 验证

当您使用 Windows 验证（前一个 SQL Server 版本称为 **整合式安全性**）时，系统管理员会授权安全性的权限，以便授权给可以存取 Windows NT 以及 Windows 2000 的账号，并且要求 Windows 的客户端软件以 **信任联机**（Trusted Connection）的方式来与服务器进行联机。假如

Windows NT 或 Windows 2000 已经验证某位使用者后，会即授权一个信任联机，SQL Server 2000 只需要确认该登入 ID 能够存取哪个数据库及哪些特定的对象即可。

重要

由于 Windows 98 并没有支持信任联机的功能。因此在 Windows 98 上执行的 SQL Server 执行个体并不能使用 **Windows 的验证模式**。

SQL Server 验证

除了使用 Windows 验证模式，SQL Server 2000 也可以自己执行安全性验证，称为 SQL Server 验证（前一个 SQL Server 版本称为「标准安全验证模式」）。当一个联机是要求使用 SQL Server 验证时，SQL Server 2000 会接受一个登入 ID 和密码这二个数据以确认此登入账户是否合法。

重要

Microsoft 推荐您使用 Windows 验证的方式来进行验证。

使用者登入账户

SQL Server 2000 内的安全性是由数个安全性对象来管理。其中最高阶层的是登入账号，它是用来识别您是一个 SQL Server 2000 的使用者、一个 Windows 使用者或者是 Windows 群组来登入到服务器中。

重要

要完整执行下列的范例，您必须是 Security Administrators 或 System Administrators 的角色，假如在 SQL Server 系统内您并没有这些权利时，恐怕您就不能管理您的数据库账户。

建立使用者登入账户

登入账号可以在 [安全性](#) 数据夹内的 [登入](#) 图示上按右钮，在其所出现的快捷菜单中选取 [新增登入](#) 项目来建立，但是最简单的方式是由 [建立登入精灵](#) 来建立。

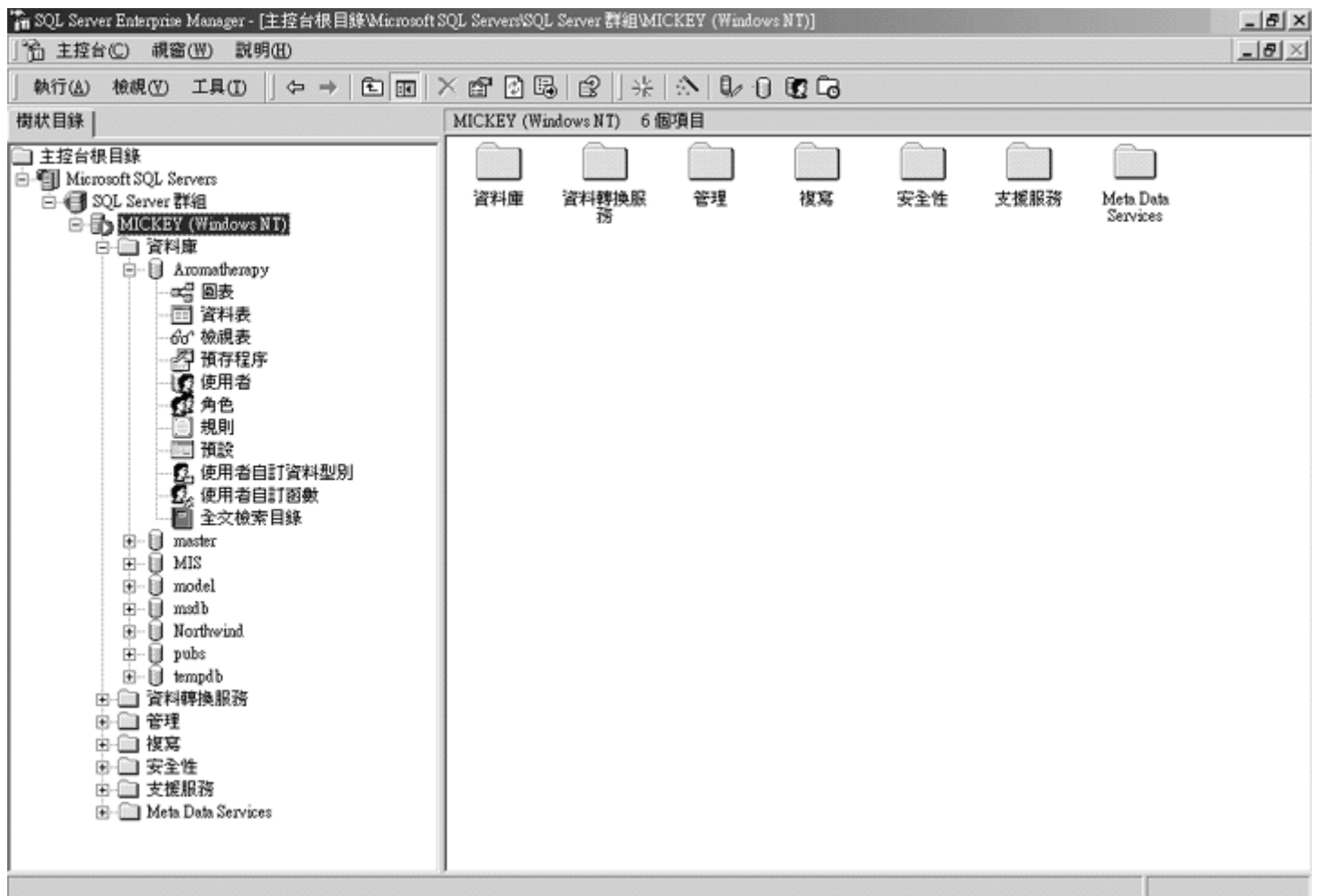
建立 SQL Server 登入账户

1. 在主控制台树状目录之下选取要建立登入账户的服务器。



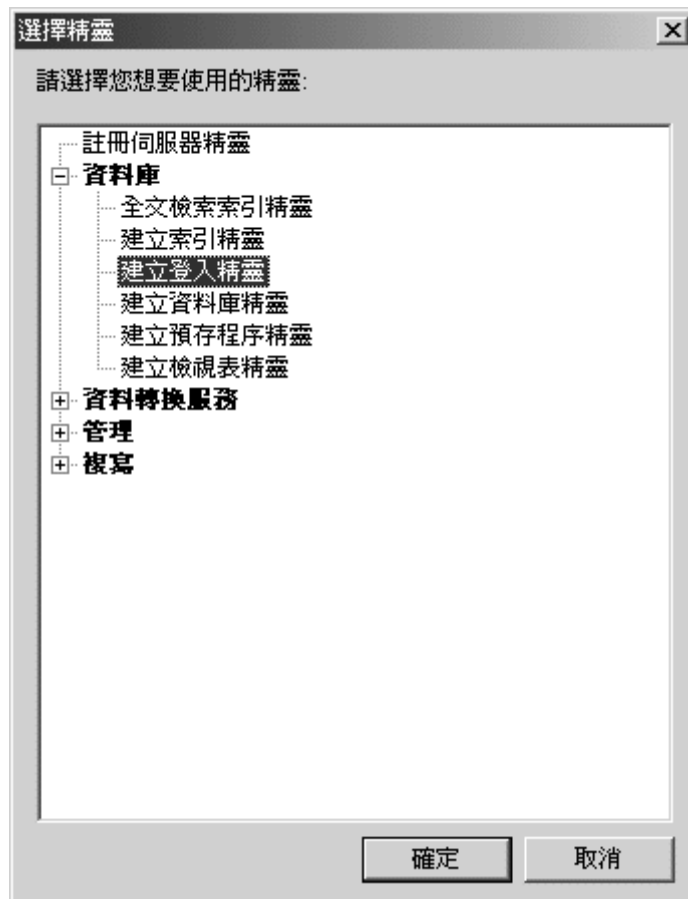
执行精灵按钮

SQL Server 会在详细资料窗格中显示此服务器内的对象详细内容。



2. 在 Enterprise Manager 的工具列上按一下 [執行精靈](#) 按钮。

SQL Server 会显示 [选择精靈](#) 对话框。



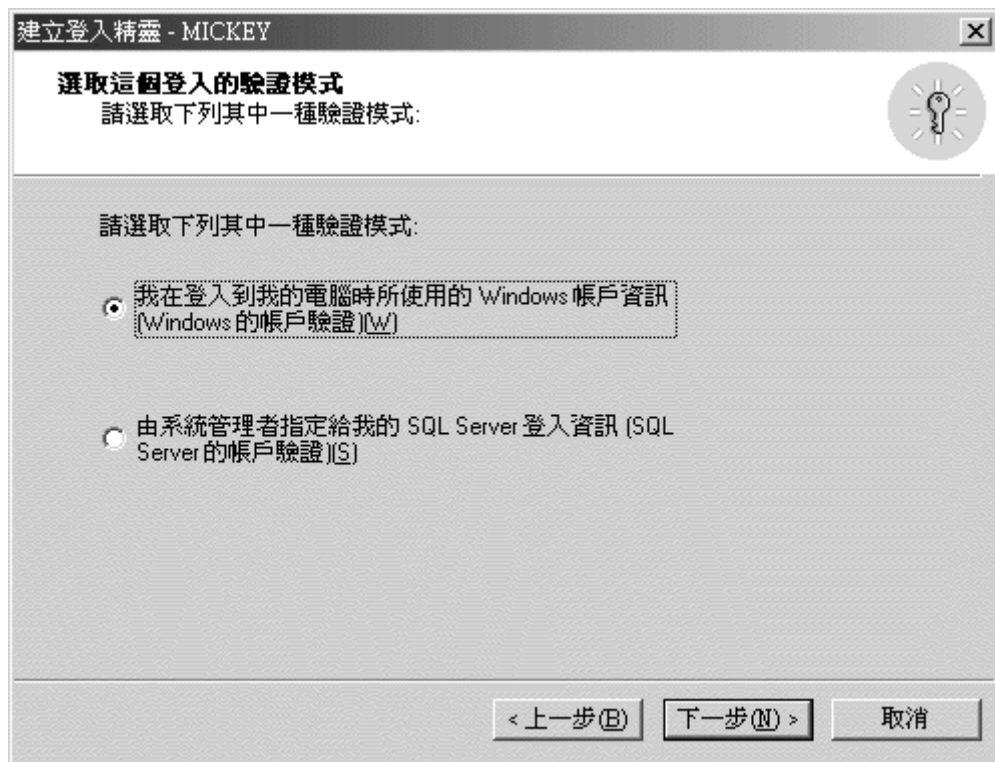
3. 在 **数据库** 数据夹选取 **建立登入精灵** 项目，然后再按一下 **确定** 按钮。

SQL Server 会显示一个 **建立登入精灵** 窗口的第一页画面。

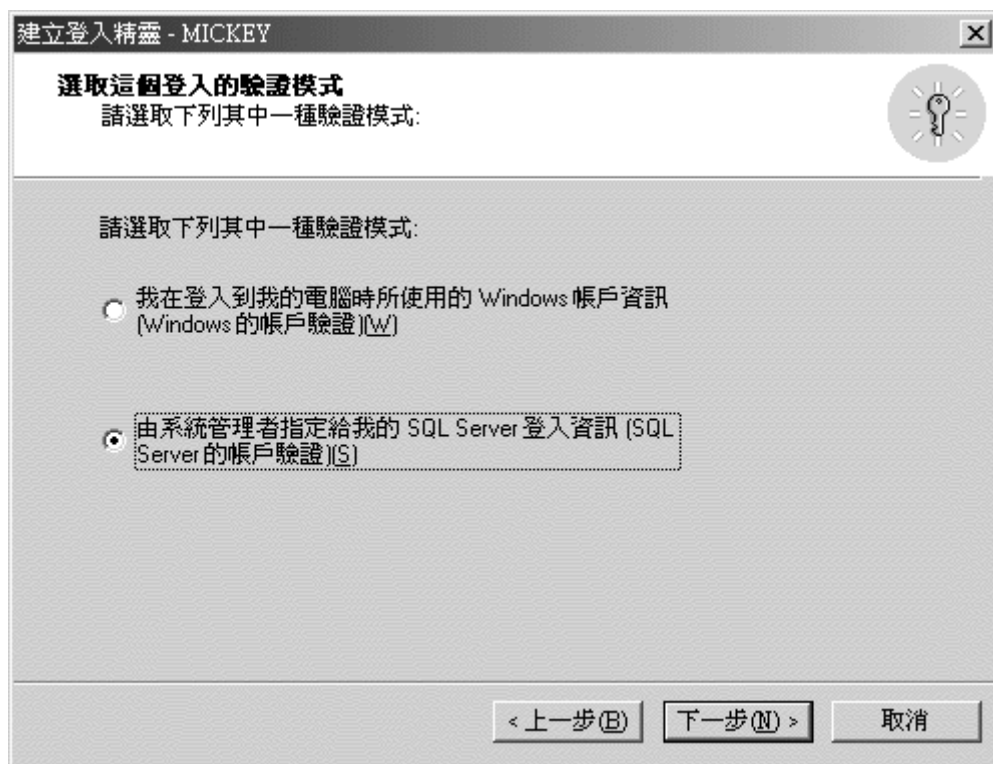


4. 按一下 **下一步** 按钮。

此精灵会询问您要选择哪一种验证模式。



5. 请选取 由系统管理者指定给我的 **SQL Server** 登入信息[**SQL Server** 的账户验证] 项目。



6. 按一下 [下一步](#) 按钮。

此精灵会要求您输入 [登入识别码](#) 与 [密码](#) 。

建立登入精靈 - MICKEY

使用 SQL Server 的帳戶驗證
輸入用來存取 SQL Server 的 SQL Server 登入識別碼與密碼。

登入識別碼(L):

密碼(P):

確認密碼(C):

< 上一步(B) 下一步(N) > 取消

7. 在 [登入识别码](#) 字段中输入『TestID』，而密码则您可以依照需求来输入内容。

建立登入精靈 - MICKEY

使用 SQL Server 的帳戶驗證
輸入用來存取 SQL Server 的 SQL Server 登入識別碼與密碼。

登入識別碼(L):

密碼(P):

確認密碼(C):

< 上一步(B) 下一步(N) > 取消

8. 按一下 **下一步** 按钮。

此精灵会显示一个要求您指定安全性角色的存取画面。



9. 请选取 **System Administrators** 。



10. 按一下 [下一步](#) 按钮。

此精灵会要求您指定数据库的使用权限。



服务器角色

服务器角色是用于指派登入账号可以存取服务器的安全性层级。表格 3-1 描述了每一个角色所指派的权利。

完整的名称	名称	说明
Bulk Insert Administrators	bulkadmin	可执行 BULK INSERT 操作。
Database Creators	dbcreator	可建立、更改与删除数据库。

Disk Administrators	diskadmin	可管理磁盘档案。
Process Administrators	processadmin	可管理 SQL Server 中执行的处理序 (processes) 。
Security Administrators	securityadmin	可管理登入账户与 CREATE DATABASE 权限，也可读取错误记录文件。
Server Administrators	serveradmin	可设定服务器范围的组态选项，关闭服务器。
Setup Administrators	setupadmin	可以管理连结服务器、启动程序和延伸预存程序。
System Administrators	sysadmin	可在 SQL Server 中执行任何活动。

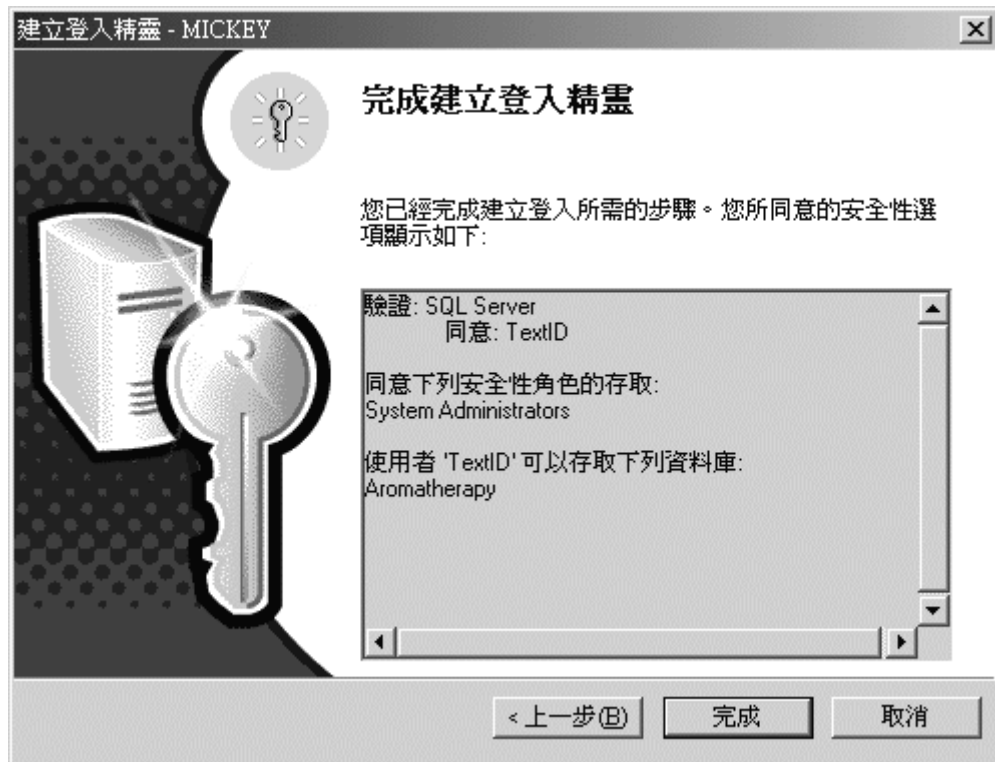
表 3-1

11. 选取 **Aromatherapy** 数据库项目。



12. 按一下 [下一步](#) 按钮。

此精灵会将您所设定的选项显示在此画面上。



重要

属于特定服务器角色的登入账户—尤其是属于 **System Administrators** 角色的登入账户，都将会具备所有数据库的使用权限，无论该登入账号是否已经被授权存取数据库或是没有被授权存取数据库。

-
13. 按一下 **完成** 按钮。

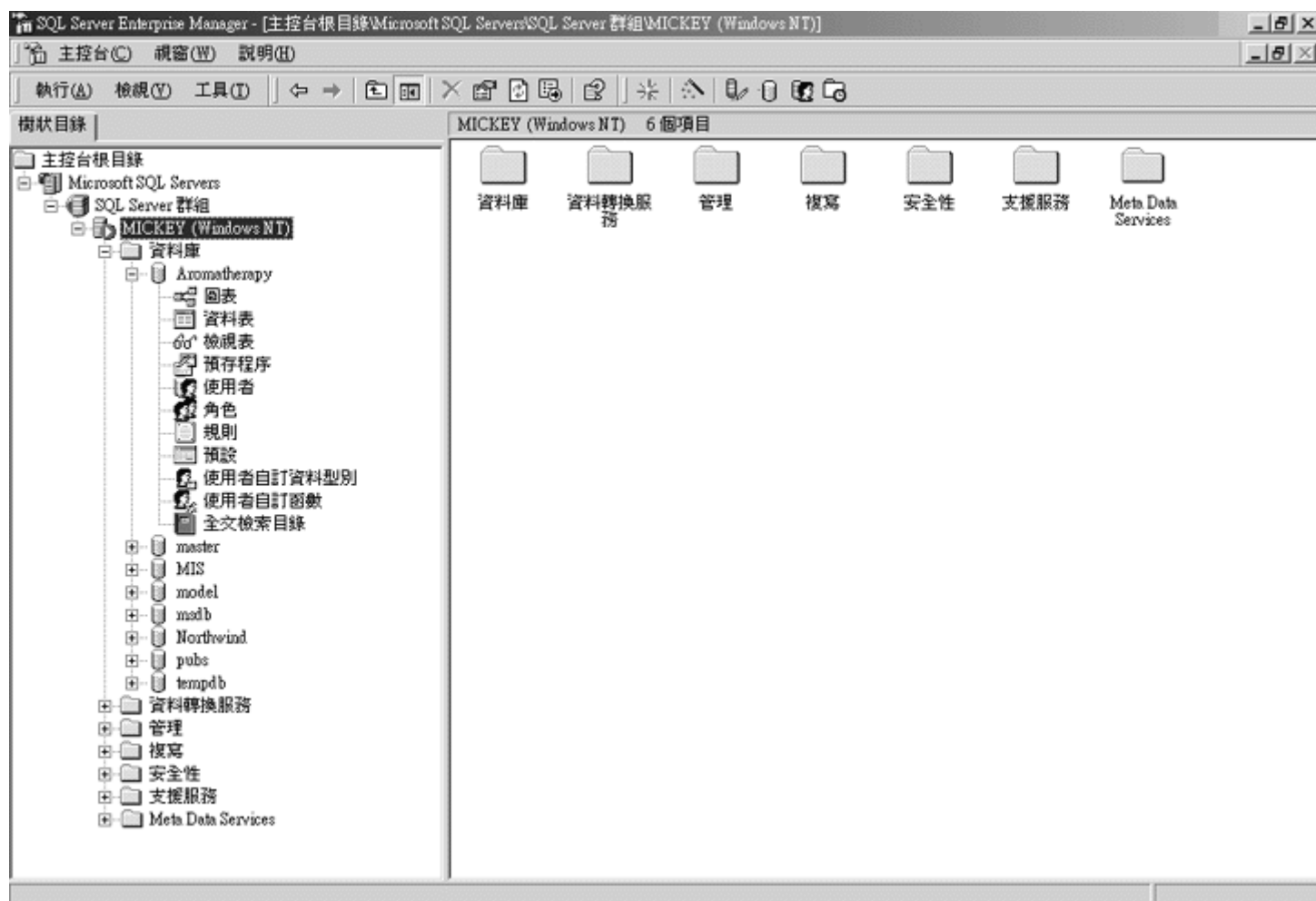
建立登入精灵 会显示一个 **登入已成功建立** 的对话框。



建立 Windows 登入账户

1. 在主控台树状目录之下选取要建立登入账户的服务器。

SQL Server 会在详细资料窗格中显示此服务器内的对象清单。

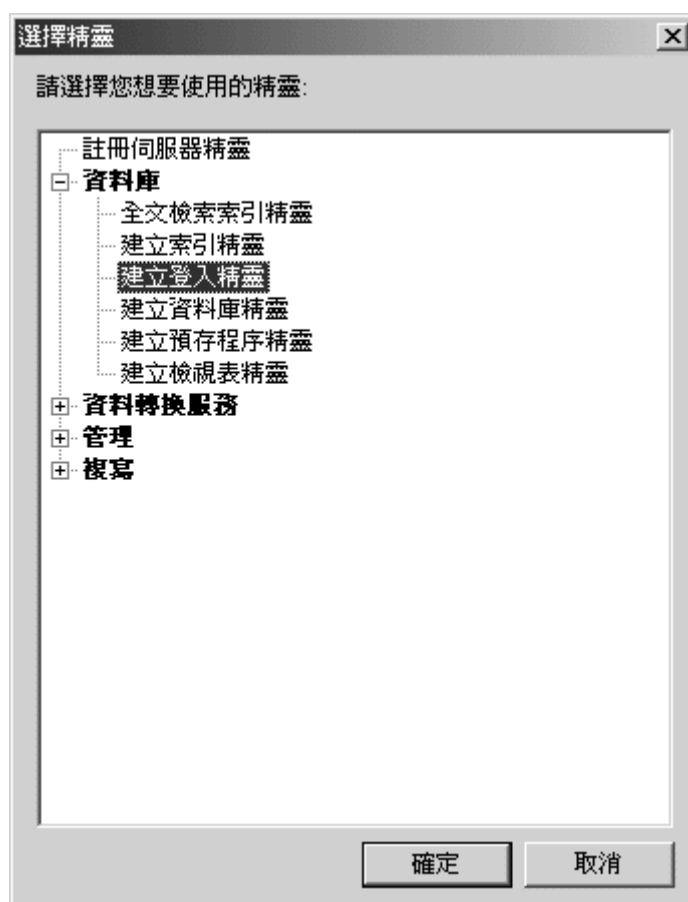


2. 在 Enterprise Manager 的工具列上按一下 [執行精灵](#) 按钮。



执行精灵按钮

SQL Server 会显示 [选择精灵](#) 对话框。



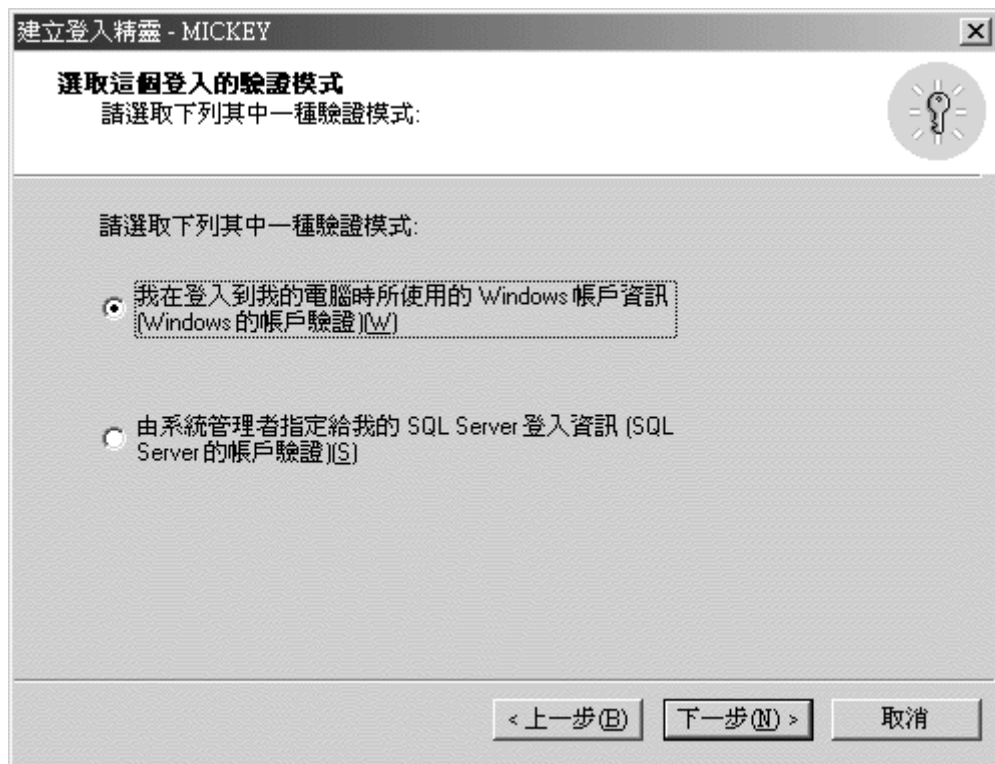
3. 在 [数据库](#) 数据夹中选取 [建立登入精灵](#) 项目，然后再按一下 [确定](#) 按钮。

SQL Server 会显示一个 [建立登入精灵](#) 窗口的第一页画面。



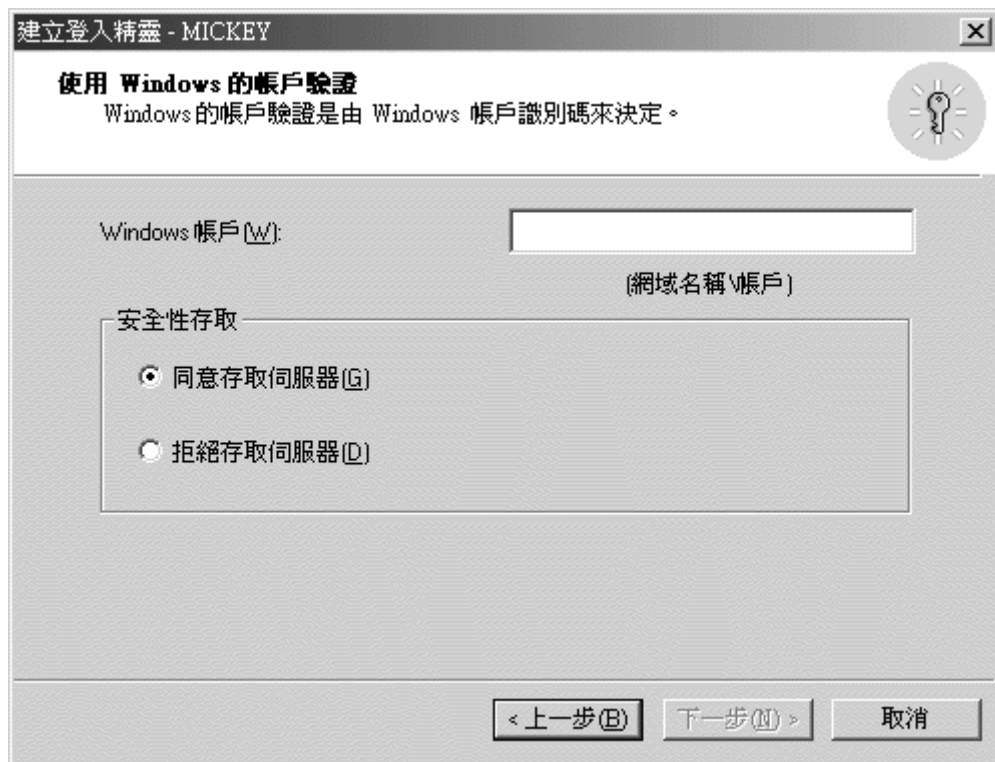
4. 按一下 **下一步** 按钮。

此精灵会询问您要选择哪一种验证模式。在此我们选择预设的项目。



5. 按一下 **下一步** 按钮。

此精灵会要求您输入 **Windows** 的账户名称。



6. 在 **Windows 账户** 项目中输入『网域名称\账户（账户可以为使用者或群组名称）』，
然后按一下 **下一步** 按钮。

此精灵会显示一个要求您指定这个登入账户的安全性角色的画面。

提示

在此画面中的 **拒绝存取服务器** 选项可以用来明确的指定该账户是拒绝存取该服务器的。



7. 选取 **Database Creators** 角色。



8. 按一下 **下一步** 按钮。

此精灵会要求您指定这个账户存取数据库的使用权限。



9. 选取 **Aromatherapy** 数据库。



10. 按一下 **下一步** 按钮。

此精灵会将您所设定的值显示在画面上。



重要

属于特定服务器角色的登入账户—尤其是属于 **System Administrators** 角色的登入账户，都会具备所有数据库的使用权限，无论该登入账号是否已经被授权存取数据库或是没有被授权存取数据库。

11. 按一下 **完成** 按钮。

建立登入精灵 会显示一个 **登入已成功建立** 的对话框。



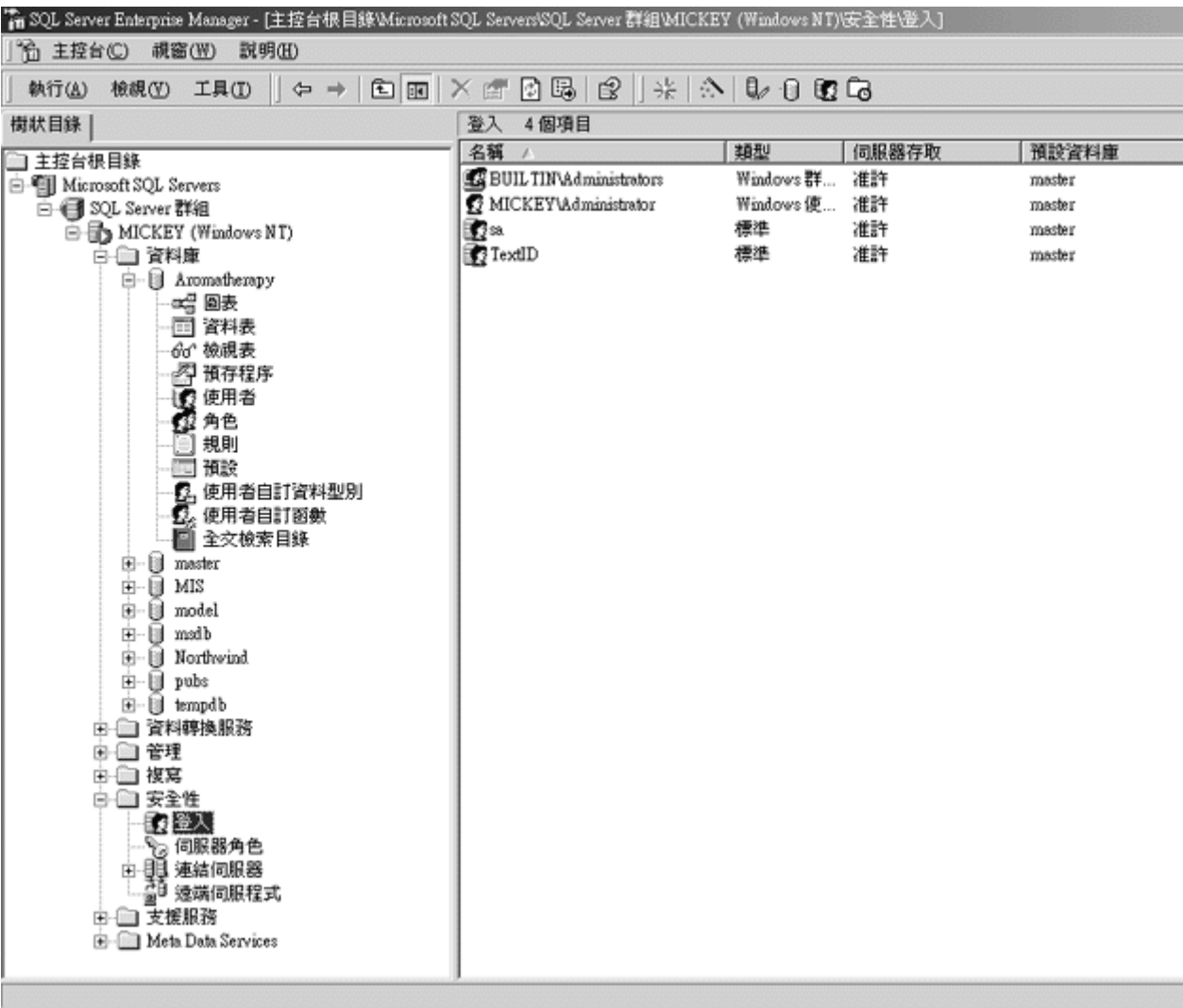
管理登入账户

登入账户和其它的数据库对象一样需要偶尔进行维护，因为您可能需要更改使用者登入账号的属性。举例来说，您可能需要更改已指派的登入账号内的安全性角色，或者是数据库的安全性角色，当然有时您也会需要完全将登入账号进行移除，这些工作您都可以藉由 **Enterprise Manager** 来管理。

▲更改登入账户属性

1. 按一下 **安全性** 数据夹中的 **登入** 图示。

SQL Server 会在详细资料窗格显示登入账户清单。



2. 在详细资料窗格的 **TestID** 登入账户名称上按二下。

SQL Server 会显示一个 **SQL Server 登入属性** 对话框。

SQL Server 登入屬性 - TextID

一般 伺服器角色 資料庫存取

名稱(N): TextID

驗證

☐ Windows 的帳戶驗證(W)

網域(M):

安全性存取:

☐ 同意存取(G)

☐ 拒絕存取(Y)

☒ SQL Server 的帳戶驗證(S)

密碼(P):

預設值

為此登入指定預設的語言及資料庫。

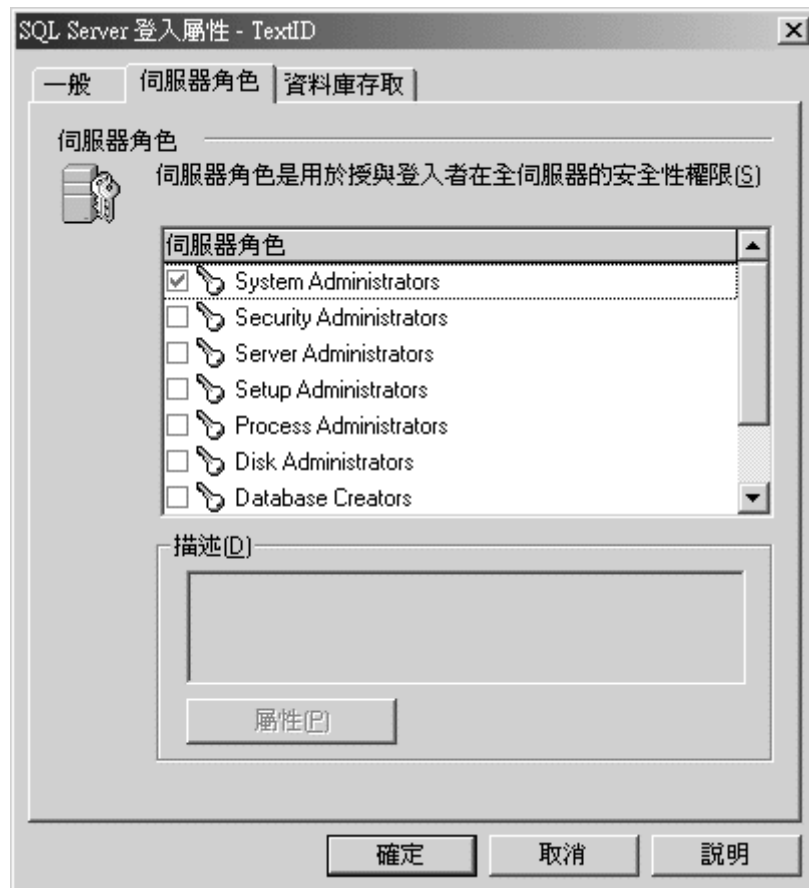
資料庫(D): master

語言(L): Traditional Chinese

確定 取消 說明

3. 选取 [伺服器角色](#) 标签页。

SQL Server 会显示此服务器的所有角色，以便让您选取。



4. 將 **System Administrators** 角色取消选取。



5. 选取 [数据库存取](#) 卷标页。

SQL Server 会显示登入者可以存取的数据库。



- 在清单中选取 **Northwind** 数据库。



7. 按一下 **確定** 按钮，以便关闭 **SQL Server 登入属性** 对话框。

SQL Server 会依照您的设定来改变登入属性。

移除登入账户

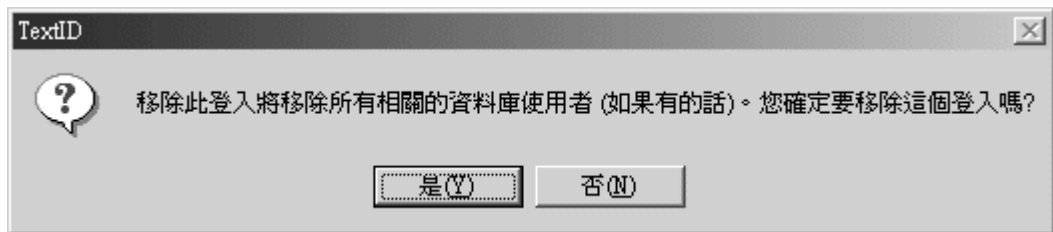
1. 按一下 **安全性** 数据夹中的 **登入** 图示。

SQL Server 会在详细资料窗格中显示登入清单。



2. 按一下详细资料窗格 的 **TestID** 登入账户名称，然后按一下 **Delete** 键。

SQL Server 会显示一个询问您是否真的要删除的讯息。



3. 按一下 **是** 按钮。

SQL Server 会将此登入账户进行删除。

数据库层级的安全性

在数据库层级中，每一个使用者的 **Windows** 账号或 **SQL Server** 登入账号都会对应到数据库内的使用者账号。虽然某个账号已经藉由其登入账号而可以存取 **SQL Server** 的执行个体，但是却没办法存取某个特定的数据库，除非该登入账号已经对应至该数据库中的某一个使用者。

登入账户一旦被分配至服务器角色中，并且藉由该角色授权特定的权限；则使用者便可以被分配至数据库层级的角色中，藉由该数据库角色授权特定的存取权限。

提示

虽然安全性权限可以针对特定使用者个别设定,但是为了管理性的问题,不建议您使用这种方式。

在一个群组中进行新增或删除使用者的安全性权限方式会比您针对每一位使用者来进行安全性权限的设定还要方便。

数据库的使用者

当您使用 [建立登入精灵](#) 来建立新的登入账号,并且分派某数据库的存取权限时,该登入账号会自动地新增至该数据库的使用者清单中。然而,当您建立一个新的数据库时,您也许想要新增一个已经存在的登入账号来当作是该数据库的使用者。

建立数据库的使用者

1. 按一下 Aromatherapy 数据库中的 [使用者](#) 图示。

SQL Server 会在详细数据窗格中显示使用者清单。



提示

您也可以在主控台树状目录之下的 [使用者](#) 图示上按右钮，并在所出现的快捷菜单中选取 [新增数据库使用者](#) 项目，以便开启 [数据库使用者属性](#) 的对话框。

3. 在下拉式清单方块中选取您的账户名称。

SQL Server 会将使用账户名称作为预设的使用者名称，假如您有需要的话，您可以改变使用者名称。

4. 选取 **db_owner** 以将该使用者作为该数据库角色成员。



5. 按一下 **确定** 按钮。

SQL Server 会将此使用者加入到数据库中。

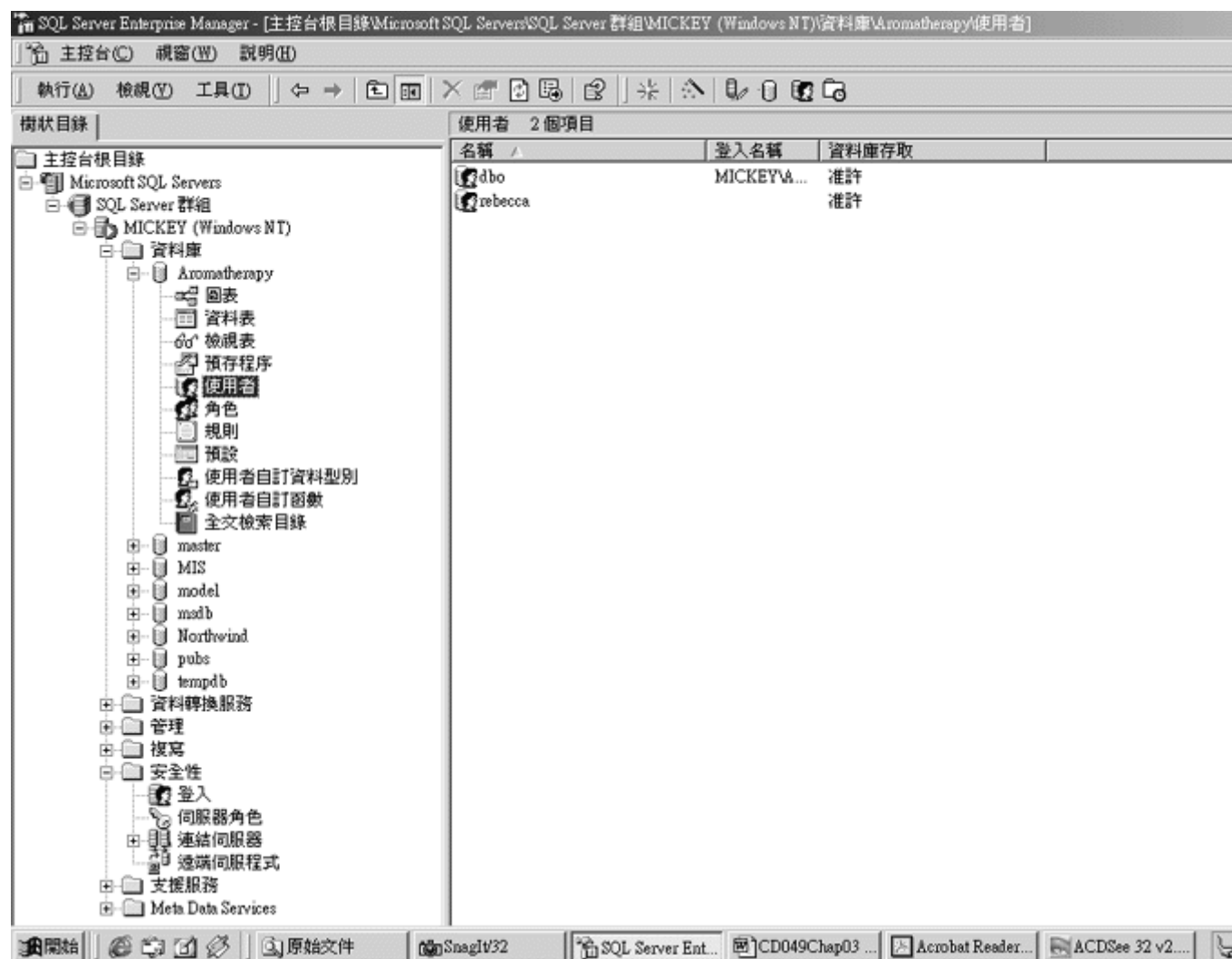
提示

数据库角色的指派会决定该使用者对该数据库的存取权限，如果您需要了解更多的信息，您可以参考 [〈数据库角色〉](#) 一节的说明。

自数据库角色中移除使用者

1. 按一下 Aromatherapy 数据库内的 **使用者** 图示。

SQL Server 会在详细数据窗格中显示使用者清单。



2. 选取您在之前的练习中所建立的使用者，然后按一下 **Delete** 按钮。

SQL Server 会显示一个询问您是否真的要删除的讯息。



3. 按一下 [是](#) 按钮。

SQL Server 将会自数据库中删除此使用者。

数据库角色

一个 [数据库角色](#)（Database Role）就像是一位虚拟的使用者一样，您可以建立数据库角色以管理数据库的存取。您可以指派许多使用者到一个角色中，并且为任何的使用者指派多个角色。当您把权利指派给数据库角色，然后再指派一位使用者给此角色，此时此使用者将会继承此角色的所有权利。SQL Server 2000 提供一些如表 3-2 所示的数据库角色，您也可以建立自己的角色，但是您所建立的角色对于您的数据库而言应该是唯一的。

完整的名称	名称	说明
Access Administrator	db_accessadmin	可新增或移除使用者
Backup Operator	db_backupoperator	可执行 DBCC、CHECKPOINT 与 BACKUP 陈述式
Data Reader	db_datareader	可在该数据库的任何使用者数据表中选取所有数据
Data Writer	db_datawriter	可在该数据库的任何使用者 数据表中修改任何数

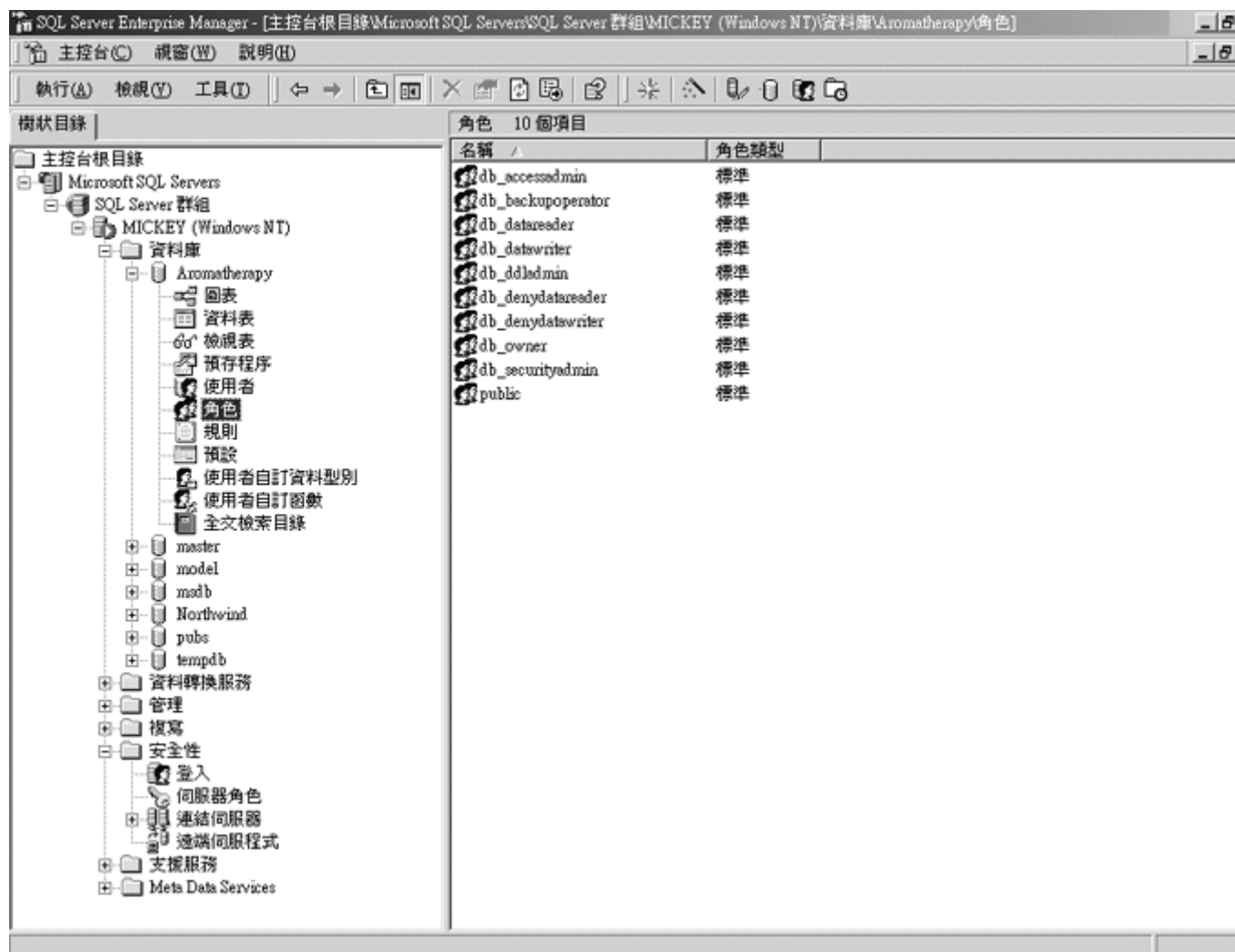
		据
Data Definition Administrators	db_ddladmin	可以在该数据库中执行数据定义语言（Data Definition Language，DDL）陈述式，但不得执行 GRANT、REVOKE 或 DENY 陈述式
Deny Data Reader	db_denydatareader	可以在该数据库中的任何对象上执行拒绝或撤销 SELECT 陈述式的权限。
Deny Data Writer	db_denydatawriter	可以在该数据库中的任何对象上执行拒绝或撤销 INSERT、UPDATE 和 DELETE 陈述式的权限。
Database Owner	db_owner	对该数据库拥有所有的使用权限
Security Administrator	db_securityadmin	可以在该数据库中管理所有权限、对象拥有权、角色与角色成员关系
Public		这是预设的角色，每一个数据库使用者都是属于 Public 数据库角色。

表 3-2

建立数据库角色

1. 按一下 Aromatherapy 数据库内的 角色 图示。

SQL Server 会在详细数据窗格中显示数据库角色清单。



2. 在 Enterprise Manager 的工具列上按一下 **新增** 按钮。



新增按钮

SQL Server 会显示 [数据库角色属性](#) 对话框。

資料庫角色屬性 - 新角色

一般

名稱(N): 權限(P)...

SQL Server 支援兩種資料庫角色: 包含成員的標準角色，以及需要密碼的應用程式角色。

資料庫角色類型:

☒ 標準角色(S)

使用者

新增(D)... 移除(R)

☐ 應用程式角色(C)

密碼(W):

確定 取消 說明

3. 在 **名称** 字段中输入『Lesson 3』。



4. 按一下 **确定** 按钮。

SQL Server 会将此 **数据库角色属性** 对话框关闭。

5. 在详细资料窗格的 **Lesson 3** 角色名称上按二下。

SQL Server 会显示一个 **数据库角色属性** 对话框。



重要

要启用 [权限](#) 按钮, 您必须关闭并重新开启 [数据库角色属性](#) 对话框。

- 按一下 [权限](#) 按钮。

SQL server 会显示一个有 权限 卷标页的对话框。



提示

数据库角色属性 对话框支持二种类型的角色：一是 标准角色 ，另一种则是 应用程序角色 。在本课程中所讨论的角色都是属于标准角色。应用程序角色是一个特殊功能的角色，

它用以支持复杂的应用程序需求。您可以在 [SQL Server Books Online](#) 的〈Establishing Application Security and Application Roles〉一节中找到关于应用程序角色的说明。

7. 将 Oils 数据表授权 SELECT 权限。



8. 按一下 **确定** 按钮。

SQL Server 会将 **权限** 对话框关闭。

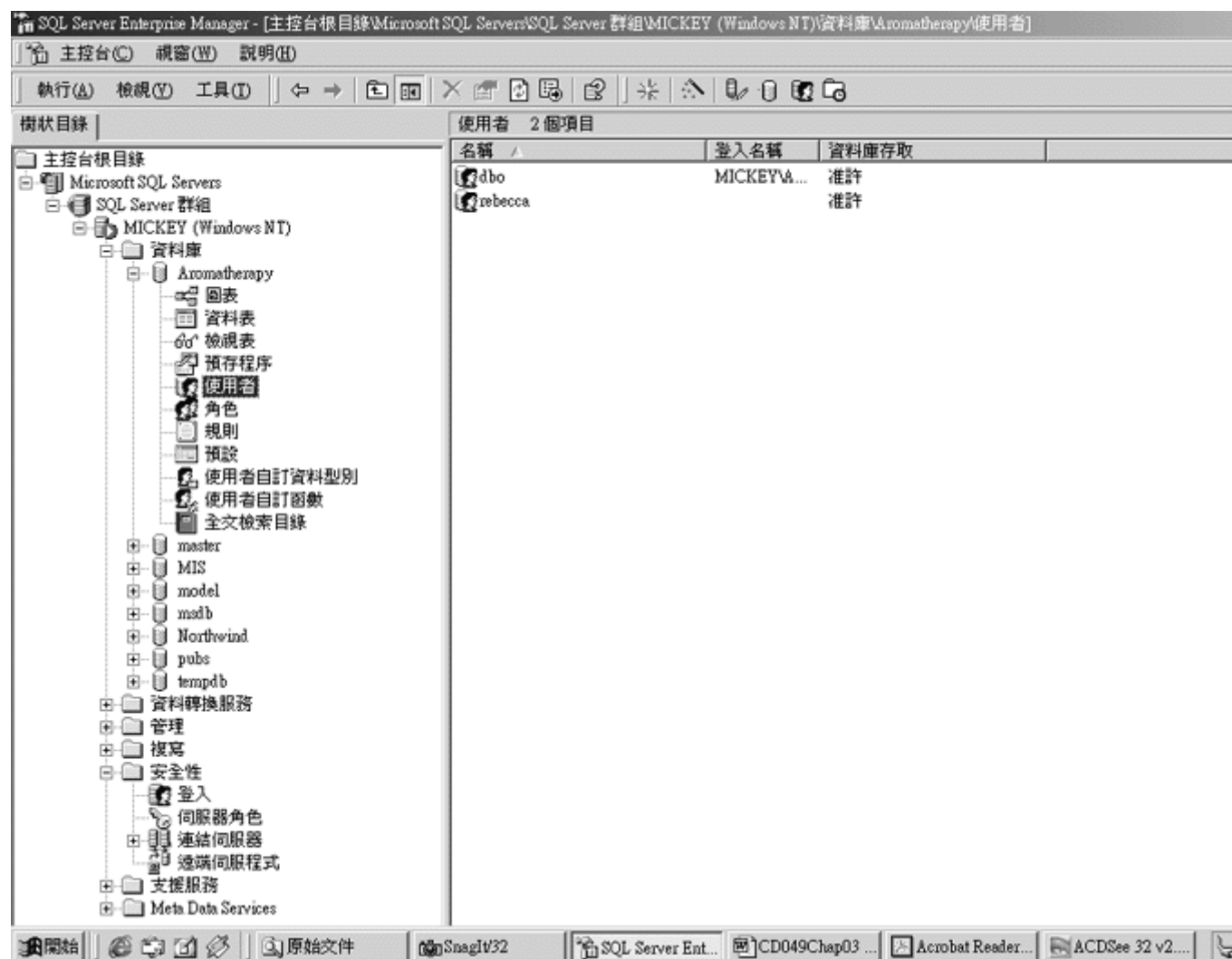
9. 按一下 **确定** 按钮。

SQL Server 将会新增此数据库角色。

将使用者指派至数据库角色中

1. 按一下 Aromatherapy 数据库内的 **使用者** 图示。

SQL Server 会在详细数据窗格中显示使用者清单。



- 按两下详细数据窗格中的使用者名称。

SQL Server 会显示一个 [数据库使用者属性](#) 对话框。



3. 在 [数据库角色成员](#) 清单中选取 Lesson 3 的复选框以加入该使用者。



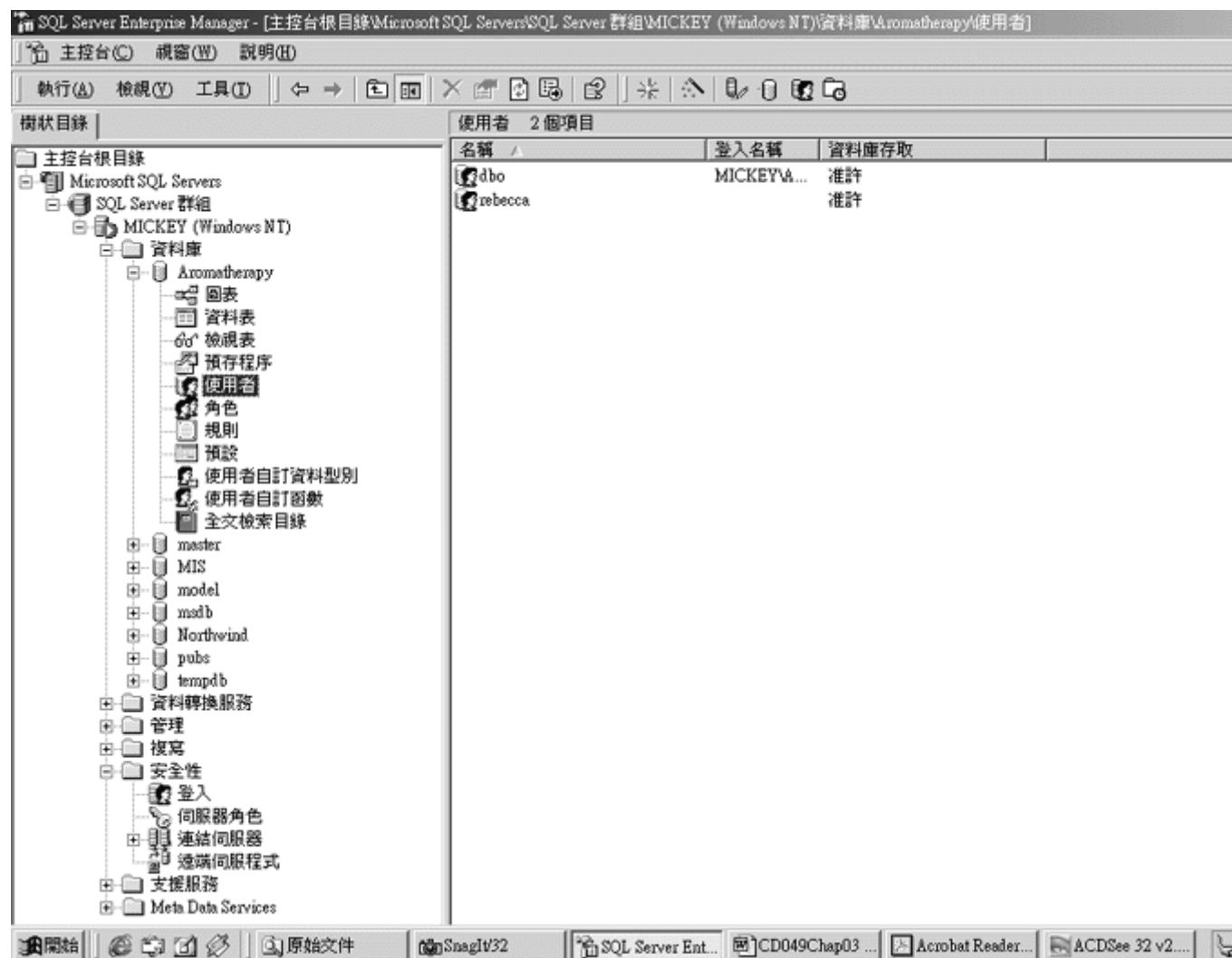
4. 按一下 **确定** 按钮。

SQL Server 会将此使用者加入至该数据库角色中，并且将 **数据库使用者属性** 对话框关闭。

自数据库角色中移除使用者

1. 按一下 Aromatherapy 数据库内的 **使用者** 图示。

SQL Server 会在详细数据窗格中显示使用者清单。



- 按两下详细数据窗格中的使用者名称。

SQL Server 会显示一个 [数据库使用者属性](#) 对话框。



3. 在 [数据库角色成员](#) 清单中，将您要移除的角色取消选取。



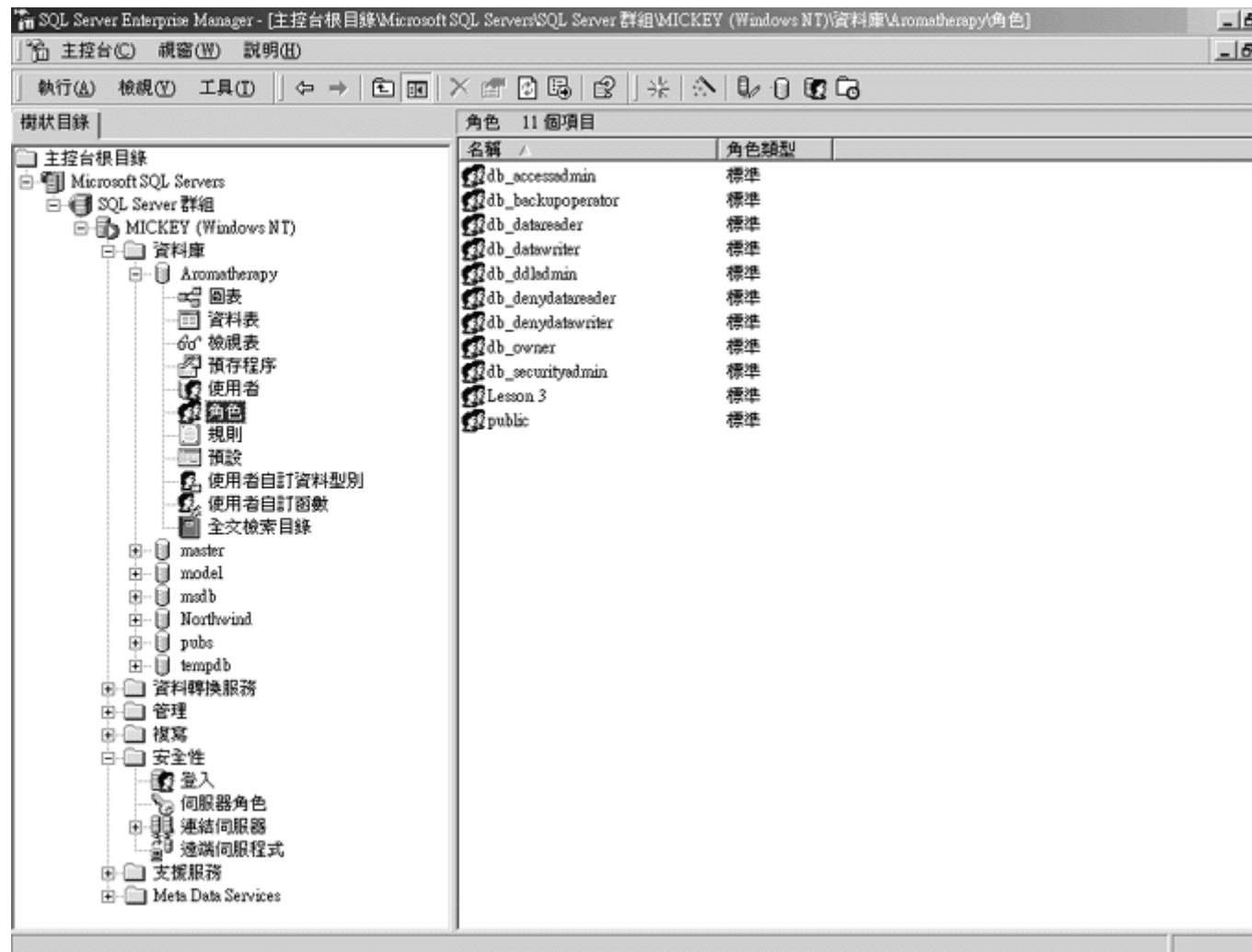
4. 按一下 **确定** 按钮。

SQL Server 会将此使用者自数据库角色中移除, 并且将 **数据库使用者属性** 对话框关闭。

移除数据库角色

1. 按一下 Aromatherapy 数据库内的 **角色** 图示。

SQL Server 会在详细数据窗格中显示角色清单。



2. 在详细数据窗格中选取 Lesson 3 的角色，然后按一下 **Delete** 键。

SQL Server 会显示一个询问您是否真的要删除的讯息。



3. 按一下 [是](#) 按钮。

SQL Server 将会自数据库中移除此角色。

本章总结

要执行的工作	执行
建立 SQL Server 登入账户	在主控制台树状目录选取服务器，然后在 Enterprise Manager 工具列上按一下 执行精灵 按钮以开启 选择精灵 对话框，在 数据库 数据夹中选取 建立登入精灵 ，选取 由系统管理者指定给我的 SQL Server 登入信息[SQL Server 的账户验证] 项目。
建立 Windows 登入账户	在主控制台树状目录选取服务器，然后在 Enterprise Manager 工具列上按一下 执行精灵 按钮以开启 选择精灵 对话框，在 数据库 数据夹中选取 建立登入精灵 ，选取 我在登入到我的计算机时所使用的 Windows 账户信息[Windows 的账户验证] 项目。
移除登入账户	按一下 安全性 资料夹中的 登入 图标，在详细数据窗格中选取您想要移除的登入账户，然后按一下 Delete 键。

建立数据库的 使用者	在 数据库 数据夹中的 使用者 图示上按右钮，在快捷菜单中选取 新增数据库使用者 项目。在下拉式方块中选取登入账户，假如有需要的话，您可以改变使用者名称，然后按一下 确定 按钮。
移除数据库的 使用者	按一下 安全性 数据夹内的 使用者 图标，在详细数据窗格中选取您想要移除的使用者，然后按一下 Delete 键。
建立数据库的 角色	在 数据库 数据夹中的 角色 图示上按右钮，在快捷菜单中选取 新增数据库角色 项目。在对话框中输入数据库角色的名称，然后按一下 权限 按钮以便指定该角色的权限。
将使用者指派 至数据库角色 中	按一下 安全性 数据夹中的 使用者 图示，按两下详细数据窗格中的使用者名称，在数据库角色成员清单中，选取该使用者要加入的角色。
自数据库角色 中移除使用者	按一下 安全性 数据夹中的 使用者 图示，按两下详细数据窗格中的使用者名称，在数据库角色成员清单中，将想要移除的使用者名称取消选取。
移除数据库角 色	按一下 安全性 数据夹中的 使用者 图标，在详细数据窗格中选取您想要移除的角色名称，然后按一下 Delete 键。

第二篇 建立数据库

4. 建立数据库

- . 建立数据库
- . 管理数据库
- . 本章总结

5. 建立数据表

- . 建立数据表
- . 新增数据行至数据表中
- . 管理数据表
- . 本章总结

6. 建立索引

- . 认识索引
- . 维护索引
- . 移除索引
- . 本章总结

7. 建立关联性

- . 认识关联性
- . 管理关联性
- . 本章总结

8. 建立检查条件约束

- . 认识检查条件约束
- . 管理检查条件约束
- . 本章总结

9. 建立数据表对象

- . 认识预设
- . 认识规则
- . 认识使用者自订数据类型
- . 本章总结

10. 建立数据库图表

- . 认识数据库图表
- . 使用数据库图表以维护数据库
- . 本章总结

4. 建立数据库

在本章中，您将学习到：

- 建立一个新的数据库。
- 更改数据库的属性。
- 删除数据库。

在 **Microsoft SQL Server** 的环境中，您可以储存数据表、检视表以及数据库中其它相关的对象。建置数据库应用程序的第一个步骤就是建立数据库，我们将在本章中说明如何建立数据库。

建立数据库

对于每一个有逻辑的数据库而言，**SQL Server** 会建立二个实体的档案：一个档案供对象使用、另一个档案则是供交易记录使用。

提示

在预设的情况下，SQL Server 的数据库档案和交易记录档案会储存在同一个地方，但是在实际执行的系统中，您可能需要将交易记录档案存放在不同的地方，甚至是存放在不同的机器上。这样可以在发生硬件故障的情形时，仍然可以从别的硬盘还原交易纪录文件。

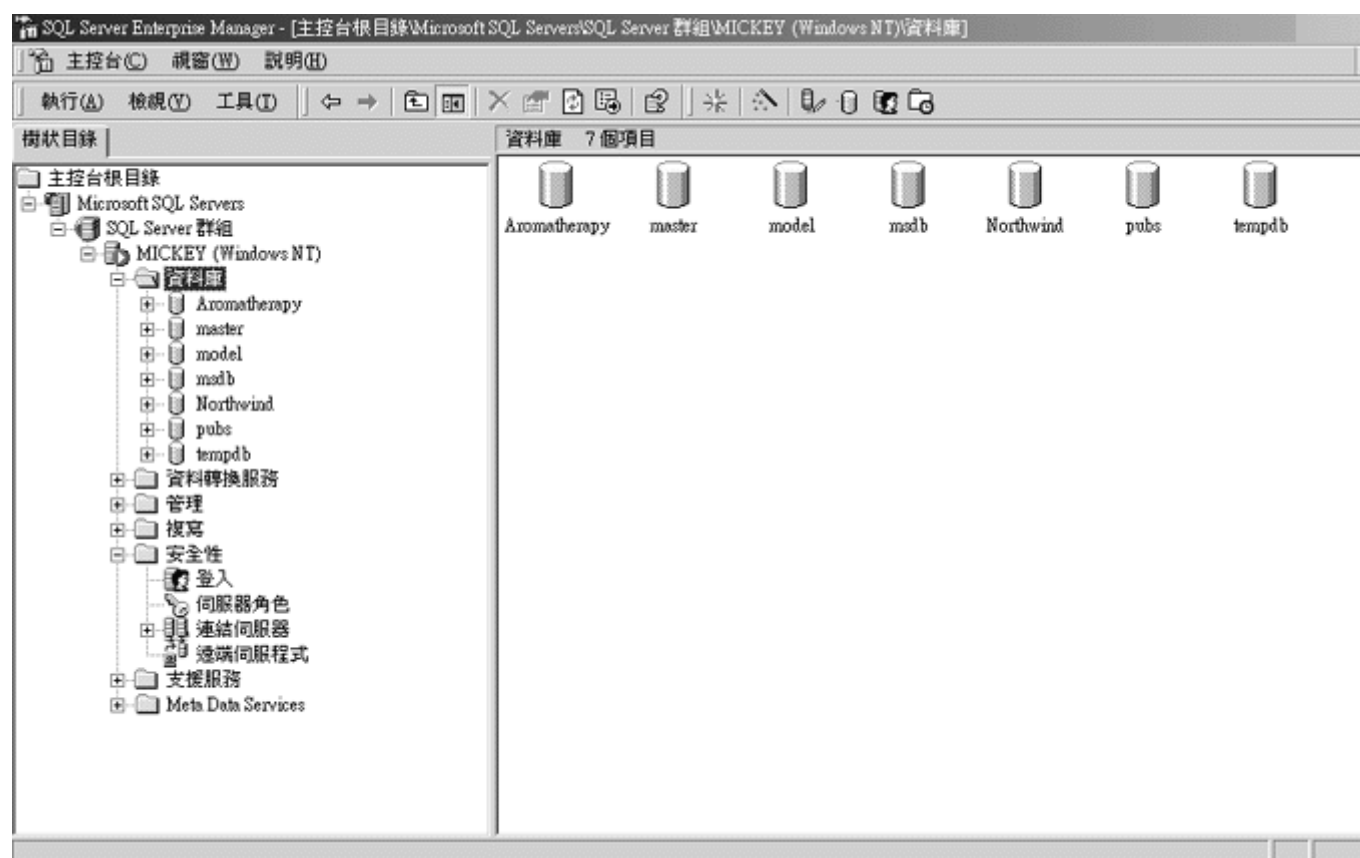
建立新的数据库

您可以在 [数据库](#) 数据夹的快捷菜单中选取 [新增数据库](#) 来建立数据库。更简单的方式是使用 [建立数据库精灵](#)。

建立新的数据库

1. 按一下您在本课程中所使用服务器的 [数据库](#) 数据夹。

SQL Server 会在内容窗格中显示此数据库内容的列表。

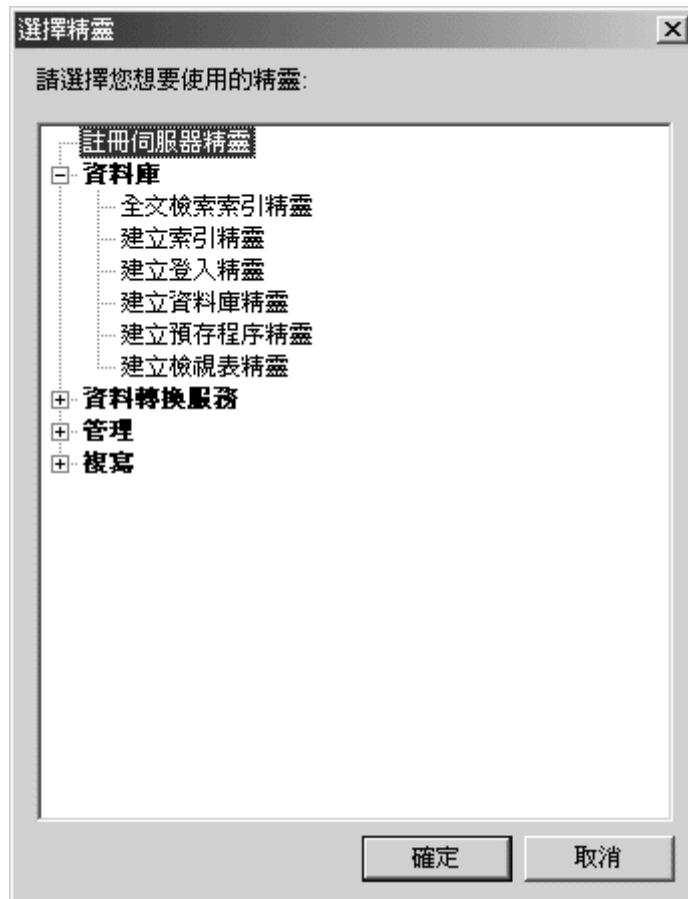


2. 在 Enterprise Manager 的工具列上按一下 [执行精灵](#) 按钮。



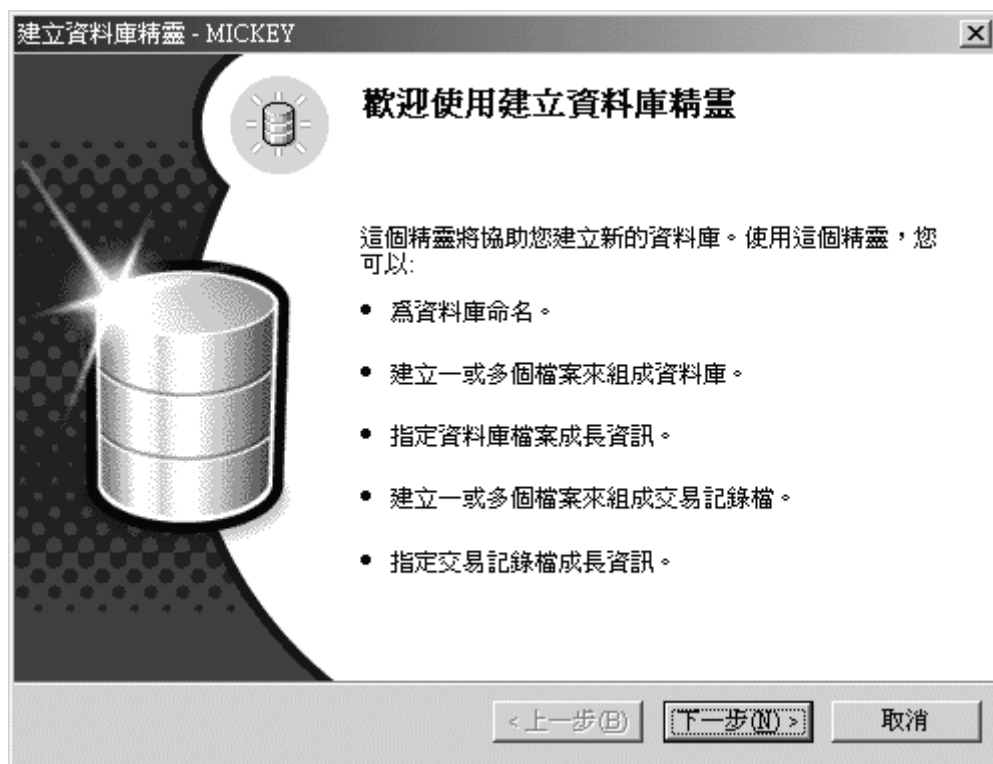
执行精灵按钮

SQL Server 会显示一个 [选择精灵](#) 的对话框。



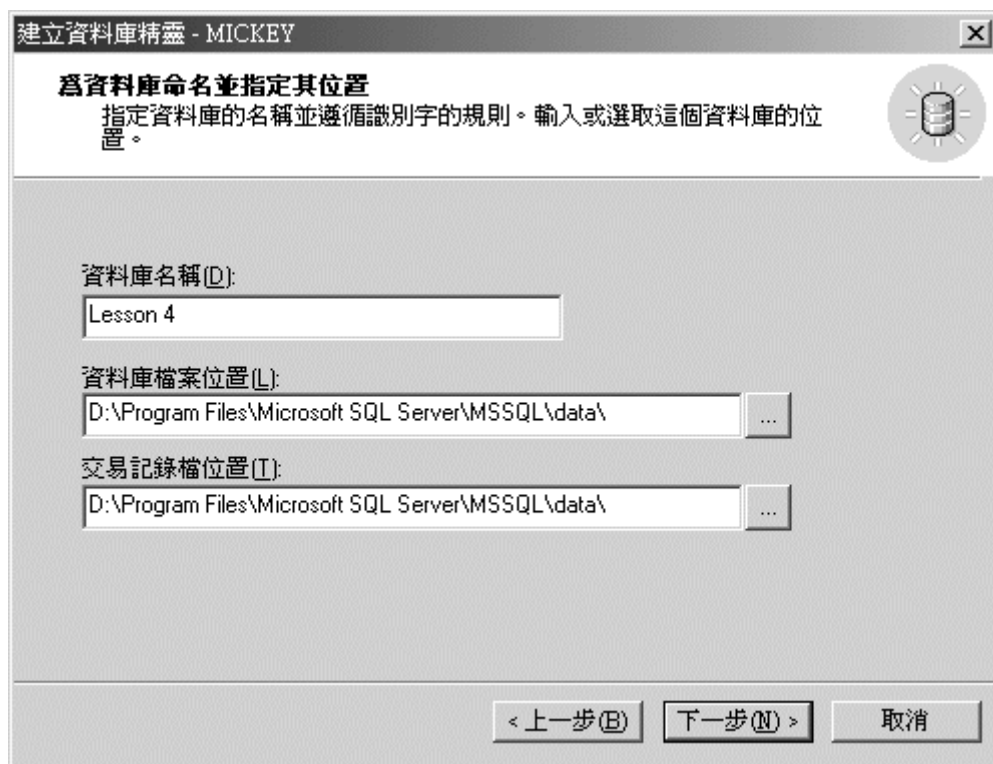
3. 在 [数据库](#) 数据夹中选取 [建立数据库精灵](#) 项目。

SQL Server 会显示此精灵的第一页画面。



4. 按一下 [下一步](#) 按钮。

[建立数据库精灵](#) 会要求您输入新建立的数据库名称。



5. 在 **数据库名称** 字段中输入『Lesson 4』。
6. 按一下 **浏览** 按钮以便更改数据库的档案目录位置。



浏览按钮

此精灵会显示一个要求您 **选取档案所使用的目录** 对话框。



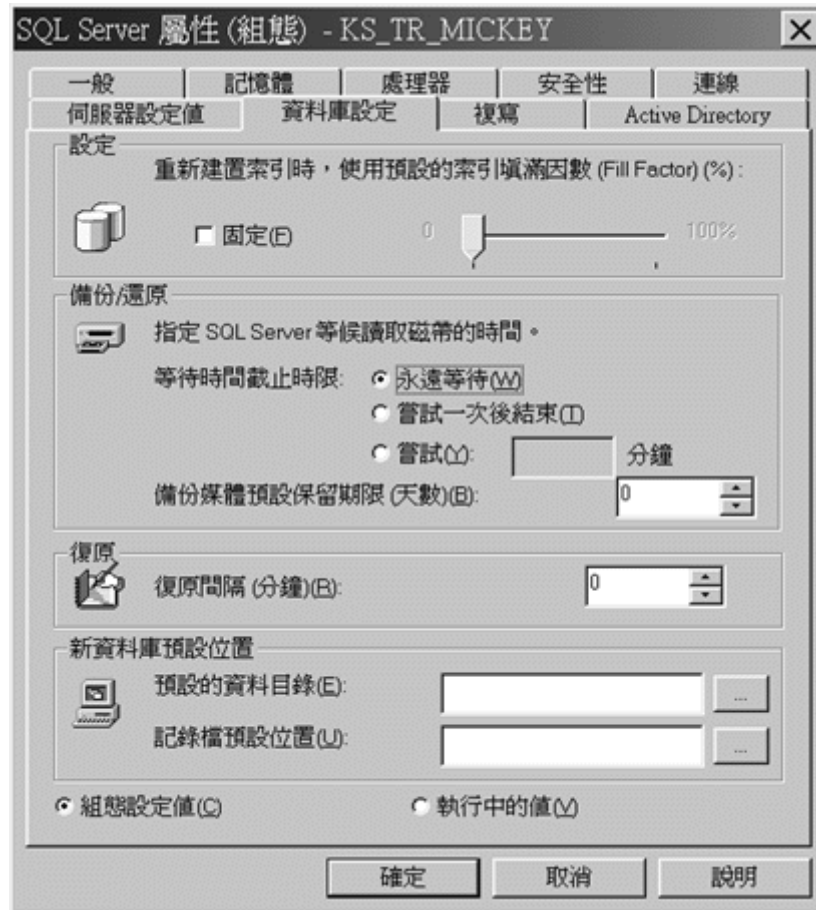
数据库档案位置

当 SQL Server 第一次安装完毕之后，预设新增的数据库档案都会储存

在 **MSSQL\data** 的数据夹之下。 [建立数据库精灵](#) 所建立的数据库档案也是放置在此。

要永久地改变其默认值，您可以在主控台树状目录的服务器名称上按右钮，选取 [内容](#)，

然后再选取 [数据库设定](#) 卷标页以设定新的储存位置。



-
7. 指向根目录之下的 SQL 2000 Step by Step 数据夹，然后按一下 **确定** 按钮。

此精灵会将此数据库档案位置设定为您所选取的档案目录。

8. 按一下 **浏览** 按钮以更改交易记录文件的档案目录位置。
-



浏览按钮

此精灵会显示一个要求您 [选取档案所使用的目录](#) 对话框。

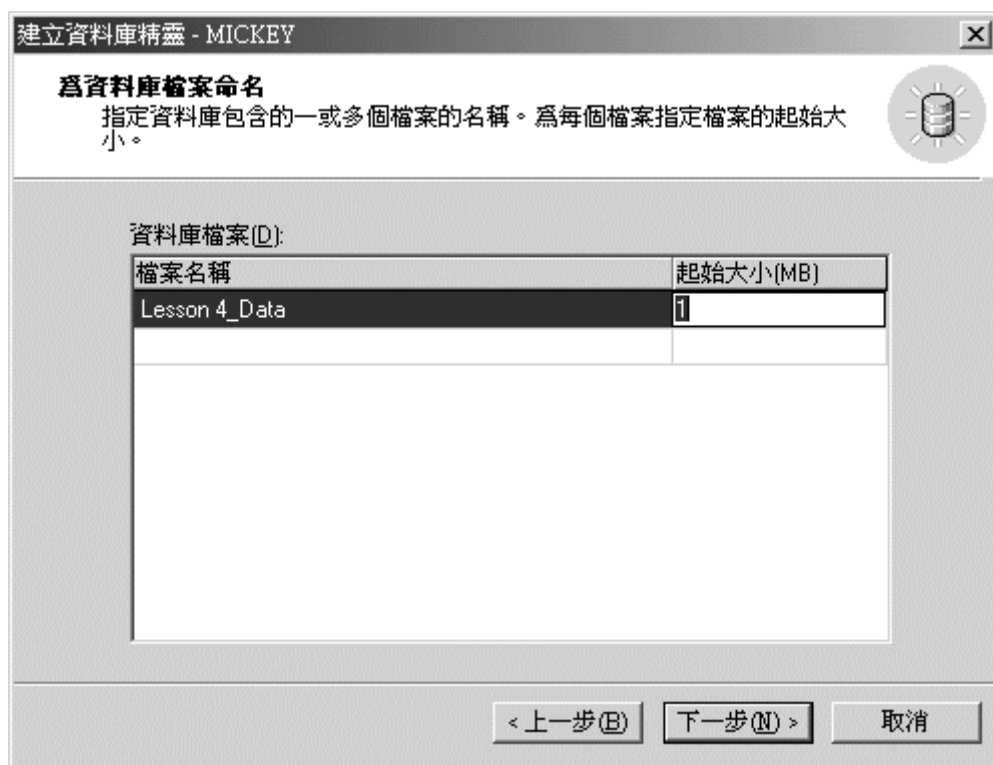


9. 指向根目录之下的 SQL 2000 Step by Step 数据夹，然后按一下 [确定](#) 按钮。

此精灵会将此交易记录档案位置设定为您所选取的档案目录。

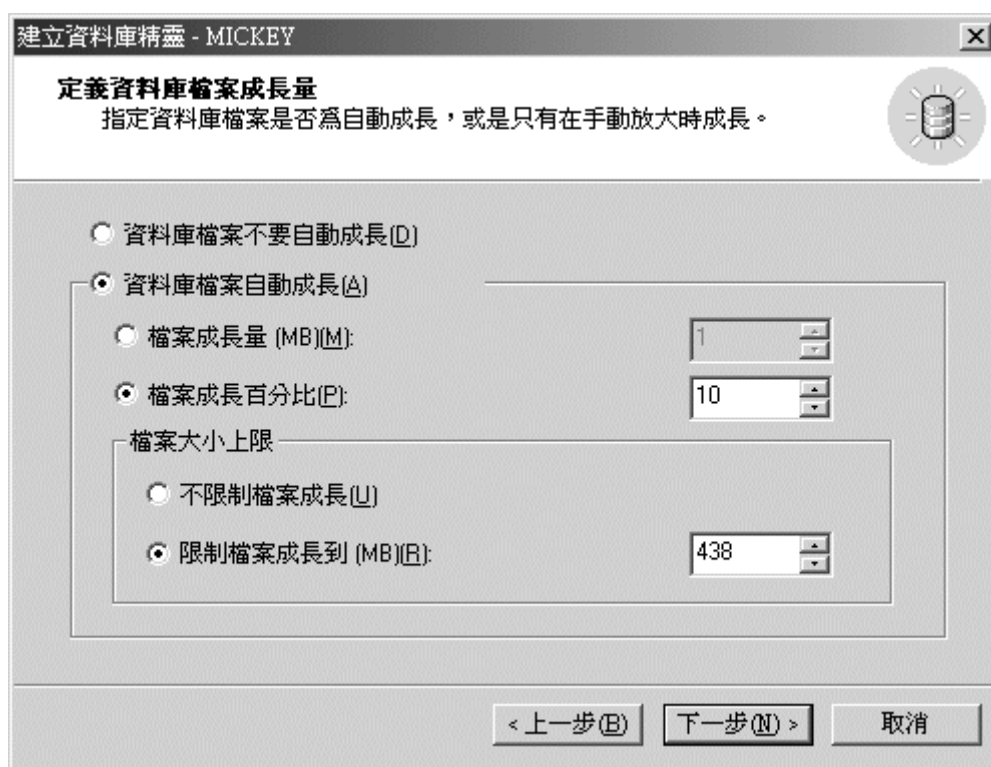
10. 按一下 **下一步** 按钮。

此精灵会要求您针对此数据库设定起始大小。



11. 按一下 **下一步** 按钮。

此精灵会让您选择数据库档案是否要自动地成长或者不要自动成长。在此练习中，我们采用预设的设定值。



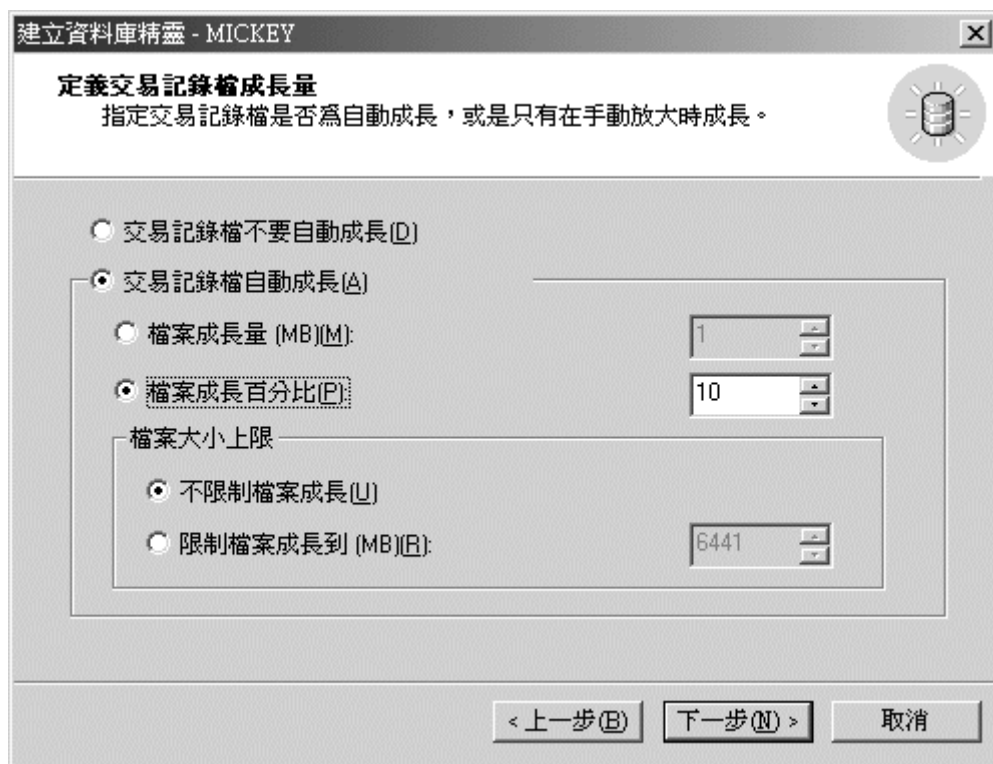
12. 按一下 **下一步** 按钮。

此精灵会要求您针对此数据库的交易记录档案来设定起始的大小。我们采用预设的设定值。



13. 按一下 [下一步](#) 按钮。

此精灵会让您选择其数据库的交易记录档案是否要自动地成长或者不要自动成长。在此练习中，我们采用预设的设定值。



14. 按一下 **下一步** 按钮。

此精灵会将您所设定的选项显示在画面上。



15. 按一下 **完成** 按钮。

此精灵会建立数据库及交易记录档案，然后会询问您是否要为此新的数据库建立维护计划。



16. 按一下 **否** 按钮。

此精灵窗口将被关闭。

设定数据库属性

当您使用 [建立数据库精灵](#) 来建立数据库时，您需要指定数据库的特性或属性，例如数据库的名称以及档案的位置。在建立数据库之后，您就可以藉由数据库的属性对话框来更改这些属性值。

举例来说，为了要扩充实体数据文件的大小（在 **SQL Server** 的说法称为档案的成长）是相当地浪费系统资源，并且会延迟服务器响应的时间。假如您发现 **SQL Server** 经常在扩充档案的大小时，那您可能要考虑在属性对话框中改变档案成长的百分比，此时您就可能会将档案成长的百分比设定为比预设的百分之十还要多。

更改数据库档案成长百分比值

1. 在主控台树状目录下选取 **Lesson 4** 数据库。
2. 在工具列上按一下 [内容](#) 按钮。



内容按钮

SQL Server 会显示一个 [数据库的属性](#) 对话框。



3. 选取 [数据文件](#) 标签页。

SQL Server 会显示此数据库的数据文件的属性。



4. 将档案成长量的百分比设定为 20。



5. 按一下 **确定** 按钮。

SQL Server 会使用您所设定的新属性值，并且将属性对话框关闭。

管理数据库

除了改变数据库的属性，您可能会需要删除整个数据库，Enterprise Manager 可以让您很容易地执行这项工作。

说明

您也可能会执行其它的管理工作，例如为数据库重新命名，但这个功能无法在 **Enterprise Manager** 中执行，我们将在第 28 章中讨论关于数据库重新命名的方式。

删除数据库

当您建立的数据库不再需要时，您就可以从服务器中删除它。删除数据库会移除与此数据库有关的实体档案和在系统数据表中引用该数据库的引用信息。

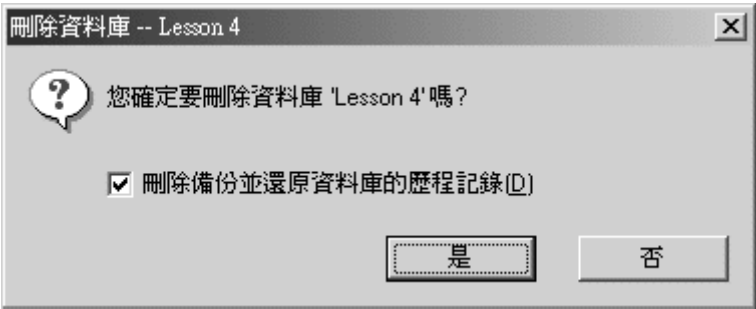
提示

在数据库被删除之后，最好将系统数据库执行备份。

删除数据库

1. 在主控台树状目录下选取 **Lesson 4** 数据库。
2. 按一下 **Delete** 键。


SQL Server 会出现一段确认您是否真的要删除此数据库的讯息。



- 3. 按一下 [是](#) 按钮。

SQL Server 会自服务器中将此数据库以及与它有关联的对象全部删除。

本章总结

要执行的工 作	执行	按 钮
建立新的数 据库	按一下 执行精灵 按钮，在 数据库 数据夹中选取 建立数据库精灵 ，然后依照 精灵所提示的步骤进行建立。	
设定数据库 属性	在主控台树状目录下选取数据库并按右钮，然后选取 内容 。	

删除数据库

在主控台树状目录下选取数据库并按一下 **Delete** 键。

5. 建立数据表

在本章中，您将学习到：

- 建立新的数据表。
- 新增数据行至数据表中。
- 储存及关闭数据表。
- 重新命名数据行。
- 移除资料行。
- 重新命名数据表。
- 移除资料表。

在 Microsoft SQL Server 的环境中，就像在任何一个关系型的数据库中，其信息是有系统地储存在数据表中。数据表为一个单一的对象，用于储存数据，并且使用数据列和数据行为数据进行有顺序的排列。

在本章中，您将学习到如何建立数据表，以及指定数据行来组成数据表。

提示

您可以将数据表想象成一个方格或活页簿，但是，在数据表中的记录本质上是沒有固定顺序的。

因此在数据表中，您不能应用「上一笔」、「下一笔」记录的观念。

假如您需要执行这类的操作时，您可以建立 **数据指针**（Cursor），它是一个实体，可以用来在记录集中指向某一个特定的数据列，我们将在第 27 章中学习什么是数据指针。

建立数据表

在关系型数据库中，数据表是数据储存的基本单位。例如在我们的范例数据库中一基本的 Oils 数据表代表一个 **实体**（entity），而实体内的每一个个体（instance），例如 Clary Sage 或 German Chamomile 所代表的是数据表内的一个资料列。

数据库的设计

定义数据库大多数的方式是使用 **实体**（entity）和 **属性**（attribute）的概念。当您从逻辑设计中移转到实体建置时，此时实体通常会被当作是数据表来建置，而属性会被当作是数据行（也就是大家都知道的字段）。

认识资料型别

在数据表中的每一个数据行都会有特定的属性，而这些属性中最重要的就是数据行的 **数据类型**（Data Type），数据类型是用来定义储存在数据行中的数据。SQL Server 提供了许多的数据类型来供您使用，如表 5-1 所示。

资料型别	可接受的值范围
整数值	
bigi nt	自-2 ⁶³ 至 2 ⁶³ -1 的整数值。
int	自-2 ³¹ 至 2 ³¹ -1 的整数值。
smallint	自-2 ¹⁵ 到 2 ¹⁵ -1 的整数值。

tinyint	从 0 到 255 的整数值。
bit	其值为 1 或 0 的整数值。
decimal	固定有效位数及小数字数的数字数据类型，其值为从 $-10^{38}+1$ 到 $10^{38}-1$ (decimal 值也可以定义为 numeric ，值的范围相同)。
money	从 -2^{63} 到 $2^{63}-1$ 的货币值，精确度到每单位千分之十。
smallmoney	从-214, 748, 3648 到+214, 748. 3647，精确度到每单位千分之十
float	从 $-1.79E+308$ 到 $1.79E+308$ 的浮点数数字。(浮点资料是近似值)
real	从 $-3.40E+38$ 到 $3.40E+38$ 的浮点数数字。(浮点资料是近似值)
日期及时间数据	
datetime	从 1753 年 1 月 1 日到 9999 年 12 月 31 日的日期时间数据，精确度为 3.33 毫秒。
smalldatetime	从 1900 年 1 月 1 日到 2079 年 6 月 6 日，精确度为一分钟。
字符串	
char	固定长度的非 Unicode 字符数据，最大长度为 8,000 个字符。
varchar	可变长度的非 Unicode 数据，最大长度为 8,000 个字符。
text	可变长度的非 Unicode 数据，最大长度为 $2^{31}-1$ 个字符
nchar	固定长度的 Unicode 数据，最大长度为 4,000 个字符。
nvarchar	可变长度的 Unicode 数据，最大长度为 4,000 个字符。

ntext	可变长度的 Unicode 数据，最大长度为 $2^{30}-1$ 个字符。
二进制	
binary	固定长度的二进制数据，最大长度为 8,000 个字节。
varbinary	可变长度的二进制数据，最大长度为 8,000 个字节。
image	可变长度的二进制数据，最大长度为 $2^{31}-1$ 个字节。
其它的值	
cursor	参照数据指针（数据指针是一个实体，会建立参照到结果集中的某数据列）。
rowversion	数据库层级的唯一数字，每当一数据列更新时，此数字便随之更新 (rowversion 数据型别在前一个版本的 SQL Server 中称为 timestamp)。
sql_variant	此数据型别可以储存除了 text、ntext、rowversion (timestamp) 与 sql_variant 以外的各种 SQLServer 支持的数据型别。
Uniqueidentifier	全域唯一识别码 (GUID)。

表 5-1 SQL Server 所提供的数据型别(这个表格只有 5 个黑体，都在数据型别那一栏)

建立新的数据表

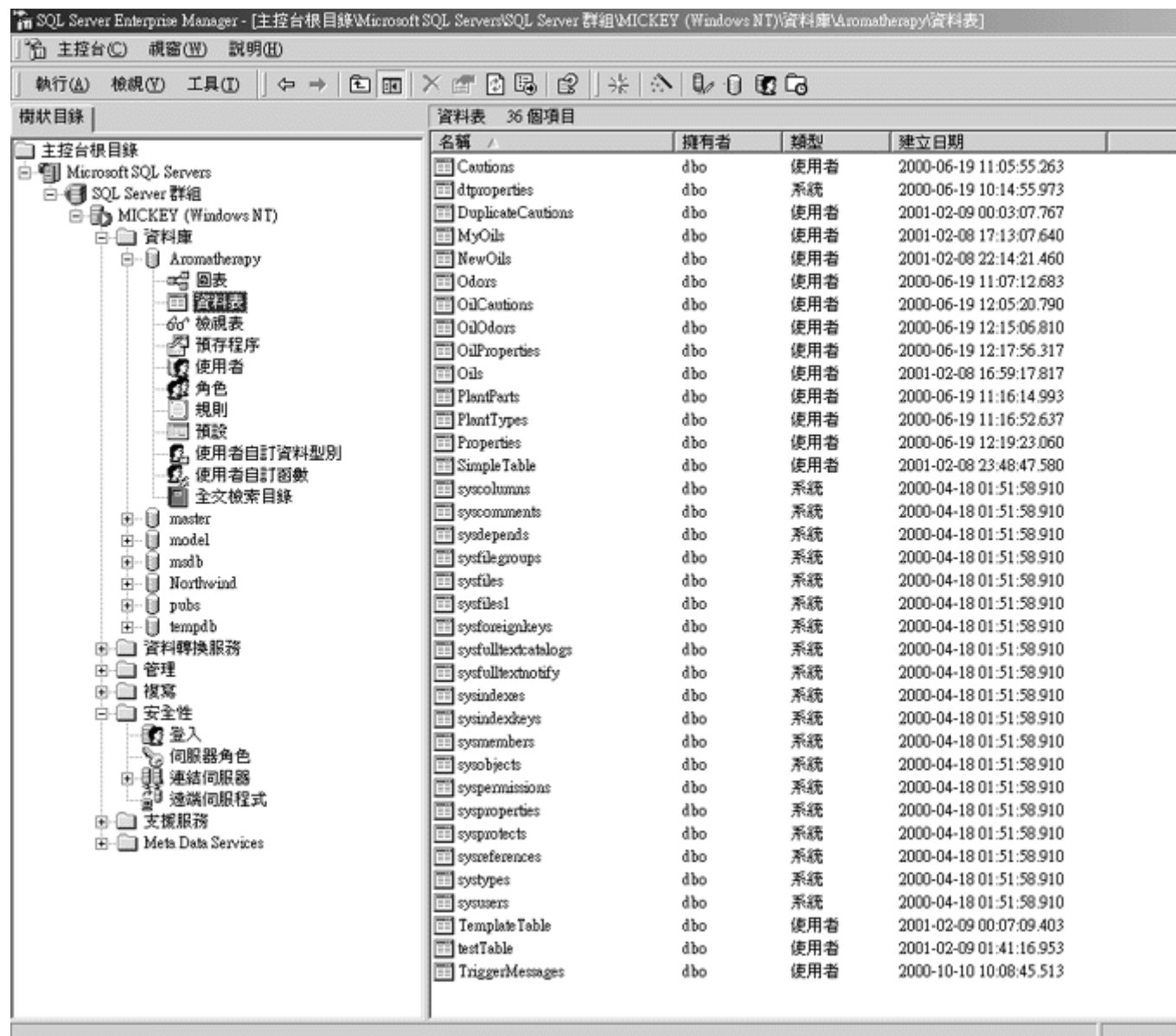
您可以使用 Enterprise Manager 的 **数据表设计师** 来建立及维护数据表。建立一个新的数据表的

第一个步骤是藉由开启 **数据表设计师** 来建立及命名数据表。

建立数据表

1. 按一下 Aromatherapy 数据库中的 [数据表](#) 数据夹。

SQL Server 会显示已存在的数据表清单。



- 在工具列上按一下 **新增** 按鈕。



新增按鈕

SQL Server 会开启 [数据表设计师](#) 。



3. 在工具列上按一下 [数据表和索引属性](#) 按钮。
-



数据表和索引属性按钮

SQL Server 会开启设计数据表的 [属性](#) 对话框。



4. 将数据表名称更改为 [Lesson 5](#)。



5. 按一下 [关闭](#) 按钮。

SQL Server 会关闭此 [属性](#) 对话框。

新增数据行至数据表中

虽然数据表可以拥有属于自己特定的属性，例如我们刚刚提到的数据表名称，但是数据表主要是数据行所组成。

新增数值数据型别的数据行至数据表中

- 1. 在 数据行名称 字段输入『MyNumber』，然后按一下 Tab 键。

SQL Server 预设 char 为资料型别。



- 2. 将数据型别更改为 decimal 。



資料行说明

可以在 Enterprise Manager 中为数据行加入描述的功能，这是新版 SQL Server 2000 所提供的新功能。它是一个称为 [延伸属性](#)（extended properties）新功能的一部份。Microsoft 已经建立

一些延伸属性，例如数据行描述（为标准服务器安装的一部份），并且您可以建立额外的延伸属性以储存特定应用程序或网站所需的关于数据库对象的信息。

每一个延伸属性都有一个使用者自订名称和值。如果延伸属性的值是使用 **sql_variant** 数据型别储存，那么它可以包含至多 **7500** 字节的数据，您可以使用预存程序来定义任何对象的多个延伸属性，关于预存程序的更多信息您可以参考 [第 28 章](#) 的说明。

精确度和小数点位数

一个数值的 **精确度**（Precision）是指小数点位数所能代表最大的值，包含小数点左边和右边的值相加；一个数值的 **小数点位数**（Scale）是指小数点右边的位数。举例来说，数值 **3647.311** 的精确度有 **7**（其数字的总合）和小数点位数有 **3**（小数点右边的数字总合）。

了解数值的精确度和小数点位数不会影响数据行的长度是很重要的。数据型别才会决定数据行的长度，而精确度和小数点位数则决定 **SQL Server** 如何解译储存在数据行中的数据。

新增一个识别数据行至数据表中

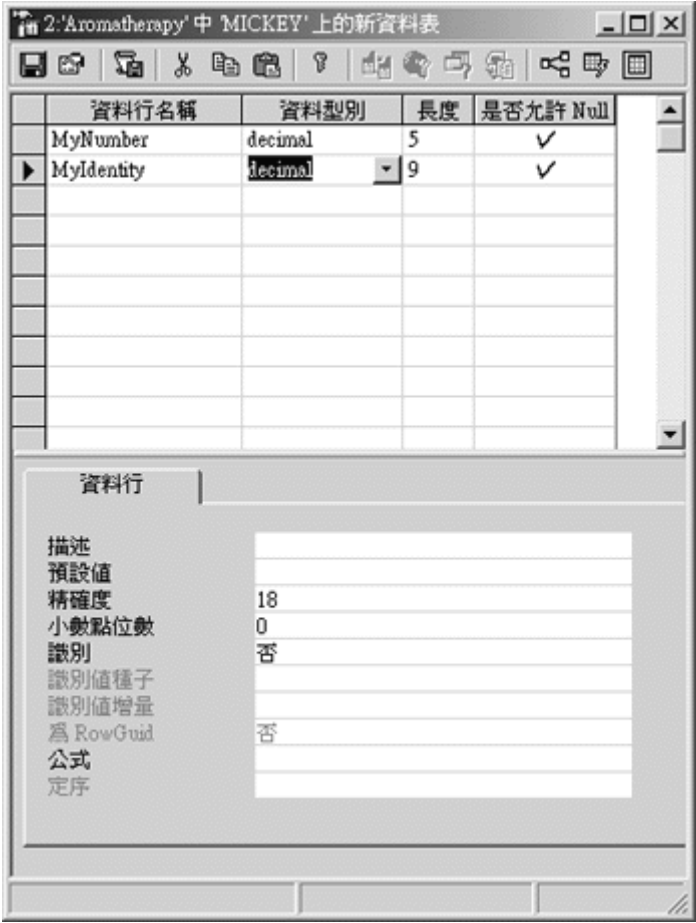
1. 在数据行名称中选取空白的储存格，并且输入『MyIdentity』，然后按一下 **Tab** 键。

SQL Server 预设 **char** 为资料型别。

[illegible]

2. 将数据型别更改为 **decimal**。

SQL Server 会将数据行长度更改为 **9**，并且启用 **精确度、小数点位数** 及 **识别** 等字段。



3. 将 是否允许 Null 字段内的设定清除。

Null

在关系型技术上，Null 值是一个特殊的值类别，它是用来指示该值不存在或遗失了。Null 值在使用上有一些问题，并且引起许多争论。

4. 在 **描述** 字段中输入『Sample Identity Column』。
5. 将 **识别** 字段中的值更改为 **是**（不是供复制使用）。

SQL Server 会将 **识别值种子** 以及 **识别值增量** 显示 **1** 的值。



识别值

当您在数据行中设定 **识别**（Identity）属性时，就是告诉 SQL Server 插入值至数据行中唯一识别的每一个资料列。选取这个数据行属性也就是决定数据行的数据类型。识别资料行只可以使用 int、smallint、tinyint 或者是 decimal。

当 SQL Server 将一数据列插入至有识别数据行的数据表时，识别数据行会自动的基于刚刚最后插入的值产生新的值（这些值是从识别值种子开始算起），并且识别值的增量是在数据表建立时就指定。

举例来说，假如有一个识别数据行是被定义为 **smallint** 的数据型别，并且将识别值种子设定为 50、识别值增量设定为 5 时，第一个所插入的值将会被指定为 50，而第二个值将被指定为 55，第三个值将被指定为 60，依此类推。

在一个数据表中只能有一个数据行具有 **识别** 属性设定。

新增一个 GUID 数据行至数据表中

1. 在数据行名称中选取空白的储存格，并输入『MyGUID』，然后按一下 **Tab** 键。

SQL Server 预设 **char** 为资料型别。

2: 'Aromatherapy' 中 'MICKEY' 上的新資料表

資料行名稱	資料型別	長度	是否允許 Null
MyNumber	decimal	5	✓
MyIdentity	decimal	9	
MyGUID	char	10	✓

資料行

描述	
預設值	
精確度	0
小數點位數	0
識別	否
識別值種子	
識別值增量	
為 RowGuid	否
公式	
定序	<資料庫預設值>

2. 將數據型別更改為 **uniqueidentifier** 。

SQL Server 會將此數據行的長度更改為 **16**，並且啟用 為 **RowGuid** 字段。



GUID

GUID 即为全域唯一识别码（Globally Unique Identifier），它是一个 16 位唯一的二进制数值—不会有其它的计算机会产生和它一样的值。 `uniqueidentifier` 的数据型别就是用来储存 GUID。

因为一个数据表中可以包含多个 GUID 数据行，所以 SQL Server 并不会以和识别数据行相同的方式来自动产生 GUID 值（GUID 所产生的识别码在任何地方都是唯一的）。当为 **RowGUID** 属

性被设定为 **是** 时，则 SQL Server 提供了 NEWID 函数来产生默认值，并且当数据列被插入时，
会传回一个新的 GUID 值。

新增一个日期数据行至数据表中

1. 在数据行名称中选取空白的储存格，并输入『MyDate』，然后按一下 **Tab** 键。

SQL Server 预设 **char** 为资料型别。

2: 'Aromatherapy' 中 'MICKEY' 上的新資料表

資料行名稱	資料型別	長度	是否允許 Null
MyNumber	decimal	5	✓
MyIdentity	decimal	9	
MyGUID	uniqueidentifie	16	✓
MyDate	char	10	✓

資料行

描述

預設值

精確度

小數點位數

識別

識別值種子

識別值增量

為 RowGuid

公式

定序

0

0

否

否

<資料庫預設值>

2. 將數據型別更改為 **datetime** 。

SQL Server 會將數據行長度更改為 **8** 。

2: 'Aromatherapy' 中 'MICKEY' 上的新資料表

資料行名稱	資料型別	長度	是否允許 Null
MyNumber	decimal	5	✓
MyIdentity	decimal	9	
MyGUID	uniqueidentifie	16	✓
MyDate	datetime	8	✓

資料行

描述

預設值

精確度

小數點位數

識別

識別值種子

識別值增量

為 RowGuid

公式

定序

0

0

否

否

- 在 **描述** 字段中输入『Sample Date Column』。

2: 'Aromatherapy' 中 'MICKEY' 上的新資料表

資料行名稱	資料型別	長度	是否允許 Null
MyNumber	decimal	5	✓
MyIdentity	decimal	9	
MyGUID	uniqueidentifie	16	✓
▶ MyDate	datetime	8	✓

資料行

描述	Sample Date Column
預設值	
精確度	0
小數點位數	0
識別	否
識別值種子	
識別值增量	
為 RowGuid	否
公式	
定序	

新增一个字符数据行至数据表中

1. 在数据行名称中选取空白的储存格，并输入『MyChar』，然后按一下 **Tab** 键。

SQL Server 预设 **char** 为资料型别。



字符数据类型

SQL Server 支持二种不同种类的字符数据行：固定长度和可变长度。每一种又可

延伸分为 **Unicode** 和非 **Unicode** 二种类型，并且也个别有三种不同的长度。

Unicode 是一种字符编码的方式，它可以支持双位组字符集。

假如一个资料行是被宣告为可变长度时（举例来说，针对非 **Unicode** 型态的 **varchar** 或 **text**；针对 **Unicode** 数据的 **nvarchar** 和 **ntext**），**SQL Server** 只会储存实际输入的数据字符。假如此数据行是宣告为固定长度时（例如针对非 **Unicode** 数据的 **char**；针对 **Unicode** 数据的 **nchar**），**SQL Server** 会以空格符来填满所输入的值。

举例来说，假如有一个数据行被宣告为长度为 10 的 **char** 数据型别，而您所输入的值是「hello」时，**SQL Server** 会以此值再加上五个空格符来当作是您所输入的实际值。

2. 变更数据行长度为 **25**。
3. 在 **描述** 字段中输入『Sample Character Column』。
4. 在 **默认值** 字段中输入『'Unknown'』（请确认有包含单引号字符串）。



默认值

默认值是一种自动插入的值，假如使用者没有直接输入值时，就会插入默认值到设有默认值的数据行中。

其实我们已经看过二种特殊种类的默认值：当您设定 **识别** 属性时，SQL Server 会自动为您在该数据行插入识别值；或者您设定 **为 RowGuid** 时，SQL Server 会自动在默认值字段加入 NEWID

函数。事实上，您可以针对任何的数据行指定默认值。默认值可以

为 'Unknown' 或 123、NEWID 或 GETDATE 函数，或者是数学表达式（例如 3+5）等内容。

储存及关闭数据表

1. 在 数据表设计师 的工具列上按一下 存盘 按钮。



存盘按钮

SQL Server 会将此数据表的定义内容储存起来。

2. 关闭此窗口。

管理数据表

虽然最好的状况是在您设计和建置数据库以后就趋于稳定状态，但是在实际的情况中却经常会有变更，这些变更都是您在设计阶段时预料不到的。幸运的是，**SQL Server** 让维护的工作可以很容易的执行。

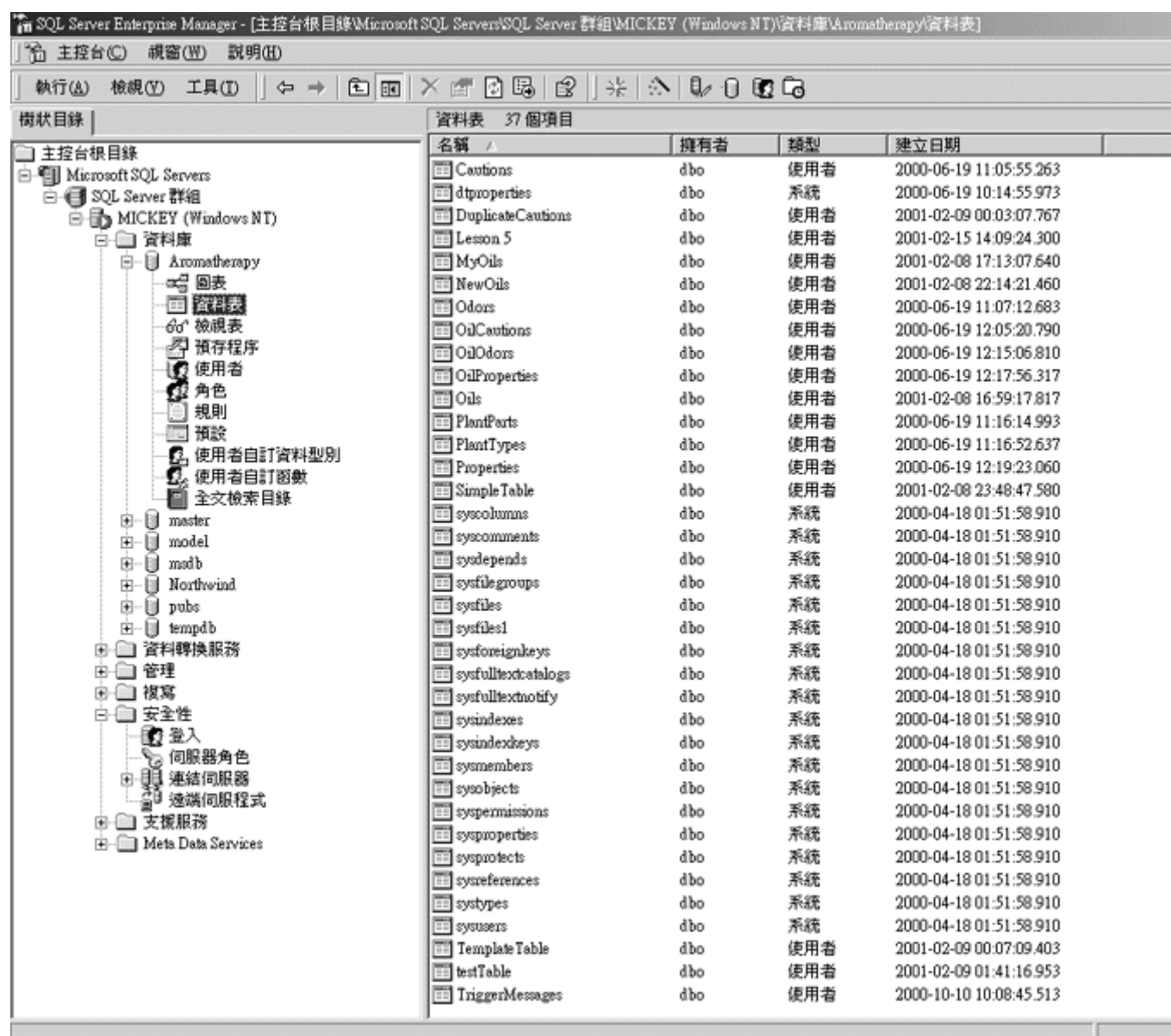
变更数据行

您可以藉由在详细数据窗格内的数据表上按右键，并且从快捷菜单中选取 **设计数据表** 以重新开启 **数据表设计师**。一旦 **数据表设计师** 被开启时，您就可以将数据行的属性进行修改、删除它们或者加入新的数据行。

重新命名数据行

1. 按一下 **Aromatherapy** 数据库内的 **数据表** 数据夹。

SQL Server 会在详细数据窗格中显示数据表对象。



- 在详细数据窗格内的 **Lesson 5** 数据表上按右键，并且选取 **设计数据表** 项目。

SQL Server 会开启 **数据表设计师**。

2.設計資料表 'Lesson 5' (在 'Aromatherapy' 中) 於 'MICKEY'

資料行名稱	資料型別	長度	是否允許 Null
MyNumber	decimal	5	✓
MyIdentity	decimal	9	
MyGUID	uniqueidentifie	16	✓
MyDate	datetime	8	✓
MyChar	char	25	✓

資料行

描述

預設值

精確度

小數點位數

識別

識別值種子

識別值增量

為 RowGuid

公式

定序

Sample Numeric Column

5

2

否

否

- 在数据行名称字段选取 **MyChar** 并且输入『MyCharacter』。

SQL Server 会更改此数据行的名称。

2.設計資料表 'Lesson 5' (在 'Aromatherapy' 中) 於 'MICKEY'

資料行名稱	資料型別	長度	是否允許 Null
MyNumber	decimal	5	✓
MyIdentity	decimal	9	
MyGUID	uniqueidentifie	16	✓
MyDate	datetime	8	✓
MyChar	char	25	✓
MyCharacter	char	25	✓

資料行

描述

預設值

精確度

小數點位數

識別

識別值種子

識別值增量

為 RowGuid

公式

定序

0

0

否

否

<資料庫預設值>

4. 在设计数据表的工具列上按一下 **存盘** 按钮以便将所更改的内容储存。

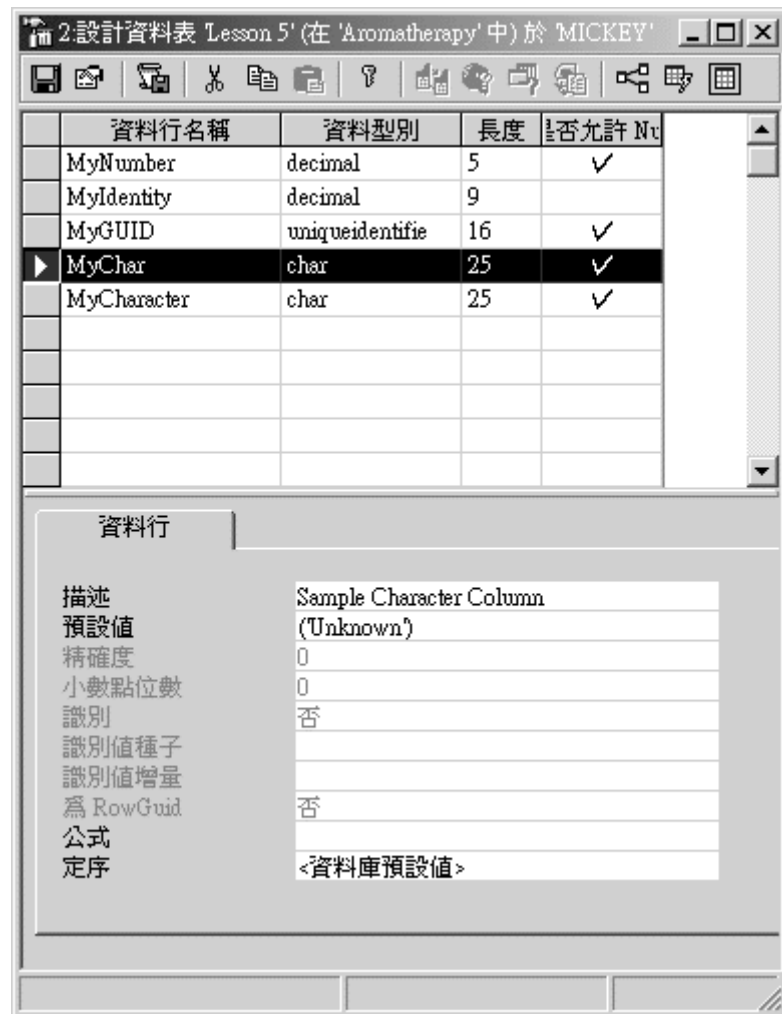
移除资料行

1. 选取 MyDate 数据行左边的灰色方块，以选取该数据行。



2. 按一下 **Delete** 键。

SQL Server 会将此数据行删除。



3. 按一下 **存盤** 按钮，以便将变更后的数据储存。

4. 将设计数据表窗口关闭。

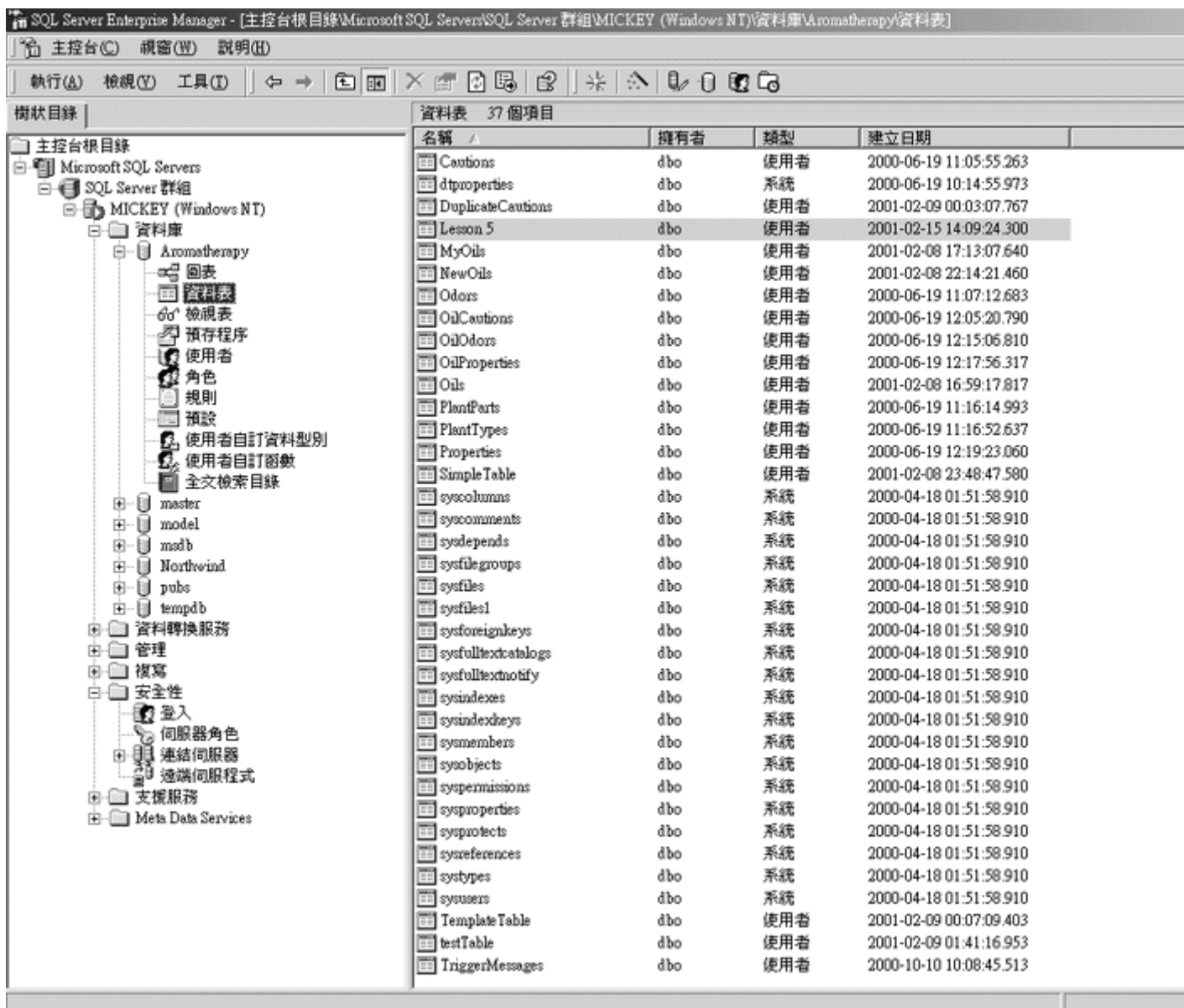
变更数据表

除了改变数据表中的数据行的定义之外，Enterprise Manager 可以让您为数据表更名或自数据库中删除数据表。

为数据表进行更名

1. 按一下 Aromatherapy 数据库内的 **数据表** 数据夹。

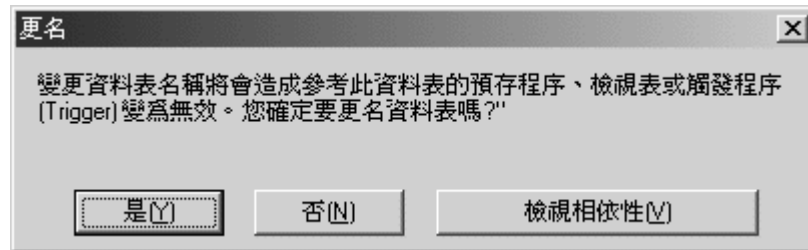
SQL Server 会在详细数据窗格中显示数据表对象。



- 在详细数据窗格中的 **Lesson 5** 资料表上按右键，并且选取 **重新命名**。
- 输入『New Lesson 5』，并且按一下 **Enter** 键。

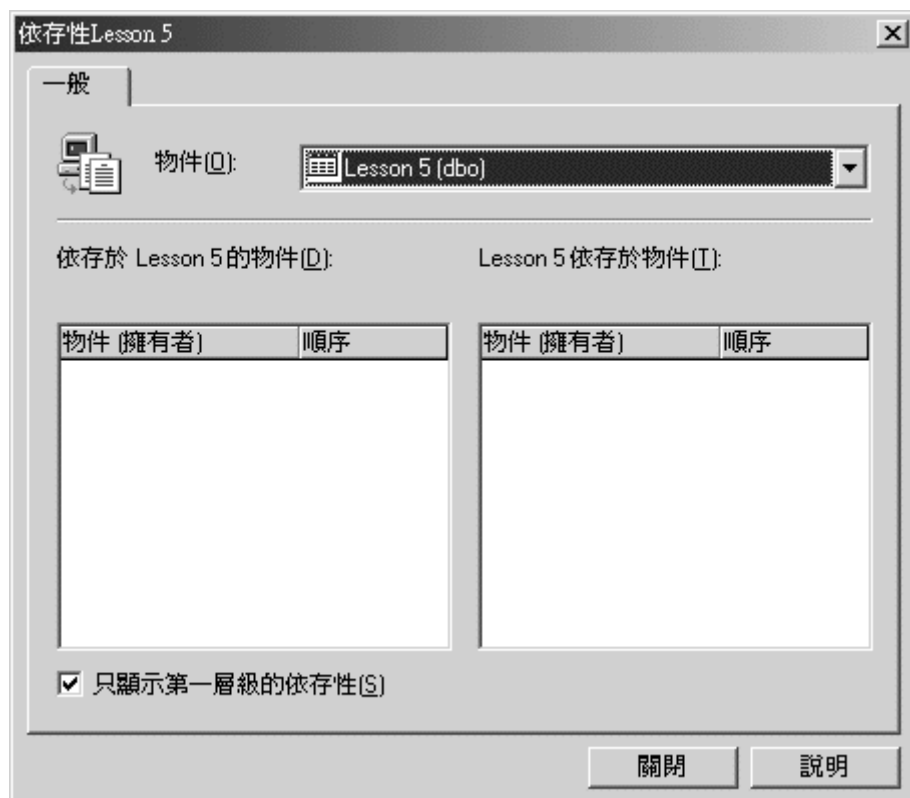
SQL Server 会显示一个 **更名** 对话框，并且警告您变更数据表将会造成参考此数

据表的对象变成无效。



4. 按一下 [檢視相依性](#) 按钮，以便显示任何可能会受影响的对象。

SQL Server 会开启一个 [依存性](#) 对话框。



5. 按一下 **关闭** 按钮以便离开此对话框。
6. 在 **更名** 对话框中按一下 **是** 按钮以确定要为数据表进行更名。

SQL Server 会显示一段已经成功地将资料表名称更名的讯息。



移除资料表

1. 在详细数据窗格中选取 **New Lesson 5** 资料表。
2. 按一下 **Delete** 键。

SQL Server 会显示一个 **卸除对象** 对话框。



提示

您可以按一下 [显示相依性](#) 按钮以便显示删除数据表所会影响的任何对象。

-
- 按一下 [卸除全部](#) 按钮。

SQL Server 会将此数据表从数据库中移除。

重要

当您要删除数据表时，此数据表以及所有它的数据将自数据库中永久地被删除。

唯一可以救回这些数据的方式就是数据库的备份。

本章总结

要执行的工 作	执行	按 钮
建立新的数据 表	按一下 Aromatherapy 数据库内的 数据表 数据夹，然后按一下 新增 按钮。	
新增数据行至 数据表中	在 数据表设计师 数据表中指定数据行属性	
重新命名数据 行	开启 数据表设计师 ，选取数据行名称，然后再输入新的数据行名称。	
移除资料行	在 数据表设计师 中选取该数据行，然后按一下 Delete 键。	
重新命名数据 表	在详细数据窗格内的数据表上按右钮，然后选取 重新命名 。输入新的数据表名称，然后在 更名 对话框中按一下 是 按钮。	
移除资料表	在详细数据窗格中选取数据表，然后按一下 Delete 键。	

6. 建立索引

在本章中，您将学习到：

- 使用建立索引精灵以建立索引。
- 建立主索引键索引。
- 建立简单索引。
- 建立复合索引。
- 重新命名索引。
- 变更索引中的数据行。
- 删除索引。

认识索引

在关系型数据库中，**索引**（index）是一个特殊的对象，这个对象允许数据库可以基于一或多个资料行的值快速的存取数据表中的数据列，就像一本书的索引可以基于特定的关键词以存取该书的特定内容。

Microsoft SQL Server 提供了二种不同型态的索引：**丛集索引**（clustered index）和 **非丛集索引**（nonclustered index）。丛集索引会决定数据表中数据列的实体储存顺序。非丛集索引是与数据库分离的对象，但是该对象会指向数据表中的特定数据列，但并不会决定数据列如何储存的次序。

一个索引可以引用在数据表中的一或多个资料行。只引用一个数据行的索引称为 **简单索引**（simple index）；引用多个数据行的索引称为 **复合索引**（composite index）

您除了可以自行定义索引以外，当您在数据表上定义主索引键时，SQL Server 将会自动地建立一个丛集索引，称为 **主索引键索引**（primary key index）。主索引键的功能是用于唯一识别的每一个数据列（您可以在一个或多个数据行上定义主索引键）。

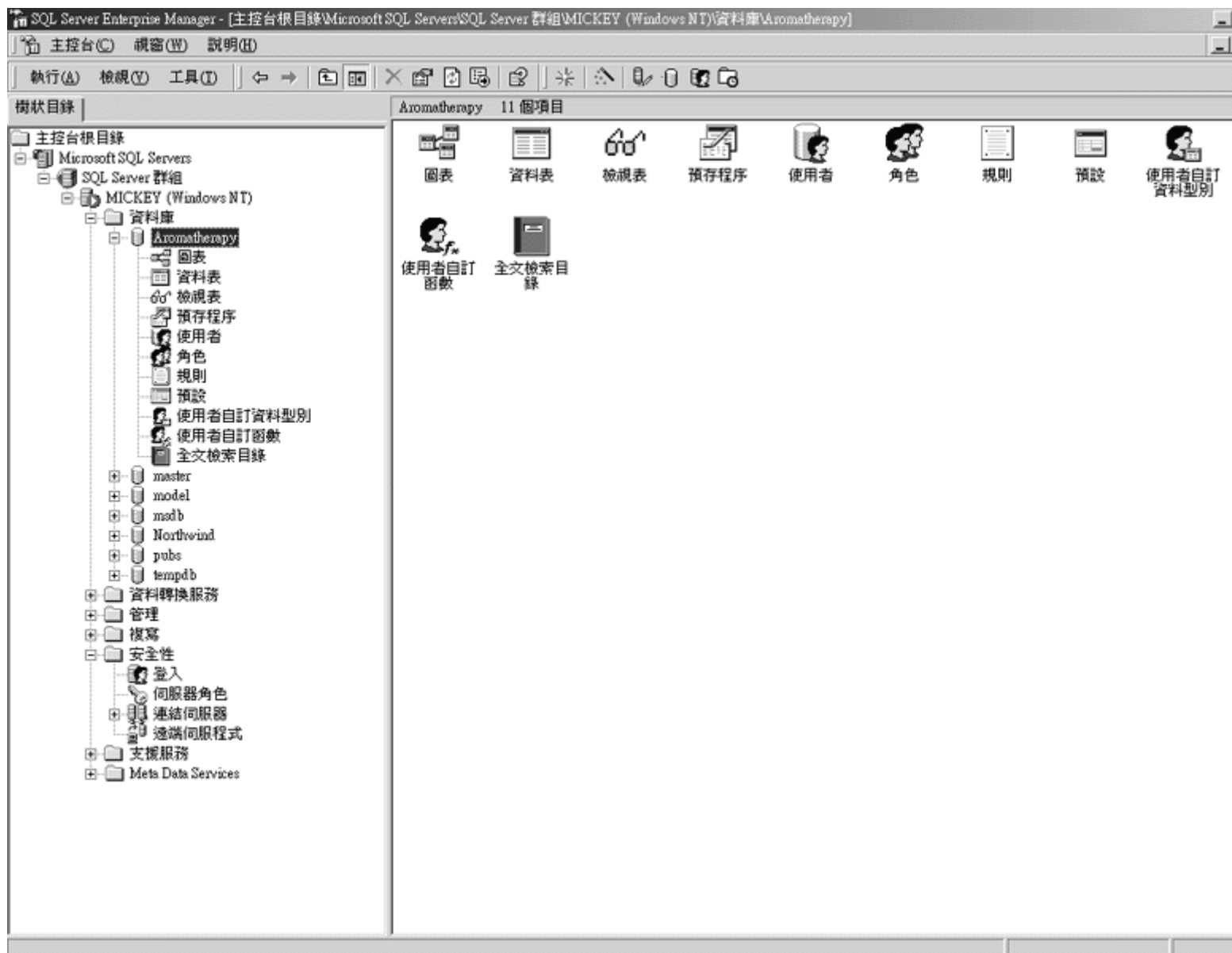
建立索引

在 Enterprise Manager 中，您可以使用 [建立索引精灵](#) 或透过 [数据表设计师](#) 建立索引。如果您使用 [资料表设计师](#)，您可以设定主索引键或在数据表的 [属性](#) 对话框中定义索引。

使用建立索引精灵来建立索引]

1. 按一下 Aromatherapy 数据库。

SQL Server 会在详细资料窗格中显示数据库对象。

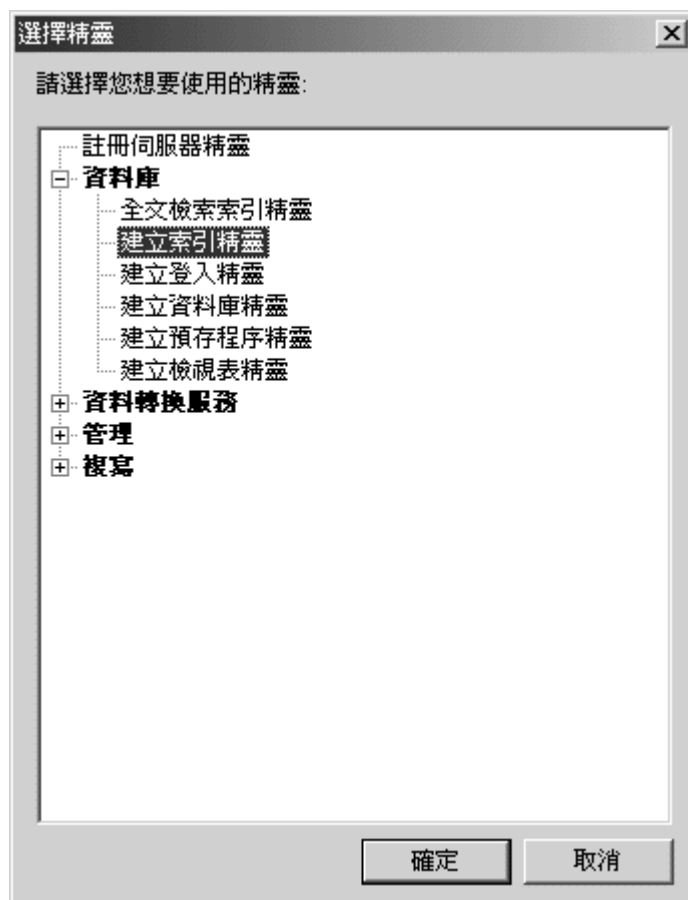


2. 在 Enterprise Manager 的工具列上按一下 [執行精靈](#) 按鈕。



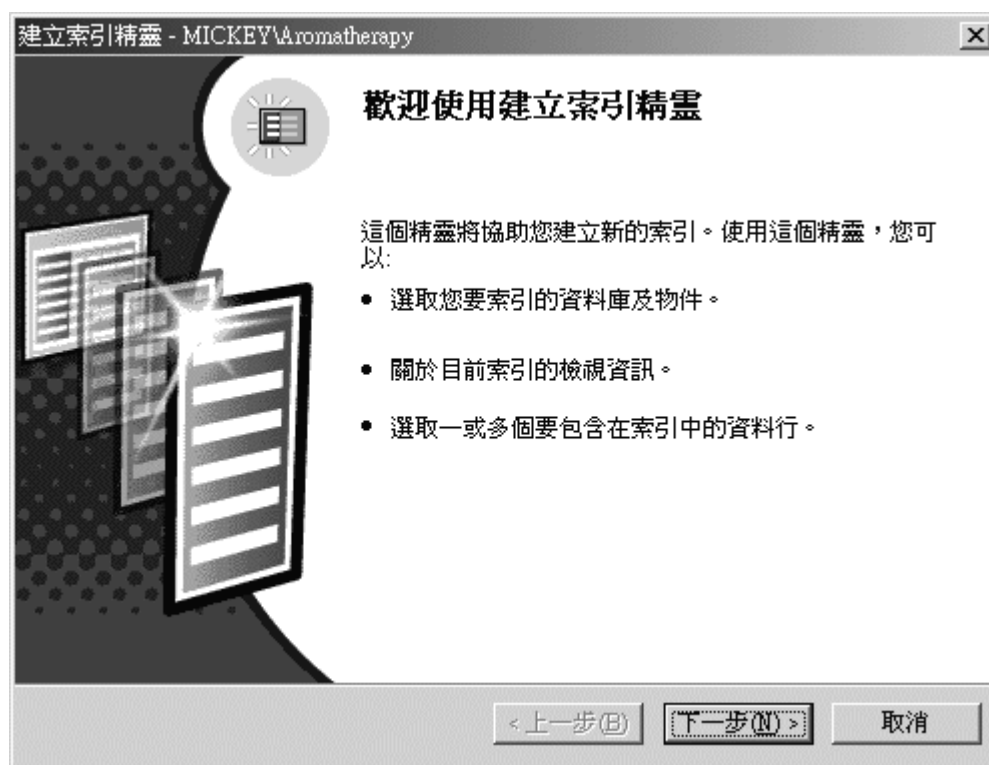
執行精灵按钮

SQL Server 会显示 [选择精灵](#) 对话框。



- 在 [数据库](#) 数据夹内选取 [建立索引精灵](#) 项目，然后按一下 [确定](#) 按钮。

SQL Server 会显示建立索引精灵的第一页画面。

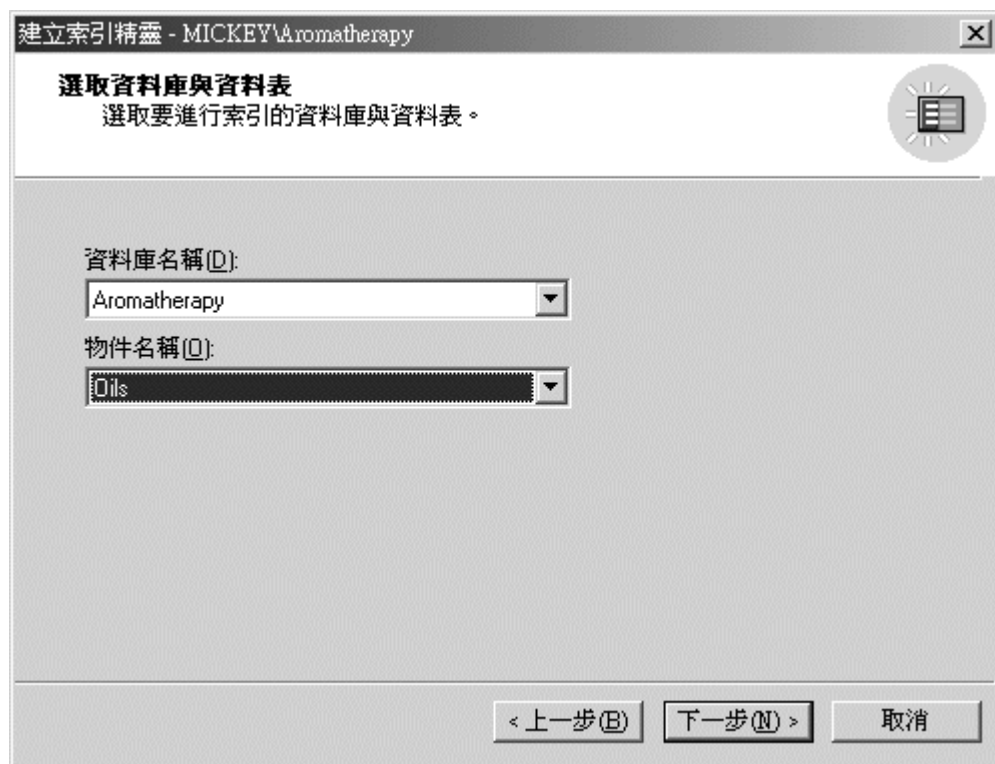


4. 按一下 **下一步** 按钮。

此精灵会个询问您要选取哪一个数据库来建立索引。



5. 確定在 **数据库名称** 上已经选取 Aromatherapy，并且在 **对象名称** 上选取 Oils。



6. 按一下 **下一步** 按钮。

精灵会显示目前数据表内已经存在的索引（这些奇怪的索引名称是由 SQL Server 所建立以强制其关联性，我们将在下一章中详细说明关联性）。

建立索引精靈 - MICKEY\Aromatherapy

選取資料行
選取一或多個要包含在索引中的資料行。

資料行名稱	資料型別	長度	包含於索...	排序(降幕)
OilID	int	4	<input type="checkbox"/>	<input type="checkbox"/>
OilName	nvarchar	50	<input type="checkbox"/>	<input type="checkbox"/>
LatinName	nvarchar	50	<input type="checkbox"/>	<input type="checkbox"/>
PlantTypeID	int	4	<input type="checkbox"/>	<input type="checkbox"/>
PlantPartID	int	4	<input type="checkbox"/>	<input type="checkbox"/>
Sample	char	10	<input type="checkbox"/>	<input type="checkbox"/>
Description	char	30	<input type="checkbox"/>	<input type="checkbox"/>

< 上一步(B) 下一步(N) > 取消

- 在 **包含于索引** 的字段中选取 **OilName** 和 **LatinName** 数据行以将它们包含在索引中。

建立索引精靈 - MICKEY\Aromatherapy

選取資料行
選取一或多個要包含在索引中的資料行。

資料行名稱	資料型別	長度	包含於索...	排序(降幕)
OilID	int	4	<input type="checkbox"/>	<input type="checkbox"/>
OilName	nvarchar	50	<input checked="" type="checkbox"/>	<input type="checkbox"/>
LatinName	nvarchar	50	<input checked="" type="checkbox"/>	<input type="checkbox"/>
PlantTypeID	int	4	<input type="checkbox"/>	<input type="checkbox"/>
PlantPartID	int	4	<input type="checkbox"/>	<input type="checkbox"/>
Sample	char	10	<input type="checkbox"/>	<input type="checkbox"/>
Description	char	30	<input type="checkbox"/>	<input type="checkbox"/>

< 上一步(B) 下一步(N) > 取消

9. 按一下 **下一步** 按钮。

此精灵会显示一个要求您指定索引属性的画面。



10. 采用默认值并且按一下 [下一步](#) 按钮。

此精灵会要求您输入索引的名称及所选取的内容，在此我们采用预设的索引名称。



11. 选取 **LatinName** 字段，并且按一下 **上移** 按钮以便更改在索引中数据行的顺序。



12. 按一下 **完成** 按钮。

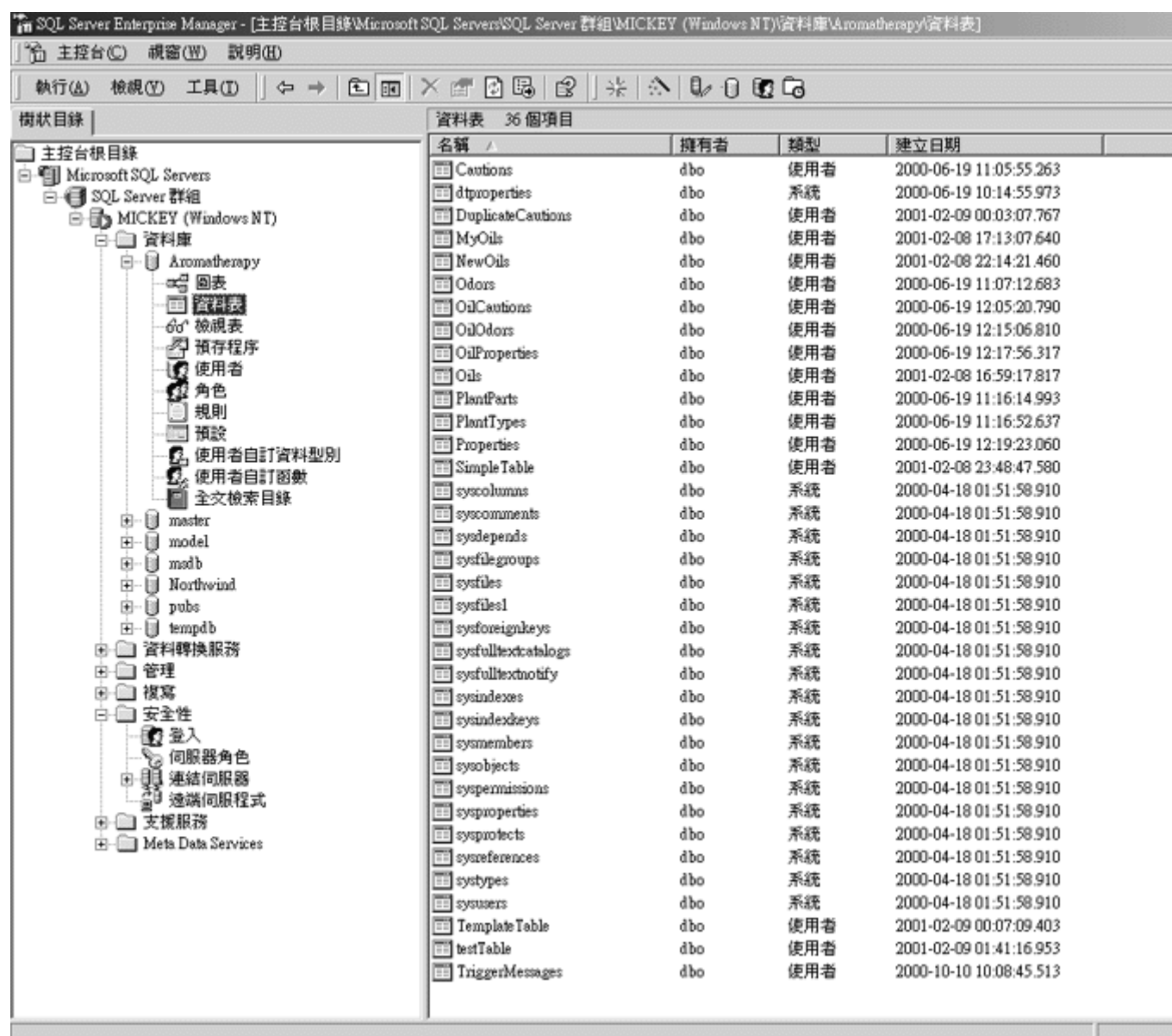
此精灵会显示一个告知您已经建立索引的讯息。



建立主索引键索引

1. 按一下 Aromatherapy 数据库内的 **数据表** 数据夹。

SQL Server 会自详细数据窗格中显示数据表对象。



- 在详细数据窗格内 **PlantTypes** 数据表名称上按右键，并在快捷菜单中选取 **设计**

数据表。

SQL Server 会开启 **设计数据表** 窗口。



- 按一下 **PlantTypeID** 数据行名称左侧的方格以选取 **PlantTypeID** 数据行。

SQL Server 会选取此数据行。



4. 在设计数据表的工具列上按一下 **设定主索引键** 按钮。

SQL Server 会将此数据行当作是主索引键。



设定主索引键按钮



5. 按一下 **存盘** 按钮以便将更改的数据储存，并且关闭此窗口。



存盘按钮

建立一个简单的索引

1. 在详细数据窗格内 **Oils** 数据表名称上按右键，并在快捷菜单中选取 [设计数据表](#)。

SQL Server 会开启 [设计数据表](#) 窗口。



2. 按一下 [管理索引/索引鍵](#) 按鈕。



設定索引/索引鍵按鈕

SQL Server 会开启此数据表 [属性](#) 对话框内的 [索引/索引键](#) 卷标页。

屬性

資料表

關聯性

索引/索引鍵

檢查條件約束

資料表名稱:

Oils

選定的索引(S):

PK_Oils

類型:

主索引鍵

新增(N)

刪除(D)

索引名稱(I):

PK_Oils

資料行名稱	順序
OilID	遞增

索引檔群組(G):

PRIMARY

☒ 建立成唯一 - UNIQUE(U)

☒ 條件約束(Q)

☐ 索引(X)

☐ 忽略重複的索引鍵(K)

填滿因數(F):

0

%

☐ 索引亦使用(P)

☒ 建立成叢集 - CLUSTERED(C)

☐ 不會自動重新計算統計資料(M)

關閉

說明

提示

您也可以按一下工具列上的 [數據表和索引屬性](#) 按鈕來開啟數據表的 [屬性](#) 對話框，然後再選取 [索引/索引鍵](#) 卷標頁即可。

3. 按一下 **新增** 按钮。



数据表和索引属性按钮

SQL Server 会显示一个建议您使用 **IX_Oils** 来作为索引名称，并且将 **OIID** 当作是索引使用的数据行。

屬性

資料表 | 關聯性 | 索引/索引鍵 | 檢查條件約束

資料表名稱: Oils

選定的索引(S): IX_Oils

類型: 索引 新增(N) 刪除(D)

索引名稱(I): IX_Oils

資料行名稱	順序
OilID	遞增

索引檔群組(G): PRIMARY

☐ 建立成唯一 - UNIQUE(U)
☐ 條件約束(C)
☐ 索引(X) ☐ 忽略重複的索引鍵(K)

☐ 建立成叢集 - CLUSTERED(C)
☐ 不會自動重新計算統計資料(M)

填滿因數(F): 0 %
☐ 索引亦使用(P)

關閉 說明

- 將索引名稱更改為 **IX_OilName**。

屬性

資料表 | 關聯性 | 索引/索引鍵 | 檢查條件約束

資料表名稱: Oils

選定的索引(S): IX_Oils

類型: 索引 新增(N) 刪除(D)

索引名稱(I): IX_OilName

資料行名稱	順序
OilID	遞增

索引檔群組(G): PRIMARY

☐ 建立成唯一 - UNIQUE(U)
☐ 條件約束(C)
☐ 索引(X) ☐ 忽略重複的索引鍵(K)

☐ 建立成叢集 - CLUSTERED(C)
☐ 不會自動重新計算統計資料(M)

填滿因數(F): 0 %
☐ 索引亦使用(P)

關閉 說明

5. 自下拉式方块中选取 OilName 当作是索引使用的数据行名称。

屬性

資料表 | 關聯性 | 索引/索引鍵 | 檢查條件約束

資料表名稱: Oils

選定的索引(S): IX_OilName

類型: 索引 新增(N) 刪除(D)

索引名稱(I): IX_OilName

資料行名稱	順序
OilName	遞增

索引檔群組(G): PRIMARY

☐ 建立成唯一 - UNIQUE(U) 填滿因數(F): 0 %
☐ 條件約束(O) ☐ 索引亦使用(P)
☐ 索引(X) ☐ 忽略重複的索引鍵(K)

☐ 建立成叢集 - CLUSTERED(C)
☐ 不會自動重新計算統計資料(M)

關閉 說明

6. 选取 **建立成唯一 - UNIQUE** 复选框以建立唯一的索引。

屬性

資料表 | 關聯性 | 索引/索引鍵 | 檢查條件約束

資料表名稱: Oils

選定的索引(S): IX_OilName

類型: 唯一 新增(N) 刪除(D)

索引名稱(I): IX_OilName

資料行名稱	順序
OilName	遞增

索引檔群組(G): PRIMARY

☒ 建立成唯一 - UNIQUE(U)
☒ 條件約束(Q)
☐ 索引(X) ☐ 忽略重複的索引鍵(K)

填滿因數(F): 0 %
☐ 索引亦使用(P)

☐ 建立成叢集 - CLUSTERED(C)
☐ 不會自動重新計算統計資料(M)

關閉 說明

7. 选取 **索引** 項目。

屬性

資料表 | 關聯性 | 索引/索引鍵 | 檢查條件約束

資料表名稱: Oils

選定的索引(S): IX_OilName

類型: 索引 新增(N) 刪除(D)

索引名稱(I): IX_OilName

資料行名稱	順序
OilName	遞增

索引檔群組(G): PRIMARY

☒ 建立成唯一 - UNIQUE(U) 填滿因數(F): 0 %

☐ 條件約束(Q) ☐ 索引亦使用(P)

☒ 索引(X) ☐ 忽略重複的索引鍵(K)

☐ 建立成叢集 - CLUSTERED(C)

☐ 不會自動重新計算統計資料(M)

關閉 說明

唯一索引

唯一索引（unique index）会确认索引中所包含的数据行中所有的数据列都是唯一的。主索引键索引通常是唯一的，假如您喜欢，您也可以加入其它的唯一索引。

-
- 按一下 **关闭** 按钮。

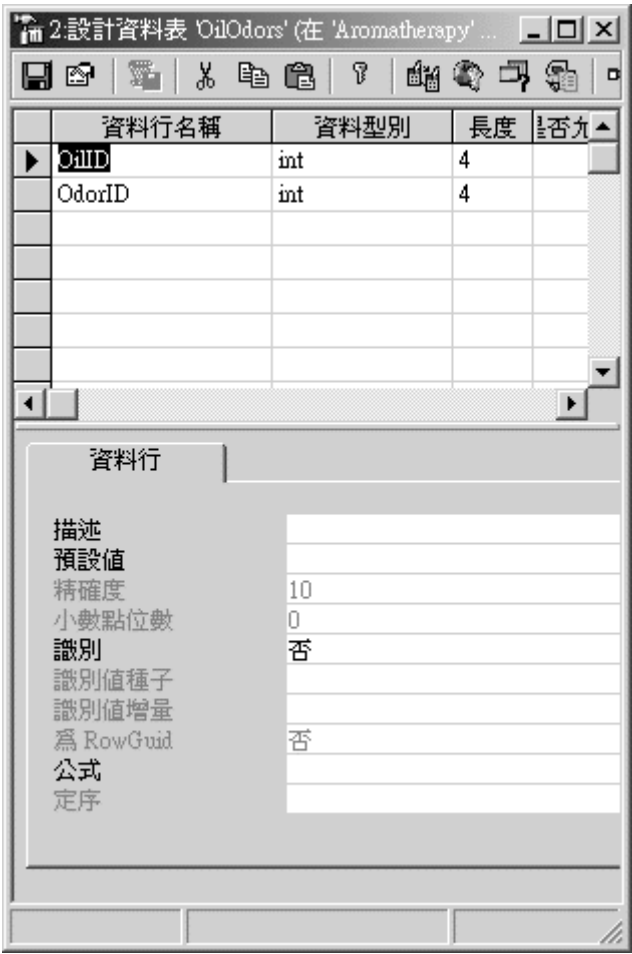
SQL Server 会将此对话框窗口关闭。

9. 在设计数据表的工具列上按一下 [存盘](#) 按钮以便储存更改后的数据，然后将设计数据表窗口关闭。

建立复合索引

1. 在详细数据窗格内 OilOdors 数据表名称上按右钮，并在快捷菜单中选取 [设计数据表](#)。

SQL Server 会开启 [设计数据表](#) 窗口。



2. 按一下 [管理索引/索引鍵](#) 按鈕。



管理索引/索引鍵按鈕

SQL Server 会开启设计数据表的 [属性](#) 对话框内的 [索引/索引键](#) 卷标页。

屬性

資料表 | 關聯性 | 索引/索引鍵 | 檢查條件約束

資料表名稱: OilOdors

選定的索引(S):

類型:

索引名稱(I):

索引檔群組(G):

☐ 建立成唯一 - UNIQUE(U)
 ☐ 條件約束(Q)
 ☐ 索引(X)
 ☐ 忽略重複的索引鍵(K)
 填滿因數(F): %
 ☐ 索引亦使用(P)

☐ 建立成叢集 - CLUSTERED(C)
 ☐ 不會自動重新計算統計資料(M)

關閉 說明

- 按一下 **新增** 按鈕。

SQL Server 會顯示一個建議您使用 **IX_OilOdors** 來作為索引名稱，並且將 OilID

當作是索引使用的數據行。

屬性

資料表 關聯性 索引/索引鍵 檢查條件約束

資料表名稱: OilOdors

選定的索引(S): IX_OilOdors

類型: 索引 新增(N) 刪除(D)

索引名稱(I): IX_OilOdors

資料行名稱	順序
OdorID	遞增

索引檔群組(G): PRIMARY

☐ 建立成唯一 - UNIQUE(U)
☐ 條件約束(C)
☐ 索引(X) ☐ 忽略重複的索引鍵(I)

☐ 建立成叢集 - CLUSTERED(C)
☐ 不會自動重新計算統計資料(M)

填滿因數(F): 0 %
☐ 索引亦使用(P)

關閉 說明

- 在 **数据行名称** 字段中，自下拉式方块中选取 **OdorID** 数据行以将它加入索引中。

屬性

資料表 | 關聯性 | 索引/索引鍵 | 檢查條件約束

資料表名稱: OilOdors

選定的索引(S): IX_OilOdors

類型: 索引 新增(N) 刪除(D)

索引名稱(I): IX_OilOdors

資料行名稱	順序
OilID	遞增
OdorID	遞增

索引檔群組(G): PRIMARY

☐ 建立成唯一 - UNIQUE(U)
☐ 條件約束(O)
☐ 索引(X) ☐ 忽略重複的索引鍵(K)

☐ 建立成叢集 - CLUSTERED(C)
☐ 不會自動重新計算統計資料(M)

填滿因數(F): 0 %
☐ 索引亦使用(P)

關閉 說明

- 按一下 [關閉](#) 按鈕。

SQL Server 会将此对话框窗口关闭。

- 在设计数据表工具列上按一下 [存盘](#) 按钮，以便储存所改变的数据，然后将此设计

数据表窗口关闭。

维护索引

就像其它的数据库对象一样，您有时会需要将您所建立的索引进行修改。索引和它的属性的维护都是在 [数据表设计师属性](#) 对话框中。

变更索引

如同您可以藉由开启 [数据表设计师](#) 来变更数据行属性一样，您也可以使用相同的方式来变更索引的属性。

重新命名索引

1. 在详细数据窗格中的 **Oils** 资料表名称上按右钮，并在快捷菜单中选取 [设计数据表](#) 项目。

SQL Server 会开启 [数据表设计师](#) 窗口。



2. 按一下 [管理索引/索引鍵](#) 按鈕。



設定索引/索引鍵按鈕

SQL Server 會開啟設計數據表的 [屬性](#) 對話框內的 [索引/索引鍵](#) 卷標頁。

屬性

資料表 | 關聯性 | **索引/索引鍵** | 檢查條件約束

資料表名稱: Oils

選定的索引(S): PK_Oils

類型: 主索引鍵 新增(N) 刪除(D)

索引名稱(I): PK_Oils

資料行名稱	順序
OilID	遞增

索引檔群組(G): PRIMARY

☒ 建立成唯一 - UNIQUE(U)
 ☒ 條件約束(Q)
 ☐ 索引(X)
 ☐ 忽略重複的索引鍵(K)

☒ 建立成叢集 - CLUSTERED(C)
 ☐ 不會自動重新計算統計資料(M)

填滿因數(F): 0 %
 ☐ 索引亦使用(P)

關閉 說明

- 在 **選定的索引** 下拉式方块中选取 **IX_OilName**。

SQL Server 会显示此索引的内容。

屬性

資料表 | 關聯性 | 索引/索引鍵 | 檢查條件約束

資料表名稱: Oils

選定的索引(S): IX_OilName

類型: 索引 新增(N) 刪除(D)

索引名稱(I): IX_OilName

資料行名稱	順序
OilName	遞增

索引檔群組(G): PRIMARY

☒ 建立成唯一 - UNIQUE(U) 填滿因數(F): 0 %
☐ 條件約束(Q) ☐ 索引亦使用(P)
☒ 索引(X) ☐ 忽略重複的索引鍵(K)
☐ 建立成叢集 - CLUSTERED(C)
☐ 不會自動重新計算統計資料(M)

關閉 說明

4. 將此索引名稱更改為 **IX_Name**。

屬性

資料表 | 關聯性 | 索引/索引鍵 | 檢查條件約束

資料表名稱: Oils

選定的索引(S): IX_OilName

類型: 索引 新增(N) 刪除(D)

索引名稱(I): IX_Name

資料行名稱	順序
OilName	遞增

索引檔群組(G): PRIMARY

☒ 建立成唯一 - UNIQUE(U) 填滿因數(F): 0 %
☐ 條件約束(Q) ☐ 索引亦使用(P)
☒ 索引(X) ☐ 忽略重複的索引鍵(K)
☐ 建立成叢集 - CLUSTERED(C)
☐ 不會自動重新計算統計資料(M)

關閉 說明

- 按一下 [關閉](#) 按鈕。

SQL Server 会将此对话框关闭。

- 在设计数据表工具列上按一下 [存盘](#) 按钮，以储存所改变的索引名称。

变更索引中的数据行

- 如果 Oils 数据表的 [数据表设计师](#) 窗口没有开启时,请您在详细数据窗格内的 Oils

资料表名称上按右钮，并在快捷菜单中选取 [设计数据表](#)。

SQL Server 会开启 [设计数据表](#) 窗口。

資料行名稱	資料型別	長度	是否允許 Null
OilID	int	4	
OilName	nvarchar	50	
LatinName	nvarchar	50	✓
PlantTypeID	int	4	✓
PlantPartID	int	4	✓
Sample	char	10	✓
Description	char	30	✓

資料行	
描述	
預設值	
精確度	10
小數點位數	0
識別	是
識別值種子	1
識別值增量	1
為 RowGuid	否
公式	
定序	

2. 按一下 [管理索引/索引键](#) 按钮。



SQL Server 会开启设计数据表的 [属性](#) 对话框内的 [索引/索引键](#) 卷标页。

屬性

資料表 | 關聯性 | 索引/索引鍵 | 檢查條件約束

資料表名稱: Oils

選定的索引(S): PK_Oils

類型: 主索引鍵 新增(N) 刪除(D)

索引名稱(I): PK_Oils

資料行名稱	順序
OilID	遞增

索引檔群組(G): PRIMARY

☒ 建立成唯一 - UNIQUE(U) ☐ 條件約束(O) ☐ 索引(X) ☐ 忽略重複的索引鍵(K)

填滿因數(F): 0 % ☐ 索引亦使用(E)

☒ 建立成叢集 - CLUSTERED(C) ☐ 不會自動重新計算統計資料(M)

關閉 說明

3. 从 [选定的索引](#) 下拉式方块中选取 [Oils_Index_1](#) 。

SQL Server 会显示此索引的属性。

屬性

資料表 | 關聯性 | 索引/索引鍵 | 檢查條件約束

資料表名稱: Oils

選定的索引(S): Oils_Index_1

類型: 索引 新增(N) 刪除(D)

索引名稱(I): Oils_Index_1

資料行名稱	順序
LatinName	遞增
OilName	遞增

索引檔群組(G): PRIMARY

☐ 建立成唯一 - UNIQUE(U) 填滿因數(F): 0 %
☐ 條件約束(Q) ☐ 索引亦使用(P)
☐ 索引(X) ☐ 忽略重複的索引鍵(K)

☐ 建立成叢集 - CLUSTERED(C)
☐ 不會自動重新計算統計資料(M)

關閉 說明

- 在 **数据行名称** 字段中选取 **LatinName** 儲存格，并且将索引数据行更改为 **PlantTypeID**。

屬性

資料表 | 關聯性 | 索引/索引鍵 | 檢查條件約束

資料表名稱: Oils

選定的索引(S): Oils_Index_1

類型: 索引 新增(N) 刪除(D)

索引名稱(I): Oils_Index_1

資料行名稱	順序
PlantTypeID	遞增
OilName	遞增

索引檔群組(G): PRIMARY

☐ 建立成唯一 - UNIQUE(U) 填滿因數(F): 0 %

☐ 條件約束(O) ☐ 索引亦使用(P)

☐ 索引(X) ☐ 忽略重複的索引鍵(K)

☐ 建立成叢集 - CLUSTERED(C)

☐ 不會自動重新計算統計資料(M)

關閉 說明

5. 按一下 [关闭](#) 按钮。

SQL Server 会将此对话框关闭。

6. 在设计数据表工具列上按一下 [存盘](#) 按钮，以便储存所改变的索引名称。

移除索引

也许您可能需要从数据表中移除索引，因为您已经不再需要它了；或者是当底层数据已经被更新，维护此索引所花费的时间会比重新建立索引的时间还要久。就像所有其它的维护工作一样，您可以藉由使用 [数据表设计师](#) 的属性对话框内的 [索引/索引键](#) 卷标页来删除索引。

删除索引

1. 如果 Oils 数据表的 [数据表设计师](#) 窗口没有开启时，请您在详细数据窗格内的 Oils 资料表名称上按右钮，并在快捷菜单中选取 [设计数据表](#)。

SQL Server 会开启 [设计数据表](#) 窗口。



2. 按一下 [管理索引/索引鍵](#) 按鈕。



設定索引/索引鍵按鈕

SQL Server 會開啟設計數據表的屬性對話框內的 [索引/索引鍵](#) 卷標頁。

屬性

資料表 | 關聯性 | 索引/索引鍵 | 檢查條件約束

資料表名稱: Oils

選定的索引(S): PK_Oils

類型: 主索引鍵 新增(N) 刪除(D)

索引名稱(I): PK_Oils

資料行名稱	順序
OilID	遞增

索引檔群組(G): PRIMARY

☒ 建立成唯一 - UNIQUE(U)
 ☒ 條件約束(C)
 ☐ 索引(X)
 ☐ 忽略重複的索引鍵(I)

☒ 建立成叢集 - CLUSTERED(C)
 ☐ 不會自動重新計算統計資料(M)

填滿因數(F): 0 %

☐ 索引亦使用(P)

關閉 說明

3. 从 选定的索引 下拉式方块中选取 **IX_Name** 。

SQL Server 会显示此索引的内容。



- 按一下 [刪除](#) 按钮。

SQL Server 会将此索引删除。

- 按一下 [关闭](#) 按钮。

SQL Server 会将此对话框关闭。

- 在设计数据表工具列上按一下 [存盘](#) 按钮，以便将所改变的索引名称储存，并且将此设计数据表窗口关闭。

本章总结

要执行的 工作	执行	按 钮
建立主索引键索引	针对某数据表开启 数据表设计师 ，选取一或多个数据行，然后按一下 设定主索引键 按钮。	
建立索引	针对某数据表开启 数据表设计师 ，按一下 管理索引/索引键 按钮，以便开启此数据表 属性 对话框内的 索引/索引键 卷标页，按一下 新增 按钮以设定索引属性。	
变更索引	针对某数据表开启 数据表设计师 ，按一下 管理索引/索引键 按钮，以便开启此数据表 属性 对话框内的 索引/索引键 卷标页，然后设定索引属性。	
移除索引	针对某数据表开启 数据表设计师 ，按一下 管理索引/索引键 按钮，以便开启此数据表 属性 对话框内的 索引/索引键 卷标页，选取要移除的索引，然后按一下 删除 按钮。	

7. 建立关联性

在本章中，您将学习到：

- 建立关联性。
- 变更关联性。
- 重新命名关联性。
- 删除关联性。

认识关联性

大多数的数据库会试图模型化实际情况中的某些部分，这些部分即为大家所熟知的 **Problem Space**。站在逻辑层的观点，在 **Problem Space** 中的对象即为 **实体**（Entity），而实体之间的关系称为 **关联性**（Relationship）。站在实体层的观点，Microsoft SQL Server 会将实体视为数据表，并且将关联性视为 **外部索引键条件约束**（Foreign Key Constraints）。

关系型模型

大多数的人相信关系型数据库会被称之为「关系型」，是因为数据表之间建立着关联性。事实上，这种描述方式是来自于「Relation」这个术语，这个术语为 Dr. E. F. Codd（这位大师是在 1960 年代末期最初关系型模型的发明人）在 SQL Server 中采用这个观念来说明实体建置为对象后就是一个数据表。

在逻辑上来说，有三种关联性型态：**一对一**（one-to-one），是指数据表内的每一数据列与其它的数据表有着 0 或 1 列数据列的关联性；**一对多**（one-to-many），是指数据表内的每一数据列与其它的数据表有着 0、1 或多列数据列的关联性；**多对多**（many-to-many），是指在第一个资料表内的每一资料列与第二个资料表有着 0、1 或多列数据列的关联性，并且在第二个数据表内的每一资料列与第一个资料表有着 0、1 或多列数据列的关联性。

一对一关联性是比较罕见的关联性，它们大部分是用于当属性的集合只套用至实体中的少数个体。举例来说，公司中的垒球队（属性集合）只是由公司中的少数员工（少数个体）组成。数据库设计者也许会将垒球队的所有属性放在一个分开的数据表中，然后再将这个数据表与公司的员工数据表之间建立一对一的关联性。

一对多关联性比较常使用到。在我们的范例数据库中，PlantParts 数据表和 Oils 数据表之间就存在着 0、1 或多列数据列的关联性。

多对多关联性也是经常使用到的。在范例数据库中，Properties 数据表和 Oils 数据表之间就存在着多对多的关联性。假如每一种 Oil 可以拥有多种 Properties 时，那么任何一种 Properties 也可以指派给许多的 Oil 使用。

SQL Server，就像其它的关系型数据库引擎一样，可以直接采用一对一以及一对多的关联性。但是在解决多对多的关联性方面，SQL Server 采用一种特殊类型的数据表—[联合数据表]（Junction Table），联合数据表由发生多对多关联性的两个数据表的主索引键组成。联合数据表分别和其它两个数据表有一对多的关联性，其关系如图 7-1 所示。

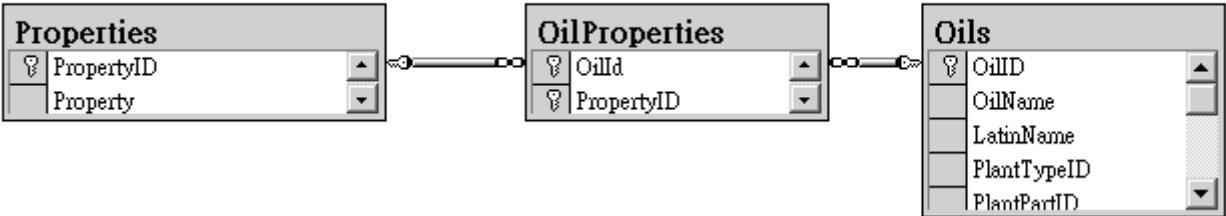
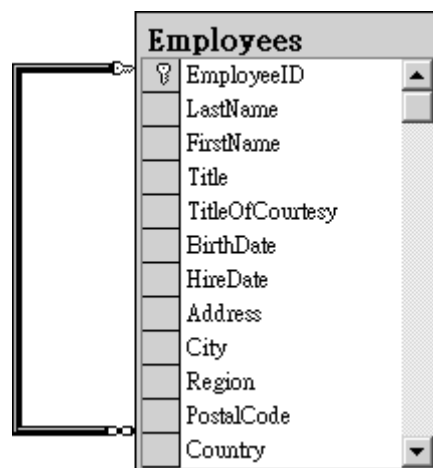


图 7-1 多对多关联性是使用联合数据表的方式解决

自反关联性

在数据库中大部分的关联性是由二个不同的数据表所建立的，但是也有可能一个数据表与其本身有一对一或一对多的关联性，像这一类的关联性我们就称之为 **自反关联性**（**reflexive relationship**）。

所谓的自反关联性通常用于模型化阶层式的架构。最典型的范例是在一个组织中其员工的阶层性。一个员工会有一个经理，而这位经理本身也是员工。藉由员工数据表中的一个数据行引用员工数据表中的主索引键，以建立一对多的自反关联性。



以数据表层级的观点而言，关联性通常是藉由一个数据表中的唯一识别项（常常是主索引键）和另一个数据表中许多的数据列发生关联而加以模型化。

提示

成为外部索引键的识别项通常是另一个数据表的主索引键，它也可以是任何的数据行，或是已经被宣告为唯一性的数据行集合。

SQL Server 可以强制执行您在数据库中所建立的关联性，称之为 [维护参考完整性](#)（maintaining referential integrity）。根据默认值，如果在外部索引键数据表中引用了主要索引键数据表中的数据列，SQL Server 将会拒绝主索引键数据表中的主索引键数据行的变更。

在新版的 Microsoft SQL Server 2000 中，已经具备串联变更主索引键数据表的能力。假如您指示 SQL Server 串联删除关联性，而当您在主索引键数据表中串联删除一数据列时，将会导致 SQL Server 将外部索引键数据表中跟删除的数据列有关联性的数据列一并全部删除。相同的，假如您指示 SQL Server 要串联更新，当您变更主索引键数据表内的主索引键值时，将会导致在外部索引键数据表中相关的数据行被更新。

重要

有些数据库结构描述是相当复杂的。使用串联删除和更新对于维护复杂的数据结构会比较简单，但是 SQL Server 使用的串联式不是循环式的。举例来说，在数据表 A 要删除一列数据列时，会导致数据表 B 中的数据列被删除，接着数据表 C 内的数据列也会被删除（这样是正确的串联式）。但是，您无法在数据表 C 中的数据列被删除后，随着数据表 A 中的数据列也被串联删除（因为这样就变成循环式）。

建立关联性

在 SQL Server 中的关联性是由 [数据表设计师](#) 的 [属性](#) 对话框中的 [关联性](#) 卷标页中来建立。

一对一和一对多这二种关联性也是采用相同的方式来建立。SQL Server 将会基于在外部索引键数据表中所指定的数据行以决定关联性的类型：假如在外部索引键数据行中有一个唯一索引时，那么就会建立一对一的关联性，否则将会建立一对多的关联性。

建立关联性

1. 在详细数据窗格内的 Oils 资料表名称上按右钮，并在快捷菜单中选取 [设计数据表](#)。

SQL Server 会开启 [数据表设计师](#) 窗口。



2. 在设计数据表的工具列上按一下 [管理关联性](#) 按钮。



管理关联性按钮

SQL Server 会开启设计数据表的属性对话框内的 [关联性](#) 卷标页。

屬性

資料表 關聯性 索引/索引鍵 檢查條件約束

資料表名稱: Oils

選定的關聯性(S): FK_OilOdours_Oils

新增(N) 刪除(D)

關聯性名稱(R): FK_OilOdours_Oils

主索引鍵表(P) 外部索引鍵表(O)

Oils OilOdors

主索引鍵表(P)	外部索引鍵表(O)
OilID	OilID

☐ 建立時立即檢查現有資料(K)

☒ 複寫動作將使用此關聯性(F)

☒ 插入和更新動作將使用此關聯性(E)

☐ 串聯更新相關欄位(U)

☒ 串聯刪除相關記錄(C)

關閉 說明

3. 按一下 [新增](#) 按钮。

SQL Server 会在 [主索引鍵表](#) 中建议您使用在数据表清单中的第一个数据表。

屬性

資料表 關聯性 索引/索引鍵 檢查條件約束

資料表名稱: Oils

選定的關聯性(S): ∞ FK_Oils_Cautions

新增(N) 刪除(D)

關聯性名稱(R): FK_Oils_Cautions

主索引鍵表(P) 外部索引鍵表(O)

Cautions Oils

☒ 建立時立即檢查現有資料(K)
☒ 複寫動作將使用此關聯性(F)
☒ 插入和更新動作將使用此關聯性(E)
☐ 串聯更新相關欄位(U)
☐ 串聯刪除相關記錄(C)

關閉 說明

4. 选取 **PlantTypes** 当作是主索引键表。

SQL Server 会建议您使用 **FK_Oils_PlantTypes** 来作为关联性的名称。

屬性

資料表 關聯性 索引/索引鍵 檢查條件約束

資料表名稱: Oils

選定的關聯性(S): ∞ FK_Oils_Cautions

新增(N) 刪除(D)

關聯性名稱(R): FK_Oils_Cautions

主索引鍵表(P) 外部索引鍵表(O)

PlantTypes Oils

☒ 建立時立即檢查現有資料(K)
☒ 複寫動作將使用此關聯性(F)
☒ 插入和更新動作將使用此關聯性(E)
☐ 串聯更新相關欄位(U)
☐ 串聯刪除相關記錄(C)

關閉 說明

5. 选取 **PlantTypeID** 来作为主索引键表的字段。

屬性

資料表 關聯性 索引/索引鍵 檢查條件約束

資料表名稱: Oils

選定的關聯性(S): ∞ FK_Oils_PlantTypes

新增(N) 刪除(D)

關聯性名稱(R): FK_Oils_PlantTypes

主索引鍵表(P) 外部索引鍵表(O)

PlantTypes Oils

PlantTypeID	

☒ 建立時立即檢查現有資料(K)

☒ 複寫動作將使用此關聯性(F)

☒ 插入和更新動作將使用此關聯性(E)

☐ 串聯更新相關欄位(U)

☐ 串聯刪除相關記錄(C)

關閉 說明

6. 选取 **PlantTypeID** 来作为外部索引键表的字段。

屬性

資料表 關聯性 索引/索引鍵 檢查條件約束

資料表名稱: Oils

選定的關聯性(S): ∞ FK_Oils_PlantTypes

新增(N) 刪除(D)

關聯性名稱(R): FK_Oils_PlantTypes

主索引鍵表(P) 外部索引鍵表(O)

PlantTypes Oils

PlantTypeID	PlantTypeID

☒ 建立時立即檢查現有資料(K)
☒ 複寫動作將使用此關聯性(F)
☒ 插入和更新動作將使用此關聯性(E)
☐ 串聯更新相關欄位(U)
☐ 串聯刪除相關記錄(C)

關閉 說明

提示

如果您想要設定串聯更新或串聯刪除時，您可以在 [屬性](#) 中選取 [串聯更新相關字](#)
[段](#) 及 [串聯刪除相關記錄](#) 即可。

7. 按一下 [關閉](#) 按鈕。

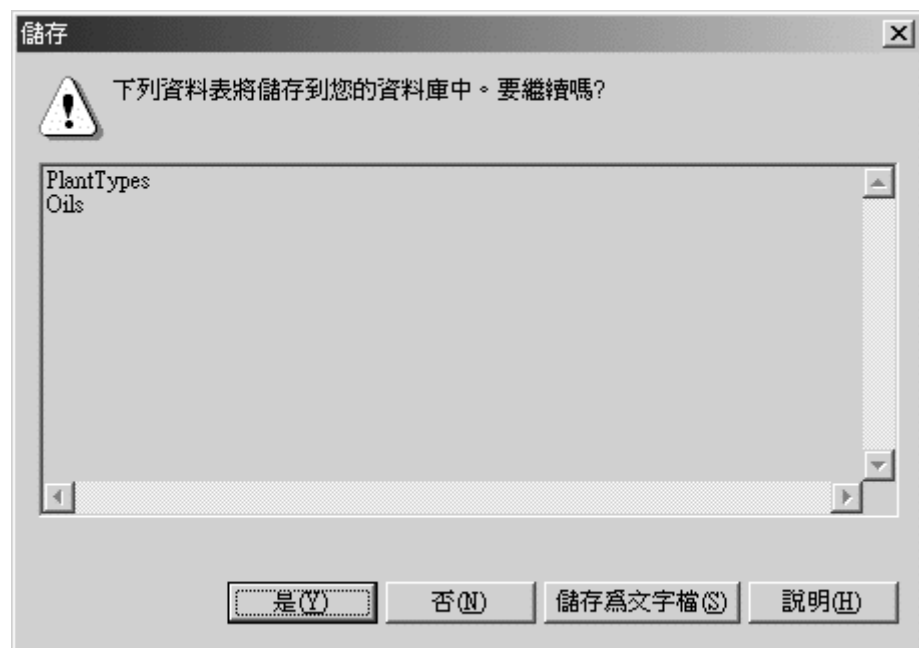
SQL Server 會將設計數據表的 [屬性](#) 對話框關閉。

8. 在设计数据表的工具列上按一下 **存盘** 按钮。



存盘按钮

SQL Server 会显示一个询问您是否要将已经改变的二个数据表储存至数据库的对话框。



9. 按一下 [是](#) 按钮。

SQL Server 会建立此关联性。

10. 将 [数据表设计师](#) 窗口关闭。

管理关联性

数据表之间的关联性应该十分稳定,但是就像其它数据库结构描述一样,它们可能随时会被修改。

Enterprise Manager 会让这些更改的管理更容易些。

变更关联性

变更关联性是非常罕见的动作,但是有时候底层数据表之一的结构变更可能就会需要变更关联

性。您可以藉由 [数据表设计师](#) 的 [属性](#) 对话框内的 [关联性](#) 卷标页以轻易地来改变其关联性。

变更关联性

1. 在详细数据窗格内的 **Oils** 资料表名称上按右键,并在快捷菜单中选取 [设计数据表](#)。

SQL Server 会开启 [数据表设计师](#) 窗口。



2. 在设计数据表的工具列上按一下 [管理关联性](#) 按钮。



管理关联性按钮

SQL Server 会开启设计数据表的 [属性](#) 对话框内的 [关联性](#) 卷标页。

屬性

資料表 關聯性 索引/索引鍵 檢查條件約束

資料表名稱: Oils

選定的關聯性(S): ∞ FK_Oils_PlantTypes

新增(N) 刪除(D)

關聯性名稱(R): FK_Oils_PlantTypes

主索引鍵表(P) 外部索引鍵表(O)

PlantTypes Oils

PlantTypeID	PlantTypeID

☐ 建立時立即檢查現有資料(K)

☒ 複寫動作將使用此關聯性(F)

☒ 插入和更新動作將使用此關聯性(E)

☐ 串聯更新相關欄位(U)

☐ 串聯刪除相關記錄(C)

關閉 說明

3. 請確定您有在 [選定的关联性](#) 下拉式方块中选取 [FK_Oils_PlantTypes](#)。

SQL Server 会显示此关联性的属性。

4. 选取 [PlantPartID](#) 来作为外部索引键字段。

屬性

資料表 關聯性 索引/索引鍵 檢查條件約束

資料表名稱: Oils

選定的關聯性(S): ∞ FK_Oils_PlantTypes

新增(N) 刪除(D)

關聯性名稱(R): FK_Oils_PlantTypes

主索引鍵表(P) 外部索引鍵表(O)

PlantTypes Oils

PlantTypeID	PlantPartID

☐ 建立時立即檢查現有資料(K)

☒ 複寫動作將使用此關聯性(F)

☒ 插入和更新動作將使用此關聯性(E)

☐ 串聯更新相關欄位(U)

☐ 串聯刪除相關記錄(C)

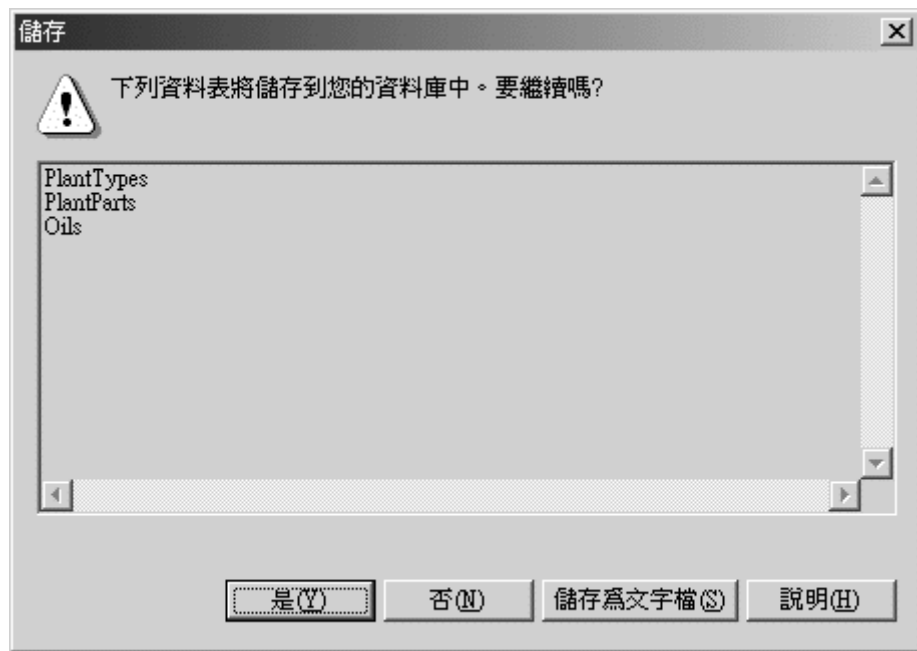
關閉 說明

- 按一下 [關閉](#) 按鈕。

SQL Server 会将 [屬性](#) 对话框关闭。

- 在设计数据表的工具列上按一下 [存盘](#) 按钮。

SQL Server 会显示一个询问您是否要将已改变的二个数据表储存的对话框。



7. 按一下 **是** 按钮。

SQL Server 会改变此关联性。

8. 关闭 **数据表设计师** 窗口。

维护关联性

如同您所预期的一样，关联性也可以使用 **数据表设计师** 的 **属性** 对话框来进行维护。

重新命名关联性

1. 在详细数据窗格内的 **Oils** 资料表名称上按右键，并在快捷菜单中选取 **设计数据表**。

SQL Server 会开启 [数据表设计师](#) 窗口。



2. 在设计数据表的工具列上按一下 [管理关联性](#) 按钮。



管理关联性按钮

SQL Server 会开启设计数据表的 [属性](#) 对话框内的 [关联性](#) 卷标页。

屬性

資料表 關聯性 索引/索引鍵 檢查條件約束

資料表名稱: Oils

選定的關聯性(S): FK_OilProperties_Oils

新增(N) 刪除(D)

關聯性名稱(R): FK_OilProperties_Oils

主索引鍵表(P) 外部索引鍵表(O)

Oils OilProperties

主索引鍵表(P)	外部索引鍵表(O)
OilID	OilId

☐ 建立時立即檢查現有資料(K)

☒ 複寫動作將使用此關聯性(F)

☒ 插入和更新動作將使用此關聯性(E)

☐ 串聯更新相關欄位(U)

☒ 串聯刪除相關記錄(C)

關閉 說明

- 在 [选定的关联性](#) 项目中选取 [FK_Oils_PlantTypes](#) 。

SQL Server 会显示此关联性的属性。

屬性

資料表 關聯性 索引/索引鍵 檢查條件約束

資料表名稱: Oils

選定的關聯性(S): ∞ FK_Oils_PlantTypes

新增(N) 刪除(D)

關聯性名稱(R): FK_Oils_PlantTypes

主索引鍵表(P) 外部索引鍵表(O)

PlantTypes Oils

PlantTypeID	PlantTypeID

☐ 建立時立即檢查現有資料(K)
☒ 複寫動作將使用此關聯性(F)
☒ 插入和更新動作將使用此關聯性(E)
☐ 串聯更新相關欄位(U)
☐ 串聯刪除相關記錄(C)

關閉 說明

- 在 [關聯性名稱](#) 項目中输入『DeleteMe』。

屬性

資料表 關聯性 索引/索引鍵 檢查條件約束

資料表名稱: Oils

選定的關聯性(S): ∞ FK_Oils_PlantTypes

新增(N) 刪除(D)

關聯性名稱(R): DeleteMe

主索引鍵表(P) 外部索引鍵表(O)

PlantTypes Oils

PlantTypeID	PlantTypeID

☐ 建立時立即檢查現有資料(K)
☒ 複寫動作將使用此關聯性(F)
☒ 插入和更新動作將使用此關聯性(E)
☐ 串聯更新相關欄位(U)
☐ 串聯刪除相關記錄(C)

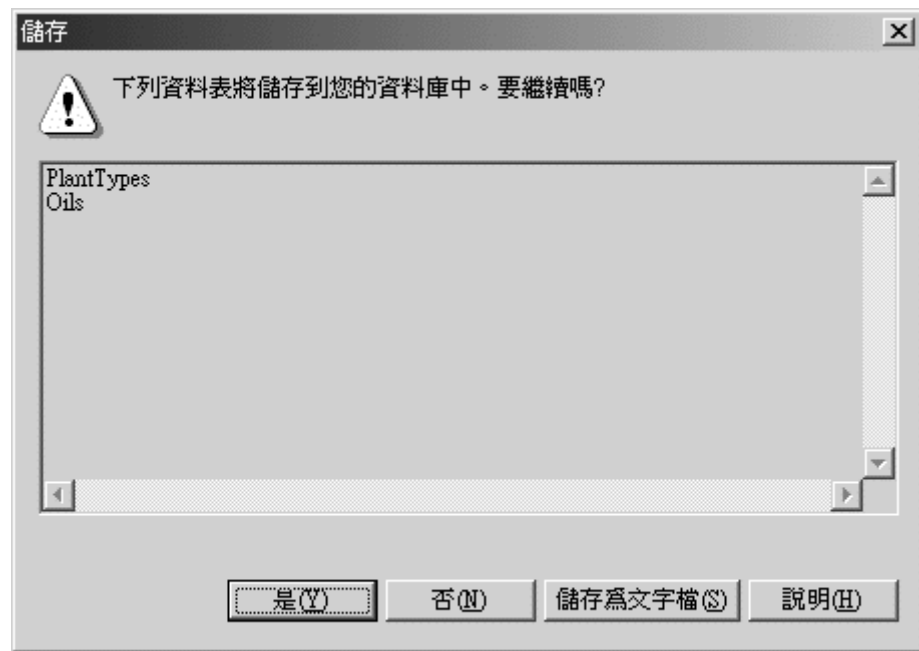
關閉 說明

- 按一下 [关闭](#) 按钮。

SQL Server 会关闭属性对话框。

- 按一下 [存盘](#) 按钮。

SQL Server 会显示一个询问您是否要将已经改变的二个数据表储存的对话框。



7. 按一下 [是](#) 按钮。

SQL Server 会改变此关联性。

8. 关闭设计数据表窗口。

删除关联性

1. 在详细数据窗格内的 **Oils** 资料表名称上按右键，并在快捷菜单中选取 [设计数据表](#)。

SQL Server 会开启 [数据表设计师](#) 窗口。



2. 在设计数据表的工具列上按一下 [管理关联性](#) 按钮。



管理关联性按钮

SQL Server 会开启设计数据表的 [属性](#) 对话框内的 [关联性](#) 卷标页。

屬性

資料表 關聯性 索引/索引鍵 檢查條件約束

資料表名稱: Oils

選定的關聯性(S): FK_OilCautions_Oils

新增(N) 刪除(D)

關聯性名稱(R): FK_OilCautions_Oils

主索引鍵表(P) 外部索引鍵表(O)

Oils OilCautions

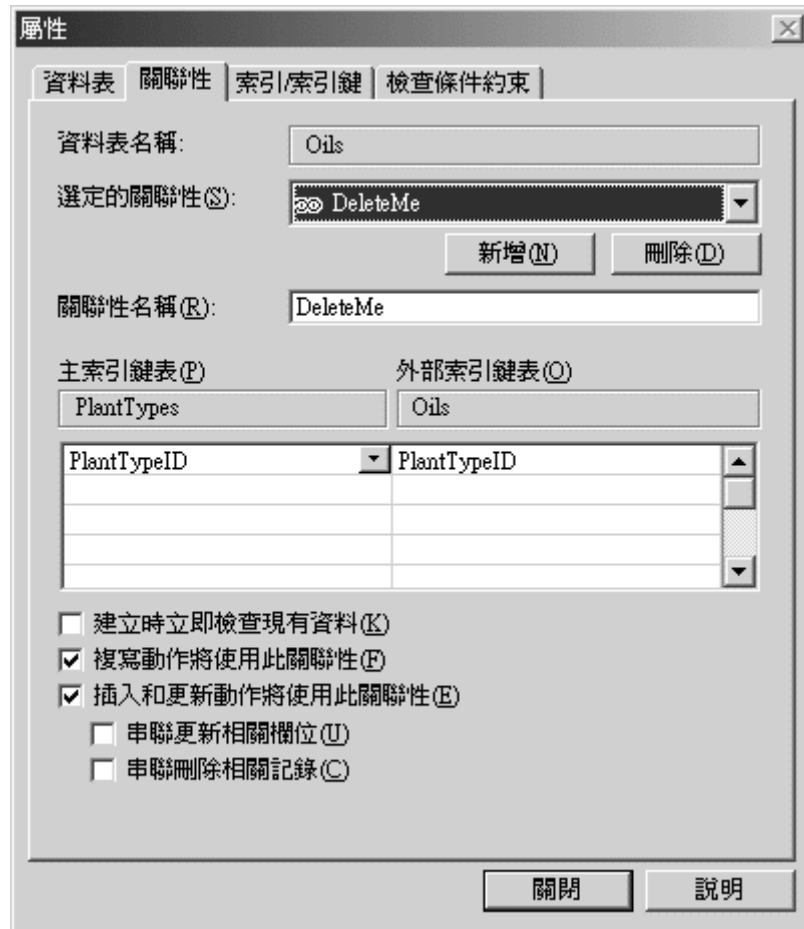
OilID	OilID

☐ 建立時立即檢查現有資料(K)
☒ 複寫動作將使用此關聯性(F)
☒ 插入和更新動作將使用此關聯性(E)
☐ 串聯更新相關欄位(U)
☒ 串聯刪除相關記錄(C)

關閉 說明

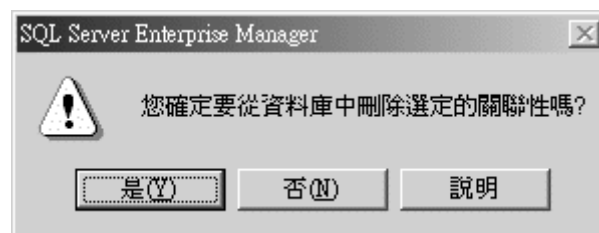
- 在 选定的关联性 项目中选取 **DeleteMe** 。

SQL Server 会显示此关联性的属性。



- 按一下 **刪除** 鈕。

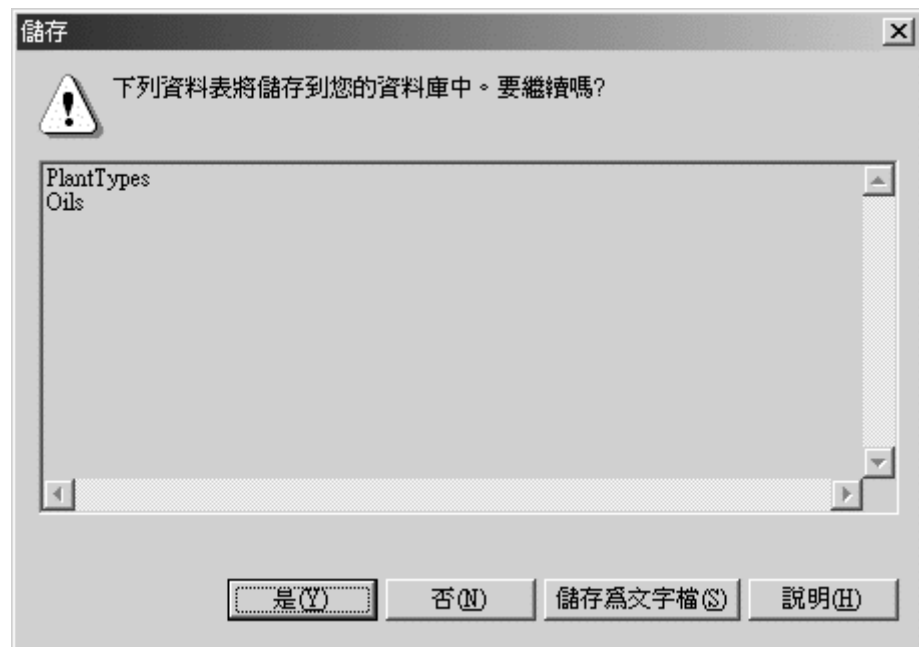
SQL Server 會顯示一段訊息來詢問您是否確定要刪除。



- 按一下 **是** 按鈕。

6. 按一下 [关闭](#) 按钮。
7. 按一下 [存盘](#) 按钮。

SQL Server 会显示一个询问您是否要将已经改变的二个数据表储存的对话框。



8. 按一下 [是](#) 按钮。

SQL Server 会删除此关联性。

9. 关闭 [数据表设计师](#) 窗口。

本章总结

要执行的工作	执行	按钮
建立关联性	针对该数据表开启 数据表设计师 ，然后按一下 管理关联性 按钮以开启 数据表设计师 的 属性 对话框，按一下 新增 按钮，并且设定关联性的属性。	
变更关联性	针对该数据表开启 数据表设计师 ，然后按一下 管理关联性 按钮以开启 数据表设计师 的 属性 对话框，选取在 选定的关联性 下拉式方块的关联性，然后选取不同的数据行为外部索引键数据行。	
重新命名关联性	针对该数据表开启 数据表设计师 ，然后按一下 管理关联性 按钮以开启 数据表设计师 的 属性 对话框，选取在 选定的关联性 下拉式方块的关联性，然后变更 关联性名称 中的属性。	
删除关联性	针对该数据表开启 数据表设计师 ，然后按一下 管理关联性 按钮以开启 数据表设计师 的 属性 对话框，选取在 选定的关联性 下拉式方块的关联性，然后按一下 删除 按钮。	

8. 建立检查条件约束

在本章中，您将学习到：

- 建立检查条件约束。
- 变更条件约束的文字。
- 重新命名检查条件约束。
- 删除检查条件约束。

认识检查条件约束

在数据库设计方面另一个重要的议题就是确认 [数据完整性](#)（data integrity）。数据完整性的规则会确保数据库中储存的数据正确性。数据完整性有数个层级，在第 7 章中我们已经描述了关联完整性，关联完整性会确认数据表之间的关联性，并且可以正确的维护数据表中的数据。

检查条件约束是本章所要讨论的主题，用在强制执行数据库完整性的另外二种形式：**值域完整性**（domain integrity）和**实体完整性**（entity integrity）。在关系型的术语中，所谓的**值域**（domain）是指一个数据行所能包含值的范围。数据行的数据类型别是指值域的属性之一，但是只定义数据类型别并不够充分。举例来说，一个 **smallint** 数据行可以包含自 -32,768 至 32767 之间的整数值，对于一个用来储存员工已经从大学毕业几年的数据行来说，这可能是一个好的数据类型别。但是该数据行实际的值范围需要有更多的限制，一般来说，这个资料行应该要介于 1900 年至目前的年份之间，为达成这个目的，您可以使用检查条件约束来确定所输入的值范围，而不会输入 1543 或 2075 以作为此数据行的值。

实体完整性条件约束是用于实体本身的完整性。实体完整性条件约束最重要的地方是在于每一个实体是必须是唯一的识别项。这个条件约束是藉由在数据表中指定主索引键以建置。实体完整性也可以配合使用检查条件约束来建置（多个数据行）。举例来说，假如有一个资料表包含 **Country** 和 **State** 资料行时，您可以使用检查条件约束以指定—如果 **Country** 资料行包含 **USA** 的值时，**State** 数据行的值是「AZ」才是有效的，否则无效。

检查条件约束通常是指定为 **布尔表达式**（Boolean Expression）。布尔表达式可以用来评估其结果为 **TRUE** 或 **FALSE**。我们将在第 13 章中学习有关布尔表达式。在本章中，我们将使用下列这个表达式：

```
LEN(<column >) >= 4
```

LEN 是一个 Transact-SQL 函数，它可以传回字符串中的字符数，因此假如 LEN(<column >) 传回 4 个或更多的字符时，则布尔表达式将传回 TRUE；假如它所包含的字符数少于 4 个时，则布尔表达式会传回 FALSE。

建立检查条件约束

就像索引和关联性一样，您可以使用数据表设计师的属性对话框来建立检查条件约束。

建立检查条件约束

1. 在详细数据窗格内的 Oils 资料表名称上按右钮，并在快捷菜单中选取 [设计数据表](#)。

SQL Server 会开启 [数据表设计师](#) 窗口。



2. 按一下 [管理条件约束](#) 按钮。



管理条件约束按钮

Microsoft SQL Server 会开启设计数据表的 [属性](#) 对话框内的 [检查条件约束](#) 标签页。



3. 按一下 **新增** 按钮。

SQL Server 会显示一个建议您使用 **CK_Oils** 来作为条件约束名称，在此练习中我们将采用它。

屬性

資料表 | 關聯性 | 索引/索引鍵 | 檢查條件約束

資料表名稱: Oils

選定的條件約束(S): CK_Oils

新增(N) 刪除(D)

條件約束名稱(C): CK_Oils

條件約束運算式(X):

☒ 建立時立即檢查現有資料(K)

☒ 複寫動作必須合乎條件約束(F)

☒ 插入和更新動作必須合乎條件約束(E)

關閉 說明

4. 輸入『LEN(OilName) >= 4』的條件約束表达式。

屬性

資料表 關聯性 索引/索引鍵 檢查條件約束

資料表名稱: Oils

選定的條件約束(S): CK_Oils

新增(N) 刪除(D)

條件約束名稱(C): CK_Oils

條件約束運算式(X):
LEN(OilName) >= 4

☒ 建立時立即檢查現有資料(K)
☒ 複寫動作必須合乎條件約束(F)
☒ 插入和更新動作必須合乎條件約束(E)

關閉 說明

提示

假如您建立新的检查条件约束，并且不在乎现存值是否符合条件约束时，您可以指示 SQL Server 忽视已经存在的值是否合乎条件约束（藉由取消选取 [建立时立即检查现有数据](#)）。

5. 按一下 [关闭](#) 按钮。

SQL Server 会关闭设计数据表的 [属性](#) 对话框。

6. 按一下 [存盘](#) 按钮。



存盘按钮

SQL Server 会依据所设定的条件约束内容以检查数据表中所有的数据列，然后再将此条件约束存盘。

管理检查条件约束

检查条件约束是数据库描述结构的一部份，在正常的情况之下，不需要什么维护。当您建立数据库时，您定义它们一次就已经足够了。然而，当数据库结构描述偶尔改变时，此时检查条件约束就需要修正。**Enterprise Manager** 会让这个动作很容易执行。

变更检查条件约束

数据表设计师 提供一个机制（藉由您建立检查条件约束时的 **属性** 对话框）以供变更检查条件约束的文字内容。

更改条件约束文字

1. 如果 **Oils** 数据表的设计数据表窗口没有开启时，请您在详细数据窗格内的 **Oils** 资料表名称上按右钮，并在快捷菜单中选取 **设计数据表**。

SQL Server 会开启 **数据表设计师** 窗口。

2.設計資料表 'Oils' (在 'Aromatherapy' 中) 於 'MICKEY'

資料行名稱	資料型別	長度	是否允許 Null
OilID	int	4	
OilName	nvarchar	50	
LatinName	nvarchar	50	✓
PlantTypeID	int	4	✓
PlantPartID	int	4	✓
Sample	char	10	✓
Description	char	30	✓

資料行

描述

預設值

精確度

小數位數

識別

識別值種子

識別值增量

為 RowGuid

公式

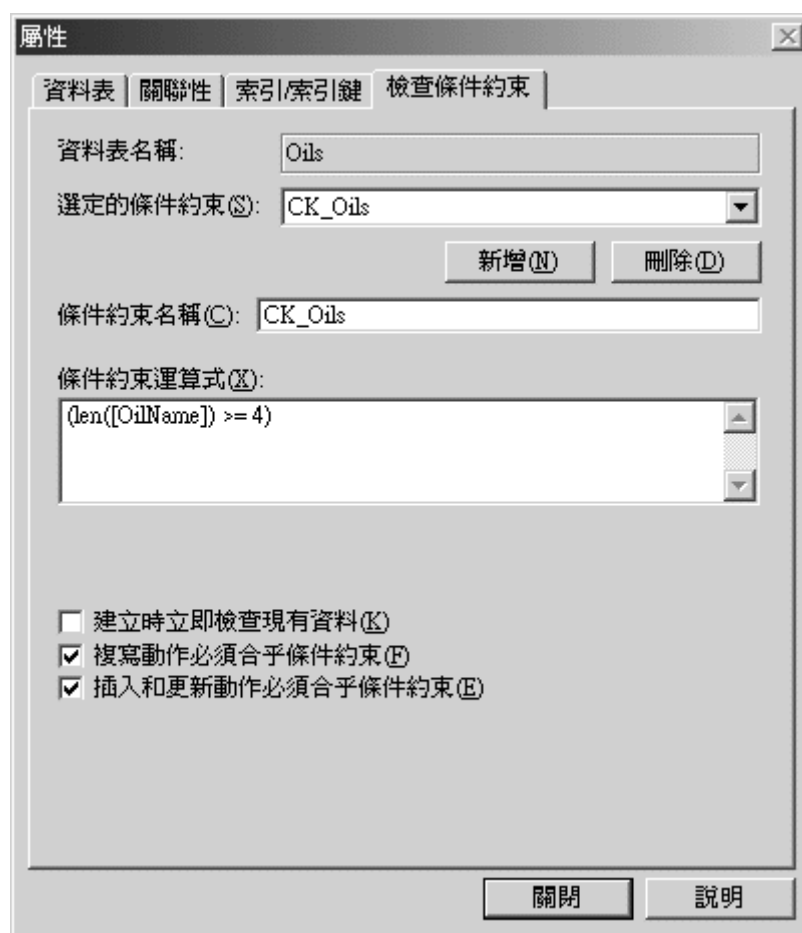
定序

- 按一下 [管理条件约束](#) 按钮。



管理条件约束按钮

Microsoft SQL Server 会开启设计数据表的 [属性](#) 对话框内的 [检查条件约束](#) 标签页。



3. 请确定您有在 [选定的条件约束](#) 下拉式方块中选取 [CK_Oils](#) 条件约束。
4. 将条件约束文字更改为『 `[LEN(OilName) > 2]` 』来作为新的条件约束表达式。



5. 按一下 [关闭](#) 按钮。

SQL Server 会将设计数据表的 [属性](#) 对话框关闭。

6. 按一下 [存盘](#) 按钮。

SQL Server 会依据所设定的条件约束内容以检查数据表中所有的数据列，然后再将此条件约束存盘。

维护检查条件约束

就像其它的数据表属性一样，检查条件约束可以藉由 [数据表设计师](#) 内的 [属性](#) 对话框来维护。

重新命名检查条件约束

1. 在详细数据窗格内的 **Oils** 资料表名称上按右钮，并在快捷菜单中选取 [设计数据表](#)。

SQL Server 会开启 [数据表设计师](#) 窗口。

2.設計資料表 'Oils' (在 'Aromatherapy' 中) 於 'MICKEY'

資料行名稱	資料型別	長度	是否允許 Null
OilID	int	4	
OilName	nvarchar	50	
LatinName	nvarchar	50	✓
PlantTypeID	int	4	✓
PlantPartID	int	4	✓
Sample	char	10	✓
Description	char	30	✓

資料行

描述

預設值

精確度

小數位數

識別

識別值種子

識別值增量

為 RowGuid

公式

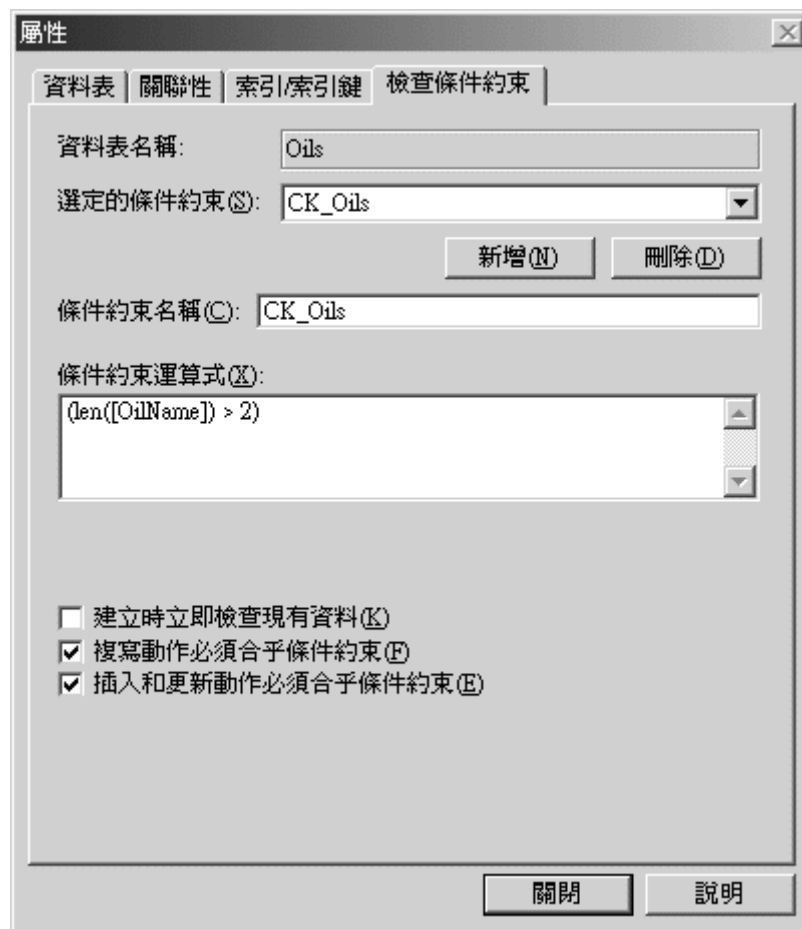
定序

- 按一下 [管理条件约束](#) 按钮。



管理条件约束按钮

Microsoft SQL Server 会开启设计数据表的属性对话框内的 [检查条件约束](#) 标签页。



3. 在 [选定的条件约束](#) 字段中选取 [CK_Oils](#)，并且将它更改为 [CK_DeleteMe](#)。



4. 按一下 [关闭](#) 按钮。

SQL Server 会将设计数据表的 [属性](#) 对话框关闭。

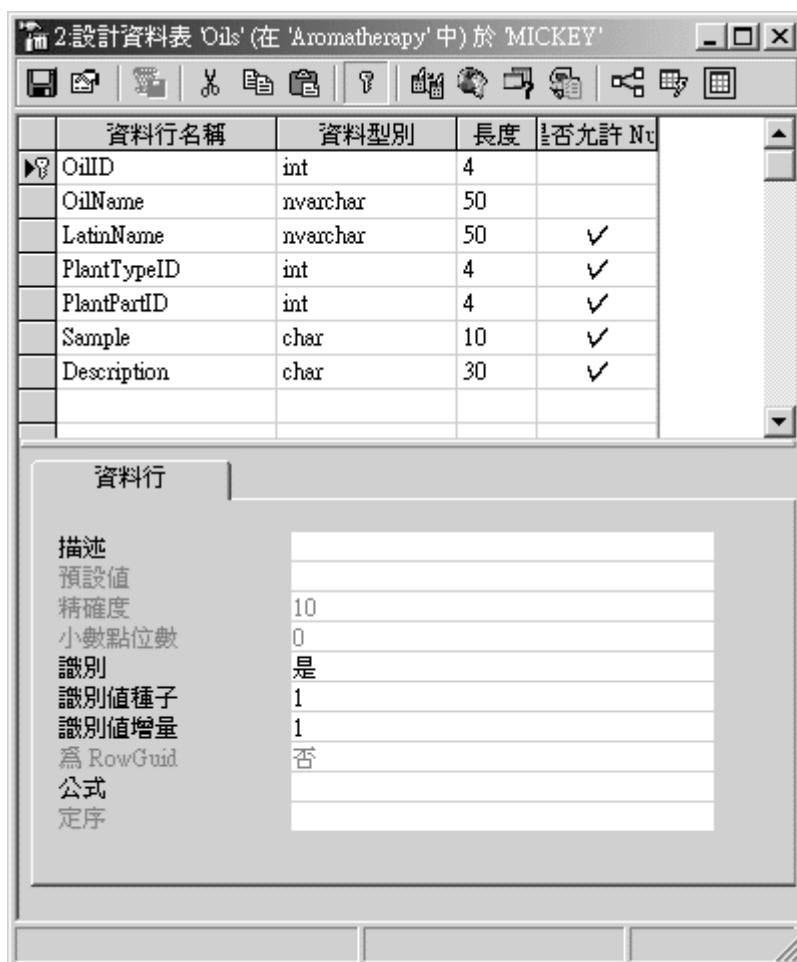
5. 按一下 [存盘](#) 按钮。

SQL Server 会依据所设定的条件约束内容以检查数据表中所有的数据列，然后再将此条件约束存盘。

删除检查条件约束

1. 在详细数据窗格内的 Oils 资料表名称上按右键，并在快捷菜单中选取 [设计数据表](#)。

SQL Server 会开启 [数据表设计师](#) 窗口。



2. 按一下 [管理条件约束](#) 按钮。



管理条件约束按钮

Microsoft SQL Server 会开启设计数据表的属性对话框内的 [检查条件约束](#) 标签页。

屬性

資料表 | 關聯性 | 索引/索引鍵 | 檢查條件約束

資料表名稱: Oils

選定的條件約束(S): CK_DeleteMe

新增(N) 刪除(D)

條件約束名稱(C): CK_DeleteMe

條件約束運算式(X):
(len([OilName]) > 2)

☐ 建立時立即檢查現有資料(K)
☒ 複寫動作必須合乎條件約束(F)
☒ 插入和更新動作必須合乎條件約束(E)

關閉 說明

3. 在 选定的条件约束 字段中选取 **CK_DeleteMe**，然后再按一下 **删除** 按钮。

SQL Server 会将此条件约束删除。



4. 按一下 **关闭** 按钮。





SQL Server 会将设计数据表的 **属性** 对话框关闭。

5. 按一下 **存盘** 按钮。

SQL Server 会移除此条件约束。

6. 关闭 数据表设计师 。

本章总结

要执行的 工作	执行	按 钮
建立检查 条件约束	针对数据表开启 数据表设计师 ，按一下 管理条件约束 按钮以便开启设计数据表的 属性 对话框，按一下 新增 按钮，然后输入条件约束表达式。	
变更条件 约束的文 字	针对数据表开启 数据表设计师 ，按一下 管理条件约束 按钮以便开启设计数据表的 属性 对话框，在 选定的条件约束 字段中选取您想要变更的条件约束名称，然后将条件约束的内容更改为新的内容。	
重新命名 检查条件 约束	针对数据表开启 数据表设计师 ，按一下 管理条件约束 按钮以便开启设计数据表的 属性 对话框，在 选定的条件约束 字段中选取您想要变更的条件约束名称，然后在 条件约束名称 字段中输入新的名称。	
删除检查 条件约束	针对数据表开启 数据表设计师 ，按一下 管理条件约束 按钮以便开启设计数据表的 属性 对话框，在 选定的条件约束 字段中选取您想要删除的条件约束名称，然后再按一下 删除 按钮。	

9. 建立数据表对象

在本章中，您将学习到：

- 建立预设。
- 系结预设和资料行。
- 解除系结预设。
- 建立规则。
- 系结规则和数据行。
- 建立使用者自订数据类型。
- 分配使用者自订数据类型给数据行。

在前几章中，您已经学习了如何指派各种属性值，例如为数据表中的数据行设定预设以及检查条件约束等等。然而，有时候会有某个特定的数据行型别必须用于数个数据表中。遇到这种情况时，最常使用的解决方式是在一个地方建立这些属性，并且将它套用到每一个数据表中。

预设、规则以及使用者自订的数据型别等等提供了在一个地方建立及维护这种类型对象的机制。举例来说，您可能会建立数据库并且提供必要的信息以响应客户调查，因此您可以针对每一个问题先将预设设定为「Unknown」。如果您建立一个预设，并且将它系结到适当的数据行中，一旦日后您需要变更响应问题的讯息时，您只需要更改建立预设的地方为「Unanswered」即可，而所有利用此预设的数据行都会自行变更，不用一个一个改，非常方便。

认识预设

预设的功能与您在数据表设计师中建立数据行时有指定默认值属性是相同的。当建立数据列时，假如使用者没有指定值时，**Microsoft SQL Server** 会自动指派一个值。预设是一个数据库层级的对象，它可以指派给多个数据行使用。

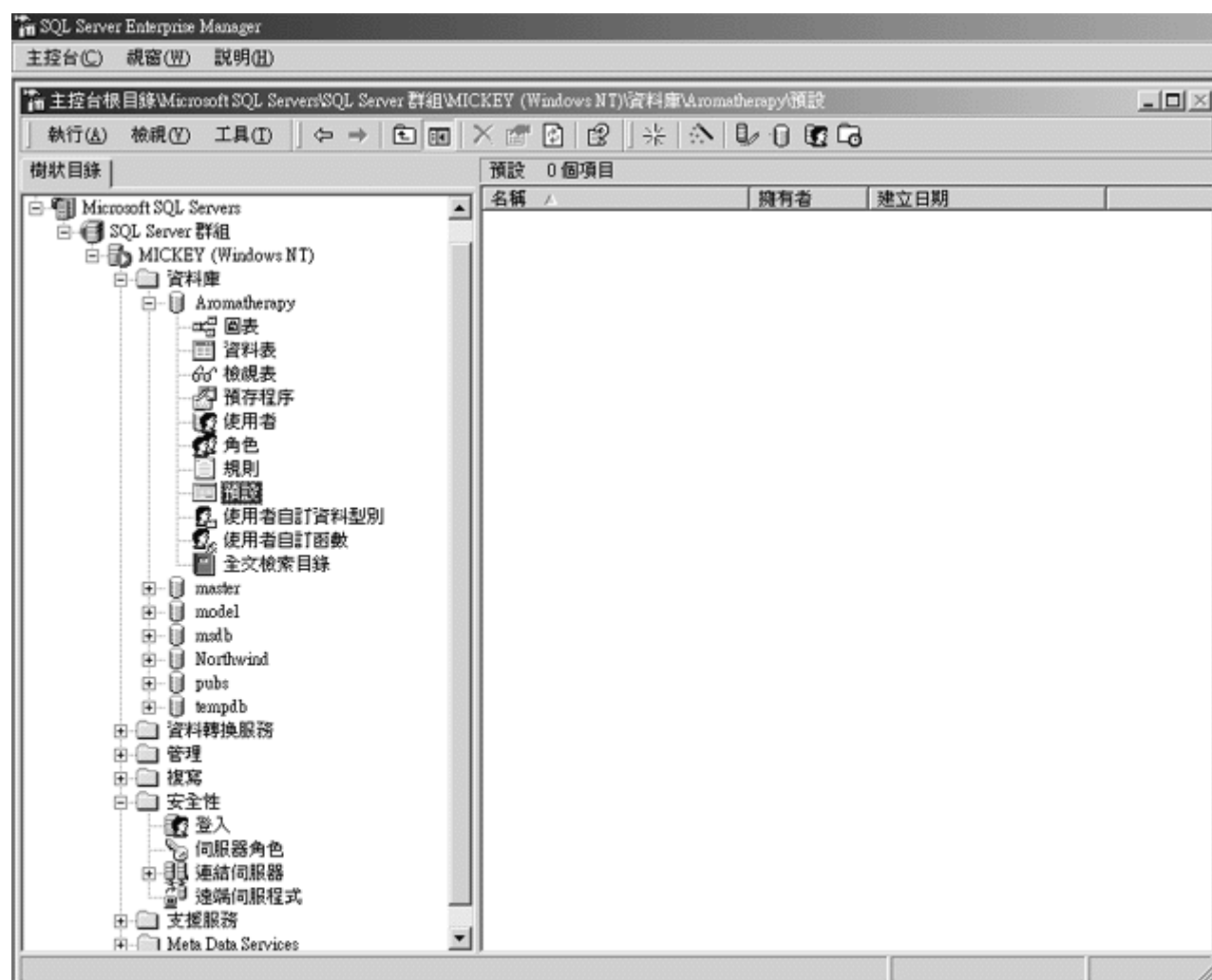
建立预设

在数据库中，预设是一个独立的对象，在您要将它系结到数据表内的数据行之前，必须先建立预设。

建立预设

1. 按一下 Aromatherapy 数据库内的 **预设** 数据夹。

SQL Server 会在详细资料窗格中显示预设清单（在范例数据库中并没有任何预设对象）。



- 按一下 **新增** 按鈕。



新增按鈕

SQL Server 会显示 [预设属性](#) 的对话框。



3. 在 [名称](#) 字段内输入『DefaultUnknown』。



4. 在值字段內輸入『'Unknown'』。



5. 按一下 **確定** 按钮。

SQL Server 会建立此预设。

系结预设和资料行

1. 指向 **数据表** 数据夹，并且在详细资料窗格内的 **Oils** 数据表上按右键，选取 **设计**

数据表。

SQL Server 会开启 **数据表设计师**。

2設計資料表 'Oils' (在 'Aromatherapy' 中) 於 'MICKEY'

資料行名稱	資料型別	長度	是否允許 Null
OilID	int	4	
OilName	nvarchar	50	
LatinName	nvarchar	50	✓
PlantTypeID	int	4	✓
PlantPartID	int	4	✓
Description	char	30	✓

資料行

描述	
預設值	
精確度	10
小數點位數	0
識別	是
識別值種子	1
識別值增量	1
為 RowGuid	否
公式	
定序	

- 在此数据表中加入一个名称为 **Sample** 的新数据行，并且接受由 SQL Server 所预设的数据型态和长度。

2. 設計資料表 'Oils' (在 'Aromatherapy' 中) 於 'MICKEY'

資料行名稱	資料型別	長度	是否允許 Null
OilID	int	4	
OilName	nvarchar	50	
LatinName	nvarchar	50	✓
PlantTypeID	int	4	✓
PlantPartID	int	4	✓
Description	char	30	✓
▶ Sample	char	10	✓

資料行

描述

預設值

精確度

小數點位數

識別

識別值種子

識別值增量

為 RowGuid

公式

定序

0

0

否

否

<資料庫預設值>

- 按一下 **默认值** 储存格，然后在清单中选取 **dbo.DefaultUnknown** 项目。

2.設計資料表 'Oils' (在 'Aromatherapy' 中) 於 'MICKEY'

資料行名稱	資料型別	長度	是否允許 Null
OilID	int	4	
OilName	nvarchar	50	
LatinName	nvarchar	50	✓
PlantTypeID	int	4	✓
PlantPartID	int	4	✓
Description	char	30	✓
Sample	char	10	✓

資料行

描述

預設值

精確度

小數點位數

識別

識別值種子

識別值增量

為 RowGuid

公式

定序

dbo.DefaultUnknown

0

0

否

否

<資料庫預設值>

- 按一下 [存盤](#) 按钮。



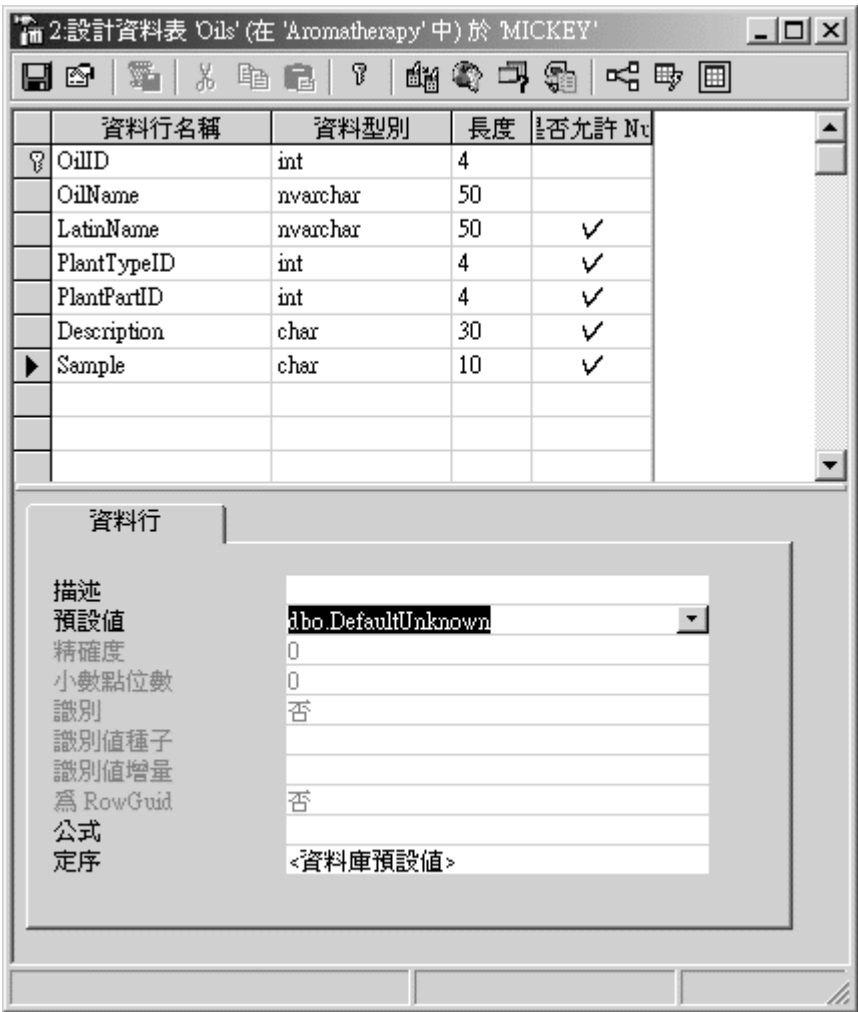
存盤按钮

此时 SQL Server 会储存此数据表。

解除系统预设

1. 如果 Oils 数据表的设计数据表窗口没有开启时，请您在详细数据窗格内的 Oils 资料表名称上按右钮，并在快捷菜单中选取 [设计数据表](#)。

SQL Server 会开启 [数据表设计师](#) 窗口。



2. 选取 **Sampl** 数据行。

此时会显示此数据行的属性。

資料行名稱	資料型別	長度	是否允許 Null
OilID	int	4	
OilName	nvarchar	50	
LatinName	nvarchar	50	✓
PlantTypeID	int	4	✓
PlantPartID	int	4	✓
Description	char	30	✓
Sample	char	10	✓

資料行	預設值
描述	
預設值	dbo.DefaultUnknown
精確度	0
小數點位數	0
識別	否
識別值種子	
識別值增量	
為 RowGuid	否
公式	
定序	<資料庫預設值>

3. 在 **默认值** 字段中选取 **dbo.DefaultUnknown**，并且按一下 **Delete** 键以便移除此

预设。



- 按一下 [存盤](#) 按钮。

SQL Server 会储存数据行定义。

认识规则

规则就如同预设一样，是一个数据库层级的对象，因此它可以被套用在多个数据表中的数据行。

规则亦如同检查条件约束一样，可以在数据行中指定可接受的资料值范围，但是它在使用上还是

多有限制。举例来说，一个数据行可以拥有多个检查条件约束，但是却只能有一个规则。

提示

Microsoft 已经声明不建议使用规则，并且推荐使用条件约束取代规则。然而，规则仍然存在于 SQL Server 数据库中，因为只有规则可以被套用至使用者自订数据类型。

不像检查条件约束一样，规则无法直接引用数据行的名称，必须以系结的方式套用至数据行。而规则所套用的值则是透过变量传递，我们将在第二十四章讨论变量。

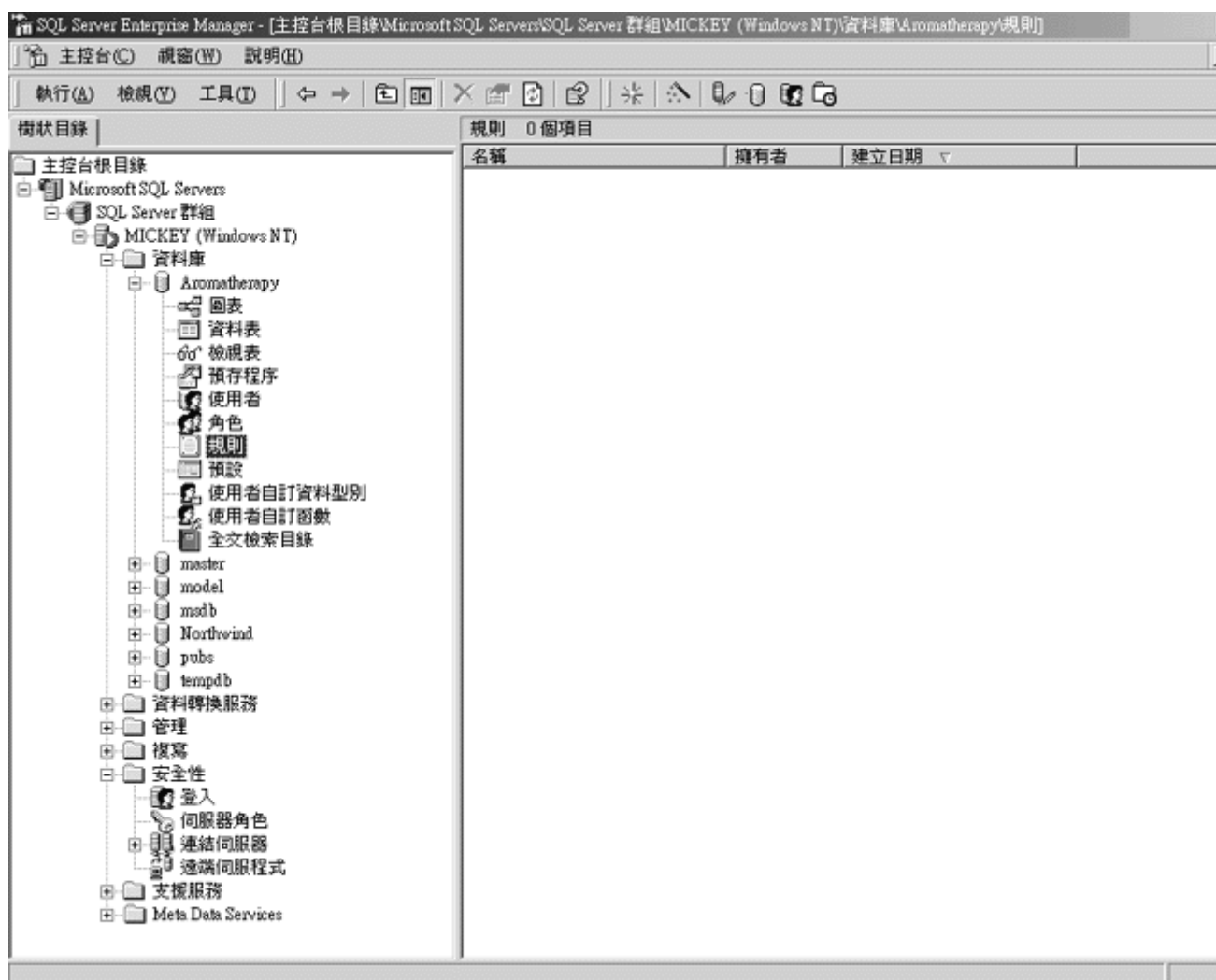
建立规则

规则就像预设一样，是一个独立的数据库对象，您必须在将它们指定给数据表内的数据行之前，就必须先建立它。

建立规则

1. 按一下 Aromatherapy 数据库内的 [规则] 数据夹。

SQL Server 会在详细数据窗格中显示规则清单（在范例数据库中并没有任何预设对象）。



2. 按一下 **新增** 按钮。



新增按钮

SQL Server 会开启一个 [规则属性](#) 的对话框。



3. 在 [名称](#) 字段中输入规则名称『SampleRule』。



4. 在 [文字](#) 字段中输入『LEN(@fldValue) > 3』。



提示

请您记住 **LEN** 是一个 **Transact-SQL** 函数，会传回字符串中的字符数，并且在 **Transact-SQL** 陈述式之后有一个 **@** 的标记是表示一个变量、一个值将会传递至陈述式中。因此在本范例中，假如数据行的长度值是大于 **3**，此规则将只会传回 **TRUE** 值。

5. 按一下 **确定** 按钮。

SQL Server 会将 **规则属性** 对话框关闭，并且建立规则。

系结规则和数据行

1. 在详细数据窗格中按两下 **SampleRule** 规则名称以开启此规则的 **规则属性** 对话框。

SQL Server 会显示 **规则属性** 对话框。



2. 按一下 [系統數據行](#) 按鈕。

SQL Server 會顯示 [系統規則到數據行](#) 對話框。

緊結規則到資料行 - SampleRule

一般

規則: SampleRule

資料表(T): [dbo].[Cautions]

未繫結的資料行(U):

名稱	資料型別
CautionID	int
CautionNum	int
Caution	nvarchar
Description	ntext

已繫結的資料行(B):

名稱	資料型別
----	------

新增(N) >>

<< 移除(R)

確定 取消 套用(A) 說明

- 在数据表下拉式方块中选取 **[dbo].[Oils]** 。

SQL Server 会显示在 Oils 数据表内的数据行。

緊結規則到資料行 - SampleRule

一般

規則: SampleRule

資料表(T): [dbo].[Oils]

未繫結的資料行(U):

名稱	資料型別
OilID	int
OilName	nvarchar
LatinName	nvarchar
PlantTypeID	int
PlantPartID	int
Description	char
Sample	char

已繫結的資料行(B):

名稱	資料型別
----	------

新增(B) >>

<< 移除(B)

確定 取消 套用(A) 說明

- 在 [未系结的数据行](#) 清单中选取 **Sample** 数据行，然后按一下 [新增](#) 按钮。

SQL Server 会将此数据行移至 [已系结的数据行](#) 清单中。



5. 按一下 **確定** 按钮。

SQL Server 会关闭 **系统规则到数据行** 对话框。

6. 按一下 **确定** 按钮。

SQL Server 会关闭 **规则属性** 对话框。

认识使用者自订数据型别

规则和预设与维护数据库条件约束而言是非常好用的机制，但是 **SQL Server** 提供一个更好用的机制，那就是使用者自订数据类型。使用者自订数据类型?是以 **SQL Server** 的系统数据类型为基础，并且包含数据行长度规格。另外，规则和预设可以选择性的套用至使用者自订资料型别。

当一个数据行是基于一个使用者自订数据类型时，此数据行将继承此数据类型的所有属性设定。

当使用者自订数据类型所指定的值改变时，基于该数据类型的数据行也会改变。

提示

如果使用者自订数据类型是建立在 **model** 数据库中，那么所有新建立的数据库将会自动地加入此使用者自订数据类型。

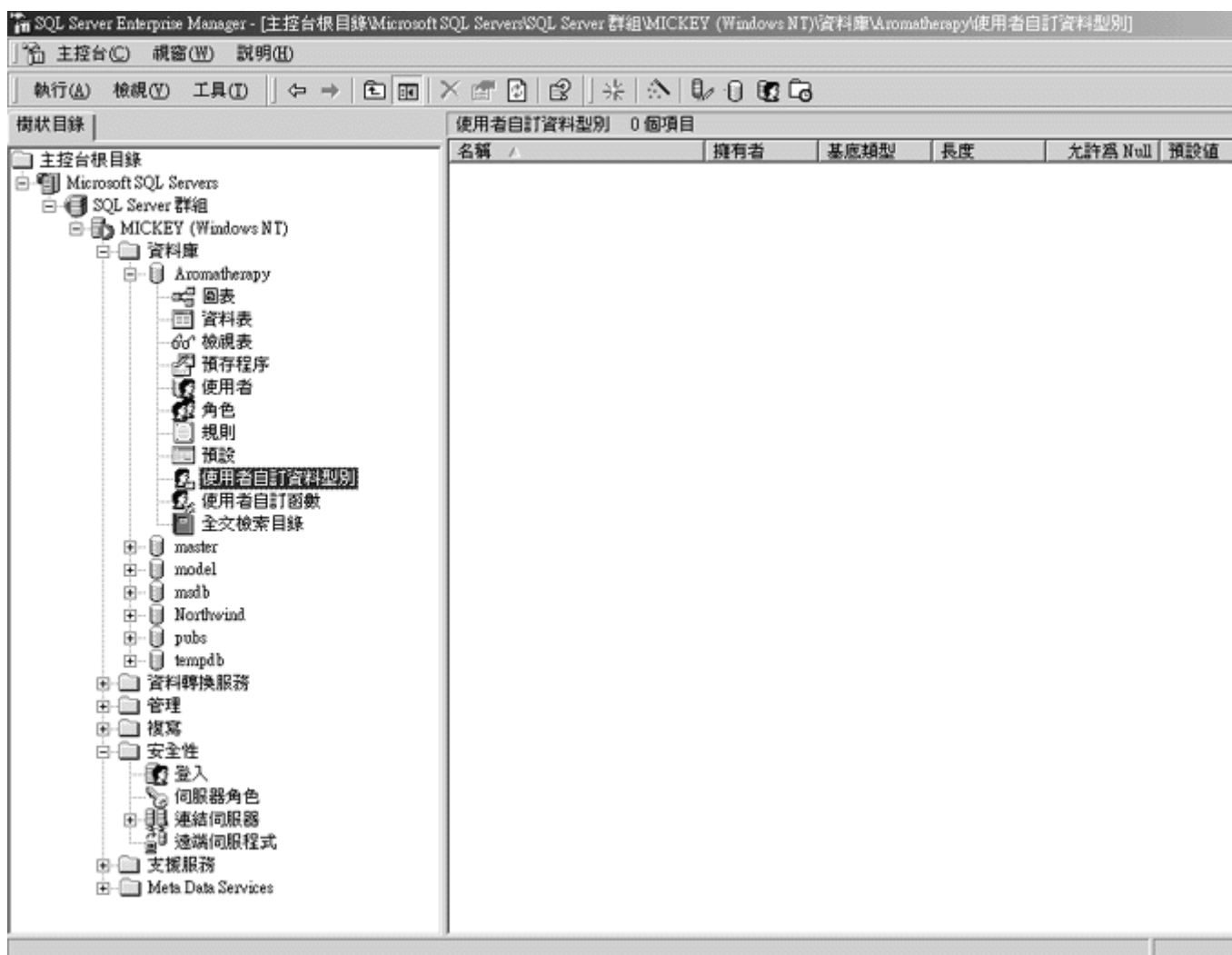
建立使用者自订数据类型

再次说明，使用者自订数据类型是一个独立的数据库对象，您必须在将它们指定给数据表内的数据行之前，就必须先建立它。

建立使用者自订数据类型

1. 按一下 Aromatherapy 数据库内的 使用者自订数据类型 数据夹。

SQL Server 会在详细数据窗格中列出已经建立的使用者自订数据类型（在范例数据库 中并没有任何预设对象）。



2. 按一下 **新增** 按钮。



新增按钮

SQL Server 会显示 **使用者自订的数据型别属性** 对话框。

A screenshot of a Windows dialog box titled '使用者自訂的資料型別屬性 - MICKEY'. The dialog has a tab labeled '一般'. Inside, there is a user icon and a text field for '名稱(N):'. Below that are several settings: '資料型別(D):' set to 'bigint', '長度(L):' set to '19', '允許 NULL(U)' with an unchecked checkbox, '規則(R):' set to '(無)', and '預設(E):' set to '(無)'. There is a button labeled '使用(W)...' and at the bottom are '確定', '取消', and '說明' buttons.

使用者自訂的資料型別屬性 - MICKEY

一般

名稱(N):

資料型別(D): bigint

長度(L): 19

允許 NULL(U) ☐

規則(R): (無)

預設(E): (無)

使用(W)...

確定 取消 說明

3. 在 **名称** 字段中输入『MySample』来当作是使用者自订的数据型别名称。

使用者自訂的資料型別屬性 - MICKEY

一般

名稱(N): MySample

資料型別(D): bigint

長度(L): 19

允許 NULL(U) ☐

規則(R): (無)

預設(E): (無)

使用(W)...

確定 取消 說明

4. 将数据型别设定为 **varchar**，并且将长度设定为 **20**。

使用者自訂的資料型別屬性 - MICKEY

一般

名稱(N): MySample

資料型別(D): varchar

長度(L): 20

允許 NULL(U) ☐

規則(R): (無)

預設(E): (無)

使用(W)...

確定 取消 說明

5. 在 **規則** 下拉式方块中选取 **dbo.SampleRule** 項目。

使用者自訂的資料型別屬性 - MICKEY

一般

名稱(N): MySample

資料型別(D): varchar

長度(L): 20

允許 NULL(U) ☐

規則(R): **dbo.SampleRule**

預設(E): (無)

使用(W)...

確定 取消 說明

6. 允许 **NULL** 和 **预设** 项目我们将采取预设的值，然后按一下 **确定** 按钮。

SQL Server 会建立此使用者自订数据型别。

分配使用者自订数据型别给数据行

1. 在详细数据窗格内的 **Oils** 资料表名称上按右钮，并且在快捷菜单中选取 **设计数据表**。

SQL Server 会开启 **数据表设计师** 窗口。



2. 选取 **Sample** 数据行，并且从数据型别的下拉式方块中选取 **MySample** 。

SQL Server 会将此数据型别设定为 **MySample** 。

提示




使用者自订数据型别会出现在数据型别清单中的底部。



3. 按一下 **存盘** 按钮。

SQL Server 会储存新定义的数据表。

本章总结

要执行的工作	执行	按钮
建立预设	按一下数据库的 预设 数据夹，并且按一下 新增 按钮和输入预设的名称及值。	
系统预设和资料行	针对该数据表开启 数据表设计师 窗口，并且从 默认值 下拉式方块中选取预设。	
解除系统预设	针对该数据表开启 数据表设计师 窗口，选取一个已经系统的数据行，然后在 默认值 字段中删除预设。	
建立规则	按一下数据库的 规则 数据夹，并且按一下 新增 按钮和输入规则的名称及文字内容。	
系统规则和 数据行	在 规则属性 对话框中按一下 系统数据行 按钮以开启 系统规则到数据行 对话框。在 未系统的数据行 清单中选取适当的字段，然后按一下 新增 按钮。	
建立使用者 自订数据型 别	按一下数据库的 使用者自订数据型别 数据夹，并且按一下 新增 按钮以设定使用者自订数据型别的属性。	
分配使用者 自订数据型 别	针对该数据表开启 数据表设计师 视给数据行窗，并且从 数据型别 清单中选取您要使用的使用者自订数据型别。	

10. 建立数据库图表

在本章中，您将学习到：

- 建立数据库图表。
- 在数据库图表中变更显示的明细程度。
- 在数据库图表中新增已经存在的数据表。
- 从数据库图表中移除数据表。
- 在数据库图表窗口中变更数据表。
- 在数据库图表窗口中建立数据表。

认识数据库图表

数据库图表提供一个将数据表结构和关联性（数据库结构描述，Database Schema）可视化的方式。同时数据库图表也会反应出您所作的任何变更。

从现存的结构描述建立数据库图表

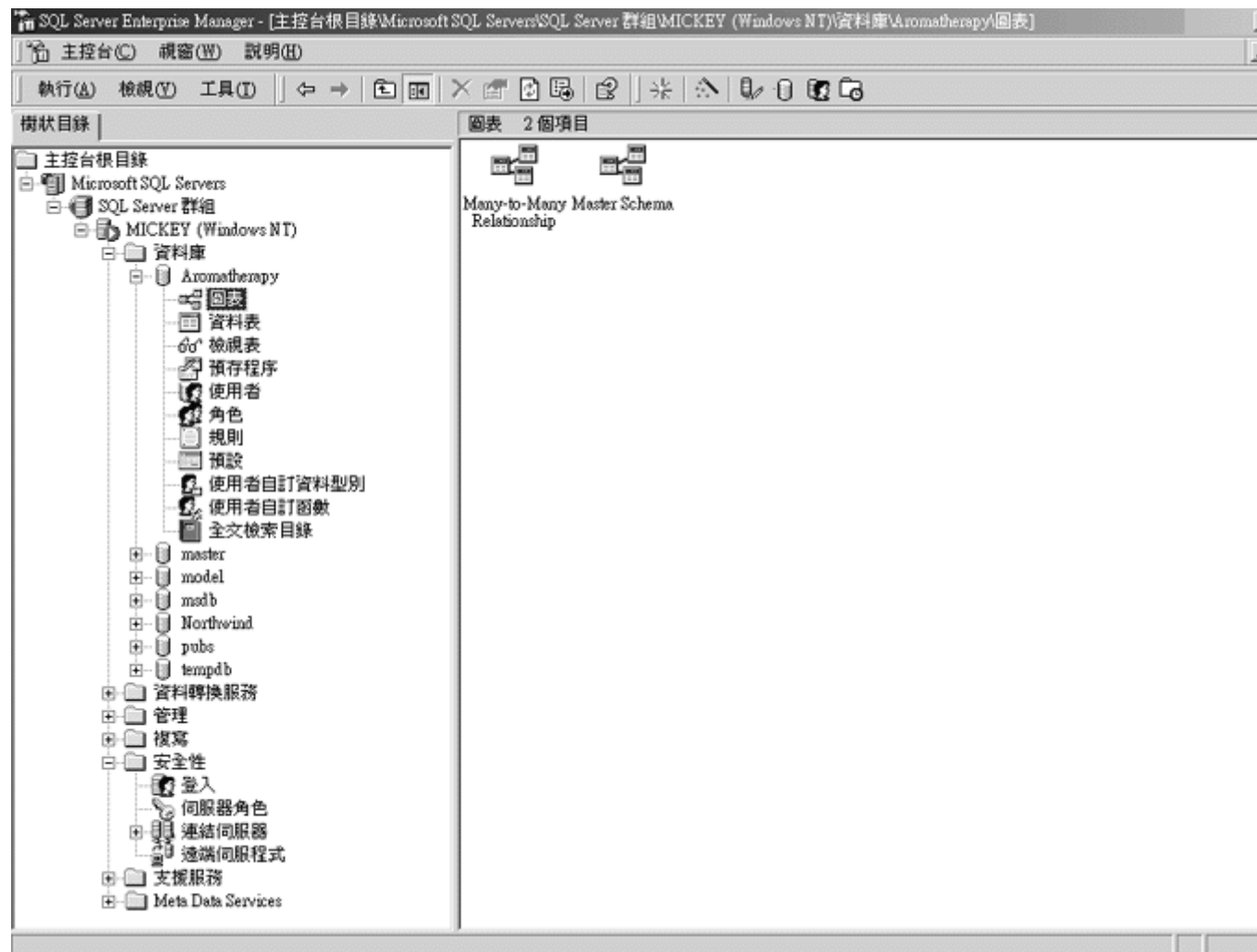
虽然我们可能可以从 [数据库图表](#) 窗口建立整个数据库结构描述，但是一般我们都会从现存的数据表中建立图表。[建立数据库图表精灵](#) 可以很容易地帮您达成这个愿望—您只需要在图表中加入您想要加入的资料表，然后接下来的工作您就可以给 [建立数据库图表精灵](#) 来处理了。

只要您曾经使用 [建立数据库图表精灵](#) 建立过数据库图表，那您就可以新增和移除数据表，并且还可以变更每一个资料表的显示明细程度。

建立一个数据库图表

1. 按一下 Aromatherapy 数据库内的 [图表](#) 数据夹。

Microsoft SQL Server 将会在详细资料窗格内显示已经存在的图表。

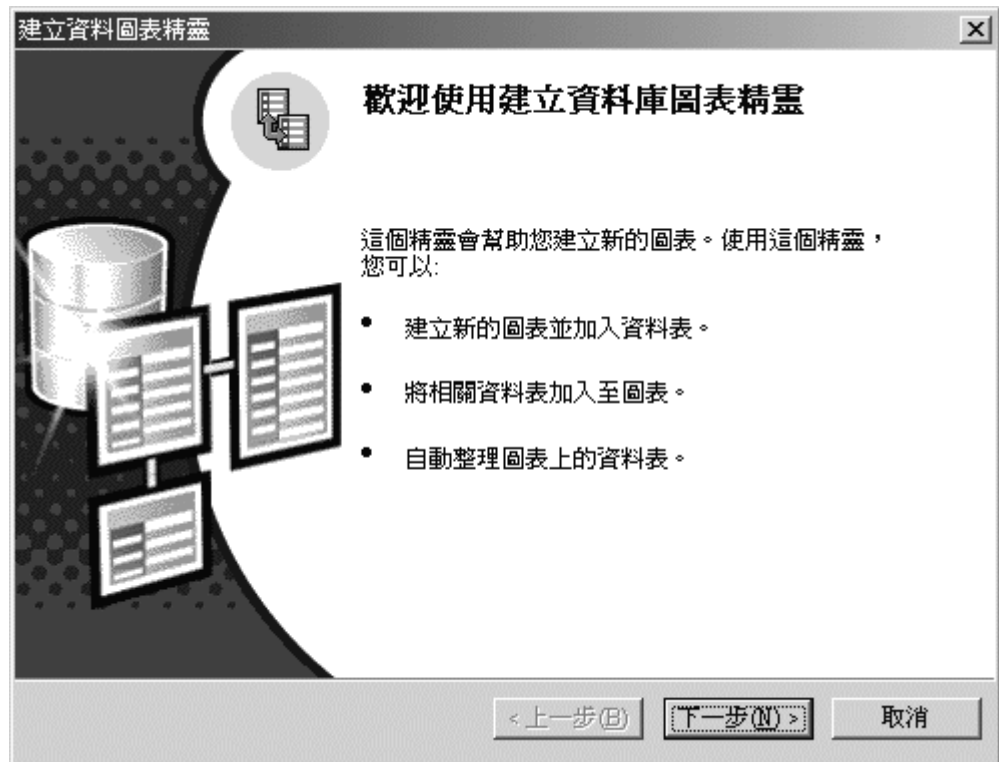


2. 按一下 **新增** 按钮。



新增按钮

此时 SQL Server 会显示 [建立数据库图表精灵](#) 的第一页画面。

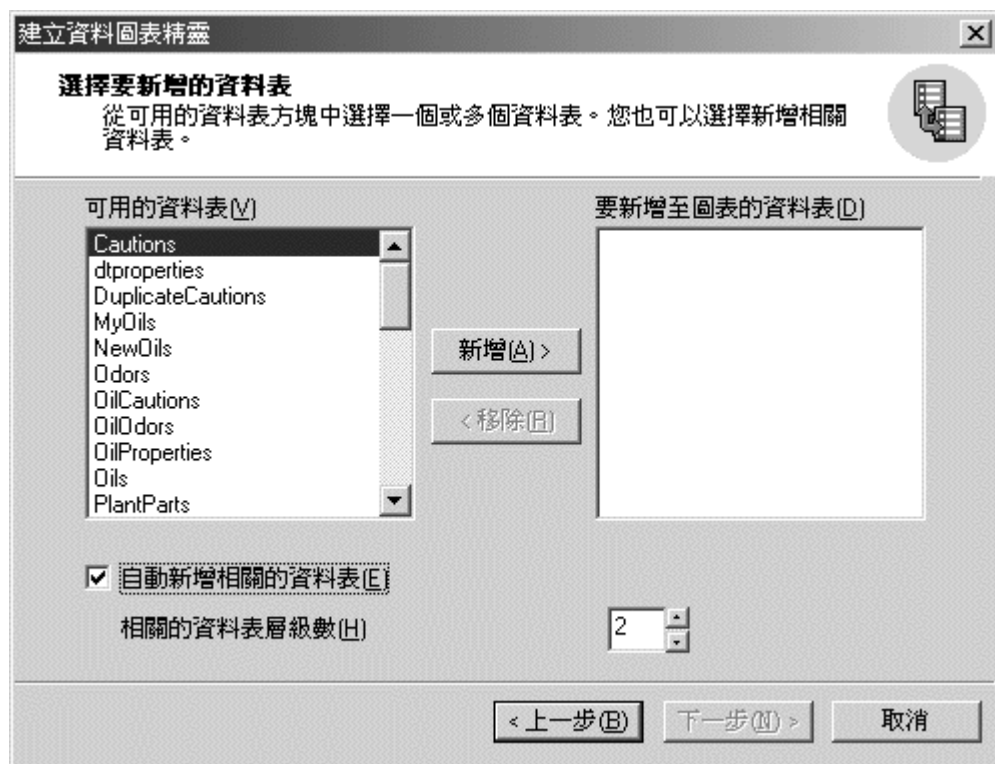


3. 按一下 [下一步](#) 按钮。

此时 [建立数据库图表精灵](#) 会询问您要将哪些数据表新增至数据库图表内。

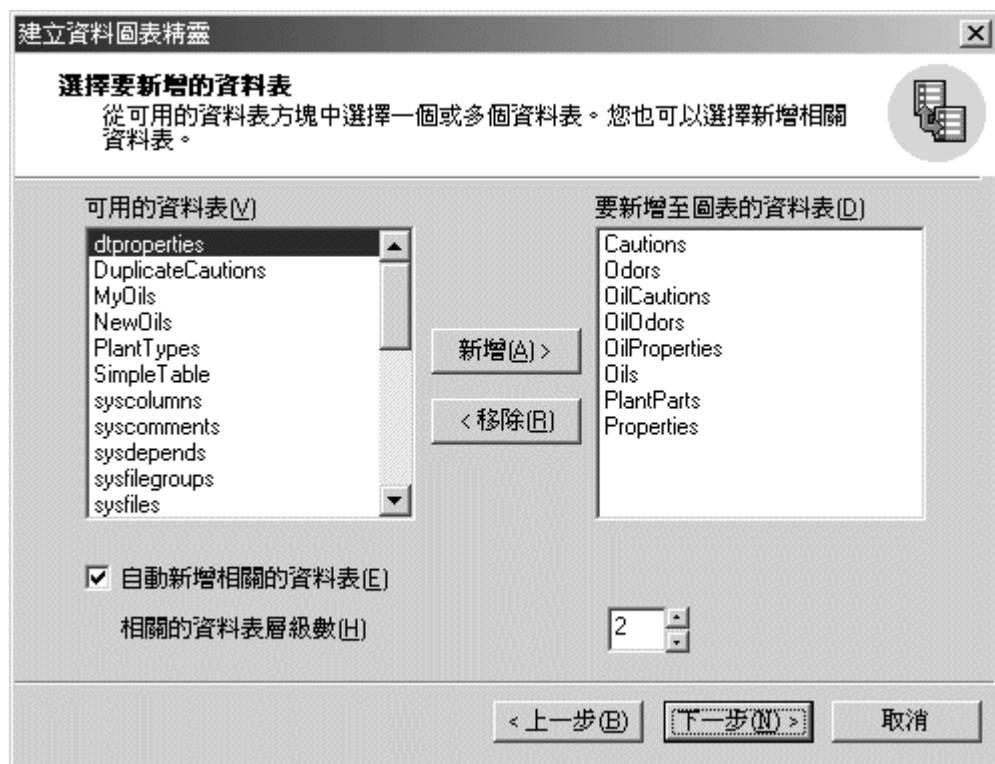


4. 选取 **自动新增相关的数据表** 复选框，并且将 **相关的数据表层级数** 项目设定为 **2**。



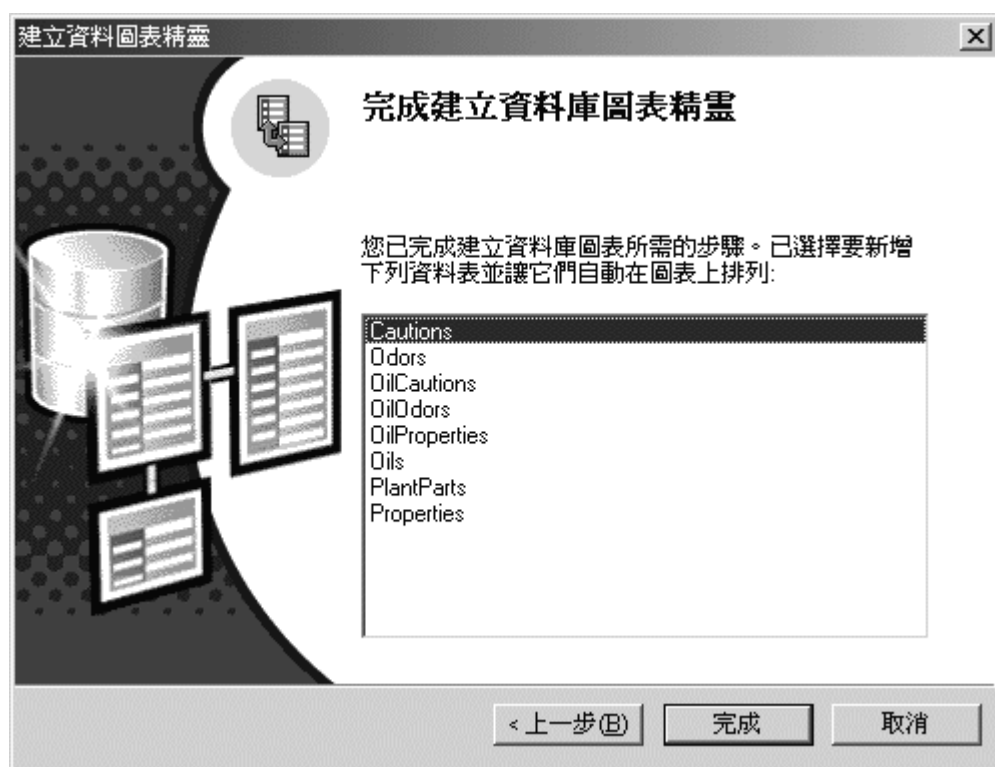
5. 在 [可用的数据表](#) 清单中选取 **Oils**，然后按一下 [新增](#) 按钮。

此时 [建立数据库图表精灵](#) 会加入 **Oils** 数据表，并且将与此数据表相关的数据表加入至图表中。



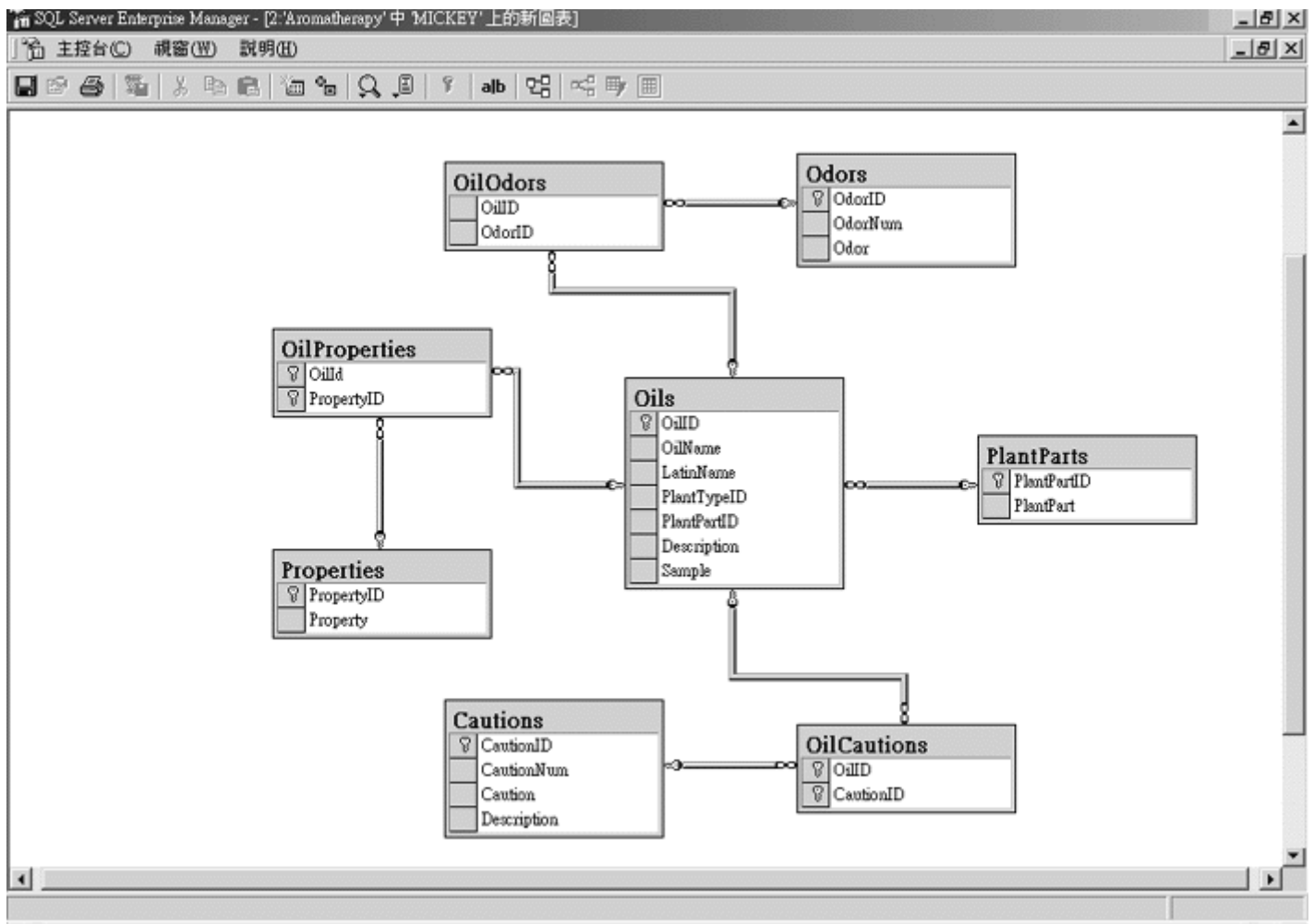
6. 按一下 [下一步](#) 按钮。

此时 [建立数据库图表精灵](#) 会询问您是否已经将您所选取的数据表都加入到数据库图表。



7. 按一下 **完成** 按钮。

此时 **建立数据库图表精灵** 会建立此图表。



提示

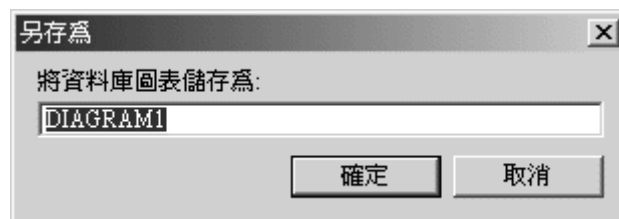
当 SQL Server 建立一个数据库图表之后，它会在每个数据表之间透过线条来建立它们之间的关联性，您可以藉由选取和拖曳将这些关联性重新排列。举例来说，如果您想要让图表内所显示的关联性能够更清楚的表现时，您就可以将这些线条重新排列。

-
8. 按一下 **存盘** 按钮。



存盘按钮

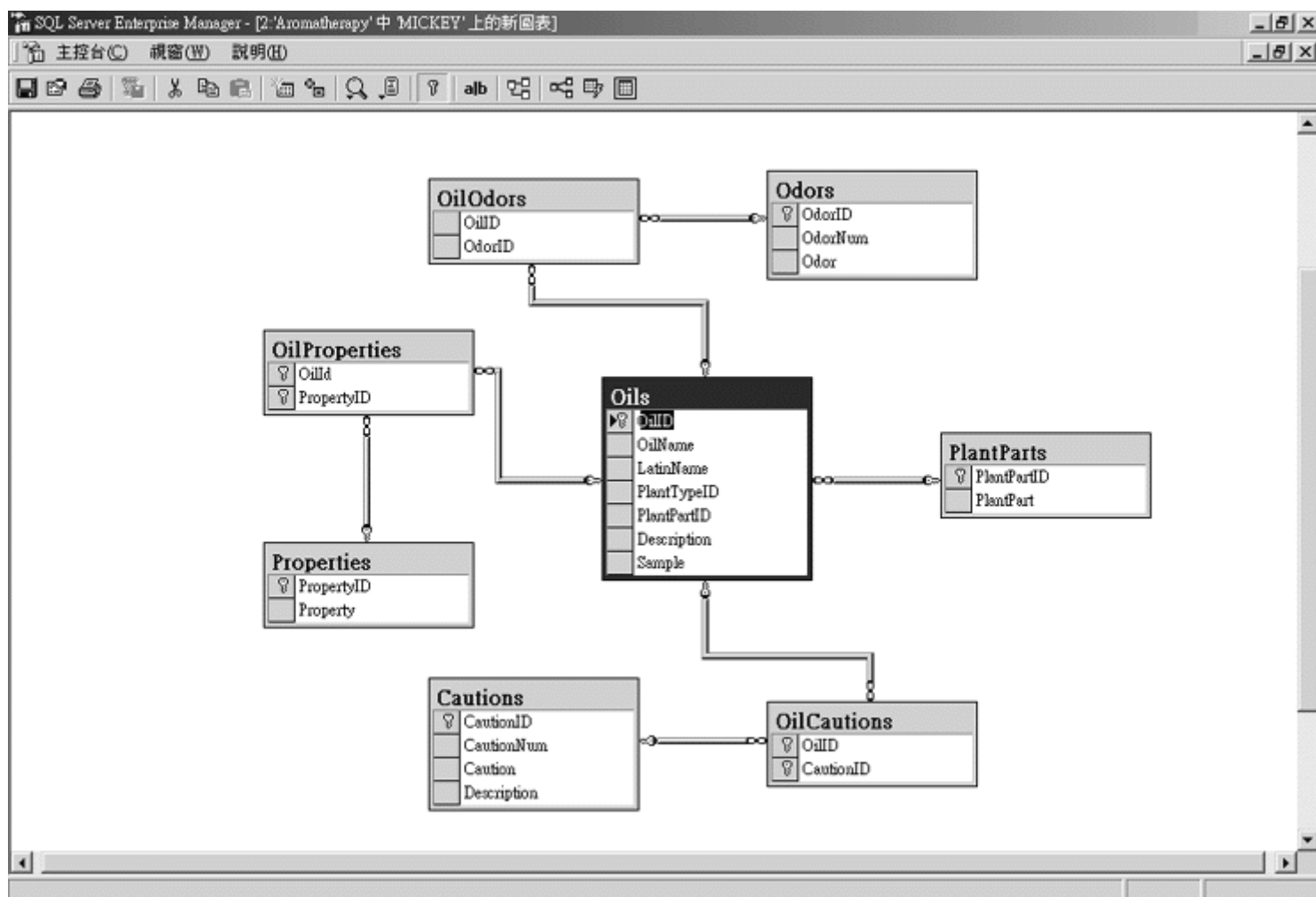
SQL Server 会显示一个对话框来要求您指定此图表的名称



9. 输入 **Lesson 10** 来作为此图表的名称，然后按一下 **确定** 按钮。

在数据库图表中变更显示的明细程度

1. 按一下图表中 **Oils** 数据表的名称以便在数据库图表中选取 **Oils** 数据表。

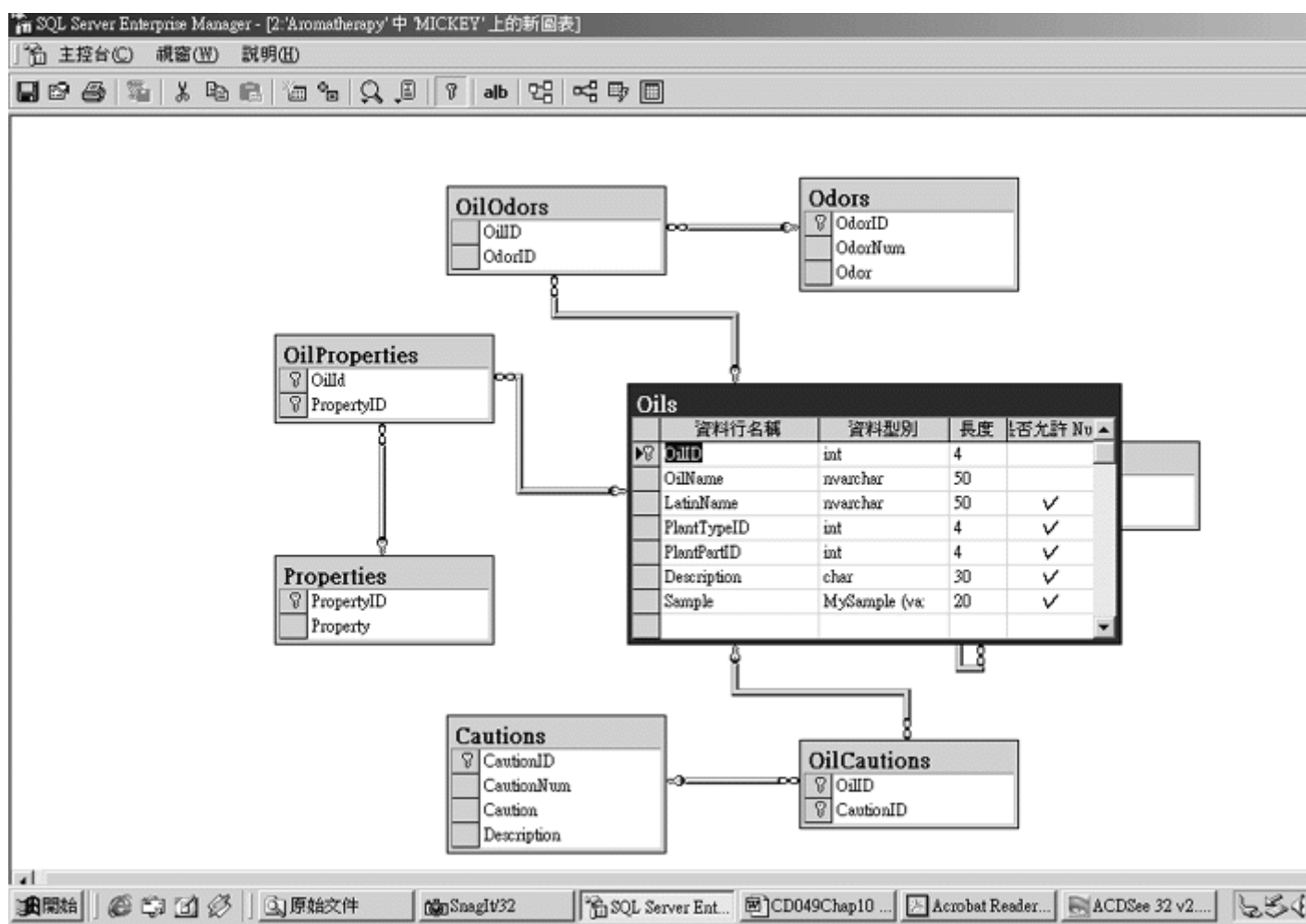


2. 当您按一下 **数据库图表** 窗口工具列上的 **显示** 按钮时，从下拉式清单中选择 **标准** 项目。



显示...按钮

SQL Server 会针对 Oils 数据表加入数据类型、长度以及是否允许 Null 等项目。

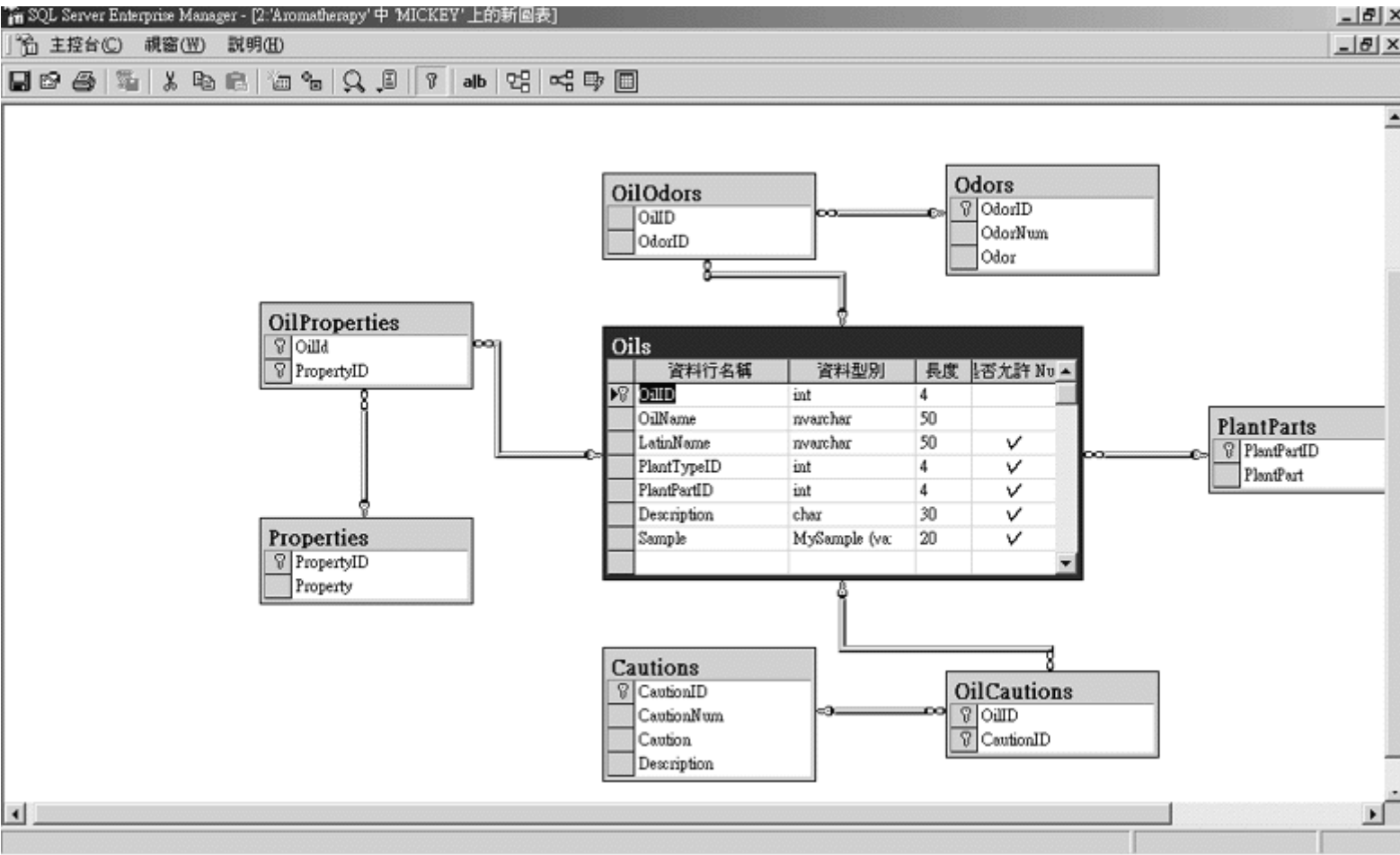


3. 在 **数据库图表** 窗口的工具列上按一下 **排列数据表** 按钮。



排列数据表按钮

此时 **SQL Server** 会重新排列数据库图表以符合 **Oils** 显示程度所需的空间。



4. 按一下 [存盘](#) 按钮。



存盘按钮

此时 SQL Server 会将此新的图表配置进行存盘。

在数据库图表中新增已经存在的数据表

1. 在数据库图表中按一下 [在图表上加入数据表](#) 按钮。



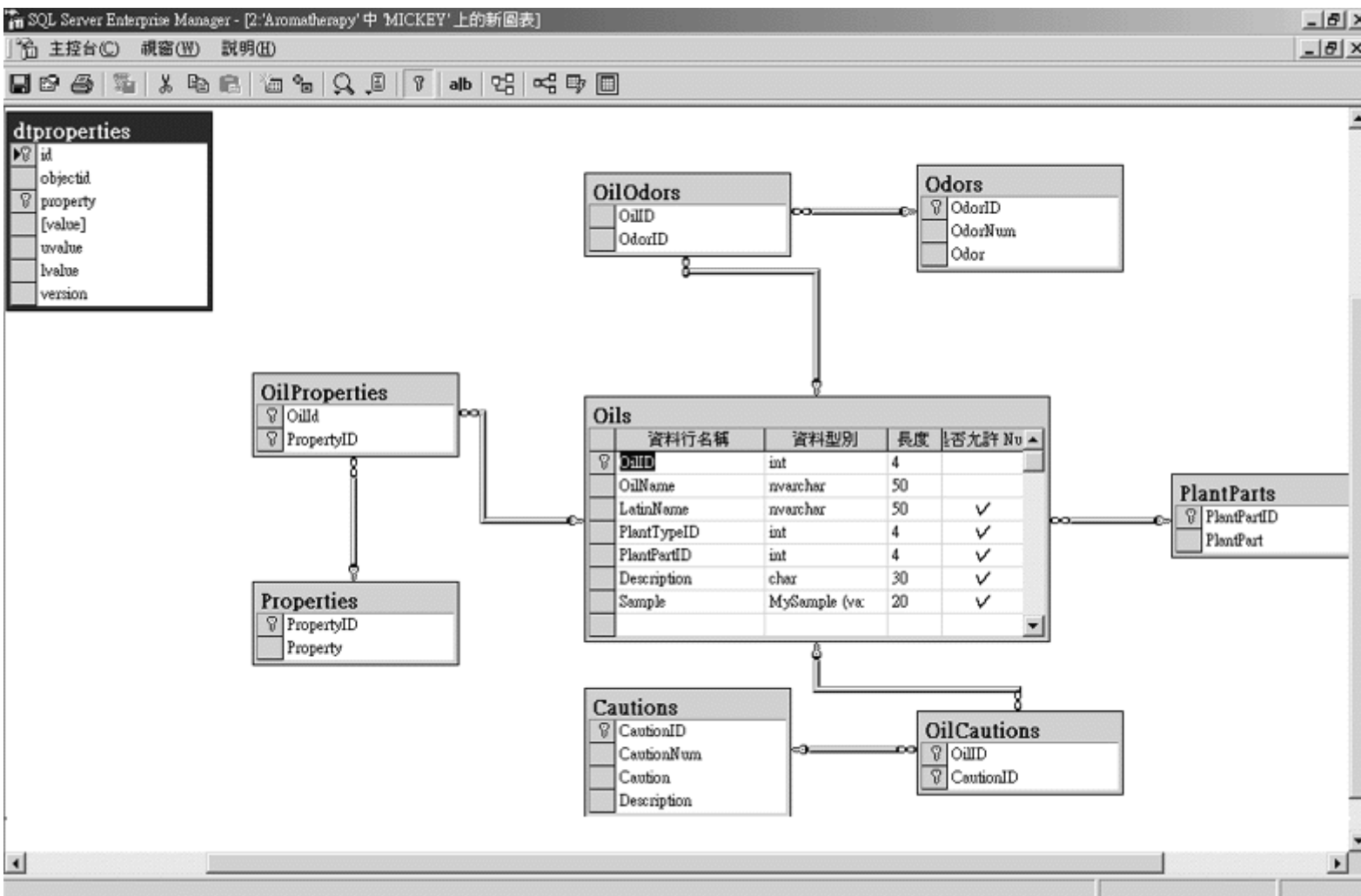
在图表上加入数据表按钮

此时 SQL Server 会显示一个 [加入数据表](#) 的对话框。



2. 在清单中选取 **dtproperties** 数据表，然后按一下 **新增** 按钮。

SQL Server 会将此数据表加入至图表中，由于 **dtproperties** 数据表是属于系统数据表，与 **Aromatherapy** 数据库没有任何关联，因此 SQL Server 并不会在图表中加入任何的关联性。



3. 按一下 [关闭](#) 按钮以将 [加入数据表](#) 的对话框关闭,然后在 [数据库图表窗口](#) 的工具列上按一下 [存盘](#) 按钮。



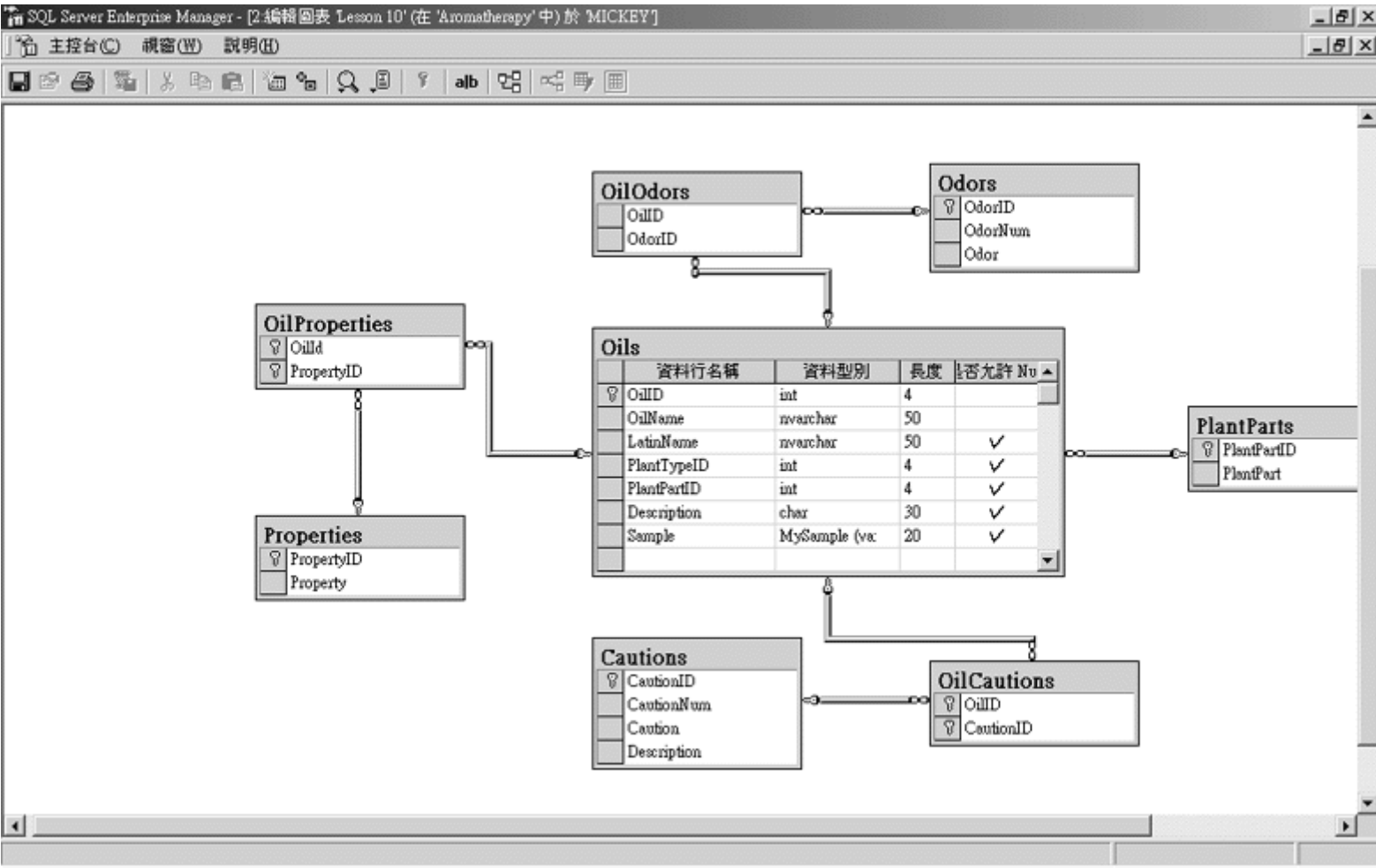
存盘按钮

SQL Server 会储存包含新数据表的图表。

从数据库图表中移除一个数据表

1. 在 [数据库图表](#) 窗口中的 `dtproperties` 数据表上按右钮,并且在快捷菜单中选择 [从图表中移除数据表](#) 。

SQL Server 会自图表中移除此数据表。



2. 按一下 [存盘](#) 按钮。



存盘按钮

此时 SQL Server 会储存此图表。

使用数据库图表以维护数据库

SQL Server Enterprise Manager 的 [数据库图表](#) 窗口也可以让您在它的图形化环境中维护数据库结构描述。您可以新增数据表、变更数据表以及在数据表之间维护其关联性。

变更数据库结构描述

[数据库图表](#) 窗口最好用的功能之一就是，在图表本身之中就可以更改数据库结构描述。针对您数据库中的数据表及其关联性，图形化显示方式是一项绝佳的工具，因为它可以让您在直接更改数据库结构描述方面更为方便。

在数据库图表窗口中新增数据表的数据行

1. 放大数据库图表中所显示的 **Oils** 资料表，直到可以将 **Oils** 数据表内的所有数据行都可以完整地显示，并且看见空白的资料列为止。

Oils				
	資料行名稱	資料型別	長度	是否允許 Null
PK	OilID	int	4	
	OilName	nvarchar	50	
	LatinName	nvarchar	50	✓
	PlantTypeID	int	4	✓
	PlantPartID	int	4	✓

- 按一下 **数据行名称** 字段的第一个空白储存格，并且新增一个新的字段 **Description**，将数据型态设定为 **varchar**、长度设定为 **50**。

Oils *				
	資料行名稱	資料型別	長度	是否允許 Null
PK	OilID	int	4	
	OilName	nvarchar	50	
	LatinName	nvarchar	50	✓
	PlantTypeID	int	4	✓
	PlantPartID	int	4	✓
	Description	varchar	50	✓

提示

请注意在 **Oils** 数据表的名称旁有一个星号(*)，**SQL Server** 使用星号来标示在 **数据库图表** 窗口中任何您已经变更，但是尚未储存的记号。**SQL Server** 会在您储存数据库图表时，才真正地变更数据库结构描述。所以您可以使用 **数据库图表** 来测试任何变更。如果您最后决定不进行变更，您只需要关闭 **数据库图表** 窗口，然后不储存它即可。

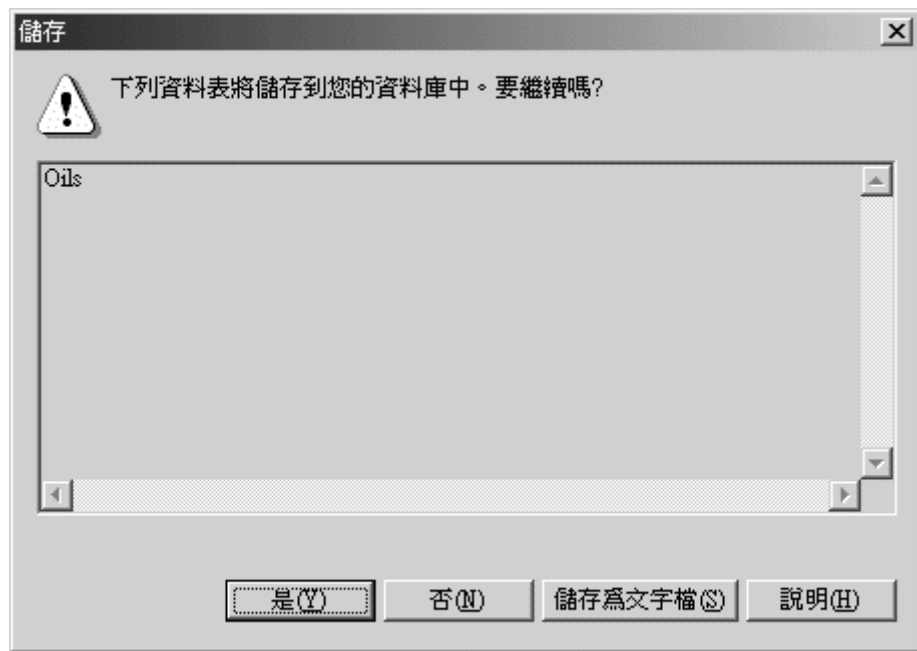
-
3. 按一下 [排列数据表](#) 按钮以便让 SQL Server 重新排列窗口。



排列数据表按钮

4. 按一下 [存盘](#) 按钮。

SQL Server 会询问您是否要将已经改变的 Oils 数据表储存起来。



5. 按一下 [是](#) 按钮。

SQL Server 会新增 Oils 数据表中的新的数据行名称，并且会在 Oils 数据表中移除星号 (*)。

建立数据库对象

除了可以让您变更现存的数据库对象之外，[数据库图表](#) 窗口也可以让您新增数据表和建立关联性。

在数据库图表窗口中建立数据表

1. 在 [数据库图表](#) 窗口中的空白位置上按右键，并且在快捷菜单中选择 [新增数据表](#)。

SQL Server 会显示 [选择名称](#) 对话框。



提示

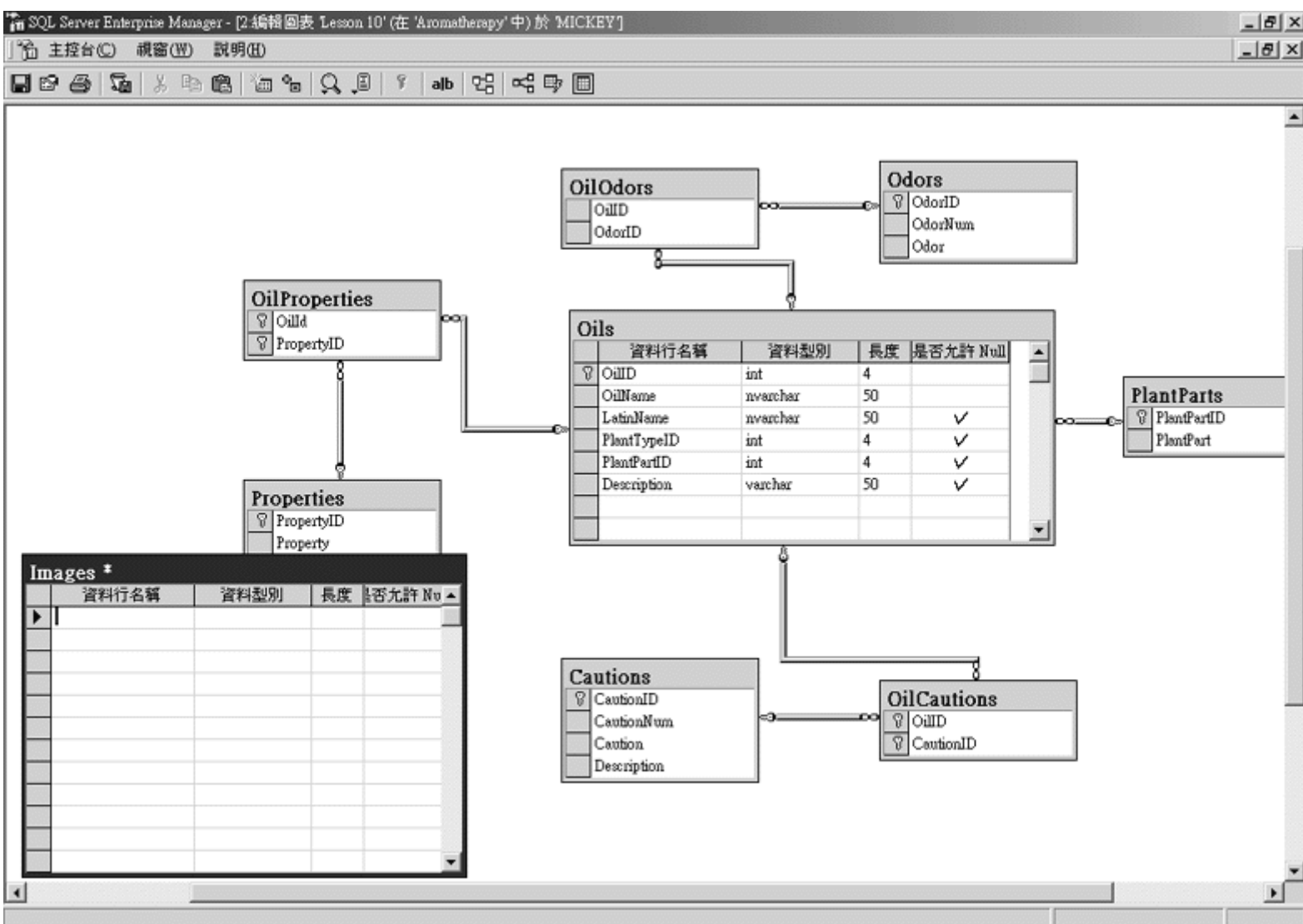
您也可以藉由按一下在 [数据库图表](#) 窗口工具列上的 [新增数据表](#) 按钮以新增数据表。

-
2. 输入『Images』来作为新数据表的名称，然后按一下 [确定](#) 按钮。



新增数据表按钮

SQL Server 会将此数据表加入至图表中。



3. 将下表所示的数据行加入到新数据表中:

数据行名称	资料型别	长度	是否允许 Null
-------	------	----	-----------

OilID	Int	4	No
Picture	Image	16	No

4.

Images *				
	資料行名稱	資料型別	長度	是否允許 Null
	OilID	int	4	✓
▶	Picture	image	16	✓

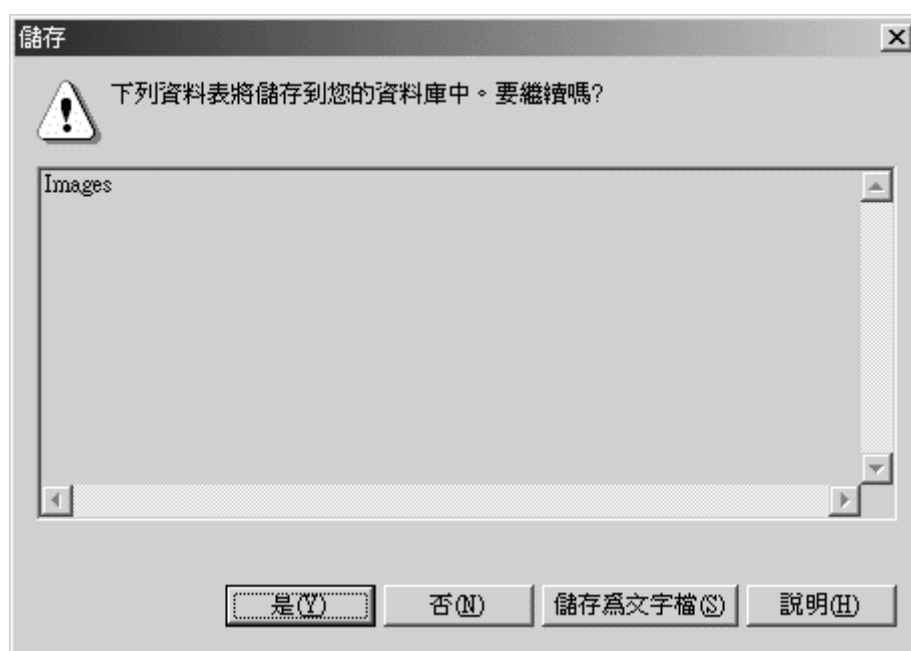
5.

6. 选取 OilID 数据行名称左侧的方块，然后在 [数据库图表](#) 窗口工具列上按一下 [设定主索引键](#) 按钮。



设定主索引键按钮

SQL Server 会显示一个对话框来询问您是否要将已经改变的数据库结构描述进行储存。



9. 按一下 **是** 按钮。

SQL Server 会将此新数据表加入至数据库中。

在数据库图表窗口中建立一个关联性

1. 在数据库图表内的 Oils 数据表中选取 **OiIID** 数据行，并且将它拖曳至 Images 数据表中的 OiIID 数据行。

SQL Server 会开启 [建立关联性](#) 对话框。

提示

您也可以在这个对话框中变更任何您认为需要变更的关联性。

2. 按一下 [确定](#) 按钮。

SQL Server 会将此对话框关闭。

提示

请注意在 **Oils** 和 **Images** 数据表之间的关联性线条，在该关联性线条的二端各有一把小钥匙，这表示其关联性是一对一的，因为双方都是使用主索引键建立关联性。而其它的关联性线条中有无限符号（ ∞ ），无限符号表示该数据表是属于端
联性中 **多** 的一方。



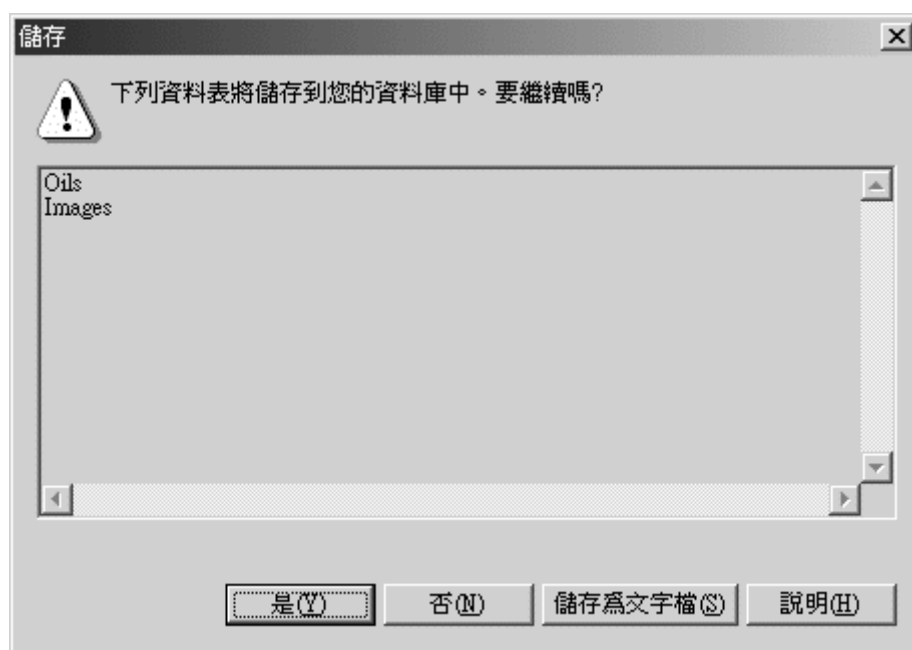
一对一联结

-
- 按一下 **存盘** 按钮。



存盘按钮

SQL Server 会显示一个对话框来询问您是否要将已经改变的数据库结构描述储存。



4. 按一下 [是](#) 按钮。

SQL Server 会储存此图表，并且更新该数据库结构描述。

本章总结

要执行的工作	执行	按钮
建立数据库图表	按一下数据库中的 图表 数据夹，按一下 新增 按钮，并且按照 建立数据库图表精灵 的步骤执行。	
在数据库图表中变更显示的明细程度	选取您想要变更的资料表，然后按一下 显示 按钮，并且选取您所需的明细程度。	
在数据库图表中新增已经存在的数据表	按一下 在图表上加入数据表 按钮，并且在清单中选取您想要加入的数据表名称，然后按一下 确定 按钮。	
从数据库图表中移除数据表	在数据表上按右键，然后在快捷菜单中选取 从图表中移除数据表 。	
在数据库图表窗口中新增字段至数据	以标准检视的方式来显示数据表，并且在数据表中加入新的表数据行。	
在数据库图表窗口中建立数据表	在 数据库图表 窗口中空白的位置按右键，并且在快捷菜单中选取 新增数据表 。	
在数据库图表窗口中建立关联性	从主索引键数据表中拖曳一个字段至外部索引键数据表，并且完成 建立关联性 对话框中的设定。	

第四篇 数据处理

[17. 新增资料列](#)

- . 认识 INSERT 陈述式
- . 使用 INSERT 陈述式
- . 本章总结

[18. 更新数据列](#)

- . 认识 UPDATE 陈述式
- . 使用 UPADTE 陈述式
- . 本章总结

[19. 删除数据列](#)

- . 认识 DELETE 陈述式
- . 使用 DELETE 陈述式
- . 使用 TRUNCATE TABLE 陈述式
- . 本章总结

[20. 复制及移动数据](#)

- . 数据转换服务精灵
- . 附加及卸离数据库
- . 复制数据库精灵
- . 本章总结

17. 新增资料列

在本章中，您将学习到：

- 在方格窗格中使用 **INSERT** 陈述式来插入数据列。
- 在 **SQL** 窗格中使用 **INSERT** 陈述式来插入数据列。
- 指定所有数据行以插入数据列。
- 使用 **DEFAULT** 和 **NULL** 值插入数据列。
- 在方格窗格中使用 **INSERT** 陈述式来插入多个数据列。
- 在 **SQL** 窗格中使用 **INSERT** 陈述式来插入多个数据列。

在第 11 章中，您已经学会如何使用查询设计师以在结果窗格中输入新值到数据表。在本章中，

您将学习如何使用 **Transact-SQL INSERT** 陈述式以程控的方式将数据列新增至数据表中。

认识 INSERT 陈述式

INSERT 陈述式的语法与 SELECT 陈述式相似，它的基本语法如下所示：

```
INSERT [INTO ] table_or_view [(column_list)]  
  
VALUES (value_list)
```

每一个 INSERT 陈述式可以更新一个数据表或检视表。当您使用 INSERT 陈述式来更新检视表时，您必须要遵守下列的原则：

- 检视表必须不包含汇总函数，例如 COUNT 或 AVG 函数。
- 检视表必须不包含 TOP、GROUP BY、UNION 或 DISTINCT 关键词。
- 检视表必须不包含计算的数据行。
- 检视表必须在 FROM 子句中引用一个数据表。
- INSERT 陈述式更新的数据行只能来自单一的资料表。

在 **INSERT** 陈述式中的数据行清单 (**column_list**) 是选择性的。如果您没有提供数据行清单时，**INSERT** 陈述式必须要包含在数据表或检视表中所有数据行的值，并且数据行清单的顺序必须要跟数据表或检视表中所定义的数据行顺序相同。然而，您可以使用 **DEFAULT** 关键词以指定某一个数据列的默认值。

如果您提供了数据行清单时，其格式和 **SELECT** 陈述式中的数据行清单一样：是以逗号「，」作为资料行的分隔符。

使用 **INSERT** 陈述式

INSERT 陈述式可以使用方格窗格来指定数据行或者是使用 **SQL** 窗格直接输入陈述式加以建立。

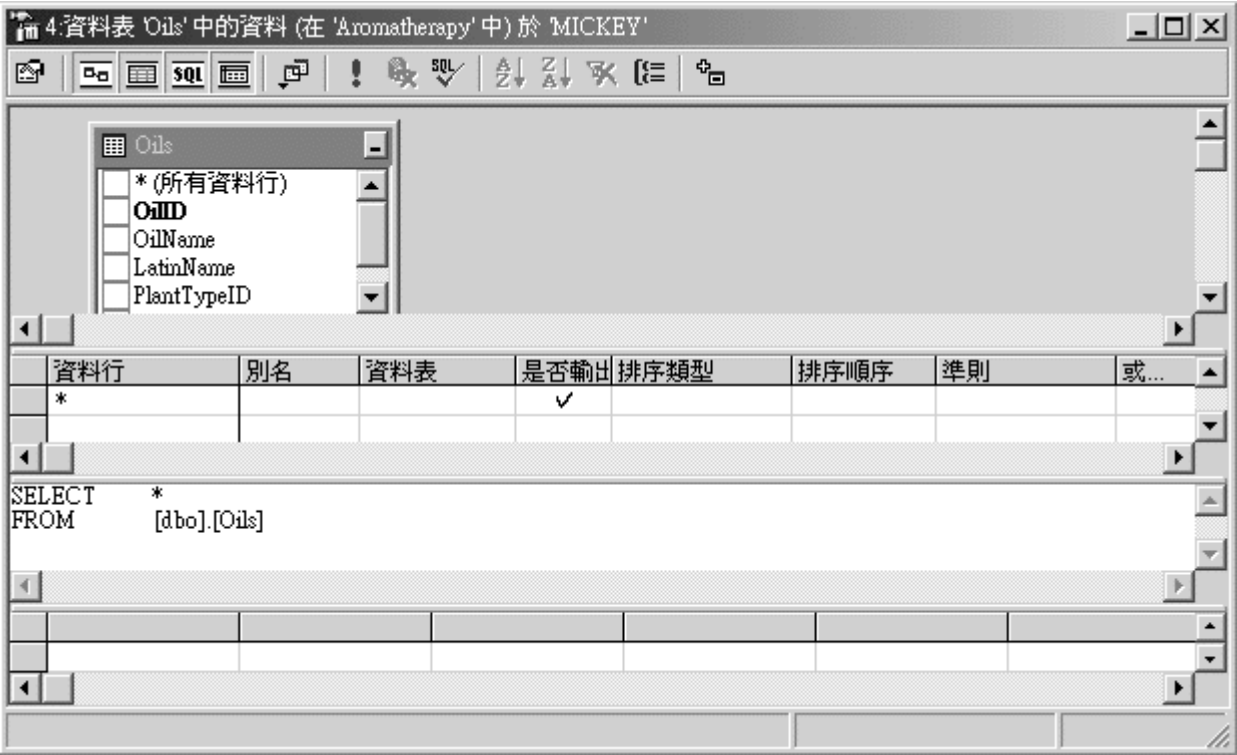
使用方格窗格插入数据列

在方格窗格中建立一个 **INSERT** 陈述式是一个比较简单的方式，这是因为它并不会要求您记住任何的语法。

使用方格窗格中插入数据列

1. 在 Aromathrtapy 数据库中 [详细数据窗格](#) 的 **Olis** 数据表上按右钮，并指向 [开启数据表](#) 菜单中的 [查询](#)。

此时，[查询设计师](#) 会开启包含四个窗格的窗口。



提示

在 [开启数据表](#) 或 [开启检视](#) 中的 [查询](#) 指令是一个提供您快速开启 [查询设计](#)

[师](#) 并且将所有的窗格开启的方式。虽然预设的 SQL 陈述式是 `SELECT * FROM`

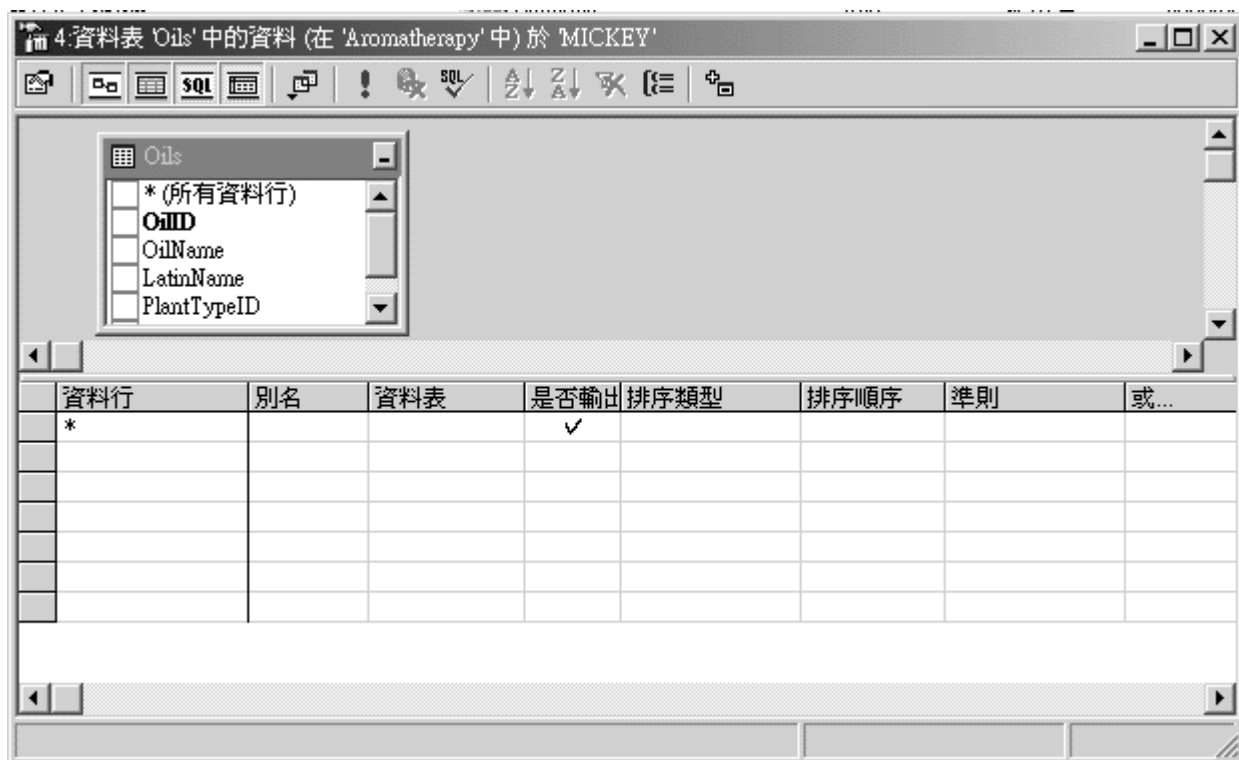
`<table_or_view>`，但是查询不会执行，因此也不会有任何的数据列被传回。

-
2. 将 **SQL 窗格** 和 **结果窗格** 隐藏。



显示/隐藏 SQL 窗格按钮

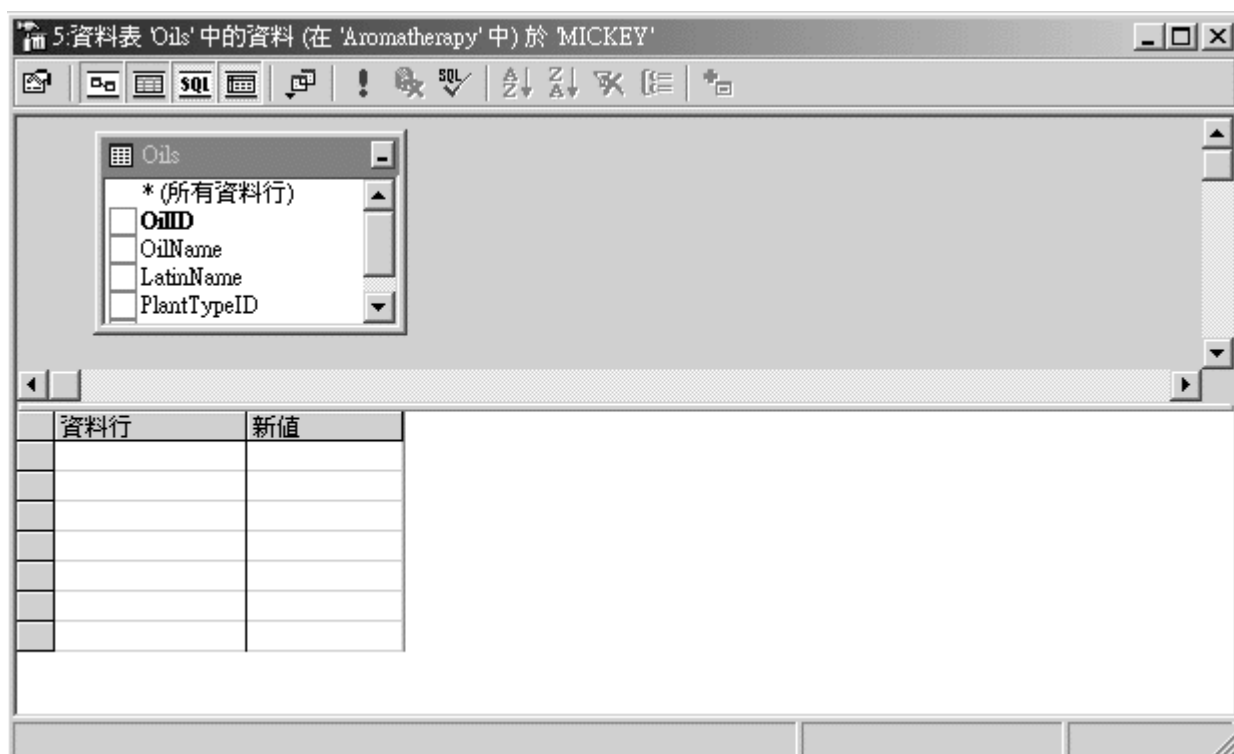




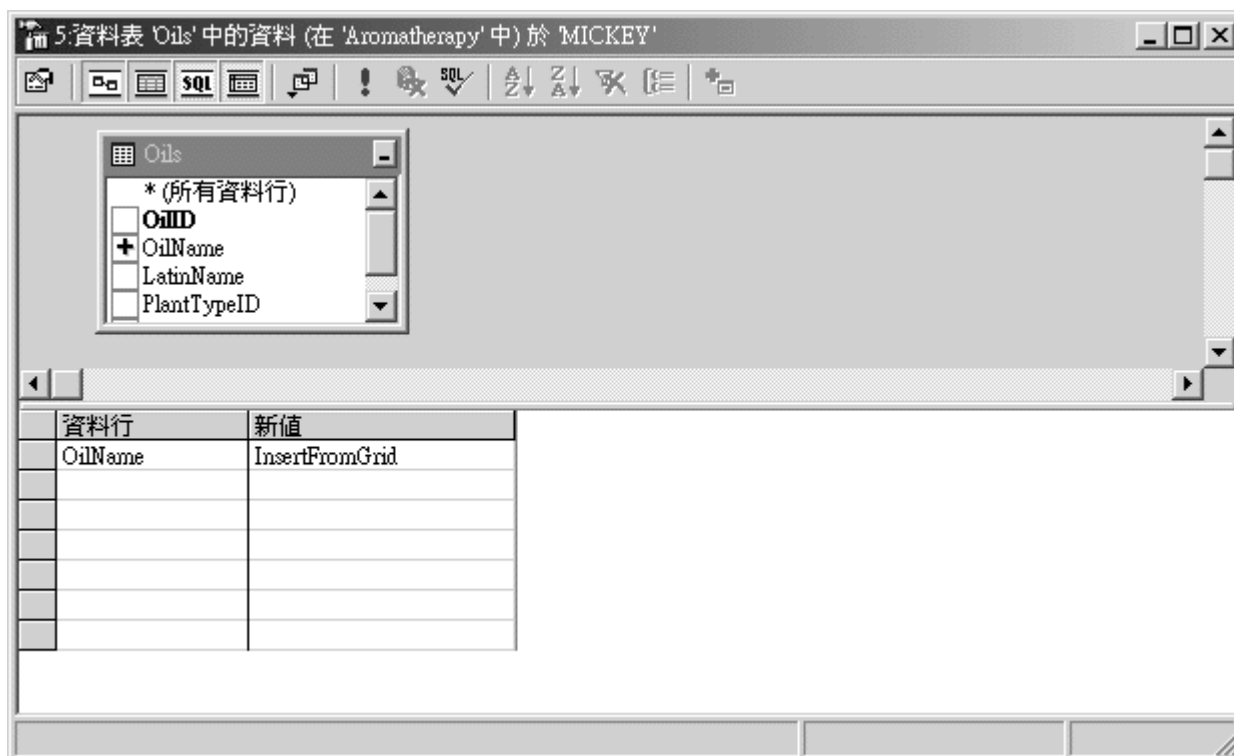
- 在 **查詢設計師** 的工具列上，按一下 **變更查詢類型** 按鈕，並在清單中選擇 **INSERT INTO** 項目。



變更查詢類型按鈕



4. 將 **OilName** 数据行新增到 **方格窗格** 内，并且将 **新值** 数据行的值设定为 **nsertFromGrid** 。



5. 在 [查询设计师](#) 的工具列上，按一下 [执行](#) 按钮，以便执行此查询。



执行按钮

[查询设计师](#) 会显示已经将数据列插入到数据表的讯息。



6. 按一下 [确定](#) 按钮，以便将此对话框关闭，但不用将 [查询设计师](#) 关闭。在

Enterprise Manager [详细数据窗格](#) 中的 [Oils](#) 资料表上按右钮，并在 [开启数据表](#) 项目的子菜单内选择 [传回所有数据列](#) 项目。

此时，会出现另一个新的 [查询设计师](#)，在此 [查询设计师](#) 内将会显示 [Oils](#) 数据表内的所有数据列的数据。

5. 資料表 'Oils' 中的資料 (在 'Aromatherapy' 中) 於 'MICKEY'

OilID	OilName	LatinName	PlantTypeID	PlantPartID
1	Basil	Ocimum Basilicum	1	1
2	Bergamot	Citrus Bergamia	2	2
3	Black Pepper	Piper Nigrum	3	3
4	Cedarwood	Cedrus Atlantica, Jr	2	4
5	German Chamomile	Matricaria Chamom	1	5
6	Roman Chamomile	Anthemis Nobilis	1	5
7	Cinnamon	Cinnamomum Zeyl	4	11
8	Citronella	Cymbopogon Nard	4	11
9	Clary Sage	Salvia Sclarea	1	5
10	Coriander	Coriandrum Sativum	1	7
11	Cypress	Cupressus Semperv	2	1
12	Eucalyptus	Eucalyptus Globulu	2	1
13	Frankincense	Buswellia Thurifera	2	9
14	Geranium	Pelargonium Grave	5	1
15	Ginger	Zingiber Officinale	5	10
16	Grapefruit	Citrus Paradisi	2	2
17	Jasmine	Jasminum Officinale	5	5
18	Juniper	Juniperus Commun	5	3
19	Lavender	Lavandulus Officin	5	5

7. 卷动 [查询设计师](#) 内的滚动条至最尾端, 您可以在最后一笔数据列中发现所插入的
资料列。

5:資料表 'Oils' 中的資料 (在 'Aromatherapy' 中) 於 'MICKEY'

	OilID	OilName	LatinName	PlantTypeID	PlantPartID
	37	Sandalwood	Santalum Album	2	4
	38	Tea Tree	Melaleuca Alternifo	2	1
	39	Thyme	Thymus Vulgaris	1	1
	40	Vetivert	Vetiveria Zizanoide	4	10
	41	Ylang-ylang	Cananga Odorata	2	5
	42	Fennel	Foeniculum Vulgar	5	14
	43	Rose Otto	Rosa Damascena	5	5
	44	Sage	Salvia Officinalis	1	1
	45	Vanilla Oleoresin	Vanilla Planifolia	2	14
	46	Benzoin	Styrax Benzoin	2	9
	47	Linden	Tilia Vulgaris	2	5
	48	InsertFromGrid	<NULL>	<NULL>	<NULL>
*					

重要

在您数据库中的 OilID 的内容可能与书上的不相同。请记得 OilID 被定义为一个识别数据行，在 SQL Server 中的识别数据行是用来确保此数据行的内容是唯一性的。

使用 SQL 窗格中插入数据列

1. 重新选取 **查询设计师** 窗口以包含 INSERT 陈述式。
2. 将 **方格窗格** 和 **图表窗格** 隐藏，并显示 **SQL 窗格**。



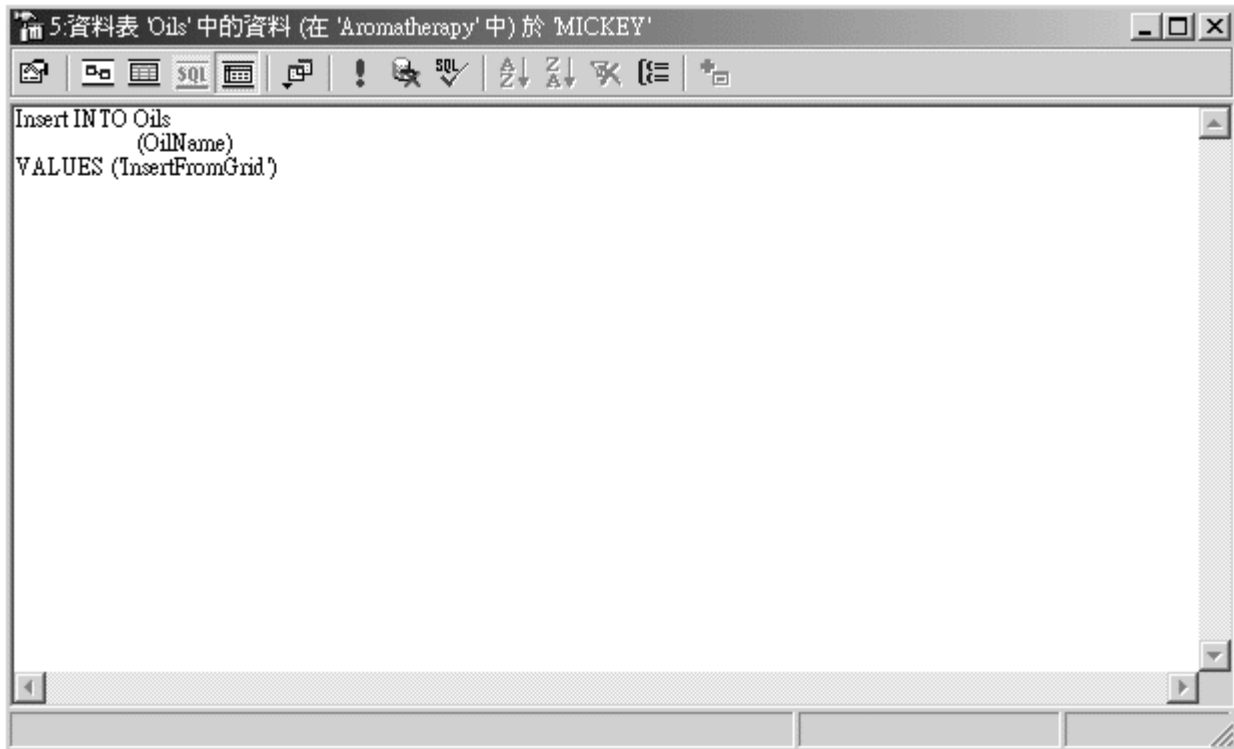
显示/隐藏方格窗格按钮



显示/隐藏图表窗格按钮



显示/隐藏 SQL 窗格按钮

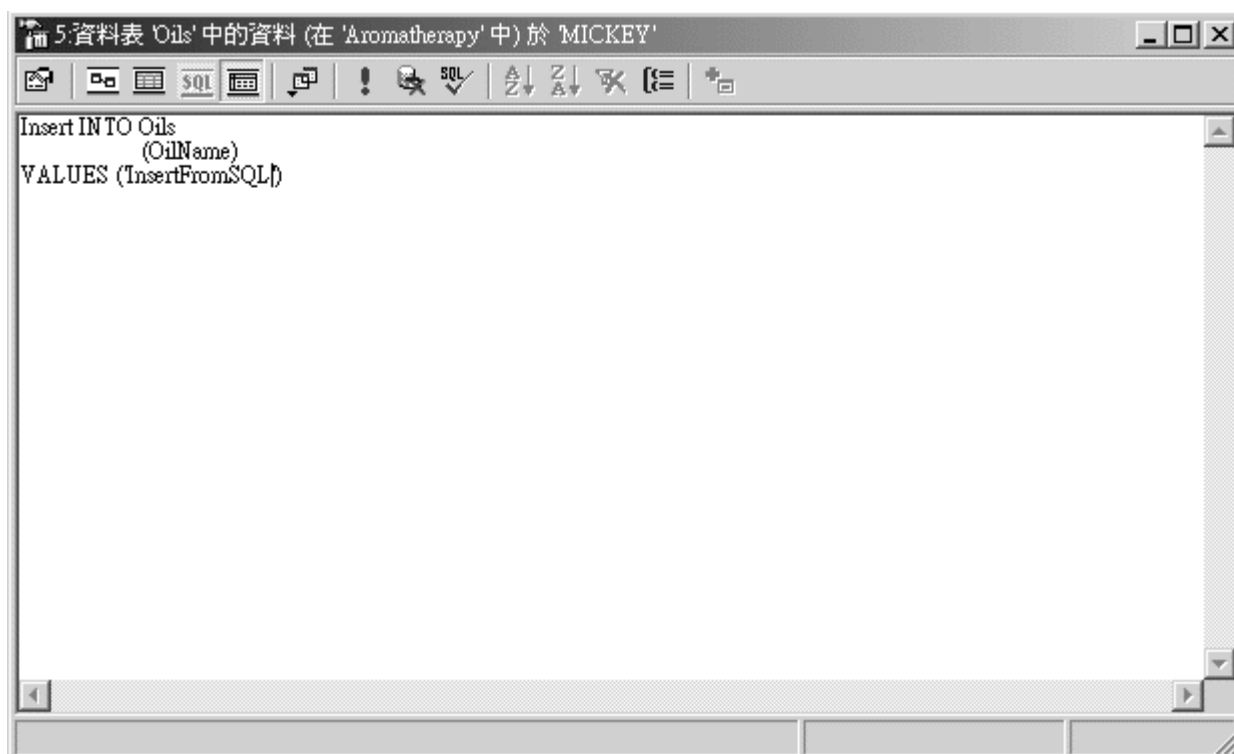


3. 将 SQL 陈述式内容更改为:

4. INSERT INTO Oils

5. (OilName)

VALUES (값 InsertFromSQL)



6. 在 [查询设计师](#) 的工具列上按一下 [执行](#) 按钮，以便执行此查询。



执行按钮

[查询设计师](#) 会显示已经将一行数据列插入到数据表的讯息。



7. 按一下 **确定** 按钮，以便关闭此消息框。将 **查询设计师** 的窗口切换到另一个包含 **Oils** 数据表内所有数据列的窗口。
8. 在 **查询设计师** 内的工具列上按一下 **执行** 按钮，以便执行 **SELECT *** 查询，并且卷动 **查询设计师** 内的滚动条至最尾端，您可以在最后一笔数据列中发现所插入的资料列。



执行按钮

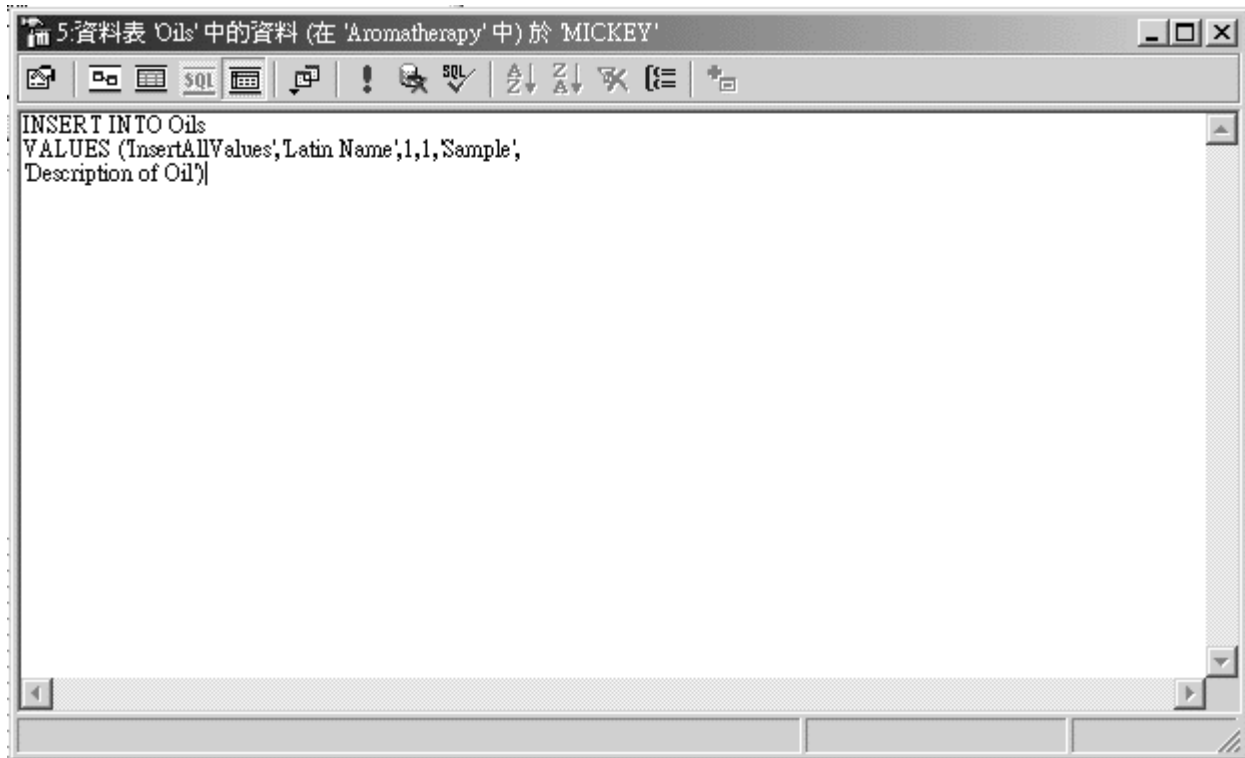
5. 資料表 'Oils' 中的資料 (在 'Aromatherapy' 中) 於 'MICKEY'

	OilID	OilName	LatinName	PlantTypeID	PlantPartID
	37	Sandalwood	Santalum Album	2	4
	38	Tea Tree	Melaleuca Alternifo	2	1
	39	Thyme	Thymus Vulgaris	1	1
	40	Vetivert	Vetiveria Zizanoide	4	10
	41	Ylang-ylang	Cananga Odorata	2	5
	42	Fennel	Foeniculum Vulgar	5	14
	43	Rose Otto	Rosa Damascena	5	5
	44	Sage	Salvia Officinalis	1	1
	45	Vanilla Oleoresin	Vanilla Planifolia	2	14
	46	Benzoin	Styrax Benzoin	2	9
	47	Linden	Tilia Vulgaris	2	5
	48	InsertFromGrid	<NULL>	<NULL>	<NULL>
▶	49	InsertFromSQL	<NULL>	<NULL>	<NULL>
*					

指定所有数据行以插入数据列

1. 将 [查询设计师](#) 的窗口切换到另一个包含 INSERT 陈述式的窗口。
2. 将 SQL 陈述式内容更改为:

```
3.      INSERT INTO Oils
4.      VALUES (값'InsertAllValues', 값'Latin Name', 1, 1, 값
' Sample',
' Description of the Oil')
```



重要

OiiID 数据行并没有包含在 SQL 陈述式内，这是因为它被定义为一个识别数据行。

SQL Server 将针对该识别数据行自动提供一个值，或是某数据行如果拥有

NewID()为默认值，SQL Server 也会自动为该数据行产生一个 GUID 值。

5. 在 **查询设计师** 的工具列上按一下 **执行** 按钮，以便执行此查询。



执行按钮

查询设计师 会显示已经将数据列插入到数据表的讯息。



6. 按一下 **确定** 按钮，以便关闭此消息框。将 **查询设计师** 的窗口切换到另一个包含 **Oils** 数据表内所有数据列的窗口。
7. 在 **查询设计师** 的工具列上按一下 **执行** 按钮，以便执行 **SELECT *** 查询，并且滚动 **查询设计师** 内的滚动条至最尾端，您可以在最后一笔数据列中发现所插入的资料列。



执行按钮

7. 资料表 'Oils' 中的资料 (在 'Aromatherapy' 中) 於 'MICKEY'							
OilID	OilName	LatinName	PlantTypeID	PlantPartID	Sample	Description	
34	Rose	Rosa Centifolia	5	5	<NULL>	<NULL>	
35	Rosemary	Rosmarinus Officin	1	1	<NULL>	<NULL>	
36	Rosewood	Aniba Rosaeodora	2	4	<NULL>	<NULL>	
37	Sandalwood	Santalum Album	2	4	<NULL>	<NULL>	
38	Tea Tree	Melaleuca Alternifo	2	1	<NULL>	<NULL>	
39	Thyme	Thymus Vulgaris	1	1	<NULL>	<NULL>	
40	Vetivert	Vetiveria Zizanoide	4	10	<NULL>	<NULL>	
41	Ylang-ylang	Cananga Odorata	2	5	<NULL>	<NULL>	
42	Fennel	Foeniculum Vulgan	5	14	<NULL>	<NULL>	
43	Rose Otto	Rosa Damascena	5	5	<NULL>	<NULL>	
44	Sage	Salvia Officinalis	1	1	<NULL>	<NULL>	
45	Vanilla Oleoresin	Vanilla Planifolia	2	14	<NULL>	<NULL>	
46	Benzoin	Styrax Benzoin	2	9	<NULL>	<NULL>	
47	Linden	Tilia Vulgaris	2	5	<NULL>	<NULL>	
48	InsertFromGrid	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>	
49	InsertFromSQL	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>	
50	InsertAllValues	Latin Name	1	1	Sample	Description of the Oil	
*							

使用 **DEFAULT** 和 **NULL** 来插入数据列

- 1. 将 **查询设计师** 的窗口切换到另一个包含 **INSERT** 陈述式的窗口。
- 2. 将 **SQL** 陈述式内容更改为：

```
3.      INSERT INTO Oils
4.                                     (OilName, LatinName, Sample)
VALUES('InsertDefault', NULL, DEFAULT)
```

- 5. 在 **查询设计师** 的工具列上按一下 **执行** 按钮，以便执行此查询。



执行按钮



查询设计师 会显示已将一行数据列插入到数据表的讯息。



6. 按一下 **确定** 按钮，以便关闭此消息框。将 **查询设计师** 的窗口切换到另一个包含 **Oils** 数据表内所有数据列的窗口。
7. 在 **查询设计师** 内的工具列上按一下 **执行** 按钮，以便执行 **SELECT *** 查询，并且卷动 **查询设计师** 内的滚动条至最尾端，您可以在最后一笔数据列中发现所插入的资料列。



执行按钮

7. 资料表 'Oils' 中的资料 (在 'Aromatherapy' 中) 於 'MICKEY'

OilID	OilName	LatinName	PlantTypeID	PlantPartID	Sample	Description
31	Peppermint	Mentha Piperata	1	1	<NULL>	<NULL>
32	Pettigrain	Citrus Aurantium	2	1	<NULL>	<NULL>
33	Pine	Pinus Sylvestris	2	13	<NULL>	<NULL>
34	Rose	Rosa Centifolia	5	5	<NULL>	<NULL>
35	Rosemary	Rosmarinus Officin	1	1	<NULL>	<NULL>
36	Rosewood	Aniba Rosaeodora	2	4	<NULL>	<NULL>
37	Sandalwood	Santalum Album	2	4	<NULL>	<NULL>
38	Tea Tree	Melaleuca Alternifo	2	1	<NULL>	<NULL>
39	Thyme	Thymus Vulgaris	1	1	<NULL>	<NULL>
40	Vetivert	Vetiveria Zizanoide	4	10	<NULL>	<NULL>
41	Ylang-ylang	Cananga Odorata	2	5	<NULL>	<NULL>
42	Fennel	Foeniculum Vulgar	5	14	<NULL>	<NULL>
43	Rose Otto	Rosa Damascena	5	5	<NULL>	<NULL>
44	Sage	Salvia Officinalis	1	1	<NULL>	<NULL>
45	Vanilla Oleoresin	Vanilla Planifolia	2	14	<NULL>	<NULL>
46	Benzoin	Styrax Benzoin	2	9	<NULL>	<NULL>
47	Linden	Tilia Vulgaris	2	5	<NULL>	<NULL>
48	InsertFromGrid	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>
49	InsertFromSQL	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>
50	InsertAllValues	Latin Name	1	1	Sample	Description of the C
51	InsertDefault	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>

8. 将二个 [查询设计师](#) 窗口关闭。

插入多个资料列

INSERT 陈述式的使用方式有第二种形式，那就是使用 SELECT 陈述式将数据列或多个数据列

新增至数据表中。使用 SELECT 陈述式新增数据列会比使用 VALUES 新增数据列的方式还要好。

这种 INSERT 陈述式的格式如下所示：

```
INSERT INTO table_or_view [(column_list)]
```

```
SELECT (column_list)

FROM table_or_view

[WHERE (condition)]
```

其中 **WHERE** 子句是选择性的。如果没有被指定 **WHERE** 子句时，则在 **FROM** 子句中的所指定的数据表或检视表中的所有数据列将会新增到 **INSERT** 陈述式内所指定的数据表或检视表中。

使用方格窗格插入多个数据列

1. 在 [详细数据窗格](#) 中的 **MyOils** 数据表上按右键，以便开启 [查询设计师](#)，并且指向 [开启数据表](#) 菜单中的 [传回所有数据列](#) 项目。

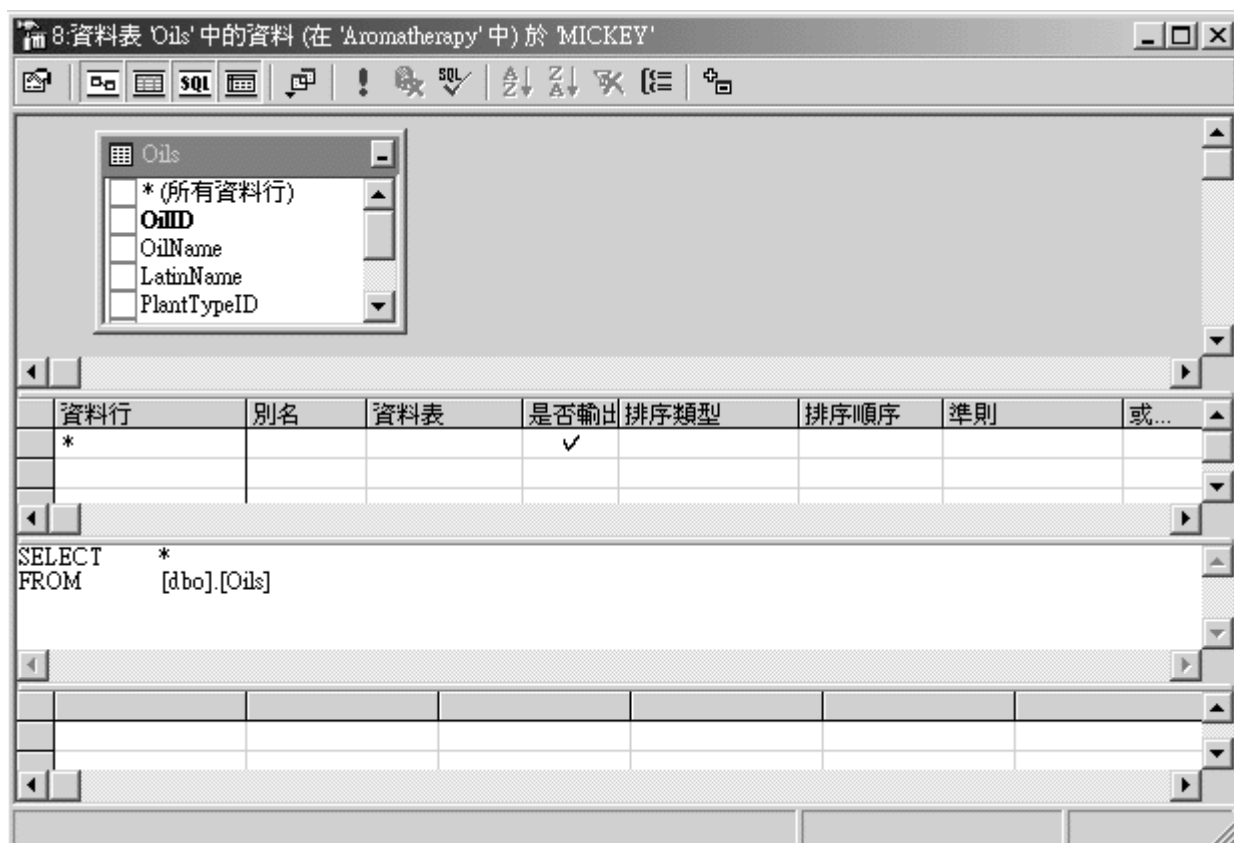
此时会显示一个新的 [查询设计师](#) 窗口。

7:資料表 'MyOils' 中的資料 (在 'Aromatherapy' 中) 於 'MICKEY'

	OilID	OilName	LatinName	PlantTypeID	PlantPartID	Sample	De
▶							

2. 不要将目前的 [查询设计师](#) 窗口关闭，在 [详细数据窗格](#) 内的 [Oils](#) 数据表上按右
 钮，并且指向 [开启数据表](#) 子菜单中的 [查询](#)。

此时会显示另一个新的 [查询设计师](#) 窗口。



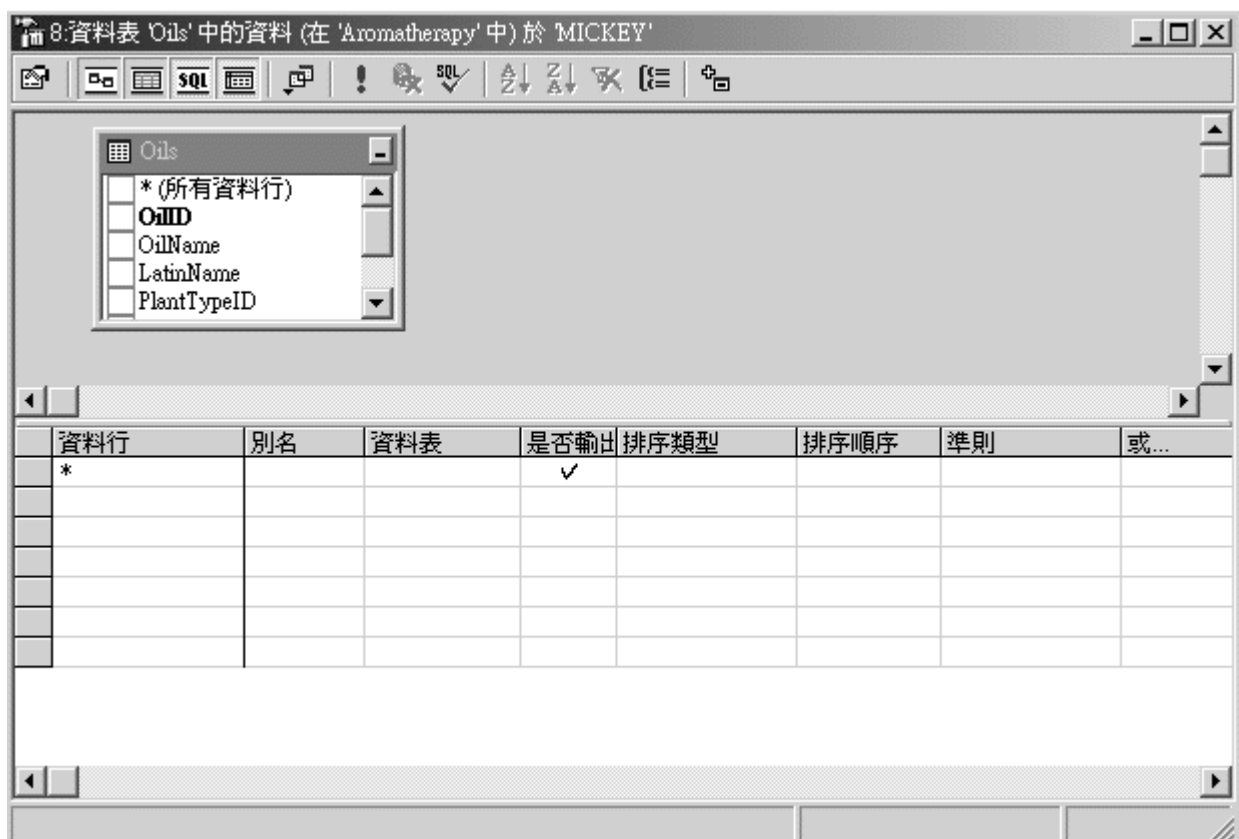
3. 将 **SQL 窗格** 和 **结果窗格** 隐藏。



显示/隐藏 SQL 窗格按钮



显示/隐藏结果窗格按钮

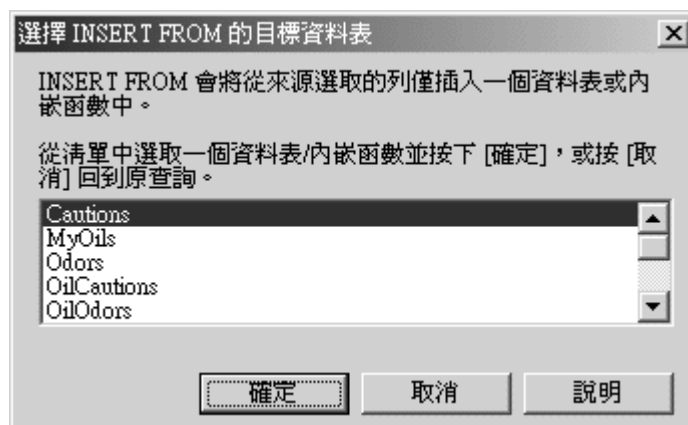


- 在 **查询设计师** 的工具列上按一下 **变更查询类型** 按钮，并在清单中选择 **INSERT FROM**。



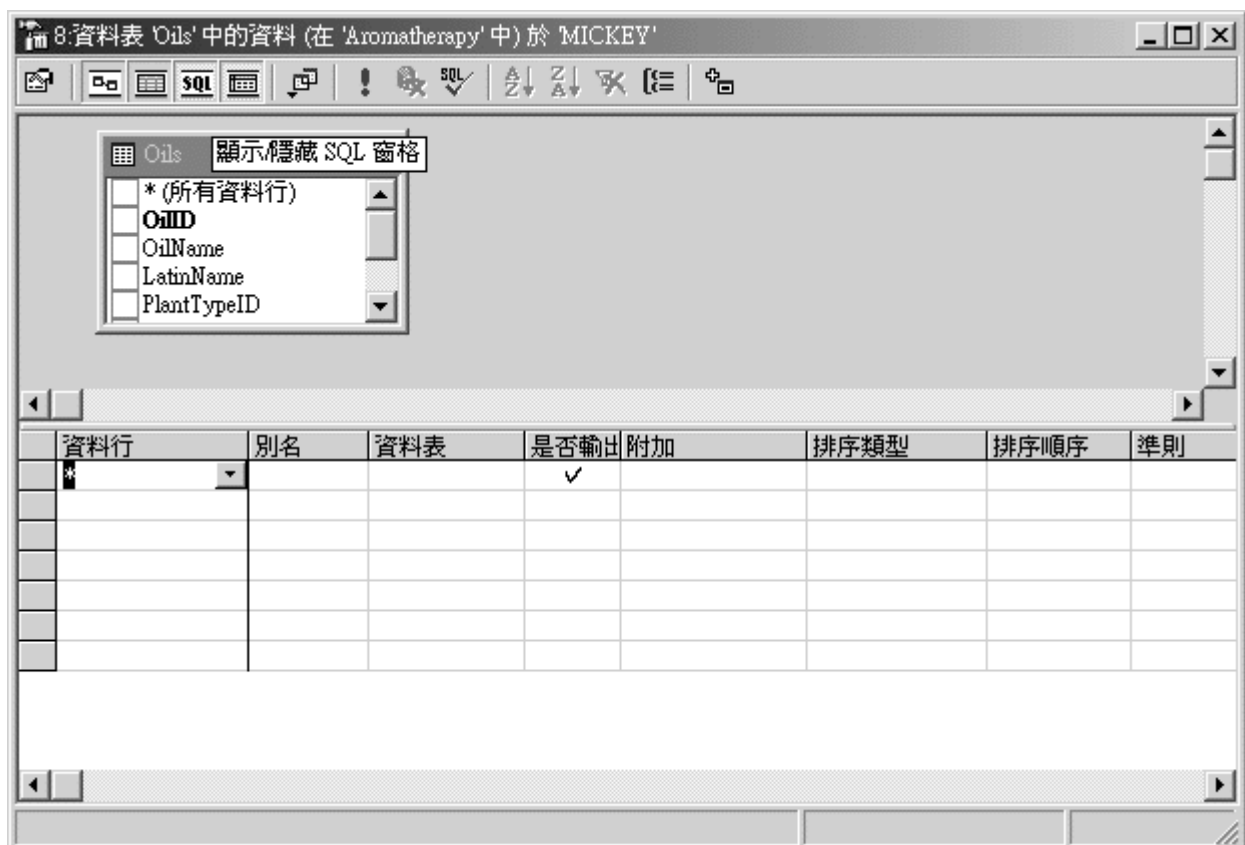
变更查询类型按钮

查询设计师 会显示一个要求您选择目标数据表的对话框。

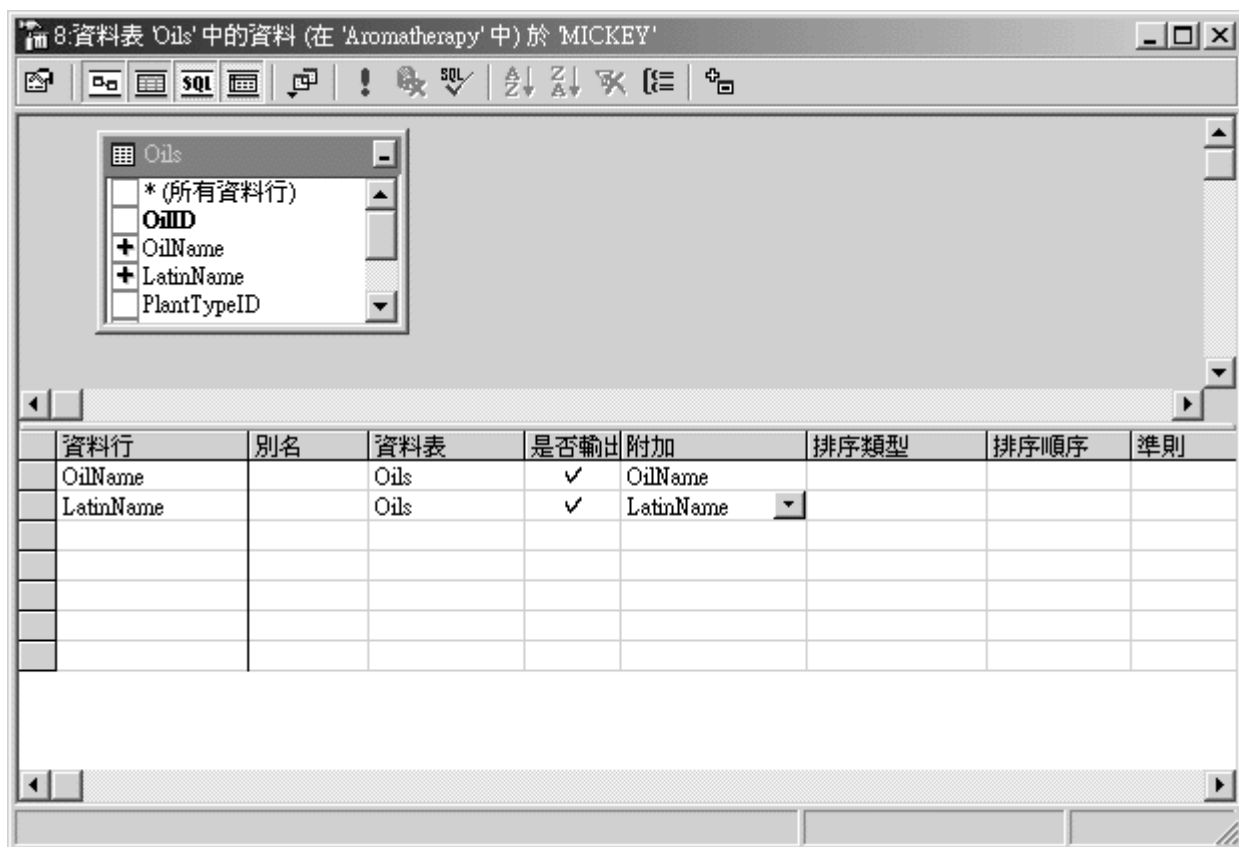


5. 自对话框内的清单中选择 **MyOils**，然后按一下 **确定** 按钮。

查询设计师 会将 **附加** 数据行新增至 **方格窗格**。

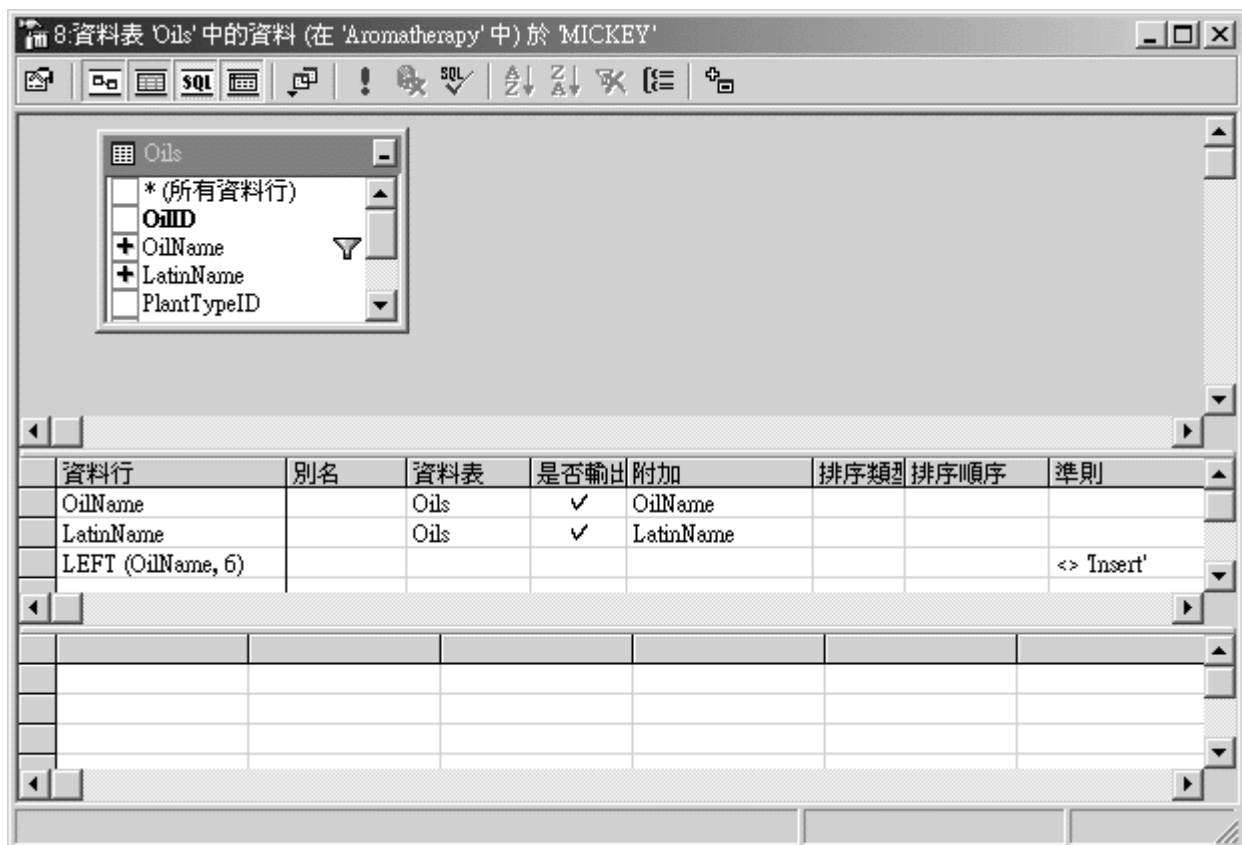


6. 變更為只插入 **OilName** 和 **LatinName** 數據行。



7. 在 **方格窗格** 中新增 **Left(OilName, 6)** 数据行来当作是计算字段，取消选取 **是否**

输出 数据行中的复选框，然后在 **准则** 数据行中新增 **<>'Insert'**。



- 在 [查詢設計師](#) 內的工具列上按一下 [執行](#) 按鈕，以便執行此查詢。



執行按鈕

[查詢設計師](#) 會顯示已將數據列插入到數據表的訊息。



9. 按一下 **确定** 按钮，以便关闭此消息框。将 **查询设计师** 的窗口切换到另一个包含 **MyOils** 数据表内所有数据列的窗口。
10. 在 **查询设计师** 的工具列上按一下 **执行** 按钮，以便执行 **SELECT *** 查询。



执行按钮

7. 資料表 'MyOils' 中的資料 (在 'Aromatherapy' 中) 於 'MICKEY'

	OilID	OilName	LatinName	PlantTypeID	PlantPartID	Sample
▶	1	Basil	Ocimum Basilicum	<NULL>	<NULL>	<NULL>
	2	Bergamot	Citrus Bergamia	<NULL>	<NULL>	<NULL>
	3	Black Pepper	Piper Nigrum	<NULL>	<NULL>	<NULL>
	4	Cedarwood	Cedrus Atlantica, Jr	<NULL>	<NULL>	<NULL>
	5	German Chamomile	Matricaria Chamom	<NULL>	<NULL>	<NULL>
	6	Roman Chamomile	Anthemis Nobilis	<NULL>	<NULL>	<NULL>
	7	Cinnamon	Cinnamomum Zeyl	<NULL>	<NULL>	<NULL>
	8	Citronella	Cymbopogon Nard	<NULL>	<NULL>	<NULL>
	9	Clary Sage	Salvia Sclarea	<NULL>	<NULL>	<NULL>
	10	Coriander	Coriandrum Sativum	<NULL>	<NULL>	<NULL>
	11	Cypress	Cupressus Semperv	<NULL>	<NULL>	<NULL>
	12	Eucalyptus	Eucalyptus Globulu	<NULL>	<NULL>	<NULL>
	13	Frankincense	Buswellia Thurifera	<NULL>	<NULL>	<NULL>
	14	Geranium	Pelargonium Grave	<NULL>	<NULL>	<NULL>
	15	Ginger	Zingiber Officinale	<NULL>	<NULL>	<NULL>
	16	Grapefruit	Citrus Paradisi	<NULL>	<NULL>	<NULL>
	17	Jasmine	Jasmineum Officina	<NULL>	<NULL>	<NULL>
	18	Juniper	Juniperus Commun	<NULL>	<NULL>	<NULL>

使用 SQL 窗格插入多个数据列

1. 将 [查询设计师](#) 的窗口切换到另一个包含 INSERT 陈述式的窗口。



显示/隐藏图表窗格按钮

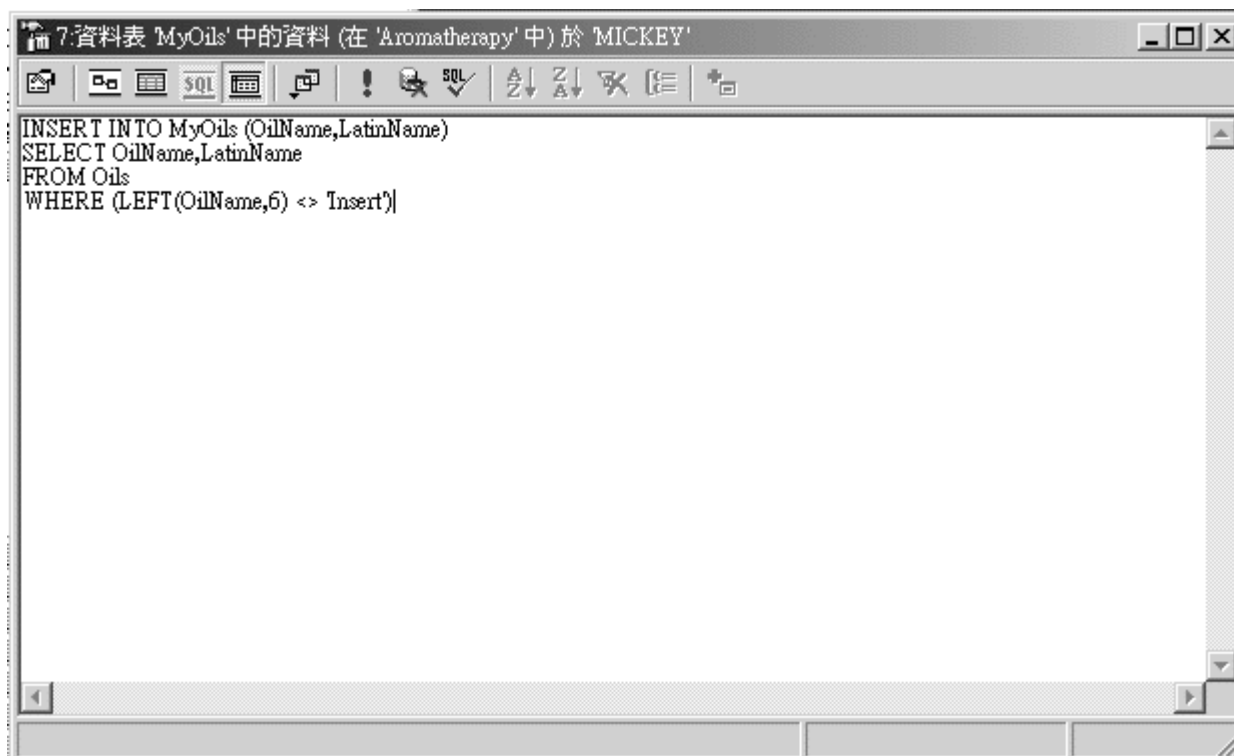


显示/隐藏方格窗格按钮



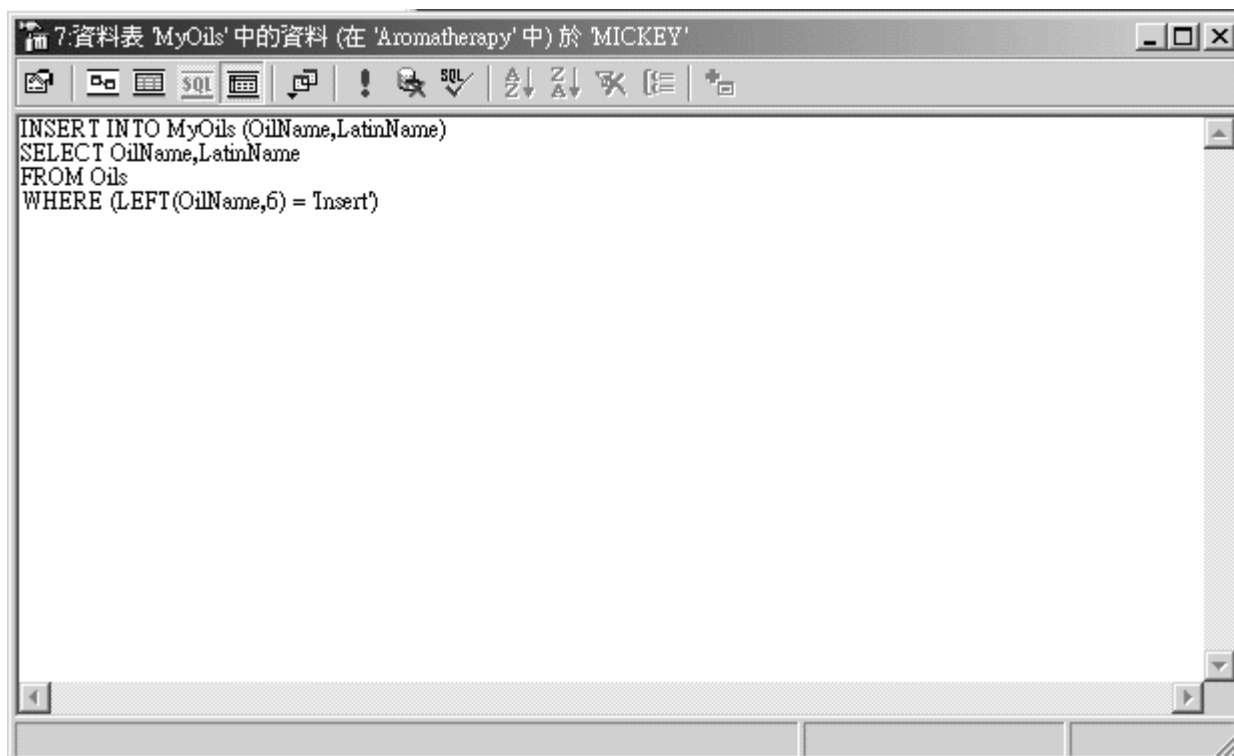
显示/隐藏 SQL 窗格按钮

2. 将 [图表窗格](#) 和 [方格窗格](#) 隐藏，再将 [SQL 窗格](#) 显示在屏幕上。



3. 将 SQL 陈述式内容更改为:

```
4.      INSERT INTO MyOils
5.
6.      (OilName, LatinName)
7.
8.      SELECT OilName, LatinName
9.
10.     FROM Oils
11.
12.     WHERE (LEFT(OilName, 6) = 'Insert')
```



提示

这里唯一的变更是 **WHERE** 陈述式。

- 在 **查询设计师** 的工具列上按一下 **执行** 按钮，以便执行此查询。Enterprise

Manager 会显示此查询已经执行的消息框。



执行按钮



9. 按一下 **确定** 按钮，以便关闭此消息框。将 **查询设计师** 的窗口切换到另一个包含 **MyOils** 数据表内所有数据列的窗口。
 10. 在 **查询设计师** 的工具列上按一下 **执行** 按钮，以便执行 **SELECT *** 查询，并且滚动 **查询设计师** 的滚动条，您可以发现所插入的数据列。
-



执行按钮

7. 資料表 'MyOils' 中的資料 (在 'Aromatherapy' 中) 於 'MICKEY'						
	OilID	OilName	LatinName	PlantTypeID	PlantPartID	Sample
	46	Benzoin	Styrax Benzoin	<NULL>	<NULL>	<NULL>
	47	Linden	Tilia Vulgaris	<NULL>	<NULL>	<NULL>
▶	48	InsertFromGrid	<NULL>	<NULL>	<NULL>	<NULL>
	49	InsertFromSQL	<NULL>	<NULL>	<NULL>	<NULL>
	50	InsertAllValues	Latin Name	<NULL>	<NULL>	<NULL>
	51	InsertDefault	<NULL>	<NULL>	<NULL>	<NULL>
*						

11. 将二个 [查询设计师](#) 窗口关闭。

本章总结

要执行的工作	SQL 语法
藉由指定数据行插入数据列	<pre>INSERT INTO table_or_view(column_list) VALUES (value_list)</pre>

插入一数据列至所有的数据行	<pre>INSERT INTO table_or_view VALUES(values_list)</pre>
使用 SELECT 陈述式来插入多个数据列	<pre>INSERT INTO table_or_view [(column_list)] SELECT (column_list) FROM table_or_view [WHERE (condition)]</pre>

18. 更新数据列

在本章中，您将学习到：

- 使用方格窗格更新数据表中所有的数据列。
- 使用方格窗格更新数据表中选取的数据列。
- 使用 SQL 窗格更新数据表中所有的数据列。
- 使用 SQL 窗格更新数据表中选取的数据列。
- 使用 FROM 陈述式更新数据列。

认识 UPDATE 陈述式

UPDATE 陈述式允许您变更数据表中一或多个数据列的值。UPDATE 陈述式的基本语法为：

```
UPDATE table_or_view
```

```
SET update_list
```

```
[WHERE (condition)]
```

就像 INSERT 陈述式一样, 单一的 UPDATE 陈述式只可以在单一的数据表或检视表中更新数据。

使用 UPDATE 陈述式更新检视表的限制与使用 INSERT 陈述式更新检视表的限制一样:

- 检视表必须不包含汇总函数, 例如 COUNT 或 AVG 函数。
- 检视表必须不包含 TOP、GROUP BY、UNION 或 DISTINCT 关键词。
- 检视表必须不包含计算的数据行。
- 检视表必须在 FROM 子句中引用一个数据表。
- UPDATE 陈述式更新的数据行只能来自单一的资料表。

SET 关键词后面会接着要更新的数据行和新的值, 以 `column_name = new_value` 的形式, 并且

以逗号「,」分隔。新的值可以是一个常数、表达式, 并且也可以引用该数据行本身。举例来说,

表达式 `SalesPrice = SalesPrice * .90` 是宣告在 SalesPrice 资料行的值有百分之 10 的折扣。

WHERE 子句是选择性的，只有 WHERE 子句所指定的数据列会被更新。如果 WHERE 子句没有包含在 UPDATE 陈述式中时，则在数据表中的所有数据列将会被更新。

使用 UPADTE 陈述式

和其它大部份的查询一样，您可以在 [查询设计师](#) 中使用 [方格窗格](#) 或直接在 [SQL 窗格](#) 中输入 SQL 陈述式以便建立一个 UPDATE 查询。

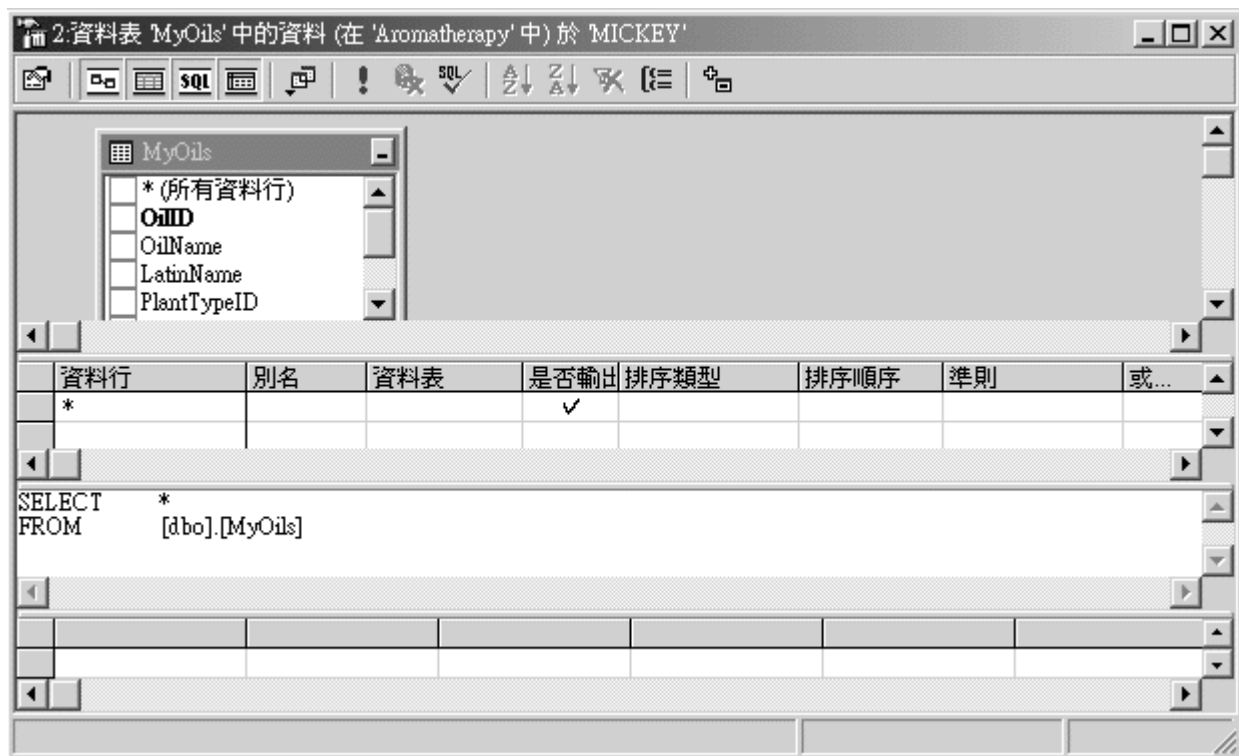
使用方格窗格更新数据列

[查询设计师](#) 的 [方格窗格](#) 针对您要建立的 UPDATE 查询提供了一个简单的方式。

使用方格窗格更新所有的数据列

1. 在 [Aromathrtapy](#) 数据库中 [MyOlis](#) 数据表上按右钮，并且指向 [开启数据表](#) 菜单中的 [查询](#)。

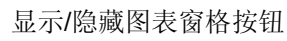
此时，[查询设计师](#) 窗口将会开启。



- 在工具列上按一下 [顯示/隱藏 SQL 窗格](#) 按鈕和 [顯示/隱藏图表窗格](#) 按鈕，以便將 [SQL 窗格](#) 和 [图表窗格](#) 隱藏。



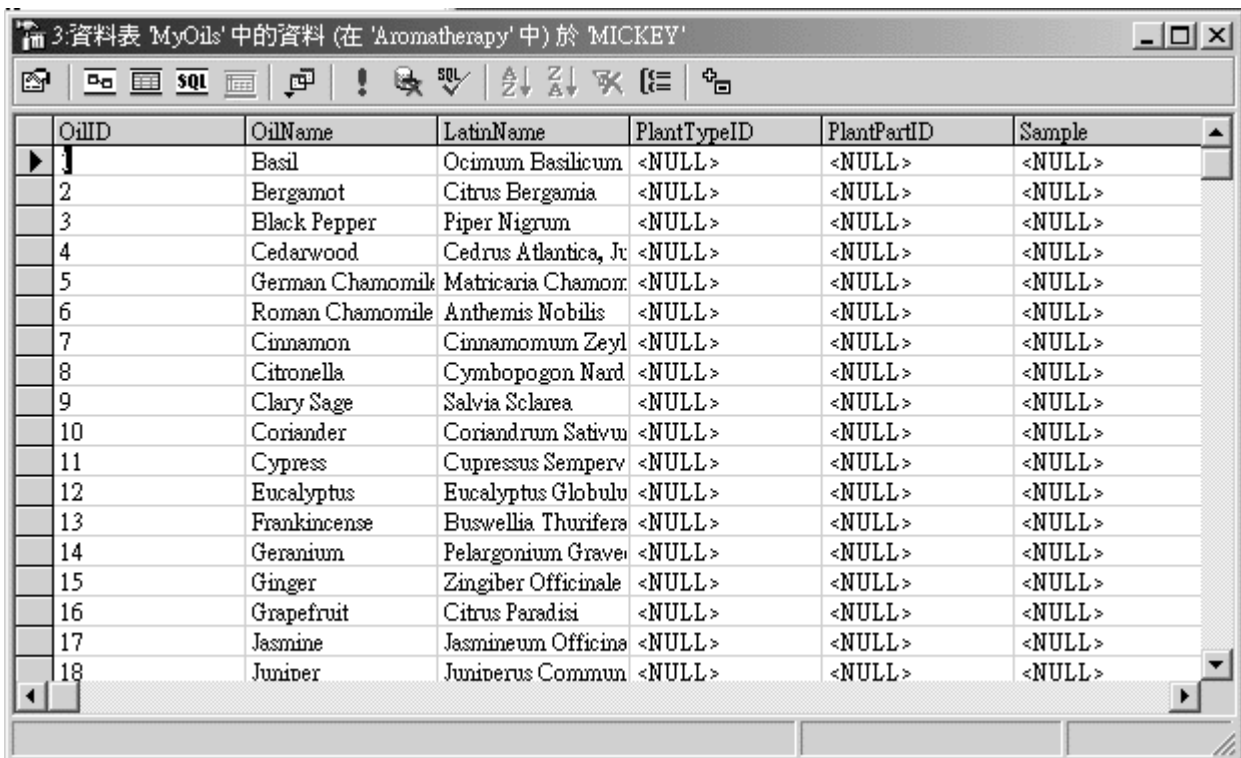
顯示/隱藏 SQL 窗格按鈕

[illegible]

3. 不用将目前的 [查询设计师](#) 关闭。在 Enterprise Manager 的详细数据窗格中

的 [MyOils](#) 资料表上右钮，并且在 [开启数据表](#) 项目的子菜单中选择 [传回所有数据列](#)。

此时，会出现另一个新的 [查询设计师](#)，在此 [查询设计师](#) 中将会显示 [MyOils](#) 数据表中的所有数据列的数据。



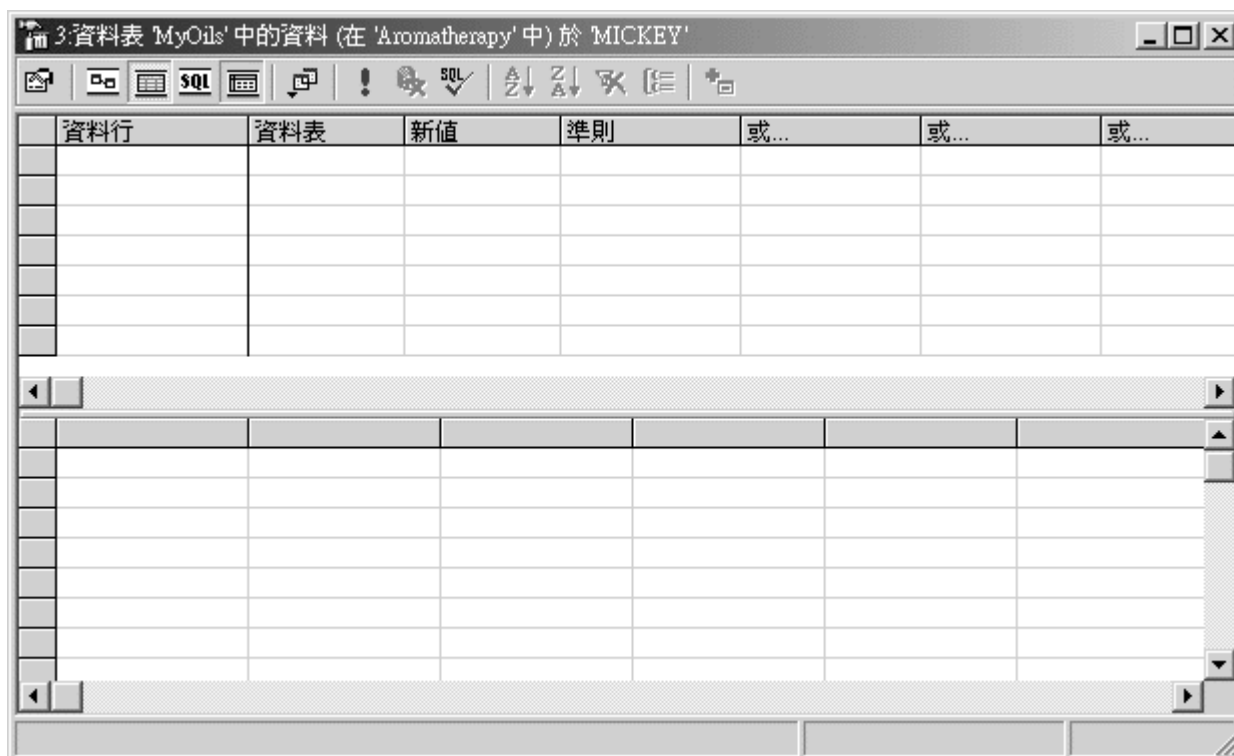
OilID	OilName	LatinName	PlantTypeID	PlantPartID	Sample
1	Basil	Ocimum Basilicum	<NULL>	<NULL>	<NULL>
2	Bergamot	Citrus Bergamia	<NULL>	<NULL>	<NULL>
3	Black Pepper	Piper Nigrum	<NULL>	<NULL>	<NULL>
4	Cedarwood	Cedrus Atlantica, Jr	<NULL>	<NULL>	<NULL>
5	German Chamomile	Matricaria Chamom	<NULL>	<NULL>	<NULL>
6	Roman Chamomile	Anthemis Nobilis	<NULL>	<NULL>	<NULL>
7	Cinnamon	Cinnamomum Zeyl	<NULL>	<NULL>	<NULL>
8	Citronella	Cymbopogon Nard	<NULL>	<NULL>	<NULL>
9	Clary Sage	Salvia Sclarea	<NULL>	<NULL>	<NULL>
10	Coriander	Coriandrum Sativu	<NULL>	<NULL>	<NULL>
11	Cypress	Cupressus Semperv	<NULL>	<NULL>	<NULL>
12	Eucalyptus	Eucalyptus Globulu	<NULL>	<NULL>	<NULL>
13	Frankincense	Buswellia Thurifers	<NULL>	<NULL>	<NULL>
14	Geranium	Pelargonium Grave	<NULL>	<NULL>	<NULL>
15	Ginger	Zingiber Officinale	<NULL>	<NULL>	<NULL>
16	Grapefruit	Citrus Paradisi	<NULL>	<NULL>	<NULL>
17	Jasmine	Jasmineum Officina	<NULL>	<NULL>	<NULL>
18	Juniper	Juniperus Commun	<NULL>	<NULL>	<NULL>

4. 将 **查询设计师** 的窗口切换到另一个包含有 **方格窗格** 及 **结果窗格** 的窗口。在 **查询设计师** 的工具列上按一下 **变更查询类型** 按钮，接着再选择 **UPDATE** 。

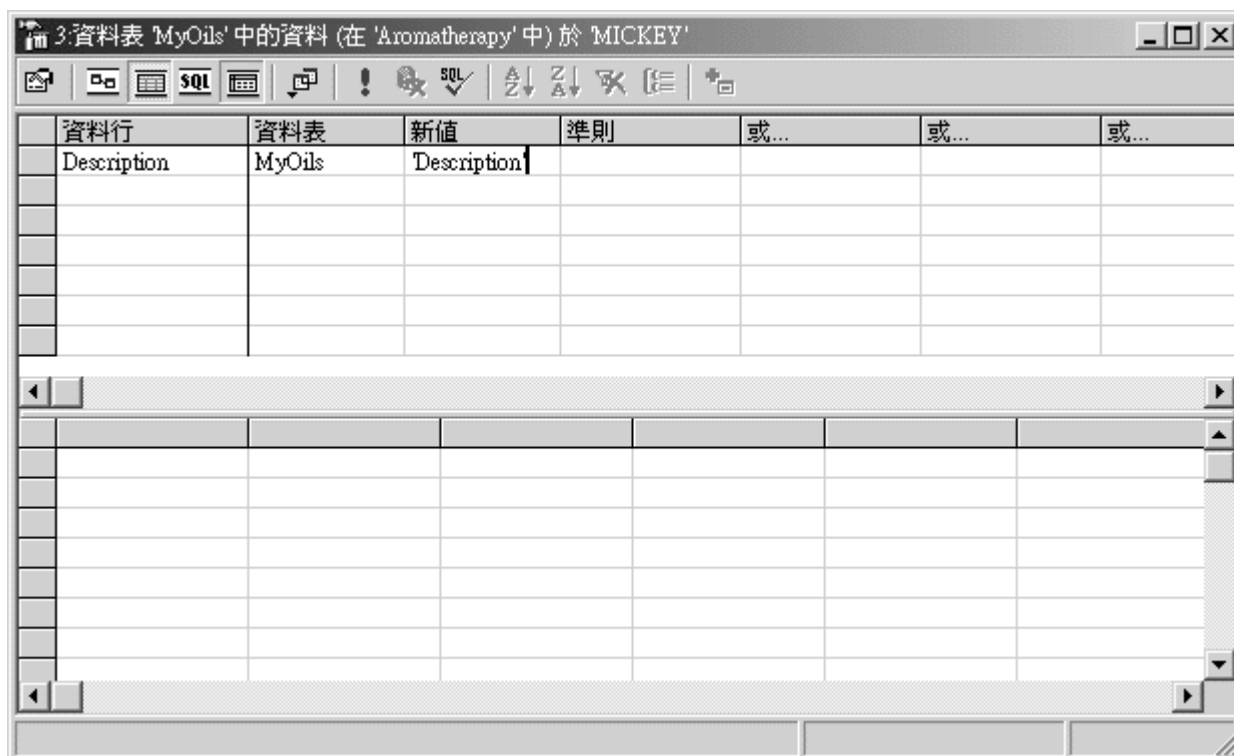


变更查询类型按钮

查询设计师 会将 **新值** 数据行新增到 **方格窗格** 。



5. 将 **Description** 数据行新增到 **方格窗格** 中，然后再将 **新值** 数据行中设定为 **'Description'**。



6. 在 [查询设计师](#) 的工具列上按一下 [执行](#) 按钮，以便执行此查询。



执行按钮

[查询设计师](#) 会显示已经将数据列更新的讯息。



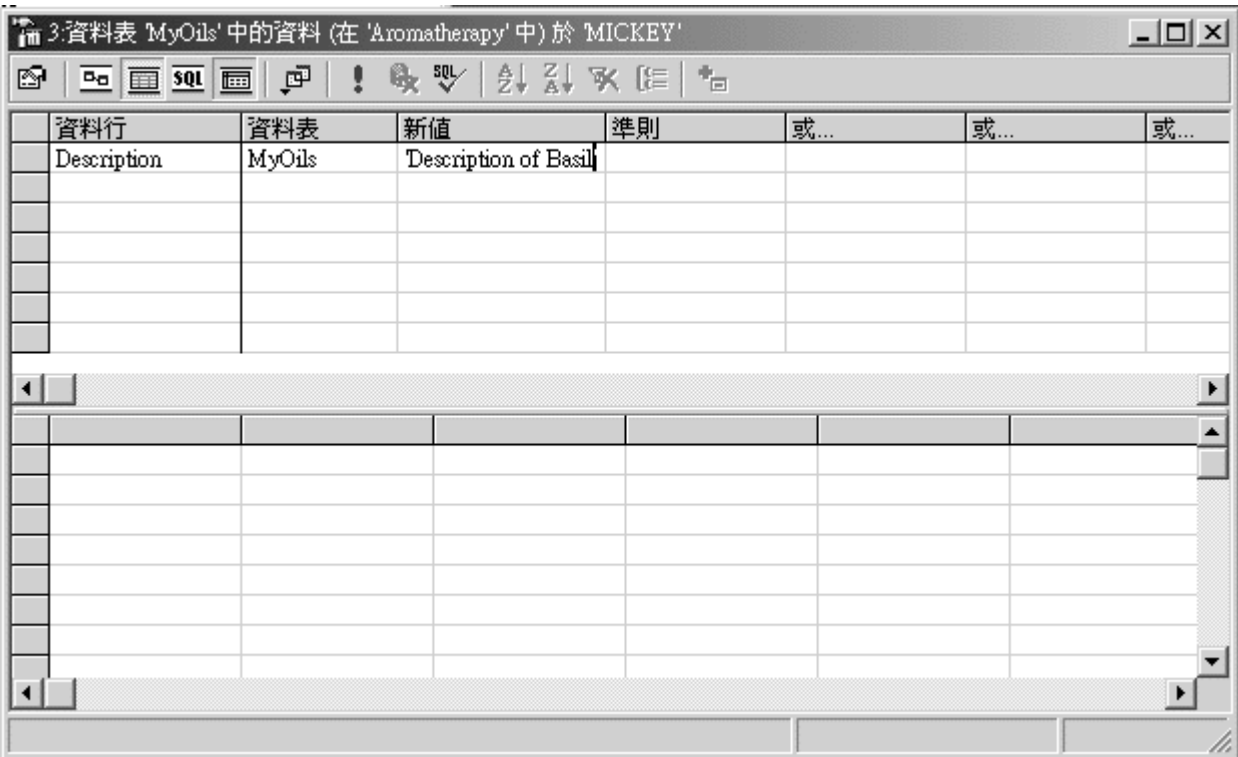
7. 將 **查詢設計師** 的窗口切換到另一個包含有 **MyOils** 數據表中所有數據列的窗口。
8. 在 **查詢設計師** 的工具列上按一下 **執行** 按鈕，以便執行 **SELECT *** 查詢。

2:資料表 'MyOils' 中的資料 (在 'Aromatherapy' 中) 於 'MICKEY'

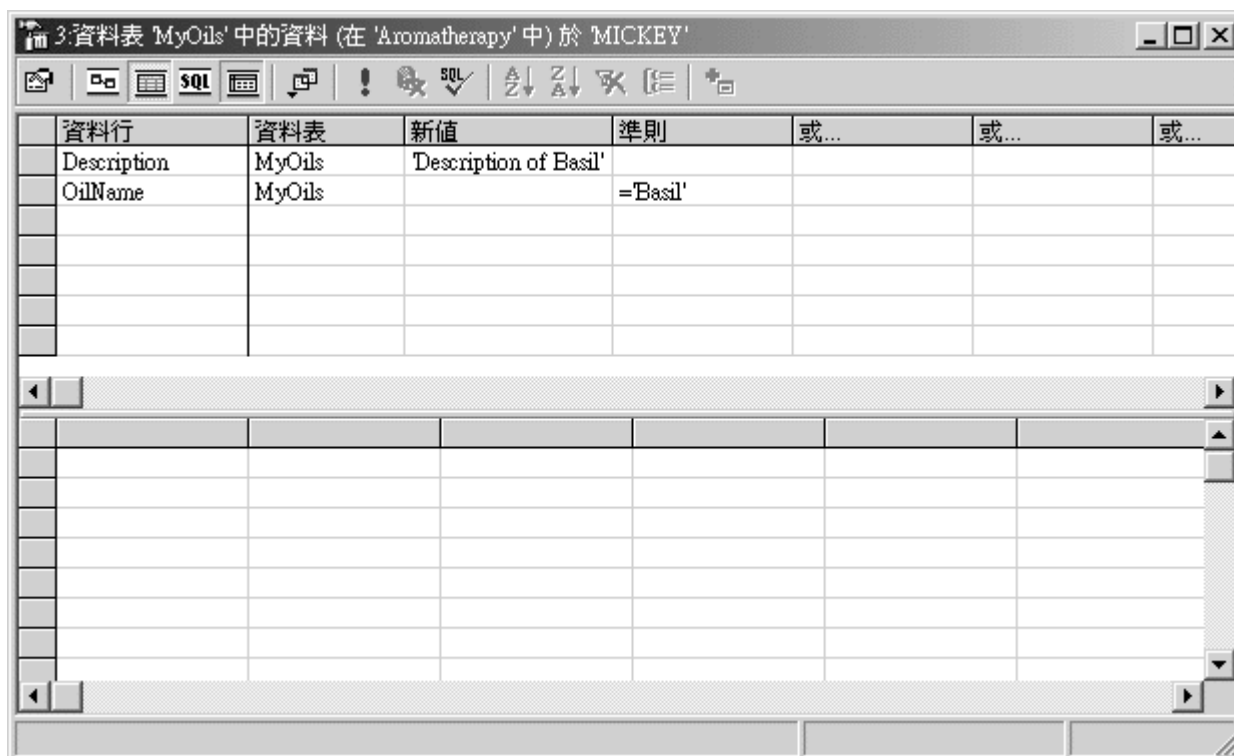
OilName	LatinName	PlantTypeID	PlantPartID	Sample	Description
Basil	Ocimum Basilicum	<NULL>	<NULL>	<NULL>	Description
Bergamot	Citrus Bergamia	<NULL>	<NULL>	<NULL>	Description
Black Pepper	Piper Nigrum	<NULL>	<NULL>	<NULL>	Description
Cedarwood	Cedrus Atlantica, Ju	<NULL>	<NULL>	<NULL>	Description
German Chamomile	Matricaria Chamom	<NULL>	<NULL>	<NULL>	Description
Roman Chamomile	Anthemis Nobilis	<NULL>	<NULL>	<NULL>	Description
Cinnamon	Cinnamomum Zeyl	<NULL>	<NULL>	<NULL>	Description
Citronella	Cymbopogon Nard	<NULL>	<NULL>	<NULL>	Description
Clary Sage	Salvia Sclarea	<NULL>	<NULL>	<NULL>	Description
Coriander	Coriandrum Sativu	<NULL>	<NULL>	<NULL>	Description
Cypress	Cupressus Semperv	<NULL>	<NULL>	<NULL>	Description
Eucalyptus	Eucalyptus Globulu	<NULL>	<NULL>	<NULL>	Description
Frankincense	Buswellia Thurifers	<NULL>	<NULL>	<NULL>	Description
Geranium	Pelargonium Grave	<NULL>	<NULL>	<NULL>	Description
Ginger	Zingiber Officinale	<NULL>	<NULL>	<NULL>	Description
Grapefruit	Citrus Paradisi	<NULL>	<NULL>	<NULL>	Description
Jasmine	Jasmineum Officina	<NULL>	<NULL>	<NULL>	Description
Juniper	Juniperus Commun	<NULL>	<NULL>	<NULL>	Description

使用方格窗格更新单一数据列

- 1. 将 查询设计师 的窗口切换到另一个包含 UPADTE 陈述式的窗口。
- 2. 将 新值 数据行中的值更新为 'Description of Basil' 。



- 3. 在 方格窗格 中加入 OilName 数据行，并且将它的 准则 数据行设定为 ='Basil' 。



- 在 **查询设计师** 的工具列上按一下 **执行** 按钮，以便执行此查询。



执行按钮



5. 将 **查询设计师** 的窗口切换到另一个包含有 **MOils** 数据表中所有数据列的窗口。
6. 在 **查询设计师** 的工具列上按一下 **执行** 按钮，以便执行 **SELECT *** 查询。



执行按钮

2.資料表 'MyOils' 中的資料 (在 'Aromatherapy' 中) 於 'MICKEY'

OilName	LatinName	PlantTypeID	PlantPartID	Sample	Description
Basil	Ocimum Basilicum	<NULL>	<NULL>	<NULL>	Description of Basil
Bergamot	Citrus Bergamia	<NULL>	<NULL>	<NULL>	Description
Black Pepper	Piper Nigrum	<NULL>	<NULL>	<NULL>	Description
Cedarwood	Cedrus Atlantica, Jr	<NULL>	<NULL>	<NULL>	Description
German Chamomile	Matricaria Chamomr	<NULL>	<NULL>	<NULL>	Description
Roman Chamomile	Anthemis Nobilis	<NULL>	<NULL>	<NULL>	Description
Cinnamon	Cinnamomum Zeyl	<NULL>	<NULL>	<NULL>	Description
Citronella	Cymbopogon Nard	<NULL>	<NULL>	<NULL>	Description
Clary Sage	Salvia Sclarea	<NULL>	<NULL>	<NULL>	Description
Coriander	Coriandrum Sativu	<NULL>	<NULL>	<NULL>	Description
Cypress	Cupressus Semperv	<NULL>	<NULL>	<NULL>	Description
Eucalyptus	Eucalyptus Globulu	<NULL>	<NULL>	<NULL>	Description
Frankincense	Buswellia Thurifera	<NULL>	<NULL>	<NULL>	Description
Geranium	Pelargonium Grave	<NULL>	<NULL>	<NULL>	Description
Ginger	Zingiber Officinale	<NULL>	<NULL>	<NULL>	Description
Grapefruit	Citrus Paradisi	<NULL>	<NULL>	<NULL>	Description
Jasmine	Jasminum Officina	<NULL>	<NULL>	<NULL>	Description
Juniper	Juniperus Commun	<NULL>	<NULL>	<NULL>	Description

使用 SQL 窗格更新数据列

就如同其它的查询形式一样，直接在 **SQL 窗格** 中输入 UPDATE 陈述式会更有弹性。

使用 SQL 窗格更新所有的数据列

1. 将 **查询设计师** 的窗口切换到另一个包含 UPADTE 陈述式的窗口。
2. 将 **SQL 窗格** 显示在屏幕上，并将 **方格窗格** 隐藏。

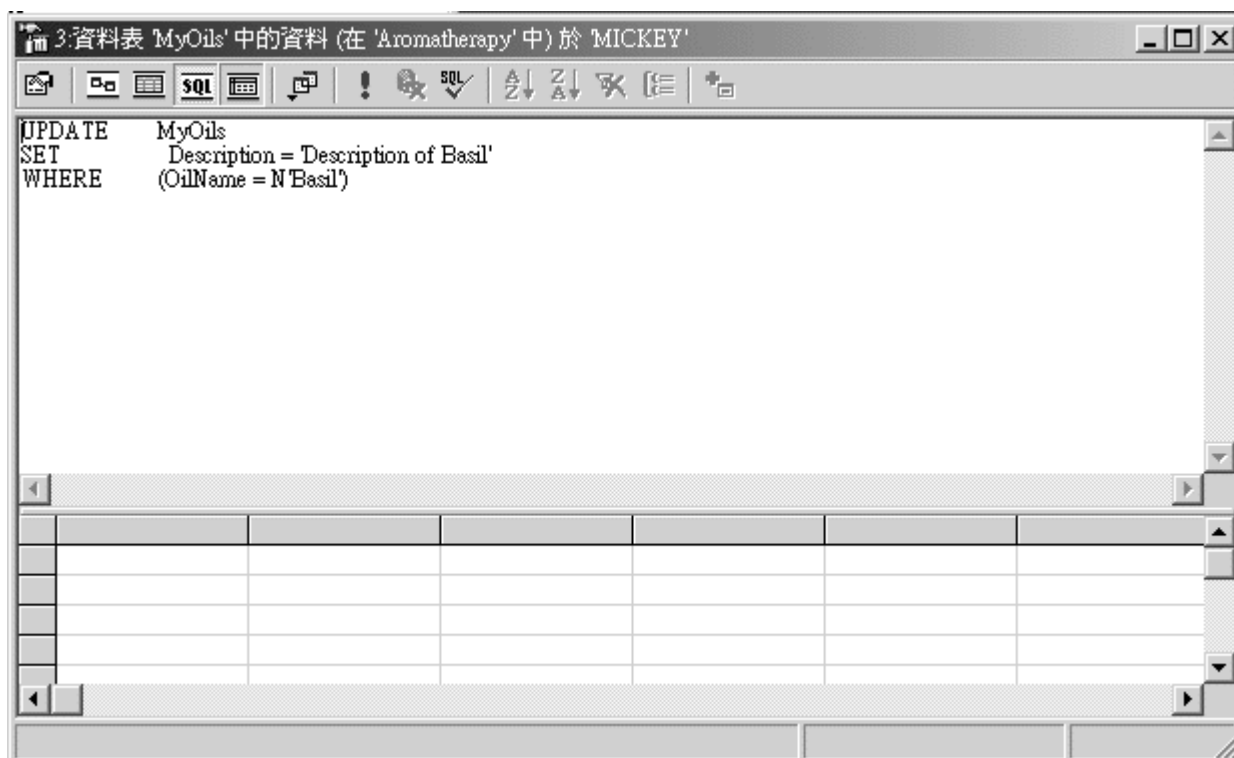


显示/隐藏 SQL 窗格按钮



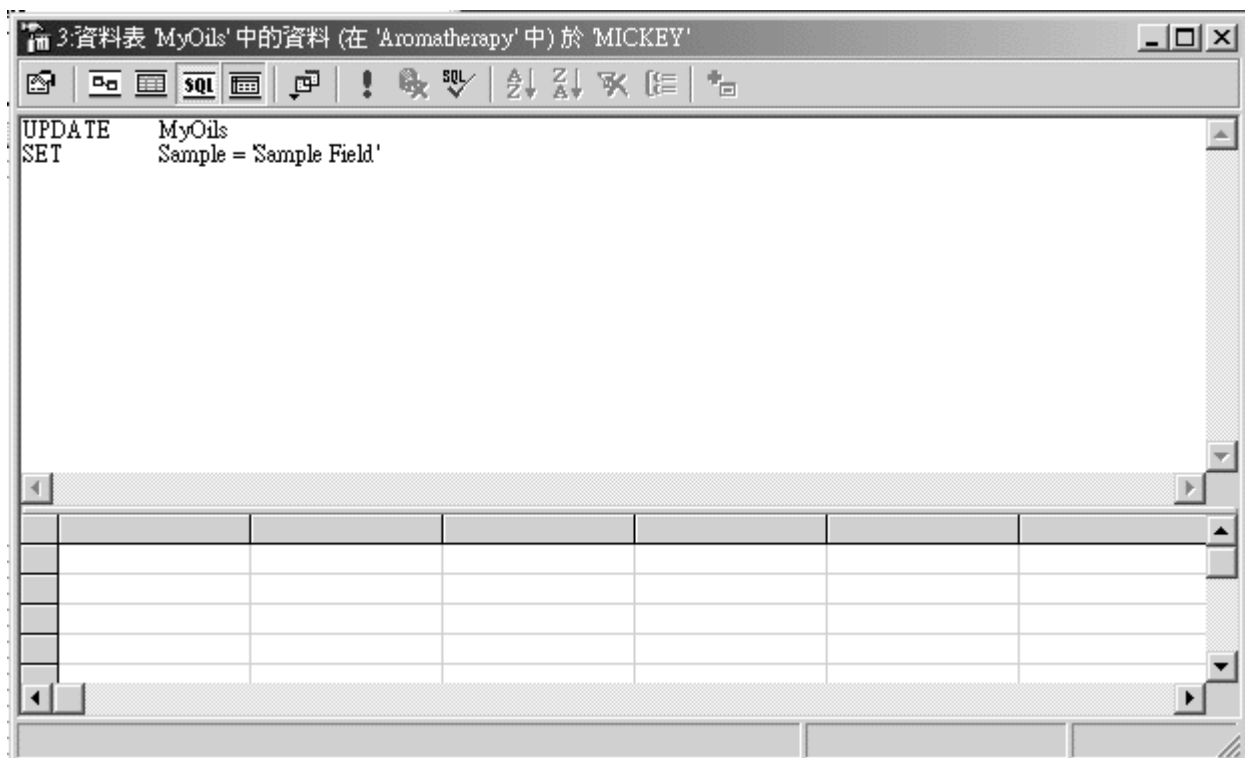
显示/隐藏方格窗格按钮





3. 将 SQL 陈述式中容更改为:

```
4. UPDATE MyOils
    SET Sample = 'Sample Field'
```



5. 在 [查询设计师](#) 的工具列上按一下 [执行](#) 按钮，以便执行此查询。



执行按钮

[查询设计师](#) 会显示已经将多笔数据列更新的讯息。



6. 将 [查询设计师](#) 的窗口切换到另一个包含有 [MyOils](#) 数据表中所有数据列的窗口。
7. 在 [查询设计师](#) 中的工具列上按一下 [执行](#) 按钮，以便执行 **SELECT *** 查询。



执行按钮

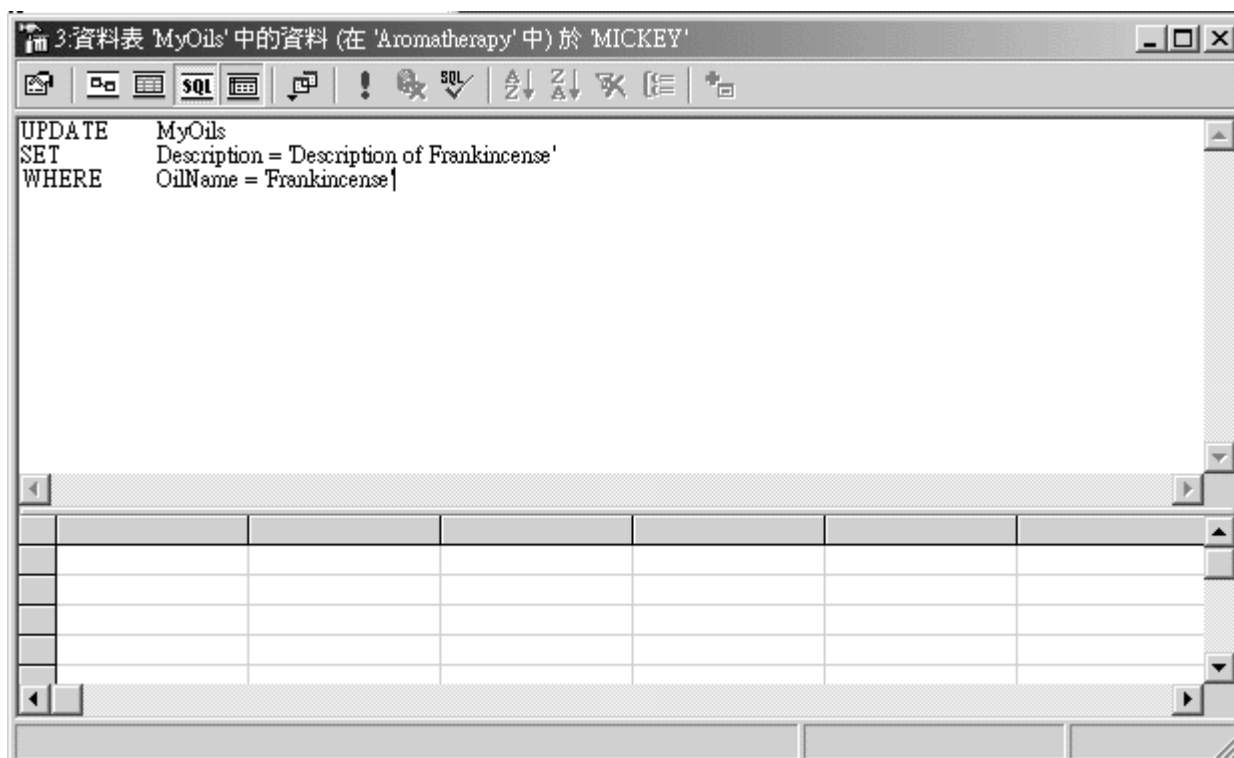
2.資料表 'MyOils' 中的資料 (在 'Aromatherapy' 中) 於 'MICKEY'

	OilID	OilName	LatinName	PlantTypeID	PlantPartID	Sample	De
▶	1	Basil	Ocimum Basilicum	<NULL>	<NULL>	Sample Field	De
	2	Bergamot	Citrus Bergamia	<NULL>	<NULL>	Sample Field	De
	3	Black Pepper	Piper Nigrum	<NULL>	<NULL>	Sample Field	De
	4	Cedarwood	Cedrus Atlantica, Ju	<NULL>	<NULL>	Sample Field	De
	5	German Chamomile	Matricaria Chamomr	<NULL>	<NULL>	Sample Field	De
	6	Roman Chamomile	Anthemis Nobilis	<NULL>	<NULL>	Sample Field	De
	7	Cinnamon	Cinnamomum Zeyl	<NULL>	<NULL>	Sample Field	De
	8	Citronella	Cymbopogon Nard	<NULL>	<NULL>	Sample Field	De
	9	Clary Sage	Salvia Sclarea	<NULL>	<NULL>	Sample Field	De
	10	Coriander	Coriandrum Sativu	<NULL>	<NULL>	Sample Field	De
	11	Cypress	Cupressus Semperv	<NULL>	<NULL>	Sample Field	De
	12	Eucalyptus	Eucalyptus Globulu	<NULL>	<NULL>	Sample Field	De
	13	Frankincense	Buswellia Thurifers	<NULL>	<NULL>	Sample Field	De
	14	Geranium	Pelargonium Grave	<NULL>	<NULL>	Sample Field	De
	15	Ginger	Zingiber Officinale	<NULL>	<NULL>	Sample Field	De
	16	Grapefruit	Citrus Paradisi	<NULL>	<NULL>	Sample Field	De
	17	Jasmine	Jasmineum Officina	<NULL>	<NULL>	Sample Field	De
	18	Juniper	Juniperus Commun	<NULL>	<NULL>	Sample Field	De

使用 WHERE 条件来更新数据列

1. 将 [查询设计师](#) 的窗口切换到另一个包含 UPADTE 陈述式的窗口。
2. 将 SQL 陈述式中容更改为:

```
3.      UPDATE MyOils
4.      SET Description = 'Derscription of Frankincense'
      WHERE OilName = 'Frankincense'
```



5. 在 [查询设计师](#) 的工具列上按一下 [执行](#) 按钮，以便执行此查询。



执行按钮

[查询设计师](#) 会显示已经将数据列更新的讯息。



6. 将 [查询设计师](#) 的窗口切换到另一个包含有 [MyOils](#) 数据表中所有数据列的窗口。
7. 在 [查询设计师](#) 中的工具列上按一下 [执行](#) 按钮，以便执行 **SELECT *** 查询。



执行按钮

2.資料表 'MyOils' 中的資料 (在 'Aromatherapy' 中) 於 'MICKEY'

OilName	LatinName	PlantTypeID	PlantPartID	Sample	Description
Basil	Ocimum Basilicum	<NULL>	<NULL>	Sample Field	Description of Basil
Bergamot	Citrus Bergamia	<NULL>	<NULL>	Sample Field	Description
Black Pepper	Piper Nigrum	<NULL>	<NULL>	Sample Field	Description
Cedarwood	Cedrus Atlantica, Jr	<NULL>	<NULL>	Sample Field	Description
German Chamomile	Matricaria Chamomr	<NULL>	<NULL>	Sample Field	Description
Roman Chamomile	Anthemis Nobilis	<NULL>	<NULL>	Sample Field	Description
Cinnamon	Cinnamomum Zeyl	<NULL>	<NULL>	Sample Field	Description
Citronella	Cymbopogon Nard	<NULL>	<NULL>	Sample Field	Description
Clary Sage	Salvia Sclarea	<NULL>	<NULL>	Sample Field	Description
Coriander	Coriandrum Sativu	<NULL>	<NULL>	Sample Field	Description
Cypress	Cupressus Semperv	<NULL>	<NULL>	Sample Field	Description
Eucalyptus	Eucalyptus Globulu	<NULL>	<NULL>	Sample Field	Description
Frankincense	Buswellia Thurifera	<NULL>	<NULL>	Sample Field	Description of Frankincense
Geranium	Pelargonium Grave	<NULL>	<NULL>	Sample Field	Description
Ginger	Zingiber Officinale	<NULL>	<NULL>	Sample Field	Description
Grapefruit	Citrus Paradisi	<NULL>	<NULL>	Sample Field	Description
Jasmine	Jasmineum Officina	<NULL>	<NULL>	Sample Field	Description
Juniper	Juniperus Commun	<NULL>	<NULL>	Sample Field	Description

使用 FROM 子句更新数据列

如同我们在第 17 章中所看到的，您可以在 INSERT 陈述式中使用 SELECT 陈述式当成数据来源的。UPDATE 陈述式使用一个 FROM 子句以便自其它的数据表取得值。

```
UPDATE table_or_view

SET update_list

FROM table_or_view join_operator join_condition

[WHERE (where_condition)]
```

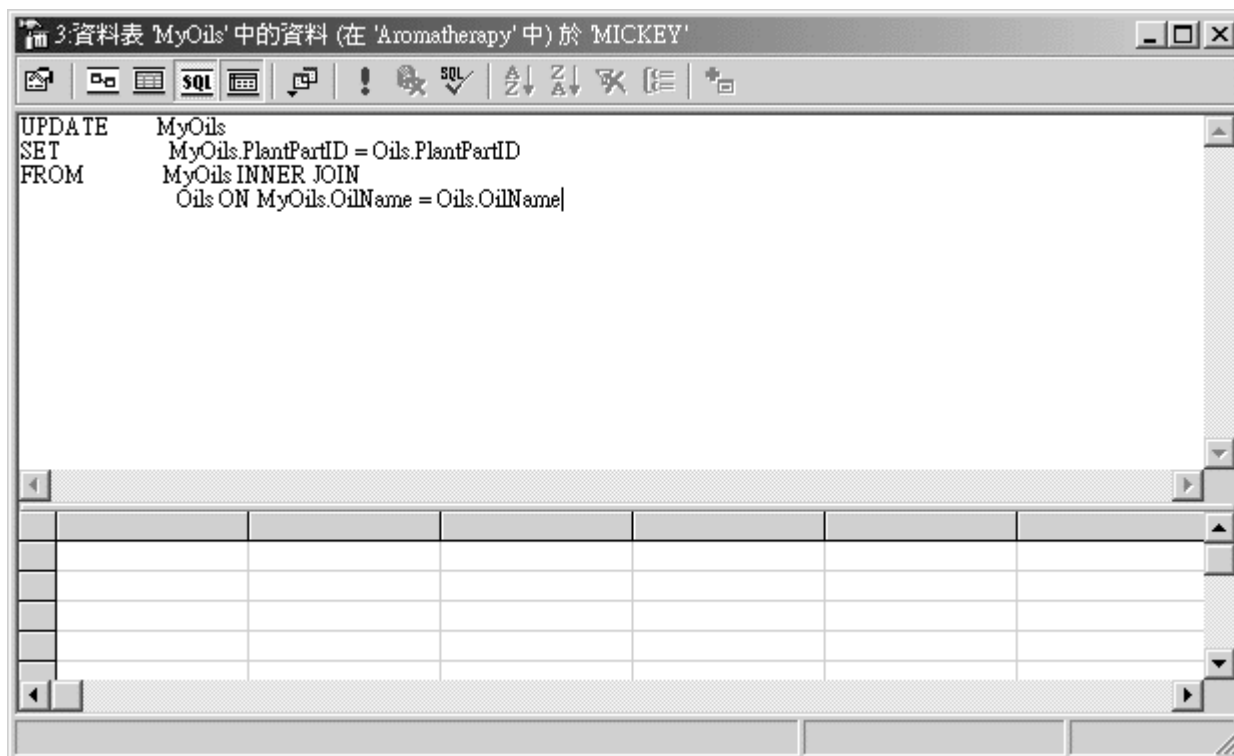

FROM 子句的格式正好与 **SELECT** 陈述式的 **FROM** 子句的格式完全相同，就像使用 **SELECT** 陈述式一样，您可以藉由联结的方式以指定更多的数据表或检视表。选择性的 **WHERE** 条件式是用来限制被更新的数据列。

使用 **FROM** 陈述式以更新数据列

1. 将 [查询设计师](#) 的窗口切换到另一个包含 **UPADTE** 陈述式的窗口。

2. 将 **SQL** 陈述式内容更改为：

```
3.      UPDATE MyOils
4.          SET MyOils.PlantPartID = Oils.PlantPartID
5.          FROM MyOils INNER JOIN Oils ON MyOils.OilName =
Oils.
OilName
```



重要

您无法使用 OilID 数据行来连结二个数据表，因为您在第 17 章中用以新增数据列的 INSERT 命令会在 MyOils 数据表自动为每一个数据列新增一个值。

-
- 在 **查询设计师** 的工具列上按一下 **执行** 按钮，以便执行此查询。

查询设计师 会显示已经将多笔数据列更新的讯息。



7. 将 [查询设计师](#) 的窗口切换到另一个包含有 [MyOils](#) 数据表中所有数据列的窗口。
8. 在 [查询设计师](#) 的工具列上按一下 [执行](#) 按钮，以便执行 **SELECT *** 查询。



执行按钮

2. 資料表 'MyOils' 中的資料 (在 'Aromatherapy' 中) 於 'MICKEY'

	OilID	OilName	LatinName	PlantTypeID	PlantPartID	Sample	Description
▶	1	Basil	Ocimum Basilicum	<NULL>	1	Sample Field	Description of Basil
	2	Bergamot	Citrus Bergamia	<NULL>	2	Sample Field	Description
	3	Black Pepper	Piper Nigrum	<NULL>	3	Sample Field	Description
	4	Cedarwood	Cedrus Atlantica, Ju	<NULL>	4	Sample Field	Description
	5	German Chamomile	Matricaria Chamomr	<NULL>	5	Sample Field	Description
	6	Roman Chamomile	Anthemis Nobilis	<NULL>	5	Sample Field	Description
	7	Cinnamon	Cinnamomum Zeyl	<NULL>	11	Sample Field	Description
	8	Citronella	Cymbopogon Nard	<NULL>	11	Sample Field	Description
	9	Clary Sage	Salvia Sclarea	<NULL>	5	Sample Field	Description
	10	Coriander	Coriandrum Sativu	<NULL>	7	Sample Field	Description
	11	Cypress	Cupressus Semperv	<NULL>	1	Sample Field	Description
	12	Eucalyptus	Eucalyptus Globulu	<NULL>	1	Sample Field	Description
	13	Frankincense	Buswellia Thurifera	<NULL>	9	Sample Field	Description of Frankincense
	14	Geranium	Pelargonium Grave	<NULL>	1	Sample Field	Description
	15	Ginger	Zingiber Officinale	<NULL>	10	Sample Field	Description
	16	Grapefruit	Citrus Paradisi	<NULL>	2	Sample Field	Description
	17	Jasmine	Jasminum Officina	<NULL>	5	Sample Field	Description
	18	Juniper	Juniperus Commun	<NULL>	3	Sample Field	Description
	19	Lavender	Lavandulus Officin	<NULL>	5	Sample Field	Description

本章总结

要执行的工作	SQL 语法
在数据表中更新所有的数据列	<pre>UPDATE table_or_view</pre> <pre>SET update_list</pre> <p>update_list 必须以逗号分隔，其格式为 column = value, column = value..</p>
在数据表中更新选取的数据列	<pre>UPDATE table_or_view</pre> <pre>SET update_list</pre>

	WHERE condition
使用 FROM 子句更新数据列	UPDATE table_or_view SET upadte_list FROM table_or_view join_operator join_condition [WHERE (where_condition)]

19. 删除数据列

在本章中，您将学习到：

- 使用方格窗格从数据表中删除选取的数据列。
- 使用 `DELETE...WHERE` 陈述式来删除数据列。
- 使用 `DELETE..FROM..WHERE` 陈述式来删除数据列。
- 藉由串联删除来删除数据列。
- 使用 `TRUNCATE TABLE` 陈述式来删除所有的数据列。

SQL Server 提供了两种陈述式以从数据表或检视表中删除数据列：`DELETE` 和 `TRUNCATE TABLE` 陈述式。其中 `TRUNCATE TABLE` 陈述式会将数据表内包含的所有数据列无条件的删除；`DELETE` 陈述式就比较有弹性，它可以允许您使用 `WHERE` 子句在资料表以及检视表中只删除特定的数据列。

认识 DELETE 陈述式

DELETE 陈述式的基本结构是比其它的 SQL 陈述式简单点，它的语法是：

```
DELETE table_or_view  
  
[FROM table_sources]  
  
[WHERE where_condition]
```

在 DELETE 陈述式之中没有数据行清单，选择性的 WHERE 子句允许您指定想要删除的数据列。

如果您忘记使用 WHERE 子句，那么您所指定的数据表或检视表中的所有数据列将会删除。

FROM 子句同样也是选择性的，允许您指定将被用在 WHERE 子句中选择基准的额外来源（数据表或检视表）。该陈述式的语法有点令人混淆，因为在 FROM 子句中所列的资料表或检视表中没有任何数据列会被删除。如果您在 FROM 陈述式内使用不只一个数据表或检视表时，在数据表或检视表的名称之间会以逗号分隔。

提示

DELETE 陈述式并不支持 JOIN 运算子，因此您必须在 WHERE 子句中将数据表或检视表进行联结。

当一个数据表与其它的数据表有关联性时，避免在主索引键数据表中删除操作时而造成「孤儿（orphan）」数据列是很重要的。所谓的「孤儿」数据列是指一在外部索引键数据表的数据列并没有与在主索引键数据表中的数据列相符合。SQL Server 2000 中的重大新功能即为关联性的串联删除。当您在关联性中指定串联删除时，SQL Server 将会在外部索引键数据表中自动删除剩下的孤儿资料列，图 19-1 显示了在 Oils 和 Plantparts 数据表之间的关联性。

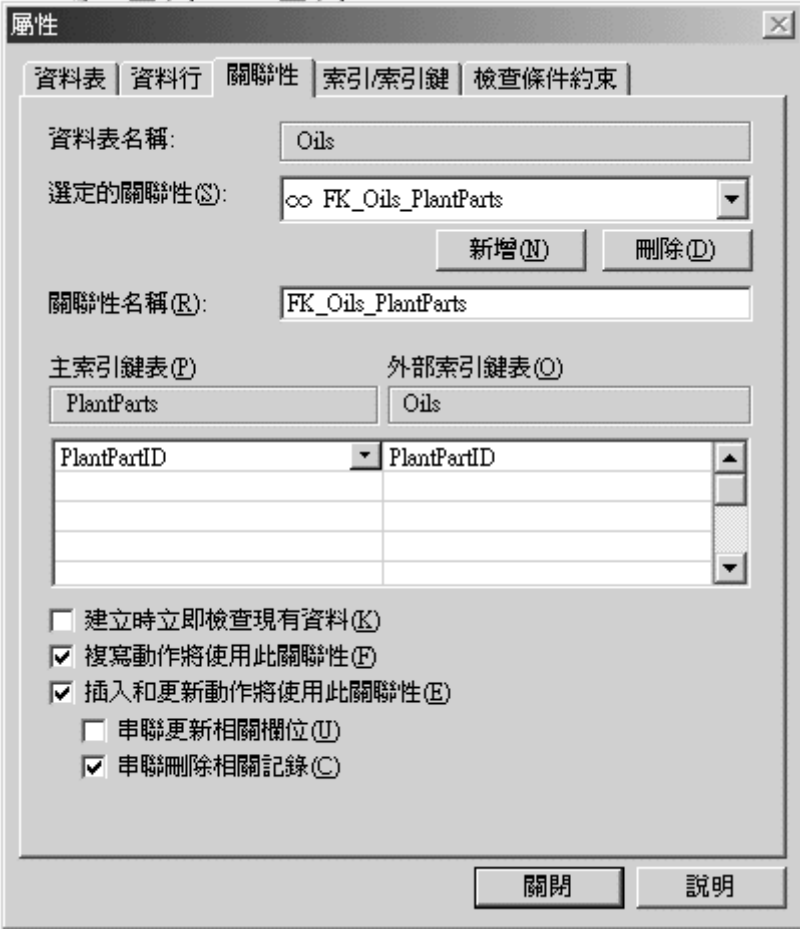


图 19-1 在数据表 属性 对话框中的 关联性 卷标页可以建置 串联删除相关纪录。

使用 DELETE 陈述式

就像其它在查询设计师内的 SQL 陈述式一样，DELETE 陈述式可以直接在 SQL 窗格内输入，

或者您也可以使用图表窗格来建立。

使用方格窗格和图表窗格来删除数据列

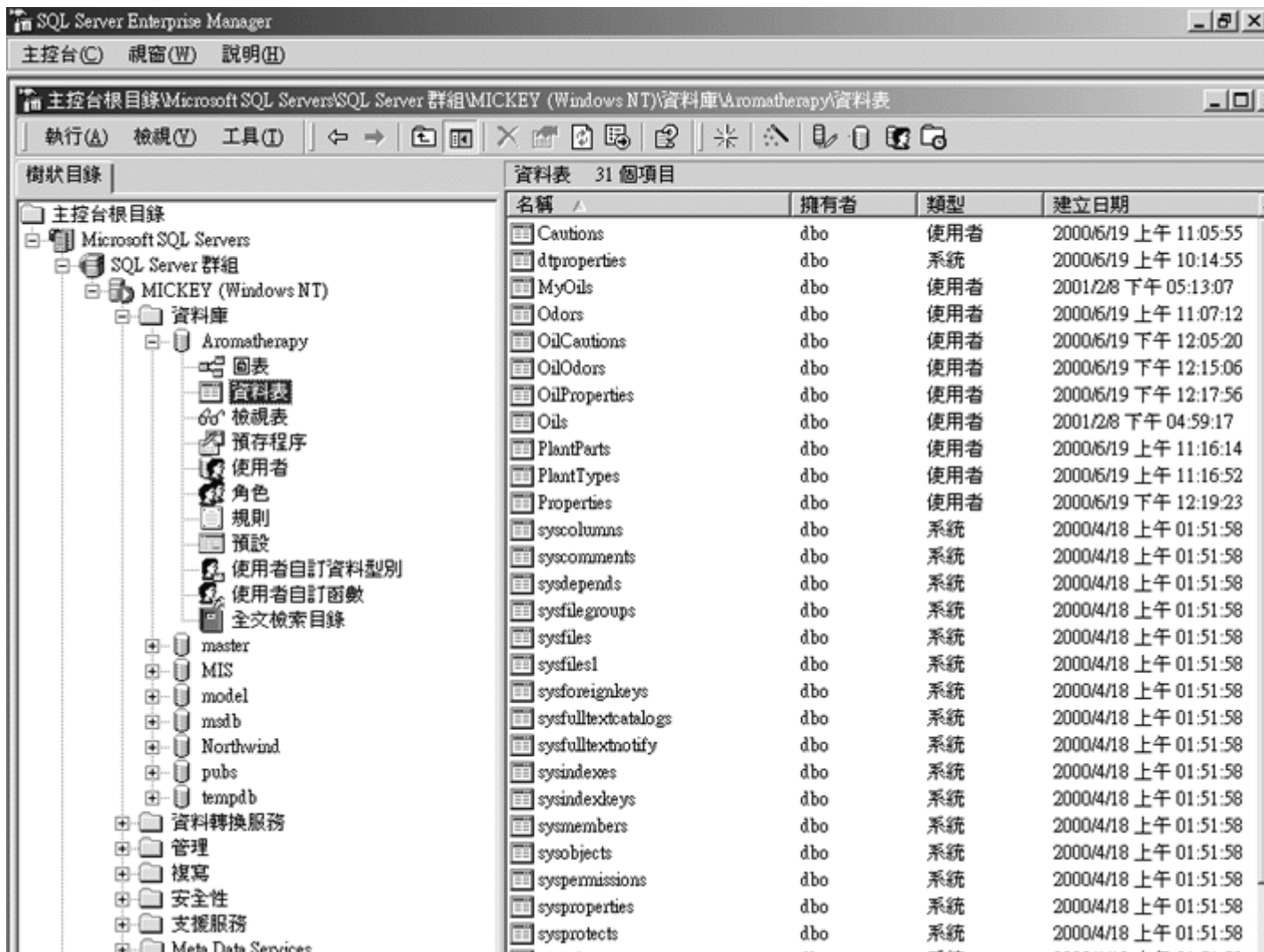
在查询设计师内的方格窗格和图表窗格都可以提供一个建立 DELETE 陈述式的图形化接口。然

而，它们并不支持 FROM 子句，因此您无法在 DELETE 陈述式内使用额外的数据表和检视表。

从数据表中删除选取的数据列

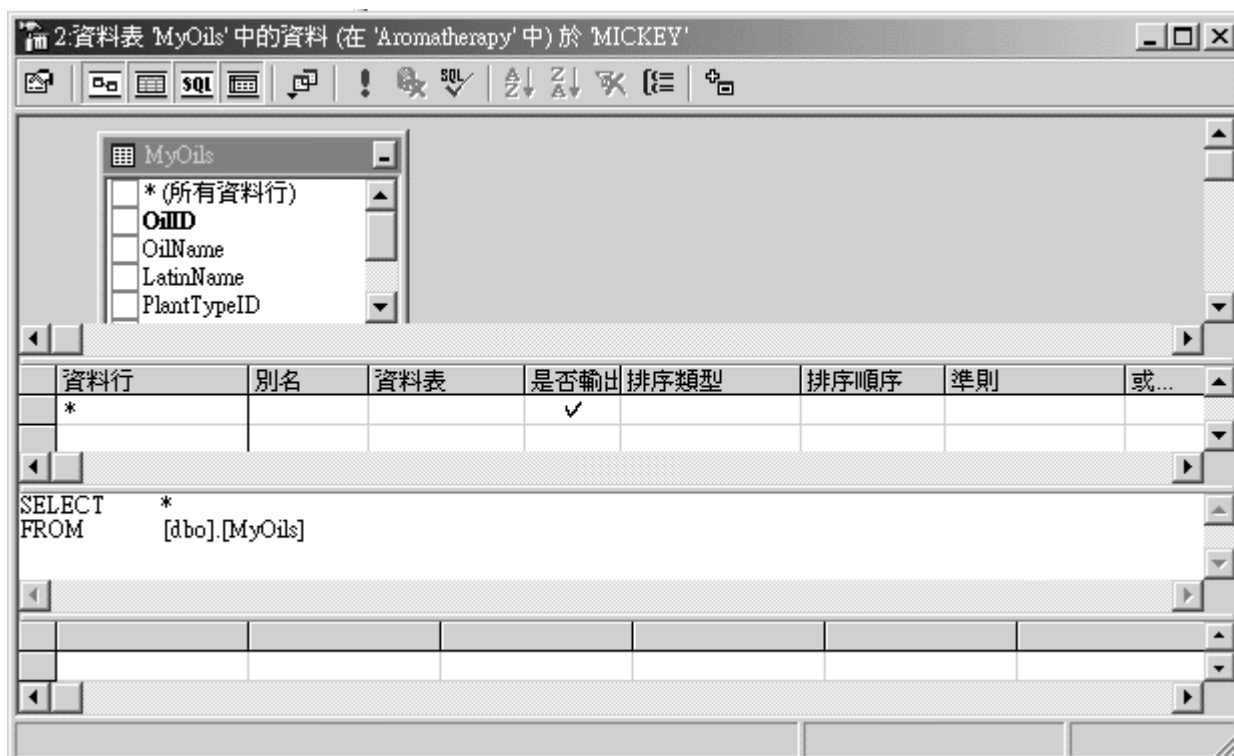
1. 选取 Aromatherapy 数据库中的 **数据表** 数据夹。

SQL Server 会在 **详细资料窗格** 中显示所有的数据表清单。



- 按一下 **MyOils** 数据表，接着指向 **开启数据表** 的子菜单并且指向 **查询**。

SQL Server 会开启 **查询设计师** 窗口。



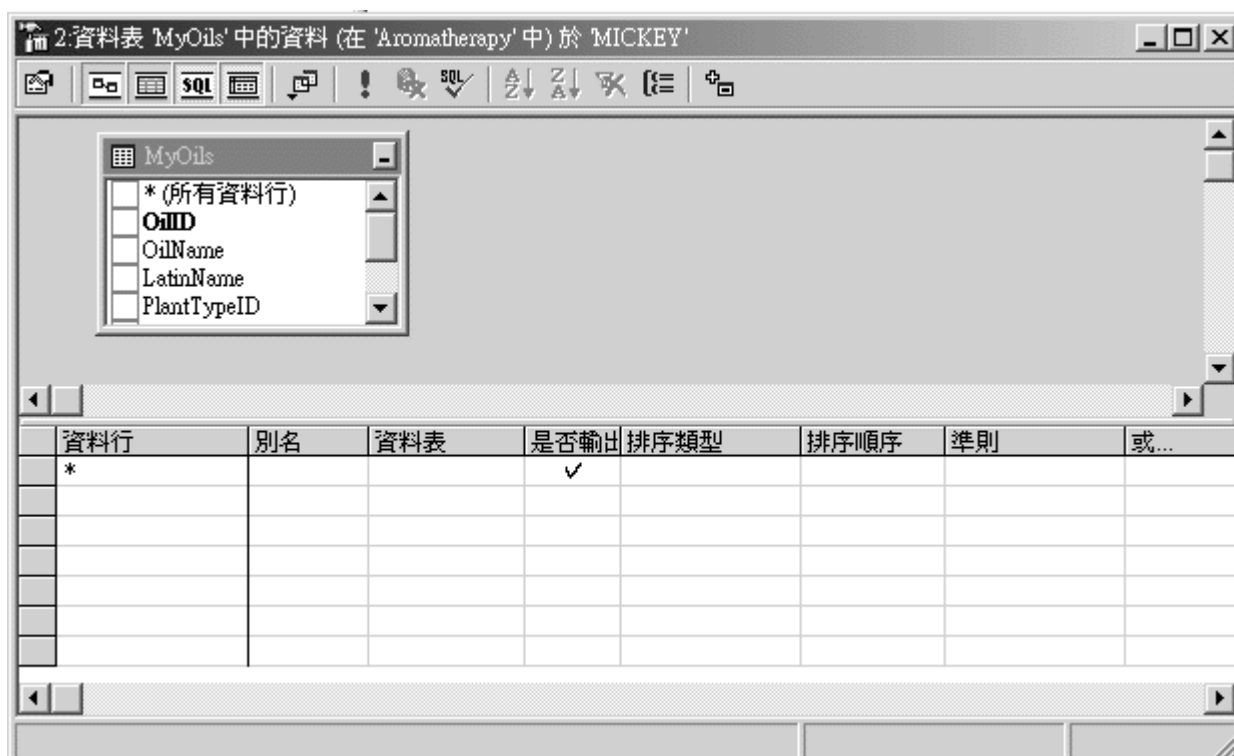
3. 将 **SQL 窗格** 隐藏，并且显示 **结果窗格**。



显示/隐藏 SQL 窗格按钮



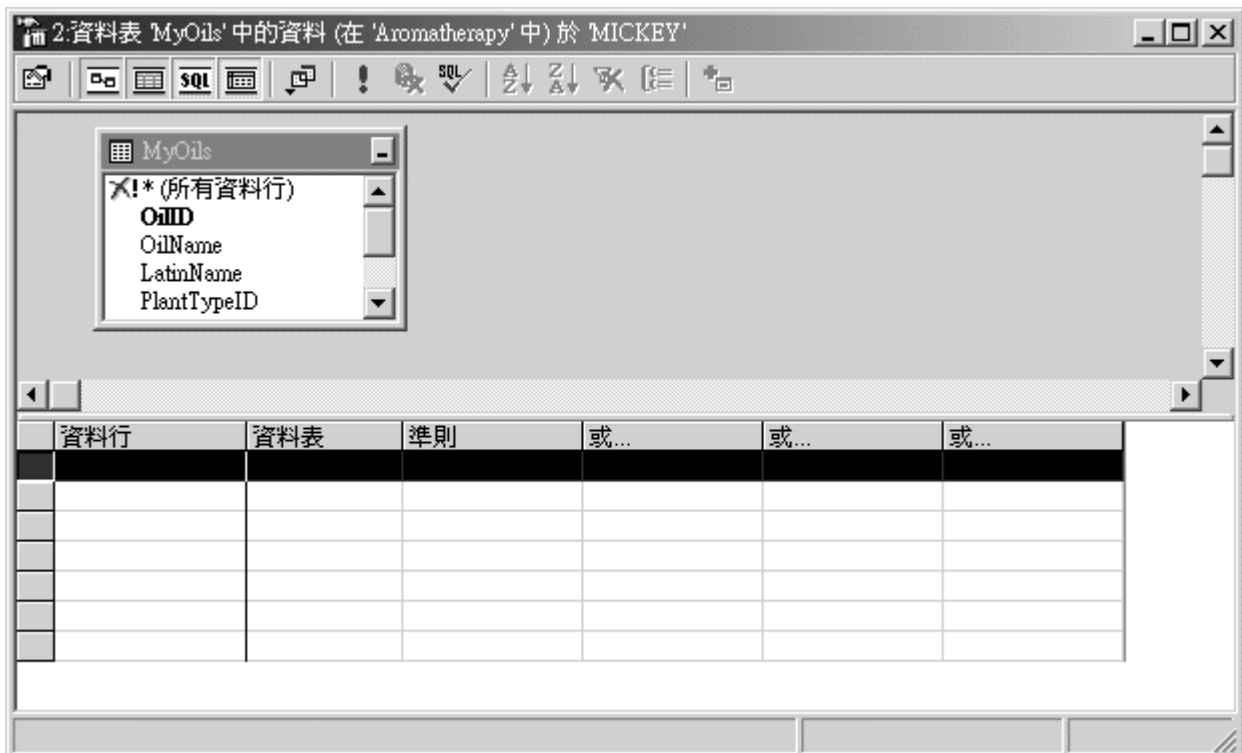
显示/隐藏结果窗格按钮



- 在 [查询设计师](#) 的工具列上按一下 [变更查询类型](#)，接着再选取 **DELETE**。



变更查询类型按钮



5. 不用将目前的 [查询设计师](#) 关闭。在 [详细数据窗格](#) 中的 **MyOils** 资料表上按右键，

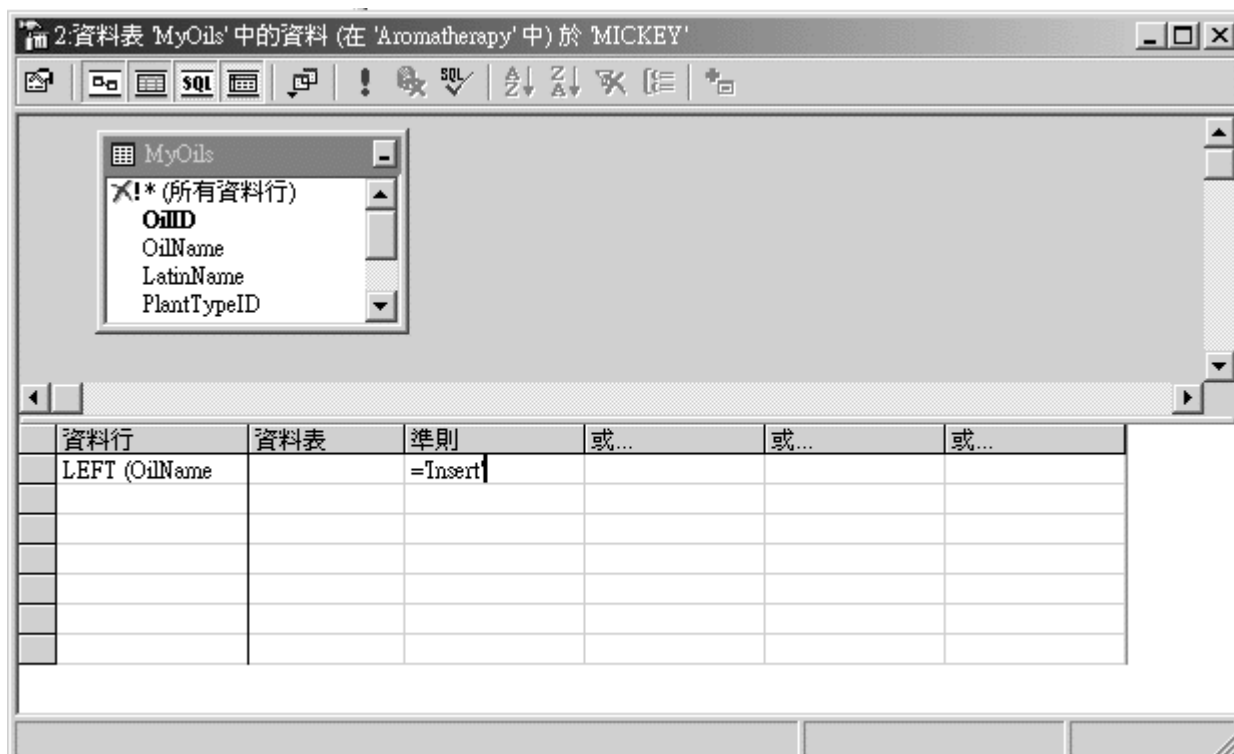
并且在 [开启数据表](#) 项目的子菜单内选择 [传回所有数据列](#)。
6. 卷动资料表中的滚动条，您将会看见四笔开头为 **Insert** 的资料列。

3. 資料表 'MyOils' 中的資料 (在 'Aromatherapy' 中) 於 'MICKEY'

OilID	OilName	LatinName	PlantTypeID	PlantPartID	Sample
40	Vetivert	Vetiveria Zizanoide	<NULL>	10	Sample Field
41	Ylang-ylang	Cananga Odorata	<NULL>	5	Sample Field
42	Fennel	Foeniculum Vulgan	<NULL>	14	Sample Field
43	Rose Otto	Rosa Damascena	<NULL>	5	Sample Field
44	Sage	Salvia Officinalis	<NULL>	1	Sample Field
45	Vanilla Oleoresin	Vanilla Planifolia	<NULL>	14	Sample Field
46	Benzoin	Styrax Benzoin	<NULL>	9	Sample Field
47	Linden	Tilia Vulgaris	<NULL>	5	Sample Field
48	InsertFromGrid	<NULL>	<NULL>	<NULL>	Sample Field
49	InsertFromSQL	<NULL>	<NULL>	<NULL>	Sample Field
50	InsertAllValues	Latin Name	<NULL>	1	Sample Field
51	InsertDefault	<NULL>	<NULL>	<NULL>	Sample Field
*					

7. 重新选取包含 DELETE 陈述式的 [查询设计师](#) 窗口。

8. 在数据行字段中新增 **Left(OilName, 6)**，在准则数据行中加入 **= 'Insert'**。



- 在 **查询设计师** 的工具列上按一下 **执行** 按钮，以便执行此查询。



执行按钮

查询设计师 会显示已经将四笔数据列删除的讯息。



10. 按一下 **确定** 按钮以便关闭此消息框。重新在 **查询设计师** 窗口内选择在 MyOils 数据表所有的数据列。
11. 在 **查询设计师** 的工具列上按一下 **执行** 按钮，以便重新执行 **SELECT *** 查询。
12. 卷动数据表中的滚动条，您将可以看见四笔开头为 **Insert** 的数据列已被移除。

3. 資料表 'MyOils' 中的資料 (在 'Aromatherapy' 中) 於 'MICKEY'

	OilID	OilName	LatinName	PlantTypeID	PlantPartID	Sample
	34	Rose	Rosa Centifolia	<NULL>	5	Sample Field
	35	Rosemary	Rosmarinus Officin	<NULL>	1	Sample Field
	36	Rosewood	Aniba Rosaeodora	<NULL>	4	Sample Field
	37	Sandalwood	Santalum Album	<NULL>	4	Sample Field
	38	Tea Tree	Melaleuca Alternifo	<NULL>	1	Sample Field
	39	Thyme	Thymus Vulgaris	<NULL>	1	Sample Field
	40	Vetivert	Vetiveria Zizanoide	<NULL>	10	Sample Field
	41	Ylang-ylang	Cananga Odorata	<NULL>	5	Sample Field
	42	Fennel	Foeniculum Vulgan	<NULL>	14	Sample Field
	43	Rose Otto	Rosa Damascena	<NULL>	5	Sample Field
	44	Sage	Salvia Officinalis	<NULL>	1	Sample Field
	45	Vanilla Oleoresin	Vanilla Planifolia	<NULL>	14	Sample Field
	46	Benzoin	Styrax Benzoin	<NULL>	9	Sample Field
	47	Linden	Tilia Vulgaris	<NULL>	5	Sample Field
*						

使用 SQL 窗格来删除数据列

跟往常一样，直接在 [查询设计师](#) 中的 **SQL 窗格** 输入 DELETE 陈述式比较具有弹性，但是您必须要注意陈述式的语法的正确性。

使用 WHERE 子句来删除数据列

1. 重新选取包含 DELETE 陈述式的 [查询设计师](#) 窗口。

2. 在工具列上按一下 [显示/隐藏 SQL 窗格](#) 按钮，以便显示 [SQL 窗格](#)；按一下 [显示/隐藏图表窗格](#) 以及 [显示/隐藏方格窗格](#) 按钮，以便将图表及方格窗格隐藏。



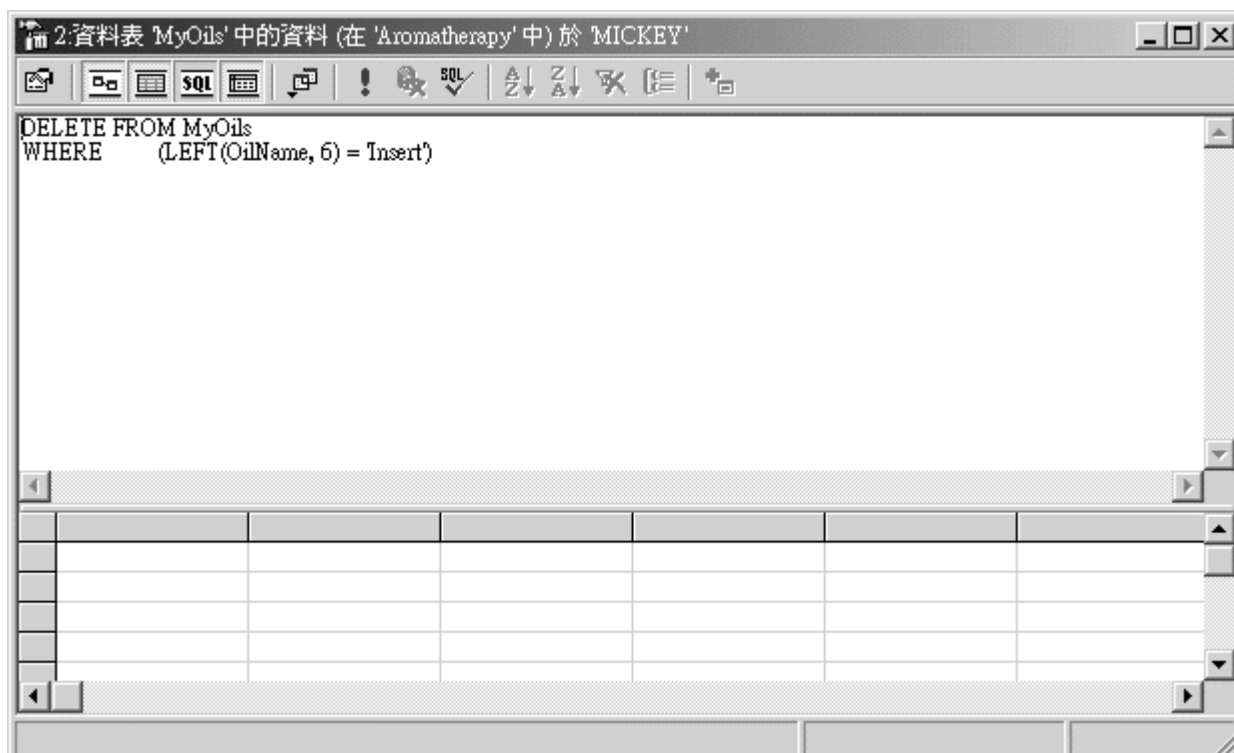
显示/隐藏 SQL 窗格按钮



显示/隐藏图表窗格按钮

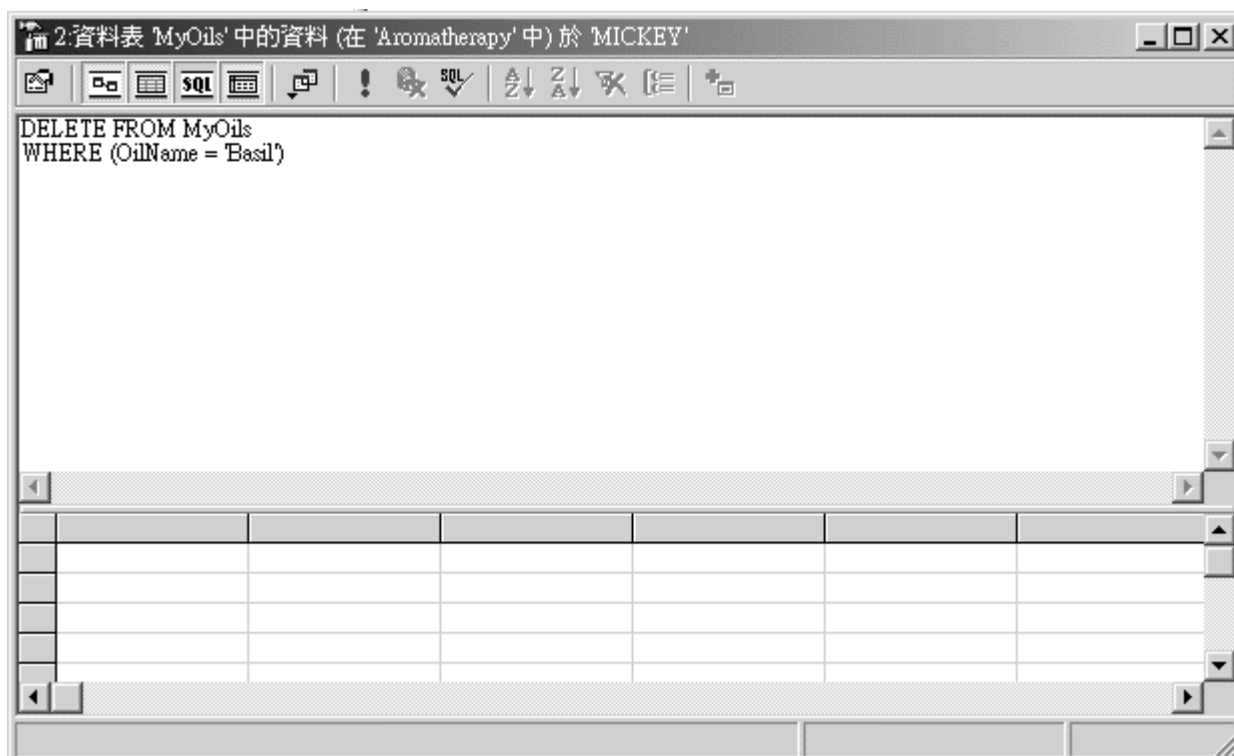


显示/隐藏方格窗格按钮



3. 将 SQL 陈述式内容更改为:

4. DELETE FROM MyOils
WHERE (OilName = 'Basil')



5. 在 [查询设计师](#) 的工具列上按一下 [执行](#) 按钮，以便执行此查询。



执行按钮

[查询设计师](#) 窗口会显示已经将数据列删除的讯息。



6. 重新在 [查询设计师](#) 窗口中选择在 **MyOils** 数据表中的所有数据列。
7. 在 [查询设计师](#) 的工具列上按一下 [执行](#) 按钮，以便重新执行 **SELECT *** 查询，您会发现其 **Basil** 的数据列已经被删除了。



执行按钮

3. 資料表 'MyOils' 中的資料 (在 'Aromatherapy' 中) 於 'MICKEY'

	OilID	OilName	LatinName	PlantTypeID	PlantPartID	Sample
▶	2	Bergamot	Citrus Bergamia	<NULL>	2	Sample Field
	3	Black Pepper	Piper Nigrum	<NULL>	3	Sample Field
	4	Cedarwood	Cedrus Atlantica, Ju	<NULL>	4	Sample Field
	5	German Chamomile	Matricaria Chamom	<NULL>	5	Sample Field
	6	Roman Chamomile	Anthemis Nobilis	<NULL>	5	Sample Field
	7	Cinnamon	Cinnamomum Zeyl	<NULL>	11	Sample Field
	8	Citronella	Cymbopogon Nard	<NULL>	11	Sample Field
	9	Clary Sage	Salvia Sclarea	<NULL>	5	Sample Field
	10	Coriander	Coriandrum Sativu	<NULL>	7	Sample Field
	11	Cypress	Cupressus Semperv	<NULL>	1	Sample Field
	12	Eucalyptus	Eucalyptus Globulu	<NULL>	1	Sample Field
	13	Frankincense	Buswellia Thurifera	<NULL>	9	Sample Field
	14	Geranium	Pelargonium Grave	<NULL>	1	Sample Field
	15	Ginger	Zingiber Officinale	<NULL>	10	Sample Field
	16	Grapefruit	Citrus Paradisi	<NULL>	2	Sample Field
	17	Jasmine	Jasminum Officin	<NULL>	5	Sample Field
	18	Juniper	Juniperus Commun	<NULL>	3	Sample Field
	19	Lavender	Lavandulus Officin	<NULL>	5	Sample Field

使用 FROM 子句来删除数据列

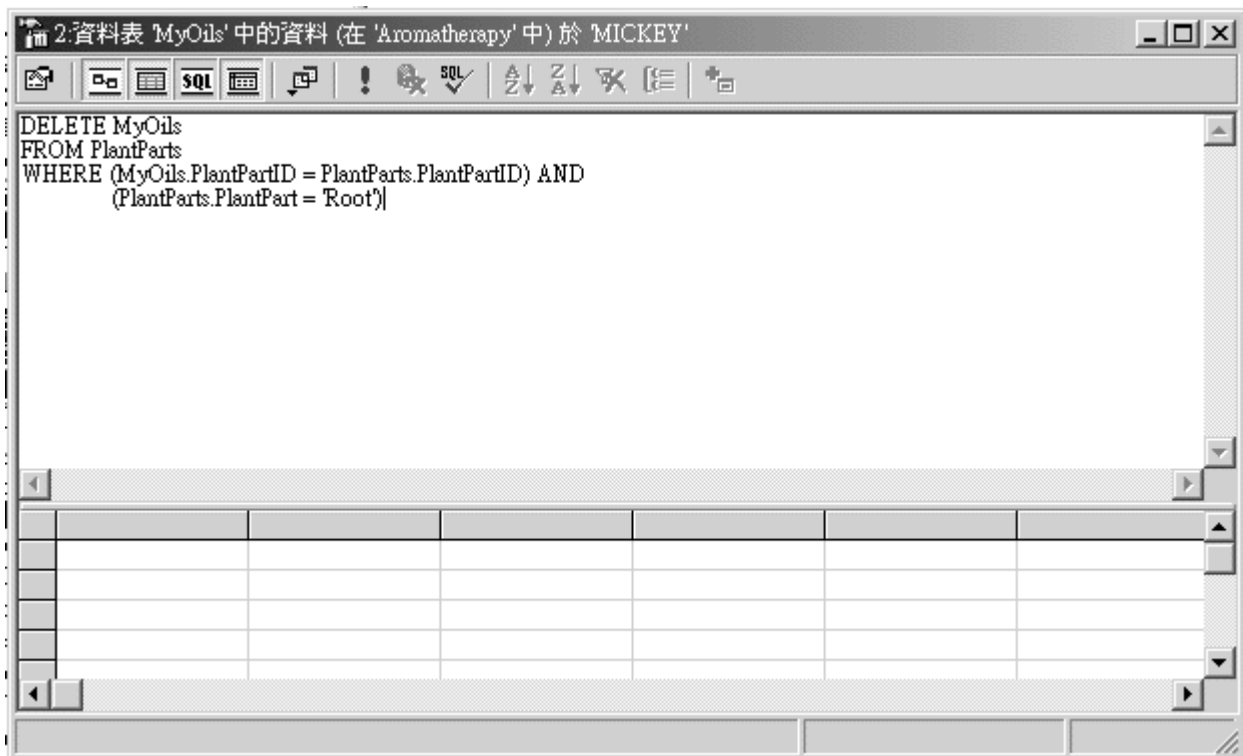
1. 重新选取包含 DELETE 陈述式的 [查询设计师](#) 窗口。

2. 将 SQL 陈述式内容更改为：

3. DELETE MyOils

4. FROM PlantParts

5. WHERE (MyOils.PlantPartID = PlantParts.PlantPartID) AND
(PlantParts.PlantPart = 'Roots')



6. 在 [查詢設計師](#) 的工具列上按一下 [執行](#) 按鈕，以便重新執行此查詢。



執行按鈕

查询设计师 会显示讯息来告知您，已经有二笔资料列已被删除。



7. 重新在 查询设计师 窗口中选择在 MyOils 数据表所有的数据列。
8. 在 查询设计师 的工具列上按一下 执行 按钮，以便重新执行 **SELECT *** 查询，并且确认引用 **PlantPartID 10**（Roots 数据列的 ID）的数据列已经被删除。



执行按钮

3. 资料表 'MyOils' 中的资料 (在 'Aromatherapy' 中) 於 'MICKEY'

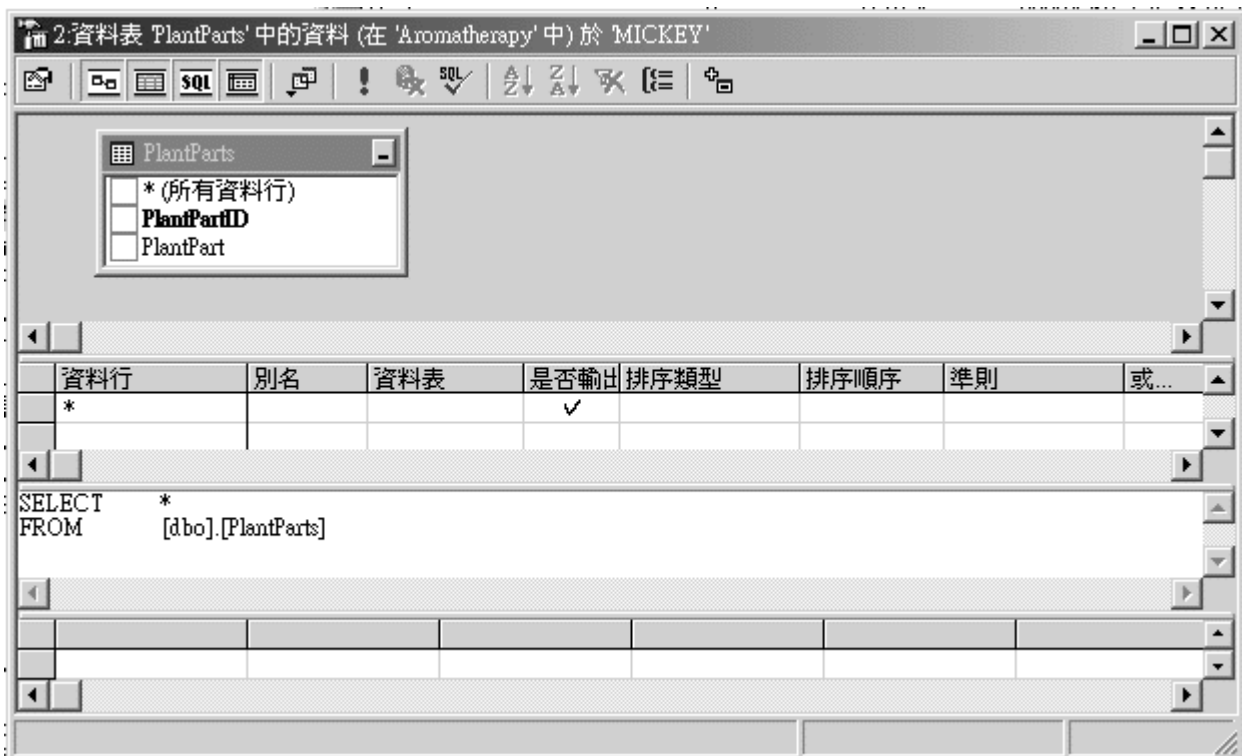
OilID	OilName	LatinName	PlantTypeID	PlantPartID	Sample
2	Bergamot	Citrus Bergamia	<NULL>	2	Sample Field
3	Black Pepper	Piper Nigrum	<NULL>	3	Sample Field
4	Cedarwood	Cedrus Atlantica, Jr	<NULL>	4	Sample Field
5	German Chamomile	Matricaria Chamom	<NULL>	5	Sample Field
6	Roman Chamomile	Anthemis Nobilis	<NULL>	5	Sample Field
7	Cinnamon	Cinnamomum Zeyl	<NULL>	11	Sample Field
8	Citronella	Cymbopogon Nard	<NULL>	11	Sample Field
9	Clary Sage	Salvia Sclarea	<NULL>	5	Sample Field
10	Coriander	Coriandrum Sativu	<NULL>	7	Sample Field
11	Cypress	Cupressus Semperv	<NULL>	1	Sample Field
12	Eucalyptus	Eucalyptus Globulu	<NULL>	1	Sample Field
13	Frankincense	Buswellia Thurifera	<NULL>	9	Sample Field
14	Geranium	Pelargonium Grave	<NULL>	1	Sample Field
16	Grapefruit	Citrus Paradisi	<NULL>	2	Sample Field
17	Jasmine	Jasminum Officina	<NULL>	5	Sample Field
18	Juniper	Juniperus Commun	<NULL>	3	Sample Field
19	Lavender	Lavandulus Officin	<NULL>	5	Sample Field
20	Lemon	Citrus limonum	<NULL>	2	Sample Field

9. 关闭 [查询设计师](#) 窗口。

使用串联删除来删除数据列

1. 在 Aromathrtapy 数据库内的 PlantParts 数据表上按右键, 并且指向 [开启数据表](#) 子

菜单中的 [查询](#)。



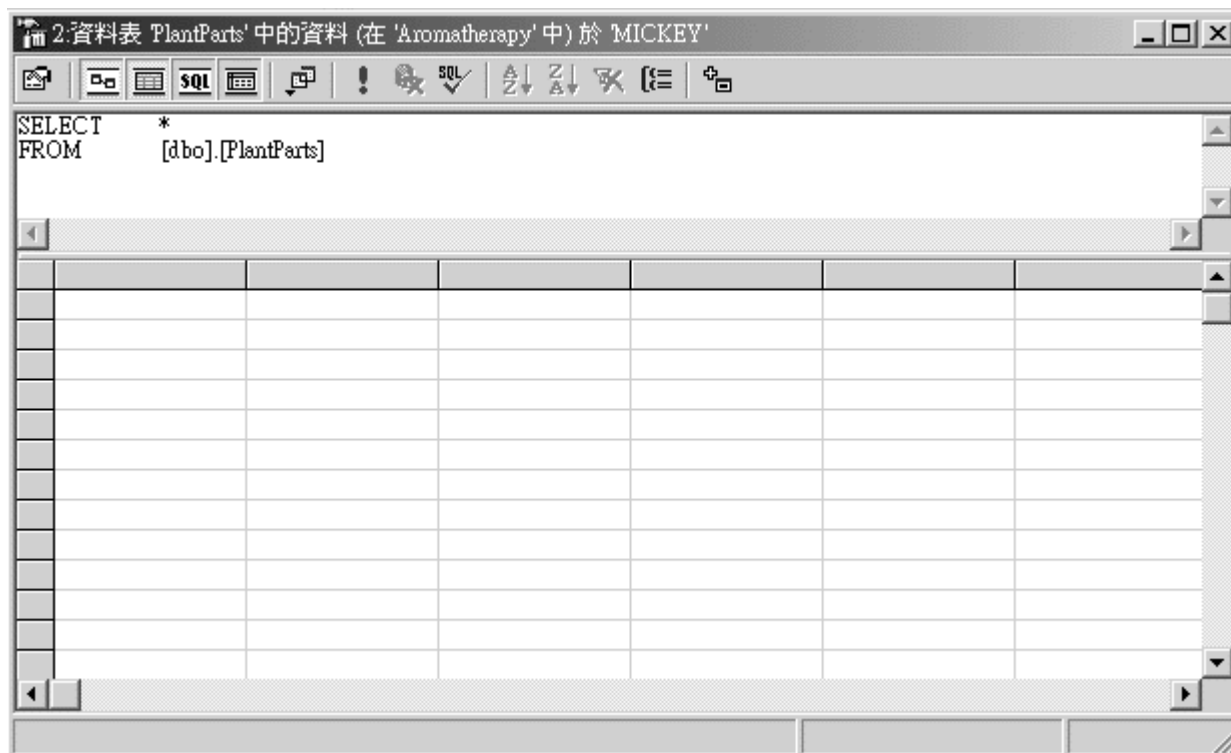
2. 将 [图表窗格](#) 及 [方格窗格](#) 隐藏。



显示/隐藏图表窗格按钮



显示/隐藏方格窗格按钮

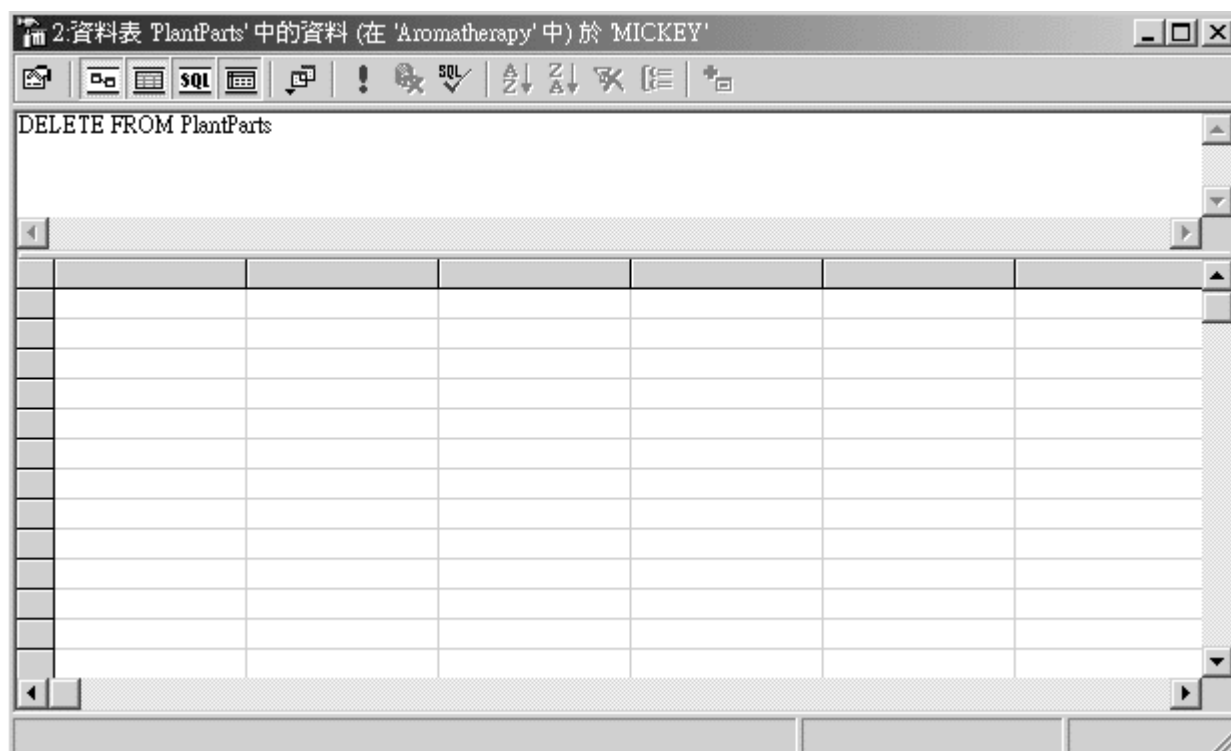


3. 在 [查询设计师](#) 中的工具列上按一下 [变更查询类型](#) 按钮，并选择 **DELETE** 。



变更查询类型按钮

此时在 [查询设计师](#) 中的 [SQL 窗格](#) 将会变更为 DELETE 陈述式。



4. 不用将目前的 [查询设计师](#) 窗口关闭，在 [详细数据窗格](#) 中的 [Oils](#) 资料表上按右
钮，并且在 [开启数据表](#) 的子菜单中选择 [传回所有数据列](#)。

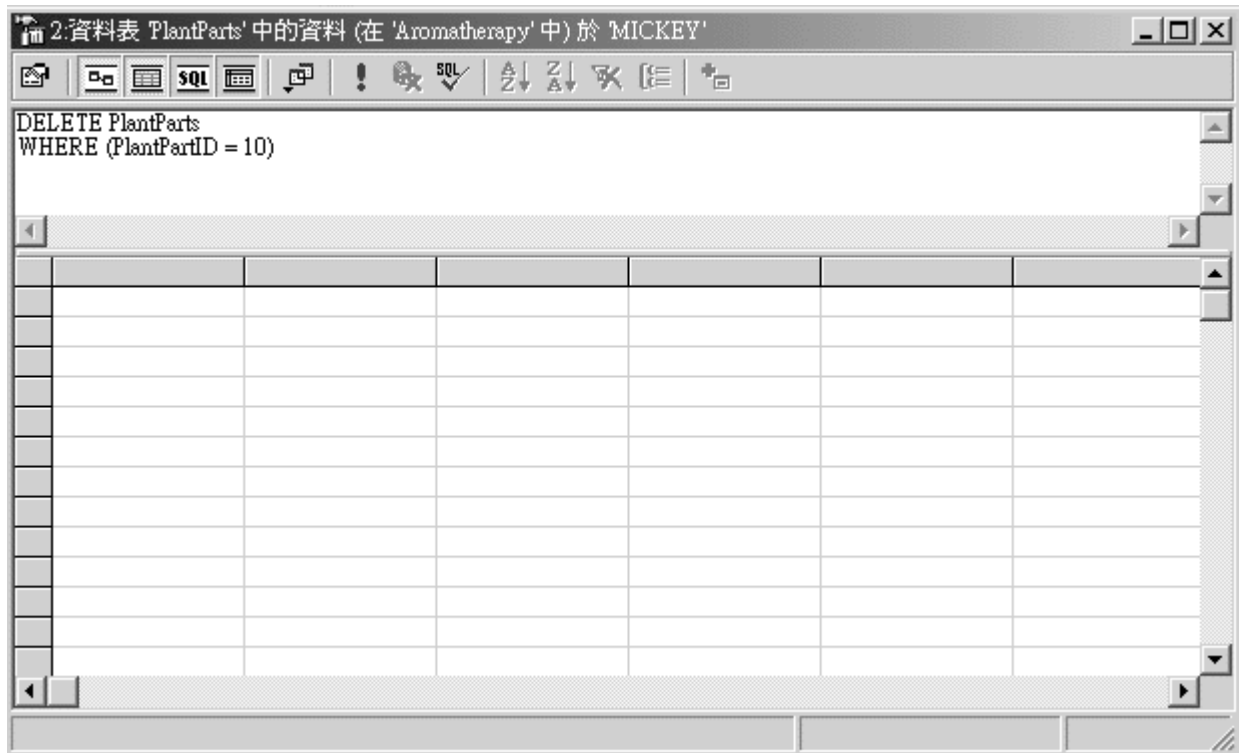
3:資料表 'Oils' 中的資料 (在 'Aromatherapy' 中) 於 'MICKEY'

	OilID	OilName	LatinName	PlantTypeID	PlantPartID	Sample
▶	1	Basil	Ocimum Basilicum	1	1	<NULL>
	2	Bergamot	Citrus Bergamia	2	2	<NULL>
	3	Black Pepper	Piper Nigrum	3	3	<NULL>
	4	Cedarwood	Cedrus Atlantica, Jr	2	4	<NULL>
	5	German Chamomile	Matricaria Chamom	1	5	<NULL>
	6	Roman Chamomile	Anthemis Nobilis	1	5	<NULL>
	7	Cinnamon	Cinnamomum Zeyl	4	11	<NULL>
	8	Citronella	Cymbopogon Nard	4	11	<NULL>
	9	Clary Sage	Salvia Sclarea	1	5	<NULL>
	10	Coriander	Coriandrum Sativum	1	7	<NULL>
	11	Cypress	Cupressus Semperv	2	1	<NULL>
	12	Eucalyptus	Eucalyptus Globulu	2	1	<NULL>
	13	Frankincense	Buswellia Thurifera	2	9	<NULL>
	14	Geranium	Pelargonium Graveol	5	1	<NULL>
	15	Ginger	Zingiber Officinale	5	10	<NULL>
	16	Grapefruit	Citrus Paradisi	2	2	<NULL>
	17	Jasmine	Jasminum Officinale	5	5	<NULL>
	18	Juniper	Juniperus Communis	5	3	<NULL>

5. 重新选取包含 DELETE 陈述式的 [查询设计师](#) 窗口。

6. 将 SQL 陈述式内容更改为：

```
7.      DELETE PlantParts
      WHERE (PlantPartID = 10)
```



8. 在 [查询设计师](#) 中的工具列上按一下 [执行](#) 按钮，以便执行此查询。



执行按钮

[查询设计师](#) 会显示讯息告知您，已经有资料列已被删除。



9. 重新在 [查询设计师](#) 窗口中选择在 Oils 数据表所有的数据列。
10. 在 [查询设计师](#) 的工具列上按一下 [执行](#) 按钮，以便重新执行 **SELECT *** 查询，并且确认引用 PlantPartID 10 的数据列都已经被删除了。



执行按钮

3.資料表 'Oils' 中的資料 (在 'Aromatherapy' 中) 於 'MICKEY'

	OilID	OilName	LatinName	PlantTypeID	PlantPartID	Sample
▶	1	Basil	Ocimum Basilicum	1	1	<NULL>
	2	Bergamot	Citrus Bergamia	2	2	<NULL>
	3	Black Pepper	Piper Nigrum	3	3	<NULL>
	4	Cedarwood	Cedrus Atlantica, Jr	2	4	<NULL>
	5	German Chamomile	Matricaria Chamom	1	5	<NULL>
	6	Roman Chamomile	Anthemis Nobilis	1	5	<NULL>
	7	Cinnamon	Cinnamomum Zeyl	4	11	<NULL>
	8	Citronella	Cymbopogon Nard	4	11	<NULL>
	9	Clary Sage	Salvia Sclarea	1	5	<NULL>
	10	Coriander	Coriandrum Sativum	1	7	<NULL>
	11	Cypress	Cupressus Semperv	2	1	<NULL>
	12	Eucalyptus	Eucalyptus Globulu	2	1	<NULL>
	13	Frankincense	Buswellia Thurifera	2	9	<NULL>
	14	Geranium	Pelargonium Grave	5	1	<NULL>
	16	Grapefruit	Citrus Paradisi	2	2	<NULL>
	17	Jasmine	Jasminum Officin	5	5	<NULL>
	18	Juniper	Juniperus Commun	5	3	<NULL>
	19	Lavender	Lavandulus Officin	5	5	<NULL>

11. 关闭 [查询设计师](#) 窗口。

使用 TRUNCATE TABLE 陈述式

使用 TRUNCATE TABLE 陈述式的结果与您使用 DELETE 陈述式并且不包含 WHERE 子句的结果

是一样的。一都会将数据表中的所有数据列删除。然而，使用 TRUNCATE TABLE 陈述式和

DELETE 陈述式并不相同，使用 TRUNCATE TABLE 陈述式并不会记录到交易记录文件之中，

因此可以快速地执行删除资料列的动作（效能会提升）。

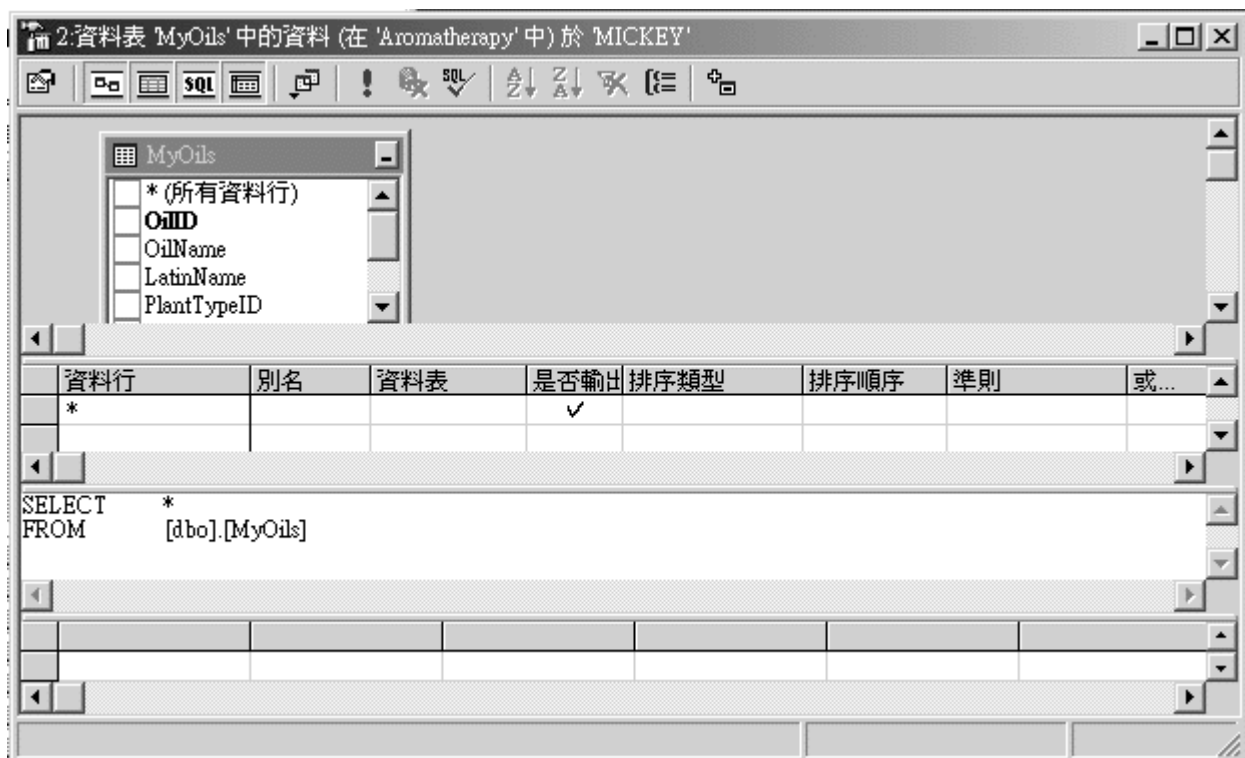
使用 TRUNCATE TABLE 陈述式来删除所有的数据列

TRUNCATE TABLE 陈述式只能在 [查询设计师](#) 中使用 [SQL 窗格](#) 执行。

使用 TRUNCATE TABLE 陈述式来删除所有的数据列

1. 在 [MyOils](#) 数据表上按右键，接着指向 [开启数据表](#) 子菜单中的 [查询](#)。

SQL Server 会开启 [查询设计师](#) 窗口。



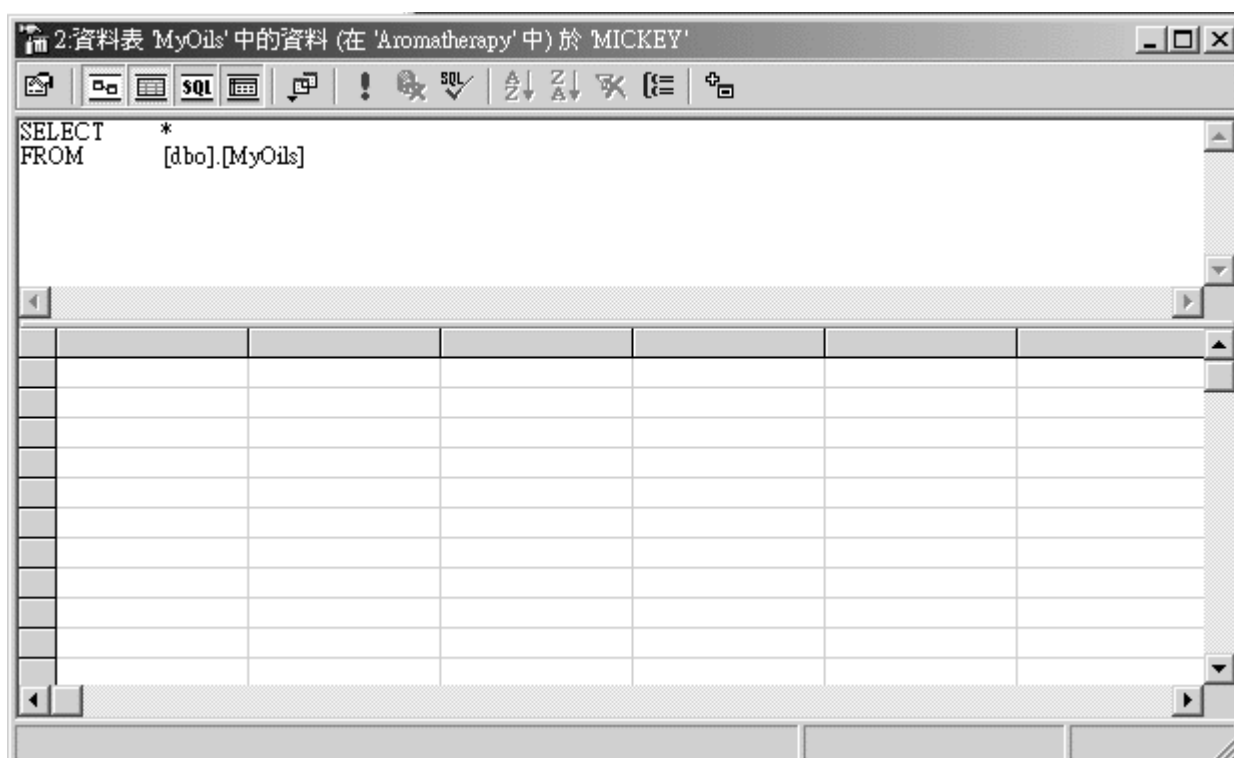
2. 将 [图表窗格](#) 及 [方格窗格](#) 隐藏。



显示/隐藏图表窗格按钮

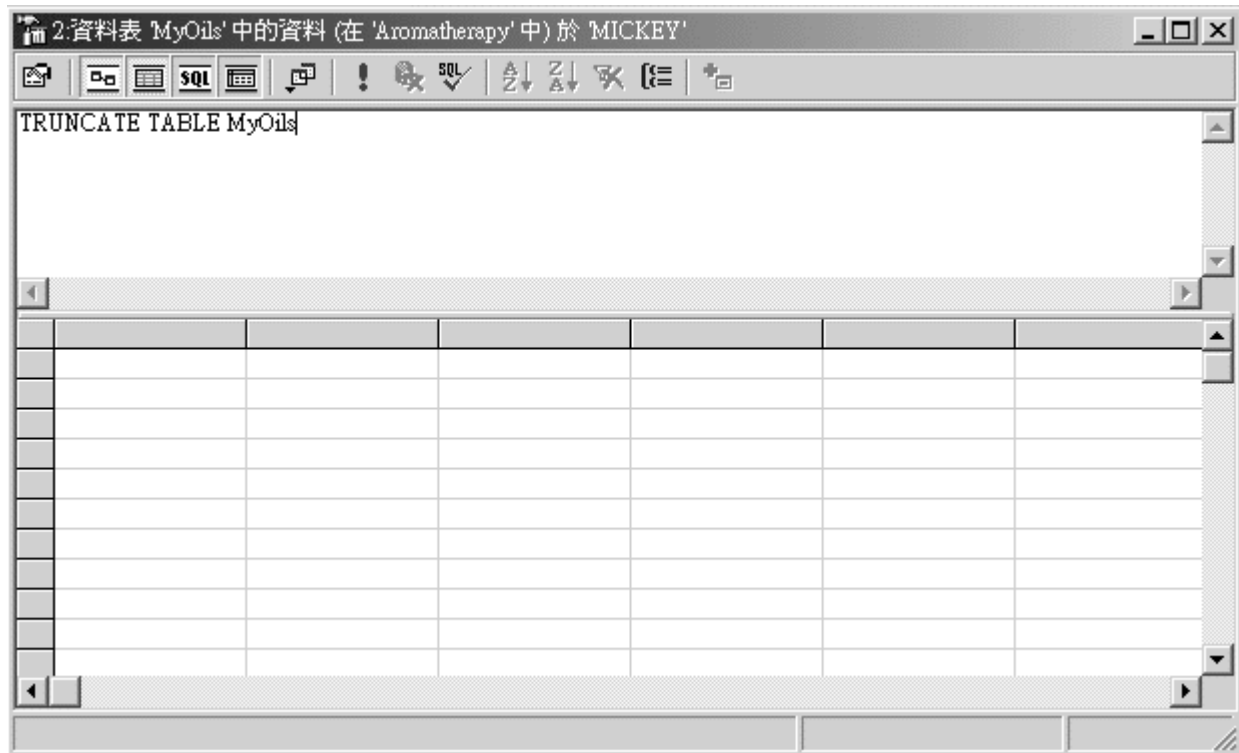


显示/隐藏方格窗格按钮



3. 将 SQL 陈述式内容更改为:

```
TRUNCATE TABLE MyOils
```



4. 不用将目前的 [查询设计师](#) 窗口关闭，在详细数据窗格内的 [MyOils](#) 上按右钮，并且在 [开启数据表](#) 子菜单中选择 [传回所有数据列](#)。

SQL Server Enterprise Manager - [3:資料表 'MyOils' 中的資料 (在 'Aromatherapy' 中...]							
主控制台(C) 視窗(W) 說明(H)							
	OilID	OilName	LatinName	PlantTypeID	PlantPartID	Sample	
	143	Bergamot	Citrus Bergamia	<NULL>	3	Sample Field	<
	144	Black Pepper	Piper Nigrum	<NULL>	4	Sample Field	<
	145	Cedarwood	Cedrus Atlantica, Ju	<NULL>	5	Sample Field	<
	146	German Chamomile	Matricaria Chamom	<NULL>	5	Sample Field	<
	147	Roman Chamomile	Anthemis Nobilis	<NULL>	11	Sample Field	<
	148	Cinnamon	Cinnamomum Zeyla	<NULL>	11	Sample Field	<
	149	Citronella	Cymbopogon Nardus	<NULL>	5	Sample Field	<
	150	Clary Sage	Salvia Sclarea	<NULL>	7	Sample Field	<
	151	Coriander	Coriandrum Sativum	<NULL>	1	Sample Field	<
	152	Cypress	Cupressus Sempervi	<NULL>	1	Sample Field	<
	153	Eucalyptus	Eucalyptus Globulus	<NULL>	9	Sample Field	<
	154	Frankincense	Buswellia Thurifera	<NULL>	1	Sample Field	<
	155	Geranium	Pelargonium Graveol	<NULL>	2	Sample Field	<
	156	Ginger	Zingiber Officinale	<NULL>	5	Sample Field	<
	157	Grapefruit	Citrus Paradisi	<NULL>	3	Sample Field	<
	158	Jasmine	Jasminum Officinale	<NULL>	5	Sample Field	<
	159	Juniper	Juniperus Communis	<NULL>	2	Sample Field	<
	160	Lavender	Lavandulus Officina	<NULL>	11	Sample Field	<
	161	Lemon	Citrus limonum	<NULL>	2	Sample Field	<
	162	Lemongrass	Cymbopogon Citratus	<NULL>	5	Sample Field	<
	163	Lime	Citrus Aurantifolia	<NULL>	1	Sample Field	<
	164	Marjoram	Origanum Marjoran	<NULL>	9	Sample Field	<
	165	Lemon Balm, Oshin	Melissa Officinale	<NULL>	<NULL>	<NULL>	<

5. 重新选取包含 TRUNCATE TABLE 陈述式的 [查询设计师](#) 窗口。

6. 在 [查询设计师](#) 的工具列上按一下 [执行](#) 按钮，以便执行查询。



执行按钮

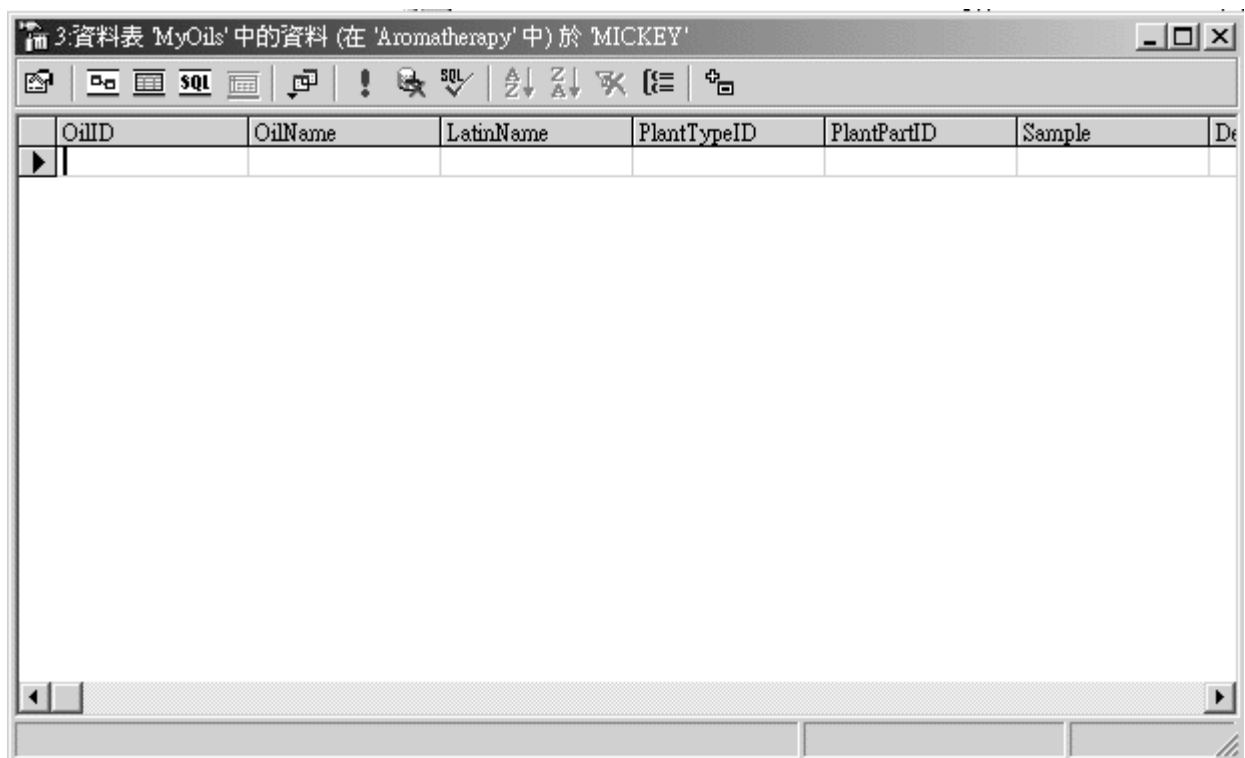
Enterprise Manager 会告知您已经执行成功。



7. 重新选取显示 **MyOils** 数据表中所有数据列的 **查询设计师** 窗口。
8. 在 **查询设计师** 的工具列上按一下 **执行** 按钮，以便重新执行 **SELECT *** 查询，并且确认所有的数据列已经被删除。



执行按钮



本章总结

要执行的工作	SQL 语法
使用 DELETE 陈述式来删除所有的数据列	DELETE table_or_view
使用 WHERE 子句来删除选取的数据列	DELETE table_or_view WHERE (where_condition)
使用基于其它数据表的准则来删除选取的数据列	DELETE table_or_view FROM table_sources WHERE (where_condition)

使用 TRUNCATE TABLE 陈述式来删除所有的数据列

```
TRUNCATE TABLE table_or_view
```


20. 复制及移动数据

在本章中，您将学习到：

- 使用 DTS 汇入精灵。
- 使用 DTS 汇出精灵。
- 从服务器卸离数据库。
- 附加数据库至服务器。
- 使用复制数据库精灵。

除了操作 SQL Server 数据库中的数据以外，有时候您也许需要转换数据为其它格式—例如 Microsoft Access 或 Oracle，某些时候也可能会需要在 SQL Server 之间复制数据。在本章中，我们将介绍由 Enterprise Manager 所提供三种不同的数据复制或移动数据的功能。

数据转换服务精灵

数据转换服务（Data Transformation Services, DTS）是一套功能非常强大的图形化工具，并且提供了可程序化对象可让您将汇入及汇出数据、转换数据结构，以及与针对分析和报表用途的原始数据进行整合而成为一个有用的信息。

由于提供了这么强大的功能，因此在使用 DTS 时会变得复杂化。但是很幸运的是 Enterprise Manager 提供了一个可以让您将汇入及汇出资料的精灵，让您在使用 DTS 功能时不至于十分复杂。

使用 DTS 汇入精灵

DTS 汇入精灵 可以让您将不同型态的数据汇入到 SQL Server 中，可以汇入的数据型态如下所示：

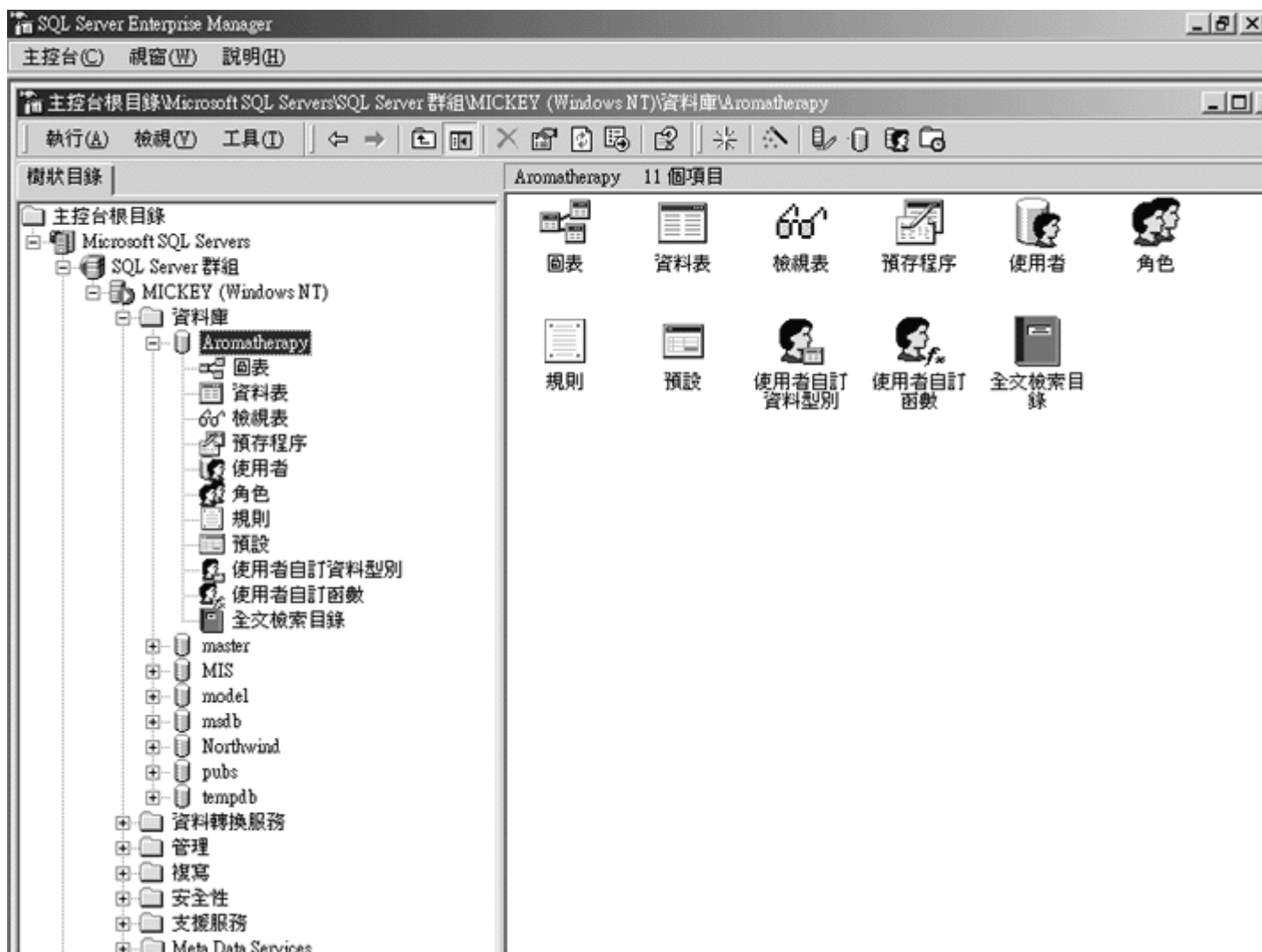
- OLE DB 和 ODBC 资料来源。
- 文本文件。
- 连接其它 Microsoft SQL Server 的执行个体。
- Oracle 和 Informix 数据库。

- Microsoft Excel 电子表格。
- Microsoft Access 和 Microsoft FoxPro 数据库。
- dBase 和 Padox 数据库。

使用 DTS 汇入精灵来汇入数据表

1. 在 Enterprise Manager 中，指向 **Aromatherapy** 数据库。

SQL Server 会显示数据库内所有对象的清单。

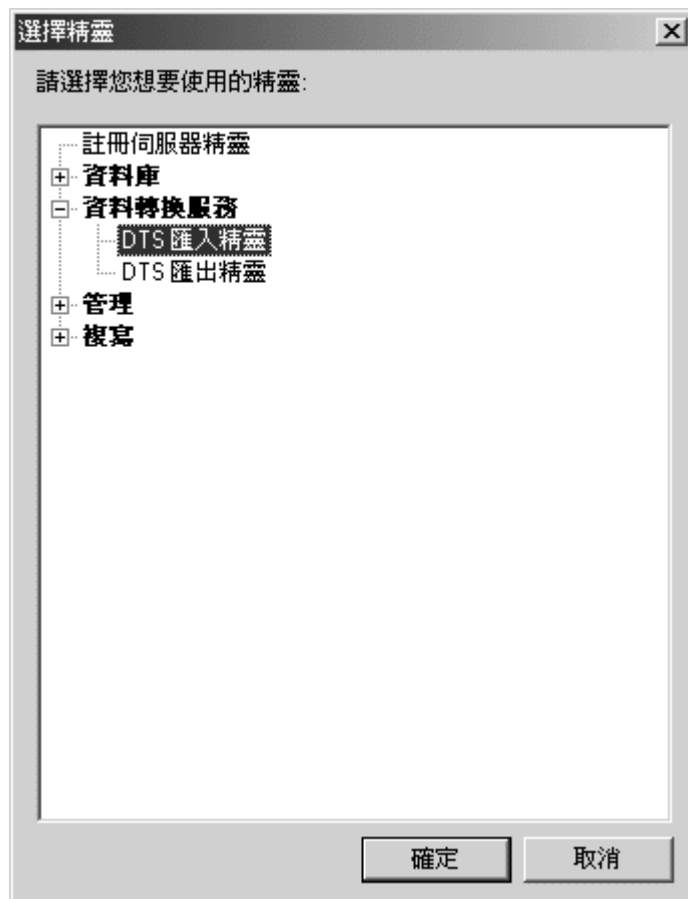


2. 在 Enterprise Manager 的工具列上按一下 [執行精靈](#) 按钮。



执行精灵按钮

SQL Server 会显示 [选择精灵](#) 的对话框。



3. 在 [数据转换服务](#) 中选择 [DTS 汇入精灵](#)，然后按一下 [确定](#) 按钮。

SQL Server 会显示 [DTS 汇入/汇出精灵](#) 的首页。



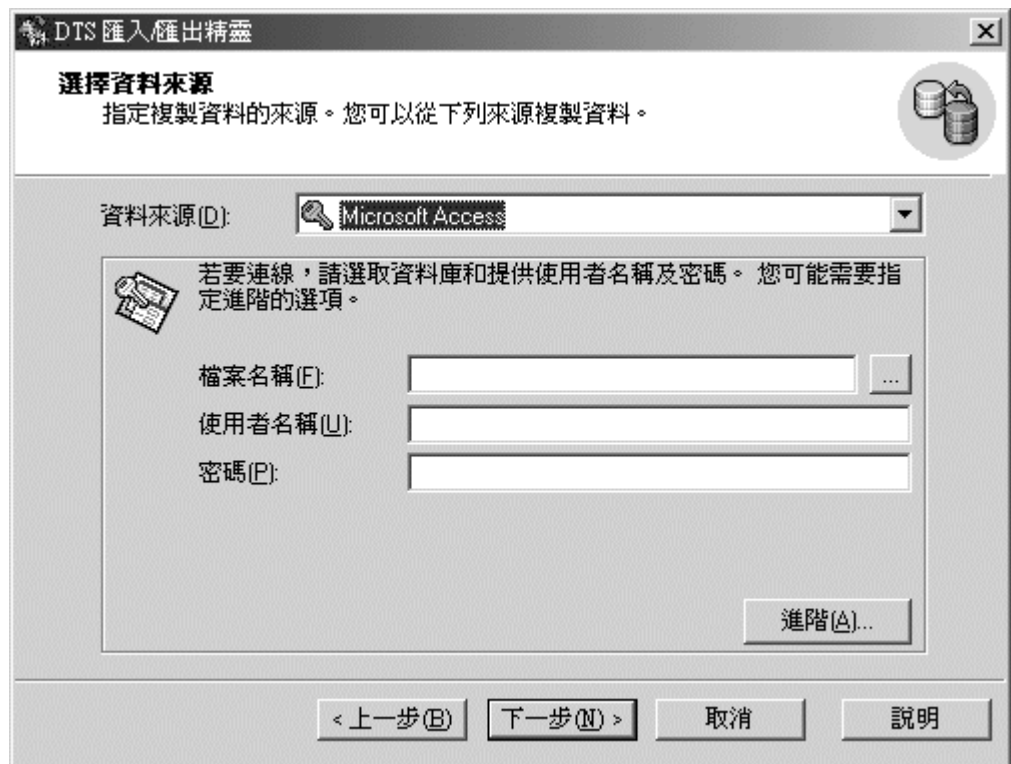
4. 按一下 **下一步** 按钮。

此时精灵会显示一个询问您要选择哪一个资料来源的画面，以便进行数据汇入/汇出的动作。



5. 请选择 **Microsoft Access** 来作为数据来源。

此精灵会显示此资料来源的内容。



6. 按一下 **浏览** 按钮以便指定数据来源档案的位置及名称。



浏览按钮

此时精灵会显示 **选择档案** 对话框。



7. 移动至根目录之下的 **SQL 2000 Step by Step** 数据夹，并且选

取 **Aromatherapy** 数据库。

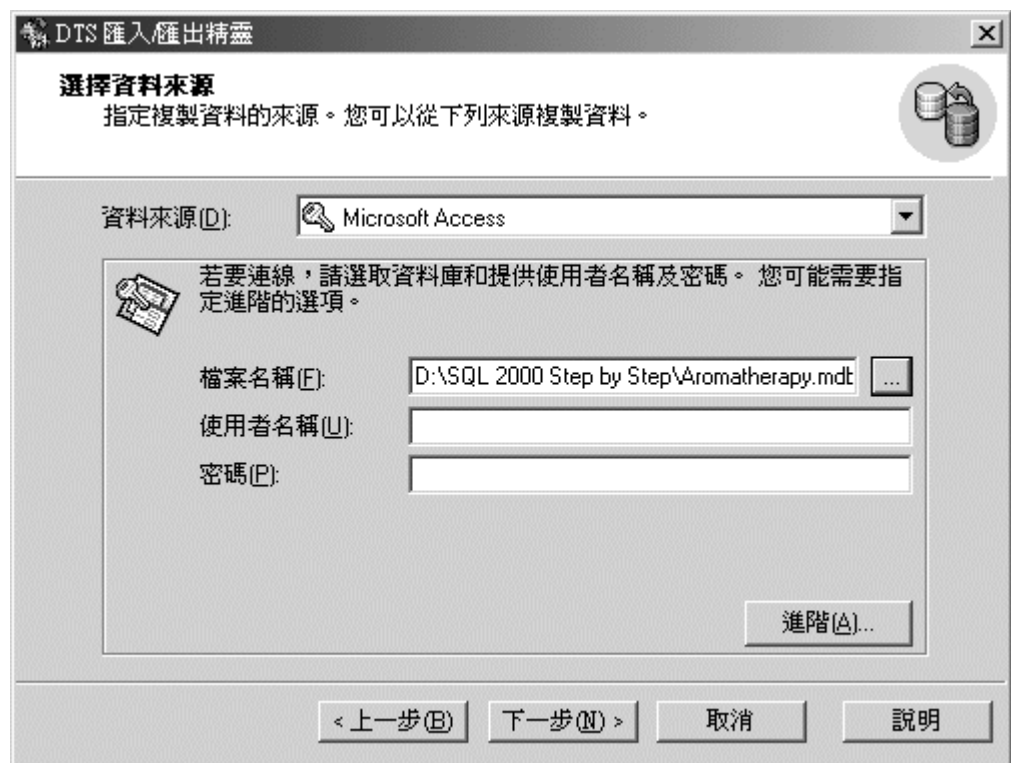


重要

在本书所附的光盘片中包含有二种版本的 **Aromatherapy** 数据库。因此您可以汇入 **Microsoft Access** 版本的数据库，而不是 **SQL Server** 版本的 **Aromatherapy** 数据库。

8. 按一下 **开启** 按钮。

此精灵会在 **文件名称** 文字方块中显示您所选取数据来源档案的位置及名称。



提示

DTS 汇入精灵的 [进阶](#) 按钮可让您设定 OLE DB 属性，通常您可以不需要设定。

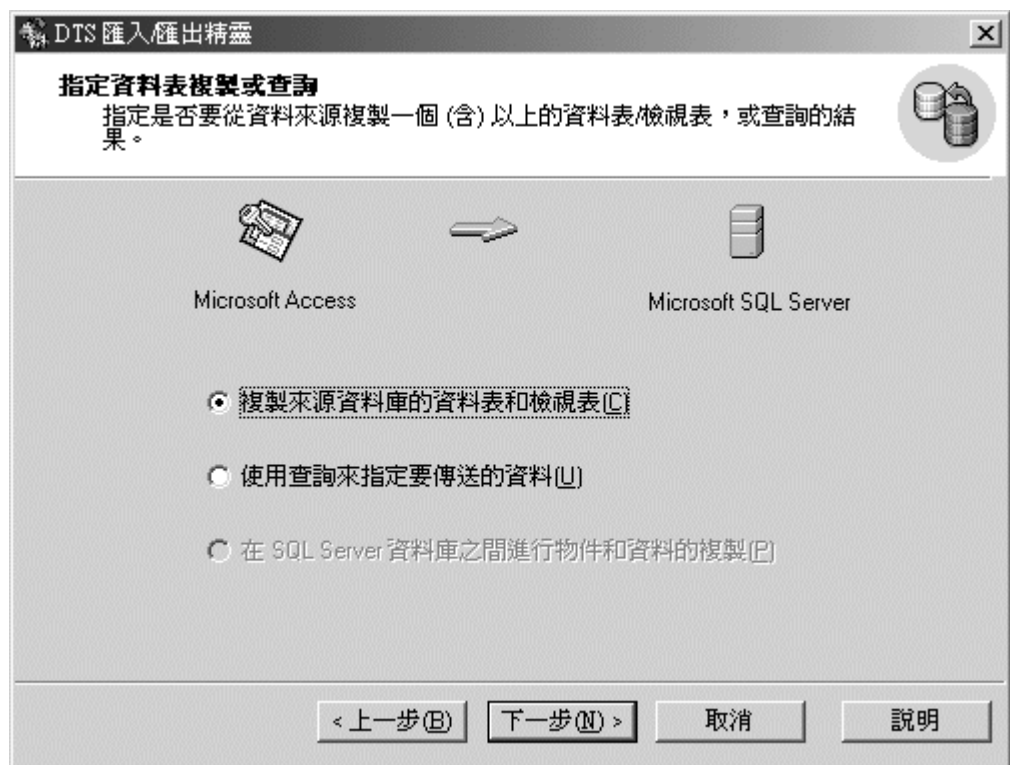
9. 按一下 [下一步](#) 按钮。

此精灵会显示一个画面来要求您指定资料汇入的目的地。



10. 当您在 Enterprise Manager 启动此精灵时，此精灵的预设是将数据汇入到启动时所选取的 SQL Server 数据库中。在此练习中，我们将使用该数据库，因此我们直接按一下 **下一步** 按钮，而不需要将设定作任何的改变。

此时精灵会显示一个询问您是否要将资料表、检视表或查询结果进行汇入的画面。



11. 我们将使用默认值，也就是从数据表来复制数据，接着我们按一下 [下一步](#) 按钮。

此精灵会显示在此数据库内所有的数据表。



提示

在前一页的另一个选项：[使用查询来指定要传送的资料](#)，当您选择该项目时，系统会显示一个对话框来要求您输入 SQL 陈述式。

-
- 选取 **NewOils** 数据表。

该精灵会以和选取数据表名称相同的名称来作为目的地数据表名称。



13. 在 [转换](#) 数据行内按一下 [浏览](#) 按钮。



浏览按钮

此时精灵会显示一个 [资料行对应与转换](#) 对话框。

資料行對應與轉換

來源: NewOils
目的地: [Aromatherapy].[dbo].[NewOils]

資料行對應 | **轉換**

☒ 建立目的資料表(B) 編輯 SQL(S)...

☐ 刪除目的資料表中的資料列(W) ☐ 卸除並重新建立目的資料表(D)

☐ 附加資料列到目的資料表(P) ☐ 啓用識別插入(I)

對應(M):

來源	目的地	類型	允許爲...	大小	有效位數	小數位數
OilID	OilID	int	<input type="checkbox"/>			
OilName	OilName	nvarchar	<input checked="" type="checkbox"/>	50		
Latin Name	Latin Name	nvarchar	<input checked="" type="checkbox"/>	50		
PlantType	PlantType	int	<input checked="" type="checkbox"/>			
PlantPart	PlantPart	int	<input checked="" type="checkbox"/>			

來源資料行: OilID Long NOT NULL

確定 取消 說明(H)

14. 将 OilName 数据行的大小更改为 [25](#)，这样只会将 OilName 数据行的开端数据汇入。

資料行對應與轉換

來源: NewOils
目的地: [Aromatherapy].[dbo].[NewOils]

資料行對應 | 轉換

☒ 建立目的資料表(B) 編輯 SQL(S)...
☐ 刪除目的資料表中的資料列(W) ☐ 卸除並重新建立目的資料表(D)
☐ 附加資料列到目的資料表(P) ☐ 啓用識別插入(I)

對應(M):

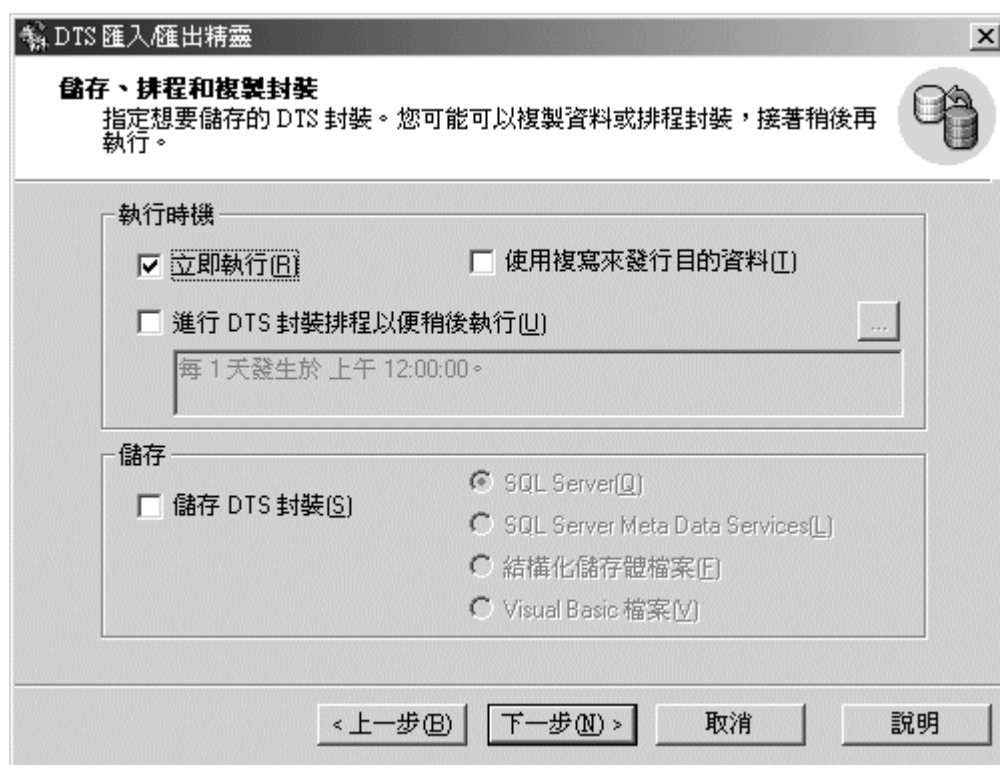
來源	目的地	類型	允許爲...	大小	有效位數	小數位數
OilID	OilID	int	<input type="checkbox"/>			
OilName	OilName	nvarchar	<input checked="" type="checkbox"/>	25		
Latin Name	Latin Name	nvarchar	<input checked="" type="checkbox"/>	50		
PlantType	PlantType	int	<input checked="" type="checkbox"/>			
PlantPart	PlantPart	int	<input checked="" type="checkbox"/>			

來源資料行: OilName VarChar(50) NULL

確定 取消 說明(H)

15. 按一下 **確定** 按钮，以便关闭此对话框。接着再按一下 **下一步** 按钮。

此精灵会要求您指定是要立即执行汇入数据，还是稍后再执行汇入数据的动作。

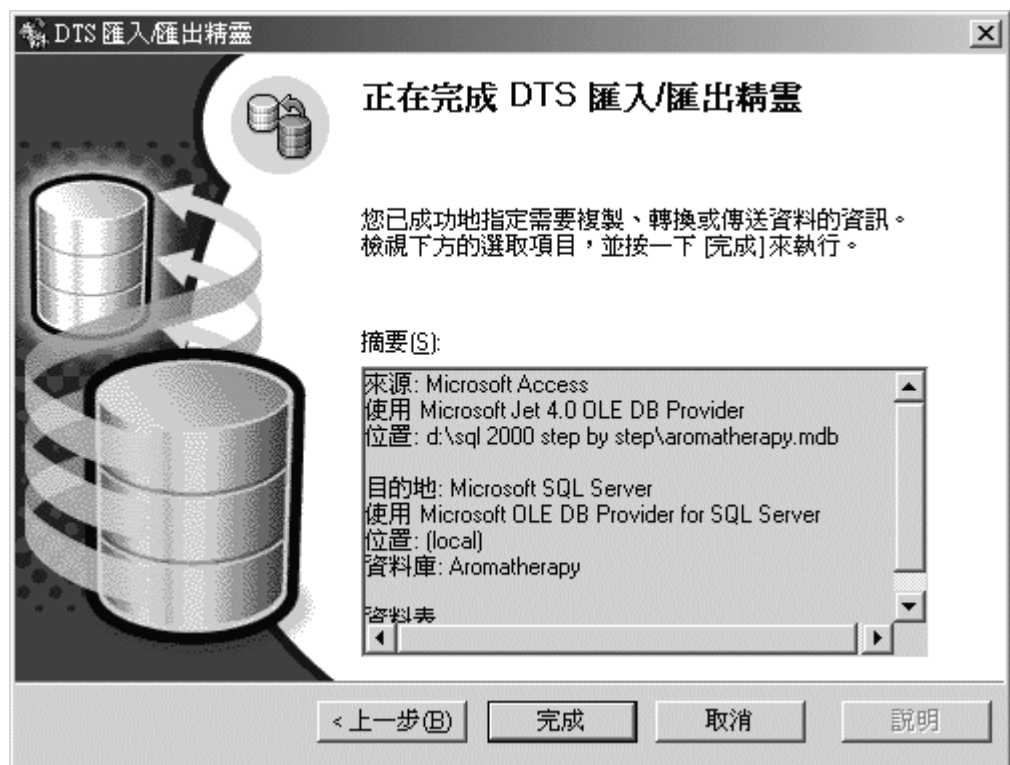


提示

本画面也可以让您将所要汇入的数据当作是一个 **DTS** 封装来储存。如果您需要再次执行汇入数据工作时，这个功能是非常好用的。

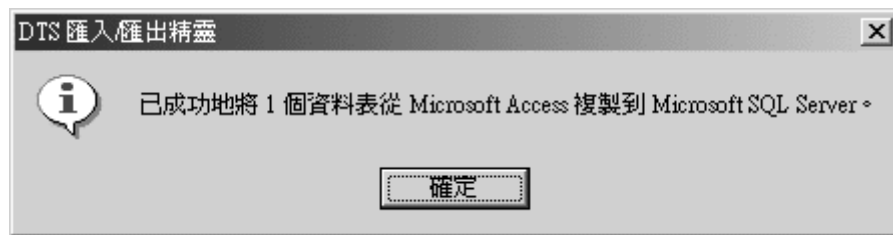
-
- 我们将采用默认值 **立即执行**，然后按一下 **下一步** 按钮。

此精灵会显示一个您所设定内容的画面。

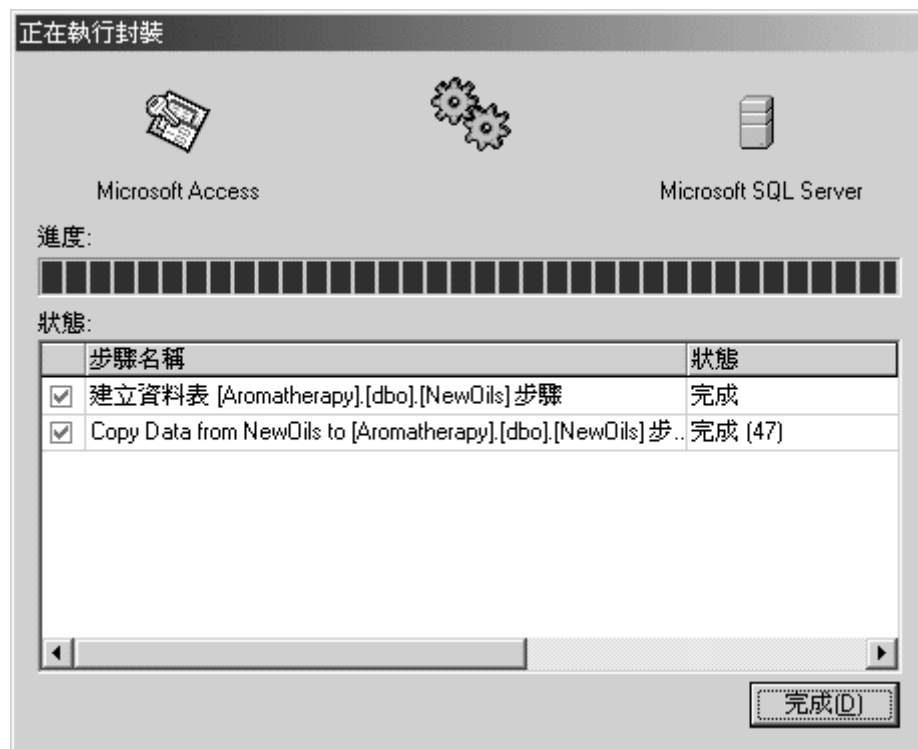


17. 按一下 **完成** 按钮。

当执行 DTS 封装时，此精灵会显示一个处理进度的对话框。当它处理完毕之后，
会显示一个此资料表已汇入成功的讯息。



18. 按一下 **確定** 按钮，以便关闭此对话框。



19. 按一下 **完成** 按钮，以便关闭此精灵。

20. 在 Enterprise Manager 中选择 **Aromatherapy** 数据库内的 **数据表** 数据夹，您可以在详细数据窗格中看到所汇入的 **NewOils** 资料表。



使用 DTS 汇出精灵

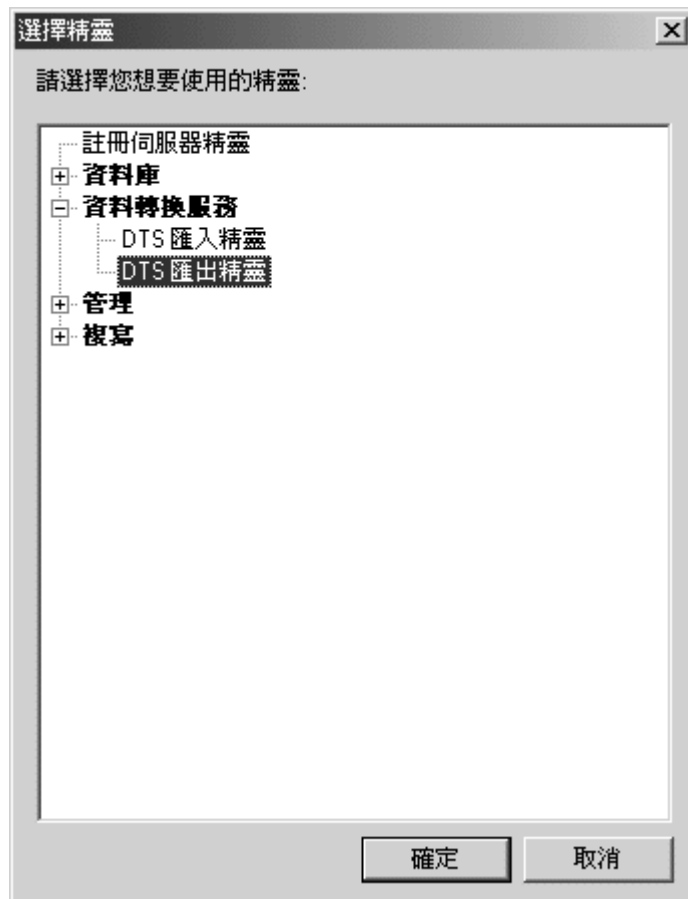
当您想要将数据库内的数据汇出时，相同地，您可以使用 DTS 汇出精灵将资料进行汇出。

使用 DTS 汇出精灵来汇出数据表

1. 在 Enterprise Manager 中，指向 **Aromatherapy** 数据库。



执行精灵按钮



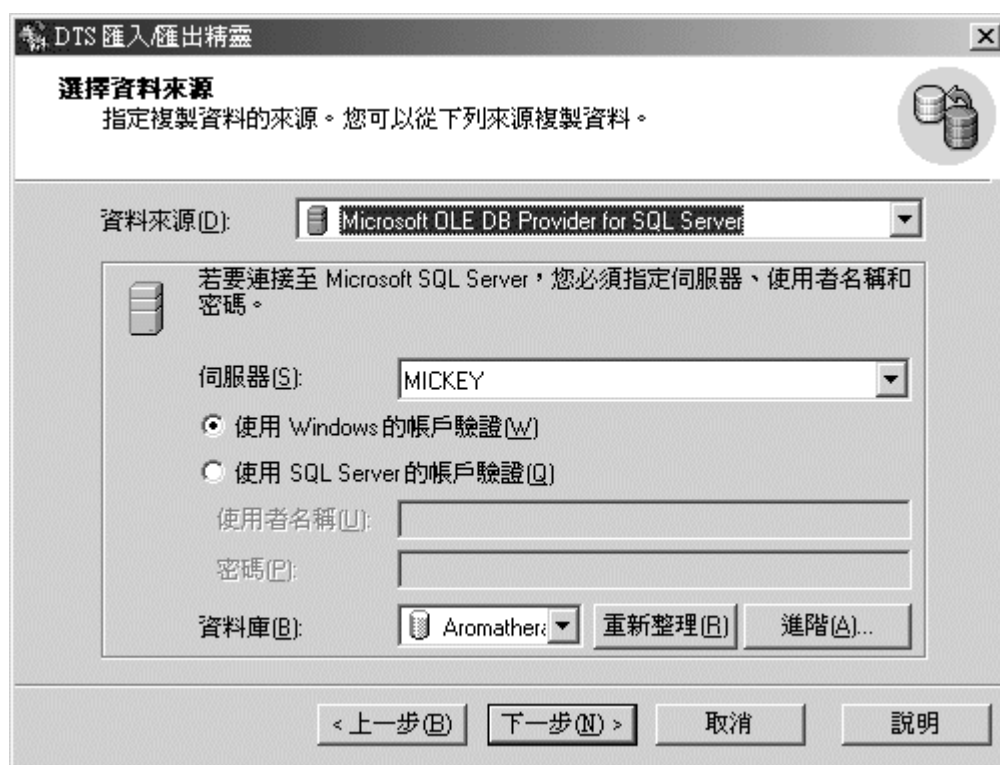
2. 在 [数据转换服务](#) 中选取 [DTS 汇出精灵](#)，然后按一下 [确定](#) 按钮。

SQL Server 会显示 [DTS 汇入/汇出精灵](#) 的首页。



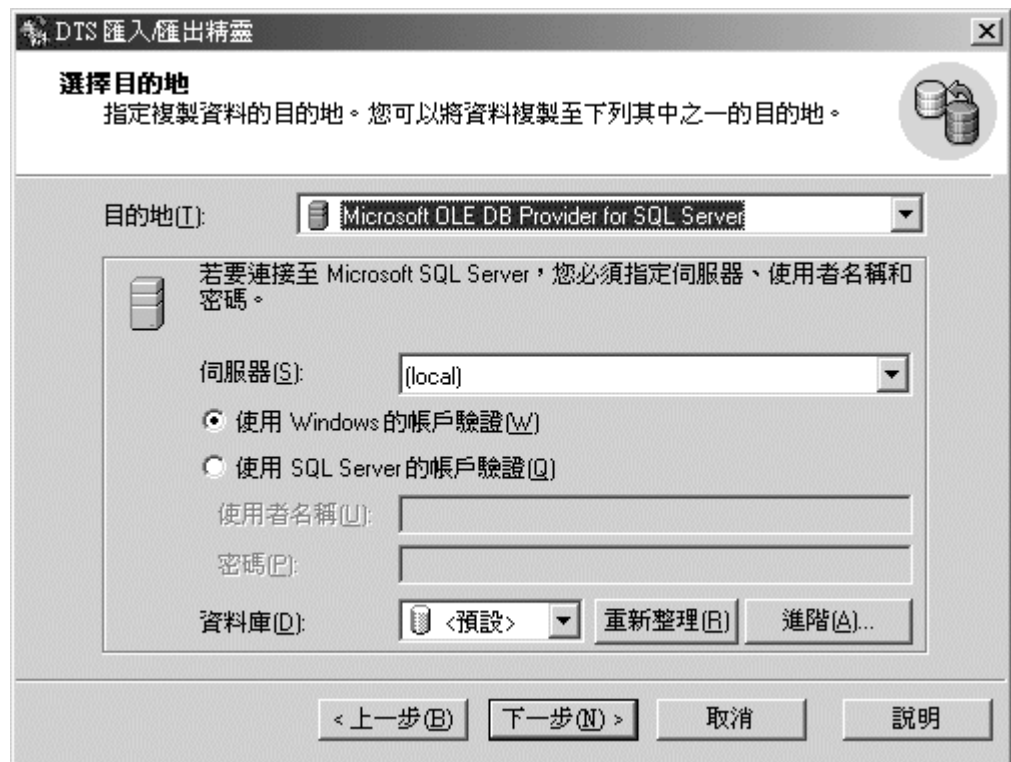
3. 按一下 **下一步** 按钮。

此时精灵会询问您要选择哪一个数据来源，以便进行数据汇入/汇出的动作。

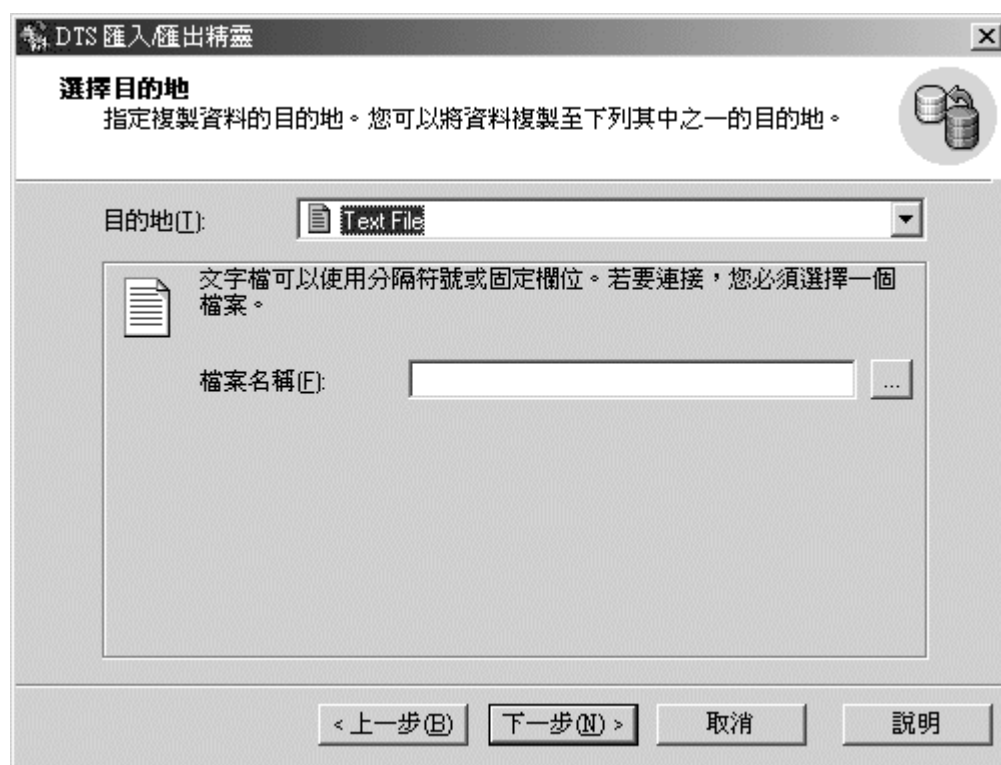


4. 确定您已经选取 **Aromatherapy** 数据库，然后按一下 **下一步** 按钮。

此精灵会显示一个画面来要求您指定资料汇出的目的地。



5. 选取 **Text File** 来当作是汇出数据的目的地。



6. 在 [文件名称](#) 文字方块中按一下 [浏览](#) 按钮。



浏览按钮

此精灵会显示 [选择档案](#) 对话框。

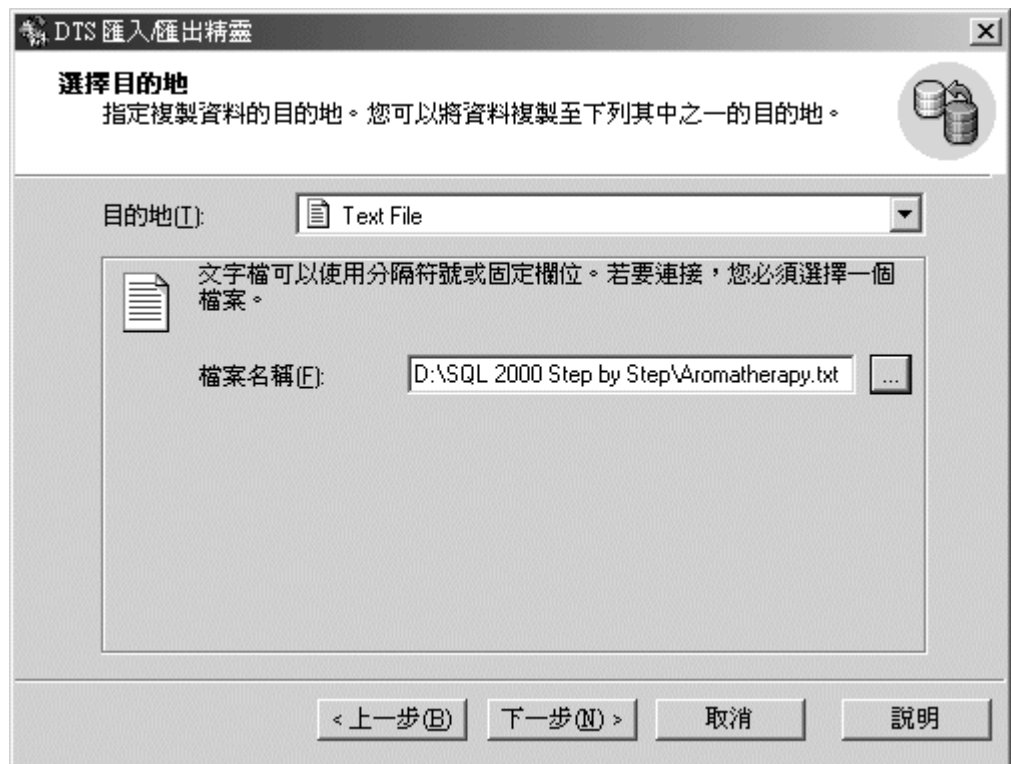


7. 移动至根目录之下的 **SQL 2000 Step by Step** 的目录中，并且输入
『Aromatherapy.txt』来当作是文件名称。



8. 按一下 [存盘](#) 按钮。

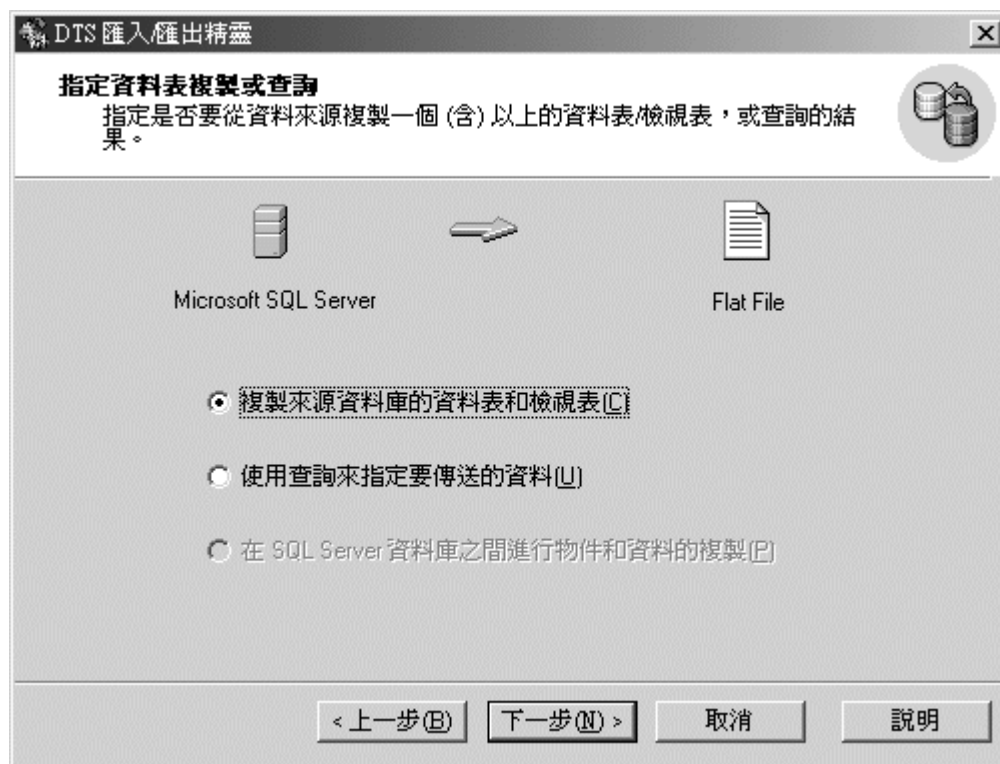
精灵会在 [文件名称](#) 对话框内，显示文本文件的位置及名称。



9. 按一下 [下一步](#) 按钮。

精灵会显示一个画面来询问您是要将资料表、检视表或查询结果进行汇出。默认

值是 [复制来源数据库的数据表和检视表](#)。



10. 按一下 **下一步** 按钮采用默认值。

此精灵会显示一个询问您目的地档案格式的画面。

DTS 匯入匯出精靈

選擇目的檔案格式
您必須指定檔案格式。選取檔案為分隔符號或固定欄位的格式。

來源(S): [Aromatherapy].[dbo].[Cautions]

目的檔案名稱: D:\SQL 2000 Step by Step\Aromatherapy.txt

☒ 使用分隔符號 - 使用任意字元來分隔資料行(D)
☐ 使用固定欄位 - 資料內容會以相同資料行寬度對齊(X)

檔案類型(T): ANSI ☐ 第一列有資料行名稱(I)
 資料列分隔符號(B): {CR}{LF}
 資料行分隔符號(C): 逗號
 文字定位項(E): 雙引號 (")

轉換(A)...

< 上一步(B) 下一步(N) > 取消 說明

- 在下拉式方块中选取 **PlantParts** 数据表，并且选取 **第一列有数据行名称** 复选框。

DTS 匯入匯出精靈

選擇目的檔案格式
您必須指定檔案格式。選取檔案為分隔符號或固定欄位的格式。

來源(S): [Aromatherapy].[dbo].[PlantParts]

目的檔案名稱: D:\SQL 2000 Step by Step\Aromatherapy.txt

☒ 使用分隔符號 - 使用任意字元來分隔資料行(D)
☐ 使用固定欄位 - 資料內容會以相同資料行寬度對齊(X)

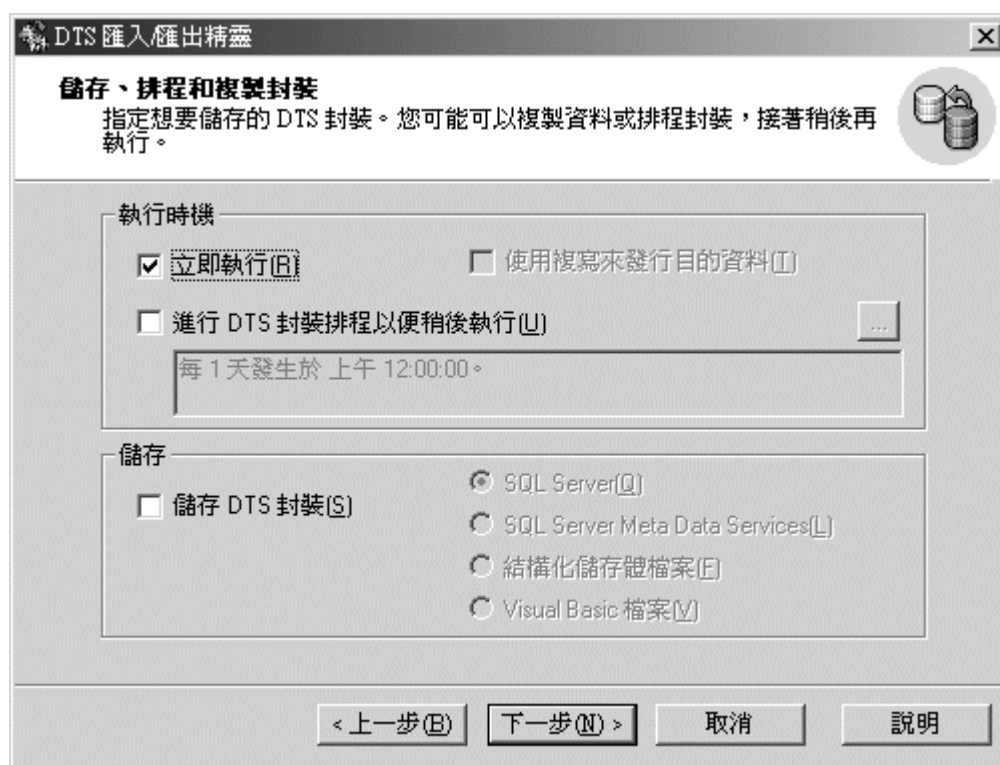
檔案類型(T): ANSI ☒ 第一列有資料行名稱(I)
 資料列分隔符號(B): {CR}{LF}
 資料行分隔符號(C): 逗號
 文字定位項(E): 雙引號 (")

轉換(A)...

< 上一步(B) 下一步(N) > 取消 說明

12. 按一下 **下一步** 按钮。

此精灵会要求您指定是要立即执行汇出数据，还是稍后再执行汇出数据的动作。



13. 我们将采用默认值 **立即执行**，然后按一下 **下一步** 按钮。

此精灵会显示一个您所设定内容的画面。

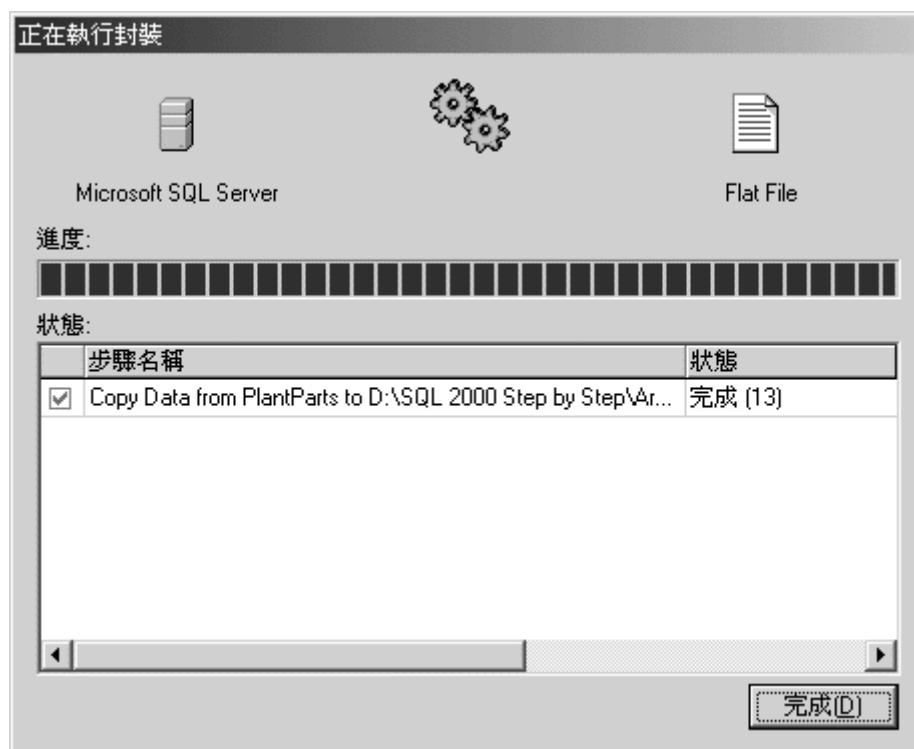


14. 按一下 **完成** 按钮。

当执行 DTS 封装时，此精灵会显示一个处理进度的对话框。当它处理完毕之后，
会显示一个此资料表已汇出成功的讯息。

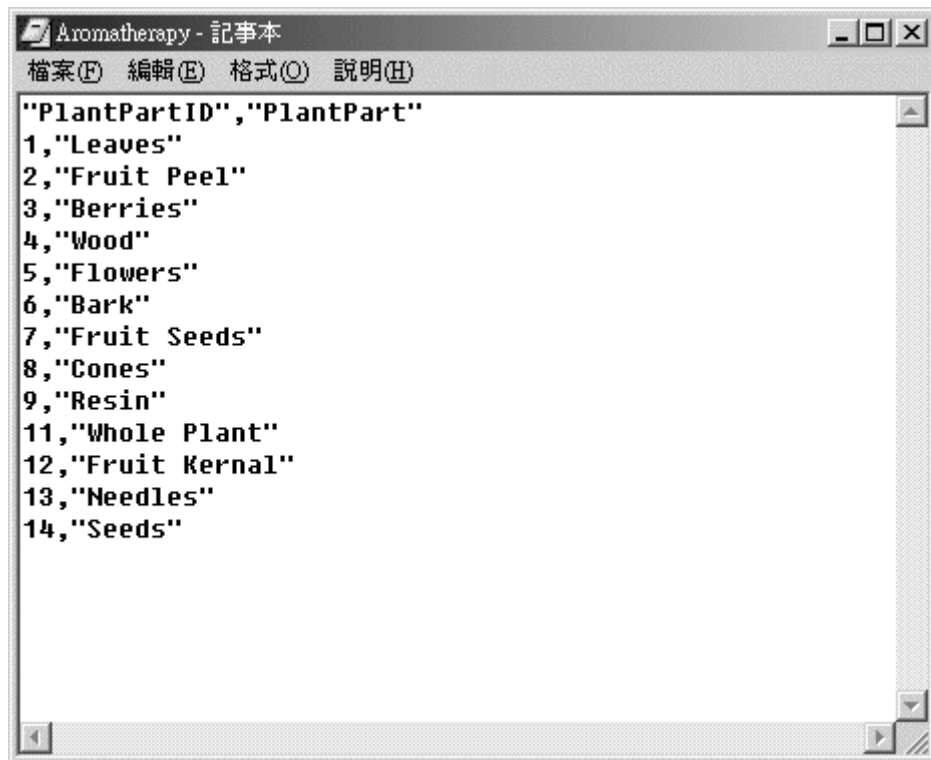


15. 按一下 **确定** 按钮，以便关闭此对话框。



16. 按一下 **完成** 按钮，以便关闭此精灵。

17. 如果您有 Microsoft Notepad，您可以开启 **Aromatherapy.txt** 以检查是否汇出成功。



附加及卸离数据库

SQL Server 使用二个或以上的实体档案来储存数据库。虽然这些用来储存数据库的档案只是普通的操作系统档案，但是您却无法轻易将这些档案任意移动，因为这些档案连结到某个 SQL Server 执行个体中的特定数据库。

然而，SQL Server 允许从服务器中卸离数据库。所谓的 **卸离**（detach）是指将数据库和其服务器之间的连结中断，但是并不会影响数据库和交易记录文件。如果某一个数据库已经从服务器卸离之后，您就可以移动或复制它，甚至是将它重新附加到相同或不同的服务器中。

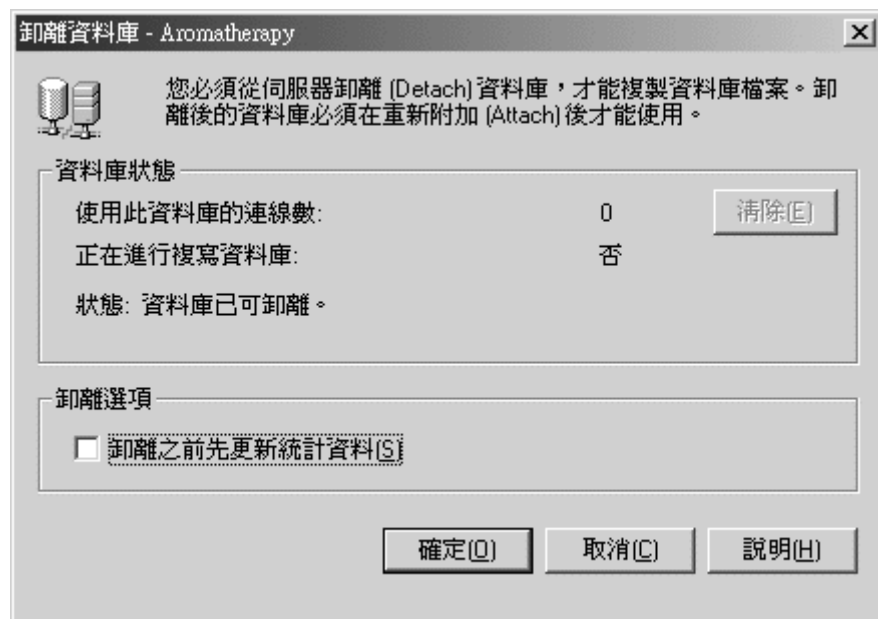
卸离数据库

当您执行从服务器卸离数据库时，只会呼叫一个对话框来简单的处理卸离数据库的程序。

卸离 Aromatherapy 数据库

1. 在主控台的树状目录中 Aromatherapy 数据库上按右钮，并且指向 [所有工作](#) 中的 [卸离数据库](#)。

SQL Server 会显示一个 [卸离数据库](#) 对话框。



2. 按一下 **确定** 按钮。

SQL Server 会显示一段数据库已经被卸离的讯息。



附加数据库

如果有一个数据库已经从服务器中卸离时，您可以移动或复制数据文件和交易记录档案（就像操作其它操作系统档案一样）。您可以将这些已经卸离的数据库重新附加到相同或不同的服务器中。

重新附加 **Aromatherapy** 数据库

1. 在主控台的树状目录服务器中的 **数据库** 数据夹上按右钮，并指向 **所有工作** 中的 **附加数据库**。

SQL Server 会显示一个 **附加数据库** 对话框。



2. 按一下 [浏览](#) 按钮。



浏览按钮

SQL Server 会显示一个 [浏览现有的档案](#) 对话框。



3. 移动至根目录之下的 **SQL 2000 Step by Step** 的目录中，并且选

取 **Aromatherapy.mdf** 数据库，然后按一下 **确定** 按钮。

SQL Server 会显示此数据库档案及交易记录档案的位置。



4. 按一下 **确定** 按钮。

SQL Server 会显示一段此数据库已经附加成功的讯息。



复制数据库精灵

[复制数据库精灵](#) 提供一种简单的方式以在 SQL Server 的执行个体之间进行复制或移动数据库。

您可以在不同的 SQL Server 2000 执行个体或 SQL 7.0 和 SQL Server 2000 之间的执行个体进行复制数据库。

使用复制数据库精灵

您可以在 [选择精灵](#) 的对话框中选取 [复制数据库精灵](#)，或者是在 [数据库](#) 和 [服务器](#) 数据夹的快捷菜单中选取 [所有工作](#) 项目即可看见 [复制数据库精灵](#) 项目。

复制一个数据库

提示

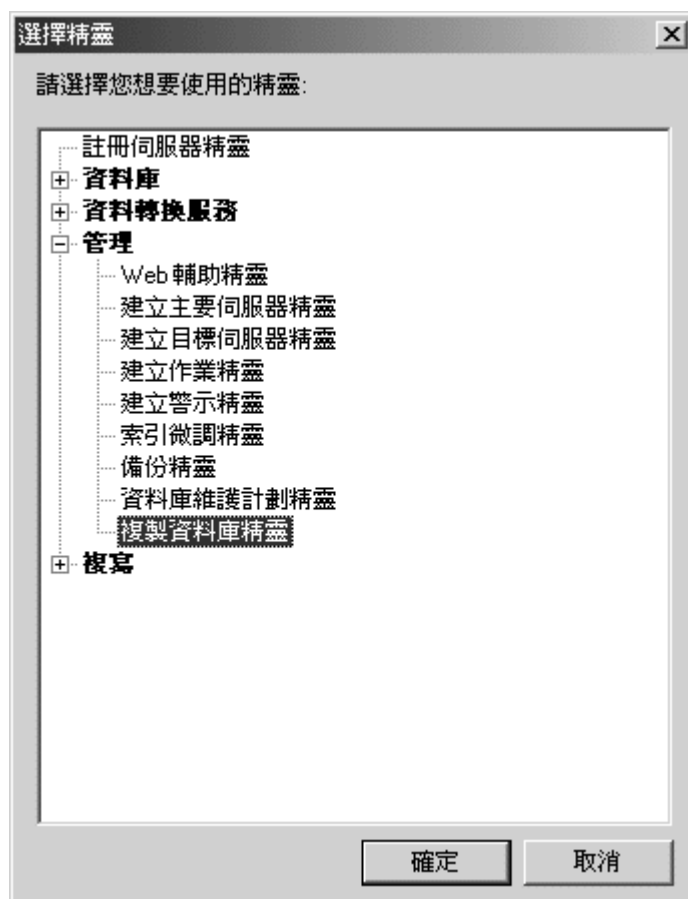
您必须要存取其它服务器以完成本练习。

-
1. 在主控台的树状目录中选取 **Aromatherapy** 数据库，并且在工具列上按一下 [执行精灵](#) 按钮。
-



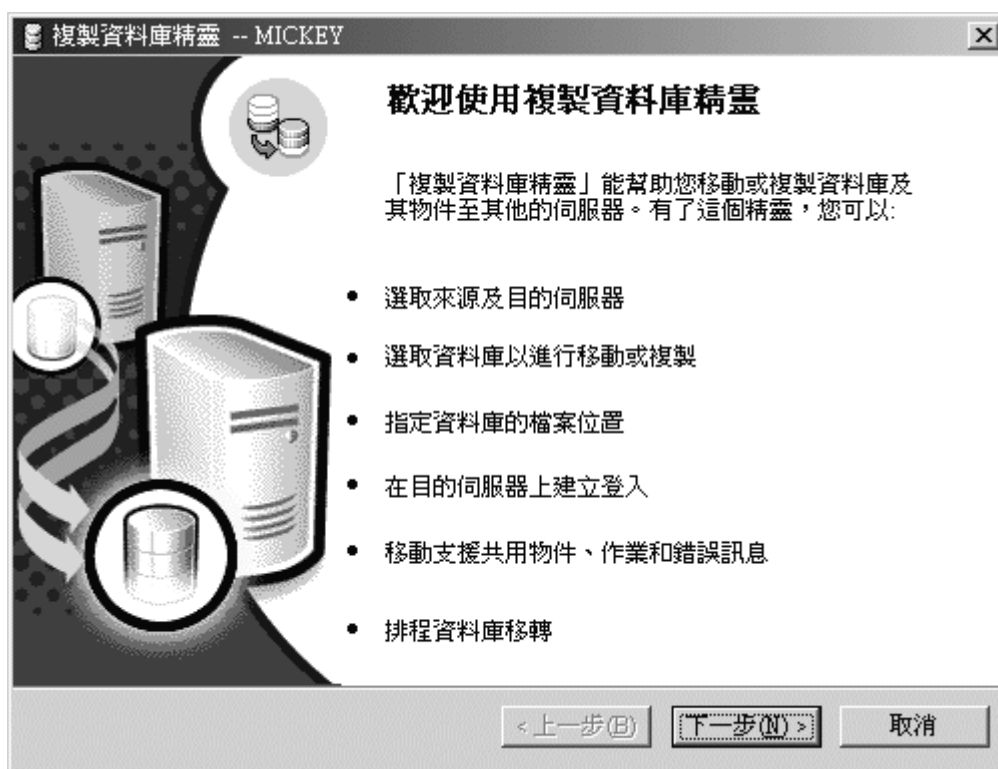
执行精灵按钮

此时 SQL Server 在屏幕上会显示一个 [选择精灵](#) 对话框。



2. 在 [管理](#) 数据夹中选取 [复制数据库精灵](#) 项目，然后按一下 [确定](#) 按钮。

此时 SQL Server 会显示 [复制数据库精灵](#) 的首页画面。



3. 按一下 [下一步](#) 按钮。

此时精灵会显示要求输入服务器的名称，此服务器中包含要进行复制的数据库。

複製資料庫精靈 -- KS_TR_MICKEY

選取來源伺服器
您想從哪一個來源搬移或複製資料庫？

要連線到 SQL Server 的執行個體，您必須指定來源伺服器、使用者名稱及密碼。

來源伺服器(S): 8N122587011

☐ 使用 Windows 的帳戶驗證(W)

☒ 使用 SQL Server 的帳戶驗證(S)

使用者名稱(U): SA

密碼(P):

< 上一步(B) 下一步(N) > 取消

4. 依您的需求改变验证的方式，然后按一下 [下一步](#) 按钮。

此时精灵会显示请求您输入目的地服务器名称的画面。



5. 从下拉式方块中选取您要复制的目的服务器名称，然后按一下 [下一步](#) 按钮。

此时精灵会要求您选取要进行复制或移动的数据库名称。

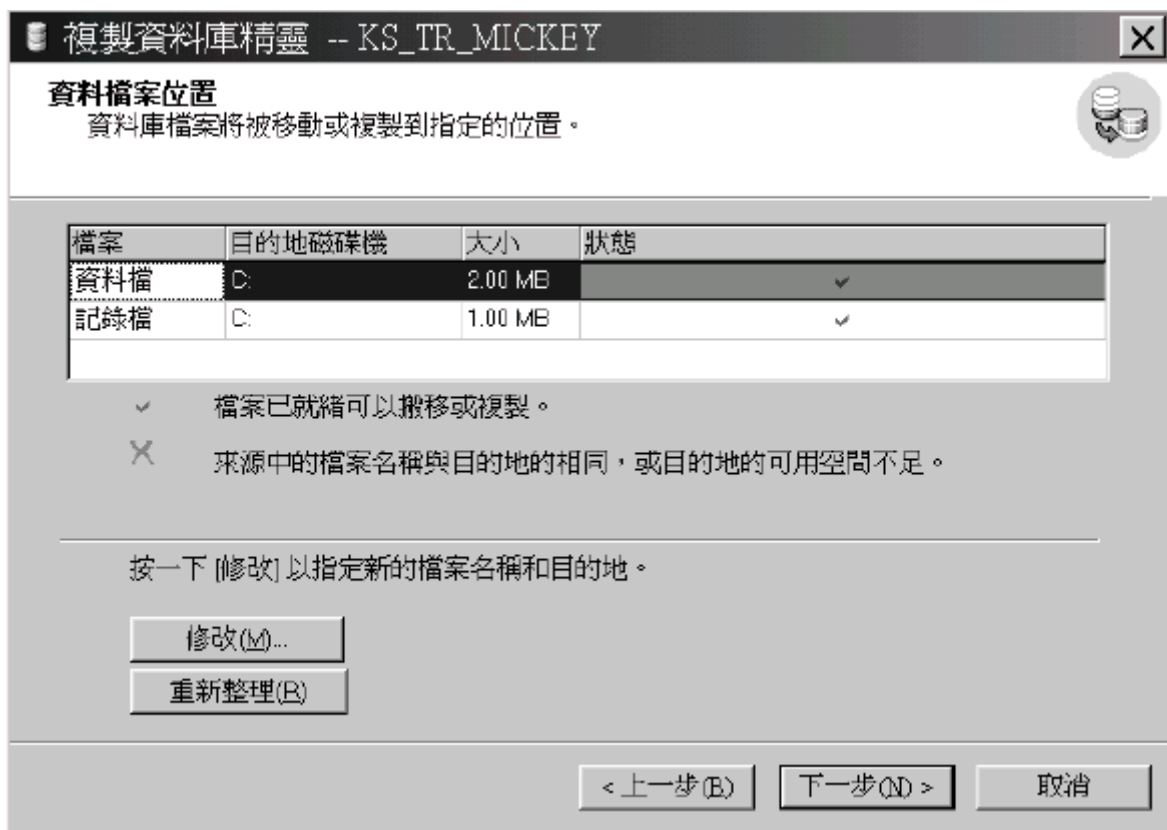


6. 选取 **Aromatherapy** 数据库中的 **复制** 复选框。



7. 按一下 [下一步](#) 按钮。

此精灵会显示一个目的地档案位置的画面。

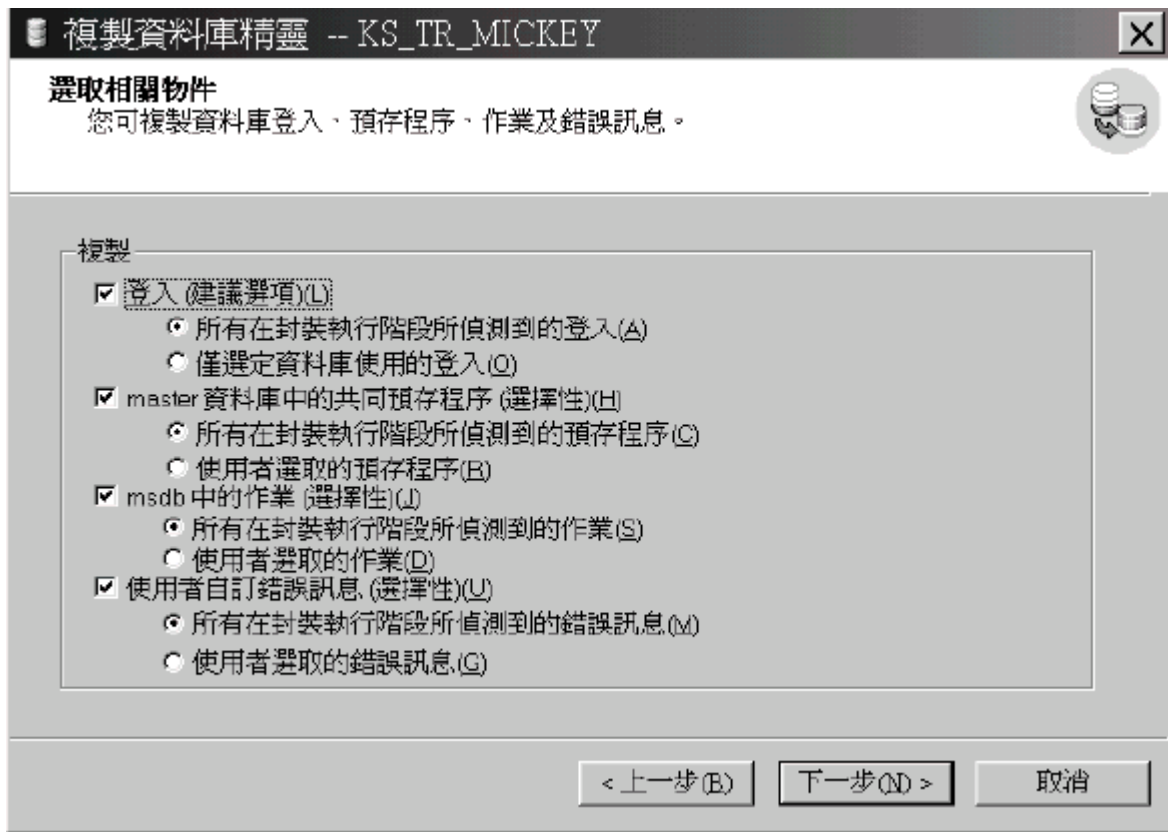


提示

如果在任何一个档案中的 **状态** 数据行中有打叉 **x** 符号时，则表示您可以按一下 **修改** 按钮以便修正所发生的问题。

-
- 按一下 **下一步** 按钮。

此精灵会显示一个询问您该数据库的哪些对象要与数据库一起进行复制的画面。



9. 我们采用默认值，然后按一下 **下一步** 按钮。

此时精灵会显示一个询问您是否要立即执行还是稍后再执行的画面。

複製資料庫精靈 -- KS_TR_MICKEY

排程 DTS 封裝

您可以立刻執行封裝或進行排程以便在日後執行。

封裝屬性

名稱(N):

CDW_8N122587011_KS_TR_MICKEY_0

執行時間

☒ 立即執行(B)

☐ 執行一次(O) 日期(D): 2001/ 3/27 時間(T): 下午 04:11:38

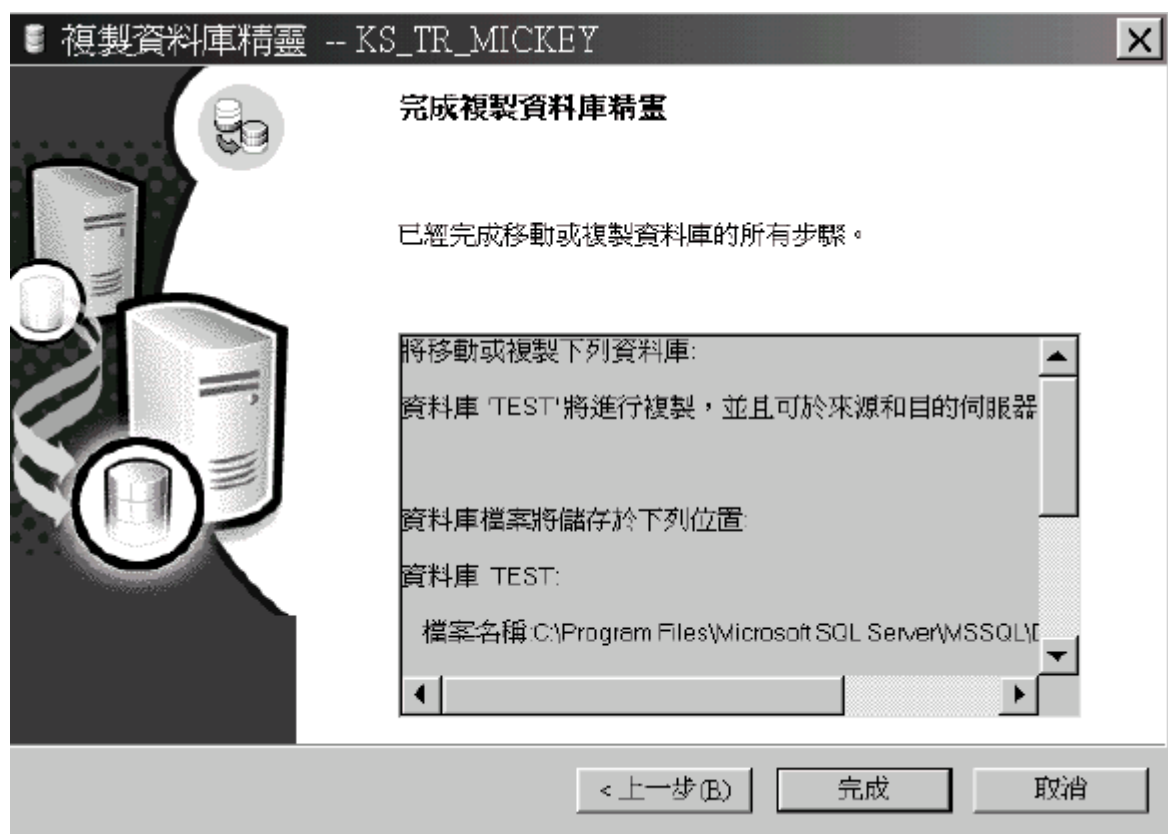
☐ 排程 DTS 封裝以便日後執行(W) ...

每 天發生於 上午 12:00:00。

< 上一步(B) 下一步(N) > 取消

10. 我们采用立即执行的预设选项值，然后按一下 [下一步](#) 按钮。

此时精灵会显示您的设定值。






11. 按一下 **完成** 按钮。

该精灵会显示一个正在处理的对话框，当处理完毕之后会显示一个讯息来告知您

执行复制数据库的工作已经成功。

本章总结

要执行的工作	执行	按钮
使用 DTS 汇入精	开启 选择精灵 对话框，在 数据转换服务 数据夹中选取 DTS 汇入精灵 ，	

灵来汇入数据	然后按一下 确定 按钮，并按照精灵所提出的步骤来进行数据汇入的工作。	
使用 DTS 汇出精灵 来汇出数据	开启 选择精灵 对话框，在 数据转换服务 数据夹中选取 DTS 汇出精灵 ，然后按一下 确定 按钮，并按照精灵所提出的步骤来进行数据汇出的工作。	
卸离数据库	在主控台树状目录下的数据库上按右钮，并且指向 所有工作 中的 卸离数据库 项目，并按照所提出的步骤来进行数据库卸离的工作。	
附加数据库	在主控台树状目录下的 数据库 数据夹上按右钮，并且指向 所有工作 中的 附加数据库 项目，并按照所提出的步骤来进行资料库附加的工作。	
使用 复制数据库精灵	开启 选择精灵 对话框，在 管理 数据夹中选取 复制数据库精灵 ，并按照精灵所提出的步骤来进行数据库复制的工作。	

第五篇 Transact-SQL

[21. Query Analyzer](#)

- . 认识 Query Analyzer
- . 使用查询窗口
- . 使用对象浏览器

[22. 数据定义语言](#)

- . 认识 DDL
- . 使用对象浏览器以设定数据定义
- . 本章总结

[23. 分析查询](#)

- . 使用 Query Analyzer 将执行效率最佳化
- . 客户端统计资料
- . 索引微调精灵
- . 本章总结

[24. Transact-SQL 语言的组件](#)

- . Transact-SQL 命令
- . Transact-SQL 运算符
- . Transact-SQL 函数

[25. 设计对象](#)

- . 暂存资料表
- . 变数
- . 本章总结

[26. 控制执行流程](#)

- . 条件式处理
- . 循环
- . 本章总结

[27. Transact-SQL 数据指针](#)

- . 认识数据指针
- . 使用数据指针
- . 监控 Transact-SQL 数据指针
- . 本章总结

[28. 预存程序](#)

- . 认识预存程序
- . 使用和建立预存程序
- . 本章总结

29. 触发程序

- . 认识触发程序
- . 建立触发程序
- . 本章总结

30. 使用者自订函数

- . 认识使用者自订函数
- . 建立使用者自订函数
- . 使用使用者自订函数
- . 本章总结

21. Query Analyzer

在本章中，您将学习到：

- 如何启动 Query Analyzer。
- 如何选取要工作的数据库。
- 如何在 Query Analyzer 中执行 Transact-SQL 陈述式。
- 如何建立 SQL Script。
- 如何从对象浏览器中开启对象。
- 如何从对象浏览器中新增对象至编辑窗格。
- 如何从对象浏览器将对象指令码化。

在前一章中，我们使用 Enterprise Manager 来建立和维护数据库对象，并且执行基本的操作。

在本章中，我们将重点放在 Microsoft SQL Server 的图形化工具—Query Analyzer。

认识 Query Analyzer

虽然我们可以在 Enterprise Manager 中建立及执行查询，以及其它的 Transact-SQL 陈述式，但是它本身是一个数据库管理工具，因此 Enterprise Manager 用来作为数据库管理工具比较适当，而 Query Analyzer 本身是一个主要用于指令码程序的工具。

Query Analyzer 是一套可以让您撰写和除错 Transact-SQL 陈述式的强大工具程序（我们将在本章中看到其中之一的功能—SQL 指令码），它也可以提供藉由执行计划以及我们将在第 23 章所要讨论的 [索引微调精灵](#)（Index Tuning Wizard）以分析查询效能。

启动 Query Analyzer

您可以从 Enterprise Manager 或 Windows 的 [开始](#) 菜单来启动 Query Analyzer。当您自 Enterprise Manager 中启动 Query Analyzer 时，Query Analyzer 会使用 Enterprise Manager 的连接信息来联机数据库：如果您连接至服务器时，Query Analyzer 将会连接到服务器中；如果您连接至数据库时，Query Analyzer 将会连接到数据库中。

提示

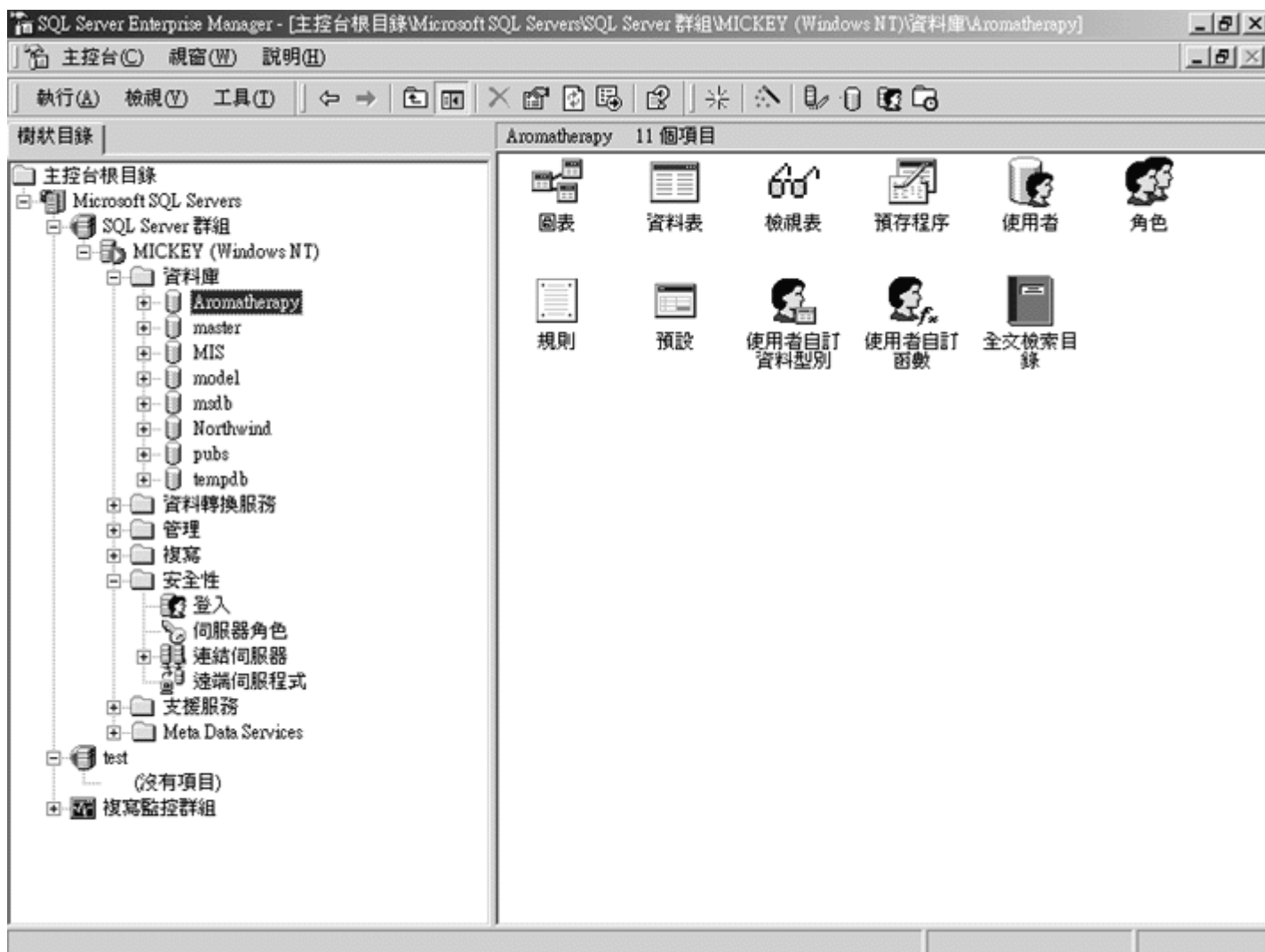
您可以使用第三种方法来启动 **Query Analyzer**，就是在命令提示列中输入『**isqlw**』命令即可启动 **Query Analyzer**。

假如您是从 [开始](#) 菜单中来启动 **Query Analyzer**，或者是在 **Enterprise Manager** 中并没有与服务
器或数据库进行连接时，您就必须在 **Query Analyzer** 中以手动的方式来连接。

从 **Enterprise Manager** 启动 **Query Analyzer**

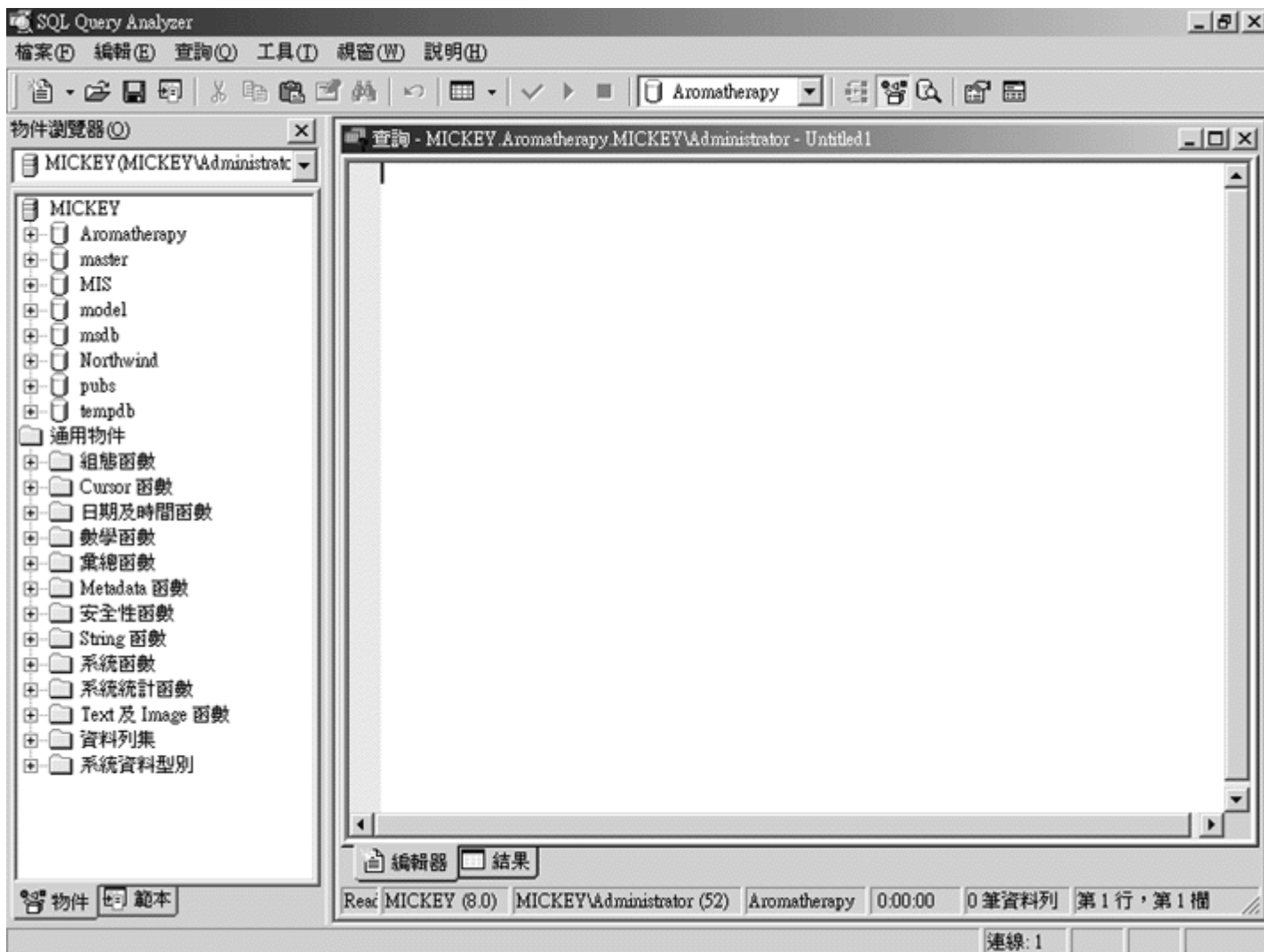
1. 在 **Enterprise Manager** 的主控台的树状目录之下，选择 **Aromatherapy** 数据库。

SQL Server 会在详细资料窗格中显示数据库对象的明细。



2. 从 [工具](#) 菜单中选取 [SQL Query Analyzer](#) 。

SQL Server 会开启 Query Analyzer，并且自动与服务器中的 Aromatherapy 数据库进行连接。



说明

如果您在 Query Analyzer 中没有看到对象浏览器时，您可以按下 [F8] 键以便将它显示在对象浏览器上。

3. 关闭 Query Analyzer。

从开始菜单中启动 Query Analyzer

1. 在工具列中按一下 **开始** 按钮。
2. 指向 **程序集** 项目中的 **Microsoft SQL Server** 。

此时在 Microsoft SQL Server 项目中会出现一些图标。

3. 按一下 **Query Analyzer** 图示。



Query Analyzer 按钮

此时 Query Analyzer 会显示连接到 SQL Server 的对话框。



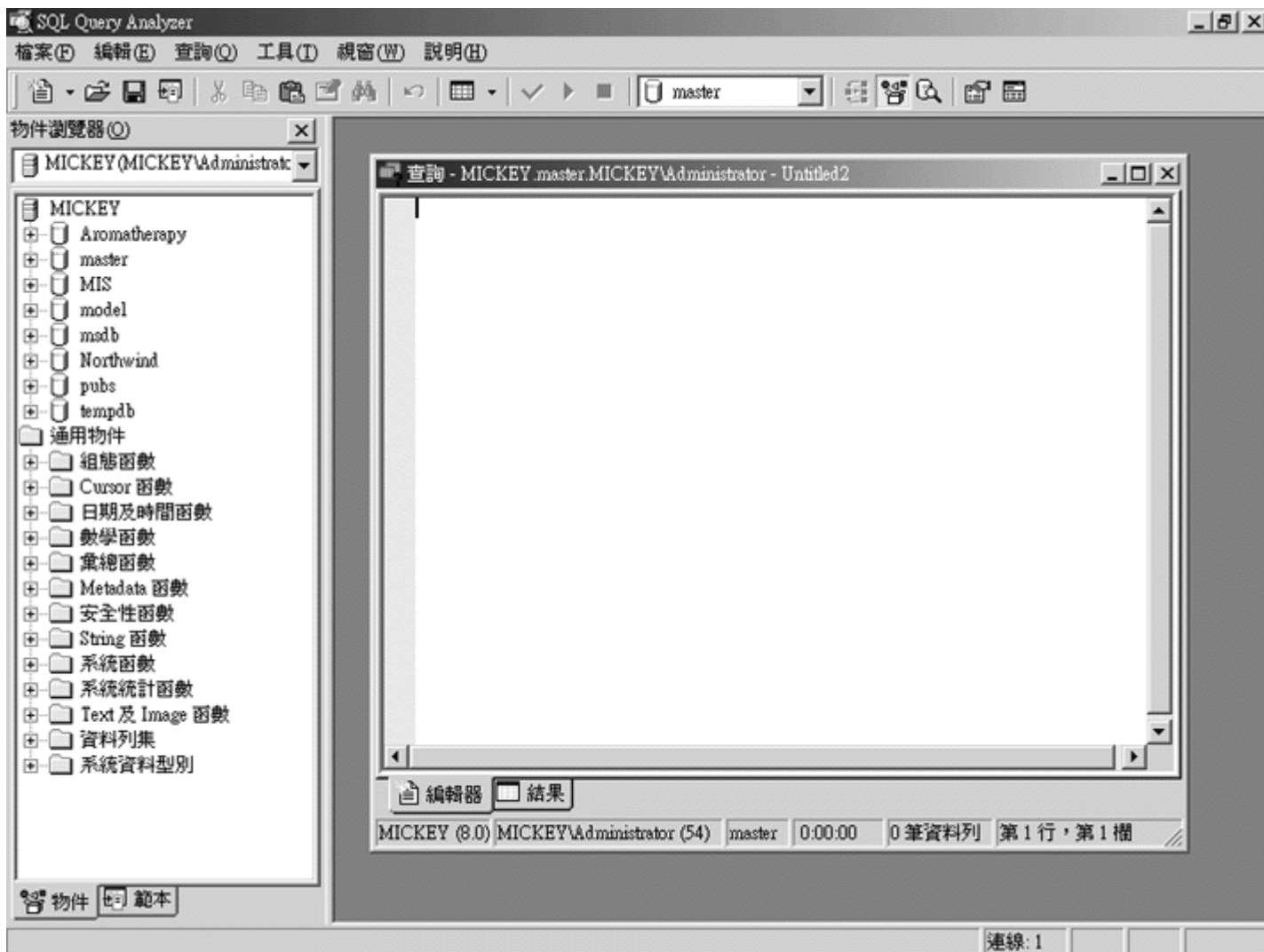
4. 请确认正确的 SQL Server 和 **Windows 的账户验证** 项目已经选取，然后再按一下 **确定** 按钮。

此时 Query Analyzer 会与 SQL Server 进行连接，并且会开启 Query Analyzer。

Query Analyzer 会在登入时与您所预设的数据库进行连接。

提示

如果您的 SQL Server 并没有自动启动时，您可以选取 **启动已停止的 SQL Server** 复选框，以便自动启动 SQL Server。



选取数据库

Query Analyzer 会使用目前所选取的数据库来作为查询或供其它的 Transact-SQL 陈述式使用。

工具列如图 21-1 所示，包含了一个下拉式方块以显示目前您所选取的数据库，您可以从该工具

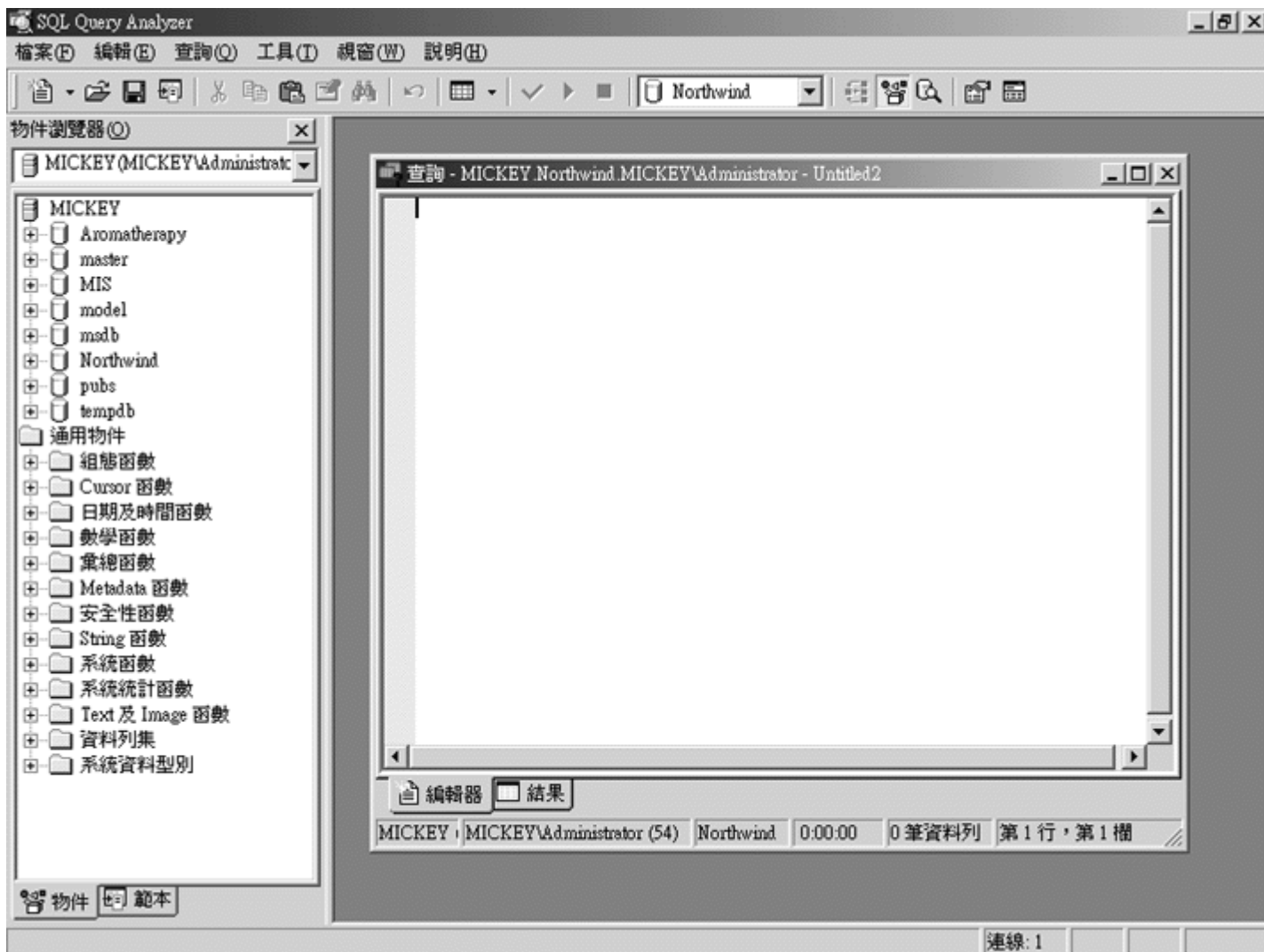
列或 [查询](#) 菜单来选取其它的数据库。



图 21-1 Query Analyzer 工具列会显示目前所选取的数据库。

使用工具列来选取数据库

1. 在工具列的下拉式方块中选取 **Northwind**。



自查询菜单中选取数据库

1. 自查询菜单中选取 [变更数据库](#) 项目。

此时 Query Analyzer 会显示一个 [选取数据库](#) 的对话框。



提示

您也可以按一下 **CTRL-U** 快速键以开启 [选取数据库](#) 对话框。

2. 在 Aromatherapy 数据列的任意位置按一下，以便选取 Aromatherapy 数据库。



3. 按一下 [確定](#) 按钮。

此时 Query Analyzer 会选取 Aromatherapy 数据库。

使用查询窗口

当 Query Analyzer 一启动时会开启二个窗口，一是 [对象浏览器](#) 窗口，另一个则是 [查询](#) 窗口。

当您第一次开启 Query Analyzer 时，则只会开启一个查询窗口，但是您可以于任何时间在 Query

Analyzer 的工具列上按一下 [新增查询](#) 按钮来开启新的窗口。



新增查询按钮

在查询窗口中会显示数据库服务器的名称、目前所选取的数据库、目前登入服务器的使用者以及该查询名称的标题列。**Query Analyzer** 与 **Enterprise Manager** 的查询设计师非常类似，但功能却比查询设计师强大许多。

身为一个程序代码的撰写者，您一定会觉得查询窗口比起查询设计师的 **SQL** 窗格更具弹性，并且具有强大的功能。查询设计师有一些限制，就是它只能处理一些 **SQL** 陈述式，而查询窗口却可以处理任何可以使用的 **Transact-SQL** 陈述式，并且您可以在单一的批次作业中，输入多行的 **Transact-SQL** 陈述式来处理某些功能。虽然查询窗口无法像查询设计师可以提供方格及图表窗格来检视，但是它却会提供我们将在第 23 章中会介绍的查询分析，来额外让您检视查询执行的效率分析。

输入 Transact-SQL 陈述式

使用查询窗口最简单的方式就是输入如同在查询设计师中 **SQL** 窗格内的 **SQL** 陈述式。但是并不像查询设计师一样，**Query Analyzer** 的查询窗口会将您所输入的 **Transact-SQL** 陈述式以颜色区别其用途。在表 21-1 中列出查询窗口所用的颜色代表的意思。

颜色	所代表的意义
蓝色	关键词

深绿色	批注
深红色	预存程序
灰色	运算符
绿色	系统数据表
紫红色	系统函数
红色	字符串

表 21-1 在 Query Analyzer 的查询窗口所用颜色代表的意义

执行一个 **SELECT** 查询

1. 在 **查询** 窗口中输入如下所示的 **SELECT** 陈述式：
2.

```
SELECT OilID, OilName, LEFT(LatinName, 10)
FROM Oils
```
3. 此时 **查询** 窗口会将您所输入的 **SELECT** 陈述式的文字内容颜色改变。



4. 自 Query Analyzer 工具列中按一下 [執行查詢](#) 按鈕。



執行查詢按鈕

Query Analyzer 会在 [查询](#) 窗口中增加一个包含二个标签页的窗格：一是包含查询结果的 [方格](#) 标签页，另一个则是 [讯息](#) 卷标页。



5. 选取 [讯息](#) 卷标页。

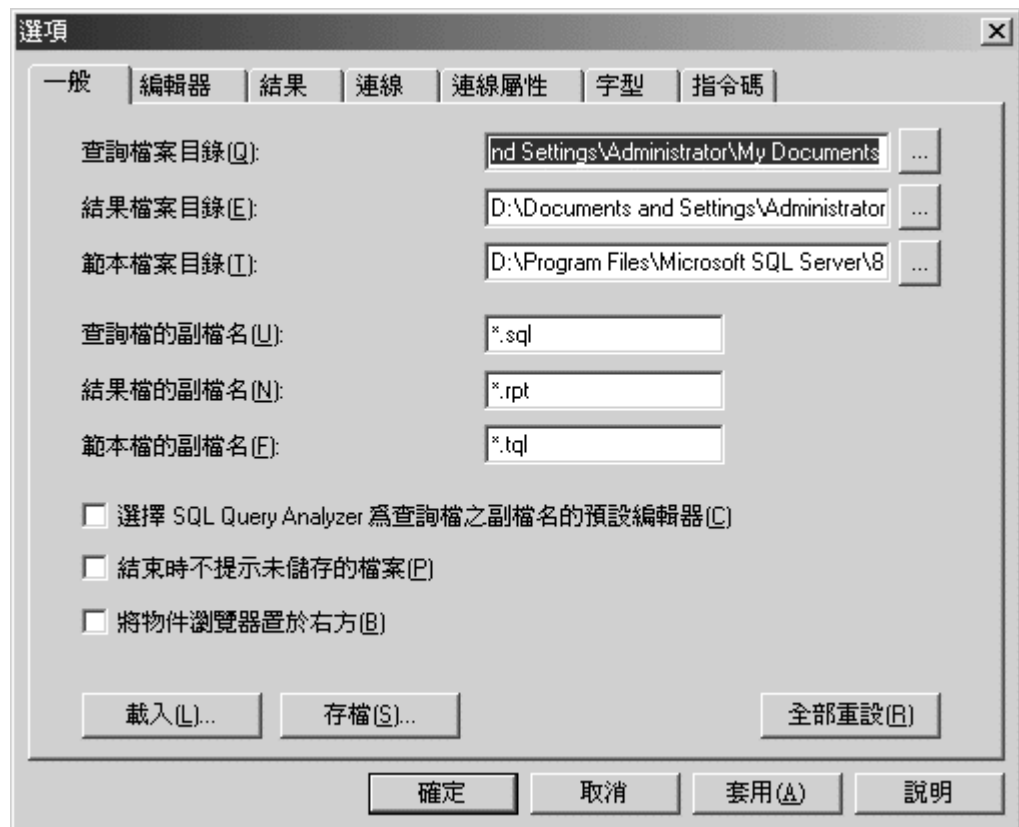
此时 [查询](#) 窗口显示此查询所产生的结果讯息。



在另一个标签页中显示查询的结果

1. 自 Query Analyzer 的 [工具](#) 菜单中选取 [选项](#) 。

此时 Query Analyzer 会显示 [选项](#) 对话框。



2. 选取 **編輯器** 标签页。

選項

一般 編輯器 結果 連線 連線屬性 字型 指令碼

復原緩衝區數目(N): 20

最大復原緩衝區大小 (單位: 行)(M): 1000

復原緩衝區限制處理(U): 顯示訊息方塊

Tab 鍵大小 (單位: 空格)(T): 8

☐ 儲存定位點為空白(S)

☐ 取消在編輯器內的拖曳文字(D)

預設 (非 Unicode) 檔案開啓格式: ☒ ANSI(I) ☐ OEM(E)

☐ 定位模式 (相對於分隔模式)(B)

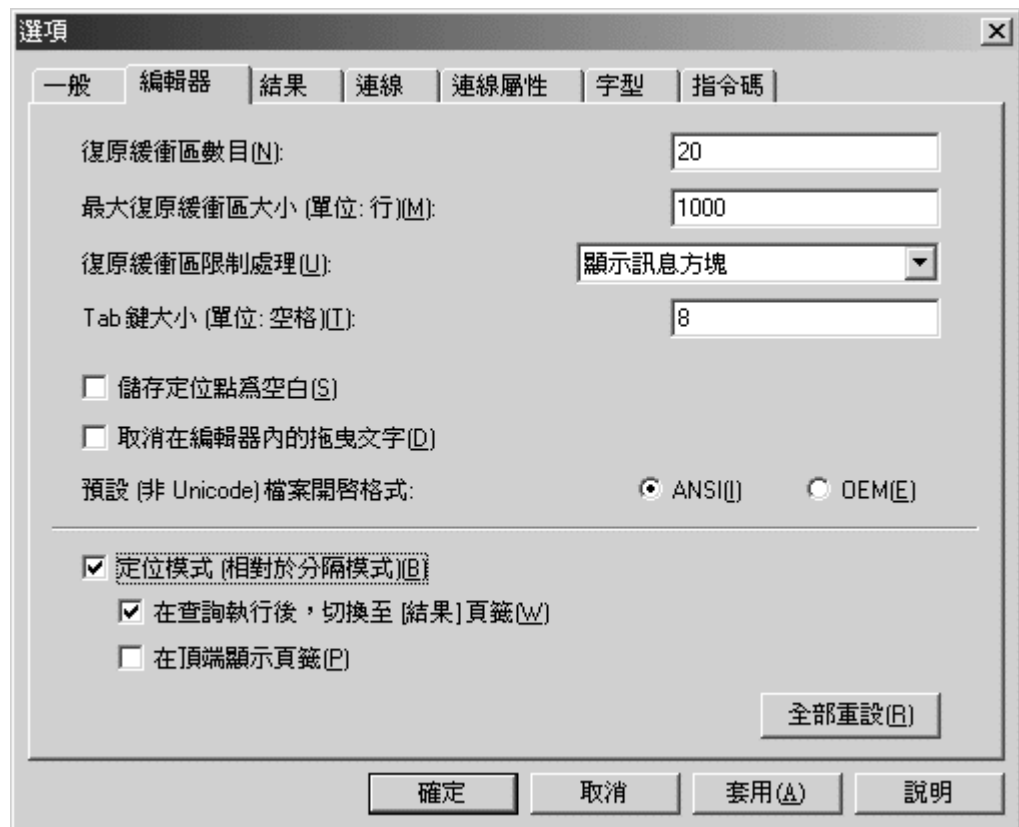
☒ 在查詢執行後，切換至 [結果] 頁籤(W)

☐ 在頂端顯示頁籤(B)

全部重設(R)

確定 取消 套用(A) 說明

3. 选取 **定位模式 (相對於分隔模式)** 的复选框。



4. 按一下 **確定** 按钮。

此时 Query Analyzer 会改变 **查询** 窗口的显示型态。



使用 SQL 指令码

指令码（script）是 Transact-SQL 陈述式的集合，并且储存于一个档案中。所谓的指令码大多数是记录某些常常执行固定用途的指令，以便建立数据库对象。其实您也可以将指令码的用途当作是在 DOS 模式下的批次文件的功能。然而这些指令码是储存在文本文件中，且您可以将它用在不同的服务器中重新建立数据库（事实上，SQL Server 就是用指令码来建立 pubs 和 Northwind 范例数据库）。

虽然指令码大多数是用来建立数据库对象，但是其用途却不只如此。任何一个有效的

Transact-SQL 陈述式都可以包含在一个指令码中。

一般来说，在一个指令码中的 **SQL** 陈述式都是以批次型态分组。一个指令码可以包含一个或多个批次操作，而每一个批次操作可以包含一个或多个 **SQL** 陈述式。在一个指令码中虽然可以包含一个或多个批次操作，但是这些批次操作需以 **GO** 命令来区隔。假如一个指令码中没有包含 **GO** 命令时，则在指令码中所有的陈述式将会被当作是单一的批次操作来执行。

建立指令码

1. 在 **查询** 窗口中建立下列 **SQL** 陈述式：

```
2.      SELECT OilID, OilName, LEFT(LatinName, 10)
3.      FROM Oils
4.      GO
5.      SELECT PlantPartID, PlantPart
        FROM PlantParts
```

提示

您只要在前一个练习的最后一行中，加入上述指令码的最后 **3** 行即可。

6. 在 **Query Analyzer** 工具列中按一下 **执行查询** 按钮。

此时 Query Analyzer 会在 [查询](#) 窗口的 [方格](#) 卷标页内，以二个窗格来显示其查询的结果。



7. 在查询窗口中重新选取 [编辑器](#) 标签页，并且在 Query Analyzer 工具列上按一下 [储存查询/结果](#) 按钮。



儲存查詢/結果按鈕

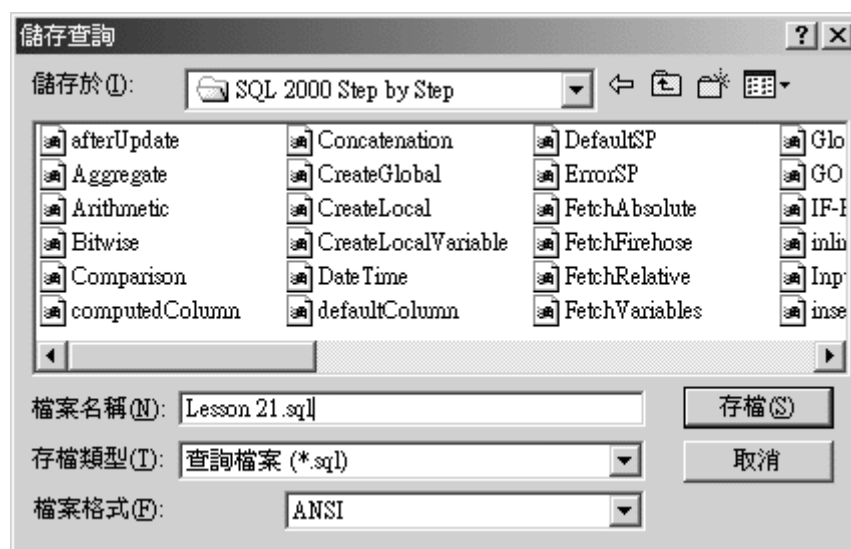
此時 Query Analyzer 會顯示 [儲存查詢](#) 的對話框。



重要

假如您沒有選取 [編輯器](#) 標籤頁之前就按一下 [儲存查詢/結果](#) 按鈕時，Query Analyzer 會儲存查詢結果，而不是查詢本身。

8. 将目前在对话框内的目录移到 **SQL 2000 Step by Step** 的数据夹中，并且将此查询的文件名称命名为 **Lesson 21**。



9. 按一下 **存盘** 按钮。

Query Analyzer 会将此查询内容储存为新的指令码档案。

开启一个指令码档案

1. 在 Query Analyzer 工具列上按一下 **新增查询** 按钮。



新增查询按钮

此 Query Analyzer 会开启一个新的 [查询](#) 窗口，并且其内容是空白的。



2. 在 Query Analyzer 的工具列上按一下 [加载 SQL 指令码](#) 按钮。
-

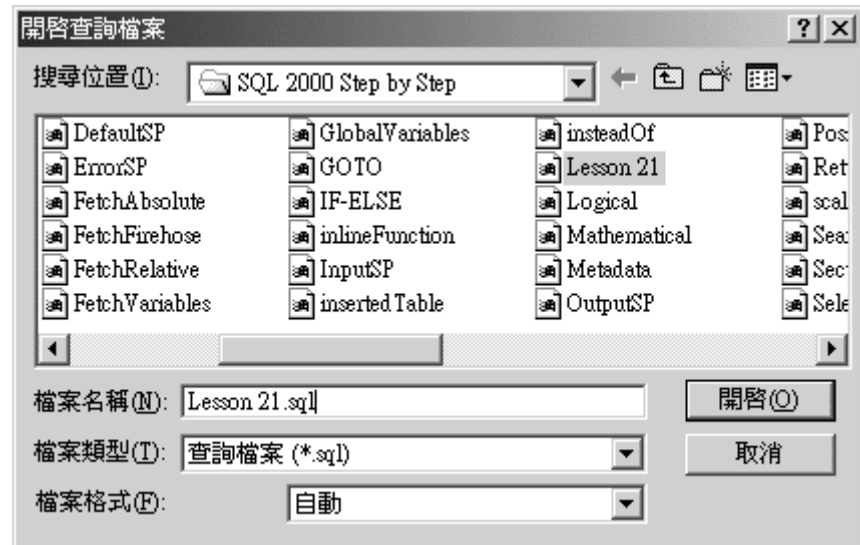


加载 SQL 指令码按钮

此 Query Analyzer 会显示 [开启查询档案](#) 对话框。



- 将目前在对话框内的目录移到 [SQL 2000 Step by Step](#) 的数据夹中，并且选取文件名称为的 [Lesson21](#) 的指令码档案。



4. 按一下 [開啟](#) 按钮。

Query Analyzer 会在 [查詢](#) 窗口内显示指令码的内容。



5. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕以便執行該指令碼。



執行查詢按鈕

Query Analyzer 會在 [方格](#) 標籤頁中顯示其查詢結果。

查詢 - MICKEY.Aromatherapy.MICKEY\Administrator - D:\SQL 2000 Step ...

	OilID	OilName	(沒有資料行名稱)
1	1	Basil	Ocimum Bas
2	2	Bergamot	Citrus Ber
3	3	Black Pepper	Piper Nigr
4	4	Cedarwood	Cedrus Atl
5	5	German Chamomile	Matricaria
6	6	Roman Chamomile	Anthemis N
7	7	Cinnamon	Cinnamomum
8	8	Citronella	Cymbopogon
9	9	Clary Sage	Salvia Scl

	PlantPartID	PlantPart
1	1	Leaves
2	2	Fruit Peel
3	3	Berries
4	4	Wood
5	5	Flowers
6	6	Bark
7	7	Fruit Seeds

編輯器 方格 訊息

MICKEY\Adn Aromatherapy 0:00:00 Grid #1: 49 筆資料列 第 1 行，第 1 欄

6. 关闭该指令码窗口。

使用对象浏览器

对象浏览器（Object Browser）包含在 Query Analyzer 窗口左边的窗格中，就像 Enterprise Manager 的主控制台树状目录中提供的，用来显示 SQL Server 中对象的阶层式浏览窗口一样。**对象浏览器** 中的 **对象** 卷标页是用来显示您已经与服务器连接的数据库对象明细以及用来建立 Transact-SQL 程序的一般对象。

提示

对象浏览器 中的 模板 卷标页包含了一些可用的程序化模板的阶层检视。我们将在下一章中讨论这些模板。

对象浏览器中的对象卷标页中所显示的对象其实是和 Enterprise Manager 的主控台树状目录中的对象内容大同小异的，但是对象浏览器的对象卷标页只会显示数据库对象，其它像是 登入 (Logins)和 资料转换服务 (DTS)是不会显示的。数据库中的数据表又分二种数据夹： 使用者数据表 (User Table) 和 系统数据表 (System Table)。

另外，与 Enterprise Manager 的主控台树状目录内所显示的数据表类似，对象浏览器会显示那些您在资料表中所定义的数据行、索引、条件约束以及触发程序。除此之外，检视表也会显示在对象浏览器中。图 21-2 是表示在对象浏览器内针对 Oils 资料表所显示的内容。

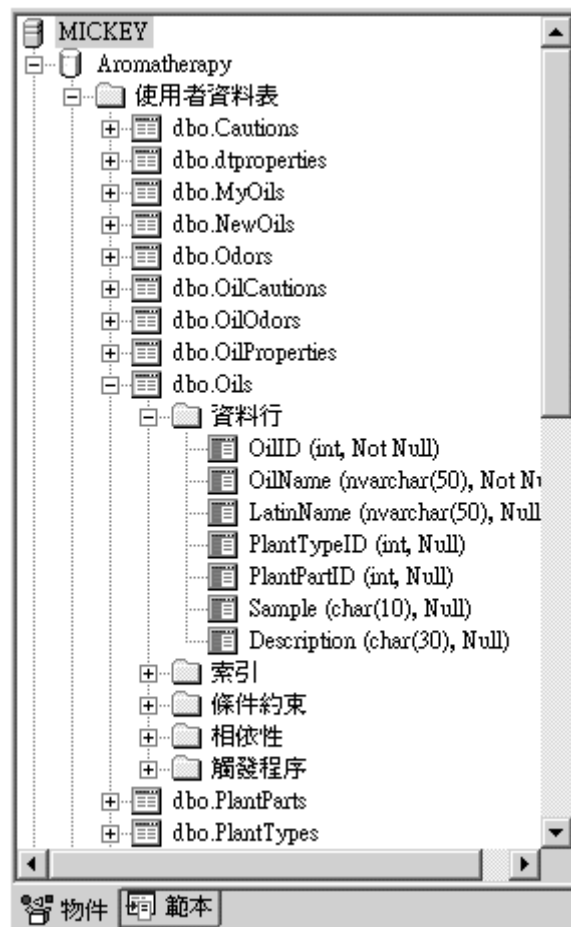


图 21-2 对象浏览器会显示您所建立数据库中的一些数据。

在对象浏览器中 **通用对象** 数据夹中所包含的是一些内建数据类型以及各种不同类别的

Transact-SQL 函数。每一种函数的 **参数** 资料夹内包含了每一个参数的说明，其中更包含了其名

称以及资料型别，让您可以更方便的使用。图 21-3 显示了对象浏览器窗口内的 LEFT 函数以及

参数。

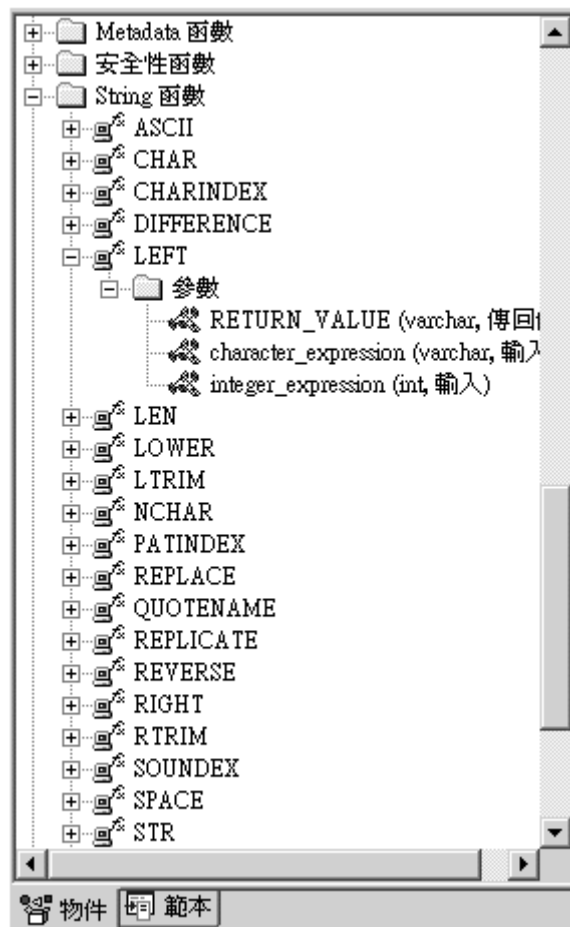


图 21-3 对象浏览器显示了在 通用对象 数据夹中的各种函数类别。

当您可以使用 [对象浏览器](#) 来开启资料表或检视表,就像是您在 Enterprise Manager 藉由开启 [查询设计师](#) 来显示资料列一样,当然您也可以检视这些资料列或者是插入新的数据列以及编辑已经存在的数据列内容。

您也可以在对象浏览器使用您已经建立好的 Transact-SQL 程序。您可以藉由拖曳的方式从对象浏览器拖曳一个对象至查询窗口中,此时 Query Analyzer 会替您自动产生指令码。

开启对象

当您在 [对象浏览器](#) 内的数据表或检视表上按右键，并且选取 [开启旧档](#) 之后，Query Analyzer

会在 [开启数据表](#) 窗口中显示该数据表中的所有数据列。

开启一个数据表

1. 在 [对象浏览器](#) 中，将 Aromatherapy 数据库的 [使用者数据表](#) 数据夹展开。

Query Analyzer 会显示在此数据库内的使用者数据表明细。



2. 在 **dbo.PlantParts** 上按右鈕，並且选取 **开启旧档**。

Query Analyzer 会在 **开启数据表** 窗口中显示此数据表内所有的数据列。

開啟資料表 - MICKEY.Aromatherapy.dbo.PlantParts		
	PlantPartID	PlantPart
1	1	Leaves
2	2	Fruit Peel
3	3	Berries
4	4	Wood
5	5	Flowers
6	6	Bark
7	7	Fruit Seeds
8	8	Cones
9	9	Resin
10	11	Whole Plant
11	12	Fruit Kernal
12	13	Needles
13	14	Seeds
*		

MICKEY\Administrat Aromatherapy 13 筆資料列 第 1 行，第 1 欄

3. 当您在结束浏览数据列时，请关闭 [开启数据表](#) 窗口。

开启一个检视表

1. 在 [对象浏览器](#) 中，将 Aromatherapy 数据库的 [检视表](#) 数据夹展开。

此时 Query Analyzer 会显示在此数据库内所有的检视表明细。



2. 在 **dbo.OilCautionsExtended** 项目按右钮，并选取 **开启旧档**。

此时 Query Analyzer 会在 **开启数据表** 窗口中显示藉由查询所传回的资料列。

開啟資料表 - MICKEY.Aromatherapy.dbo.OilCautionsExtended(...)				
	Caution	Description	OilName	OilID
1	Sensitive Skin	(...)	Basil	1
2	Increases Photosensitivity	(...)	Bergamot	2
3	Kidneys	(...)	Black Pepper	3
4	Pregnancy	(...)	Cinnamon	7
5	Skin Irritant	(...)	Cinnamon	7
6	Pregnancy	(...)	Clary Sage	9
7	Alcohol	(...)	Clary Sage	9
8	Pregnancy	(...)	Geranium	14
9	Inflamed & reddened skin	(...)	Geranium	14
10	Pregnancy	(...)	Jasmine	17
11	Pregnancy	(...)	Juniper	18
12	Skin Irritant	(...)	Lemon	20
13	Skin Irritant	(...)	Lemongrass	21
14	Pregnancy	(...)	Marjoram	23
15	Skin Irritant	(...)	Lemon Balm (Melissa)	24
16	Mental Disturbances	(...)	Nutmeg	27
17	Increases Photosensitivity	(...)	Orange	28
18	Pregnancy	(...)	Peppermint	31
19	Homeopathic	(...)	Peppermint	31
20	Pregnancy	(...)	Rose	34

MICKEY\Administrator (52) Aromatherapy 25 筆資料列 第 1 行，第 1 欄

- 当您结束浏览数据列时，请关闭 [开启数据表](#) 窗口。

加入对象至编辑器窗格中

对象浏览器提供一项既简单又便利的功能，那就是 [拖曳](#)（drag-and-drop）功能，您只要在对象浏览器中指定您需要的对象，然后将此对象拖曳至 [查询](#) 窗口内的 [编辑器](#) 窗格中，此时在 [编辑器](#) 窗格即会贴上该对象的名称。

提示

如果您在 [通用对象](#) 数据夹中拖曳函数时，对象浏览器会贴上函数名称，但是不会贴上参数。要贴上函数的完整语法，您可以使用下一节将要说明的 [指令码化](#)（Scripting）。

加入一个数据库对象

1. 选取 [查询](#) 窗口，并确认选取 [编辑器](#) 标签页，然后在 Query Analyzer 的工具列上按一下 [清除窗口](#) 按钮。



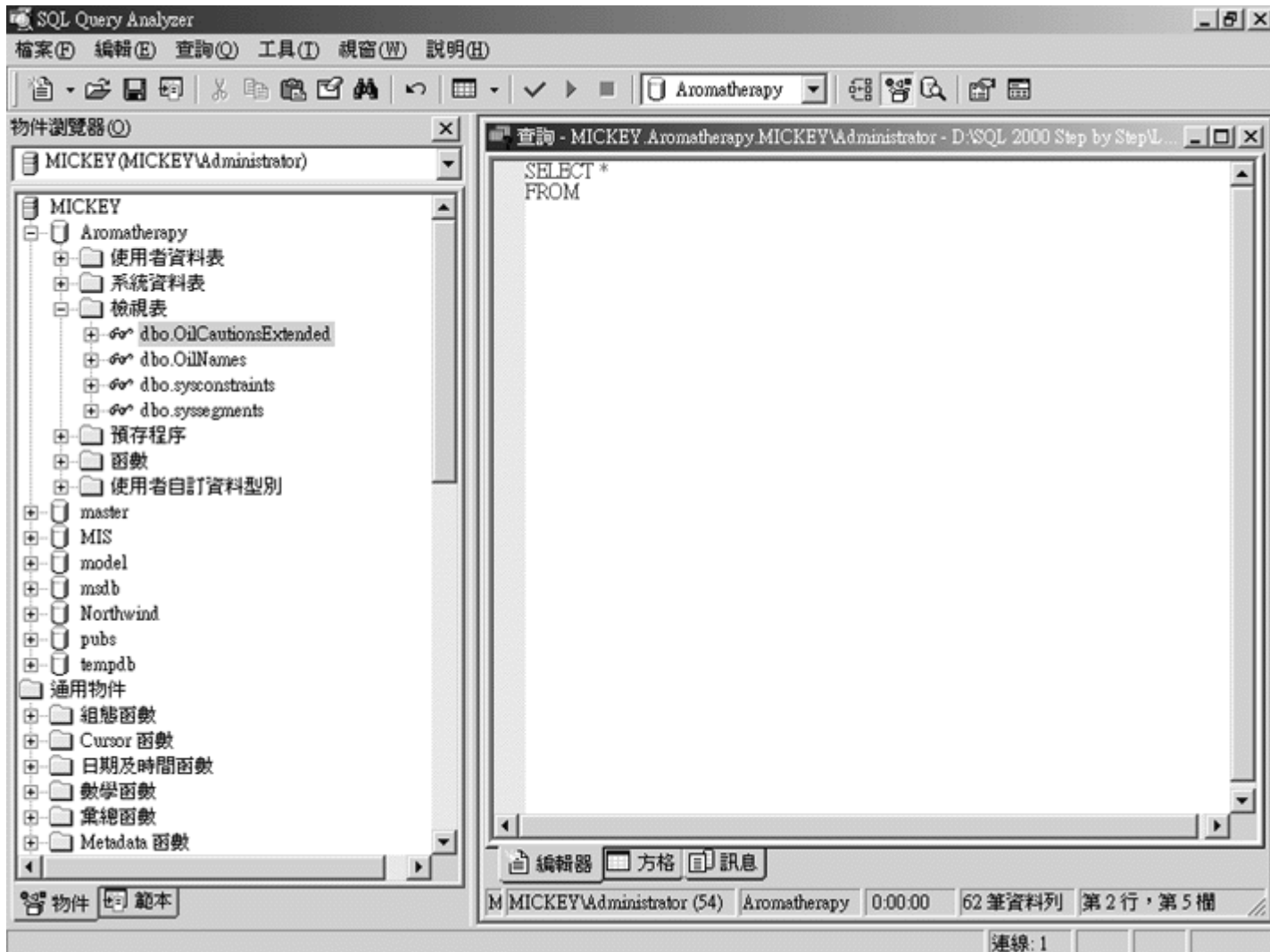
清除窗口按钮

Query Analyzer 会将 [编辑器](#) 窗格内的内容全部清除。

2. 在 [编辑器](#) 窗格内输入如下列所示的 SQL 陈述式（必须要在 FROM 后面加入一个空格）：

```
3.      SELECT *  
      FROM
```

4. 在 **对象浏览器** 中展开 **使用者数据表** 数据夹。



5. 拖曳 **对象浏览器** 中的 **dbo.Properties** 数据表至 **编辑器** 窗格内 FROM 关键词的后面。

Query Analyzer 会在 FROM 后面贴上此数据表的名称。

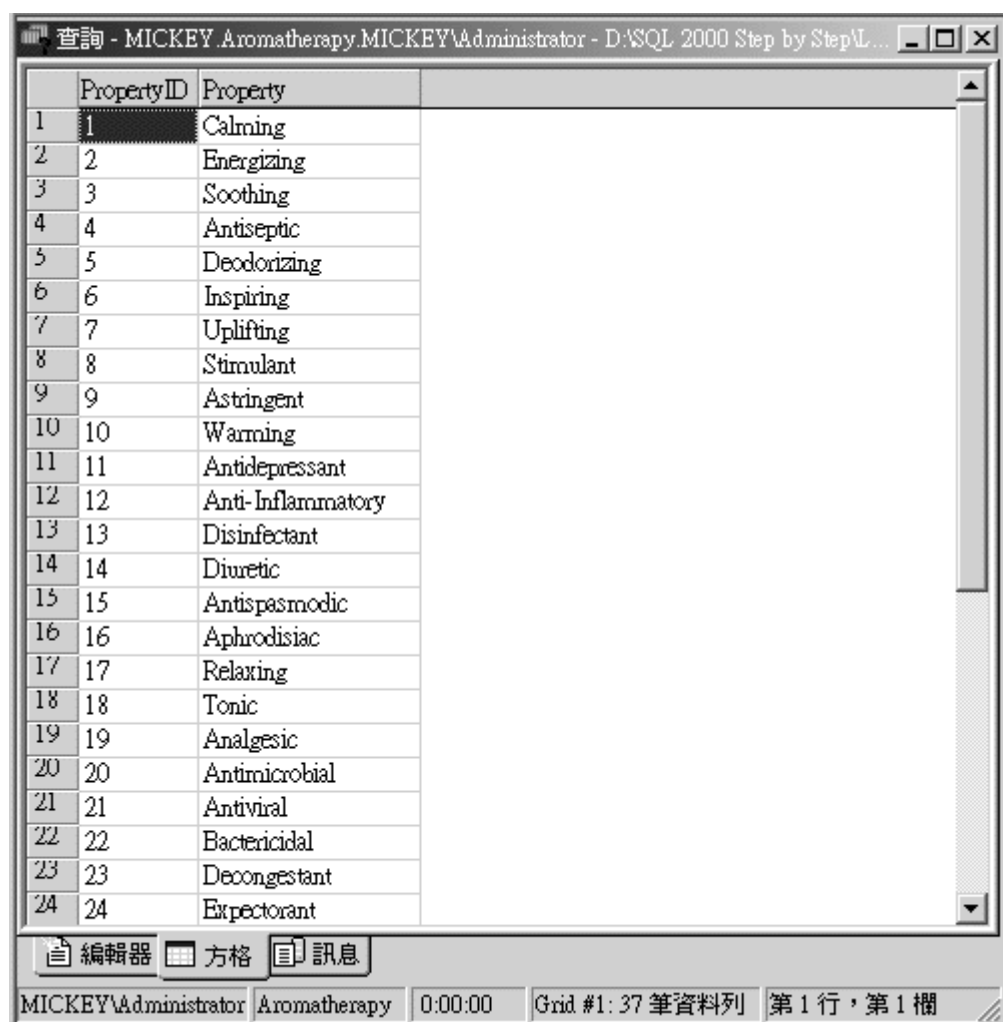


6. 按一下 [執行查詢](#) 按钮以便执行此查询。



执行查询按钮

Query Analyzer 会在 [方格](#) 窗格内显示该查询执行的结果。

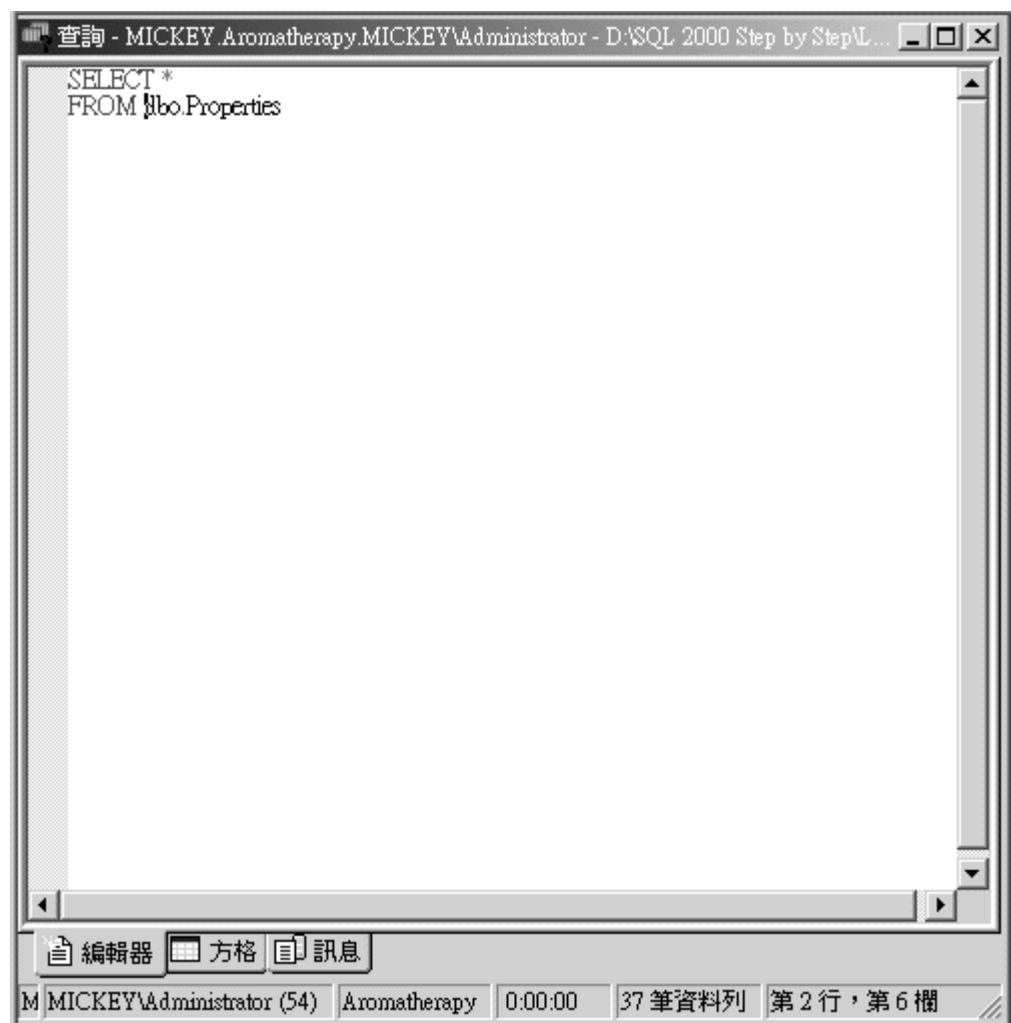


The screenshot shows a window titled "查詢 - MICKEY Aromatherapy.MICKEY\Administrator - D:\SQL 2000 Step by Step\L...". The window contains a table with 24 rows and 3 columns. The first column contains row numbers 1 through 24. The second column is labeled "PropertyID" and contains numbers 1 through 24. The third column is labeled "Property" and contains various aromatherapy properties. The table is displayed in a grid view, and the status bar at the bottom indicates "Grid #1: 37 筆資料列 第 1 行, 第 1 欄".

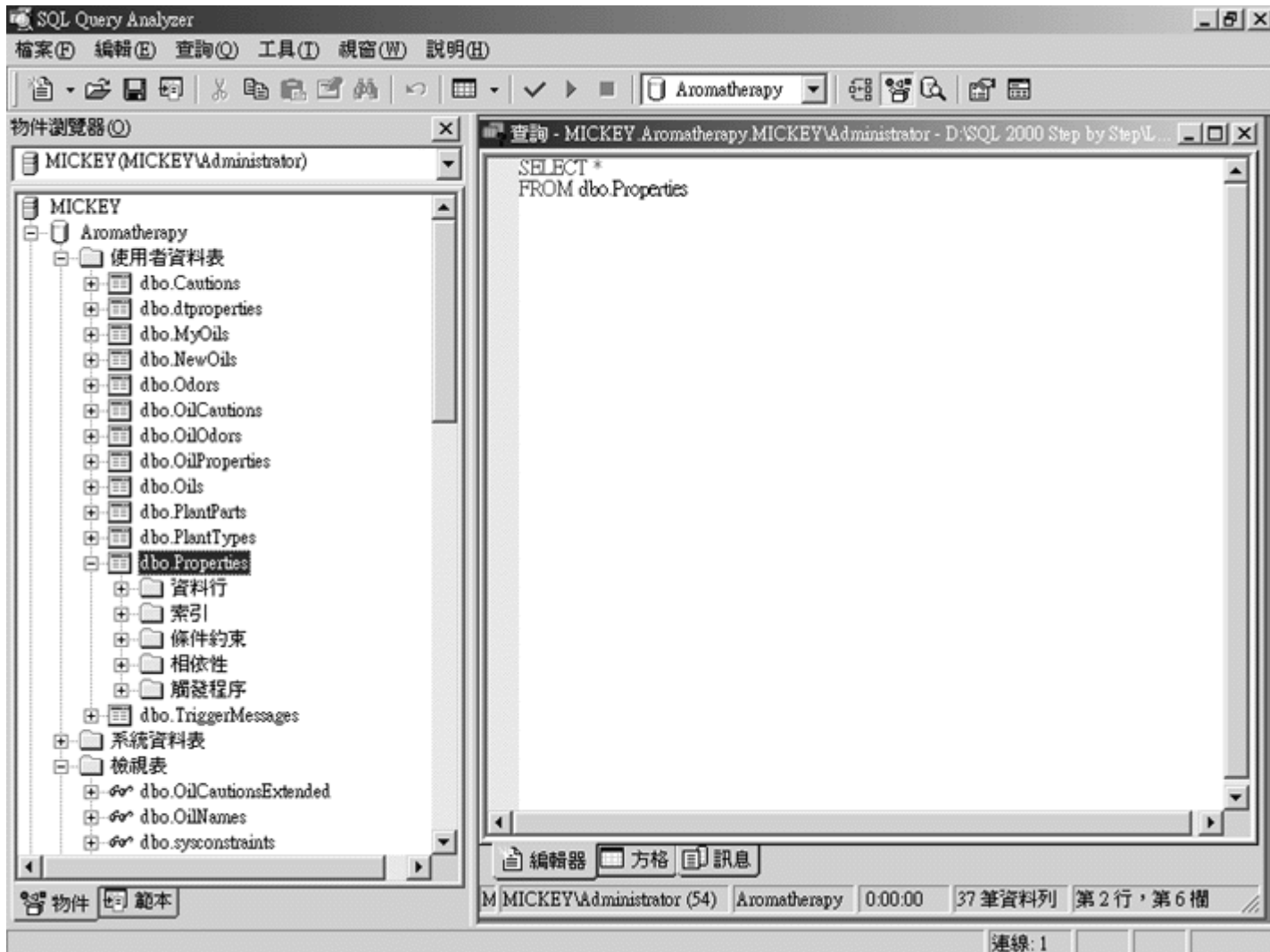
	PropertyID	Property
1	1	Calming
2	2	Energizing
3	3	Soothing
4	4	Antiseptic
5	5	Deodorizing
6	6	Inspiring
7	7	Uplifting
8	8	Stimulant
9	9	Astringent
10	10	Warming
11	11	Antidepressant
12	12	Anti-Inflammatory
13	13	Disinfectant
14	14	Diuretic
15	15	Antispasmodic
16	16	Aphrodisiac
17	17	Relaxing
18	18	Tonic
19	19	Analgesic
20	20	Antimicrobial
21	21	Antiviral
22	22	Bactericidal
23	23	Decongestant
24	24	Expectorant

在数据夹中新增所有的对象

1. 选取 [查询](#) 窗口内的 [编辑器](#) 标签页。
2. 删除 SELECT 陈述式后面的星号。

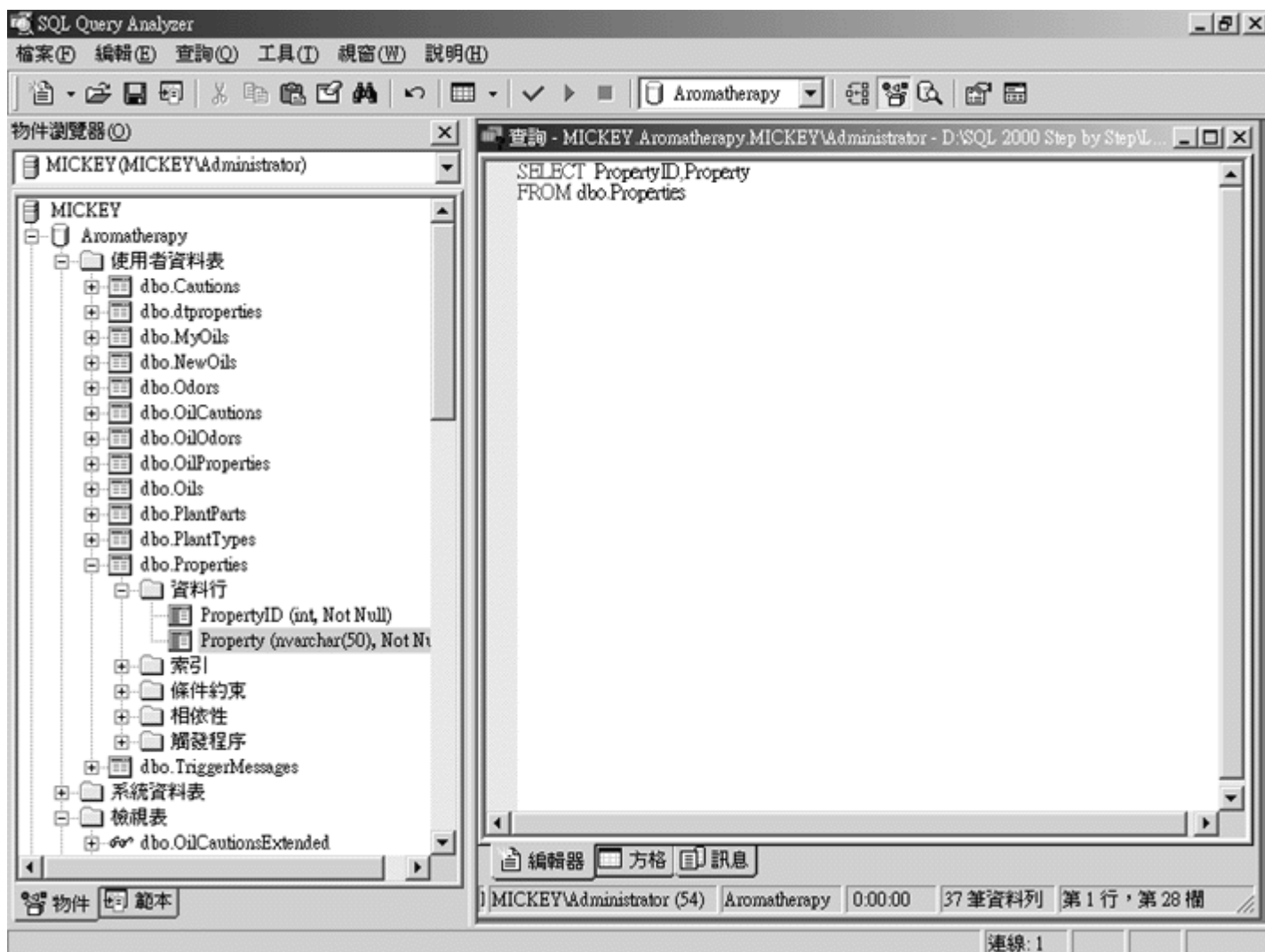


3. 展开 **对象浏览器** 内的 **dbo.Properties** 资料夹。



4. 拖曳 **对象浏览器** 中的 **数据行** 数据夹至 **编辑器** 窗格内的 **SELECT** 陈述式的后面。

Query Analyzer 会将所有的数据行名称贴至 **SELECT** 陈述式的后面。

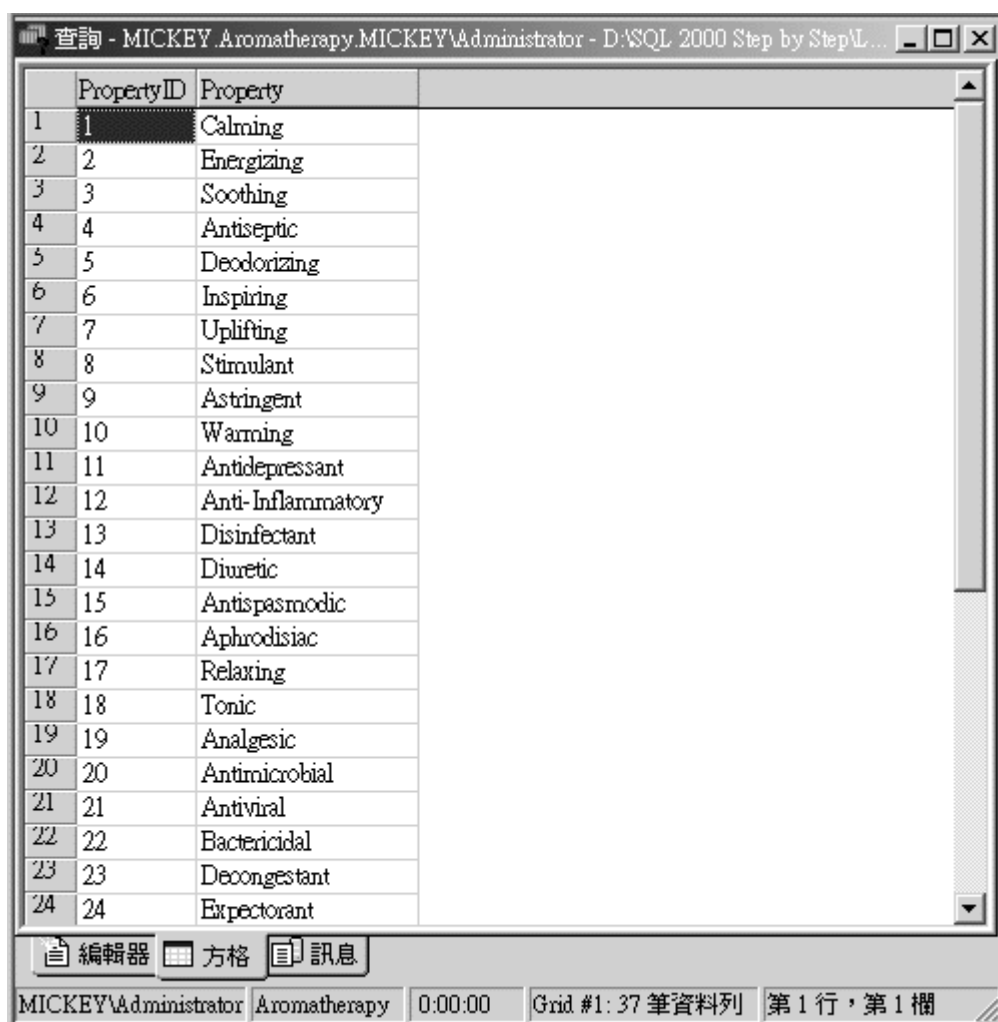


5. 按一下 [執行查詢](#) 按钮以便执行该查询。



执行查询按钮

Query Analyzer 会在 [方格](#) 窗格内显示此查询执行的结果。



The screenshot shows the 'Query - MICKEY Aromatherapy.MICKEY\Administrator - D:\SQL 2000 Step by Step\...' window. The main area displays a table with two columns: 'PropertyID' and 'Property'. The table contains 24 rows of data. At the bottom, there are tabs for '編輯器' (Editor), '方格' (Grid), and '訊息' (Messages). The 'Grid' tab is active, showing the table data. The status bar at the bottom indicates 'MICKEY\Administrator Aromatherapy 0:00:00 Grid #1: 37 筆資料列 第 1 行，第 1 欄'.

	PropertyID	Property
1	1	Calming
2	2	Energizing
3	3	Soothing
4	4	Antiseptic
5	5	Deodorizing
6	6	Inspiring
7	7	Uplifting
8	8	Stimulant
9	9	Astringent
10	10	Warming
11	11	Antidepressant
12	12	Anti-Inflammatory
13	13	Disinfectant
14	14	Diuretic
15	15	Antispasmodic
16	16	Aphrodisiac
17	17	Relaxing
18	18	Tonic
19	19	Analgesic
20	20	Antimicrobial
21	21	Antiviral
22	22	Bactericidal
23	23	Decongestant
24	24	Expectorant

指令码化对象

对象浏览器比较复杂的拖曳功能是指令码化。指令码化可以用来建立一个完整的 Transact-SQL 陈述式，它也可以透过多数对象的快捷菜单建立。并不是所有的指令码类型都可以应用在所有的对象类型。例如并非所有的函数参数或数据行都可以指令码化。

在表 21-2 中所显示的是一些指令码化可以应用的对象列表，我们将在下一章中说明关于建立及维护数据库对象。

指令码命令	可应用的对象
Create	数据表、索引、条件约束、触发程序、检视表、预存程序
Alter	触发程序、检视表
Drop	数据表、索引、条件约束、触发程序、检视表、预存程序
Select	资料表、检视表
Insert	资料表、检视表
Update	资料表、检视表
Delete	资料表、检视表
Execute	预存程序、函数

表 21-2 指令码选项

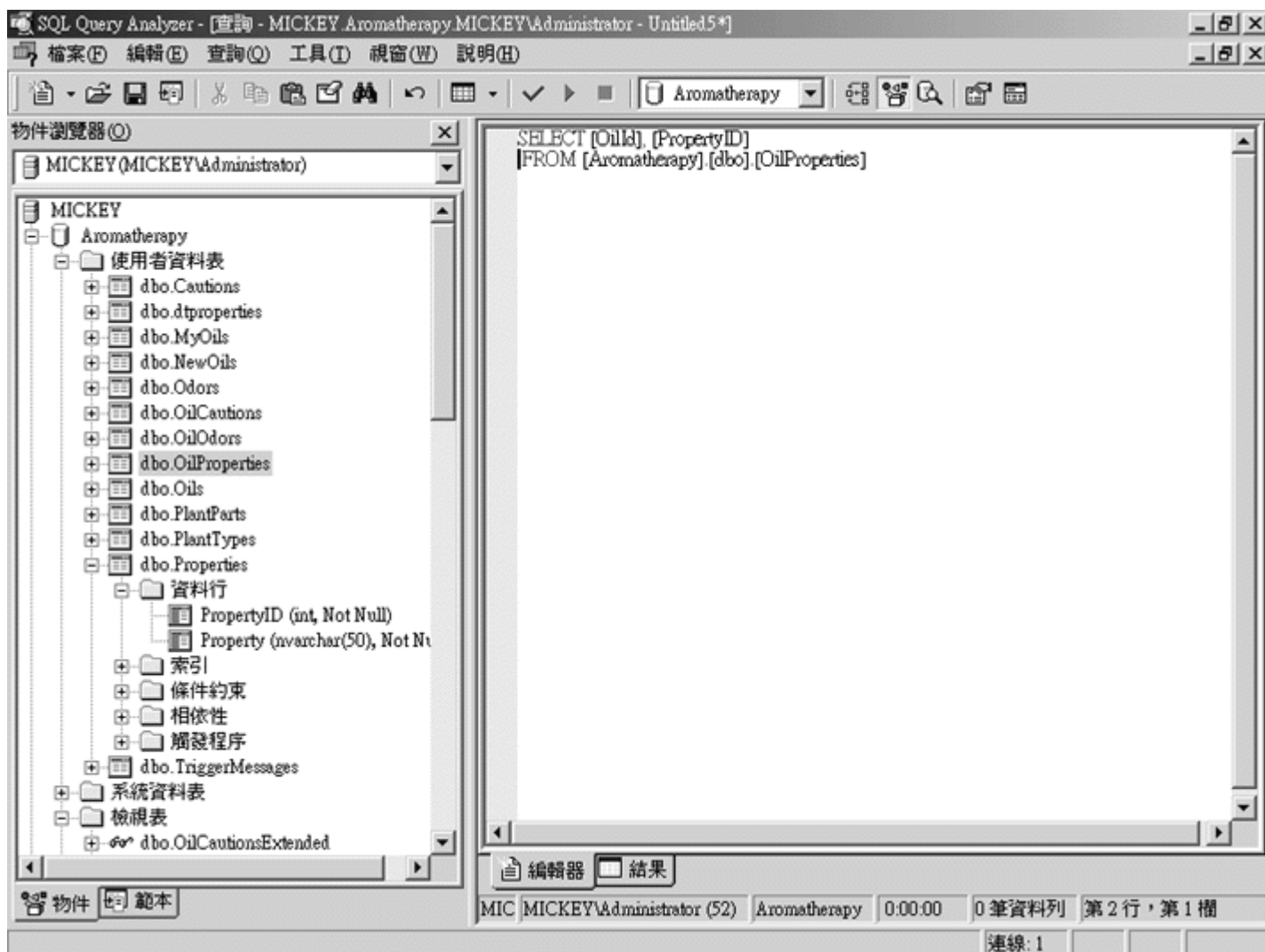
您可以在新的查询窗口写入指令码、储存在一个指令码档案，或者是暂存在剪贴簿中（它们可以贴至一个已经存在的查询窗口中）。有些指令码，例如函数的执行指令码就会使用替换性的参数。

Query Analyzer 提供对话框功能以便很容易地以适当的值来取代这些参数。

SELECT 陈述式的指令码

1. 在 [对象浏览器](#) 内，展开 Aromatherapy 数据库的 [使用者数据表](#) 数据夹。
2. 在 [dbo.OilProperties](#) 数据表上按右键，并且选取 [将对象转换为新窗口中的指令码](#) 菜单中的 **SELECT**。

Query Analyzer 会开启一个 SELECT 陈述式的新查询窗口。



提示

对象浏览器 会产生单行的 **SELECT** 陈述式。您可以将所显示的 **SELECT** 陈述式

格式重新排列，以方便您个人更容易地阅读。

3. 在 Query Analyzer 的工具列上按一下 [执行查询](#) 按钮以便执行此查询。



执行查询按钮

Query Analyzer 会在 [方格](#) 窗口内显示此查询的结果。

查詢 - MICKEY Aromatherapy.MICKEY\Administrator - Untitled5*

	OilId	PropertyID
1	1	1
2	1	2
3	1	3
4	1	15
5	1	35
6	2	1
7	2	4
8	2	5
9	2	6
10	2	7
11	3	2
12	3	8
13	3	10
14	4	2
15	4	4
16	4	8
17	4	9
18	5	1
19	5	3
20	5	11
21	5	12
22	5	13
23	5	14
24	5	30

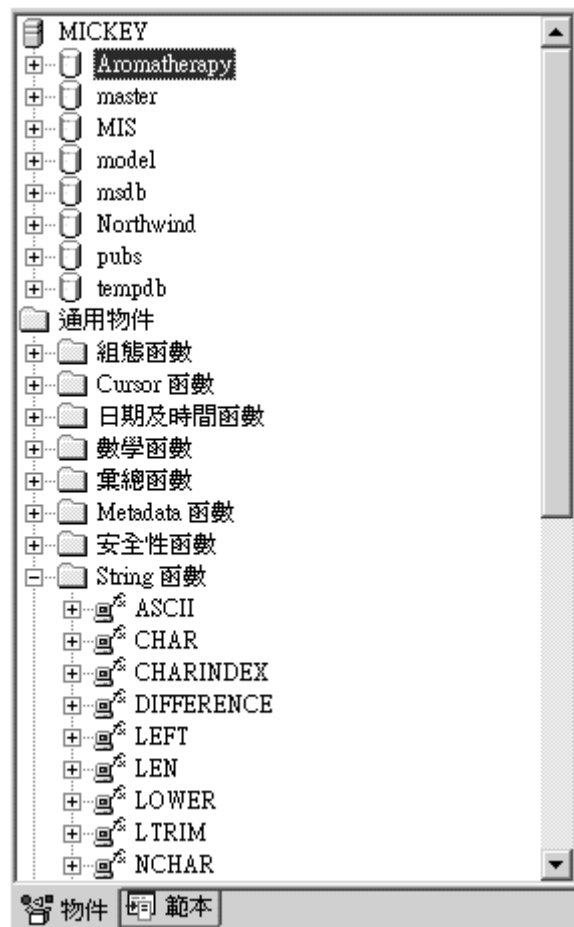
編輯器 方格 訊息

MICKEY\Administrato Aromatherapy 0:00:00 Grid #1: 224 筆資料列 第 1 行，第 1 欄

4. 当您在数据表结束浏览数据列时，请关闭查询窗口。

函数的指令码

1. 在 **通用对象** 数据夹选取 **String** 函数并展开该数据夹。



2. 在 LEFT 函数上按右键，并选取 [将对象转换为新窗口中的指令码](#) 菜单内的 [Execute](#)。

Query Analyzer 会开启一个新的 [查询](#) 窗口，并且在此窗口内包含 LEFT 函数的 SELECT 陈述式。



3. 自 [編輯](#) 菜单中选取 [取代模板参数](#)。

Query Analyzer 会开启一个 [取代模板参数](#) 对话框。

取代範本參數

參數	類型	值
character_expression	varchar	
integer_expression	int	

全部取代(R)

說明

關閉(C)

- 輸入『Test Expression』当作是参数 **character_expression** 的值；输入『4』

当作是 **integer_expression** 参数的值。

取代範本參數

參數	類型	值
character_expression	varchar	'Test Expression'
integer_expression	int	4

全部取代(R)

說明

關閉(C)

- 按一下 **全部取代** 按钮。

Query Analyzer 会在查询中将所有的参数取代。



6. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕，以便執行此查詢。




執行查詢按鈕

此 Query Analyzer 会在 [方格](#) 窗格中显示此查询的结果。



本章总结

要执行的工作	执行	按钮
启动 Query Analyzer	在工作列上按一下 开始 按钮， 并选取 程序	

	集、 Microsoft SQL Server 中的 Query Analyzer 。	
选择数据库以便执行查询	在 Query Analyzer 工具列上，自下拉式方块中选取 新增查询 。	
在查询窗口中执行 Transact-SQL 陈述式	在 编辑器 窗格内输入 Transact- SQL 陈述式，并且在 Query Analyzer 工具列上按一下 执行查询 按钮。	
建立 SQL 指令码	在 查询 窗口内编辑 Transact-SQL 陈述式，然后在 Query Analyzer 的工具列上按一下 储存查询/结果 按钮。	
加载 SQL 指令码	在 Query Analyzer 工具列上按一下 加载 SQL 指令码 按钮，并且在 开启查询档案 对话框内选取要执行的 指令码。	
在对象浏览器内开启数据表或检视表	在 对象浏览器 内的数据表或检视 表上按右键，并选取 开启旧文件 项目。	
自对象浏览器内将对象新增至编辑器窗格内窗格内	自 对象浏览器 内拖曳对象至 编辑器 的目的地位置中。	
自对象浏览器内产生	自 对象浏览器 内的对象上按右键生对象的指令码 ，并选取将对象转换为 <目的>。	

22. 数据定义语言

在本章中，您将学习到：

- 使用 **CREATE** 陈述式来建立数据库对象。
- 使用 **ALTER** 陈述式来变更数据库对象。
- 使用 **DROP** 陈述式来移除数据库对象。
- 使用对象浏览器来建立 **DLL** 指令码。
- 使用模板以便产生 **DLL** 陈述式。

在第二和第三篇中，我们已经讨论过如何在 **Enterprise Manager** 中使用 **Microsoft Visual**

Database Tools 来建立数据库对象—数据表、索引、关联性以及检视表。在本章中，我们将学

习如何使用 **Transact-SQL** 来建立、变更以及移除对象。

认识 **DLL**

SQL 语言提供了二个组件：数据操作语言（Data Manipulation Language，DML）以及数据定义语言（Data Definition Language，DDL）。DML 是让您取回数据的陈述式所组成；DDL 是您在数据库中建立对象和设定对象属性时使用的陈述式。

DML 与 DDL

为什么要特别将这二个名词分开呢？虽然 DML 陈述式非常类似于 DDL，但是 DDL 在许多的数据库产品之间都各有不同。每一个数据库管理系统厂商在实体层中所建置的关系型模块是不同的，以及每一个厂商的 DDL 会不可避免的反映这些不同。大多数的厂商针对数据定义也提供图形化工具，也有许多的厂商不限制您使用 SQL DDL（包括 Microsoft）。举例来说，Microsoft 提供 ADO 和 DAO 二种数据定义能力。

我们已经看过基本的 DML 陈述式：SELECT、INSERT、UPDATE 以及 DELETE。而基本的 SQL DDL 陈述式是 CREATE、ALTER 以及 DROP，每一个陈述式针对所建立的不同对象型态拥有一些变量。我们将在本章以及其它的章节中详细说明这些陈述式的用法。

建立对象

数据库对象是使用 **CREATE** 陈述式来建立的。**CREATE** 陈述式的语法是针对每一个不同对象的建立而不同。表 22-1 列出您可以藉由 **CREATE** 陈述式的基本语法所建立的对象。

CREATE 陈述式语法	建立的对象
CREATE DATABASE <name>	建立一个数据库对象
CREATE DEFAULT <name>	建立一个默认值 AS <条件式>
CREATE FUNCTION <name> RETURNS <return_value> AS <tsql_statements>	建立一个使用者自订函数（请参考第 30 课的 〈使用者自订函数〉 ）
CREATE INDEX <name> ON <table_or_view> (<index_columns>)	在数据表或检视表内建立索引
CREATE PROCEDURE <name> AS <tsql_statements>	建立预存程序（请参考第 28 章 〈预存程序〉 ）
CREATE RULE <name> AS <conditional_expression>	建立数据库规则
CREATE SCHEMA AUTHORIZATION <owner> <object_definitions>	建立数据表、检视表和权限为单一对象

<pre>CREATE STATISTICS <name> ON <table_or_view>(<column>)</pre>	藉由查询最佳化器来建立统计数据
<pre>CREATE TRIGGER <name> { FOR AFTER INSTEAD OF}<dml_action> AS <tsql_statements></pre>	建立触发程序（请参考第 29 章 〈触发程序〉 ）
<pre>CREATE VIEW <name> AS <select_statement></pre>	建立检视表

表 22-1 CREATE 陈述式

表 22-1 中所示的 CREATE 陈述式，其中只有 CREATE TABLE 比较复杂，这是因为它包含了数据表定义的许多不同项目，您必须要加入数据行以及每一个数据行的定义，必须至少要有名称和数据型别。您也可以任意地指定允许 NULL 的数据行、识别数据行或 GUID 数据行等等，其中包含有任何限制的默认值以及数个不同的属性值等课题，我们将不在此说明。针对数据行其简单的宣告语法如下所示：

```
<column_name><data_type>

[NULL |NOT NULL ]

[

[DEFAULT <default_value>] |
```



```
[IDENTITY [(<seed_value>, <increment_value>)[NOT FOR REPLICATION]]]  
  
[ROWGUIDCOL]  
  
[<column_constraint>[, <column_constraint>...]]
```

您要注意的是默认值以及识别数据行是相互独立的（只能择其一）。因此您可以针对数据行设定默认值，或者指定其为识别数据行，或者都不设定。

<column_constraint> 的指定方式为：

```
[CONSTRAINT <constraint_name>]  
  
[  
  
[PRIMARY KEY | UNIQUE ] [CLUSTERED |NONCLUSTERED ] |  
  
[[FOREIGN KEY] REFERENCES <referenced_table>(column_name)] |  
  
[CHECK [NOT FOR REPLICATION ] (<logical expression>)]  
  
]
```

您可以针对一个资料行来指定多于一个的**<column_constraint>**，但是您必须要指定每一个限制条件的型态（**PRIMARY**、**KEY/UNIQUE**、**FOREIGNKEY** 或 **CHECK**）个别设定。

虽然这些看起来很复杂又难懂，其实它并不是真的难懂。例如您开始下达基本定义，并且简单的新增必要子句时（举例来说，**MyColumn varchar 20**），这样看起来就很简单。也许您很少在单一的数据行定义中使用 **2** 或 **3** 个，甚至更多的子句，如下例所示：

```
MyColumn varchar(20)
```

```
MyColumn varchar(20) NOT NULL
```

```
MyColumn varchar(20)
```

```
PRIMARY KEY CLUSTERED
```

```
MyColumn varchar(20)
```

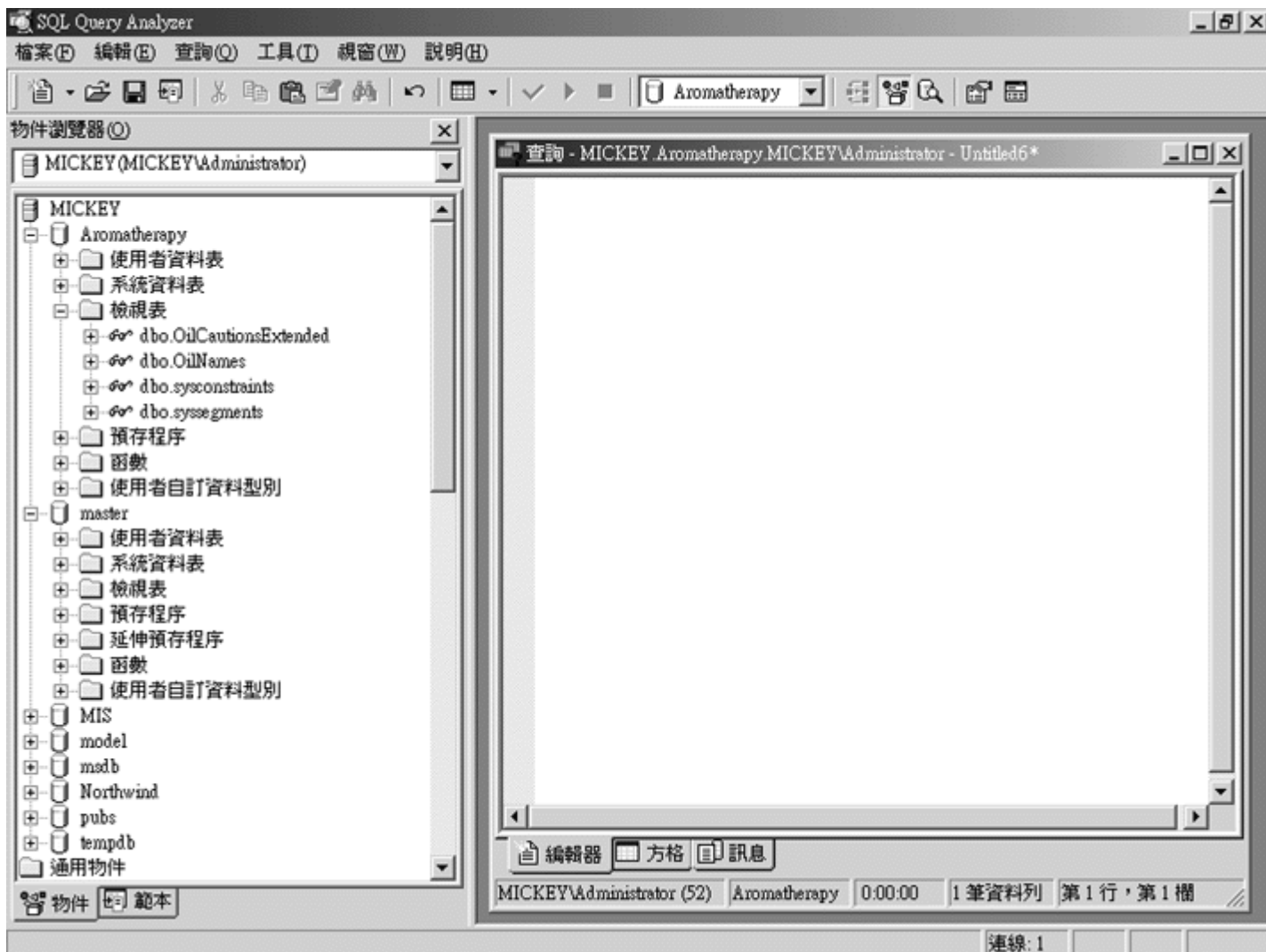
```
IDENTITY (1, 1)
```

```
PRIMARY KEY CLUSTERED
```

```
MyColumn varchar(20) NOT NULL FOREIGN KEY REFERENCES Oils(OilName)
```

建立有主索引键条件约束的数据表

1. 请确认在 **Query Analyzer** 工具列内所选取的数据库名称为 **Aromatherapy** 。



2. 在 查詢 窗口中的 編輯器 窗格，輸入如下所示的 Transact-SQL 陳述式：

```
3.      CREATE TABLE SimpleTable
4.      (
5.          SimpleID smallint
6.          IDENTITY (1, 1)
7.          PRIMARY KEY CLUSTERED,
8.          SimpleDescription varchar(50)
```

)



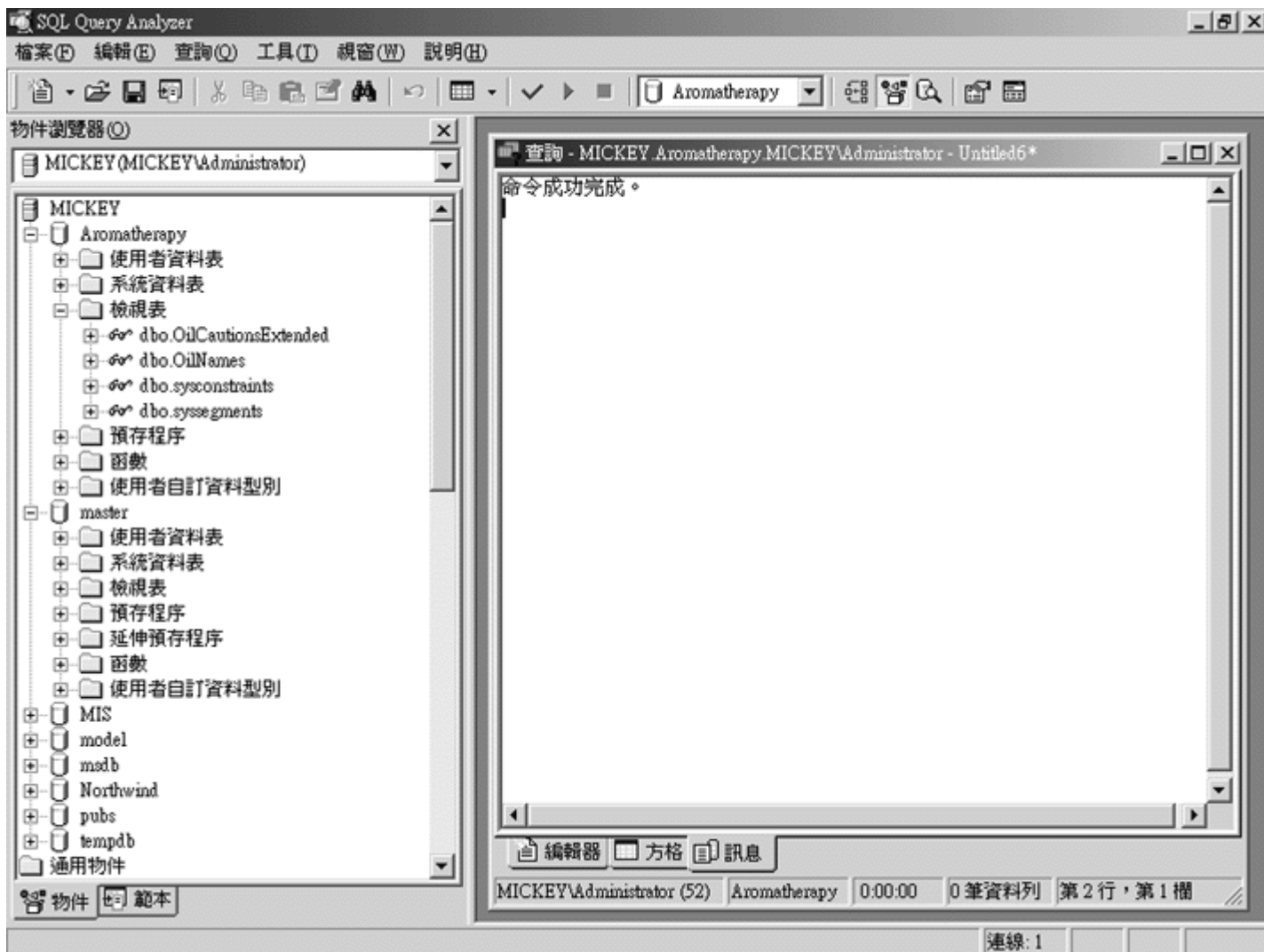
9. 在 Query Analyzer 的工具列上, 按一下 [執行查詢](#) 按钮以便执行此查询。



执行查询按钮

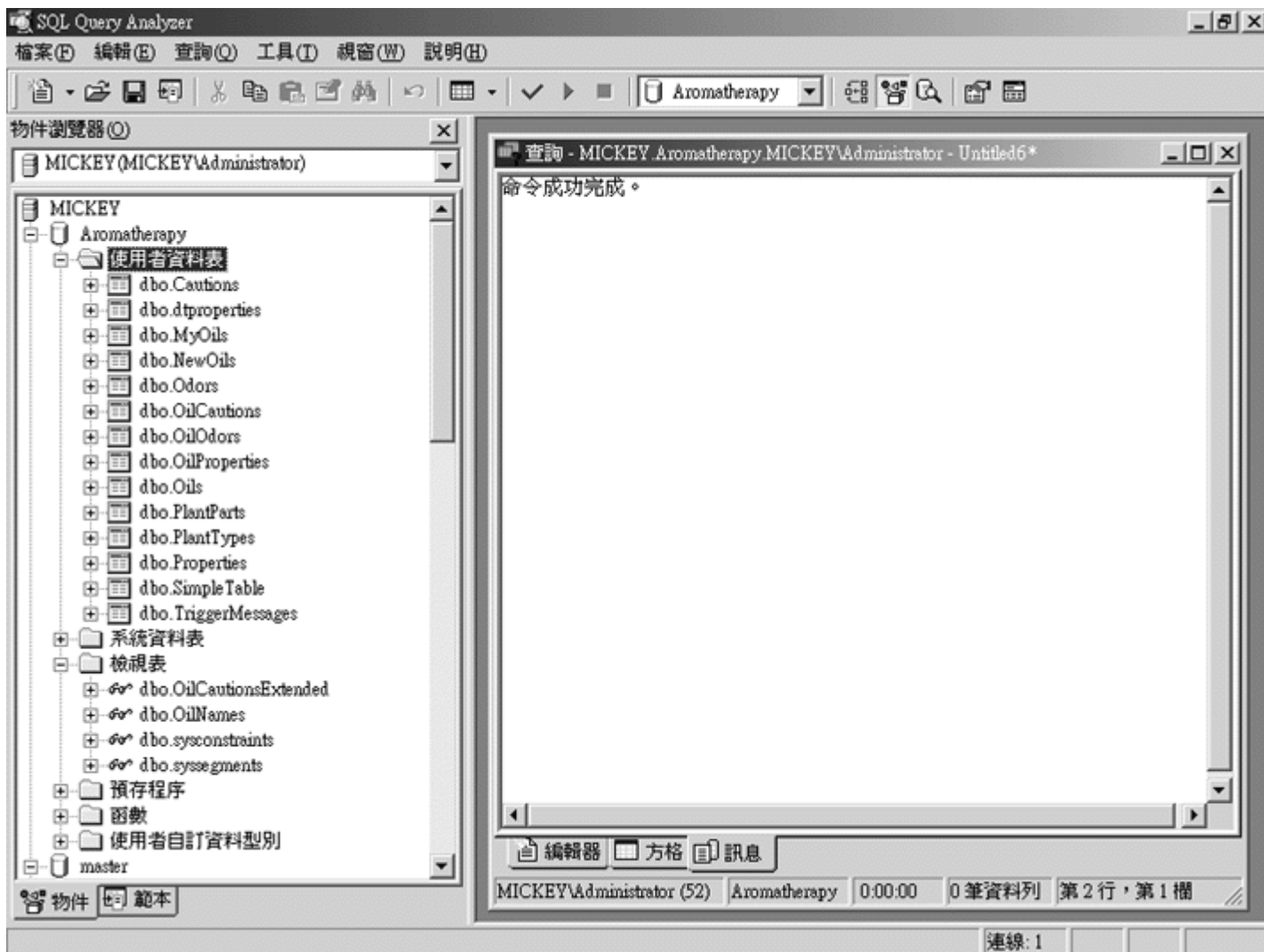
Query Analyzer 会建立一个 SimpleTable 的数据表。

10. 在 [对象浏览器](#) 中的 Aromatherapy 数据库的数据夹中，将 [使用者数据表](#) 数据夹展开。



11. 按一下 **F5** 按钮以便重新显示。

您会发现 SimpleTable 数据表已经出现在清单上。



建立有外部索引键条件约束的数据表

1. 在 **查詢** 窗口内选取 **編輯器** 标签页，并在 Query Analyzer 工具列上按一下 **清除**

窗口 按钮以便清除 **編輯器** 窗格内的内容。



清除窗口按钮

2. 在 [编辑器](#) 窗格内输入如下所示的 **Transact-SQL** 陈述式：

```
3.      CREATE TABLE RelatedTable
4.      (
5.          RelatedID SMALLINT
6.              IDENTITY (1, 1)
7.              PRIMARY KEY CLUSTERED,
8.          SimpleID SMALLINT
9.              REFERENCES SimpleTable
10.         (SimpleID),
11.         RelatedDescription varchar(20)
12.     )
```




11. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕以便執行 Transact-SQL 陈述式。



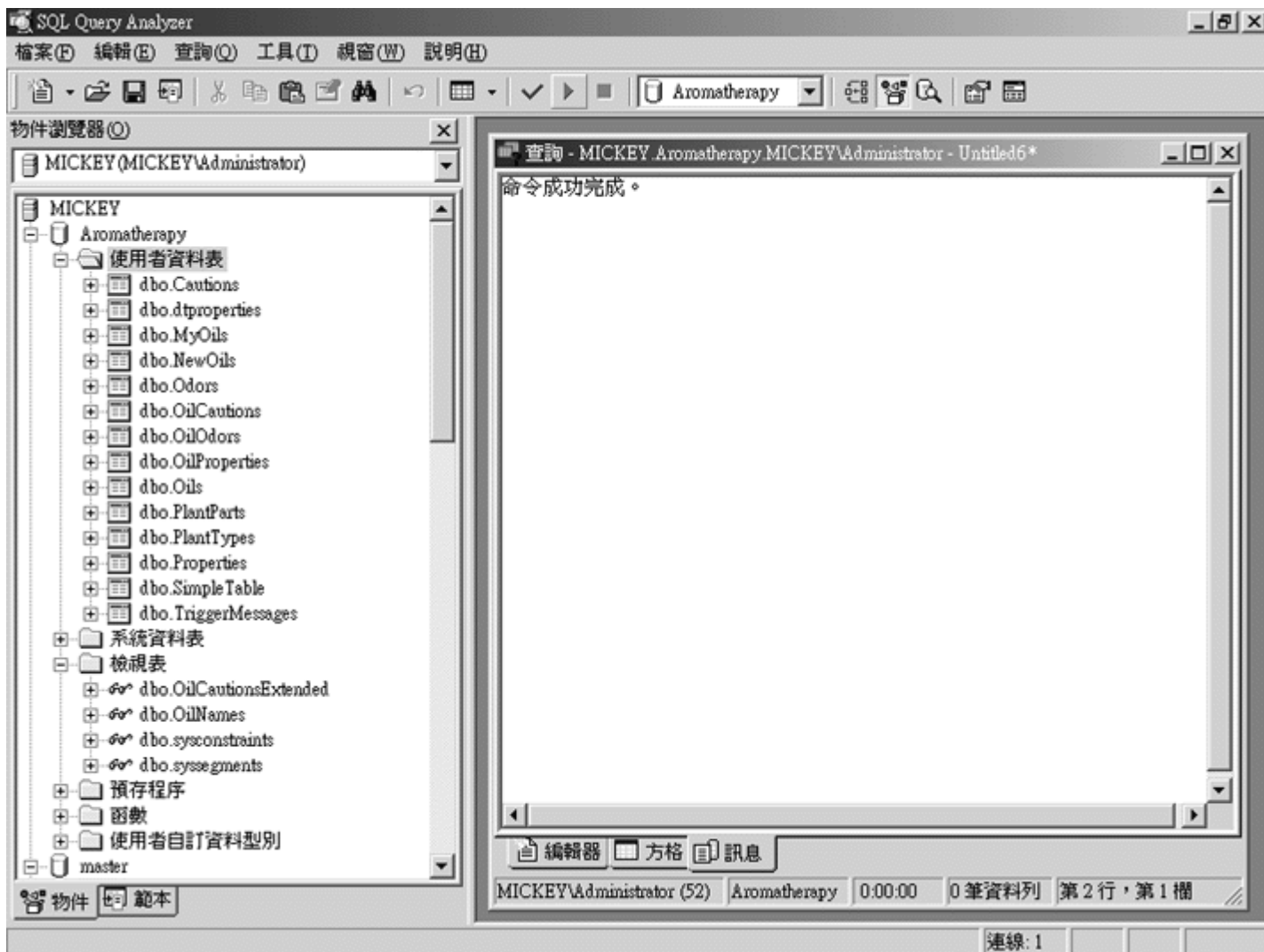
執行查詢按鈕

Query Analyzer 会建立此数据表。

12. 在 [对象浏览器](#) 内的任何位置按一下。

13. 按一下 **F5** 按钮以便重新显示。

此时 [对象浏览器](#) 会在 [使用者数据表](#) 数据夹内显示新的 RelatedTable 资料表。



建立一个检视表

1. 在 **查询窗口** 内选取 **编辑器** 标签页，并在 Query Analyzer 工具列上按一下 **清除**

窗口 按钮以便清除 **编辑器** 窗格的内容。



清除窗口按钮

2. 在 [编辑器](#) 窗格内输入如下所示的 **Transact-SQL** 陈述式：

```
3.      CREATE VIEW SimpleView
4.      AS
5.      SELECT RelatedID, SimpleDescription, RelatedDescription
6.      FROM RelatedTable
7.      INNER JOIN SimpleTable
      ON RelatedTable.SimpleID = SimpleTable.SimpleID
```



8. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕以便執行 Transact-SQL 陈述式。



執行查詢按鈕

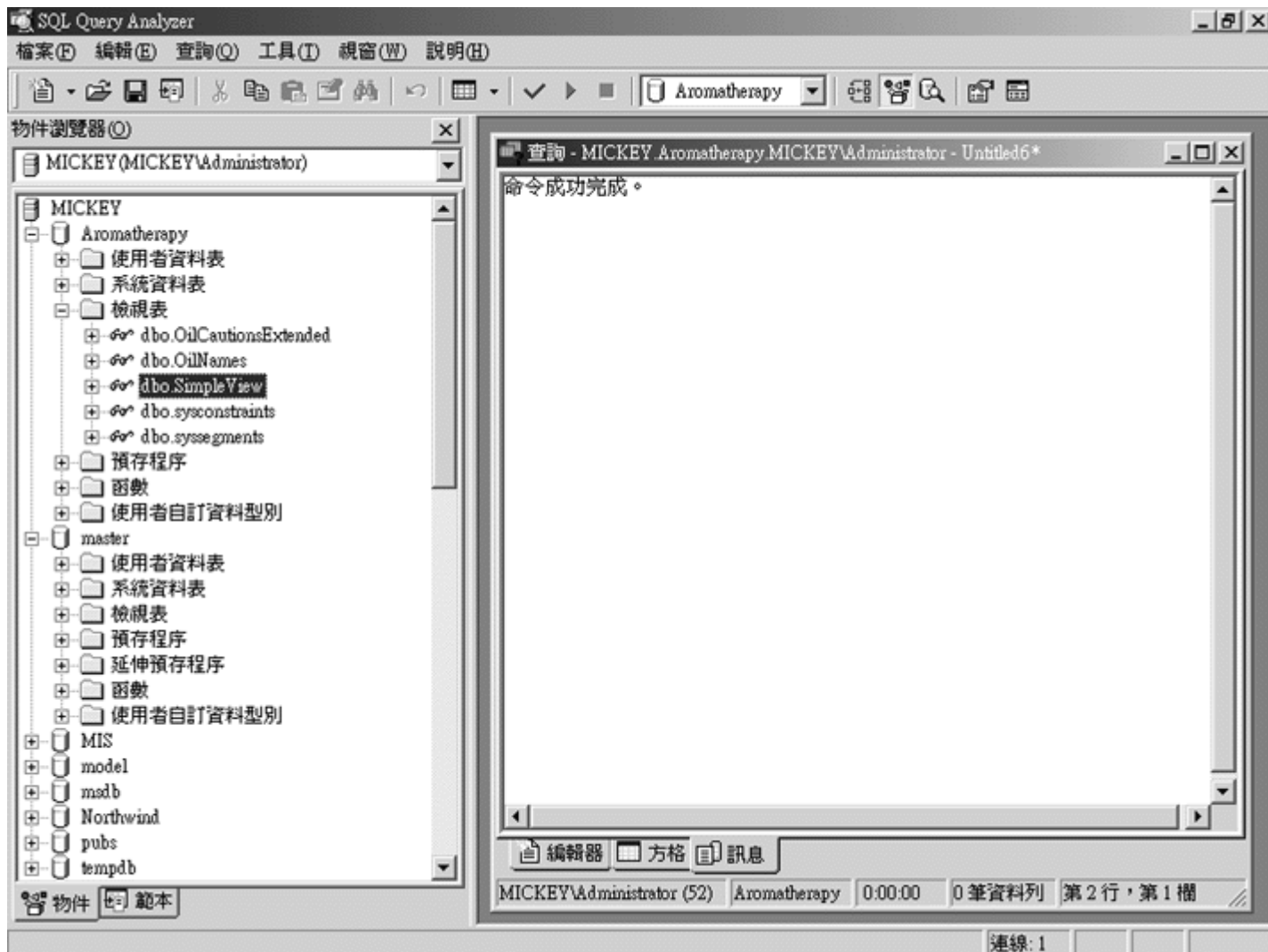
Query Analyzer 会建立此检视表。

9. 在 **对象浏览器** 内展开 **Aromatherapy** 数据库中的 **检视表** 数据夹。



10. 按一下 **F5** 按钮以便重新显示。

对象浏览器 会在 **检视表** 资料夹内显示新的 SimpleView 检视表。



建立索引



清除窗口按钮

1. 在 [查询](#) 窗口内选取 [编辑器](#) 标签页，并在 Query Analyzer 工具列上按一下 [清除窗口](#) 按钮以便清除 [编辑器](#) 窗格的内容。
2. 在 [编辑器](#) 窗格内输入如下所示的 Transact-SQL 陈述式：

```
CREATE INDEX SimpleIndex ON SimpleTable (SimpleDescription)
```



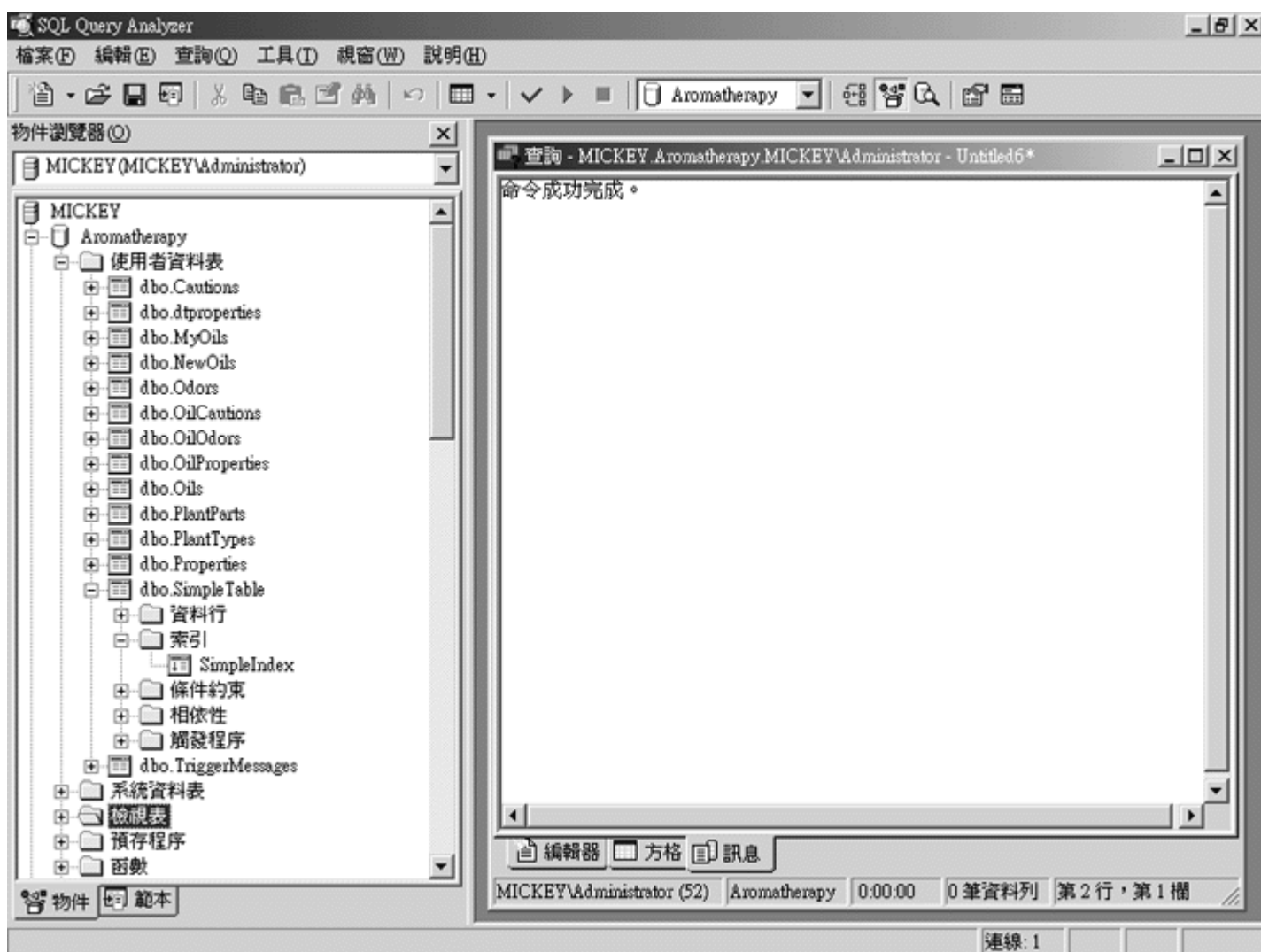

3. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕以便執行 Transact-SQL 陈述式。



執行查詢按鈕

Query Analyzer 将会建立此索引。

4. 展开 SimpleTable 数据表内的 [索引](#) 数据夹, 请确认 SimpleIndex 索引已经被新增。



变更数据表

CREATE 陈述式是用来建立新的对象，ALTER 陈述式则是提供了变更存在对象定义的机制。并非所有使用 CREATE 陈述式所建立的对象都可以使用 ALTER 陈述式变更。表 22-2 中所示的语法是可以使用 ALTER 陈述式以变更的对象。

ALTER 陈述式语法	结果
ALTER DATABASE <name> <file_specification>	变更用于储存数据库的档案
ALTER FUNCTION <name> RETURNS <return_value> AS <tsql_statements>	变更组成函数的 T-SQL 陈述式
ALTER PROCEDURE <name> AS <tsql_statements>	变更组成预存程序的 T-SQL 陈述式（请参考第 28 章 〈预存程序〉 ）
ALTER TABLE <name> <change_definition>	变更资料表定义（本章中将会针对<change_definition>来详细说明）
ALTER TRIGGER <name> {FOR AFTER INSTEADOF} <dml_action>AS <tsql_statements>	变更组成触发程序的 T-SQL 陈述式（请参考第 29 章 〈触发程序〉 ）
ALTER VIEW <name>	变更建立检视表的 SELECT 陈述式

AS <select_statement>	
-----------------------	--

表 22-2 ALTER 陈述式

使用 **ALTER TABLE** 陈述式的复杂原因和使用 **CREATE TABLE** 陈述式是一样的：建立数据表定义分成许多不同的部份。下列所示为 **ALTER TABLE** 陈述式的简单的版本：

```
ALTER TABLE <name>

{

[ALTER COLUMN <column_definition>] |

[ADD <column_definition>] |

[DROP COLUMN <column_name>] |

[ADD [WITH NOCHECK] CONSTRAINT <table_constraint>]

}
```

在数据表条件约束之前的 **CHECK** 和 **NOCHECK** 关键词是要指示 **SQL Server** 必须测试（或不测试）数据表中现存的资料是否有违反新的限制条件。**WITH NOCHECK** 使用的情形非常少。

变更数据行

ALTER COLUMN 子句在使用上有许多的限制，假如它是下列其中之一的情形时，您就无法变

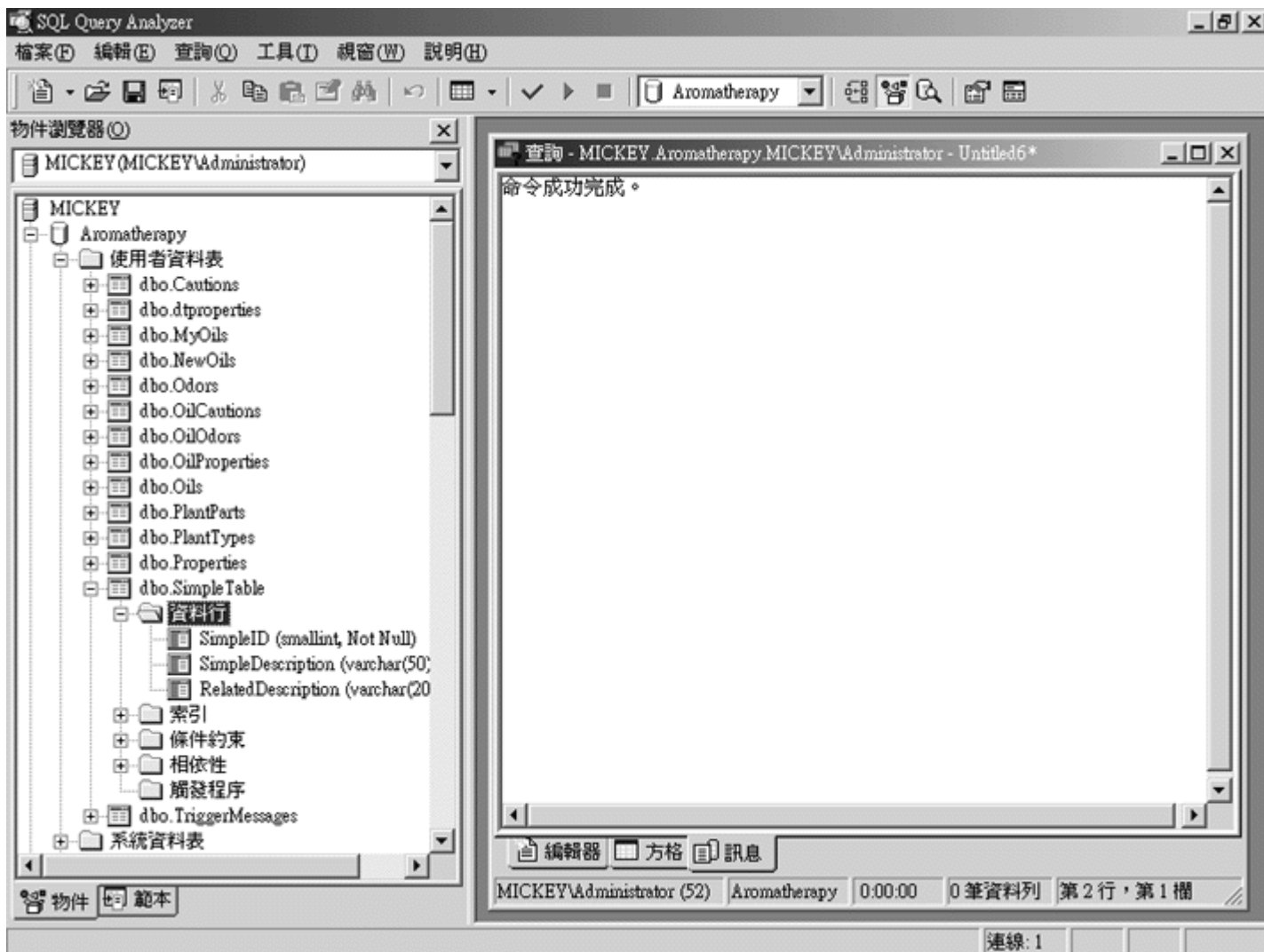
更该资料行：

- 资料型别为 text、image、ntext 或 timestamp。
- 定义为数据表的 ROWGUIDCOL。
- 自身为计算数据行，或者被计算数据行引用。
- 被复写。
- 被用于索引—除非该数据行拥有 varchar、nvarchar 或 varbinary 资料型别；这些数据型别不能被变更，并且数据行的大小不能减少。
- 用于 CREATE STATISTICS 陈述式产生统计数据的数据行。
- 用于 PRIMARY KEY 条件约束。
- 用于 FOREIGN KEY 条件约束。

- 用于 CHECK 条件约束。
 - 用于 UNIQUE 条件约束。
 - 连结了 DEFAULT（预设）。
-

变更检视表

1. 展开 [对象浏览器](#) 中 SimpleView 的 [数据行](#) 数据夹。



2. 在 **查詢** 窗口內选取 **編輯器** 標籤頁，並在 Query Analyzer 工具列上按一下 **清除**

窗口 按鈕以便清除 **編輯器** 窗格內的內容。



清除窗口按鈕

3. 在 [编辑器](#) 窗格内输入如下所示的 Transact-SQL 陈述式:

```
4.      ALTER VIEW SimpleView
5.      AS
6.      SELECT SimpleDescription, RelatedDescription
7.      FROM RelatedTable
8.      INNER JOIN SimpleTable
ON RelatedTable.SimpleID =SimpleTable.SimpleID
```



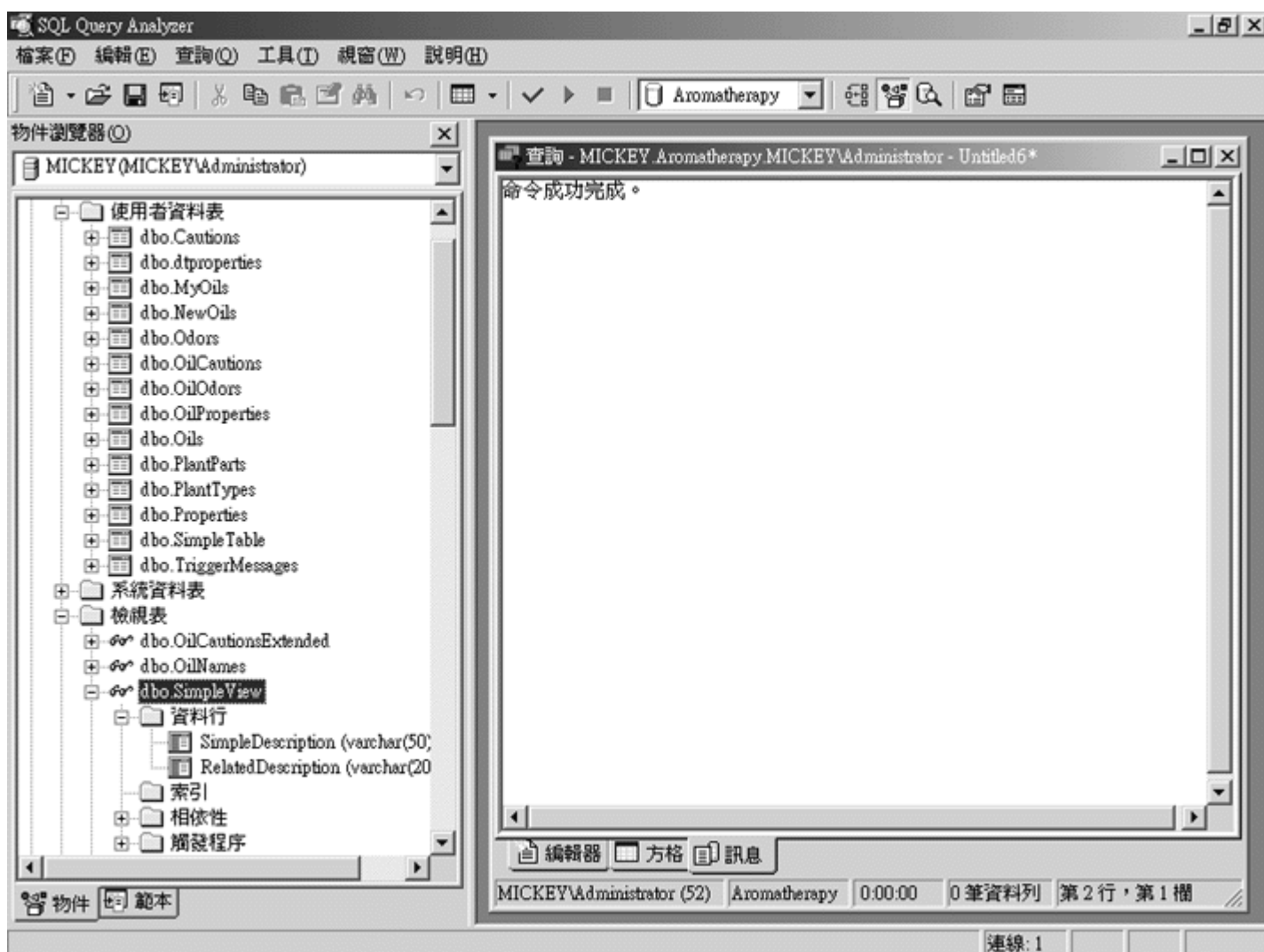

9. 在 Query Analyzer 工具列上按一下 [執行查詢](#) 按鈕以便執行該陳述式。



執行查詢按鈕

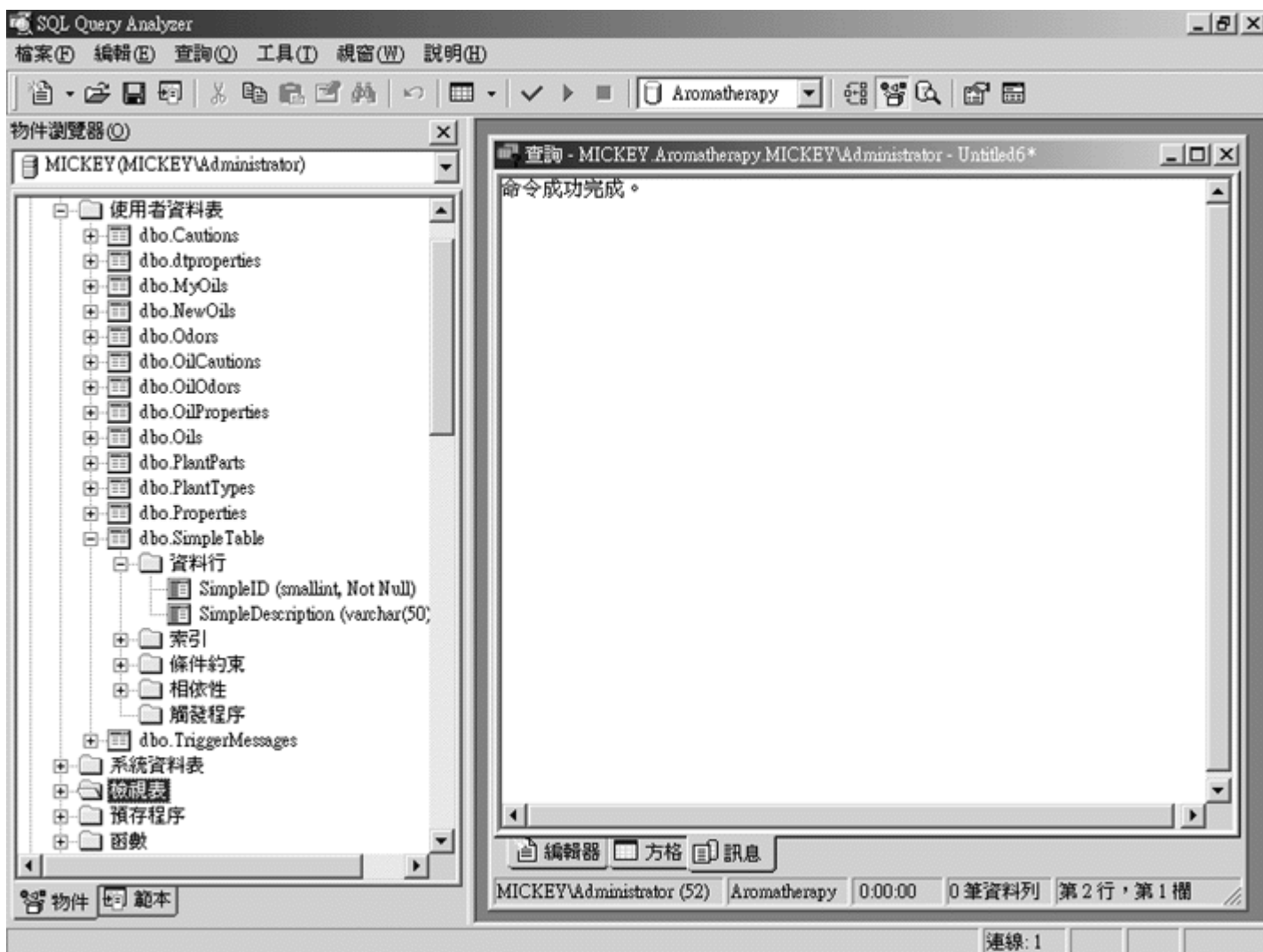
10. 在 [对象浏览器](#) 中的任何位置按一下，并且按下 **F5** 按钮以便重新显示。

现在 [对象浏览器](#) 中只会显示 SimpleDescription 以及 RelatedDescription 资料行。



新增数据行至数据表中

1. 展开 [对象浏览器](#) 中 SimpleTable 数据表的 [数据行](#) 数据夹。



2. 选取 [查詢](#) 窗口內的 [編輯器](#) 標籤頁, 然后在 [Query Analyzer](#) 的工具列上按一下 [清除窗口](#) 按鈕以便清除 [編輯器](#) 窗格內的内容。



清除窗口按钮

3. 在 [编辑器](#) 内输入如下列所示的 Transact-SQL 陈述式:

```
4.      ALTER TABLE SimpleTable  
      ADD NewColumn varchar(20)
```



5. 在 Query Analyzer 工具列上按一下 [執行查詢](#) 按鈕，以便執行此陳述式。

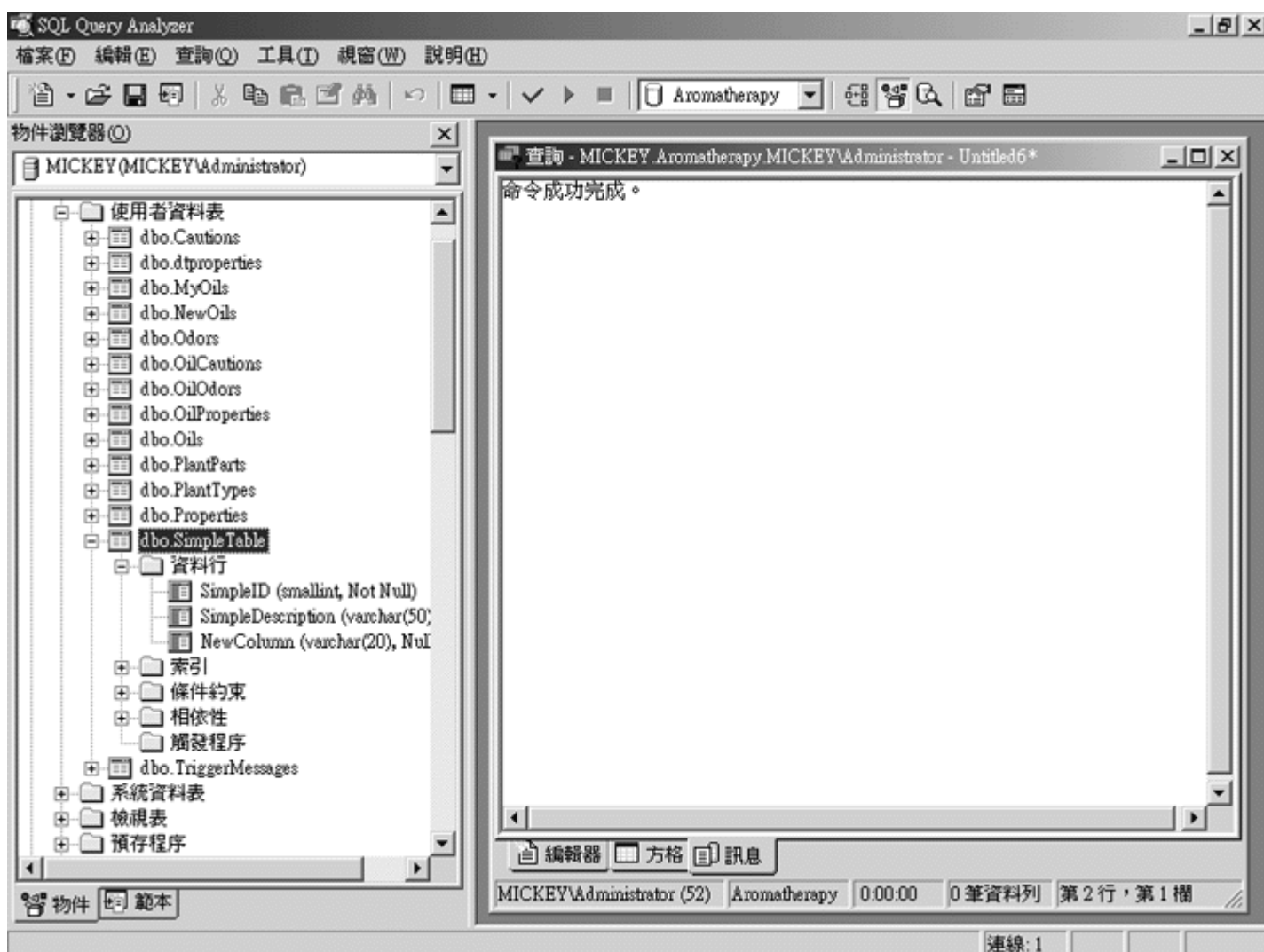


執行查詢按鈕

Query Analyzer 会在此数据表内加入数据行。

6. 在 [对象浏览器](#) 中的任何位置按一下，并且按下 **F5** 按钮以便重新显示。

此时 [对象浏览器](#) 会显示新的资料行。



在数据表中变更数据行

- 1. 选取 查询窗口 内的 编辑器 标签页，然后在 Query Analyzer 的工具列上按一下 清除窗口 按钮以便清除 编辑器 窗格内的内容。

- 2. 在 编辑器 窗格内输入下列的 Transact-SQL 陈述式：

3. ALTER TABLE SimpleTable

ALTER COLUMN NewColumn varchar(10)



清除窗口按钮



4. 在 Query Analyze 工具列上按一下 [執行查詢](#) 按鈕，以便執行此陈述式。



执行查询按钮

Query Analyzer 会在此数据表内变更 NewColumn 数据行。

5. 在 [对象浏览器](#) 内的任何位置按一下，并且按下 **F5** 按钮以便重新显示。

[对象浏览器](#) 会显示新的资料型别。



从数据表中移除数据行

1. 在 **查询** 窗口内选取 **编辑器** 标签页，并在 **Query Analyzer** 工具列上按一下 **清除窗口** 按钮以便清除 **编辑器** 窗格的内容。



清除窗口按钮

2. 在 **编辑器** 窗格中输入入如下所示的陈述式内容：

```
3.      ALTER TABLE SimpleTable  
      DROP COLUMN NewColumn
```



4. 在 Query Analyzer 工具列上执行 [执行查询](#) 按钮，以便执行此陈述式。

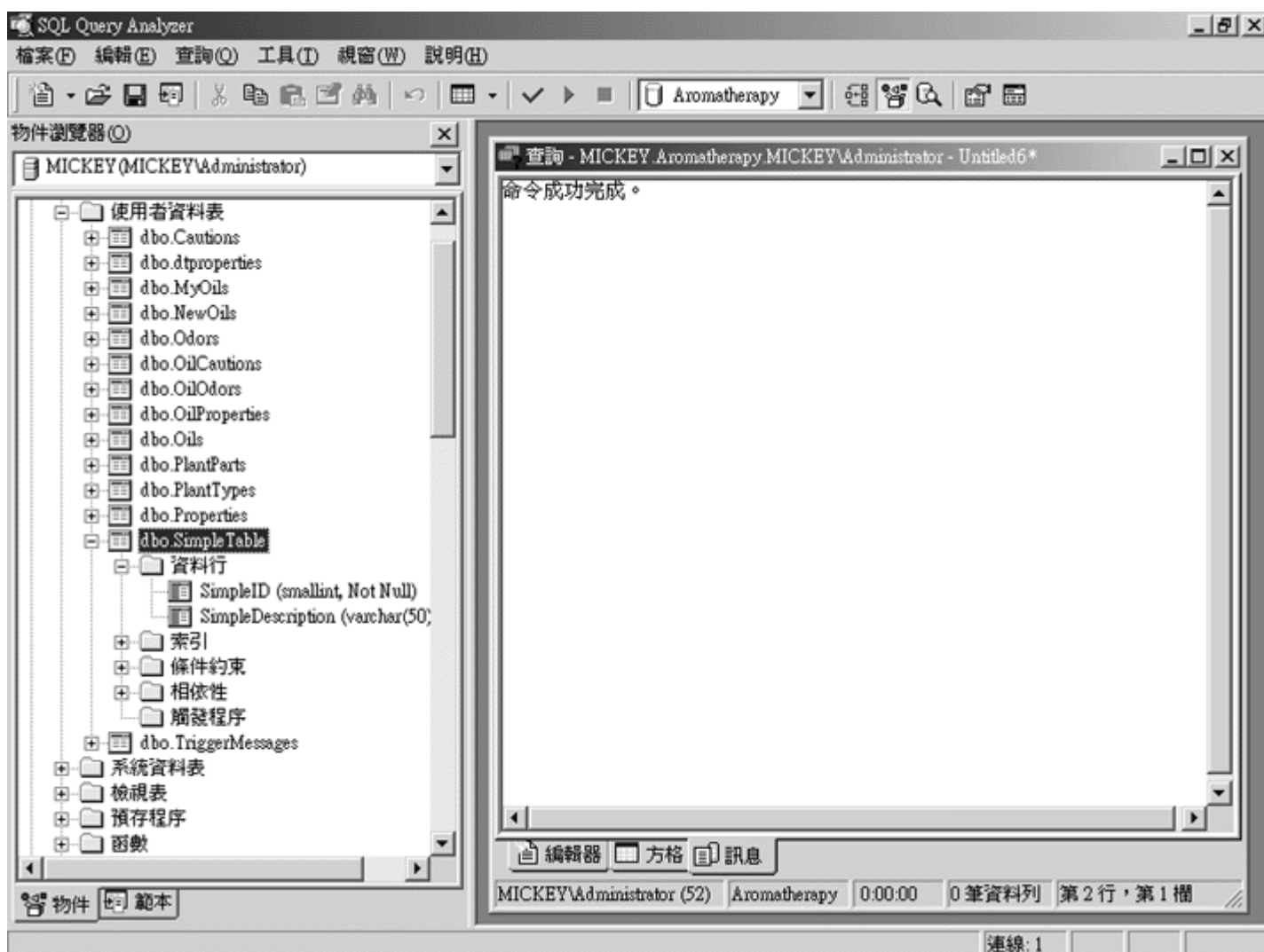


执行查询按钮

Query Analyzer 会自数据表中移除数据行。

5. 在 **对象浏览器** 内的任何位置按一下，并且按下 **F5** 按钮以便重新显示。

对象浏览器并不会显示 **NewColumn** 资料行。



移除对象

DROP 陈述式可以用来移除一个数据库对象。不像 CREATE 和 ALTER 陈述式一样，所有的

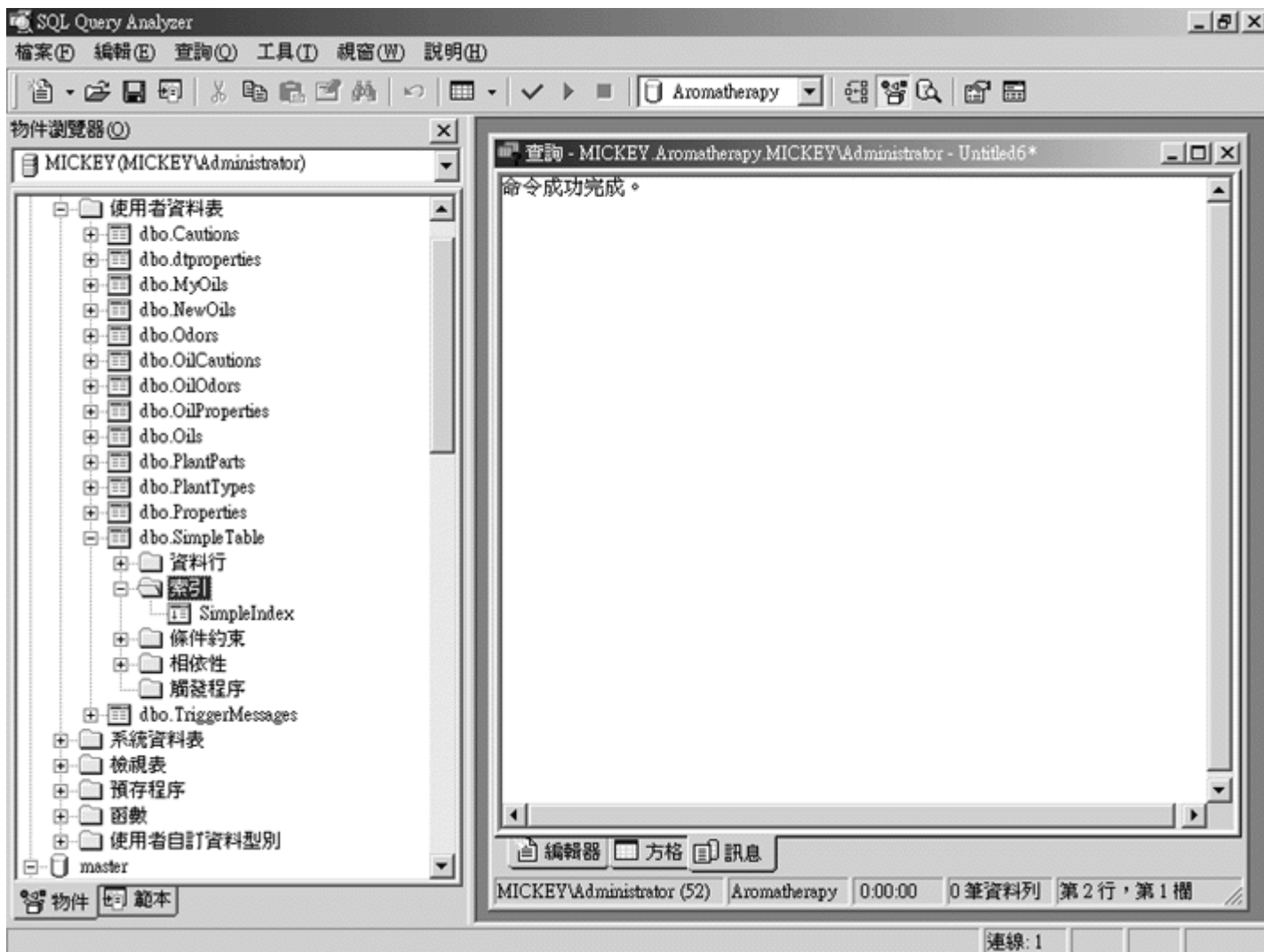
DROP 陈述式都拥有相同的简单语法：

```
DROP <object_type> <name>
```

其中<object_type>是指在 [表 22-1](#) 所示的对象，除了架构对象（schema）。

移除索引

1. 展开 [对象浏览器](#) 中 SimpleTable 数据表的 [索引](#) 数据夹。



2. 在 **查詢** 窗口內選取 **編輯器** 標籤頁，並且在 Query Analyzer 工具列上按一下 **清除窗口** 按鈕以便清除 **編輯器** 窗格的内容。



清除窗口按鈕

-
3. 在 [編輯器](#) 窗格內輸入如下所示的陈述式內容：

```
DROP INDEX SimpleTable.SimpleIndex
```



4. 在 Query Analyzer 工具列上執行 [執行查詢](#) 按鈕，以便執行此陈述式。

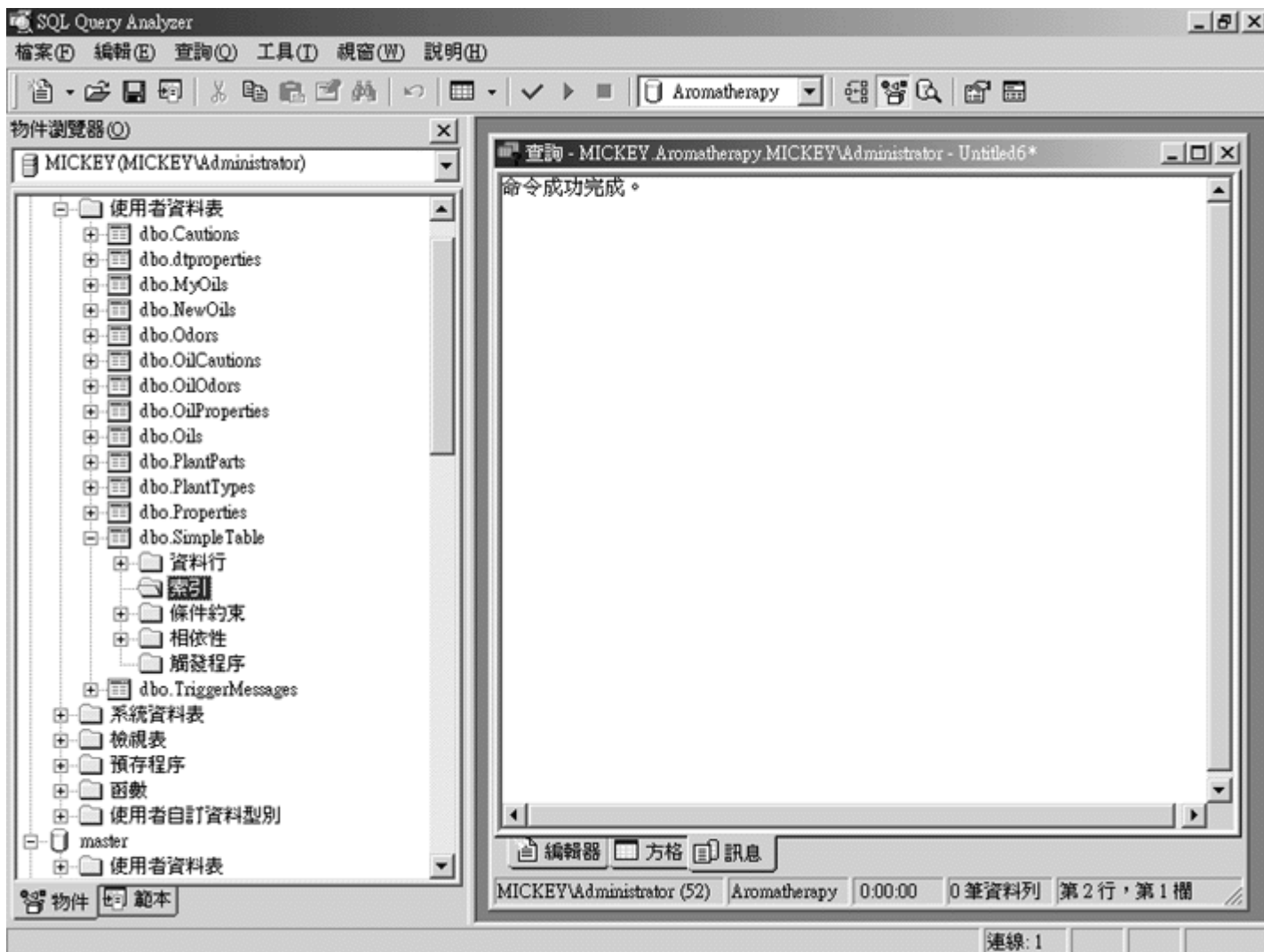


执行查询按钮

Query Analyzer 会移除此索引。

5. 在 [对象浏览器](#) 内的任何位置按一下，并且按下 **F5** 按钮以便重新显示。

[对象浏览器](#) 会显示一个没有包含任何内容的 [索引](#) 数据夹。



移除资料表

1. 在 [查询](#) 窗口中选取 [编辑器](#) 标签页，并在 Query Analyzer 工具列上按下 [清除窗](#)

[□](#) 按钮以便清除 [编辑器](#) 窗格内的内容。



清除窗口按钮

2. 在 **编辑器** 窗格内输入如下所示的陈述式内容:

```
DROP TABLE RelatedTable
```



3. 在 Query Analyzer 工具列上执行 [执行查询](#) 按钮，以便执行此陈述式。

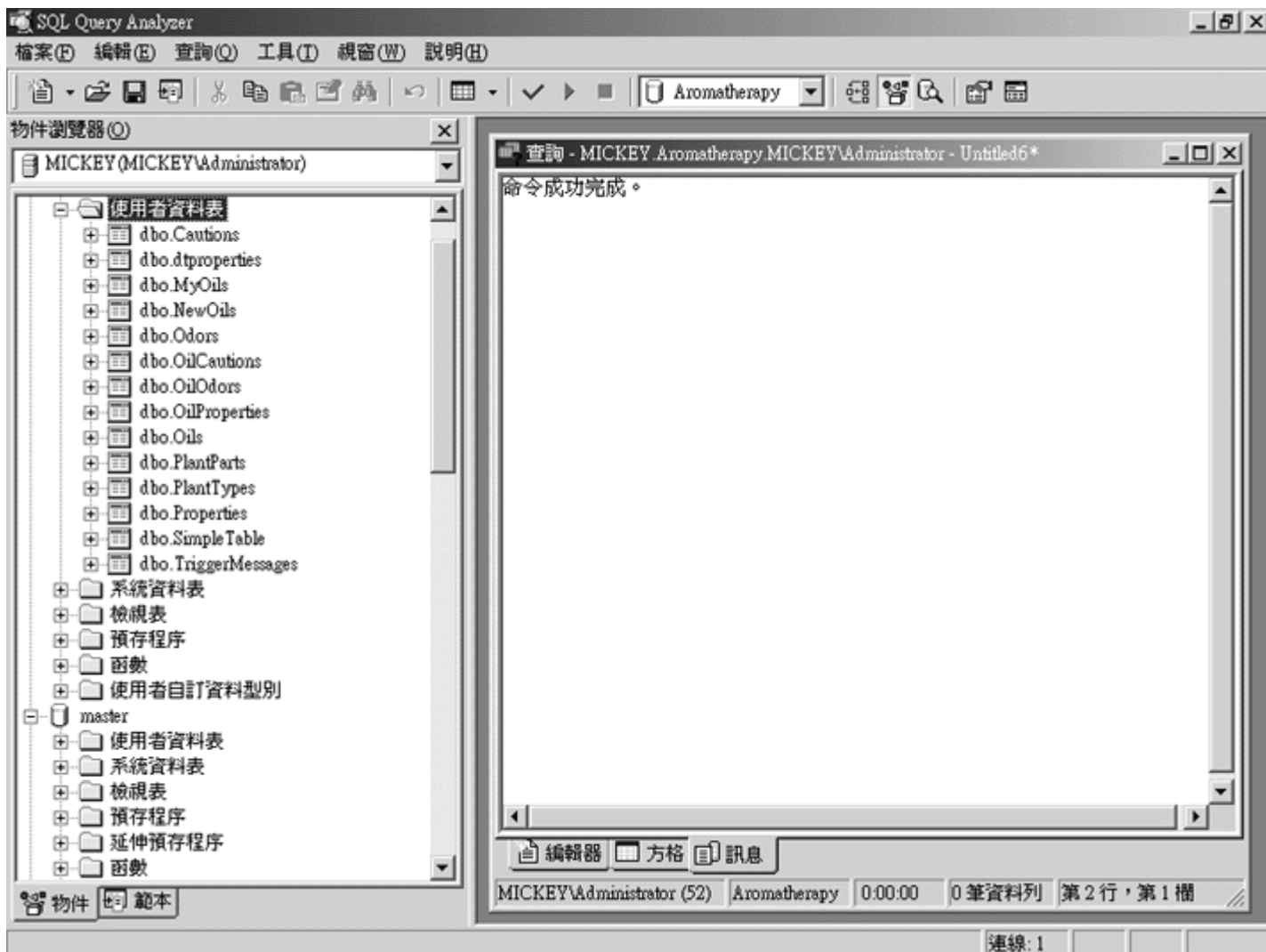


执行查询按钮

Query Analyzer 会移除此资料表。

4. 在 [对象浏览器](#) 中的 Aromatherapy 数据库中，将 [使用者数据表](#) 数据夹展开，然后按下 **F5** 按钮以便重新显示。

此时 **RelatedTable** 数据表并不会显示在对象浏览器内，这是因为此数据表已经被移除。



使用对象浏览器以设定数据定义

DDL 陈述式并不是特别的困难，只是比较复杂而已。但是 Query Analyzer 提供了二个功能，可以透过对象浏览器来让 DDL 更容易使用。如同您在前一章所看到的，多数对象的快捷菜单是支持指令码化，您可以使用这个功能针对这些对象产生 CREATE、ALTER 和 DROP 陈述式。

Query Analyzer 也提供了 **模板**（Template）以供您建立 SQL 指令码档案。您可以建立属于您自己的模板，但是 SQL Server 2000 提供基本的模板是针对多数的 CREATE 陈述式所建立的。

指令码化 DDL

在前一章中，我们看到了当我们建立 SELECT 指令码时，使用对象浏览器可以非常快速地建立指令码。对象浏览器针对管理数据库对象也支持了 CREATE、ALTER 和 DROP 指令。一旦这些指令码已经产生时，您就可以修改它直到符合您的需求为止。

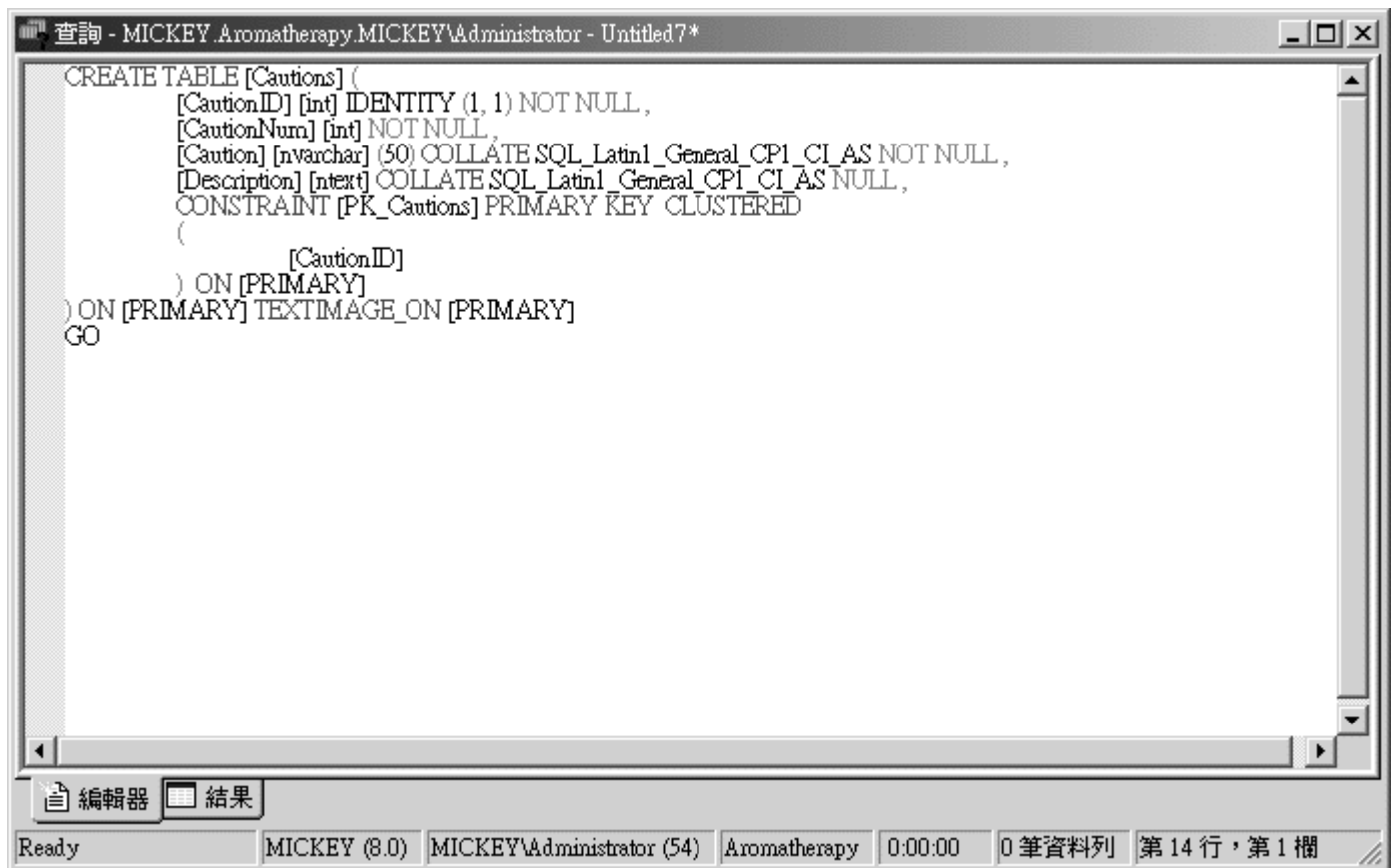
提示

指令码化功能所产生的 CREATE 陈述式可以储存为指令码档案，这对于储存数据库结构来说是一个十分方便的方式。

产生 CREATE TABLE 指令码

1. 在 **对象浏览器** 内的 **Cautions** 数据表上按右键，并且选取 **将对象转换为新窗口中的指令码** 中的 **CREATE**。

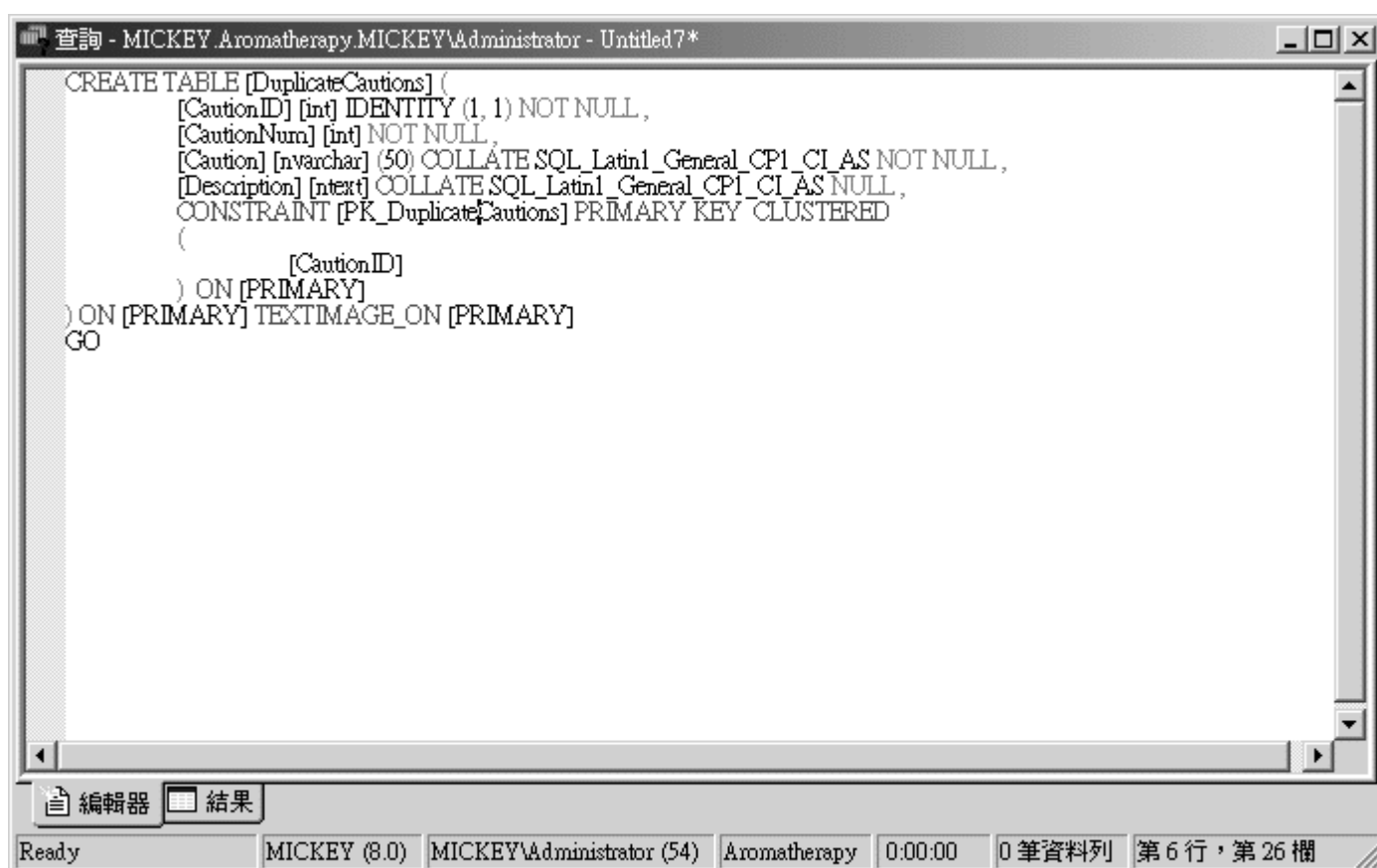
此时 Query Analyzer 会开启一个产生 Cautions 数据表的 CREATE 陈述式的新查询窗口。



重要

因为数据库中不能包含多个相同的数据表名称，因此假如您要立即执行时，您必须重新编辑 **CREATE** 陈述式内容。假如您不想要建立新的数据表时，您可以将此查询窗口关闭并且直接跳至下一节。

2. 将数据表的名称更改为 **DuplicateCautions**，并且将 PRIMARY KEY 的条件约束名称更改为 **PK_DuplicateCautions**。



3. 在 Query Analyzer 工具列上执行 **执行查询** 按钮，以便执行此查询。

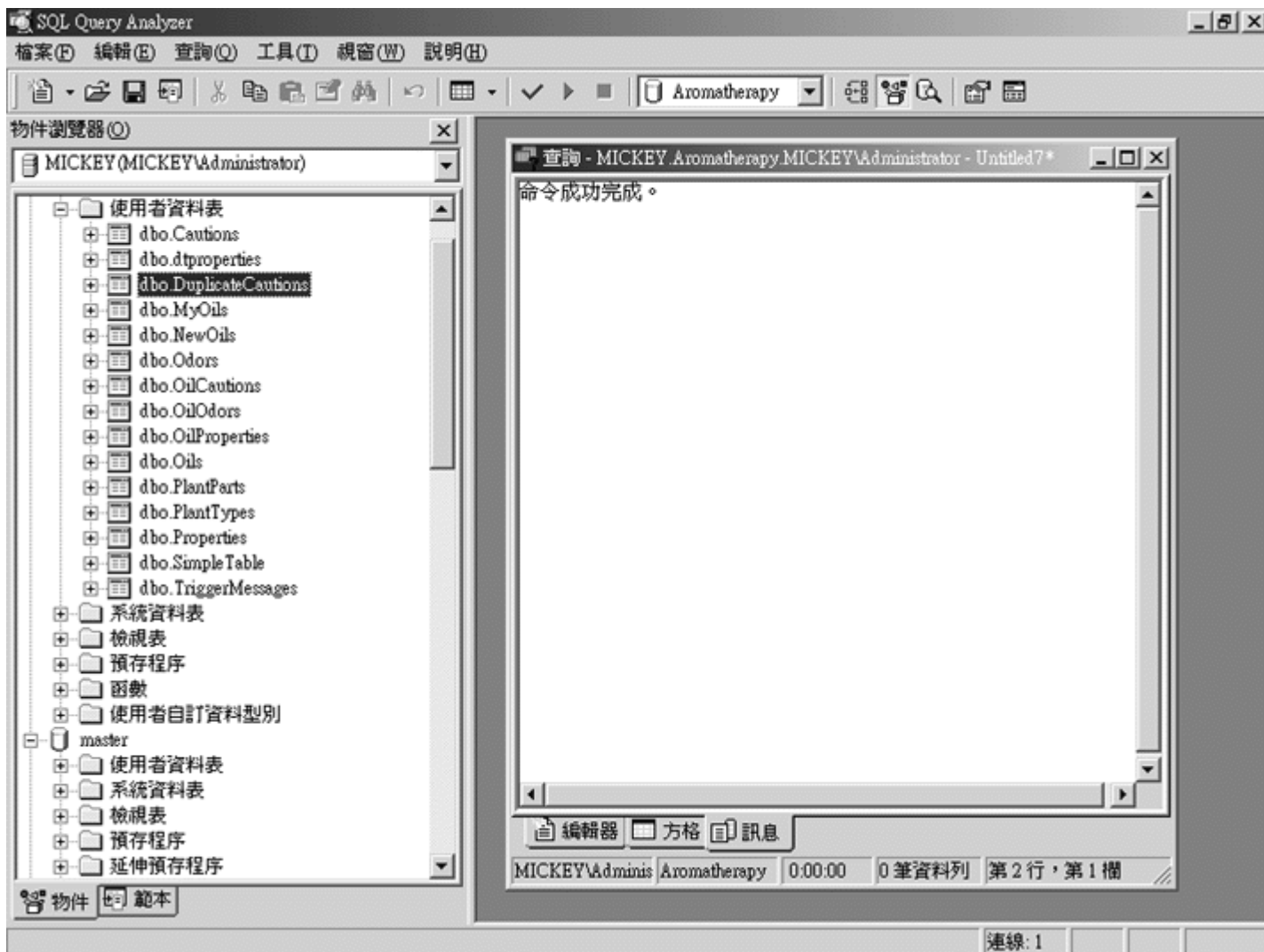


执行查询按钮

此时 Query Analyzer 会建立一新的数据表。

4. 在对象浏览器内的任何位置按一下，并且按下 **F5** 按钮以便重新显示。

对象浏览器会在列表中显示此新的 DuplicateCautions 资料表。



5. 关闭包含 CREATE 陈述式的查询窗口。

使用模板

SQL 语言与其它的程序语言不同—例如 C++或 Microsoft Visual Basic 等等，只有某些陈述式是有关系的，但是这些陈述式的语法十分复杂。事实上，在 SQL 语言中比较特殊的部份是所有的数据取得是使用单一的陈述式：**SELECT** 陈述式。

对于应付 SQL 陈述式的复杂性时，**范本**就是一种最佳的工具。当您使用 SQL 工作一段时间之后，您会发现您自己或多或少会使用标准的基本指令。

一旦这些陈述式建置为模板以后，单一的指令就会自动产生所有的模板文字，然后您可以使用一个适当的对话框来自订这些陈述式。

提示

模板不受限于单一的命令，它们可以拥有所有复杂的 SQL 指令码档案，也可以包含多行指令及多个批次。

模板不但具备强大的功能性，而且非常容易使用和建立，它们是由简单的 SQL 指令码档案（默认值）并且以 **.tql** 作为扩展名。您可以自订模板中的项目，例如以 CREATE TABLE 陈述式来建立数据行的名称以及数据表，并且定义其参数。在范本中，其参数的形式是

以 **<parameter_name、data_type、value>** 呈现。举例来说，下列的模板指令码包含二个参数：

table_name 和 **sort_column**。

```
SELECT *  
  
FROM <table_name, sysname, test_view>  
  
ORDER BY <sort_column, sysname, test_column>
```

该指令码是定义这二个参数的数据型别为 **sysname**，它是一种特殊的型别，是用来指示对象的名称。**table_name** 参数有一个默认值 **test_view**，而 **sort_column** 参数有一个默认值 **test_column**。

Query Analyzer 提供 **取代模板参数** 的对话框，以便简单的来自订模板文字。在查询窗口中简单地开启一个范本，并自 **编辑** 菜单中选取 **取代模板参数** 项目，以便开启此对话框。

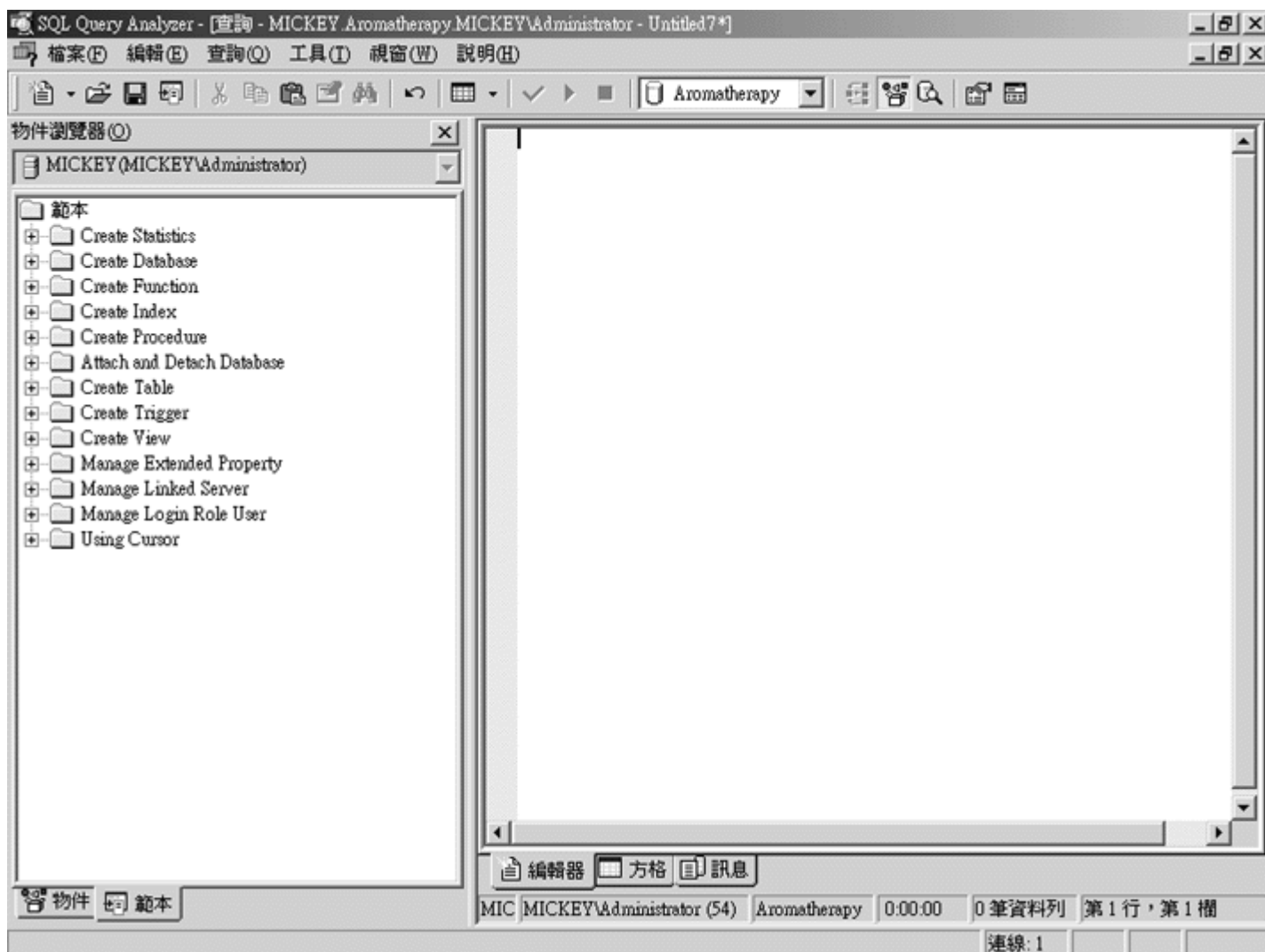
提示

您可以使用 **Ctrl-Shift-M** 快速键来开启 **取代模板参数** 的对话框。

使用模板来建立 **CREATE TABLE** 陈述式

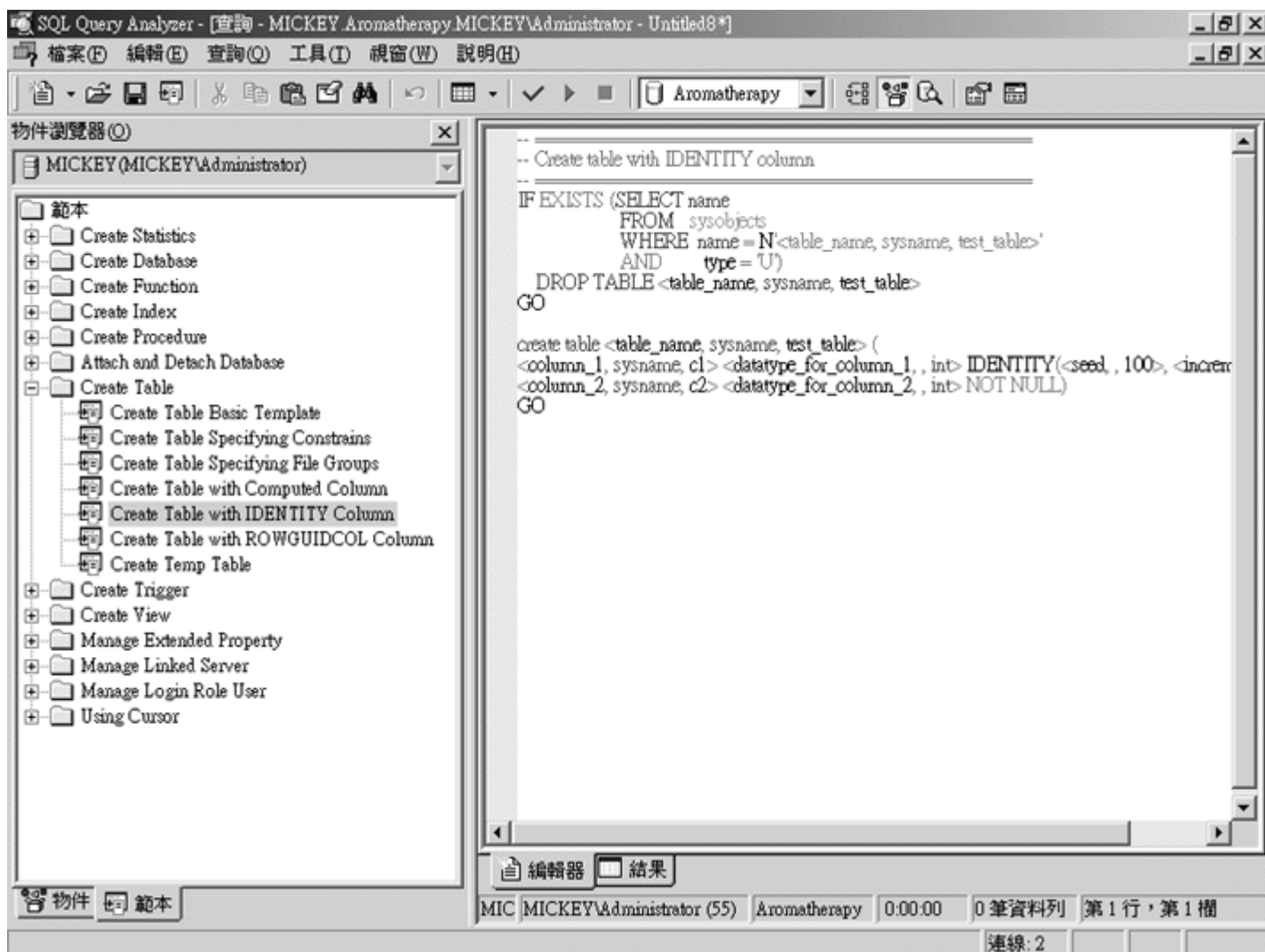
1. 在 **对象浏览器** 内选取 **模板** 卷标页。

此时 Query Analyzer 会显示一些可用模板的列表。



2. 將 **Create Table** 資料夾展開，并按两下 **Create Table with IDENTITY Column**。

此时 Query Analyzer 会开启一个插入模板文字的新查询窗口。



3. 自 **編輯** 菜单上选取 **取代模板参数** 项目。

此时 Query Analyzer 会开启 **取代模板参数** 对话框。

取代範本參數

參數	類型	值
table_name	sysname	test_table
column_1	sysname	c1
datatype_for_column_1		int
seed		100
increment		1
column_2	sysname	c2
datatype_for_column_2		int

全部取代(R)

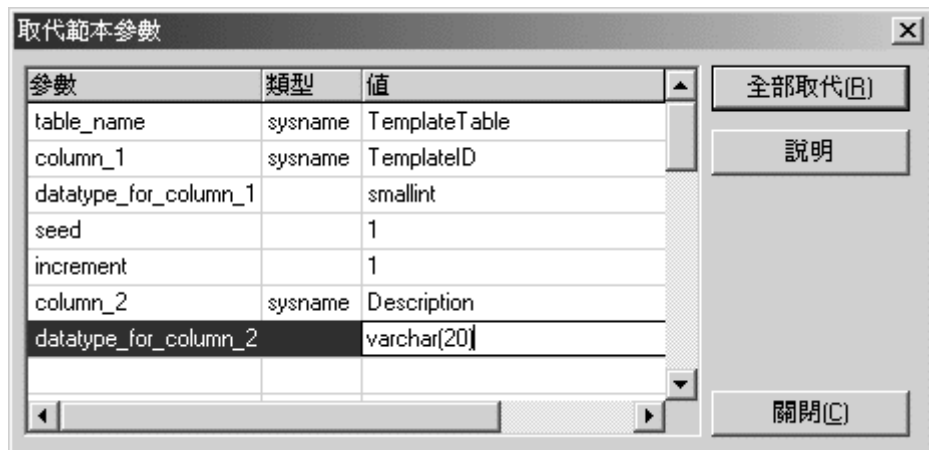
說明

關閉(C)

4. 設定下列參數值：

參數名稱	值
table_name	TemplateTable
column_1	TemplateID
datatype_for_column_1	Smallint
Seed	1
Increment	1
column_2	Description
datatype_for_column_2	varchar(20)

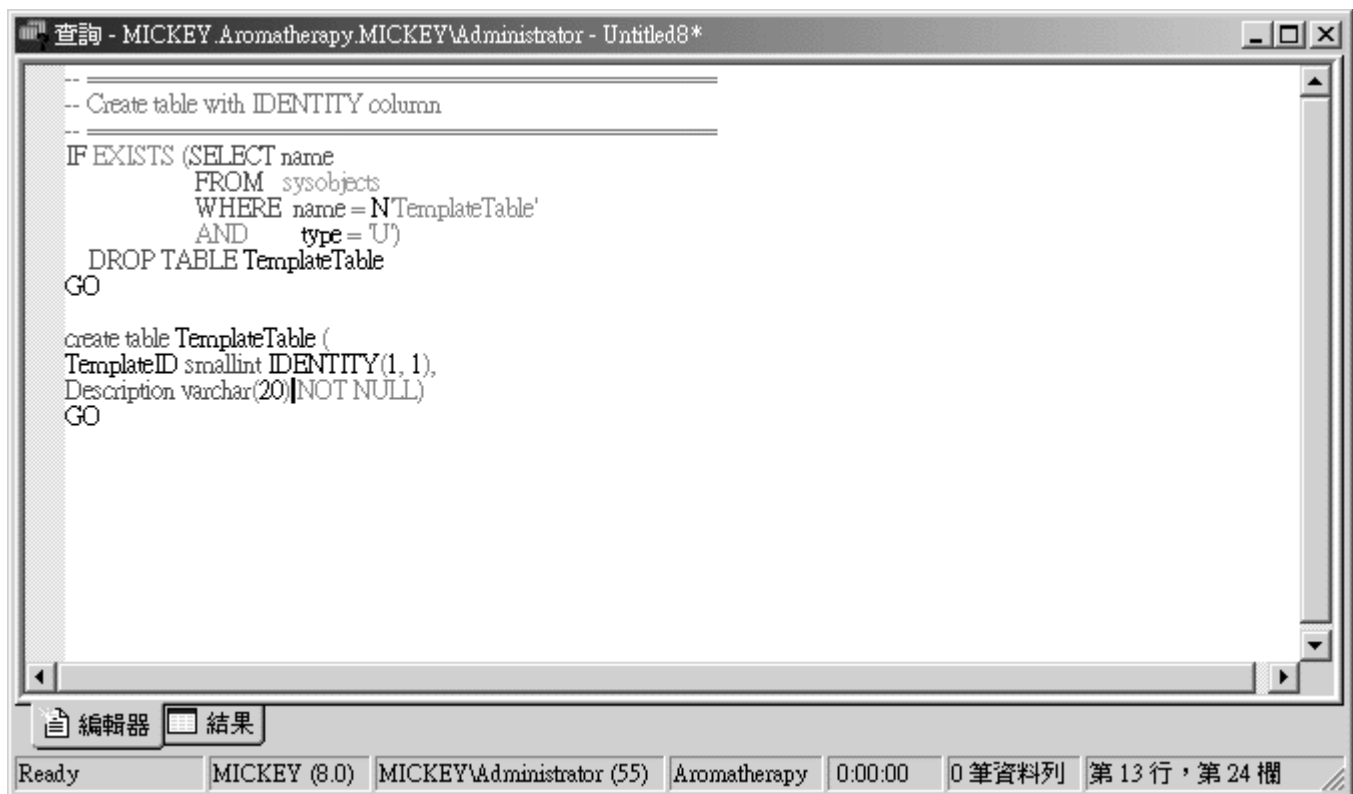
5.



6.

7. 按一下 **全部取代** 按钮。

此时 Query Analyzer 会关闭此对话框，并且依照您所定义的参数值将原先的参数值取代掉。



8. 在 Query Analyzer 工具列上确认 Aromatherapy 已经被选取。



9. 在 Query Analyzer 工具列上执行 [执行查询](#) 按钮，以便执行该陈述式。

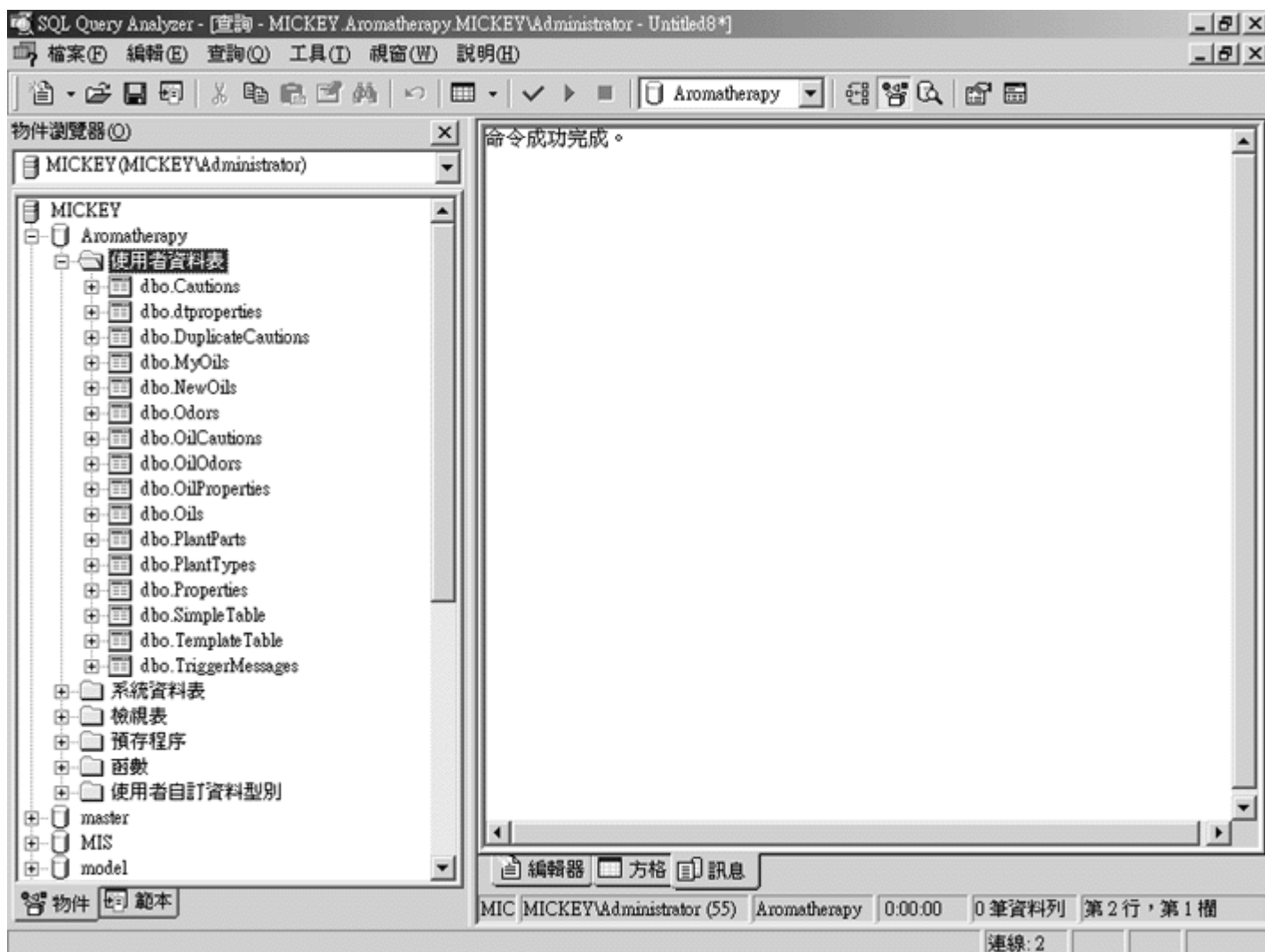


执行查询按钮

Query Analyzer 会建立此数据表。

10. 在 [对象浏览器](#) 内选取 [对象](#) 卷标页并展开 [使用者数据表](#) 数据夹，然后按下 **F5** 按键以便重新显示。

此时 [对象浏览器](#) 会在清单中显示此新的 TemplateTable 资料表。



11. 关闭包含 CREATE TABLE 陈述式内容的查询窗口。

本章总结

所要执行的工作	要执行..
建立数据库对象	使用 CREATE 陈述式（请参考表格 22-1 ）。

变更数据库对象	使用 ALTER 陈述式（请参考表格 22-2 ）。
移除数据库对象	使用 DROP 陈述式。
使用 对象浏览器 以建立 DLL 指令码	在 对象浏览器 内的某对象上按右钮，并且在 将对象转换为新窗口中的指令码 的子菜单中选取 Create 。
使用模板	在 对象浏览器 中的模板名称上按两下，并且在 编辑 菜单内选取 取代模板参数 对话框，以便取代模板中的参数。

23. 分析查询

在本章中，您将学习到：

- 显示 SQL 指令码的执行计划。
- 从执行计划窗格中变更数据库结构描述。
- 针对 SQL 指令码来显示服务器追踪。
- 针对 SQL 指令码来显示客户端统计资料。
- 使用索引微调精灵以最佳化数据库结构描述。

当一个数据库应用程序无法依照预期的效能执行时，有时候适当的处理方式就是升级实体结构—

即在服务器中加入更多的内存、新增更快的处理器或增加额外的处理器、或升级网络通讯的骨干。

但是有时候这并非最佳的处理方式，因为有时候是因为应用程序的本身出了问题，以及应用程序

中的查询执行出现了问题。在本章中，我们将详细说明 **Microsoft SQL Server** 所提供的一些针对

您的应用程序中的查询分析，以及最佳化的工具的用法。

使用 Query Analyzer 将执行效率最佳化

除了 [编辑器](#)、[方格](#) 以及 [讯息](#) 窗格之外，SQL Server Query Analyzer 查询窗口中提供了三个额外的显示方式来帮助您分析特定的查询执行效率。[执行计划](#) 窗格是以图形描述的方式来显示 SQL Server 执行查询的效率。[服务器追踪](#) 窗格会显示关于伺服器端执行查询的详细数据，这其中包括了执行的时间以及读写的数目。最后，[客户端统计数据](#) 窗格是用来显示客户端执行查询的详细数据，这其中包括了网络的容量以及服务器的来回执行的数目。

执行计划

当您欲执行查询时，查询窗口的 [执行计划](#) 窗格以图形化的方式呈现 SQL Server 将要执行的步骤。图 23-1 显示一个简单 SELECT 陈述式的执行计划。

```
SELECT OilName, LatinName  
  
FROM Oils  
  
ORDER BY LatinName
```

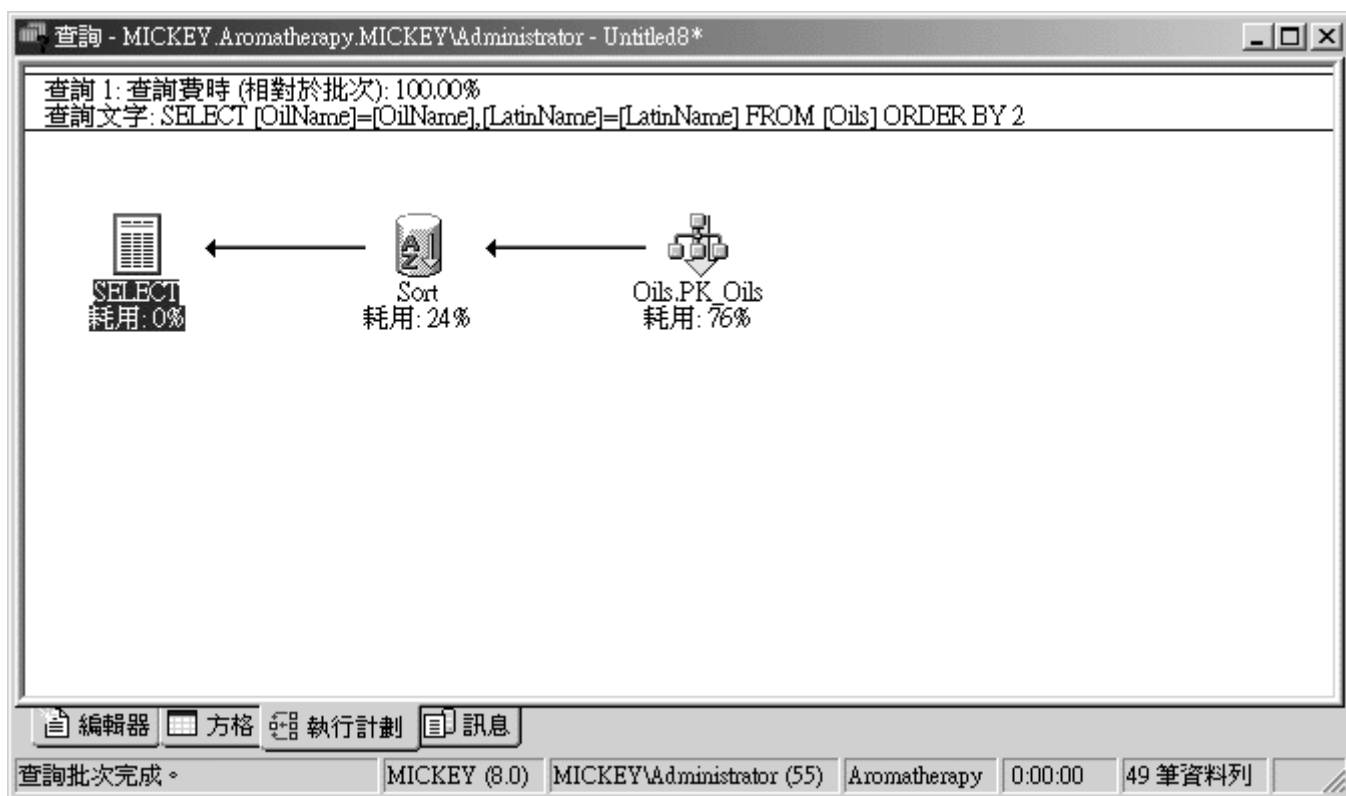


图 23-1 执行计划 窗格

提示

在 [执行计划](#) 窗格内所显示的信息与 SHOWPLAN 数据库选项所显示的文字模式内容是相同的。

在早期的 SQL Server 版本中，SHOWPLAN 数据库选项对许多使用者来说应该是很熟悉的，它也是 SQL Server 2000 中的一部份。如果您在查询窗口中将 SET SHOWPLAN_ALL ON 陈述式当成指令码的一部份时，SHOWPLAN 的执行结果将会在 [方格](#) 窗格中显示出来。然而，对于许多使用者来说，在 [执行计划](#) 窗格中会显示更多的分析信息。

执行计划窗格是使用大量的图示代表查询处理器执行的操作。这些图标在 [SQL Server 在线丛书](#) 中有记载，但是您并不需要去学习它。因为您只要将您的鼠标移到其中任何一个图示之后并保持不动，您就可以藉由 [工具提示](#)（ToolTip）所显示的内容来了解目前系统的执行状态等相关数据，这其中还包括了一些重要的统计资料，例如 I/O 耗用、CPU 耗用、预估的数据列数目以及总耗用值等。图 23-2 是显示 Clustered Index Scan 的工具提示内容。

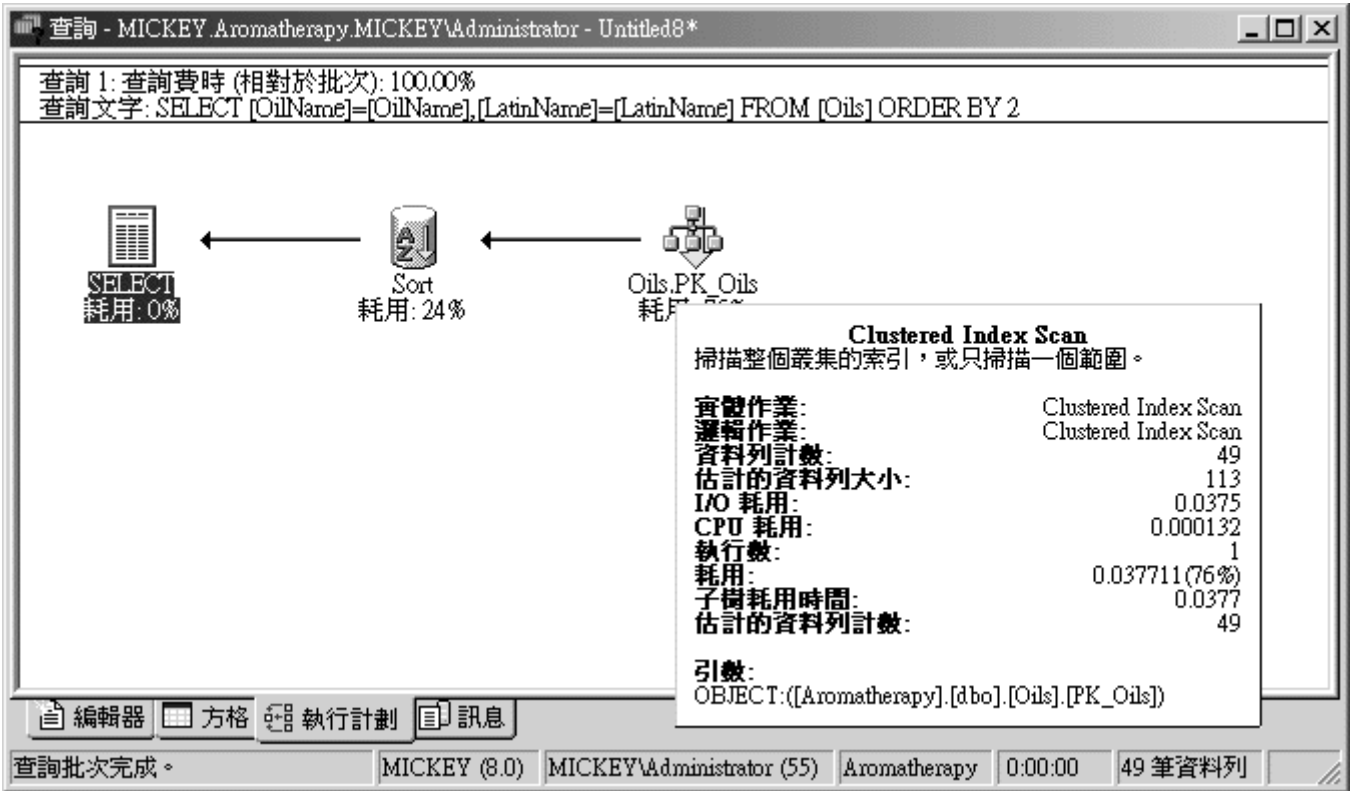


图 23-2 Clustered Index Scan 的工具提示内容。

在执行计划中的操作顺序是从右到左的方向来进行。连接每一个操作步骤中箭号的工具提示会显示从前一个操作步骤中传递的资料列，以及每一个数据列的估计大小。如图 23-3 所示。



图 23-3 连接箭头的工具提示内容。

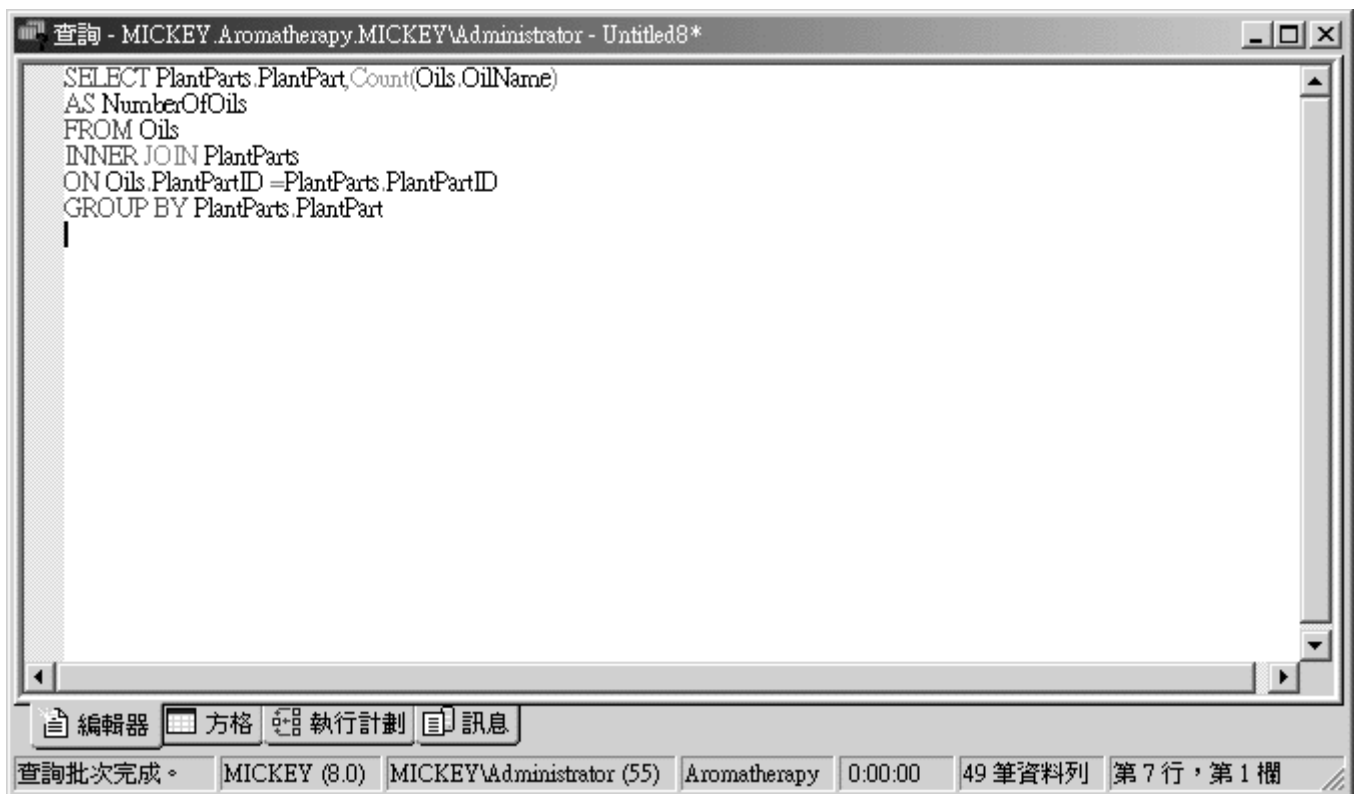
除了显示当您执行查询时 **SQL Server** 的操作步骤之外，执行计划也提供了最佳化查询的机制。

您可以使用 [执行计划](#) 的快捷菜单，以更新查询最佳化器用来决定查询执行策略的统计数据，并且新增索引以最佳化效能。

显示查询的执行计划

1. 在 **Query Analyzer** 的 [编辑器](#) 窗格内，输入如下所示的 **Transact-SQL** 陈述式：

```
2.      SELECT PlantParts.PlantPart, Count(Oils.OilName)
3.      AS NumberOfOils
4.      FROM Oils
5.      INNER JOIN PlantParts
6.      ON Oils.PlantPartID = PlantParts.PlantPartID
GROUP BY PlantParts.PlantPart
```



7. 从 [查询](#) 菜单中选取 [显示执行计划](#) 项目。

提示

除非执行查询，否则 [执行计划](#) 窗格并不会显示。

8. 在 Query Analyzer 的工具列上按一下 [执行查询](#) 按钮，以便执行此查询。
-

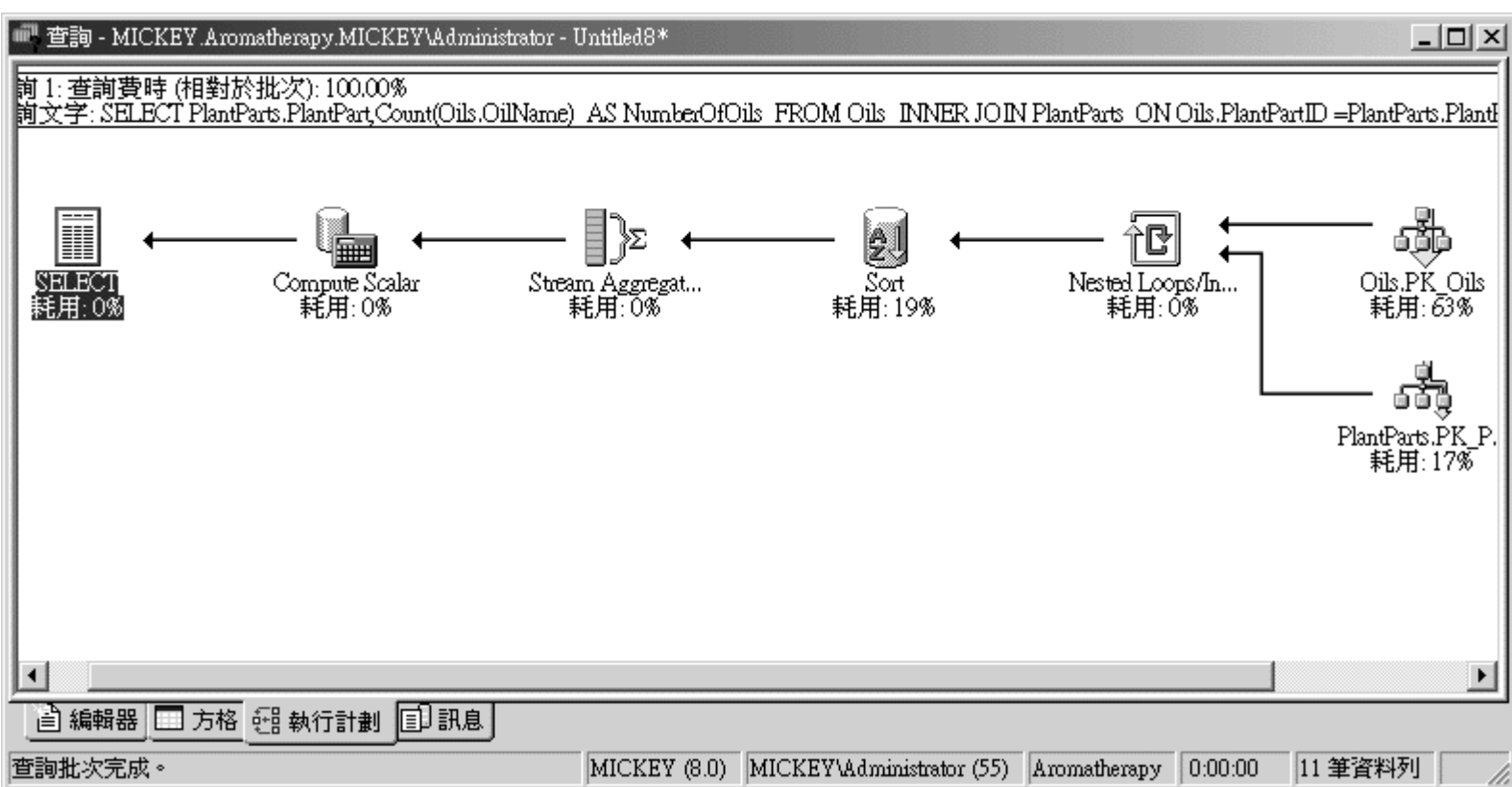


执行查询按钮

此时 Query Analyzer 会执行此查询并且在 [方格](#) 窗格内显示执行的结果。

	PlantPart	NumberOfOils
1	Berries	2
2	Flowers	11
3	Fruit Kemal	1
4	Fruit Peel	5
5	Fruit Seeds	1
6	Leaves	13
7	Needles	1
8	Resin	3
9	Seeds	2
10	Whole Plant	4
11	Wood	3

9. 选取 [执行计划](#) 标签页。



从执行计划窗格中新增索引

1. 在 [执行计划](#) 窗格中显示的 **Clustered Index Scan** 图示上按右钮。您可以发现在此

操作的耗用是百分之 63。如果可能的话，我们应该最佳化此查询。



Oils.PK_Oils Cost:63%

2. 从快捷菜单中选取 [管理索引](#) 项目。

此时 Query Analyzer 会显示 [管理索引](#) 的对话框。



3. 按一下 [新增](#) 按钮。

此时 Query Analyzer 会显示 [建立新的索引](#) 对话框。

建立新的索引 - MICKEY

在資料表 [dbo].[Oils][資料庫 [Aromatherapy]] 中建立新的索引。

索引名稱(I):

資料行	排序順序 (降...)	資料型別
<input type="checkbox"/> OilID		int
<input type="checkbox"/> OilName		nvarchar
<input type="checkbox"/> LatinName		nvarchar
<input type="checkbox"/> PlantTypeID		int
<input type="checkbox"/> PlantPartID		int
<input type="checkbox"/> Sample		char
<input type="checkbox"/> Description		char

變更資料行順序(C): 上移(U) 下移(D)

索引選項

☐ 叢集索引(L) ☐ 索引頁預留空間(P)

☐ 唯一值(N) ☐ 卸除現存的(B)

☐ 忽略重複值(G) ☐ 填滿因數(E): 80

☐ 不重新計算統計資料(不建議)(S)

☐ 檔案群組(Q): PRIMARY

編輯 SQL(E)... 確定 取消 說明

- 輸入『Oils_PlantParts』當作是索引的名稱，並且选取 **PlantPartID** 數據行來當作索引。

建立新的索引 - MICKEY

在資料表 [dbo].[Oils][資料庫 [Aromatherapy]] 中建立新的索引。

索引名稱(I):

資料行	排序順序 (降...)	資料型別
<input type="checkbox"/> OilID		int
<input type="checkbox"/> OilName		nvarchar
<input type="checkbox"/> LatinName		nvarchar
<input type="checkbox"/> PlantTypeID		int
<input checked="" type="checkbox"/> PlantPartID	<input type="checkbox"/>	int
<input type="checkbox"/> Sample		char
<input type="checkbox"/> Description		char

變更資料行順序(O):

索引選項

☐ 叢集索引(L) ☐ 索引頁預留空間(P)

☐ 唯一值(N) ☐ 卸除現存的(R)

☐ 忽略重複值(G) ☐ 填滿因數(E):

☐ 不重新計算統計資料(不建議)(S)

☐ 檔案群組(O):

5. 按一下 **確定** 按钮。

此时 Query Analyzer 会在 **管理索引** 对话框内建立此索引。



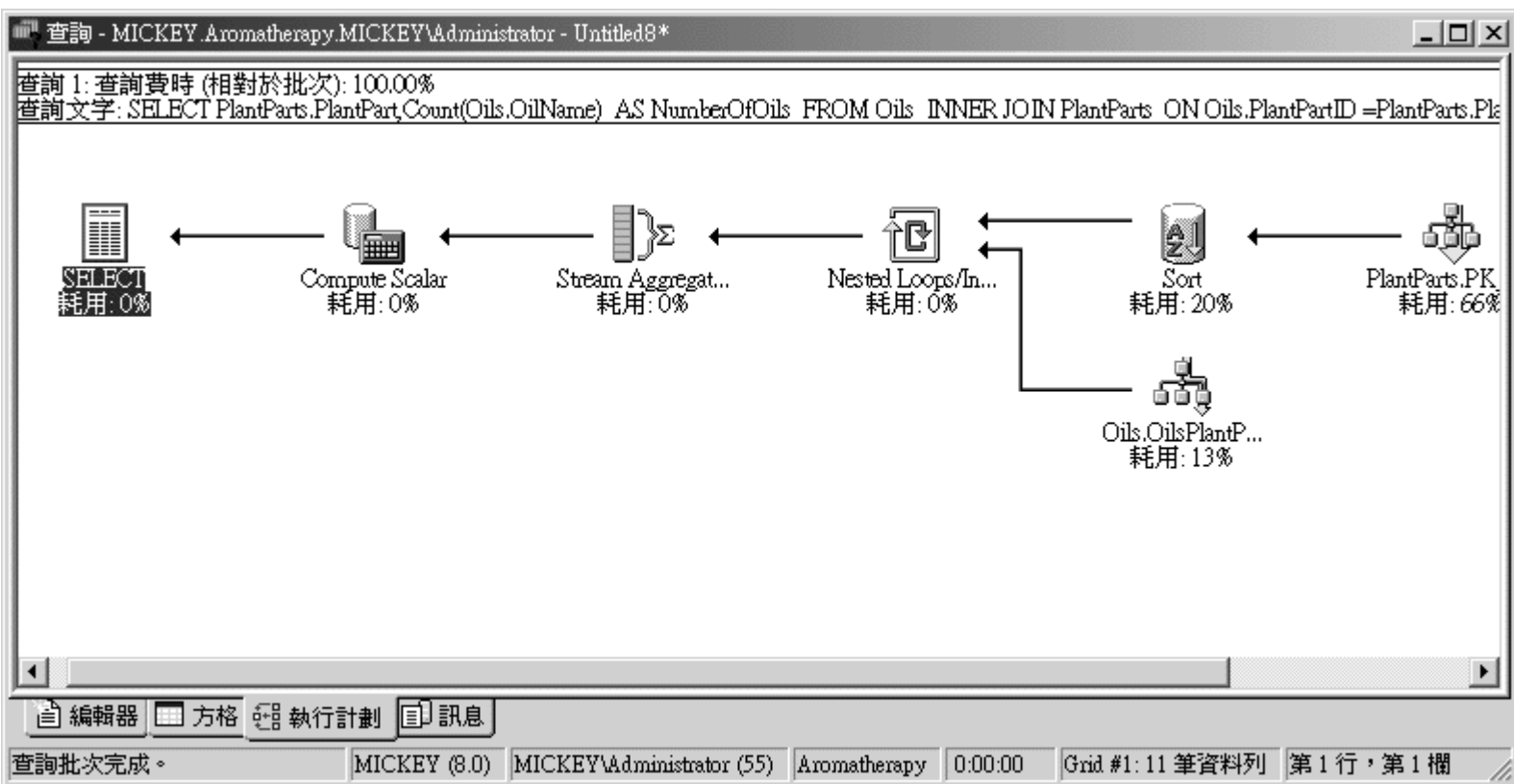
6. 关闭 [管理索引](#) 对话框。
7. 在 Query Analyzer 的工具列上按一下 [执行查询](#) 按钮，以便执行此查询。



执行查询按钮

8. 选取在查询窗口内的 **执行计划** 标签页，您可以发现在 **Oils** 数据表的 **Clustered**

Index Scan 已经由 **Index Seek** 取代，并且该操作的耗用从百分之 63 调整为百分之 13。



服务器的追踪

Query Analyzer 所提供的第二个工具是服务器追踪，它是用来提供分析查询的执行效率等相关

数据。 **追踪** 窗格会显示当查询执行时，服务器所执行的命令。

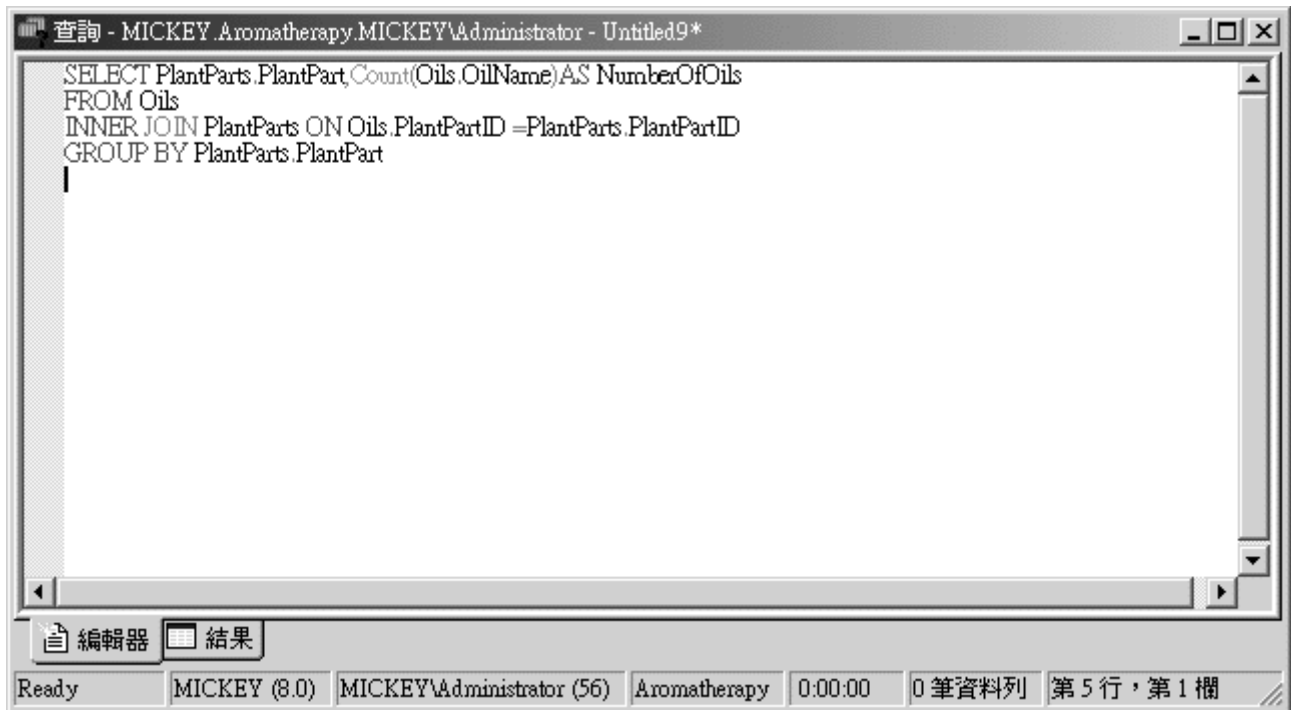
提示

SQL Server 2000 提供了另一种执行服务器追踪的工具，那就是 **SQL Profiler**。我们在本书中并不讨论 **SQL Profiler** 的功能。

显示服务器追踪

1. 假如您在前一个练习题中已经将查询窗口关闭时，您可以在 [编辑器](#) 窗格内开启另一个新的查询窗口，并且输入如下所示的 **Transact-SQL** 陈述式：

```
2.      SELECT PlantParts.PlantPart, Count(Oils.OilName) AS  
        NumberOfOils  
3.      FROM Oils  
4.      INNER JOIN PlantParts ON Oils.PlantPartID =  
        PlantParts.PlantPartID  
        GROUP BY PlantParts.PlantPart
```



5. 从 [查询](#) 菜单中选取 [显示服务器追踪](#) 项目。
6. 在 Query Analyzer 的工具列上按一下 [执行查询](#) 按钮，以便执行此查询。



执行查询按钮

7. 在查询窗口内选取 [追踪](#) 标签页。

查詢 - MICKEY.Aromatherapy.MICKEY\Administrator - Untitled9*						
Text	Event Class	Duration	CPU	Reads	Writes	
set noexec off set parseonly off	SQL:StmtCompleted	0	0	0	0	
select IS_SRVROLEMEMBER ('sysa...	SQL:StmtCompleted	0	0	0	0	
SELECT PlantParts.PlantPart,Count(Oi...	SQL:StmtCompleted	10	10	19	0	

編輯器 方格 追蹤 訊息

查詢批次完成 MICKEY (8.0) MICKEY\Administrator (56) Aromatherapy 0:00:00 3 筆資料列 第 0 行，第 1 欄

客户端统计资料

Query Analyzer 所提供的最后一个分析工具就是 [统计资料](#) 窗格，该 [统计资料](#) 窗格是放置在

Query Analyzer 的查询窗口内，它可以用来显示查询在客户端执行的影响。

在 [统计资料](#) 窗格内所传回的统计资料是分成 3 个区段：[应用程序设定统计数据](#) 会传回关于所

执行的 Transact-SQL 陈述式数量以及受影响的数据列信息；[网络统计数据](#) 会提供关于产生网

络流量的相关信息；[时间统计资料](#) 可以辅助您了解是否客户端或伺服器端产生延迟。

提示

如果您连接到本机服务器，[统计数据](#) 卷标页也一样会传回并显示 [网络统计数据](#)。

显示客户端统计资料

1. 假如您在前一个练习题中已经将查询窗口关闭，您可以在 [编辑器](#) 窗格内开启另一个新的查询窗口，并且输入如下所示的 **Transact-SQL** 陈述式：

```
2.      SELECT PlantParts.PlantPart, Count (Oils.OilName)
3.      AS NumberOfOils
4.      FROM Oils
5.      INNER JOIN PlantParts ON
6.      Oils.PlantPartID = PlantParts.PlantPartID
GROUP BY PlantParts.PlantPart
```



7. 从 [查詢](#) 菜单中选取 [显示客户端统计数据](#) 项目。
8. 在 Query Analyzer 的工具列上按一下 [执行查询](#) 按钮，以便执行此查询。



执行查询按钮

9. 在查询窗口中选取 [统计数据](#) 卷标页。

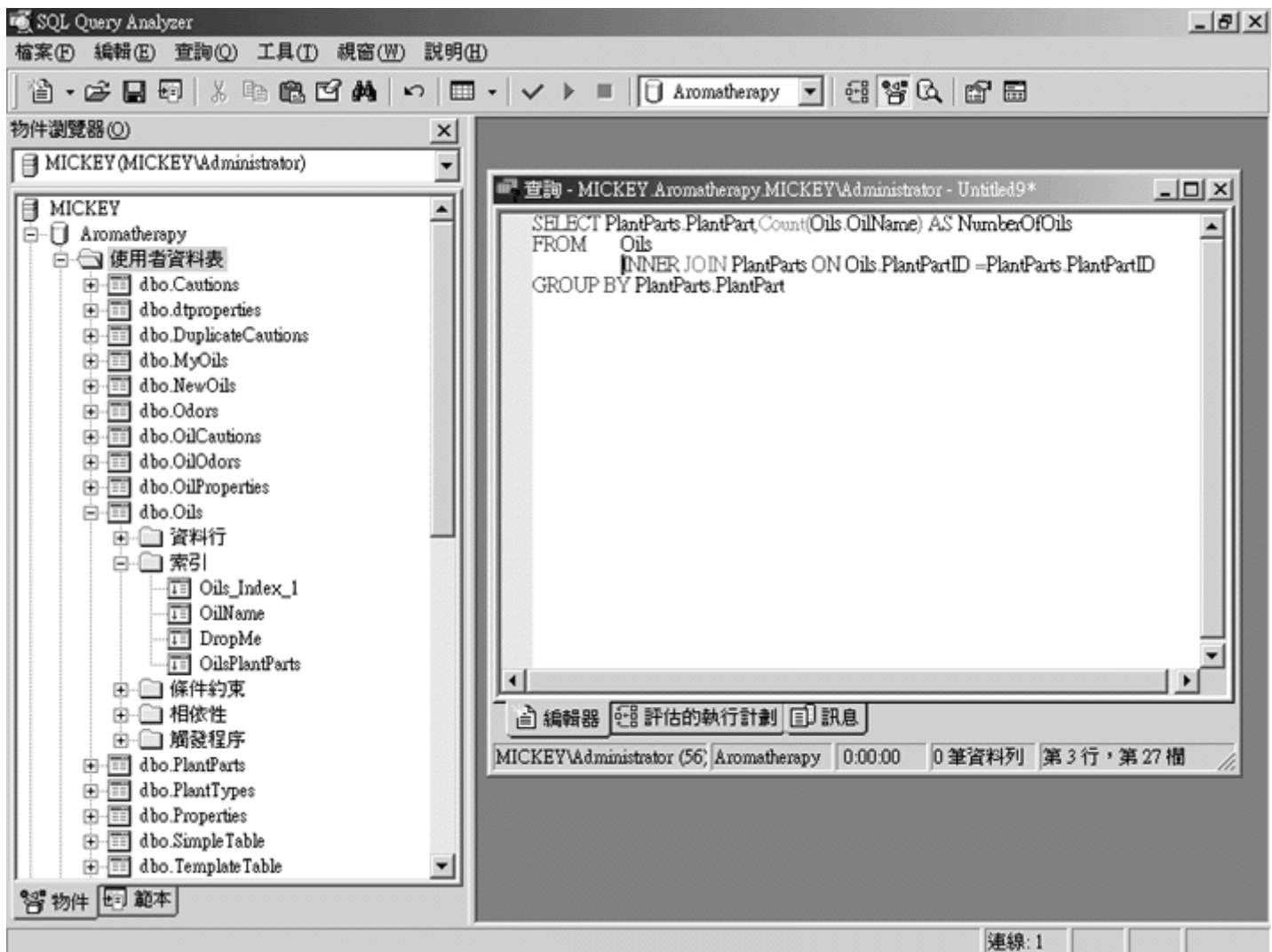
数据库进行微调时并不是独立作业，因为影响效能的因素不一定是特定的原因，因此我们需要联合特定的检视表和索引，并且执行特定的操作以取得最佳的效能。因为这个原因，索引微调精灵会要求您使用特定的 **工作负载**（workload），可以是 SQL 指令码或 SQL Profiler 的服务器追踪。

精灵的第二页会要求您指定被分析的服务器和数据库，并且提供两个额外的选项：**保留所有现存的索引** 以及 **微调模式**。**微调模式** 选项决定此精灵要进行分析效率的深度。清除 **保留所有现存的索引** 复选框（预设是选取），这样会使精灵完全利用工作负载进行建议，而原有的索引都不使用。您必须要很小心地使用该选项，因为纵使现存的索引不会被包含在工作负载中加以分析，但是这些现存的索引也许会在某些查询之下提供明显的执行效率。

一旦此精灵完成分析，您就可以选择立即建置该精灵建议的选项、设定未来建置的日期和时间，或者将建议转换成 SQL 指令码以供稍后建置。

使用索引微调精灵来微调数据库

1. 展开 Aromatherapy 数据库中 Oils 数据表的 **索引** 数据夹。



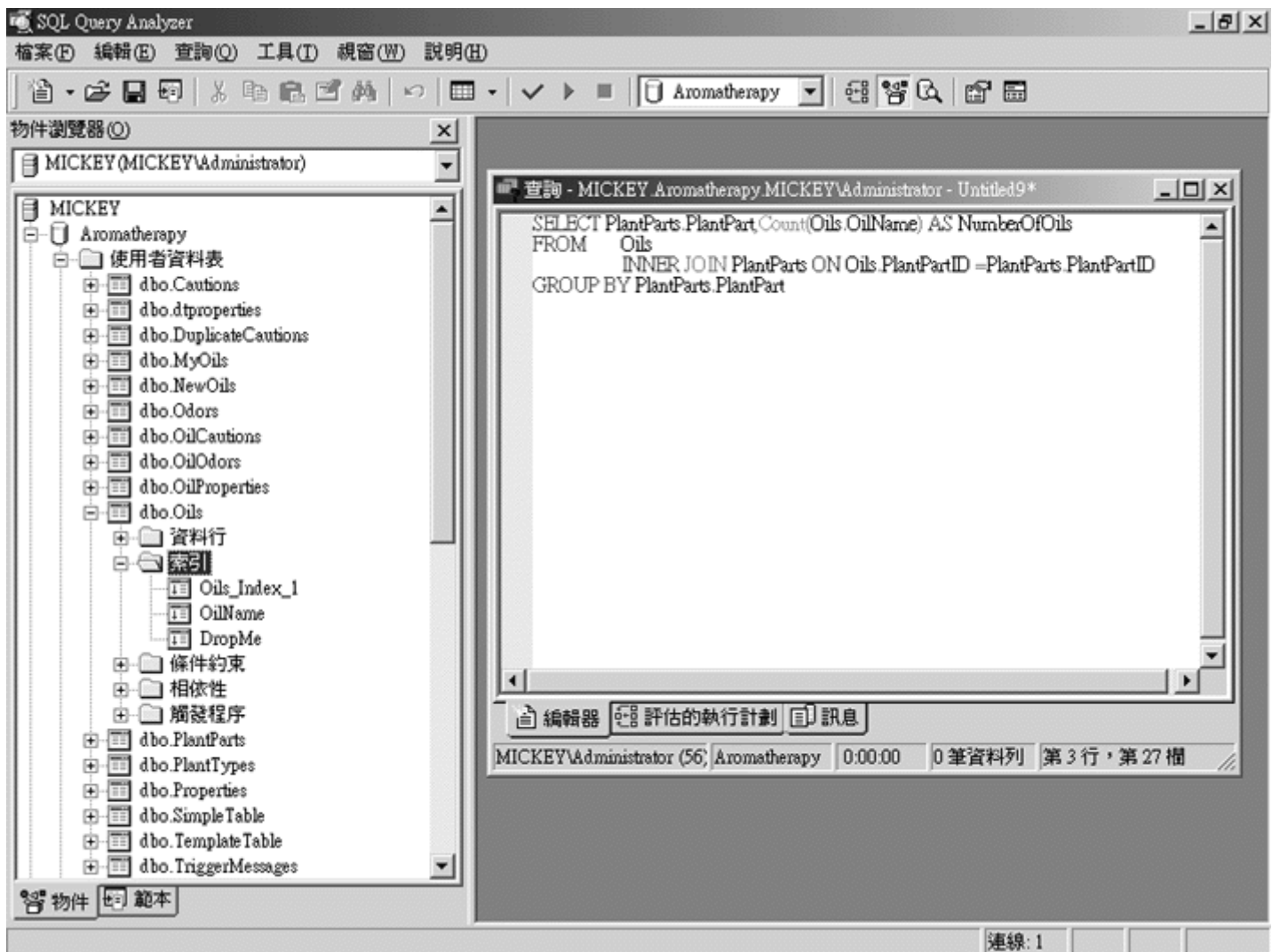
2. 选取 **Oil_PlantParts** 索引，然后按一下 **Delete** 键。

此时 Query Analyzer 会询问您是否确定要删除此索引。



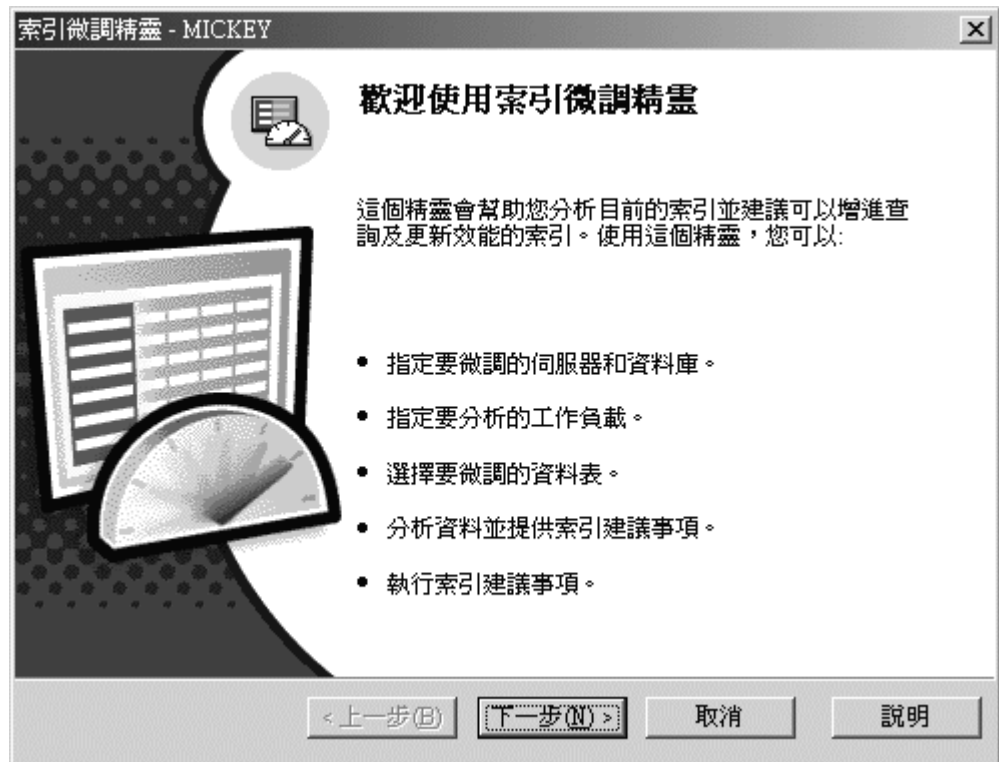
3. 按一下 **确定** 按钮。

此时 Query Analyzer 会把此索引删除。



4. 选取查询窗口，并且从 [查询](#) 菜单中选取 [索引微调精灵](#) 项目。

此时 Query Analyzer 会显示 [索引微调精灵](#) 的第一个画面。



5. 按一下 **下一步** 按钮。

此精灵会要求您指定要进行微调的数据库名称及微调的模式。

索引微調精靈 - MICKEY

選擇伺服器 and 資料庫

在精靈分析資料之前，您必須選擇要微調的伺服器和資料庫。

伺服器(S): MICKEY

資料庫(D): Aromatherapy

☒ 保留所有現存的索引(K)
選擇此選項來微調新的或有問題的查詢工作負載。取消選取此選項將會提供所微調的工作負載之最大效能增進。如果您選擇不保留現有的索引，索引可能會被卸除或取代。

☒ 新增索引檢視表(V)

微調模式
越完整的分析需要越多的微調時間，但是將會產生較完善的索引

☐ 快速(F) ☒ 適中(M) ☐ 完整(I)

< 上一步(B) 下一步(N) > 取消 說明

6. 请您确定已经选取 Aromatherapy 数据库，并且选取 [完整](#) 的微调模式。

索引微調精靈 - MICKEY

選擇伺服器 and 資料庫
在精靈分析資料之前，您必須選擇要微調的伺服器和資料庫。

伺服器(S): MICKEY

資料庫(D): Aromatherapy

☒ 保留所有現存的索引(K)
選擇此選項來微調新的或有問題的查詢工作負載。取消選取此選項將會提供所微調的工作負載之最大效能增進。如果您選擇不保留現有的索引，索引可能會被卸除或取代。

☒ 新增索引檢視表(V)

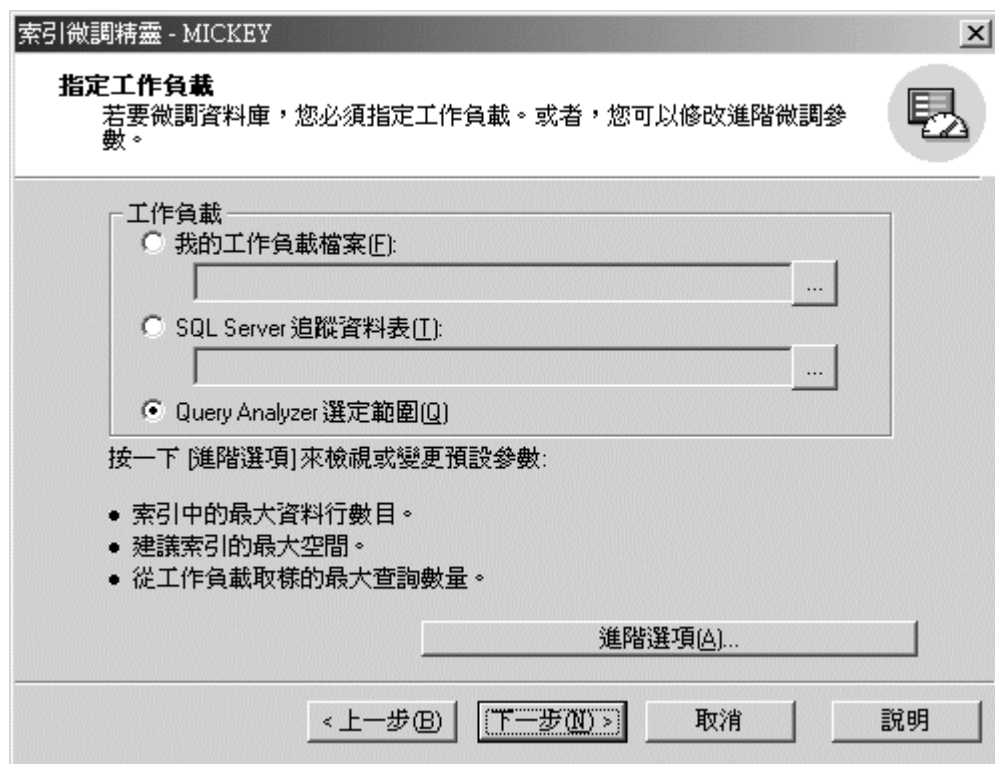
微調模式
越完整的分析需要越多的微調時間，但是將會產生較完善的索引

☐ 快速(F) ☐ 適中(M) ☒ 完整(T)

< 上一步(B) 下一步(N) > 取消 說明

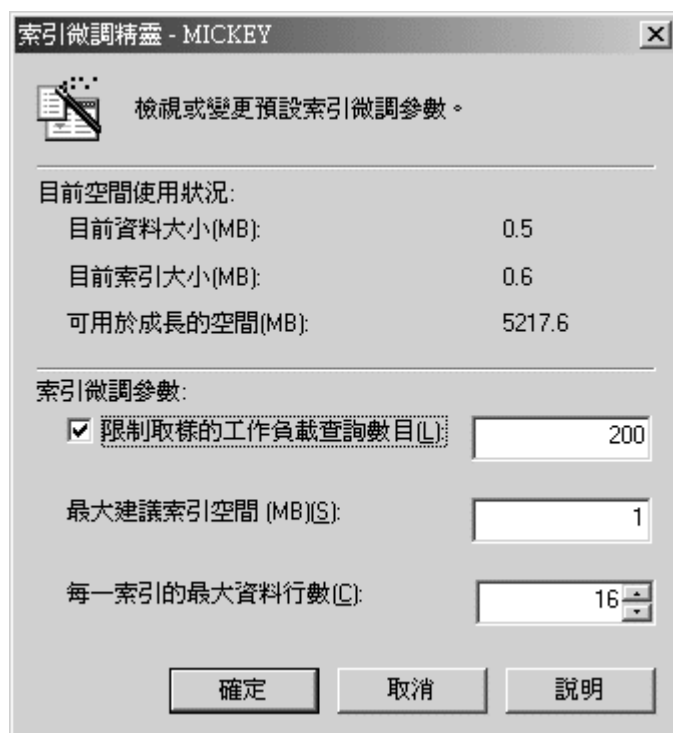
7. 按一下 **下一步** 按钮。

此精灵会要求您指定要进行微调的工作负载。



8. 按一下 [进阶选项](#) 按钮。

此精灵会显示一个要求您指定微调参数的对话框。



9. 采用预设的微调参数值，并且按一下 **确定** 按钮。

10. 采用预设的 **Query Analyzer 选定范围** 项目，并且按下 **下一步** 按钮。

此时精灵会要求您指定要进行微调的数据表。



11. 请选取 **Oils** 数据表来进行微调，并且将 **预计的数据列** 的值设定为 **1000**。

索引微調精靈 - MICKEY

選擇要微調的資料表
選擇精靈所要微調的資料表。

這是精靈可以分析的所有資料表清單。這項分析常常是很費時的。若要減少持續時間，請選擇最重要的資料表進行分析。透過變更所預計的資料列數目，您可以得知資料庫未來成長的大小。

資料表	實際的資料列	預計的資料列
<input type="checkbox"/> [dbo].[Odors]	35	35
<input type="checkbox"/> [dbo].[OilCautions]	25	25
<input type="checkbox"/> [dbo].[OilOdors]	128	128
<input type="checkbox"/> [dbo].[OilProperties]	224	224
<input checked="" type="checkbox"/> [dbo].[Oils]	49	1000
<input type="checkbox"/> [dbo].[OilTypes]	10	10

選取所有資料表(A) 取消選取所有資料表(C)

< 上一步(B) 下一步(N) > 取消 說明

12. 按一下 **下一步** 按钮。

此精灵会显示它所建议的索引。



13. 按一下 [下一步](#) 按钮。

此精灵会显示一个要求您指定要执行的选项。

索引微調精靈 - MICKEY

排程索引更新作業

現在執行建議事項、排程稍後要執行的作業及(或)將建議事項儲存為指令碼。

☐ **套用變更(A)**

套用變更可能需要一些時間。您可以選擇立刻套用變更，或是排程執行時間。

☐ 立刻執行建議事項(E)

☐ 排程要執行建議事項的時間(S)

日期(D): 2001 / 2 / 10

時間(T): 上午 12:00:00

☐ **儲存指令碼檔案(E)**

...

如果您選擇不保留所有現存的索引，則現存索引可能會被卸除或取代。

< 上一步(B) 下一步(N) > 取消 說明

14. 請选取 **套用變更** 項目，並採用預設選擇項目 **立即執行建議事項**。

索引微調精靈 - MICKEY

排程索引更新作業

現在執行建議事項、排程稍後要執行的作業及(或)將建議事項儲存為指令碼。

☒ 套用變更(A)

套用變更可能需要一些時間。您可以選擇立刻套用變更，或是排程執行時間。

☒ 立刻執行建議事項(E)

☐ 排程要執行建議事項的時間(S)

日期(D): 2001 / 2 / 10

時間(T): 上午 12:00:00

☐ 儲存指令碼檔案(E)

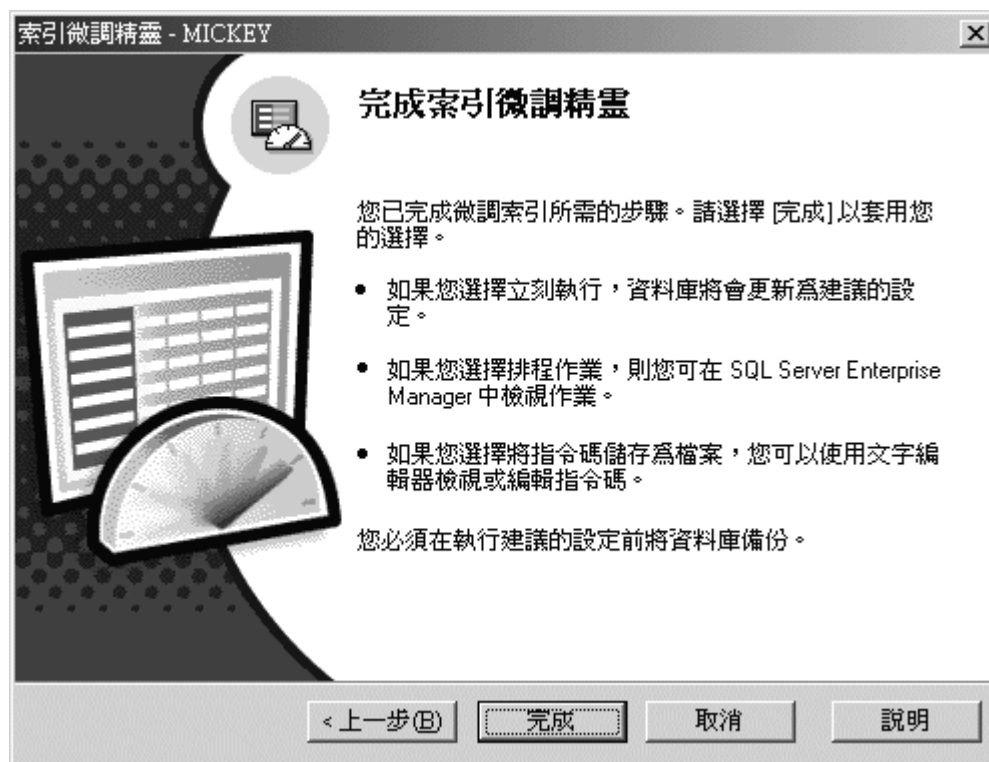
...

如果您選擇不保留所有現存的索引，則現存索引可能會被卸除或取代。

< 上一步(B) 下一步(N) > 取消 說明

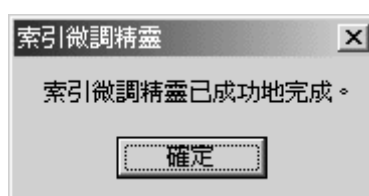
15. 按一下 **下一步** 按钮。

此精灵会显示一个您已经选取项目内容的最后画面。



16. 按一下 **完成** 按钮。

此精灵会显示一个告知您 **索引微調精靈已經成功地完成** 的讯息。



本章总结

要执行的工作	执行...	按钮
显示 SQL 指令码的执行计划	从 查询 菜单中选取 显示执行计划 项目，并且执行查询。	
从 执行计划 窗格来变更数据库结构描述	在执行计划中按右钮，并选取 管理索引 项目。	
显示 SQL 指令码的服务器追踪	从 查询菜单 中选取 显示服务器追踪 项目，并执行查询。	
显示 SQL 指令码的客户端统计数据	从 查询 菜单中选取 显示客户端统计数据 项目，并执行查询。	
使用索引微调精灵最佳化数据库结构描述	从 查询 菜单中选取 索引微调 精灵 项目，并按照所提示的步骤来执行。	

24. Transact-SQL 语言的组件

在本章中，您将学习到：

- 在 **SELECT** 陈述式中使用算术运算符。
- 在 **WHERE** 子句中使用比较运算符。
- 在 **SELECT** 陈述式中使用逻辑运算符。
- 在 **SELECT** 陈述式中使用位运算符。
- 在 **SELECT** 陈述式中使用串连运算符。
- 使用日期函数。
- 使用数学函数。
- 使用汇总函数。

- 使用中继数据函数。
- 使用安全性函数。
- 使用字符串函数。
- 使用系统函数。

就像其它的程序语言一样，Transact-SQL 程序是由一系列的陈述式所组成。所谓的 **陈述**

式（statement）是指一个 **指令**，它可以用来描述某些 Microsoft SQL Server 要执行的动作。

一句陈述式是由一些指令、表达式、函数、运算子以及符号等的组合。陈述式也可以十分的简单，

例如只包含一行 GO 指令。而 GO 指令在指令码中也用以表示一个批次操作的结束。在本章中，

我们将详细说明 Transact-SQL 陈述式的组件。

Transact-SQL 命令

Transact-SQL 语言最核心的部分是它的命令：核心的陈述式命令定义了该语言可以执行的基本工作。

保留字

所谓的 **保留字**（Reserved Word）是给该语言使用的。假如您将保留字当作一般陈述式使用时，例如一个数据行的名称，那么您必须以特殊字符包含该保留字，这些特殊的符号就称为 **分隔符**（delimiter）。在 Microsoft SQL Server 中，这些分隔符是 **[** 和 **]**。举例来说，假如您使用 **SELECT** 这个字当作是数据行名称时，您必须要在查询中以 **SELECT** 来表示它，这样 SQL Server 就会将其视为一个识别项（您必须尽量避免将保留字当作识别项来使用）。

如果我们以复杂的应用程序观点来看，其实搭配 Transact-SQL 命令来撰写的应用程序，复杂度已经降低了许多。Transact-SQL 包含了许多实用的命令以降低应用程序撰写的复杂程度，其中有许多命令我们早就知道了。

假如您依照 Transact-SQL 的功能性来划分，您就会发现认识 Transact-SQL 命令并不是难事。

例如在数据表或检视表中操作数据，或者是控制 SQL Server 作业环境等等。

本书已经在各种不同的情况之下使用 Transact-SQL 指令来执行某些操作，并且我们也在 Query Analyzer 查询窗口内的编辑器，以及 Enterprise Manager 的查询设计师中 SQL 窗格中直接输入 Transact-SQL 陈述式，同时我们也透过操作一些可以执行 Transact-SQL 命令的工具来间接使用它们。举例来说，Enterprise Manager 的数据表设计师，是使用 CREATE 和 ALTER 陈述式来完成您的需求。

与 SQL Server 沟通

许多数据库应用程序是使用一般的程序语言，例如使用 **Microsoft Visual Basic** 来建立一个可与 **SQL Server** 建立互动关系的接口程序。使用这些语言所提供的接口控制方式，让这些应用程序可以用 **方便** 和 **友善** 的方式将数据呈现给使用者。然而，在将数据提供给使用者的过程中，您同时也必须利用 **Transact-SQL** 指令来完成提取数据的动作。例如 **Enterprise Manager** 和 **Query Analyzer** 就是这类型的数据库应用程序。

当您使用一般的程序语言时，语言的本身将决定如何执行指令。在某些环境之下，例如 **Microsoft Access**，也会提供一个类似于 **Enterprise Manager** 以及 **Query Analyzer** 的交互式工具。其它例如 **Visual Basic** 或 **Microsoft Visual C++** 等等，即是使用 **ADO** 的对象模块来与服务器进行通讯。

数据操作命令

大多数重要的 **SQL** 命令是归类于我们在第二及三部份已经讨论过的数据操作语言 (**DML**)。**DML** 命令是用于插入、更改、移除，以及取得数据。

表 24-1 是一般我们最常使用到的 DML 命令清单，其中大多数的内容在本书的第三及四部份最常使用。下表中只有 **BULK INSERT** 命令我们尚未讨论过，它可以让您从数据文件中插入多列的数据列至数据库的数据表；**USE** 命令是用来在 **SQL** 指令码中指定将会使用的数据库。

命令	功能
INSERT	将数据列插入到数据表或检视表中。
UPDATE	在数据表或检视表中更新数据列。
DELETE	在数据表或检视表中删除数据列。
SELECT	从数据表或检视表中选取数据列。
TRUNCATE TABLE	在数据表或检视表中移除所有的数据列。
BULK INSERT	从数据文件中插入多列的数据列至数据库的数据表或检视表中。
USE	连接至某数据库。

表 24-1 DML 命令

数据定义命令

一般我们最常使用的数据定义语言（DDL）都显示在表 24-2 中。DDL 命令是使用在建立、更改以及移除数据库对象。它们只有三个主要的命令，这三个命令我们已经在第 22 章中看过，每一个命令都有一些变量，这些变量的使用方式是取决于您欲建立的数据库对象而定。

命令	功能
CREATE	定义一个新的数据库对象。
ALTER	变更数据库对象的定义。
DROP	从数据库中移除一个数据库对象。

表 24-2 DDL 命令

数据库管理命令

大多数的 Transact-SQL 命令支持数据库管理工作，以便可以与 Enterprise Manager 建立互动性。

管理命令提供了以程序化的方式执行管理工作的能力。

表 24-3 为一般我们最常用到的数据管理命令。第一组 GRANT、DENY 及 REVOKE 是用来控制数据库的安全性。BACKUP、RESTORE 以及 UPDATE STATISTICS 命令与 Enterprise Manager 的数据库维护计划工具有相同的功能。

SET 命令是与 DATEFORMAT 和 LANGUAGE 等关键词搭配使用，它们是用以控制 SQL Server 目前工作阶段的一些行为。在 Enterprise Manager 中，大多数的变量设定可以由 数据库属性 对话框来设定。

最后二个数据库管理命令 KILL 和 SHUTDOWN 是用来控制 SQL Server 的执行方式。KILL 命令会终止特定使用者联机的相关处理序；SHUTDOWN 命令是用来无条件的终止 SQL Server 的执行。

命令	功能
GRANT	授与特定的权限给某个对象。
DENY	拒绝授与某个对象的权限，并且同时禁止该对象继承其群组或角色成员的权限。
REVOKE	撤销特定的权限给某个对象。
BACKUP	备份数据库或交易记录文件。
RESTORE	从之前的备份还原数据。
UPDATE STATISTICS	重新整理查询处理器所使用的统计资料。
SET	控制 SQL Server 环境。
KILL	终止联机以及其相关联处理序。
SHUTDOWN	无条件地停止 SQL Server。

表 24-3 数据库管理命令

其它的命令

其它的 Transact-SQL 命令有三类。第一类是用来控制程序变量的使用，我们将在第 25 章中说明。

第二类命令是用来在 SQL 指令码档案中控制陈述式执行的流程控制命令，我们将在第 26 章中说明。另一类数据指针控制命令是用来控制特定对象的行为，例如数据指针是指在数据表或检视表中来指向记录的指针，我们将在第 27 章中说明。

Transact-SQL 运算符

运算符其实是一种符号，可以让 SQL Server 执行特定的动作。我们在第 12 章中有使用过串连运算符（+）。

Transact-SQL 运算符是依据它们所操作的值加以分类，这就是我们都知道的运算符 **基数**（cardinality）。Transact-SQL 运算符有二种基数：二元（binary）或一元（unary）。

大多数的运算符是二元的，如果某个运算符操作二个值时，该运算符就是二元的。在 $4 + 3$ 的表达式内的 $+$ 运算符以及 $\text{MonthSales} < \text{MonthBuget}$ 表达式内的 $<$ 运算符，都是使用二元运算符；如果某运算符只操作单一值时，它就是一元的。在 -10 表达式内的负号（-）就是一元运算符。

运算符的优先级

当我们建立复杂的 Transact-SQL 陈述式时，我们必须了解在表达式中运算符的执行优先级。不是说执行的优先级在执行的过程中会发生问题，但它有时却会造成您的困扰。举例来说， $3 * (4 + 1)$ 表达式其执行结果为 15，而 $3 * 4 + 1$ 表达式其结果为 13，这是因为乘法运算符的执行优先级比加法还要前面，因此乘法运算符拥有比较高的执行优先级。

运算符的执行优先级如下所示，假如在一个表达式内有相同优先级的运算符时，其执行顺序将是自左到右依序执行。

1. +（正数），-（负数），~（NOT 位运算符）
2. *, /, %
3. +（加号），+（串连），-（减号）
4. =（比较运算符），>, <, >=, <=, <>
5. ^, &, |
6. NOT
7. AND
8. OR
9. =（指派运算符）

通常您可以使用括号来控制执行的优先级，例如同前一个范例一样。

就像 Transact-SQL 命令一样，运算符依照它们的型态来进行分类会比较容易了解。

批注运算符

Transact-SQL 支持二种特殊的运算符，更确切的说法是指定一种运算符来告诉 SQL Server 在此指令码内所指定的内容可以忽略不执行。Transact-SQL 支持二种批注运算符，二个破折号 (--) 是用来指示 SQL Server 在它后面的那行文字内容可以忽略。它可以使用在每一行的前面，这样会告诉 SQL Server 要忽略整行的文字内容，或者是使用在每一行的里面，这样会告诉 SQL Server 要忽略自 (--) 符号开始的位置到行尾内的内容。

其它的批注运算符是 /* 和 */，该运算符是一对的。SQL Server 会将第一个批注运算符 /* 起的位置，到第二个批注运算符 */ 为止，将这二个运算符所围住的内容忽略。

图 24-1 所示为批注运算符的使用范例。

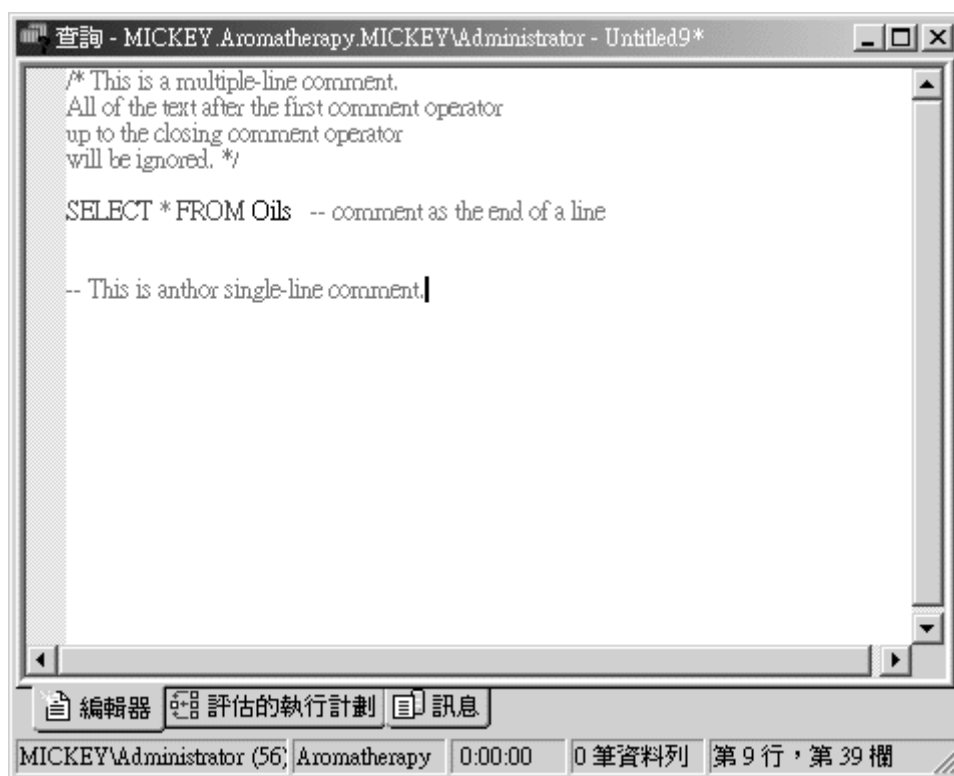


图 24-1 Transact-SQL 支持二种批注运算符

提示

当您进行程序除错时，有一些有问题的 Transact-SQL 陈述式，您可以使用 `/*` 和 `*/` 将该 Transact-SQL 陈述式前后括住，这种程序除错方式是非常好用的。

算术运算符

Transact-SQL 提供基本算术方面的运算符，如表 24-4 所示。这其中比较不常使用的算术运算符是求余数，它会传回一个整数余数，举例来说，16%3 的求余数结果是 1，而不是 5 又 3 分之 1。

表 24-4 算术运算符

运算符	意义
+	加法。
-	减法。
*	乘法。
/	除法。
%	求余数。
+	正的数值。
-	负的数值。

在 SELECT 陈述式中使用算术运算符

- 1. 在 Query Analyzer 工具列上按一下 新增查询 按钮，以便开启一个新的 查询 窗口。



新增查询按钮

此时 Query Analyzer 会开启一个空白的 [查询](#) 窗口。

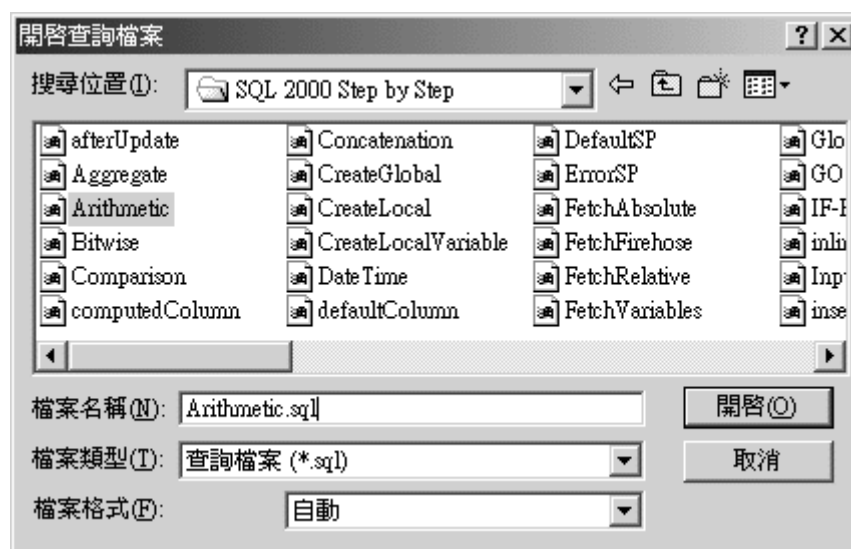


2. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。
-



加载 SQL 指令码按钮

Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



3. 将档案目录位置移至 [SQL 2000 Step by Step](#) 的数据夹中，并按一下 [开启](#) 按钮。

Query Analyzer 会将指令码档案加载到 [查询](#) 窗口中。



4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕，以便執行此查詢。



執行查詢按鈕

Query Analyzer 会在 [方格窗格](#) 中显示此查询的执行结果。



5. 关闭此 [查询](#) 窗口。

比较运算符

我们在第 13 章中曾经在 WHERE 子句中使用过比较运算符，这些运算符如表 24-5 所示。比较

运算符会传回一个 TRUE 或 FALSE 的布尔值（Boolean Value）。

运算符	意义
=	等于。
>	大于。

<	小于。
>=	大于或等于。
<=	小于或等于。
< >	不等于。

表 24-5 比较运算符

在 WHERE 子句中使用比较运算符

- 1. 在 Query Analyzer 工具列上按一下 新增查询 按钮,以便开启一个新的 查询 窗口。



新增查询按钮

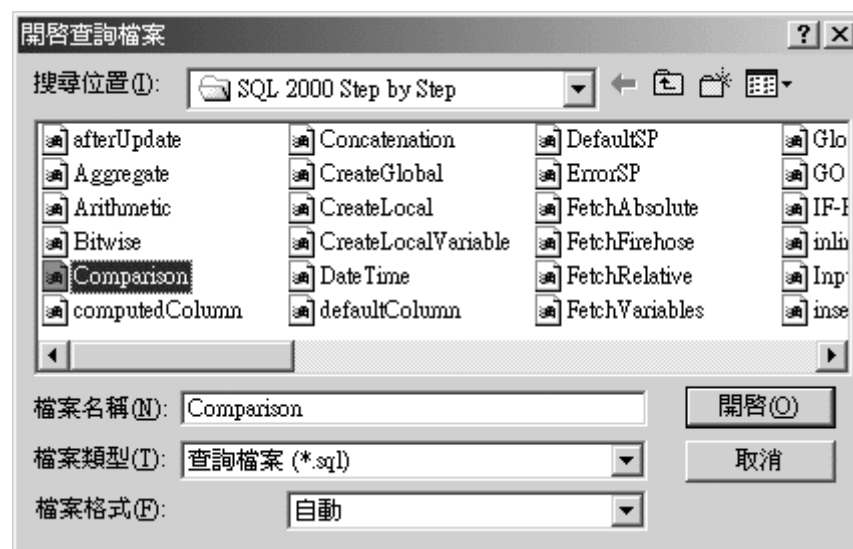
此时 Query Analyzer 将会开启一个空白的 查询 窗口。

- 2. 在 Query Analyzer 工具列上按一下 加载 SQL 指令码 按钮。



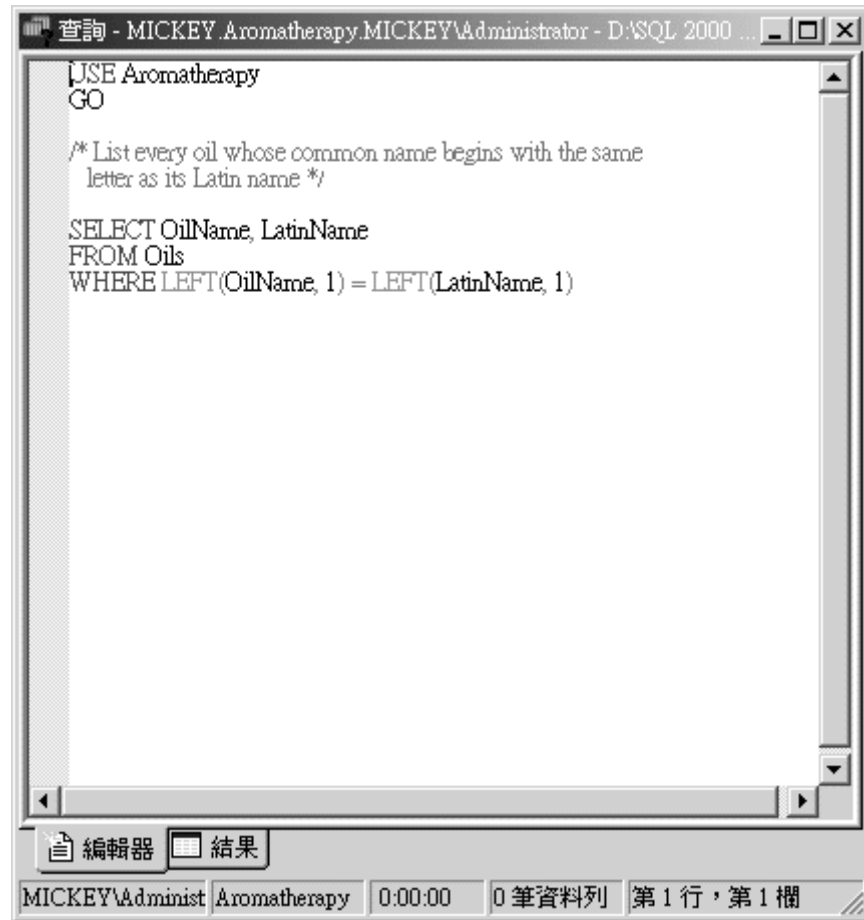
加载 SQL 指令码按钮

Query Analyzer 会显示 [开启查询档案](#) 的对话框。



3. 选取文件名称为 [Comparison](#) 的档案，然后按一下 [开启](#) 按钮。

Query Analyzer 会将指令码档案加载到 [查询](#) 窗口中。



4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕，以便執行此查詢。



執行查詢按鈕

Query Analyzer 会在 [方格窗格](#) 中显示此查询的执行结果。

查詢 - MICKEY.Aromatherapy.MICKEY\Administrator - D:\SQL 2000 ...

	OilName	LatinName
1	Cedarwood	Cedrus Atlantica, Juniperus Virginiana
2	Cinnamon	Cinnamomum Zeylanicum
3	Citronella	Cymbopogon Nardus
4	Coriander	Coriandrum Sativum
5	Cypress	Cupressus Sempervirens
6	Eucalyptus	Eucalyptus Globulus
7	Jasmine	Jasmineum Officinale
8	Juniper	Juniperus Communis
9	Lavender	Lavandulus Officinalis, Lavandula Of...
10	Patchouli	Pogostemon Patchouli
11	Pine	Pinus Sylvestris
12	Rose	Rosa Centifolia
13	Rosemary	Rosmarinus Officinalis
14	Sandalwood	Santalum Album
15	Thyme	Thymus Vulgaris
16	Fennel	Foeniculum Vulgare
17	Rose Otto	Rosa Damascena
18	Sage	Salvia Officinalis
19	Vanilla Oleoresin	Vanilla Planifolia

編輯器 方格 訊息

MICKEY Aromatherapy 0:00:00 Grid #1: 19 筆資料列 第 1 行，第 1 欄

5. 关闭此 查詢 窗口。

逻辑运算符

就像比较运算符一样，逻辑运算符也会传回 TRUE 或 FALSE 的值，但是它们是用来测试某些条

件是否为 TRUE 或 FALSE。表 24-6 所示为 SQL Server 所支持的三种逻辑运算符。

运算符	意义
AND	假如二个值都为 TRUE 时，其结果则为 TRUE。
NOT	反向的布尔数值。
OR	如果一个或二个值为 TRUE 时，其结果则为 TRUE。

表 24-6 逻辑运算符

在 SELECT 陈述式中使用逻辑运算符

1. 在 Query Analyzer 工具列上按一下 [新增查询](#) 按钮，以便开启一个新的 [查询](#) 窗口。



新增查询按钮

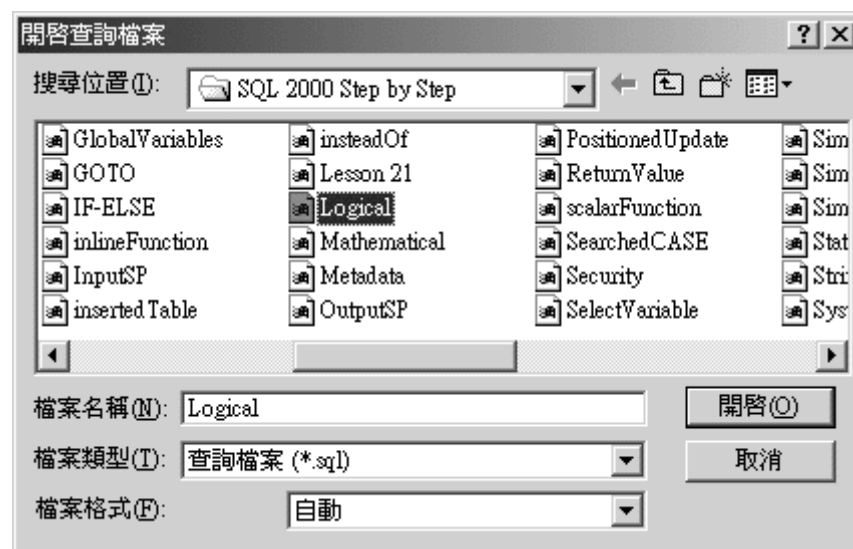
此时 Query Analyzer 会开启一个空白的 [查询](#) 窗口。

2. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。
-



加载 SQL 指令码按钮

Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



3. 选取文件名称为 **Logical** 的档案，然后按一下 [开启](#) 按钮。

Query Analyzer 会将该指令码档案加载到 [查询](#) 窗口中。

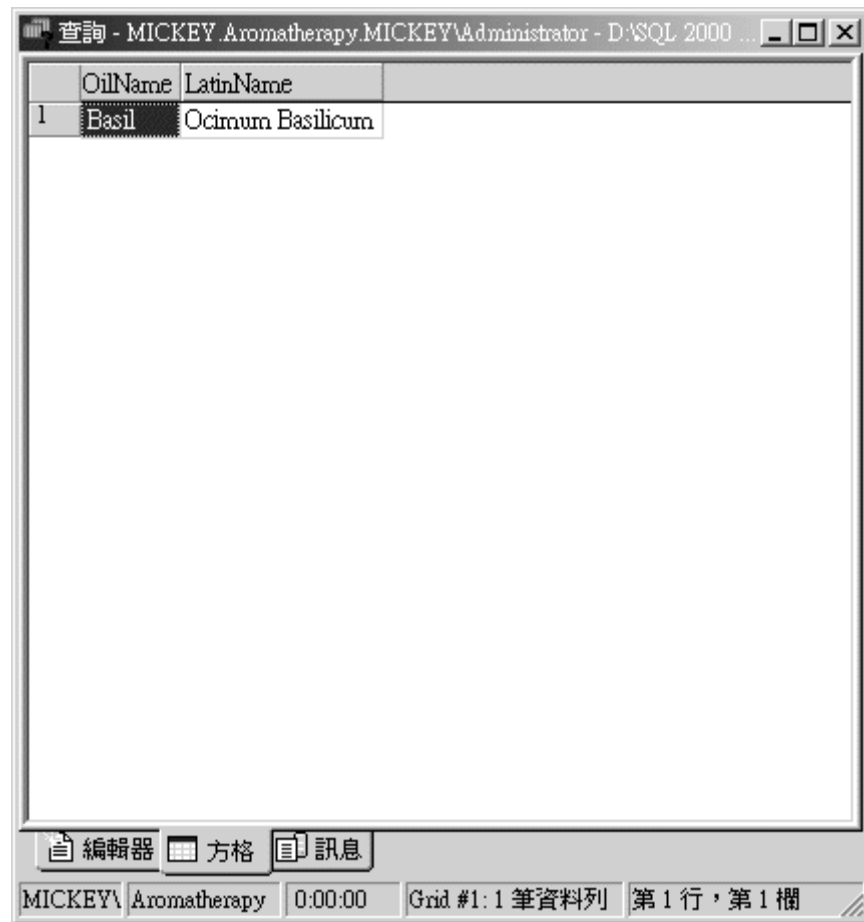


4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕，以便執行該查詢。



執行查詢按鈕

Query Analyzer 会在 [方格窗格](#) 中显示此查询的执行结果。



5. 关闭此 查询 窗口。

位运算符

位运算符的执行操作方式与逻辑运算符非常相似，然而逻辑运算符会传回基于布尔值的结果；而

位运算符是在整数值有位层级上操作。表 24-7 为最常使用的位运算符。

位运算符 **&** 和 **|** 的功能也一样（也有相对应的逻辑运算符—AND 和 OR），除了一点不同，它们是比较两个值的每一个位，而不是比较整个值。

^ 运算符并没有相对应的逻辑运算符。在布尔代数中，如果一个或二个值都为 TRUE 时，则 OR 会传回一个布尔值 TRUE。如果有一个（但不是两个）的值为 TRUE 时，**Boolean Exclusive OR** 将会传回 TRUE，这正是 **^** 运算符所要做的事：如果有一个（但不是二个）的值为 TRUE 时，则它会传会 TRUE。

运算符	意义
&	在两个整数值之间执行位逻辑 AND 运算。
	在两个整数值之间执行位逻辑 OR 运算。
^	在两个整数值之间执行位逻辑 Exclusive OR 运算。
~	在两个整数值之间执行位逻辑 NOT 运算。

表 24-7 位运算符

位封装

位运算符必须操作整数值中的每一个位以指出其不同的属性，这种技术就称为 **位封装**（bit packing）。根据定义，如果单一个整数用于储存多种属性时，该值就不是纯量，并且该数据表没有正规化。

在正常的情况之下，在关系型数据库内，您不应该使用位封装。然而，如果您是使用旧型系统数据时，有时候使用位封装值是不可避免的，而在这种情况下，使用位运算符是非常方便的。

在 **SELECT** 陈述式中使用位运算符

1. 在 Query Analyzer 工具列上按一下 **新增查询** 按钮，以便开启一个新的 **查询** 窗口。



新增查询按钮

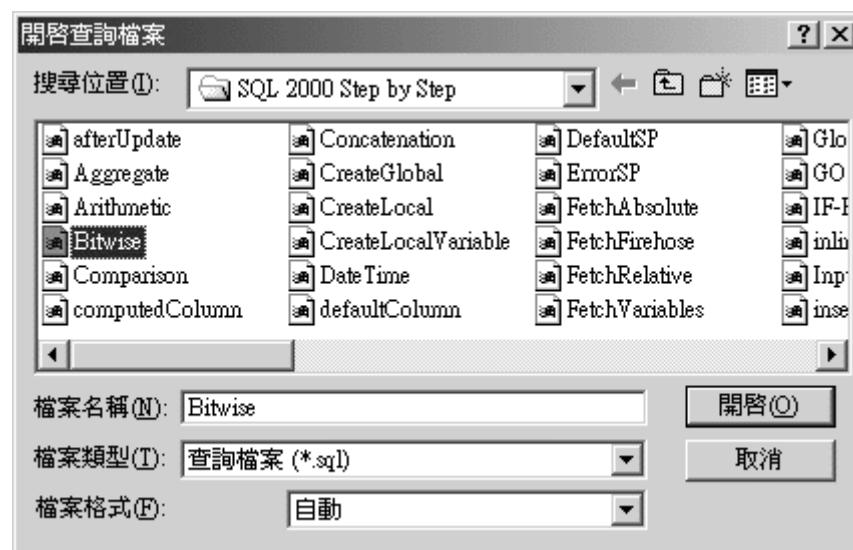
此时 Query Analyzer 会开启一个空白的 **查询** 窗口。

2. 在 Query Analyzer 工具列上按一下 **加载 SQL 指令码** 按钮。



加载 SQL 指令码按钮

Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



3. 选取文件名称为 **Bitwise** 的档案，然后按一下 [开启](#) 按钮。

Query Analyzer 会将指令码档案加载到 [查询](#) 窗口中。

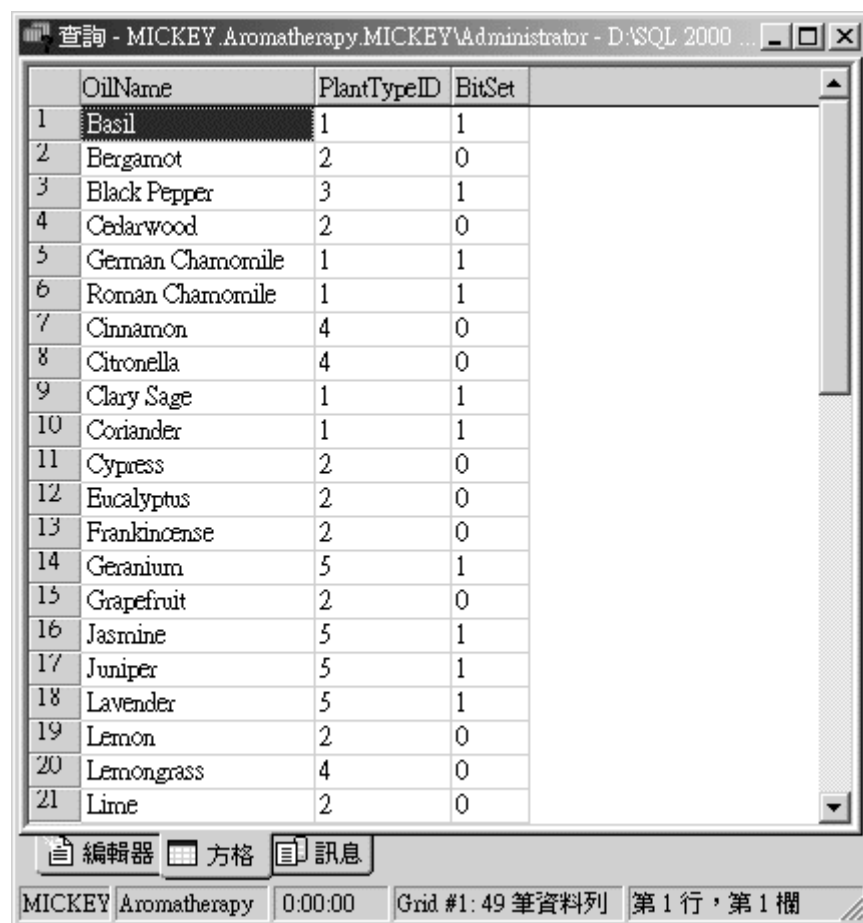


4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕，以便執行此查詢。



執行查詢按鈕

Query Analyzer 会在方格窗格中显示此查询的执行结果。



The screenshot shows the 'Query - MICKEY.Aromatherapy.MICKEY\Administrator - D:\SQL 2000 ...' window. The main area displays a table with 4 columns: an index, OilName, PlantTypeID, and BitSet. The table contains 21 rows of data. The bottom status bar indicates 'MICKEY Aromatherapy 0:00:00 Grid #1: 49 筆資料列 第 1 行，第 1 欄'.

	OilName	PlantTypeID	BitSet
1	Basil	1	1
2	Bergamot	2	0
3	Black Pepper	3	1
4	Cedarwood	2	0
5	German Chamomile	1	1
6	Roman Chamomile	1	1
7	Cinnamon	4	0
8	Citronella	4	0
9	Clary Sage	1	1
10	Coriander	1	1
11	Cypress	2	0
12	Eucalyptus	2	0
13	Frankincense	2	0
14	Geranium	5	1
15	Grapefruit	2	0
16	Jasmine	5	1
17	Juniper	5	1
18	Lavender	5	1
19	Lemon	2	0
20	Lemongrass	4	0
21	Lime	2	0

5. 关闭此 查询 窗口。

其它的运算符

Transact-SQL 提供二种好用的运算符，如表 24-8 所示。我们已经在第 12 章中使用过串连运算符（+），这个串连运算符是将一个字符串的内容加到另一个字符串中。指派运算符（=），是用来将右边的值指派给左边。请注意这与您在学校所学的顺序是相反的，举例来说，不是 $a + b = c$ ，而是 $c = a + b$ 。我们在本章稍后的部分说明变量时，就会使用指派运算符了。

运算符	意义
+	字符串连接
=	指派运算符

表 24-8 其它的运算符

在 **SELECT** 陈述式中使用串连运算符

1. 在 Query Analyzer 工具列上按一下 **新增查询** 按钮，以便开启一个新的 **查询** 窗口。



新增查询按钮

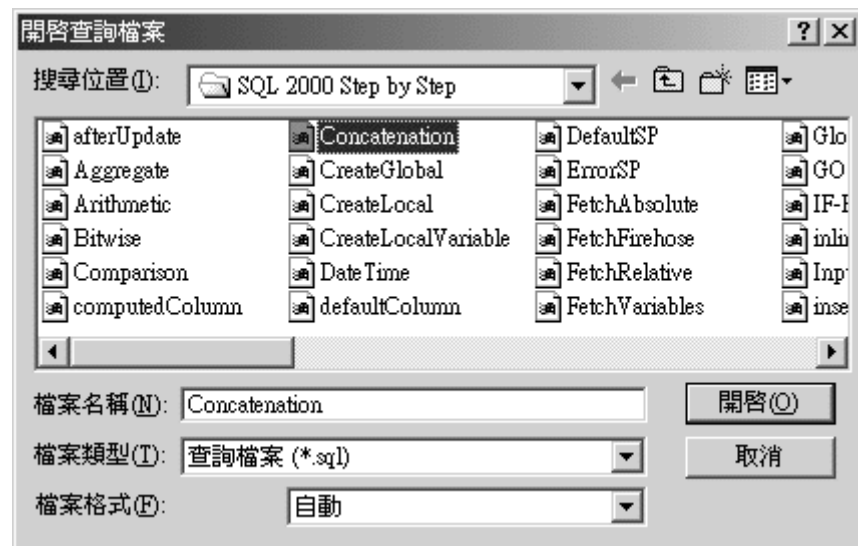
此时 Query Analyzer 会开启一个空白的 **查询** 窗口。

2. 在 Query Analyzer 工具列上按一下 **加载 SQL 指令码** 按钮。



加载 SQL 指令码按钮

Query Analyzer 会显示一个 **开启查询档案** 的对话框。



3. 选取文件名称为 **Concatenation** 的档案，然后按一下 **开启** 按钮。

Query Analyzer 将指令码档案加载到 **查询** 窗口中。

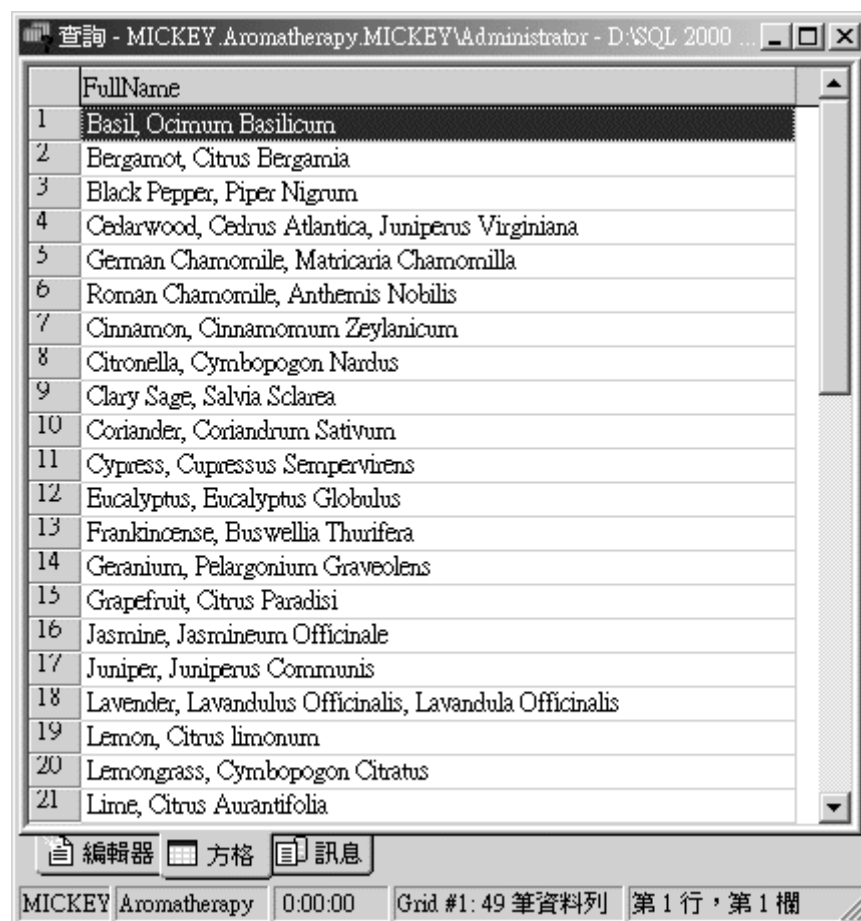


4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕，以便執行此查詢。



執行查詢按鈕

Query Analyzer 会在 [方格窗格](#) 中显示此查询的执行结果。



The screenshot shows a window titled "查詢 - MICKEY.Aromatherapy.MICKEY\Administrator - D:\SQL 2000 ...". The window contains a table with 21 rows of data. The first column contains row numbers from 1 to 21, and the second column contains the full names of various aromatic plants. The table is displayed in a grid view. Below the table, there are buttons for "編輯器" (Editor), "方格" (Grid), and "訊息" (Messages). At the bottom, there is a status bar showing "MICKEY Aromatherapy", "0:00:00", "Grid #1: 49 筆資料列", and "第 1 行, 第 1 欄".

	FullName
1	Basil, Ocimum Basilicum
2	Bergamot, Citrus Bergamia
3	Black Pepper, Piper Nigrum
4	Cedarwood, Cedrus Atlantica, Juniperus Virginiana
5	German Chamomile, Matricaria Chamomilla
6	Roman Chamomile, Anthemis Nobilis
7	Cinnamon, Cinnamomum Zeylanicum
8	Citronella, Cymbopogon Nardus
9	Clary Sage, Salvia Sclarea
10	Coriander, Coriandrum Sativum
11	Cypress, Cupressus Sempervirens
12	Eucalyptus, Eucalyptus Globulus
13	Frankincense, Boswellia Thurifera
14	Geranium, Pelargonium Graveolens
15	Grapefruit, Citrus Paradisi
16	Jasmine, Jasmineum Officinale
17	Juniper, Juniperus Communis
18	Lavender, Lavandulus Officinalis, Lavandula Officinalis
19	Lemon, Citrus limonum
20	Lemongrass, Cymbopogon Citratus
21	Lime, Citrus Aurantifolia

5. 关闭此 [查询](#) 窗口。

Transact-SQL 函数

函数是由数行陈述式所组成。函数需执行一个单一的 [逻辑] 动作，而这个逻辑动作是由许多实体动作所组成。举例来说，一个函数自一个数据表中将一系列数据列移至另一个数据表时所执行的逻辑动作（移动），事实上是执行了二个实体动作（INSERT 和 DELETE）。

在 SQL Server 2000 所提供的新功能中，允许您自己建立自己的函数，这种函数就称为 **使用者自订函数**（user-defined function），我们将在第 30 章中说明。Transact-SQL 也提供了多种内建的函数，我们将在此说明。

使用函数

内建的 Transact-SQL 函数是藉由它们所传回的结果分类：**决定性**（deterministic）或 **非决定性**（non-deterministic）。所谓决定性函数是指在任何时候以特定的输入值呼叫决定性函数时，一律传回相同的结果：例如 SQRT(9)的结果是 3，因此 SQRT 函数是属于决定性的函数。所谓的非决定性函数是指每次以特定的输入值呼叫非决定性函数时，都会传回不同的结果，例如 RAND 函数每次所传回的结果都不一样，因此 RAND 函数是属于非决定性的函数。

日期和时间函数

日期和时间函数接受以日期和时间来当作输入值，并且它会传回字符串、数值或日期及时间的值（请记得，在 SQL Server 中，其时间所使用的数据型别是 **datetime**）。

datepart 函数则可以取出日期的特定部分。表 24-9 为 Transact-SQL 所提供的日期和时间函数。

函数	参数	功能
DATEADD	datepart, number, date	新增特定的 datepart 间隔数至 date，并 且传回新的 date 值。
DATEDIFF	datepart, startdate, enddate	传回两个特定的 date 之间的 datepart 数。
DATENAME	datepart, date	在 date 之中传回特定的 datepart 名称（字符串）。
DATEPART	datepart, date	在 date 之中传回特定的 datepart（整数）。
DAY	date	在 date 之中传回特定的天（整数）。
GETDATE		传回目前的系统日期和时间（整数）。
MONTH	date	在 date 之中传回特定的月（整数）。
YEAR	date	在 date 之中传回特定的年（整数）。

表 24-9 日期和时间函数

使用日期函数

1. 在 Query Analyzer 工具列上按一下 **新增查询** 按钮，以便开启一个新的 **查询** 窗口。



新增查询按钮

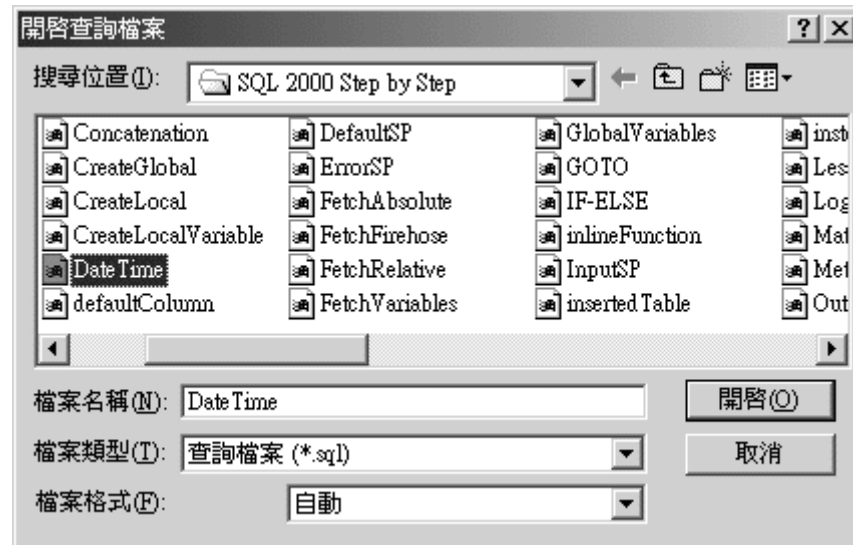
此时 Query Analyzer 会开启一个空白的 [查询](#) 窗口。

2. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。



加载 SQL 指令码按钮

Query Analyzer 会显示 [开启查询档案](#) 的对话框。



3. 选取文件名称为 **DateTime** 的档案，然后按一下 **开启** 按钮。

Query Analyzer 会将指令码档案加载到 **查询** 窗口中。



4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕，以便執行此查詢。



執行查詢按鈕

Query Analyzer 会在 [方格窗格](#) 中显示此查询的执行结果。

查詢 - MICKEY.Aromatherapy.MICKEY\Administrator - D:\SQL 2000 ...							
	DateAdd	DateDiff	DateName	DatePart	Day	Month	Year
1	2002-02-09 01:05:05.833	2206	二月	1	9	2	2001

編輯器
方格
訊息

MICKEY\Aromatherapy
0:00:00
Grid #1: 1 筆資料列
第 1 行，第 1 欄

5. 关闭此 查询 窗口。

数学函数

表 24-10 所示为常用的数学函数清单。

函数	参数	功能
----	----	----

ABS	numeric_expression	传回指定数值表达式的绝对值。
ACOS	float_expression	传回 float_expression 的反余弦值。
ASIN	float_expression	传回 float_expression 的反正弦值。
ATAN	float_expression	传回 float_expression 的反正切值。
ATN2	float_expression, float_expression	传回以弧度为单位的角度，其正切值 介于两个指定的 float_expression 之间。
CEILING	numeric_expression	传回大于等于 numeric_expression 的最小整数。
COS	float_expression	传回 float_expression 的余弦值。
COT	float_expression	传回 float_expression 的余切值。
DEGREES	numeric_expression	以弧度传回的角度，以刻度传回其对应的角度。
EXP	float_expression	传回 float_expression 的指数值。
FLOOR	numeric_expression	传回小于或等于给定数值表达式的最大整数。
LOG	float_expression	传回 float_expression 的自然对数值。
LOG10	float_expression	传回 float_expression 之 10 的对数值。
PI		传回 PI 的常数值。
POWER	numeric_expression, y	传回 numeric_expression 之 Y 乘幂的值。
RADIANS	numeric_expression	输入 numeric_expression 时，以度为单位传回弧度。

RAND	seed	传回从 0 到 1 的随机 float 值。
ROUND	numeric_expression, length	传回已经进位到 length 或有效位数的 numeric_expression。
SIGN	numeric_expression	传回 numeric_expression 之正 (+1)、零 (0) 或负 (-1) 号。
SIN	float_expression	传回 float_expression 的正弦函数值。
SQUARE	float_expression	传回 float_expression 的平方值。
SQRT	float_expression	传回 float_expression 的平方根值。
TAN	float_expression	传回 float_expression 的正切函数值。

表 24-10 数学函数

使用数学函数

1. 在 Query Analyzer 工具列上按一下 [新增查询](#) 按钮,以便开启一个新的 [查询](#) 窗口。



执行查询按钮

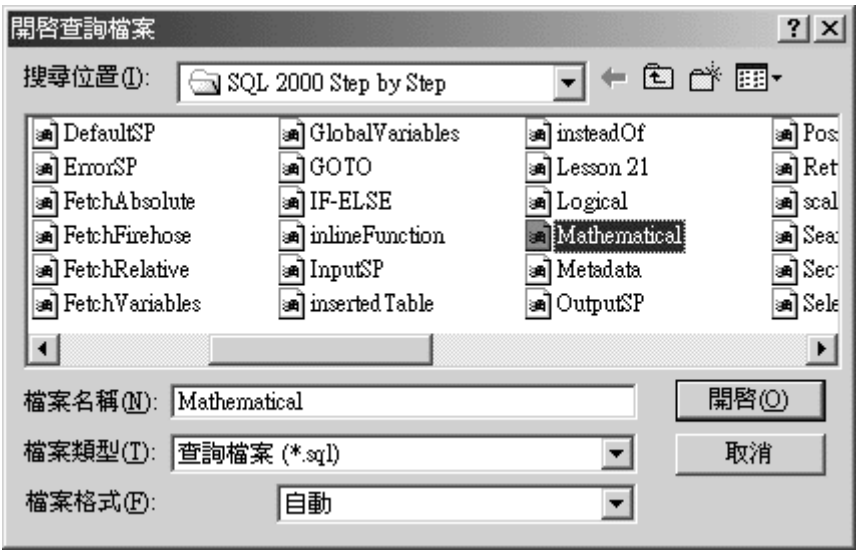
此时 Query Analyzer 会开启一个空白的 [查询](#) 窗口。

2. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。



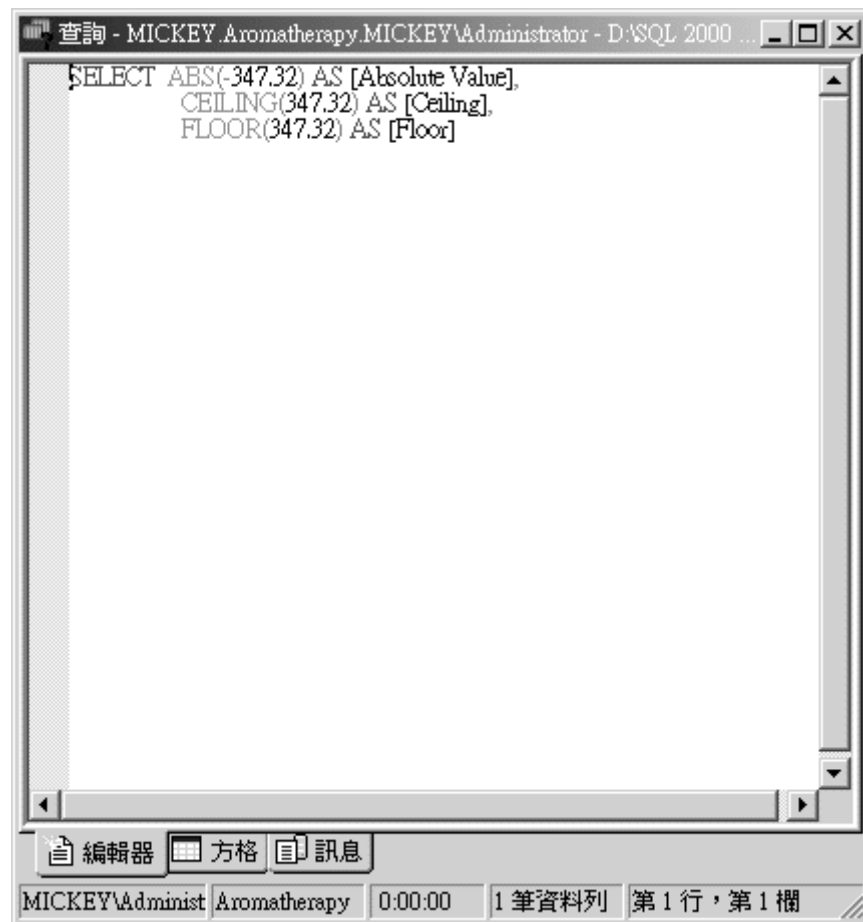
加载 SQL 指令码按钮

Query Analyzer 会显示 [开启查询档案](#) 的对话框。



3. 选取文件名称为 **Mathematical** 的档案，然后按一下 **开启** 按钮。

Query Analyzer 会将指令码档案加载到 **查询** 窗口中。

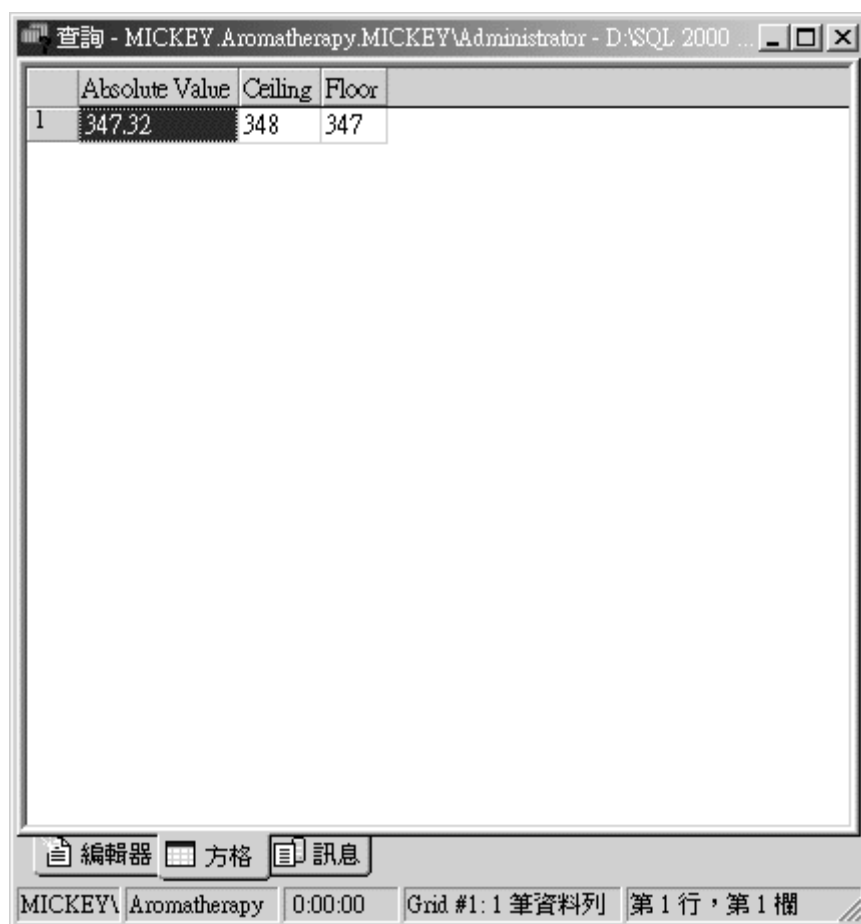


4. 在 Query Analyzer 的工具列上按一下 **执行查询** 按钮，以便执行此查询。



执行查询按钮

Query Analyzer 会在 [方格窗格](#) 中显示此查询的执行结果。



The screenshot shows the 'Query - MICKEY.Aromatherapy.MICKEY\Administrator - D:\SQL 2000 ...' window. The main area displays a grid with the following data:

	Absolute Value	Ceiling	Floor
1	347.32	348	347

At the bottom, the status bar indicates 'MICKEY\ Aromatherapy 0:00:00 Grid #1: 1 筆資料列 第 1 行，第 1 欄'.

5. 关闭此 [查询](#) 窗口。

汇总函数

SQL Server 2000 所提供的汇总函数种类如表 24-11 所示，所谓的汇总函数是指将一些数值集合进行运算，并且只传回单一的汇总值。

函数	功能
AVG	传回集合中值的平均值，忽略 Null 值。
COUNT	传回集合中项目的数目，忽略 Null 值。
MAX	传回集合中的最大值。
MIN	传回集合中的最小值。
SUM	传回集合中的总和，忽略 Null 值。
STDEV	传回集合中所有值的统计标准差。
STDEVP	传回集合中所有数值扩展的统计标准差。
VAR	传回群组中所有值的统计变异数。
VARP	传回集合中所有值的群组统计变异数

表 24-11 汇总函数

使用汇总函数

1. 在 Query Analyzer 工具列上按一下 [新增查询](#) 按钮,以便开启一个新的 [查询](#) 窗口。



新增查询按钮

此时 Query Analyzer 会开启一个空白的 [查询](#) 窗口。

2. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。



加载 SQL 指令码按钮

Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



3. 选取文件名称为 **Aggregate** 的档案，然后按一下 **开启** 按钮。

Query Analyzer 会将指令码档案加载到 **查询** 窗口中。



4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕，以便執行此查詢。



執行查詢按鈕

Query Analyzer 会在 [方格窗格](#) 中显示此查询的执行结果。

查詢 - MICKEY.Aromatherapy.MICKEY\Administrator - D:\SQL 2000 ...

	Largest Value	Smallest Value	Number of Rows
1	Ylang-ylang	Basil	49

編輯器 方格 訊息

MICKEY\ Aromatherapy 0:00:00 Grid #1: 1 筆資料列 第 1 行，第 1 欄

5. 关闭此 查詢 窗口。

中继数据函数

中继数据（Metadata）函数会传回关于数据本身的相关数据，并且 SQL Server 2000 提供了许多的中继数据函数供您使用。表 24-12 所显示的是大多数所使用的中继数据函数。

函数	参数	功能
COL_LENGTH	table, column	传回 column 之位组长度。
COL_NAME	tableID, columnID	传回 column 的名称。
COLUMNPROPERTY	ID, column, property	传回有关 column 的 property 参 数信息。
DATABASEPROPERTY	database, property	传回 property 值。
DB_ID	database_name	传回 database_name 的数据库识别码。
DB_NAME	databaseID	传回 databaseID 的数据库名称。
INDEX_COL	table, indexID, keyID	传回 indexed 及 keyed 的索引数据行名称。
INDEXPROPERTY	tableID, index, property	传回 index 的 property 值。
OBJECT_ID	object	传回数据库 object 识别码。
OBJECT_NAME	objectID	传回 objectID 的对象名称。
OBJECTPROPERTY	ID, property	传回相关对象 ID 的 property 信息。
SQL_VARIANT_PROPERTY	SQL_variant, property	传回 SQL_variant 的特定 property。
TYPEPROPERTY	datatype, property	传回相关 datatype 的 property 的信息。

表 24-12 中继数据函数

使用中继数据函数

1. 在 Query Analyzer 工具列上按一下 [新增查询](#) 按钮,以便开启一个新的 [查询](#) 窗口。



新增查询按钮

此时 Query Analyzer 会开启一个空白的 [查询](#) 窗口。

2. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。



加载 SQL 指令码按钮

Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



3. 选取文件名称为 **Metadata** 的档案，然后按一下 **开启** 按钮。

Query Analyzer 会将指令码档案加载到 **查询** 窗口中。

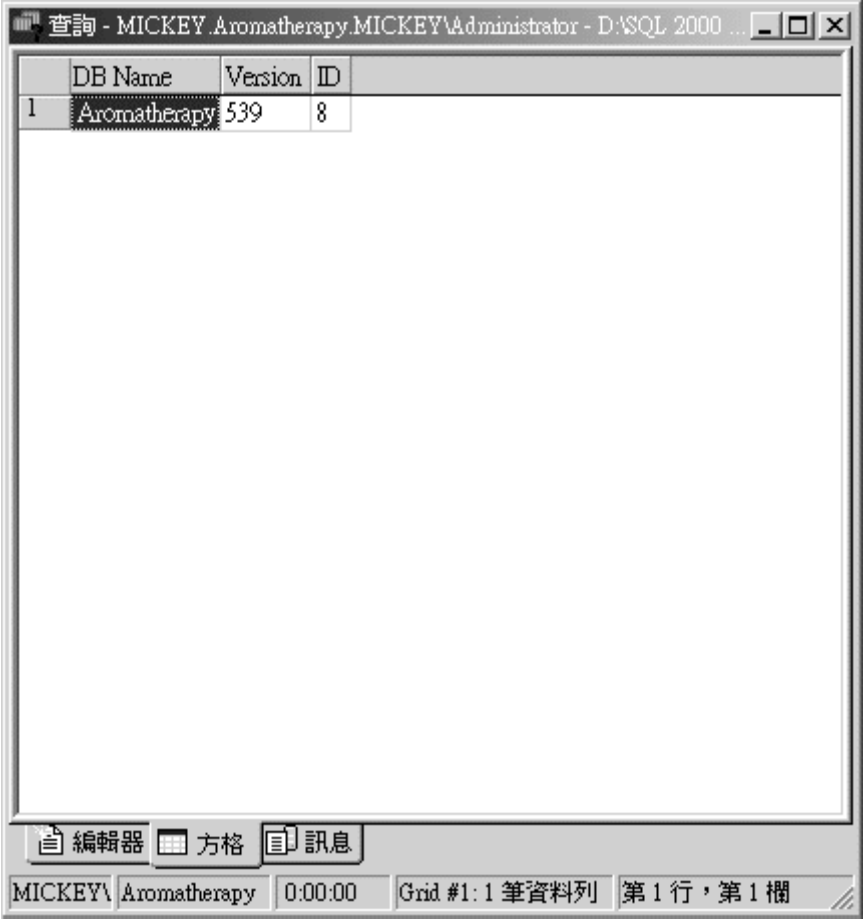


4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕，以便執行此查詢。



執行查詢按鈕

Query Analyzer 会在 [方格窗格](#) 中显示此查询的执行结果。



5. 关闭此 查询 窗口。

使用安全性函数

表 24-13 为常用的安全性函数清单。安全性函数会传回关于使用者和角色的相关信息。

函数	参数	功能
----	----	----

HAS_DBACCESS	database_name	指出目前使用者是否可以存取 database_name。
IS_MEMBER	group_or_role	指出目前使用者是否为 group_or_role 的成员。
IS_SRVROLEMEMBER	role [, login]	指出目前 login 是否为指定 role 的成员。
SUSER_SID	[login]	传回目前或指定 login 的安全性识别码 (SID)。
SUSER_SNAME	[SID]	传回 SID 的登入名称。
USER_ID	[user]	传回目前或特定 user 的识别码。
USER		传回目前使用者的数据库使用者名称。

表 24-13 安全性函数

使用安全性函数

1. 在 Query Analyzer 工具列上按一下 [新增查询](#) 按钮,以便开启一个新的 [查询](#) 窗口。



新增查询按钮

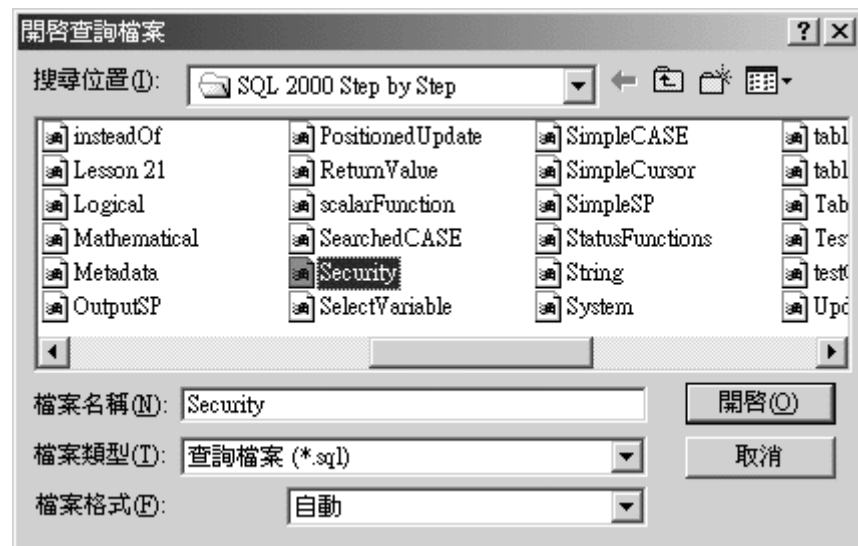
此时 Query Analyzer 会开启一个空白的 [查询](#) 窗口。

2. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。



加载 SQL 指令码按钮

Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



3. 选取文件名称为 [Security](#) 的档案，然后按一下 [开启](#) 按钮。

Query Analyzer 会将指令码档案加载到 [查询](#) 窗口中。



4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕，以便執行此查詢。



執行查詢按鈕

Query Analyzer 会在 [方格窗格](#) 中显示此查询的执行结果。



The screenshot shows the 'Query - MICKEY.Aromatherapy.MICKEY\Administrator - D:\SQL 2000 ...' window. The main area displays a grid with the following data:

	User Name	ID	Access Master
1	dbo	1	1

At the bottom, the status bar indicates 'MICKEY\ Aromatherapy 0:00:00 Grid #1: 1 筆資料列 第 1 行，第 1 欄'.

5. 关闭此 [查询](#) 窗口。

使用字符串函数

字符串函数会执行字符串值的操作，并且传回字符串或数值。表 24-14 为大多数使用者常用的字符串函数。

函数	参数	功能
ASCII	char_expression	传回 char_expression 最左边字符的 ASCII 字码。
CHAR	integer_expression	传回 integer_expression 的 ASCII 字符。
CHARINDEX	char_expression, char_expression, [, start_position]	传回第一个 char_expression 在第二个 char_expression 的位置。
LEFT	char_expression, integer_expression	传回在 char_expression 中由 integer_expression 指定最左边的所有字符数。
LEN	char_expression	传回在 char_expression 中的字符数。
LOWER	char_expression	传回 char_expression 中的字符全部转成小写。
LTRIM	char_expression	传回 char_expression 中的字符（但是去除最前面的空格）。
NCHAR	integer_expression	传回 integer_expression 中的 Unicode 字符。
REPLACE	char_expression, char_expression, char_expression	在第一个 char_expression 搜寻第二个 中指定的字符，并且以 char_expression 第三个 char_expression 中的字符取代。

RIGHT	char_expression, integer_expression	传回在 char_expression 中由 integer_expression 指定最右边的所有字符数。
RTRIM	char_expression	传回 char_expression 中的字符（但是去除最后面的空格）
SOUNDEX	char_expression	传回 char_expression 的四个字符 SOUNDEX 码。
SPACE	integer_expression	传回 integer_expression 指定的空格符数。
SUBSTRING	char_expression, start, length	传回 char_expression 中的 length 字符数，由 start 指定开始处。
UNICODE	unicode_expression	传回在 unicode_expression 中第一个字 元的 Unicode 整数值。
UPPER	char_expression	传回 char_expression 中的字符全部转成大写。

24-14 字符串函数

使用字符串函数

1. 在 Query Analyzer 工具列上按一下 [新增查询](#) 按钮，以便开启一个新的 查询 窗口。





新增查询按钮

此时 Query Analyzer 会开启一个空白的 [查询](#) 窗口。

2. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。
-



加载 SQL 指令码按钮

Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



3. 选取文件名称为 **String** 的档案，然后按一下 **开启** 按钮。

Query Analyzer 会将指令码档案加载到 **查询** 窗口中。



4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕，以便執行此查詢。



執行查詢按鈕

Query Analyzer 会在 [方格窗格](#) 中显示此查询的执行结果。

查詢 - MICKEY.Aromatherapy.MICKEY\Administrator - D:\SQL 2000 ...

	OilName	Uppercase	Lowercase
1	Basil	BASIL	basil
2	Benzoin	BENZOIN	benzoin
3	Bergamot	BERGAMOT	bergamot
4	Black Pepper	BLACK PEPPER	black pepper
5	Cedarwood	CEDARWOOD	cedarwood
6	Cinnamon	CINNAMON	cinnamon
7	Citronella	CITRONELLA	citronella
8	Clary Sage	CLARY SAGE	clary sage
9	Coriander	CORIANDER	coriander
10	Cypress	CYPRESS	cypress
11	Eucalyptus	EUCALYPTUS	eucalyptus
12	Fennel	FENNEL	fennel
13	Frankincense	FRANKINCENSE	frankincense
14	Geranium	GERANIUM	geranium
15	German Chamomile	GERMAN CHAMOMILE	german chamomile
16	Grapefruit	GRAPEFRUIT	grapefruit
17	InsertAllValues	INSERTALLVALUES	insertallvalues
18	InsertDefault	INSERTDEFAULT	insertdefault
19	InsertFromGrid	INSERTFROMGRID	insertfromgrid
20	InsertFromSQL	INSERTFROMSQL	insertfromsql

編輯器 方格 訊息

MICKEY Aromatherapy 0:00:00 Grid #1: 49 筆資料列 第 1 行，第 1 欄

5. 关闭此 查詢 窗口。

使用系统函数

系统函数会传回关于 SQL Server 环境的信息。就像是中继数据函数一样，有许多的系统函数，

在 对象浏览器 中都可以看到。表 24-15 为大多数使用者常用的系统函数。

函数	参数	功能
APP_NAME		传回目前工作阶段的应用程序名称。
DATALength	expression	传回用以代表 expression 的字节数目。
ISDATE	expression	决定 expression 是否为有效的日期。
ISNULL	expression	决定 expression 是否为 NULL。
ISNUMERIC	expression	决定 expression 是否为有效的数值。
NEWID		建立 uniqueidentifier 数据型别的唯一值。
NULLIF	expression, expression	如果二个 expression 相同时，则传回 NULL。
PARSENAME	object_name, name_part	传回 object_name 的 name_part。
SYSTEM_USER		传回目前的系统使用者名称。
USER_NAME	[id]	传回目前或特定使用者 id 的使用者名称。

表 24-15 系统函数

使用系统函数

1. 在 Query Analyzer 工具列上按一下 [新增查询](#) 按钮，以便开启一个新的 [查询](#) 窗口。





新增查询按钮

此时 Query Analyzer 会开启一个空白的 [查询](#) 窗口。

2. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。
-



加载 SQL 指令码按钮

Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



3. 选取文件名称为 **System** 的档案，然后按一下 **开启** 按钮。

此 Query Analyzer 会将指令码档案加载到 **查询** 窗口中。



4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕，以便執行此查詢。



執行查詢按鈕

Query Analyzer 会在 [方格窗格](#) 中显示此查询的执行结果。

查詢 - MICKEY.Aromatherapy.MICKEY\Administrator - D:\SQL 2000 ...				
	Application	User	Is a Date	Is Not a Date
1	SQL Query Analyzer	MICKEY\Administrator	0	0

編輯器	方格	訊息
-----	----	----

MICKEY\Aromatherapy	0:00:00	Grid #1: 1 筆資料列	第 1 行，第 1 欄
---------------------	---------	-----------------	-------------

5. 关闭此 查詢 窗口。

25. 设计对象

在本章中，您将学习到：

- 建立暂存数据表。
- 建立全域暂存数据表。
- 建立区域变量。
- 建立全域变量。
- 使用 **SET** 命令来设定变量值。
- 使用 **SELECT** 指令来设定变量值。

就像任何的程序语言一样，**Transact-SQL** 也提供可以建立重复使用程序的机制。这些程序大多

数都包含二个重点，就是可以建立暂存对象，并且可以传递和接收参数值。

在 Transact-SQL 中对暂存对象的支持是透过建立暂存数据表和变量的使用；而使用参数就可以传递特定值至函数和预存程序中，我们将在本章中详细说明关于暂存数据表以及变量，而参数的使用说明我们将在第 28 章中说明。

暂存资料表

除了当暂存数据表被使用时才会存在之外，暂存数据表和一般的数据表是差不多的。当所有使用者结束使用暂存数据表时，Microsoft SQL Server 会自动地将它们删除。

提示

建立暂存数据表相当地浪费服务器的资源以及 CPU 周期，因此现在通常会使用数据表变量来取代暂存数据表。

认识暂存资料表

建立暂存数据表与 Transact-SQL 建立一般数据表所使用的命令是相同的(CREATE 或 SELECT INTO)。与一般数据表不同的是一暂存数据表的名称是以「#」或「##」来作为前置字。因此在

下面的范例中，第一个陈述式是用来建立一般性的数据表，而第二个陈述式则是用来建立暂存数据表的。

```
CREATE TABLE NormalTable (theKey INT PRIMARY KEY, theValue CHAR(20))  
CREATE TABLE #TemporaryTable (theKey INT PRIMARY KEY, theValue CHAR(20))
```

暂存数据表在设计时其限制条件是不能使用 **FOREIGN KEY** 条件约束，除此之外，**CREATE TABLE** 陈述式的全部功能都可以使用。暂存数据表可以使用 **CHECK** 条件约束、默认值以及其它选项。

暂存数据表通常是在 **tempdb** 系统数据库内建立的。每当 **SQL Server** 重新启动时，其 **tempdb** 数据库内的暂存数据表都会被移除。一般来说，当暂存数据表不再使用时就会被移除。

辨别暂存数据表及一般数据表的方式是从它的名称来决定的。暂存数据表的名称是以井号「#」来作为前置字，例如 **[#MyTable]**。一个「#」所代表的是一个区域暂存数据表；而「##」所代表的的是一个全域暂存资料表。

区域暂存数据表（有时也会被称为私用的暂存数据表）只有建立该暂存数据表的工作阶段中的使用者可看到它们，而其它工作阶段的使用者（甚至于是相同的使用者所建立的工作阶段）无法看到它们，或是存取它们。建立暂存资料表的工作阶段在任何时间都可以将该暂存资料表移除，假如使用者已经注销，而该暂存数据表还存在时，**SQL Server** 也会自动地将该暂存数据表移除。

全域暂存数据表是使用二个井号「##」来表示，当它建立之后，其数据库中的任何使用者都可以使用此数据表，同时您也不需要特别授权特殊的权限给全域暂存数据表，全域暂存数据表会自动有特殊权限。

您可以明确的删除全域暂存资料表，或是当建立该全域暂存数据表的使用者注销并且使用完成后，SQL Server 会自动删除它。举例来说，假如 USER A 建立了全域暂存数据表##MyTable，而 USER B 正在使用该全域暂存数据表时，USER A 就注销服务器，那么 SQL Server 会等所有目前执行的命令完成就删除##MyTable，如果此时 USER B 再度对##MyTable 下达指令，则该指令就无效了。

使用暂存数据表

您可以使用 CREATE 和 SELECT INTO 指令就像是建立一般数据表那样来建立暂存数据表。联机即为存取暂存数据表最主要的问题，而在其它方面暂存数据表和一般数据表都是一样的。

建立区域暂存数据表

1. 在 Query Analyzer 工具列上按一下 [新增查询](#) 按钮，以便开启一个新的查询窗口。



新增查询按钮

2. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。
-



加载 SQL 指令码按钮

此 Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



3. 选取文件名称为 **CreateLocal** 的指令码档案，然后按一下 **开启** 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



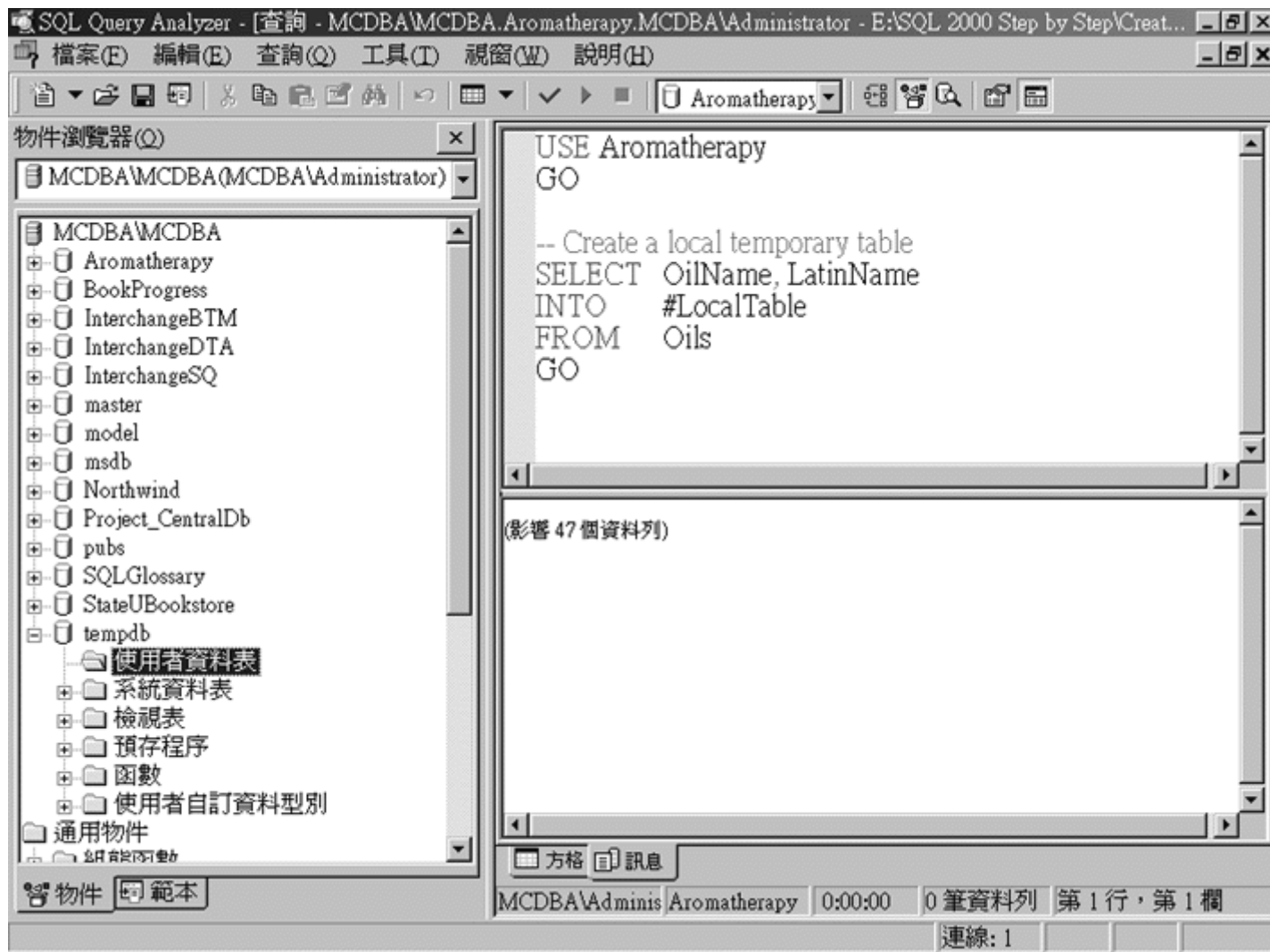
4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



執行查詢按鈕

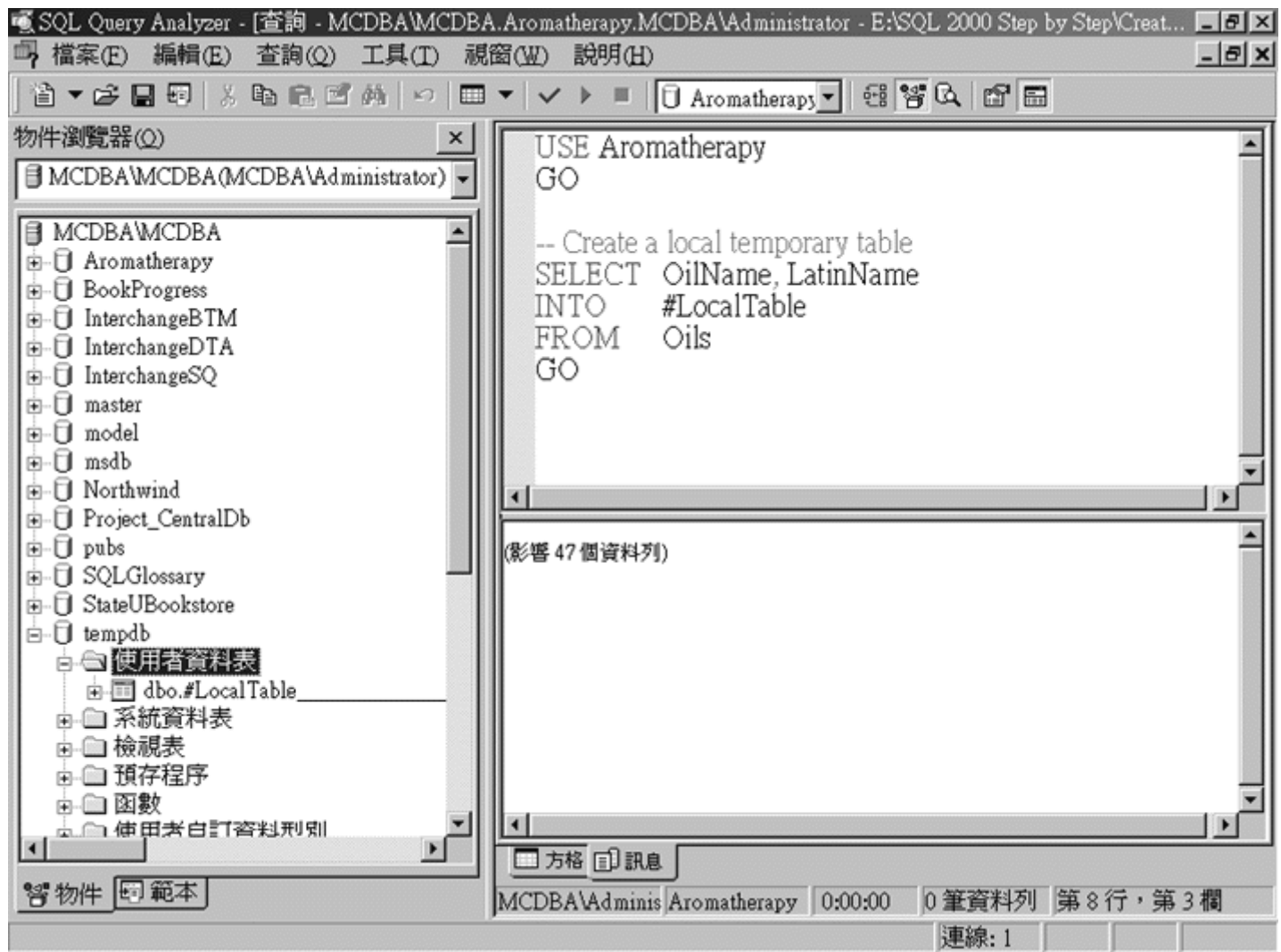
此 Query Analyzer 會建立一個暫存數據表。

5. 自对象浏览器中，选取 **tempdb** 数据库内的 **使用者数据表** 数据夹。



6. 按一下 **F5** 按钮以便将 **对象浏览器** 的内容重新显示，并且将 **使用者数据表** 数据夹展开。

此时 Query Analyzer 会显示 **dbo.#LocalTable** 的暂存资料表。



建立全域暂存数据表

1. 不要将包含 CreateLocal 指令码的窗口关闭，在 Query Analyzer 工具列上按一

下 **新增查询** 按钮以便开启一个新的查询窗口。



新增查询按钮

2. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。



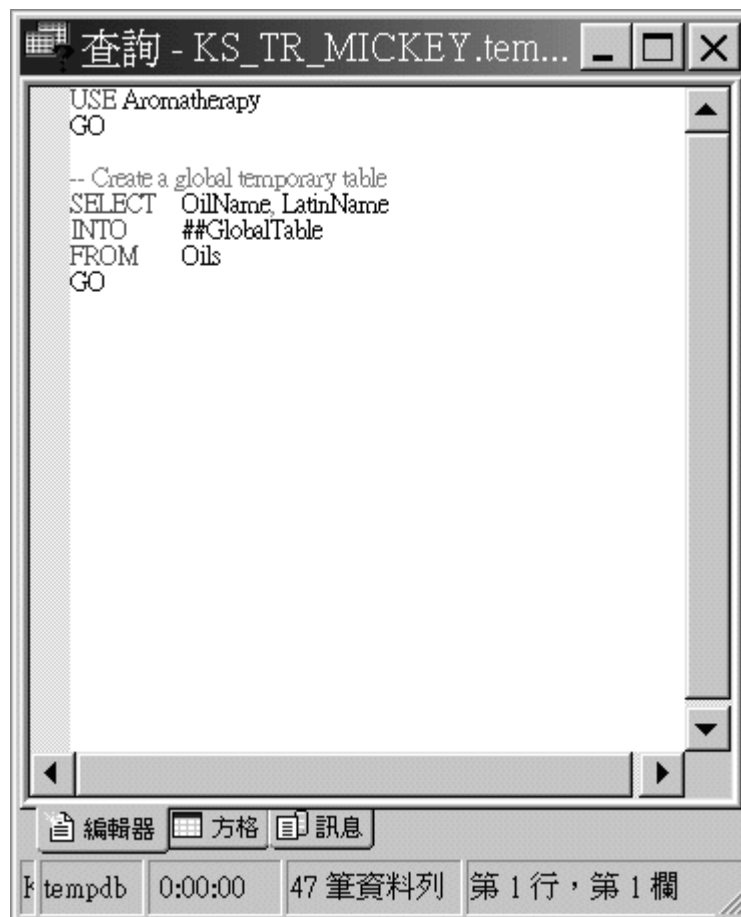
加载 SQL 指令码按钮

此 Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



3. 选取文件名称为 **CreateGlobal** 的指令码档案，然后按一下 **开启** 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



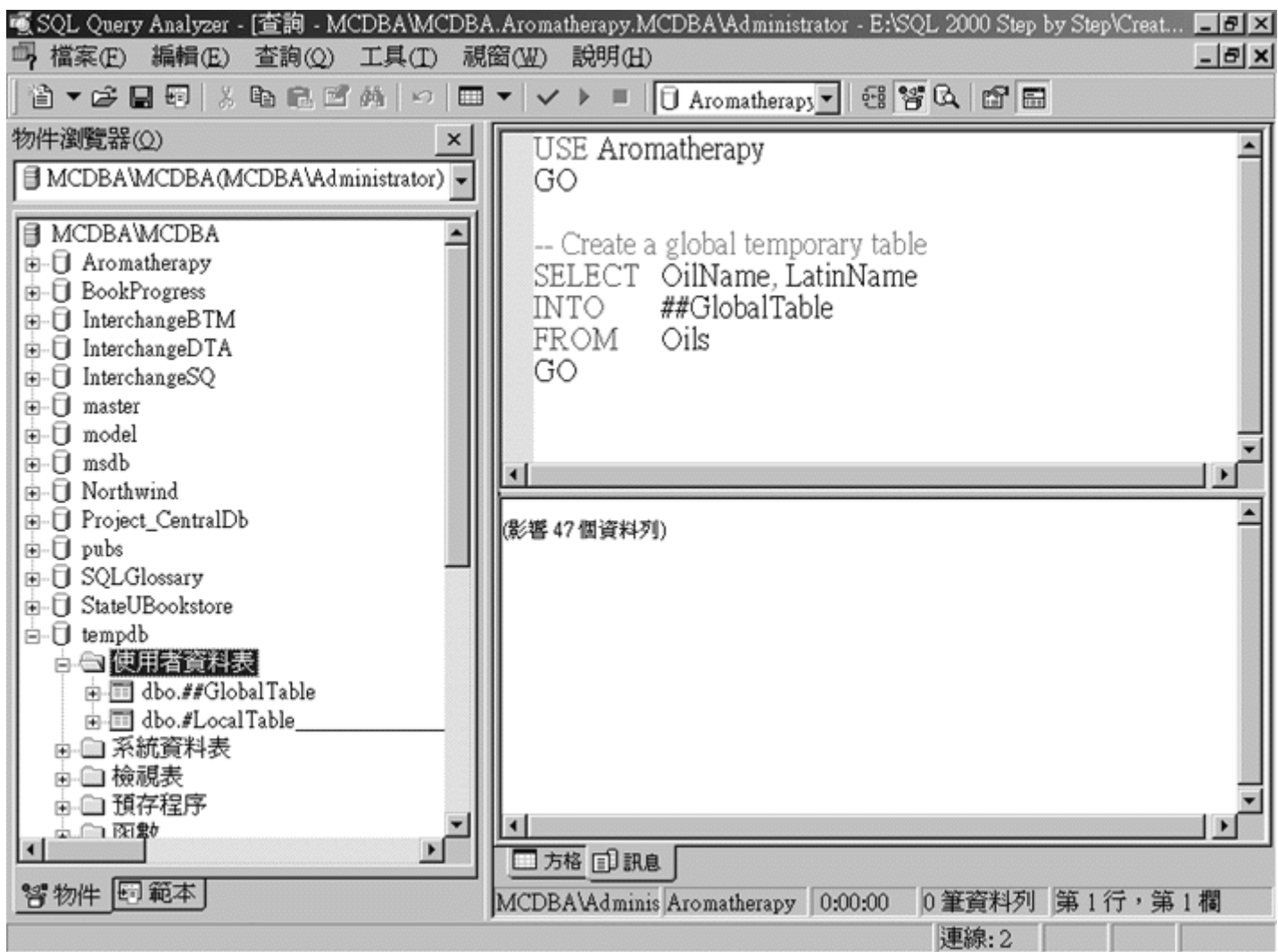
執行查詢按鈕

此 Query Analyzer 會建立一個暫存數據表。

5. 自 [對象瀏覽器](#) 中，選取 **tempdb** 資料庫內的 [使用者數據表](#) 數據夾。

6. 按一下 **F5** 按键以便将 **对象浏览器** 的内容重新显示，并且将 **使用者数据表** 数据夹展开。

此时 Query Analyzer 会显示 **dbo.##GlobalTable** 的暂存资料表。



从目前的工作阶段使用区域暂存数据表

- 1. 选取包含有 **CreateLocal** 指令码内容的查询窗口。

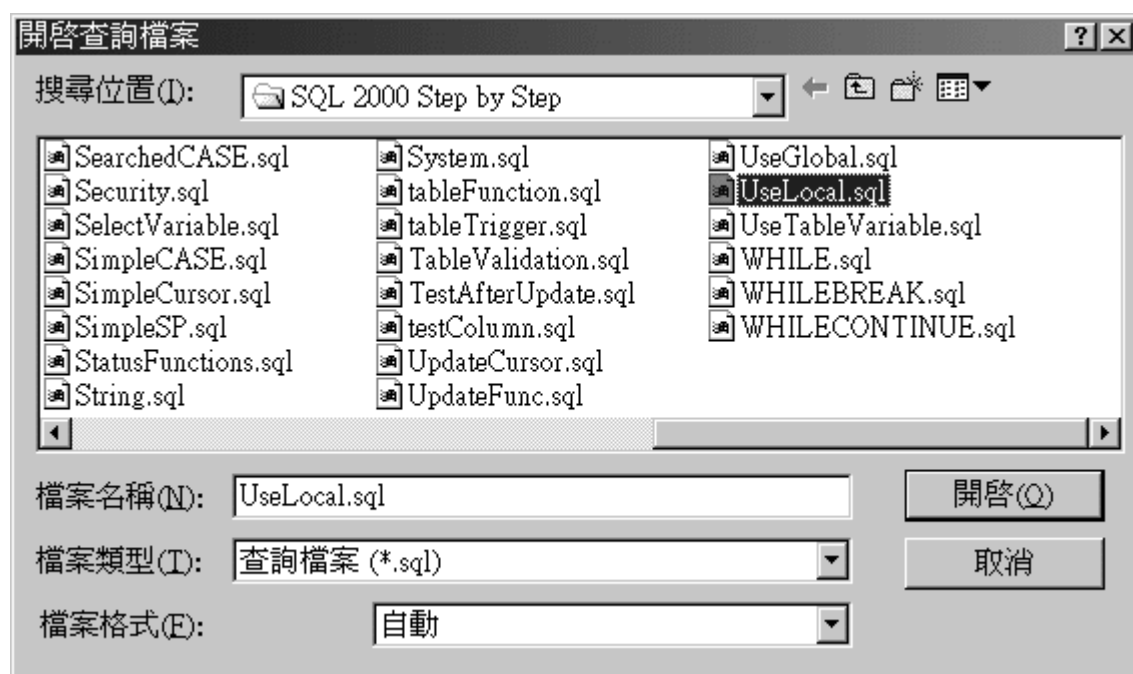


- 2. 在 Query Analyzer 工具列上按一下 **加载 SQL 指令码** 按钮。



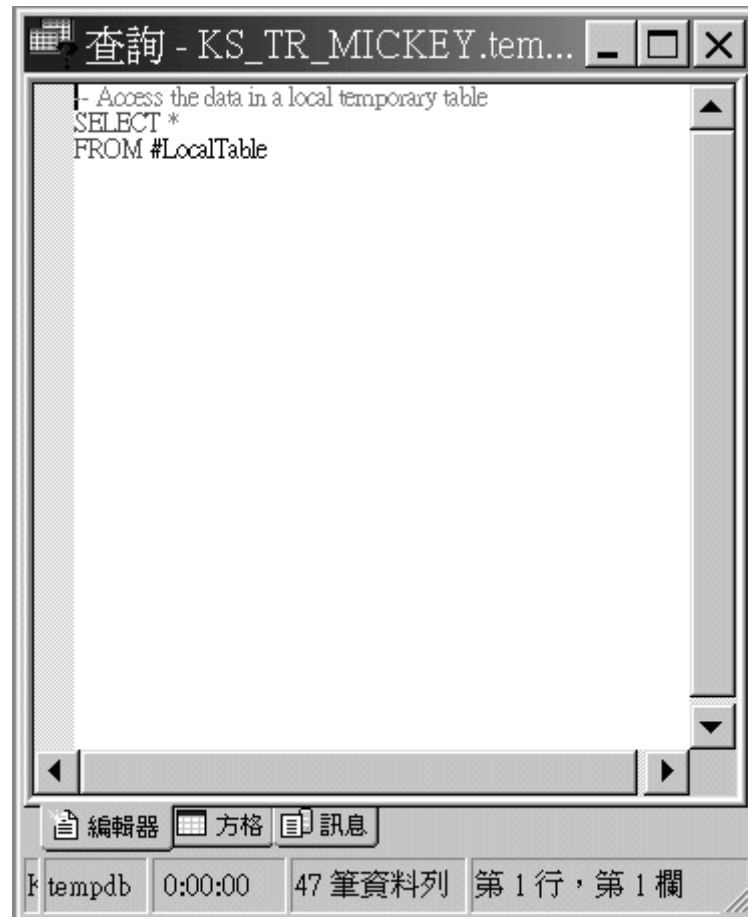
加载 SQL 指令码按钮

此 Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



3. 选取 [UseLocal](#) 指令码档案，并按一下 [开启](#) 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。

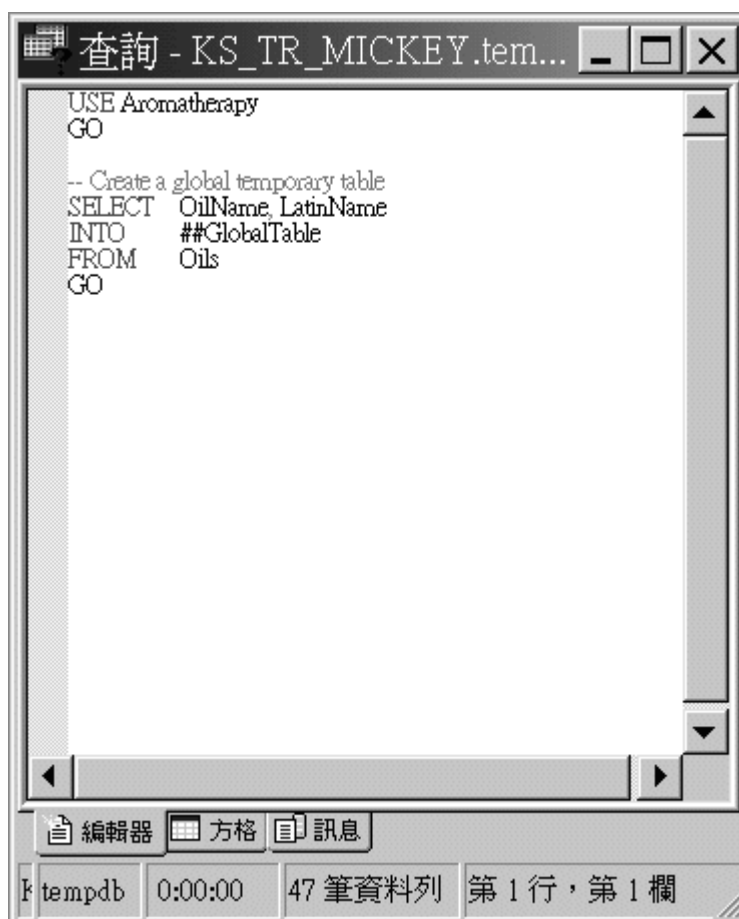


執行查詢按鈕

此时 Query Analyzer 会执行此 SELECT 陈述式。

从目前的工作阶段使用全域暂存数据表

1. 选取包含有 **CreateGlobal** 指令码内容的查询窗口。

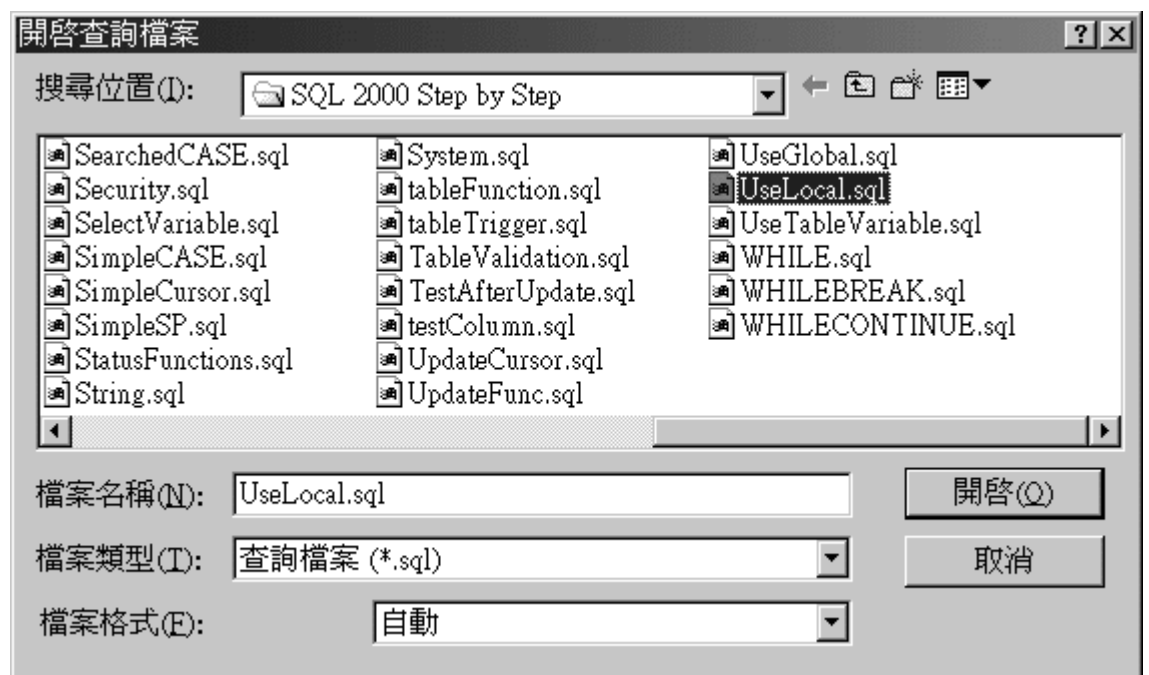


2. 在 Query Analyzer 工具列上按一下 **加载 SQL 指令码** 按钮。



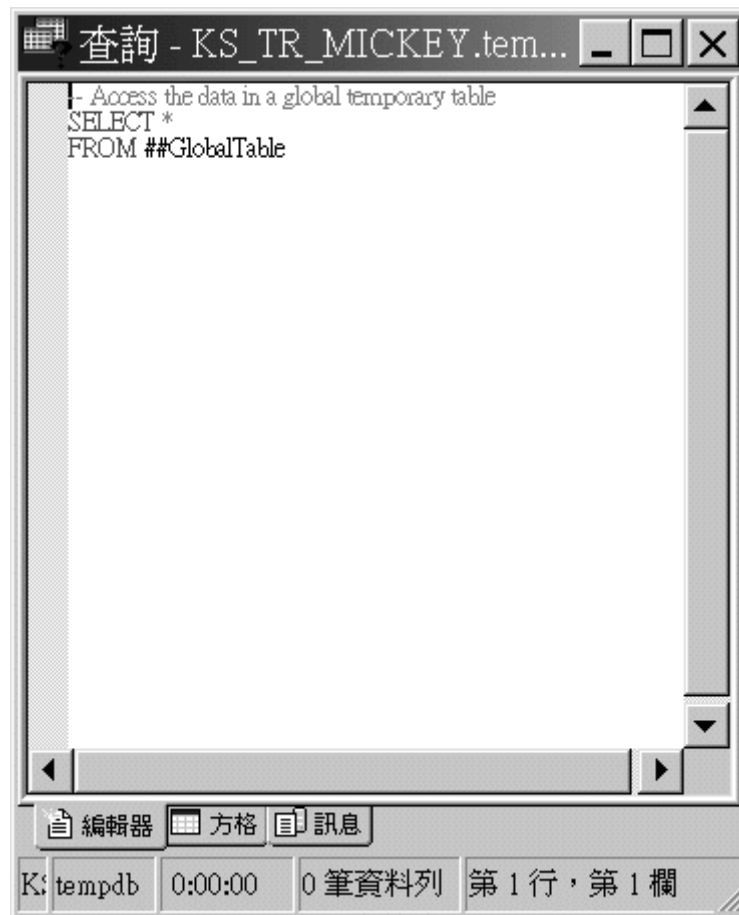
加载 SQL 指令码按钮

此 Query Analyzer 会显示 [开启查询档案](#) 的对话框。



3. 选取 [UseGlobal](#) 指令码档案，并按一下 [开启](#) 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。

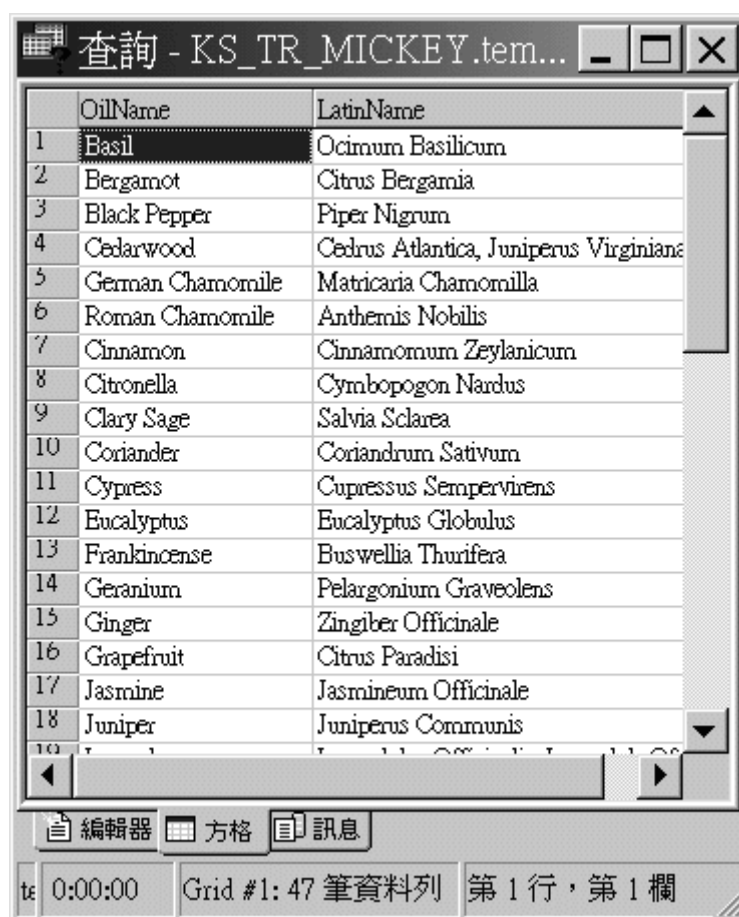


4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



執行查詢按鈕

此时 Query Analyzer 会执行此 SELECT 陈述式。



	OilName	LatinName
1	Basil	Ocimum Basilicum
2	Bergamot	Citrus Bergamia
3	Black Pepper	Piper Nigrum
4	Cedarwood	Cedrus Atlantica, Juniperus Virginiana
5	German Chamomile	Matricaria Chamomilla
6	Roman Chamomile	Anthemis Nobilis
7	Cinnamon	Cinnamomum Zeylanicum
8	Citronella	Cymbopogon Nardus
9	Clary Sage	Salvia Sclarea
10	Coriander	Coriandrum Sativum
11	Cypress	Cupressus Sempervirens
12	Eucalyptus	Eucalyptus Globulus
13	Frankincense	Buswellia Thurifera
14	Geranium	Pelargonium Graveolens
15	Ginger	Zingiber Officinale
16	Grapefruit	Citrus Paradisi
17	Jasmine	Jasminum Officinale
18	Juniper	Juniperus Communis

从其它的工作阶段使用区域暂存数据表

1. 在 Query Analyzer 工具列上按一下 **新增查询** 按钮，以便开启一个新的查询窗口。



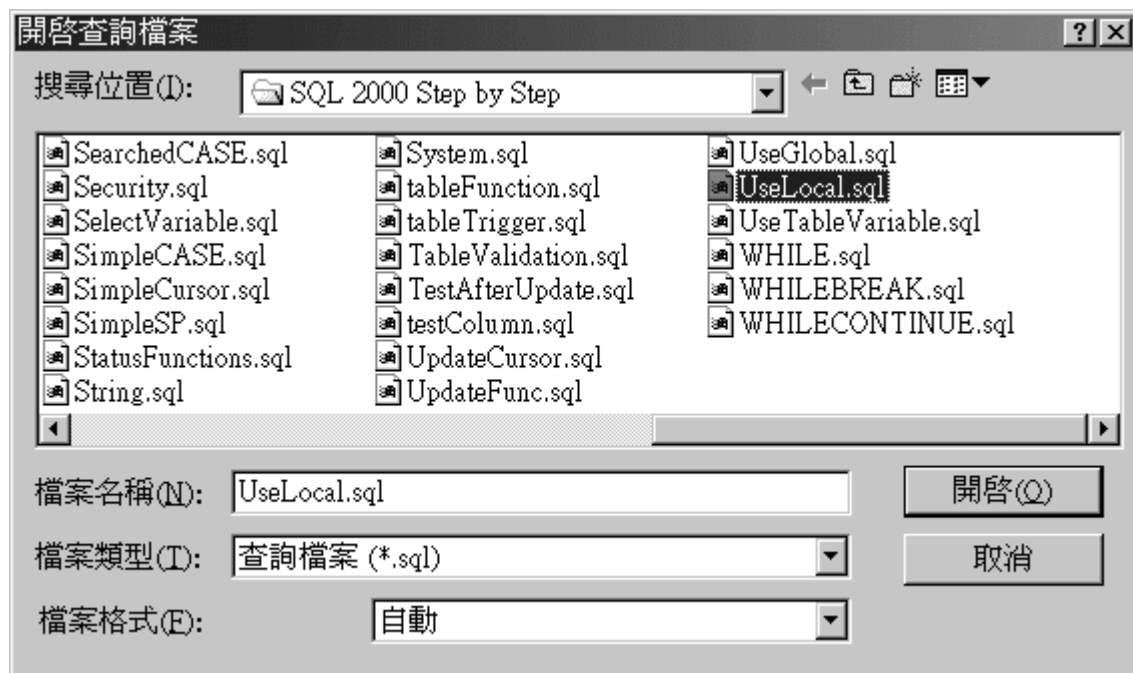
新增查询按钮

2. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。
-



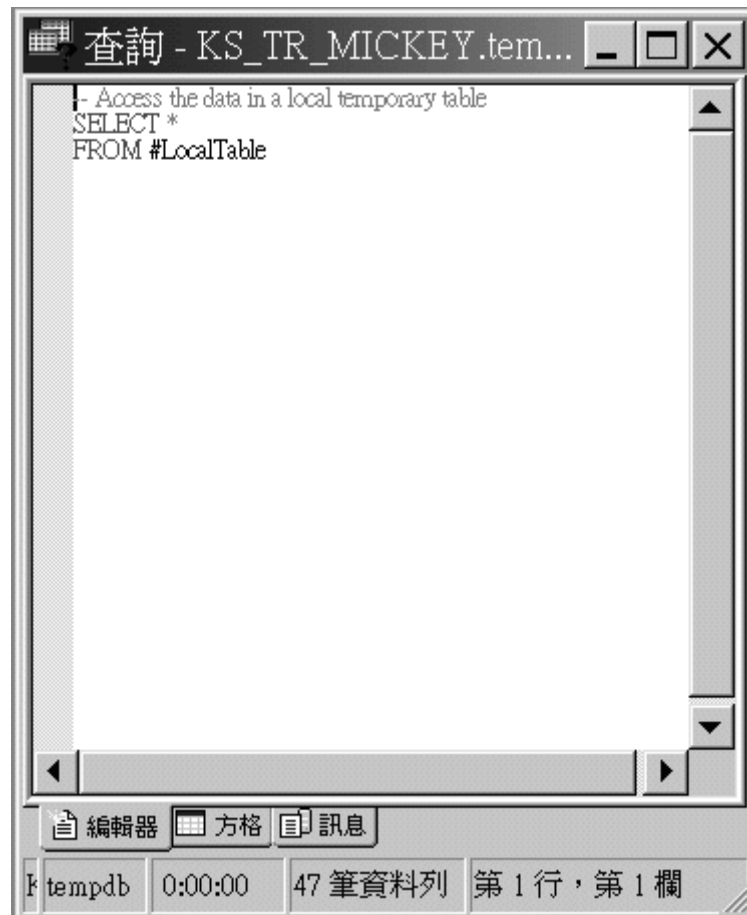
加载 SQL 指令码按钮

此 Query Analyzer 会显示 [开启查询档案](#) 的对话框。



3. 选取文件名称为 **UseLocal** 的指令码档案，然后按一下 **开启** 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



執行查詢按鈕

由于此区域暂存数据表并不是存在新增查询的工作阶段中，因此 Query Analyzer 会显示一个错误的讯息。



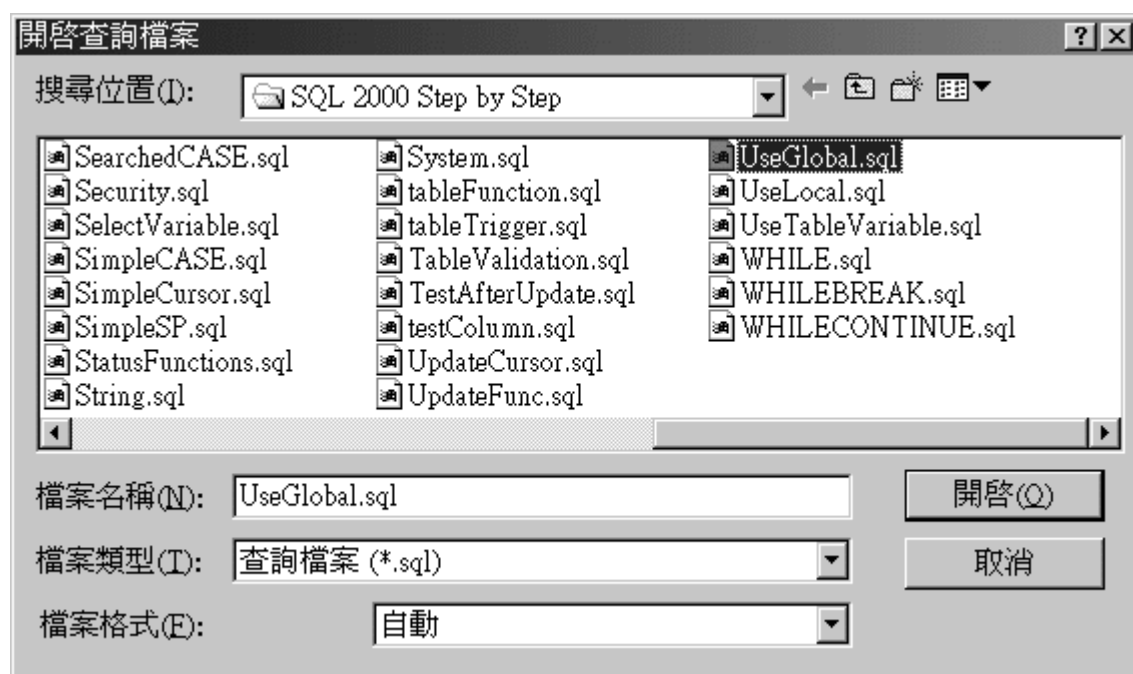
从其它的工作阶段使用全域暂存数据表

1. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。



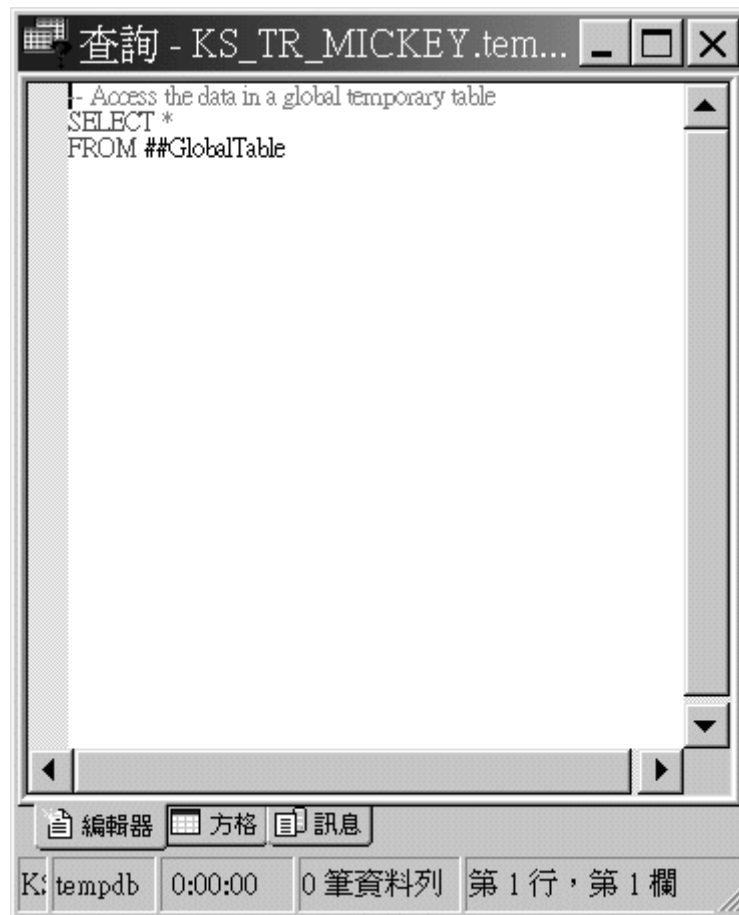
加载 SQL 指令码按钮

此 Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



2. 选取文件名称为 **UseGlobal** 的指令码档案，然后按一下 [开启](#) 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



3. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



執行查詢按鈕

由于此全域暂存数据表是存在于新增查询的工作阶段中，因此 Query Analyzer 会显示其结果的内容。



	OilName	LatinName
1	Basil	Ocimum Basilicum
2	Bergamot	Citrus Bergamia
3	Black Pepper	Piper Nigrum
4	Cedarwood	Cedrus Atlantica, Ju
5	German Chamomile	Matricaria Chamomi
6	Roman Chamomile	Anthemis Nobilis
7	Cinnamon	Cinnamomum Zeyla
8	Citronella	Cymbopogon Nardu
9	Clary Sage	Salvia Sclarea
10	Coriander	Coriandrum Sativum
11	Cypress	Cupressus Sempervi
12	Eucalyptus	Eucalyptus Globulus
13	Frankincense	Buswellia Thurifera
14	Geranium	Pelargonium Gravec
15	Ginger	Zingiber Officinale
16	Grapefruit	Citrus Paradisi
17	Jasmine	Jasmineum Officina

4. 关闭这二个已经开启的查询窗口，并中断与服务器的连结，假如系统有显示一个
- 是否要将已改变的内容储存的讯息时，请您选择 否。

变数

虽然暂存数据表非常好用，但是没有一套程序语言只使用暂存数据表而不使用变量的，在 SQL Server 的程序设计过程中，我们会将暂时使用的值储存在变量中。

认识变数

变量是以前缀「@」来作为分辨的字符，如 @myVariable。和暂存数据表一样，变量也拥有二种格式：区域(local)和全域(global)，其中全域性变量是以「@@」来作为前置字，如 @@VERSION。

变量和暂存数据表这二者之间还是有差异的。所有的全域变量是由 SQL Server 所定义的，而不是由您所定义的，因此区域变量可使用的范围就会比较小：它只能在批次操作，或在定义了变量的程序中使用。

区域变量

区域变量是使用 DECLARE 陈述式建立的，其语法如下所示：

```
DECLARE @local_variable data_type
```

其中 local_variable 必须要遵守一般数据库的识别项规则；[data_type 是指除了 text、ntext 或 image 之外的任意的系统数据型别。您可以在 DECLARE 陈述式内使用多个区域变量，而每一个区域变量必须以逗号「，」隔开。

```
DECLARE @var1 int, @var2 int
```

大部份的资料型别是属于 **纯量 (scalar)**，换句话说，它们包含单一值，例如一个数值或一个字符串。**SQL Server 2000** 提供了一个新功能，那就是您可以将变量宣告为数据表的数据型别。建立数据表的数据型别变量其语法如下所示：

```
DECLARE @local_variable TABLE ({table_definition})
```

在此范例中，**table_definition** 就如同 **CREATE TABLE** 陈述式中的使用方式一样，此数据表宣告包含数据行定义、名称、数据型别、以及条件约束。但是允许使用的条件约束型别只有 **PRIMARY KEY**、**UNIQUE KEY**、**NULL** 和 **CHECK**。

提示

另一种好用的数据型别是 **sql_variable**，该型别可以储存任何的数据，这样可以让您在程序的进行中分配不同的数据型别给区域变量。

当一个区域变量建立之后是以 **NULL** 值来作为初始值，您可以依照下列的方式来分配变量中的值：

- 使用 **SET** 指令，并且配合常数或表达式来设定：

```
SET @myCharVariable ='Hello, World'
```

- 使用 **SELECT** 指令，并且配合常数或表达式来设定：

```
SELECT @myCharVariable ='Hello, World'
```

- 使用 **SELECT** 指令，并且配合其它 **SELECT** 指令来设定：

```
SELECT @myCharVariable =MAX(OilName) FROM Oils
```

- 使用 **INSERT INTO** 指令，并且配合数据表型别变量来设定：

```
INSERT INTO @myTableVariable  
SELECT *FROM Oils
```

请注意在第三个格式中（使用 **SELECT** 指令和其它的 **SELECT** 指令），其指派运算符是取代第二个 **SELECT** 关键词。在最后一个范例所示的 **INSERT INTO...SELECT** 的 **INSERT INTO** 指令的语法当中，您也可以使用 **INSERT INTO...VALUES** 语法：

```
INSERT INTO @myTableVariable  
  
VALUES (' The Value')
```

全域变数

全域性变量是使用二个「@」符号来作为识别的依据，它是由 SQL Server 所建立而不是您自己所建立的。

许多的全域变量都可以使用，大部份的全域变量会提供关于 SQL Server 目前状态的资料，您可以在 [对象浏览器](#) 内的 [组态函数](#) 数据夹中找到。

组态变数

一般最常使用的组态函数列在表 25-1 中，它们提供目前 SQL Server 设定值以及参数的数据。

变数	值
@@CONNECTIONS	传回联机或尝试联机的次数，自上次启动 SQL Server 时算起。
@@DATEFIRST	传回 SET DATEFIRST 参数的现行值，表示指定的每周第一天：1 代表星期一，7 代表星期日。
@@DBTS	传回现行数据库中现行 timestamp 数据型别的值。
@@LANGID	传回使用中语言的本机语言识别项（ID）。

@@LANGUAGE	传回目前使用中的语言名称。
@@OPTIONS	传回现行 SET 选项的相关信息。
@@SERVERNAME	传回执行 SQL Server 的本机服务器名称。
@@VERSION	传回现行 SQL Server 安装的日期、版本和处理器类型。

表 25-1 组态变数

统计变数

统计变量是提供自 SQL Server 最后一次启动之后至今的数据，一般最常使用的统计变量列在表 25-2 中。

变数	值
@@CPU_BUSY	传回 CPU 花在工作上的时间，并且以毫秒为单位，自上次启动 SQL Server 时算起。
@@IDLE	传回 SQL Server 已经闲置（Idle）的时间，自上次启动算起。
@@IO_BUSY	传回 SQL Server 已经花在执行输入和输出作业的时间，自上次启动算起。
@@TOTAL_ERRORS	传回自上次启动后，SQL Server 已进行的磁盘读取/写入错误的数目。
@@TOTAL_READ	传回自上次启动后，SQL Server 读取磁盘的数目。
@@TOTAL_WRITE	传回自上次启动后，SQL Server 写入磁盘的数目。

表 25-2 统计变数

系统变量

表 25-3 所示为常用的系统变量列表，它们提供了执行运算并传回 SQL Server 中的值、对象与设定信息。

变数	值
@@IDENTITY	传回上次插入数据库中 Indetity 数据行的值。
@@ROWCOUNT	传回受到上一个陈述式影响的数据列数目。

表 25-3 系统变量

使用变量

变量可以藉由 Transact-SQL 语言使用于表达式中，但是它们不能用于取代对象名称或关键词，下面所列出的是正确的表示方式：

```
DECLARE @theOil char(20)

SET @theOil ='Basil '

--正确执行

SELECT OilName, Description
```

```
FROM Oils  
  
WHERE OilName = @theOil
```

而下列的 **SELECT** 陈述式的表示方式是错误的：

```
DECLARE @theCommand char(10), @theField char(10)  
  
SET @theCommand = 'SELECT'  
  
SET @theField = 'OilName'  
  
--执行失败  
  
@theCommand *FROM Oils  
  
--正确执行  
  
SELECT @theField from Oils
```

宣告一个区域变量

1. 在 **Query Analyzer** 工具列上按一下 **新增查询** 按钮，以便开启一个新的查询窗口。



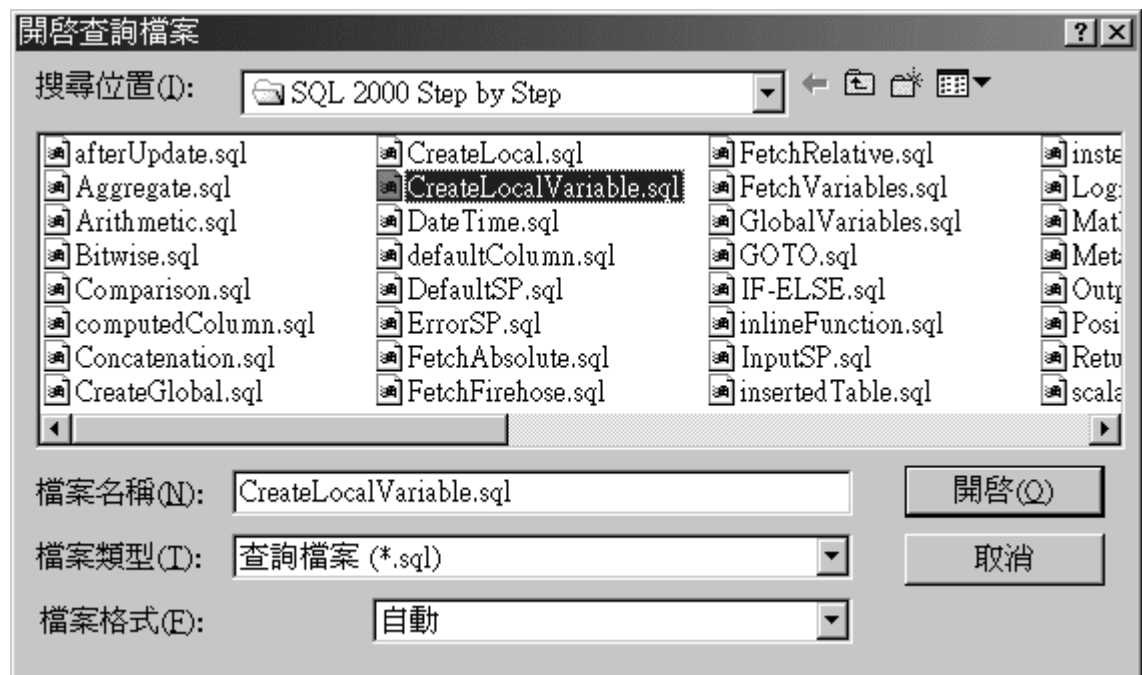
新增查询按钮

-
2. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。



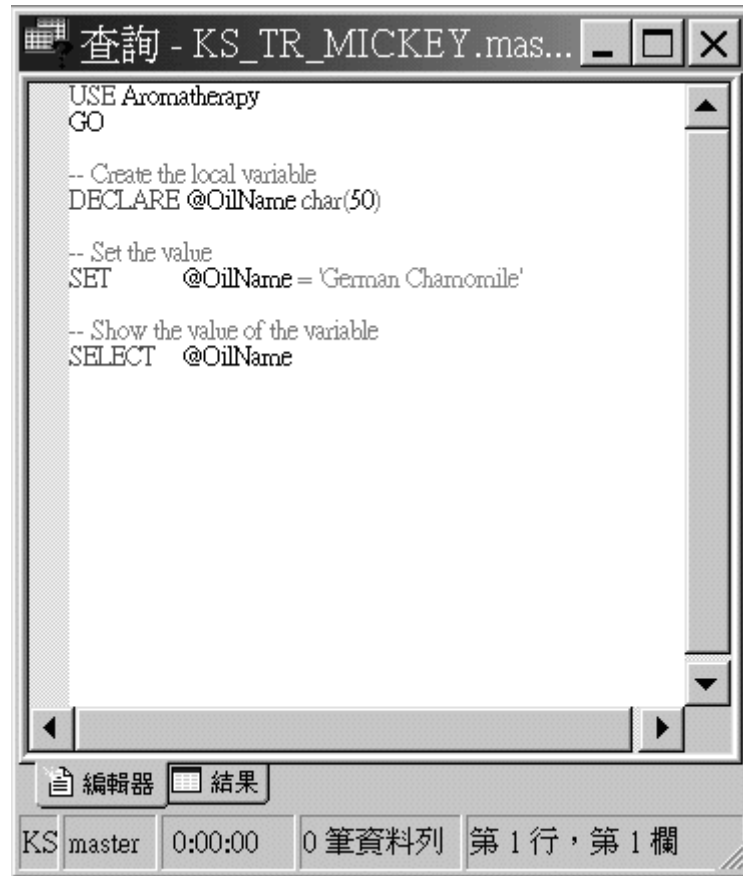
加载 SQL 指令码按钮

此 Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



3. 选取文件名称为 **CreateLocalVariable** 的指令码档案，然后按一下 **开启** 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。

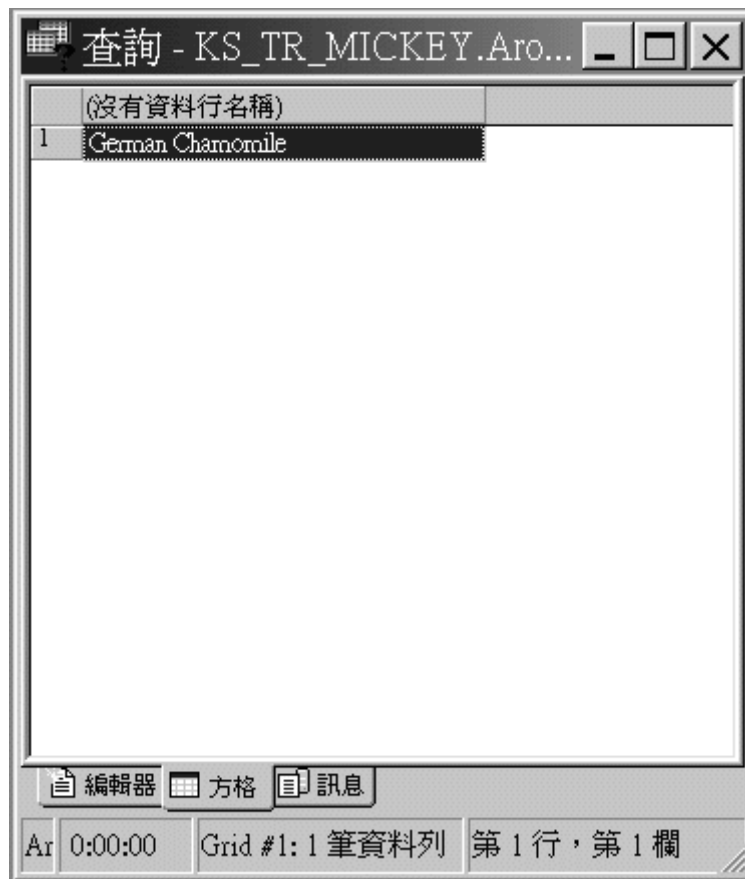


4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



執行查詢按鈕

此 Query Analyzer 會執行此指令碼並且顯示其結果。



使用 **SELECT** 来分配变量值

1. 在查询窗口中选取 **編輯器** 窗格，或者假如您在前一个练习中已将查询窗口关闭时，您可以开启一个新的查询窗口。



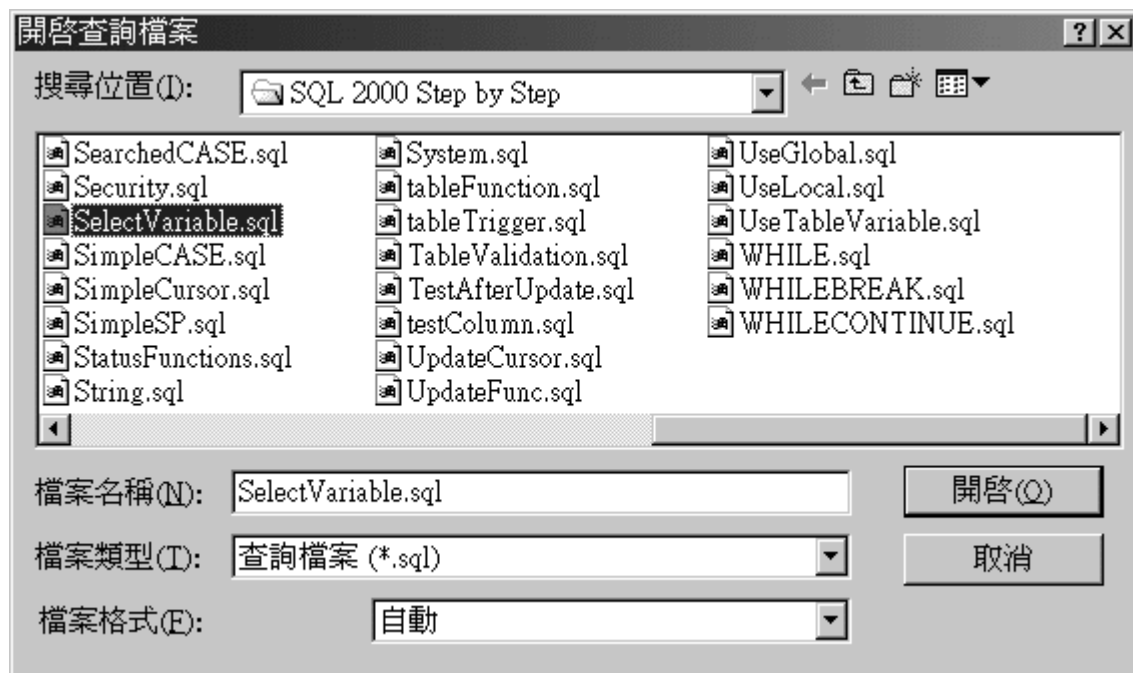
新增查询按钮

2. 在 Query Analyzer 工具列上按一下 **加载 SQL 指令码** 按钮。
-



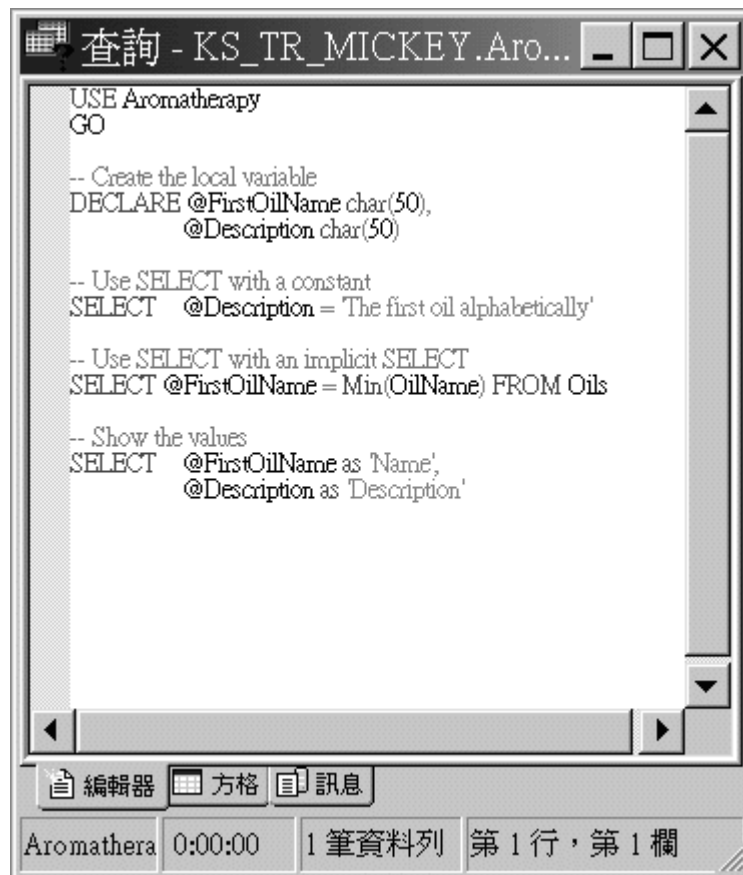
加载 SQL 指令码按钮

此 Query Analyzer 会显示一个开启查询档案的对话框。



3. 选取文件名称为 **SelectVariable** 的指令码档案，然后按一下 **开启** 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。

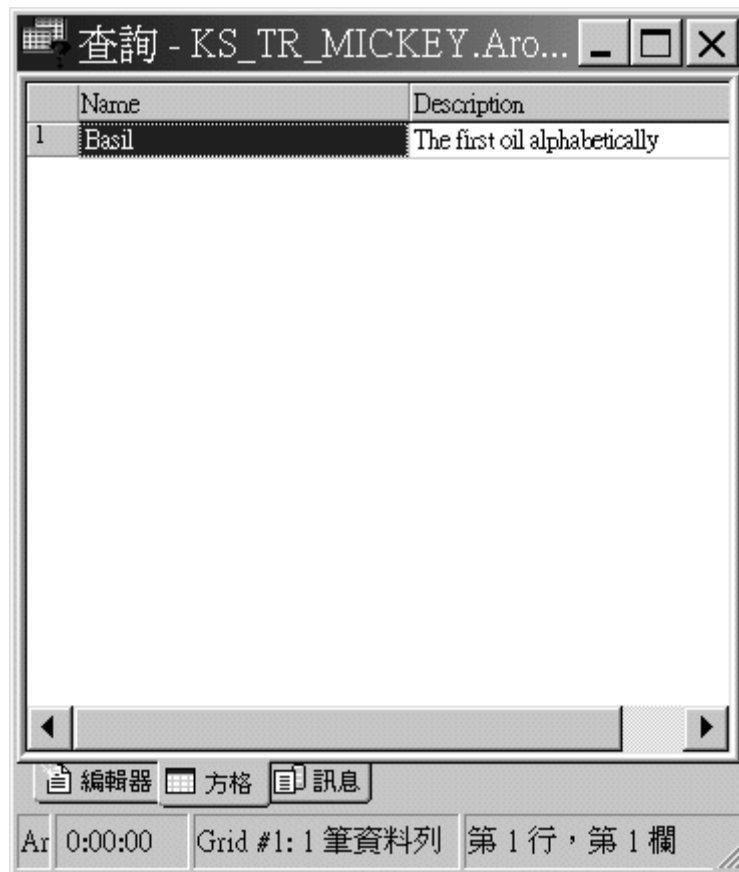


4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



執行查詢按鈕

此 Query Analyzer 會執行此指令碼並且顯示其結果。



使用数据表变量

1. 在查询窗口中选取 [编辑器](#) 窗格，或者假如您在前一个练习中已将查询窗口关闭时，您可以开启一个新的查询窗口。



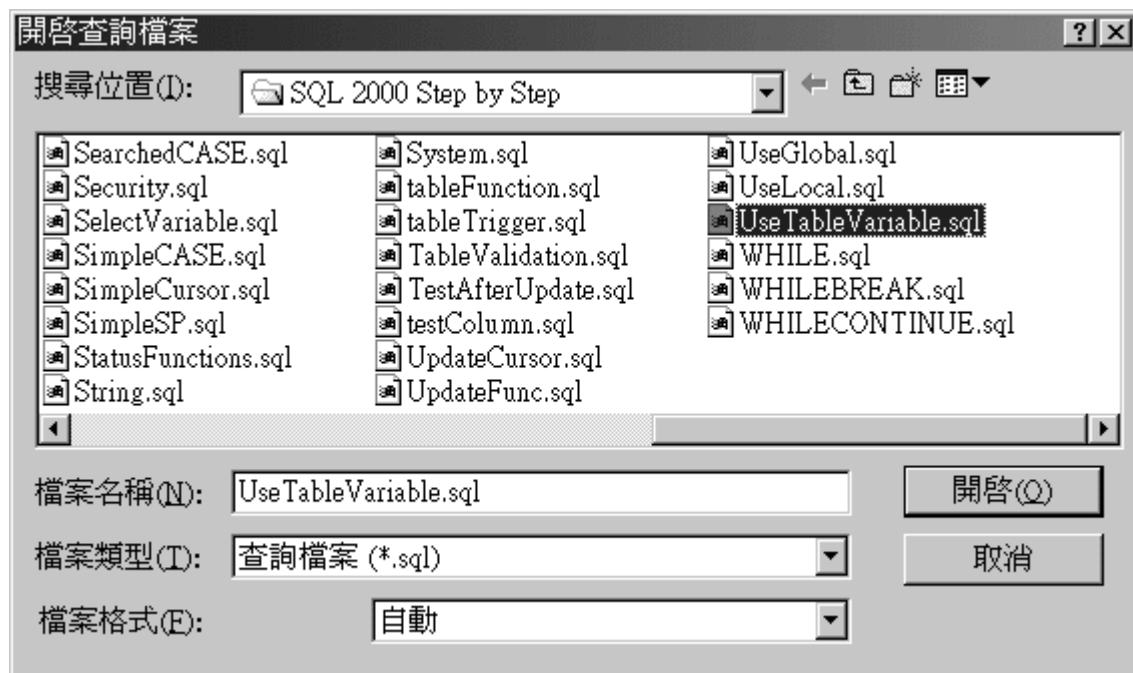
新增查询按钮

2. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。
-



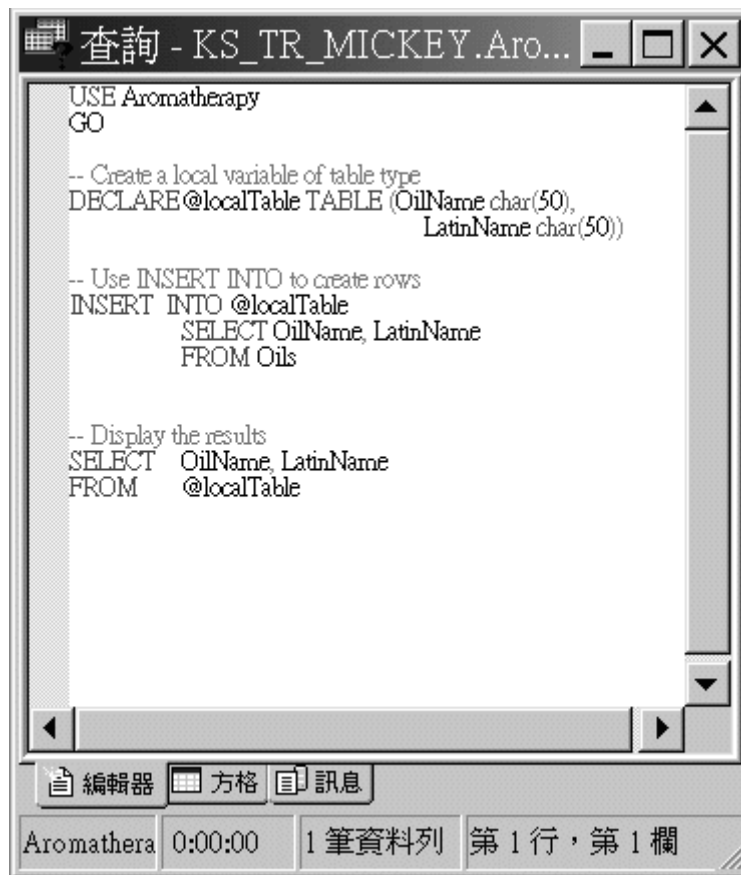
加载 SQL 指令码按钮

此 Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



3. 选取文件名称为 **UseTableVariable** 的指令码档案，然后按一下 **开启** 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



執行查詢按鈕

此 Query Analyzer 會執行此指令碼並且顯示其結果。

查詢 - KS_TR_MICKEY.Aro...		
	OilName	LatinName
1	Basil	Ocimum Basilicum
2	Bergamot	Citrus Bergamia
3	Black Pepper	Piper Nigrum
4	Cedarwood	Cedrus Atlantica, Ju
5	German Chamomile	Matricaria Chamomi
6	Roman Chamomile	Anthemis Nobilis
7	Cinnamon	Cinnamomum Zeyla
8	Citronella	Cymbopogon Nardu
9	Clary Sage	Salvia Sclarea
10	Coriander	Coriandrum Sativum
11	Cypress	Cupressus Sempervi
12	Eucalyptus	Eucalyptus Globulus
13	Frankincense	Buswellia Thurifera
14	Geranium	Pelargonium Gravec
15	Ginger	Zingiber Officinale
16	Grapefruit	Citrus Paradisi
17	Jasmine	Jasmineum Officina

編輯器 方格 訊息

A 0:00:00 Grid #1: 47 筆資料列 第 1 行，第 1 欄

使用全域变量来显示服务器的相关信息

1. 在查询窗口中选取 編輯器 窗格，或者假如您在前一个练习中已将查询窗口关闭
- 时，您可以开启一个新的查询窗口。



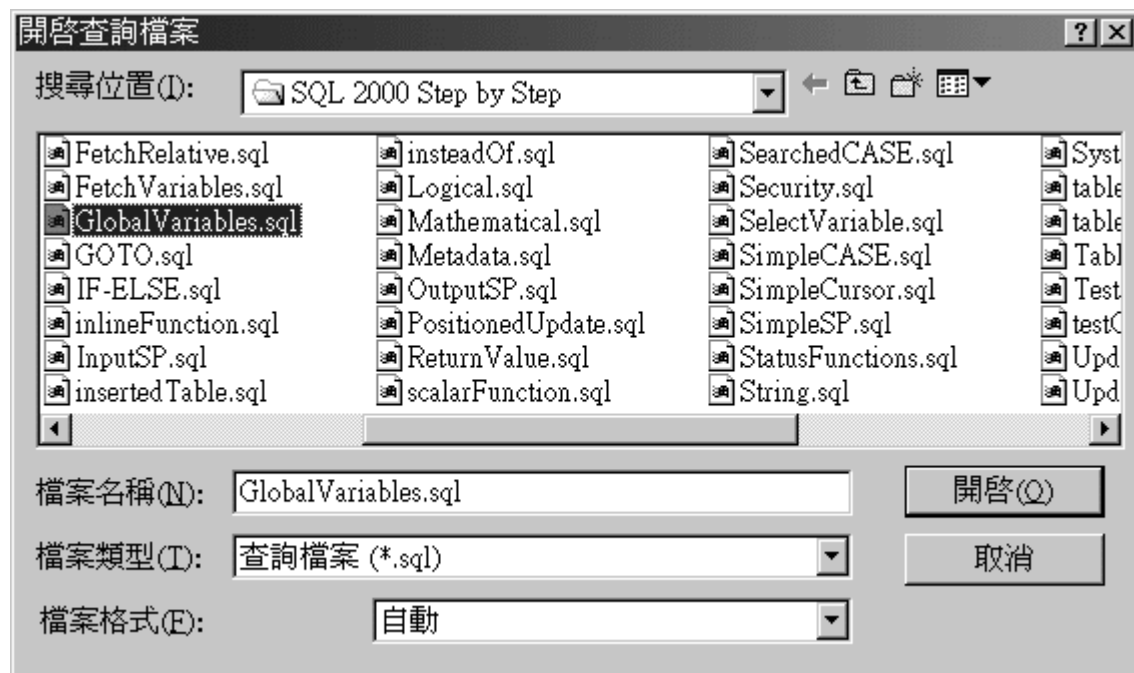
新增查询按钮

2. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。
-



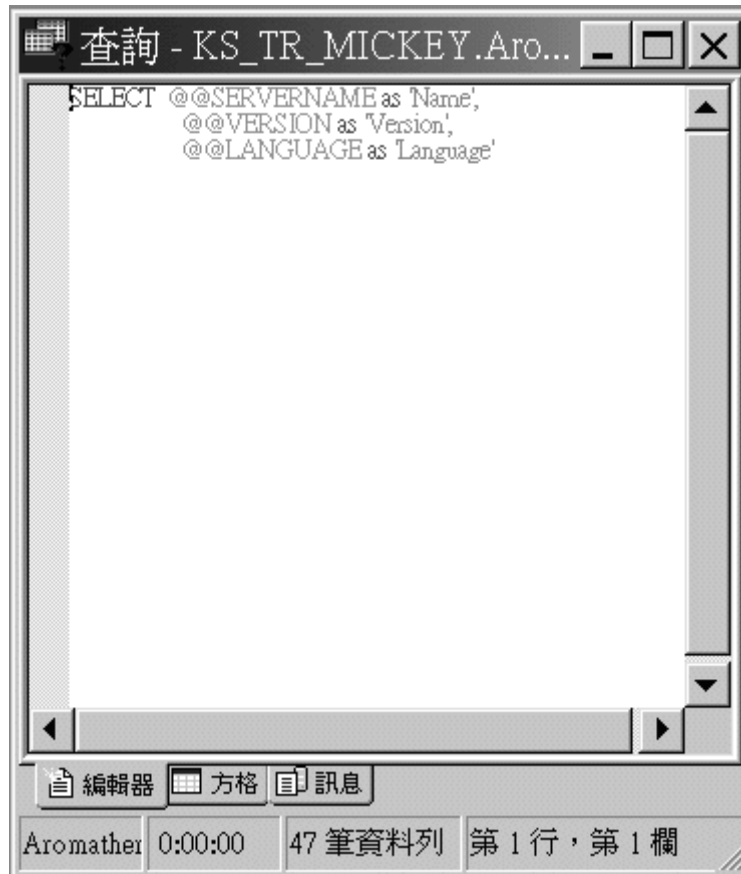
加载 SQL 指令码按钮

此 Query Analyzer 会显示 [开启查询档案](#) 的对话框。



3. 选取文件名称为 **GlobalVariables** 的指令码档案，然后按一下 **开启** 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



執行查詢按鈕

此 Query Analyzer 會執行此指令碼並且顯示其結果。



本章总结

要执行的工作	SQL 语法
建立区域暂存数据表	<pre>CREATE TABLE @table_name (table_definition)</pre>
建立全域暂存数据表	<pre>CREATE TABLE @@table_name (table_definition)</pre>

宣告纯量区域变量	DECLARE @variable_name
宣告数据表数据型别的区域变量	DECLARE @variable_name TABLE (table_definition)
设定纯量变量值的命令	SET @variable_name = using the SET constant_or_expression
使用 SELECT 命令和常数来设定纯量变量的值	SELECT @variable_name = constant_or_expression
使用 SELECT 来设定变量值	SELECT @variable_name = animplicit column_expression FROM table_name
将数据列加入数据表数据型别的变量	INSERT INTO @variable_name select_statement

26. 控制执行流程

在本章中，您将学习到：

- 使用 **IF...ELSE** 命令控制流程的处理。
- 使用简单式 **CASE** 函数基于某值符合条件判断以传回结果。
- 使用搜寻式 **CASE** 函数基于布尔表达式以传回结果。
- 使用 **GOTO** 命令以移转执行方向至其它陈述式。
- 使用 **WHILE** 循环以执行陈述式，或执行陈述式区块数次。
- 使用 **BREAK** 子句以中断 **WHILE** 循环的执行。
- 使用 **CONTINUE** 子句以回到 **WHILE** 循环的起始位置继续开始执行。

除非您特别指定，否则 **Transact-SQL** 陈述式会将您的指令码从头到尾执行一次。但是有时这种情形是您不希望的，有时您会希望当某个条件式成立时才会执行某些特定的陈述式，有时您会希望某些陈述式执行数次或者是重复执行直到某些条件式成立时才停止执行。

Transact-SQL 控制流程命令可以让您的希望梦想成真，我们将在本章中说明控制流程的命令。

陈述式区块

当您开始要处理 **Transact-SQL** 所要执行的陈述式内容时，您可以将一组陈述式的内容当作一个区块，这样会比较方便。**Transact-SQL** 提供了 **BEGIN...END** 可以让您达成这个目的。

将任何一组控制流程命令配合使用 **BEGIN** 子句时，**Transact-SQL** 会执行在 **BEGIN** 与对称的 **END** 之间所有的陈述式。

您可以在此区块中加入任意的 **Transact-SQL** 陈述式，这其中包括了其它的 **BEGIN..END** 区块在内。话虽如此，在使用上还是有一些限制，那就是您不能将 **CREATE DEFAULT**、**CREATE PROCEDURE**、**CREATE RULE**、**CREATE TRIGGER**、以及 **CREATE VIEW** 和其它的陈述式联合使用。同样的，您也不能修改数据结构，然后在相同的区块中引用新的数据行。

条件式处理

我们在设计流程控制的第一个范例就是要建立基于布尔代数（**Boolean**）表达式执行的陈述式区块。（请记住所谓的布尔表达式其值不是为 **TRUE** 就是为 **FALSE**。）

IF...ELSE

IF 陈述式是在条件式流程控制命令中最简单的一种。假如在 **IF** 命令内的布尔表达式其值为 **TRUE** 时，则在它之后的陈述式或陈述式区块将会被执行。假如布尔表达式其值为 **FALSE** 时，那么在它之后的陈述式或陈述式区块将不会被执行。

ELSE 是一种选择性的命令，它可以让您当布尔表达式指定陈述式或陈述式区块其值为 **FALSE** 时，指定要执行的陈述式或陈述式区块内的陈述式。举例来说，假如 **@test** 的值为 **true** 时，那么 **Transact-SQL** 命令会传回「**It's true**」；假如 **@test** 为「**false**」时，那 **Transact-SQL** 命令会传回「**It's false**」。

```
IF @test

    SELECT 'It 's true '

ELSE

    SELECT 'It 's false '

```

提示

IF...ELSE 陈述式可以包含在其它 IF...ELSE 陈述式中而成为一个巢状结构，这与其它程序语法所提供的 IF..ELSEIF..ELSE 陈述式的结构是相似的，要使用这种技巧时千万要特别小心。

使用 IF...ELSE 条件式控制执行流程

1. 在 Query Analyzer 工具列上按一下 **新增查询** 按钮，以便开启一个新的查询窗口。



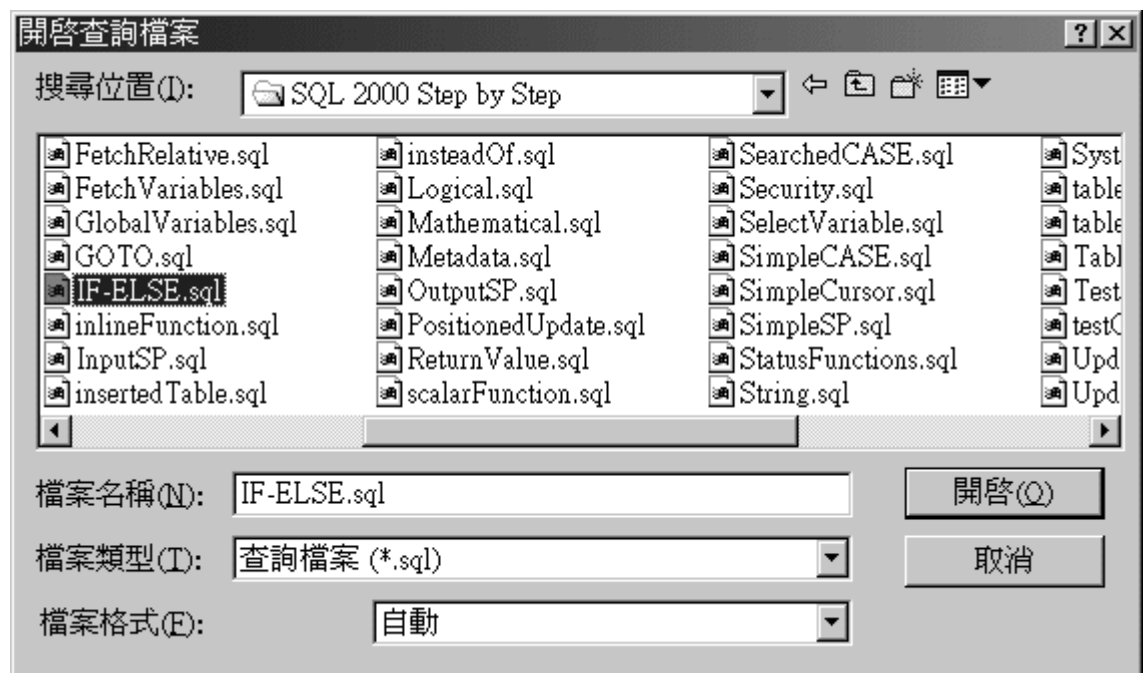
新增查询按钮

2. 在 Query Analyzer 工具列上按一下 **加载 SQL 指令码** 按钮。



加载 SQL 指令码指令按钮

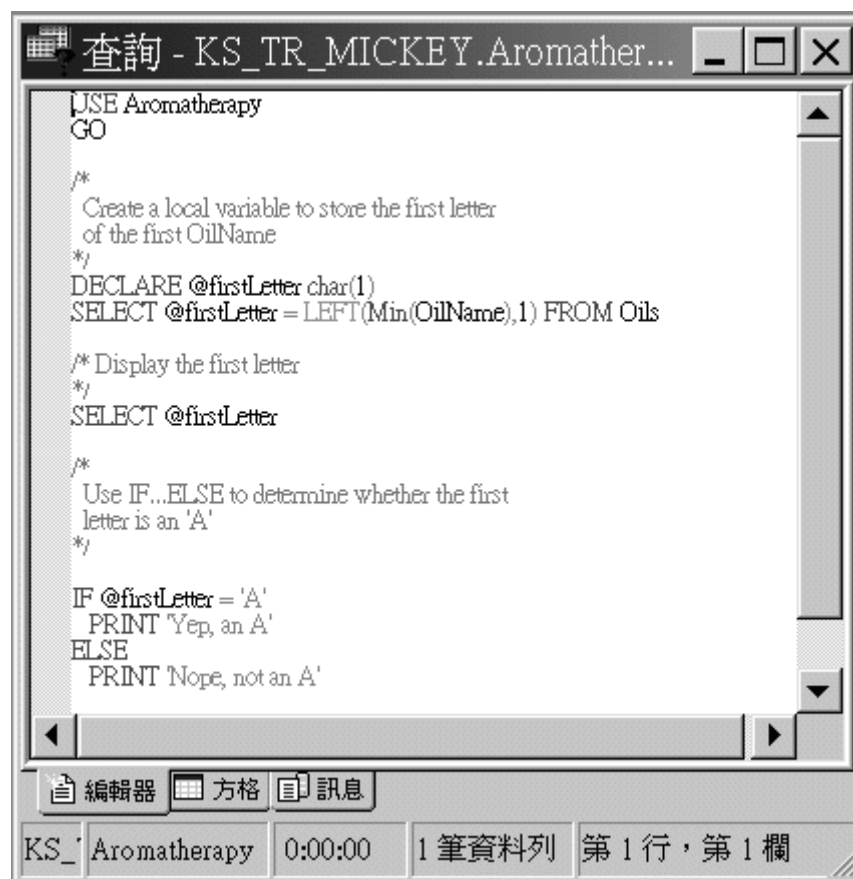
此 Query Analyzer 会显示 [开启查询档案](#) 的对话框。



3. 选取 [SQL 2000 Step by Step](#) 数据夹，并选取 [IF-ELSE](#) 指令码档案，然后按一

下 [开启](#) 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



提示

该指令码会使用 **PRINT** 命令是为了要在 **查詢** 窗口内的 **訊息** 卷标页显示讯息。当

您第一次设计指令码时其 **PRINT** 命令是非常好用的命令，但是当您实际在设计应

用程序时则最好少用。

4. 在 Query Analyzer 的工具列上按一下 **执行查询** 按钮。



执行查询按钮

此 Query Analyzer 会执行此查询并且在 [方格](#) 标签页中显示结果。



5. 选取 **訊息** 卷标页。

此 Query Analyzer 会显示 IF...ELSE 陈述式的执行结果。



CASE

在大多数的程序语言中，假如在单一个陈述式内包含多个布尔表达式时，您就可以使用 **CASE** 来取代 **IF** 陈述式。在 **SQL Server** 中，**CASE** 是一种 [函数](#)，而不是一个命令。通常它用于代替某部份 **SELECT** 或 **UPDATE** 命令的执行。

包含 **CASE** 的陈述式可以有二种语法撰写方式，它是依据其表达式是否有改变来决定要使用哪一种语法。其中最简单的形式是采用如下所示的方式：

```
value = expression
```

其中 **value** 可以为任何复杂的值，您可以使用常数、数据行名称或者是复杂的表达式，您可以依照您的需求来设定。比较运算符通常是使用等于。最简单的 **CASE** 语法如下所示：

```
CASE value

WHEN expression_one THEN result_expression_one

WHEN expression_two THEN result_expression_two

:

WHEN expression_n THEN result_expression_n

[ELSE else_result_expression ]

END
```

在这个 **CASE** 形式中，其中 **result_expression** 只有在假如 **WHEN** 关键词后面的表达式是等于您所指定的值时才会被传回。您可以在表达式中拥有多个 **WHEN** 子句。**ELSE** 子句是一个选择性项目，只有当所有的 **WHEN** 子句的值为 **FALSE** 时才会被执行。

在一些不同的值中取一个值是非常普遍的事，但是有时候您需要更多的弹性。在本例中，您可以

使用 **Transact-SQL** 来呼叫搜寻式 **CASE** 语法，如下所示：

```
CASE

WHEN Boolean_expression_one THEN result_expression_one

WHEN Boolean_expression_two THEN result_expression_two

:

WHEN Boolean_expression_n THEN result_expression_n

[ELSE else_result_expression ]

END
```

在此种 **CASE** 形式中，您可以在每一个 **WHEN** 子句中指定整个布尔表达式，比起您依赖简单式的等式表达式还要好。请注意假如不只一个 **Boolean_expression** 为 **true** 时，**Transact-SQL** 将会传回第一个 **result_expression**，并且直接跳至 **END** 后面继续执行。

使用简单式 **CASE**

1. 在 **查询** 窗口内选取 **编辑器** 标签页，并且在 **Query Analyzer** 工具列上按一下 **清除窗口** 按钮。





清除窗口按钮

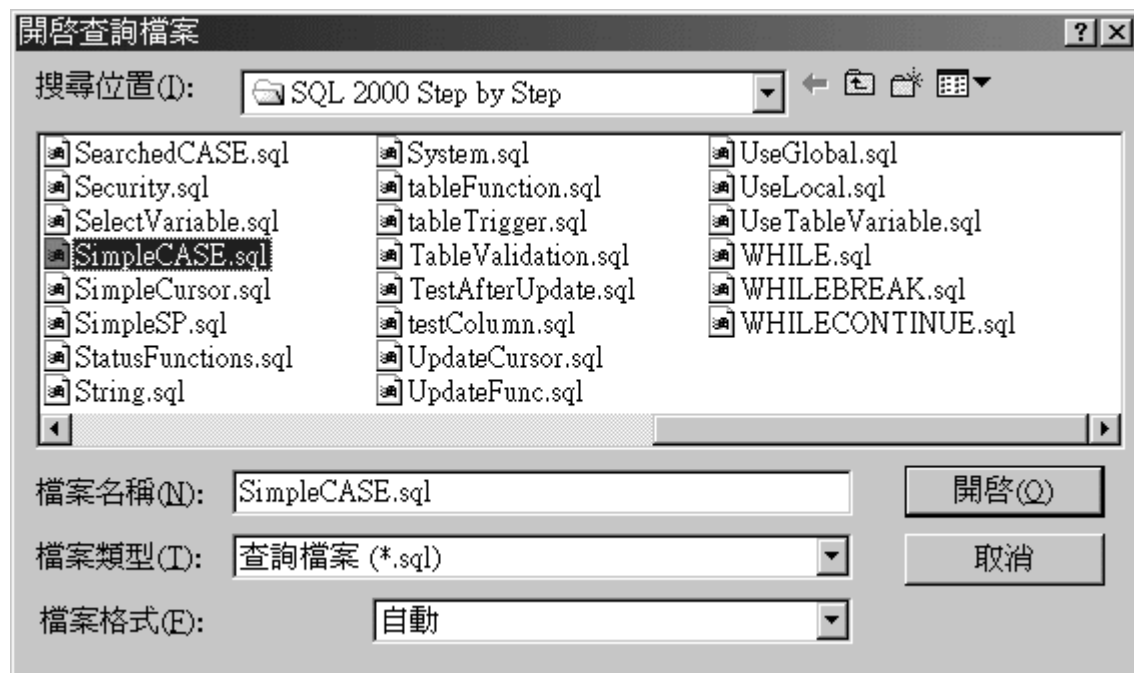
此时 Query Analyzer 会清除窗口内容。

2. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。
-



加载 SQL 指令码指令按钮

此 Query Analyzer 会显示 [开启查询档案](#) 的对话框。



3. 选取文件名称为 **SimpleCASE** 的指令码档案，然后按一下 **开启** 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



執行查詢按鈕

此 Query Analyzer 會顯示其執行的結果。

查詢 - KS_TR_MICKEY.Aromather...		
	OilName	PlantCategory
1	Bergamot	Fruit
2	Coriander	Fruit
3	Grapefruit	Fruit
4	Lemon	Fruit
5	Lime	Fruit
6	Nutmeg	Fruit
7	Orange	Fruit
8	Patchouli	Nuts and Berries
9	Peppermint	Nuts and Berries
10	Pettigrain	Nuts and Berries
11	Lemon Balm (Melissa)	Nuts and Berries
12	Juniper	Nuts and Berries
13	Geranium	Nuts and Berries
14	Cypress	Nuts and Berries
15	Eucalyptus	Nuts and Berries
16	Basil	Nuts and Berries
17	Black Pepper	Nuts and Berries
18	Rosemary	Nuts and Berries
19	Tea Tree	Nuts and Berries

編輯器 方格 訊息

Aromather: 0:00:00 Grid #1: 47 筆資料列 第 1 行，第 1 欄

使用搜寻式 CASE

1. 在 查詢 窗口内选取 編輯器 标签页，并且在 Query Analyzer 工具列上按一下 清除窗口 按钮。



清除窗口按钮

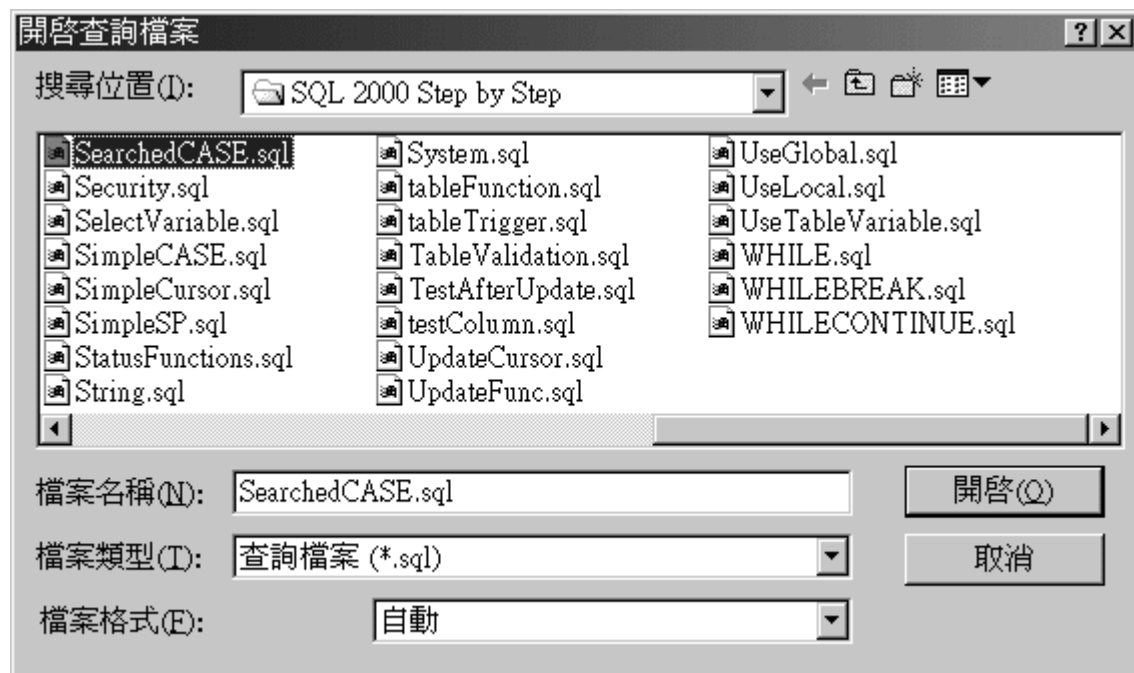
此时 Query Analyzer 会清除窗口内容。

2. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。



加载 SQL 指令码指令按钮

此 Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



3. 选取文件名称为 **SearchedCASE** 的指令码档案，然后按一下 **开启** 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



4. 在 Query Analyzer 的工具列上按一下 [执行查询](#) 按钮。



执行查询按钮

此 Query Analyzer 会显示其结果内容，请注意第三笔记录，其 OilName 字段值包含了 [Bergamot](#)，符合第一个表达式；第二个 WHEN 子句(LEFT(LatinName, 1)

= 'C') 所传回的值也是 True。所以 TestResults 值为 **Name B**，因为都是 True

时会传回 **第一个** 布尔表达式的值。



	OilName	LatinName	TestResults
1	Basil	Ocimum Basilicum	Name B
2	Benzoin	Styrax Benzoin	Name B
3	Bergamot	Citrus Bergamia	Name B
4	Black Pepper	Piper Nigrum	Name B
5	Cedarwood	Cedrus Atlantica, Juniperus Virginiana	LatinName C
6	Cinnamon	Cinnamomum Zeylanicum	LatinName C
7	Citronella	Cymbopogon Nardus	LatinName C
8	Clary Sage	Salvia Sclarea	Neither
9	Coriander	Coriandrum Sativum	LatinName C
10	Cypress	Cupressus Sempervirens	LatinName C

GOTO 命令

IF...ELSE 和 CASE 是依据布尔运算的结果以决定要执行的陈述式，而 GOTO 命令不需要使用

条件式，它会直接移转至其所标示的标签的后面继续执行命令。

在 SQL Server 中，标签是不需要执行的陈述式，其语法如下所示：

```
label_name:
```

该卷标名称必须要遵守识别项的规则。GOTO 命令本身的语法是非常简单的：

```
GOTO label_name
```

如同意大利面一样混乱的程序代码

在程序语言中，其实 GOTO 是一个非常不好的命令，这是有原因的：在早期的语言中仅提供非常少的流程控制机制，通常只提供 IF 和 GOTO 二个命令。而复杂又难读的程序代码就称为「Spaghetti Code」。

GOTO 有它存在的原因，尤其在错误掌控方面。假如您可以很小心地使用它，它也可以让程序代码更容易阅读。如果您不使用 GOTO 命令来执行其它的流程控制命令或其它的函数，这样会更好。

在非条件式流程控制中使用 GOTO 命令

1. 在 [查询](#) 窗口内选取 [编辑器](#) 标签页，并且在 Query Analyzer 工具列上按一下 [清除窗口](#) 按钮。



清除窗口按钮

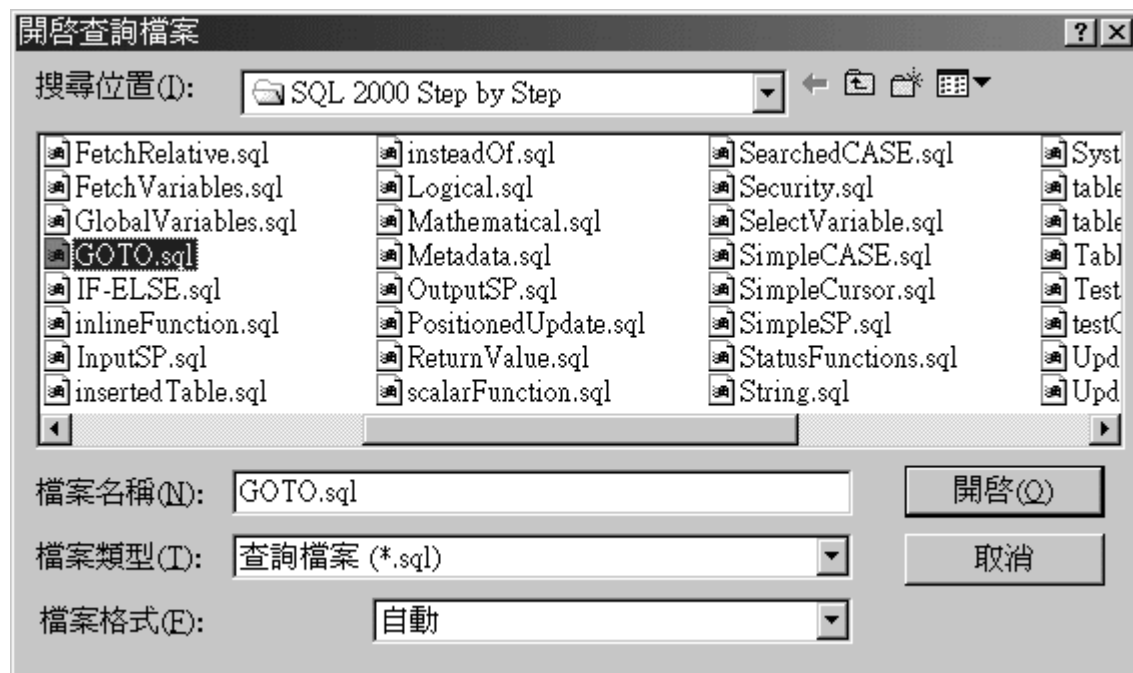
此时 Query Analyzer 会清除窗口内容。

2. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。



加载 SQL 指令码指令按钮

此 Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



3. 选取文件名称为 **GOTO** 的指令码档案，然后按一下 **开启** 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



執行查詢按鈕

此 Query Analyzer 會顯示其結果內容。



循环

最后一个流程控制命令可以让您指定执行陈述式或陈述式区块，直到符合某一个条件式为止。

简单式 **WHILE** 循环

简单式 **WHILE** 循环形式指定了一个布尔表达式以及一个陈述式或陈述式区块，此陈述式会重复执行直到布尔表达式的值为 **FALSE** 为止。当此 **WHILE** 陈述式是第一次执行并且布尔表达式又为 **FALSE** 时，则全部的陈述式或陈述式区块将不会执行。

使用简单式 **WHILE** 循环

1. 在 **查询** 窗口内选取 **编辑器** 标签页，并且在 **Query Analyzer** 工具列上按一下 **清除窗口** 按钮。



清除窗口按钮

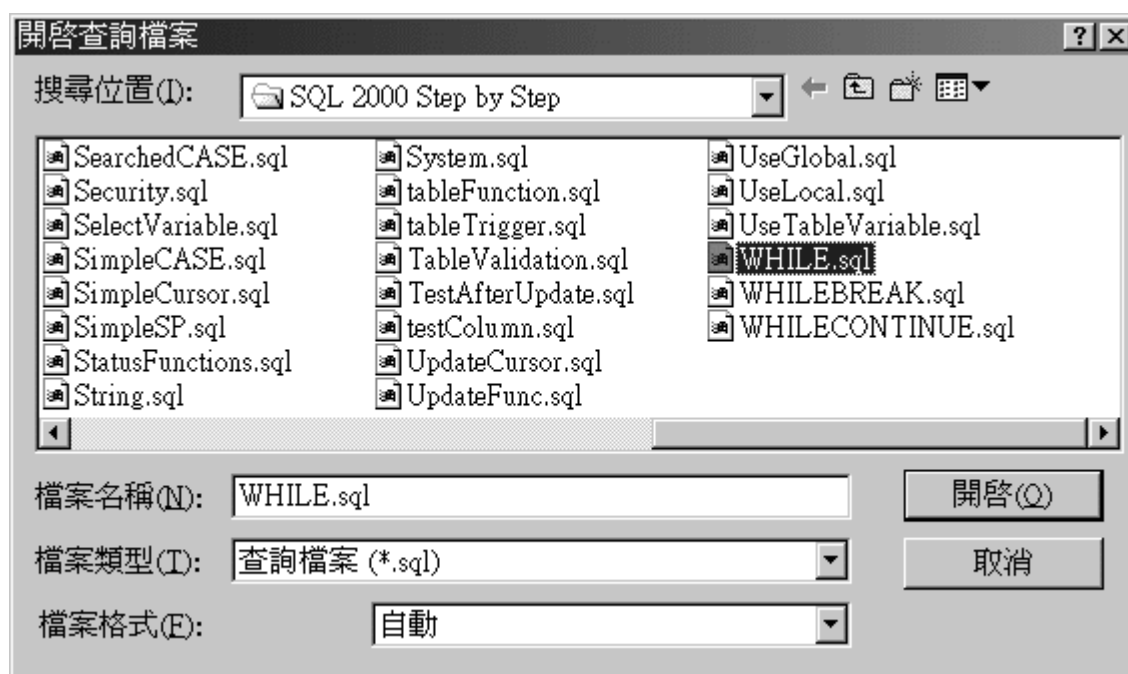
此时 **Query Analyzer** 会清除窗口内容。

2. 在 **Query Analyzer** 工具列上按一下 **加载 SQL 指令码** 按钮。



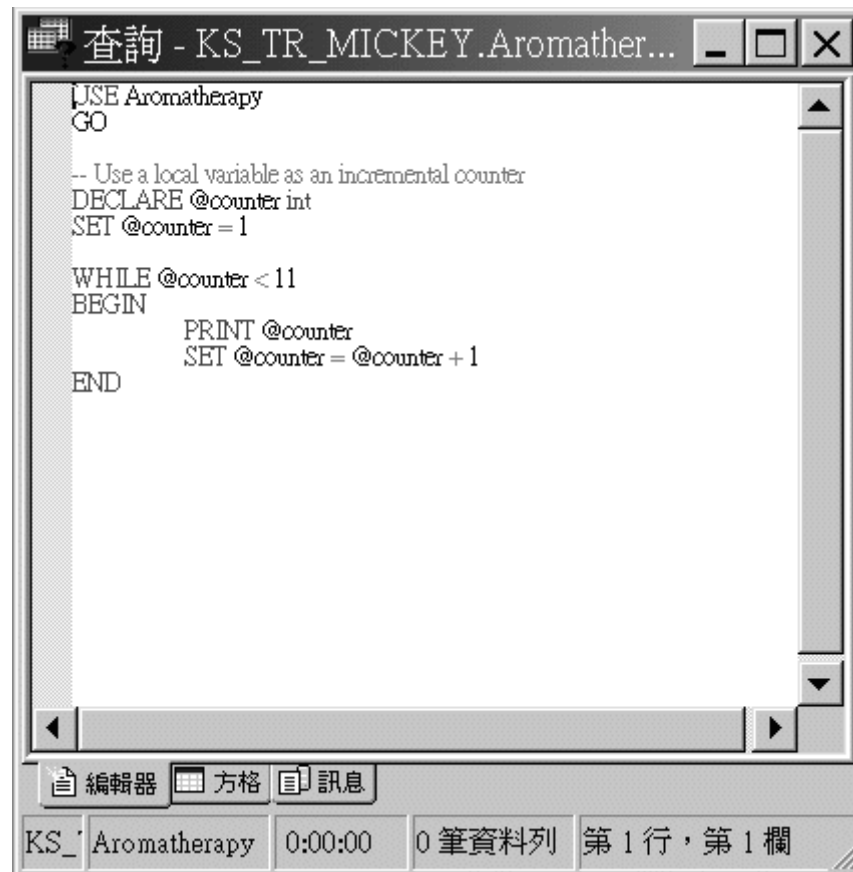
加载 SQL 指令码指令按钮

此 Query Analyzer 会显示 [开启查询档案](#) 的对话框。



3. 选取文件名称为 **WHILE** 的指令码档案，然后按一下 [开启](#) 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



執行查詢按鈕

此 Query Analyzer 会显示其结果内容。



复合式 WHILE 循环

WHILE 陈述式语法也提供了比前一个范例更复杂的处理方式。**BREAK** 子句是用来离开循环的，并直接到 WHILE 陈述式区块的 **END** 子句后面的陈述式继续执行。**CONTINUE** 子句会回到此循环的第一行开始执行，并且会导致此陈述式区块中 **CONTINUE** 后面的陈述式被忽略而不执行。

BREAK 和 **CONTINUE** 二者都是基于 IF 陈述式的条件执行。

您可以在相同的 **WHILE** 陈述式中使用 **BREAK** 和 **CONTINUE**。虽然只有单一的情况将会被执行，您也可以在此陈述式区块内执行每一个子句好几次。

使用 **WHILE...BREAK**

1. 在 **查询** 窗口内选取 **编辑器** 标签页，并且在 Query Analyzer 工具列上按一下 **清除窗口** 按钮。



清除窗口按钮

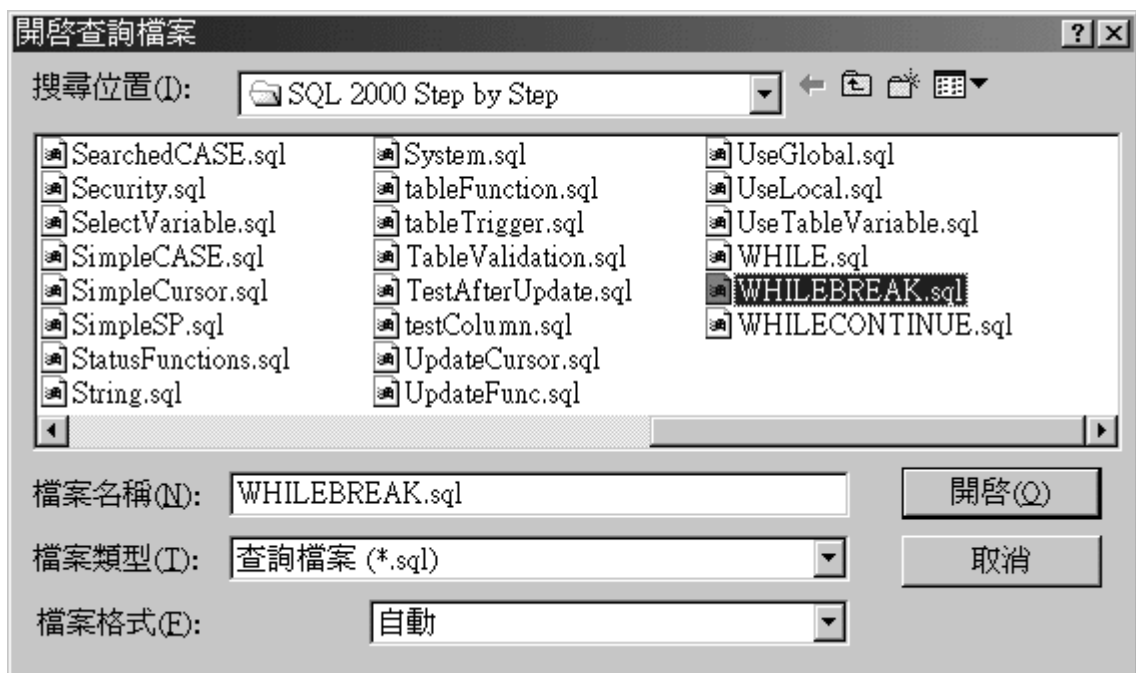
此时 Query Analyzer 会清除窗口内容。

2. 在 Query Analyzer 工具列上按一下 **加载 SQL 指令码** 按钮。



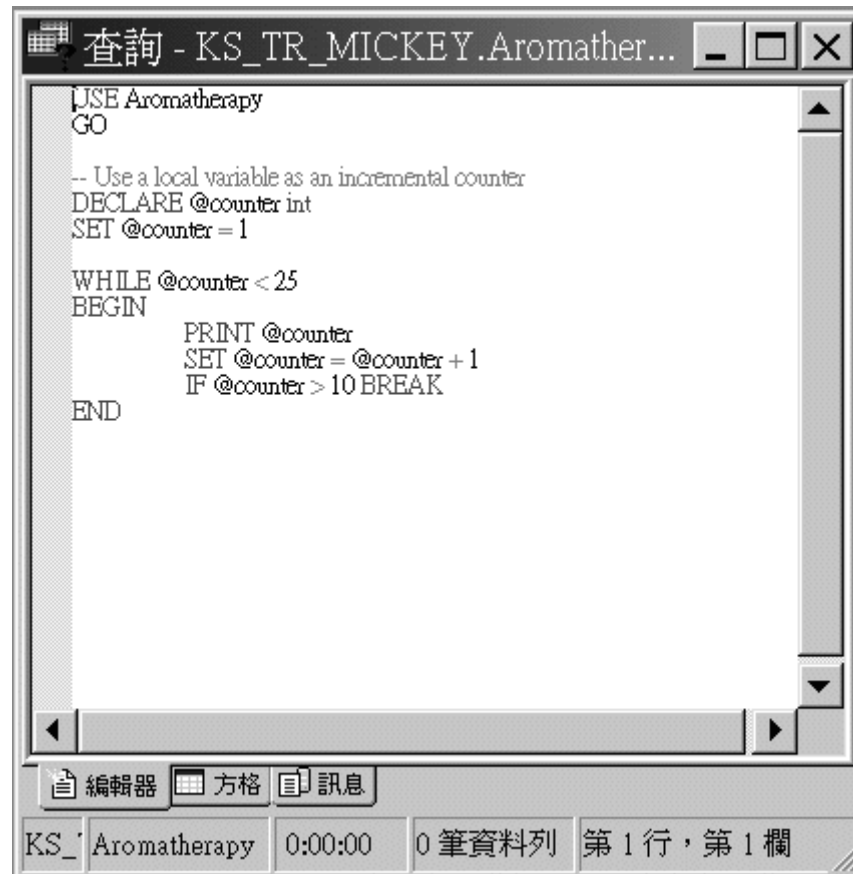
加载 SQL 指令码指令按钮

此 Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



3. 选取文件名称为 **WHILEBREAK** 的指令码档案，然后按一下 [开启](#) 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



執行查詢按鈕

此 Query Analyzer 会显示其结果内容。



使用 WHILE...CONTINUE

1. 在 **查詢** 窗口內選取 **編輯器** 標籤頁，並且在 Query Analyzer 工具列上按一下 **清除窗口** 按鈕。



清除窗口按钮

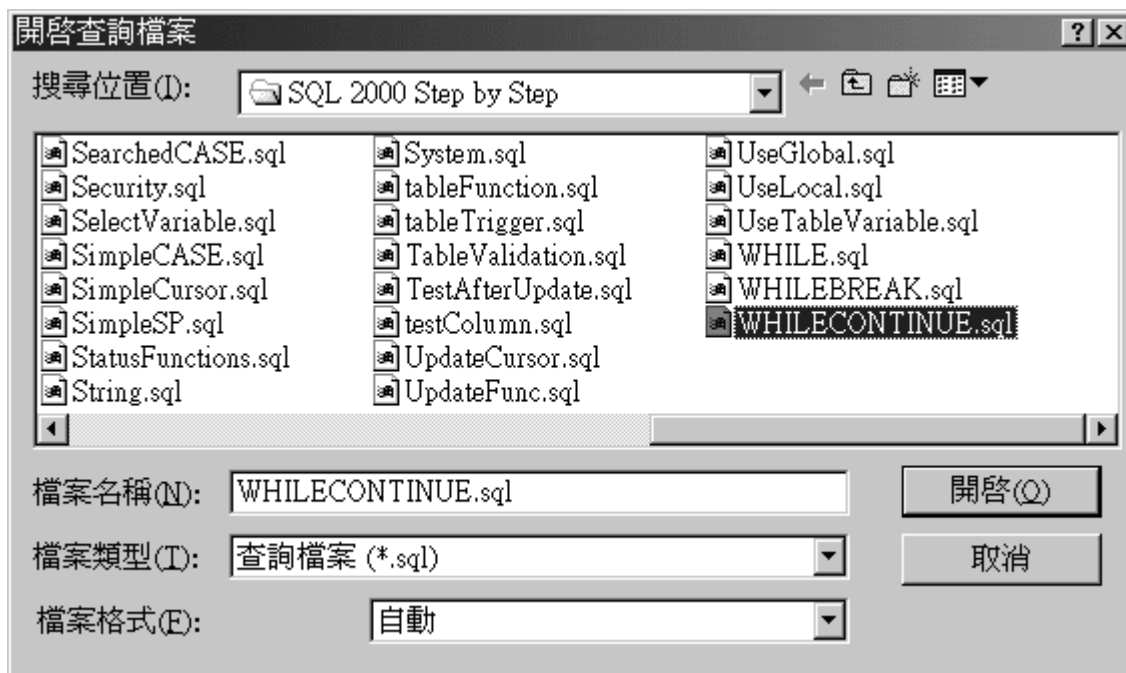
此时 Query Analyzer 会清除窗口内容。

2. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。



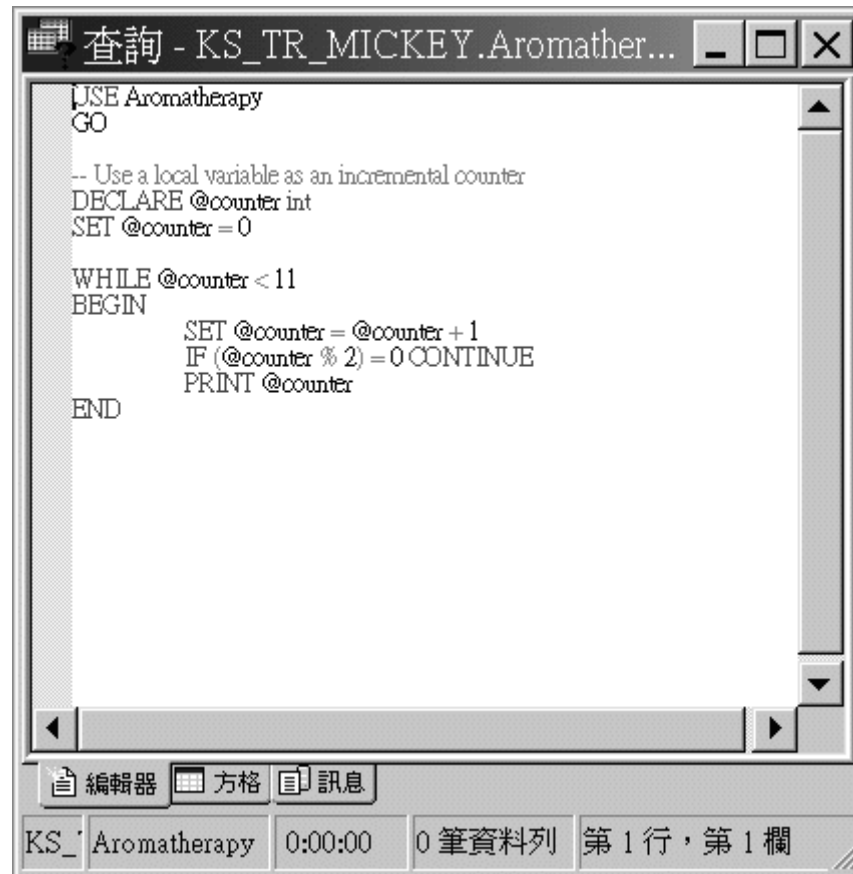
加载 SQL 指令码指令按钮

此 Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



3. 选取文件名称为 **WHILECONTINUE** 的指令码档案，然后按一下 **开启** 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



執行查詢按鈕

此 Query Analyzer 会显示其结果内容。



本章总结

要执行的工作	SQL 语法
执行条件陈述式	<div>IF Boolean_expr</div> <div>statement_or_block</div> <div>[ELSE statement_or_block]</div>
基于某值相等以在 SELECT 或 UPDATE 陈述式中传回	<div>CASE value</div>

结果	<pre>WHEN expr_one THEN result_expr_one WHEN expr_two THEN result_expr_two : WHEN expr_n THEN result_expr_n [ELSE else_result_expr] END</pre>
基于布尔表达式判断以在 SELECT 或 UPDATE 陈述式 中传回结果	<pre>CASE WHEN expr_one THEN result_expr_one WHEN expr_two THEN result_expr_two : WHEN expr_n THEN result_expr_n [ELSE else_result_expr] END</pre>
非条件式的移转执行	<pre>GOTO label_name label_name:</pre>

当条件式为 TRUE 时，则重复执行某一陈述式或陈述式区块的内容	<pre>WHILE Boolean_expr statement_or_block</pre>
结束 WHILE 循环的执行	<pre>WHILE Boolean_expr BEGIN : BREAK : END</pre>
回到 WHILE 循环起始位置执行	<pre>WHILE Boolean_expr BEGIN : CONTINUE : END</pre>

27. Transact-SQL 数据指针

在本章中，您将学习到：

- 宣告数据指针。
- 开启数据指针。
- 关闭数据指针。
- 解除配置数据指针。
- 使用简单的 **FETCH** 命令。
- 撷取数据列至变量中。
- 藉由数据列的绝对位置撷取数据列。
- 藉由数据列的相对位置撷取数据列。

- 执行定位更新。
- 执行定位删除。
- 使用 @@CURSOR_ROWS 以判定数据指针集合中的数据列数目。
- 使用 @@FETCH_STATUS 以判定 FETCH 陈述式的状态。
- 使用 CURSOR_STATUS 以查询数据指针的状态。

在关系型数据库中所定义的特性之一就是一操作都是基于数据列集为单位执行，数据列集也许是空的，或者是该集合只包含一个数据列，而就算只有一个数据列，该数据列仍然会被视为一个集合。对于关系型的操作来说，这样是必要的，但是这样对于应用程序来说却造成麻烦。

举例来说，因为没有办法简单的直接指向一个集合中的特定数据列，所以一次给使用者一个数据列就变得十分困难。而且纵使藉由 Transact-SQL 提供更多的程序化功能，仍会需要许多繁杂、困难、昂贵的操作。

为了要控制这些状态，SQL Server 提供了 **数据指针**（Cursor）的功能。一个数据指针是一个对象，它可以指向一个集合中的某特定数据列。取决于您所建立的数据指针而定，您可以在集合中移动数据指针，并且更新或删除数据。

认识数据指针

Microsoft SQL Server 实际上支持二种不同的数据指针型态：**Transact-SQL** 数据指针，以及应用程序开发接口（Application Programming Interface，API）数据指针。API 数据指针是使用 Microsoft ActiveX Data Object（ADO）、OLE DB、Open Database Connectivity（ODBC）或者是 DB-Library 在应用程序中所建立。在这些 API 中的每一个数据指针都支持不同的功能以及使用的不同的语法。我们并不会在本章中讨论 API 数据指针更详细的内容，假如您要使用它们时，您可以和其它比较厉害的程序设计师进行更详细的讨论。

您可以使用 DECLARE CURSOR 命令来建立 Transact-SQL 数据指针。当数据指针对象和该数据指针指向的数据列集位于服务器时，这个就称为 **伺服器端数据指针**（server-side cursor）。当您从应用程序中透过网络使用伺服器端数据指针来与 SQL Server 进行连接时，在该数据指针中的每一个操作都会在网络上来回游走。API 数据指针链接库同时支持伺服器端数据指针和 **客户端数据指针**（client-side cursor），客户端数据指针储存在客户端系统，并且会将数据列以快取方式储存在客户端系统以供数据的操作。

数据指针所指向的数据列集由 **SELECT** 命令加以定义，当建立一个 Transact-SQL 数据指针时，

在 **SELECT** 陈述式的使用上会有一些限制：

- 无法传回多重结果集。
- 无法包含 **INTO** 子句以建立新的数据表。
- 无法在包含 **COMPUTE** 或 **COMPUTE BY** 子句以汇总结果（但是可以包含汇总函数，例如 **AVG** 函数）。

数据指针的特性

Transact-SQL 支持数种不同型态的数据指针。在每一个型态中包含的形形色色的特性可能会让

您有所混淆，但如果您将这些特性分为 3 大类，那就很容易了解了：

1. 可修改底层的基本数据。
2. 可滚动资料列集
3. 可更新数据列集

反应变更

当数据指针可反映变更至实际的底层数据时，这就是所谓的 **敏感性数据指针**（sensitive cursor）。

举例来说，当您为如下所示的陈述式建立数据指针：

```
SELECT * FROM Oils
WHERE Left (OilName, 1)=' B '
```

当您执行时，Aromatherapy 数据库会传回如图 27-1 所示的四个数据列，假如当您使用该数据指针期间，某个人加入了 **Bayberry** 数据列，您的数据指针所指向的数据列集会发生什么事呢？



	OilID	OilName	LatinName	PlantTypeID	PlantPartID
1	1	Basil	Ocimum Basilicum	1	1
2	2	Bergamot	Citrus Bergamia	2	2
3	3	Black Pepper	Piper Nigrum	3	3
4	46	Benzoin	Styrax Benzoin	2	9

图 27-1 Aromatherapy 范例数据库内包含有以字符 B 为开头字符的四笔数据列。

当您建立数据指针时，有二种敏感数据指针可以使用：变更数据列集中的数据列，以及变更底层数据列的值。

卷动

第二个资料指针的特性就是您可以使用向前卷动、向后卷动，或者是只有向前卷动的数据指针。

您必须在这些卷动的数据指针当中作最佳的取舍，顺向（forwardonly）数据指针会有明显的效能表现，但是却缺乏弹性。

更新

最后一个数据指针的特性就是将资料指针分类，分成是否可以透过数据指针更新数据这两类。再次强调，只读数据指针会有明显的效能表现，但是却缺乏弹性。

数据指针的种类

Transact-SQL 支持四种不同型态的数据指针：**静态**（static）、**索引键集**（keyset）、**动态**（dynamic）以及**消防管**（firehose）。每一种数据指针型态都会指到数据列内不同的数据。

每一种数据指针型态会将它所指向的数据列以不同的方式储存，并且每一种数据指针型态都支持不同的特性。

静态数据指针

静态数据指针会藉由 **SELECT** 陈述式来撷取特定数据的快照（**snapshot**），并且将这些数据储存在 **tempdb** 数据库中。它并不会感知底层数据值的变更，因为它只会反映出其所拷贝副本的值，因此它也是只读的。静态数据指针可以宣告为只读或可卷动。

索引键集数据指针

所谓的索引键集数据指针是，只将唯一识别每一数据列的数据行复制到 **tempdb** 数据库中。为了要宣告索引键集数据指针，在 **SELECT** 陈述式内定义的每一个数据表都必须有唯一索引键，用来定义被复制的索引键集。

索引键集数据指针可以设定为可更新或只读，也可以设定为可卷动或顺向。

当您宣告此型态的数据指针时，在索引键集内的成员是固定的。当数据指针被开启后，假如有满足选择标准的数据列被加入，则该数据列不会被加入到此数据列集中。如我们之前的范例一般，有一个选择标准是 **LEFT(OilName,1)='B'**，当一个符合此选择标准的 **Bayberry** 数据列被加入时，它将被加入到数据指针中。

相同的，当有一个索引键集内的成员数据列被改变时，例如若您将 **Basil** 更改为 **Kumquat**，而该数据列仍然是索引键集内的成员，所以会被反映。甚至于有一个数据列被删除时，它也算是索引键集内的成员，但是 **SQL Server** 会为此数据行的值传回一个 **NULL**。

纵使您开启一个数据指针后在此数据指针集内的成员是固定的，但是您在一些底层数据表中更改数据值时是会反映出来的。举例来说，当数据指针将 **Bergamot** 数据列的 **Description** 数据行的值更改时，该值将会被传回。然而，您只有透过索引键值数据指针修改数据时才会有修改的数据值传回。我们以上一个范例来说明，假如 **OilName** 的值藉由索引键值数据指针从 **Basil** 更改为 **Kumquat** 时，此数据指针将传回 **Kumquat**。假如这种改变是由其它的使用者所造成时，此数据指针将会传回'Basil'。

提示

如同我们将在下一个节讨论的，建立一个数据指针并且将它开启有不同的操作方式。当您将它关闭并且再重新开启它时，就会重新显示索引键集数据指针的内容。

动态数据指针

在概念上，一个动态的数据指针的行为是依据每次一有数据列被引用时，就重新执行 **SELECT** 陈述式（实际上的情形是不同的，但这是您了解动态数据指针最好的方式）。动态数据指针会同时反应成员关系和与底层数据值的变更，不论这些改变是由数据指针或者是其它的使用者都一样。

使用动态数据指针有一个限制：如果要在定义动态数据指针中的 **SELECT** 陈述式包含 **ORDER BY** 子句，必须在 **ORDER BY** 子句所包含的数据行上有建立索引。

消防管数据指针

SQL Server 支持一种只能只读、无法卷动的特殊的数据指针，这一类的资料指针您可以使用 **FAST_FORWARD** 来进行宣告，那就是一般我们所称的 **消防管** 数据指针。

消防管数据指针是非常有效率的，但是在使用它时会有二个重要的限制。第一是

假如您用来定义此数据指针的 **SELECT** 陈述式中包含有 **text**、**ntext** 或 **image** 数据型别的数据行，以及包含 **TOP** 子句时，**SQL Server** 会将消防管数据指针转换为索引键集数据指针。

第二个限制是，假如您用来定义此数据指针的 **SELECT** 陈述式中联合使用了拥有触发程序的数据表和没有触发程序的数据表时，此数据指针将会转换为一个静态数据指针。触发程序是

Transact-SQL 指令码，当数据操作语言（**Data Manipulation Language**，**DML**）在数据表上执行时，触发程序将会由服务器自动执行。我们将在第 29 章中详细说明关于触发程序的一些特性，

但是在此有一个非常重要的观念：假如有人在您的数据指针中的数据表中加入一个触发程序时，您的应用程序会意外地强迫停止，这是因为 **SQL Server** 正在将快速的数据指针转换成慢速的数据指针所致。

使用数据指针

使用数据指针的方式和使用一个区域变量的方式是相似的一宣告它、设定值，然后再使用。但是与区域变量不相同的是，区域变量在超过有效范围时会自动被移除；而您必须明确的释放数据指针所使用的数据列，然后再移除数据指针。

建立数据指针

要使用数据指针的第一个步骤便是建立它，您可以使用 **DECLARE CURSOR** 陈述式来建立 Transact-SQL 数据指针。

重要

SQL Server 提供二种不同的方式来建立数据指针：**SQL-92** 语法和 **Transact-SQL** 语法。**SQL-92** 语法是一种标准的 **ANSI** 语法，但是其功能并没有 **Transact-SQL** 语法来得强。

DECLARE CURSOR 陈述式的语法如下所示：

```
DECLARE cursor_name CURSOR

[visibility ]

[scroll ]

[type ]

[lock ]

[TYPE_WARNING ]

FOR select_statement

[FOR UPDATE [OF column_names ]]
```

请注意所有的参数都是用来定义数据指针的特性，其中 **visibility**、**type** 等等都是选择性的。无论您有或没有指定这些参数值，其默认值都会和一些底层记录或检视表以及数据库的选项有相互的影响。由于这些原因，在您建立数据指针时必须将这些参数指定好，那么您所建立的数据指针将会符合您的需求。

数据指针内的 **visibility** 可以和关键词 **LOCAL** 和 **GLOBAL** 配合使用，就如同使用 **@local_table** 或 **@@global_table** 来宣告暂存数据表是一样的意思。

提示

当区域数据指针超过有效范围后，SQL Server 会自动将它关闭并且解除配置该数据指针，我们将在稍后说明。

scroll 参数会接受 **FORWARD_ONLY** 和 **SCROLL** 关键词，该数据指针会以您想要的方式工作。

type 参数是用来指定您所建立的资料指针的类型，您可以将数据指针设定为 **STATIC**、**KEYSET**、**DYNAMIC** 以及 **FAST_FORWARD** 等关键词。**FAST_FORWARD** 型态参数和 **FORWARD_ONLY** 卷动性参数是彼此独立的。

lock 参数是用来决定资料列是否可以藉由数据指针来进行更新，或者是否可以让其它的使用者可以将它们进行更新。假如您使用 **READ_ONLY** 关键词时，透过数据指针无法对底层数据进行更新；只能透过标准的 **UPDATE** 陈述式或其它使用者进行数据更新。假如您将 **lock** 参数指定为 **SCROLL_LOCKS** 时，那就表示只有藉由数据指针才可以将数据进行更新。而其它的 **UPDATE** 陈述式或其它的使用者进行数据更新的动作将会失败。

最后的 **lock** 选项 **OPTIMISTIC**，是指您可以透过数据指针或在数据指针以外更新数据列。这是最具备弹性的选项，但是有时候透过数据指针更新数据列可能会失败（因为在数据指针读取某数据列后，该数据列又被更新了，发生了数据冲突）。

假如一个数据指针已经目前的型态转换为其它的型态时，**TYPE_WARNING** 参数可以用来告知 **SQL Server** 传送警告讯息给客户端，这个讯息甚至在您所宣告的数据指针没有指定 **SELECT** 陈述式也会发生。

select_statement 参数是 **FOR** 子句的必要参数。它是用来指定在数据指针内所包含的数据列。

FOR UPDATE 子句是选择性的，根据默认值其数据指针是可以更新的，除非您在 **lock** 参数指定为 **READ_ONLY**。但话又说回来，当您宣告数据指针时有使用此参数时是比较安全的方式，可以确保您取得您想要的结果。您可以使用 **OF column_names** 以便指定您允许哪个特定的数据列可以更新。假如您忽略使用 **OF column_names** 时，在 **SELECT** 陈述式内的所有数据行都是可更新的。

数据指针变量

Transact-SQL 允许您宣告 **CURSOR** 型态的变量。您必须使用 **SET** 来设定数据指针变量的值：

```
DECLARE myCursor CURSOR

    LOCAL

    FAST_FORWARD

    FOR SELECT OilName FROM Oils

DECLARE @myCursorVariable CURSOR
```

```
SET @myCursorVariable = myCursor
```

当您想要建立分配不同数据指针的变量时，这种语法是非常好用的。假如您想要建立可以用来操作多重结果集的程序时，您就可以这样做。

您可以宣告一个数据指针变量，然后使用它直接建立一个数据指针：

```
DECLARE @myCursorVariable CURSOR  
  
SET @myCursorVariable CURSOR  
  
    LOCAL FAST_FORWARD FOR SELECT OilName FROM Oils
```

使用这种语法，该数据指针没有识别项，并且只能藉由此变量来引用。这种语法在预存程序中是非常好用的，我们将在第 28 章中讨论。

开启一数据指针

宣告数据指针会建立一个数据指针对象，但是无法建立此数据指针可以操作的纪录集。因此您必须开启数据指针以建立纪录集。在使用复杂的 **DECLARE CURSOR** 陈述式建立数据指针之后，就要使用 **OPEN** 陈述式开启数据指针：

```
OPEN [GLOBAL] cursor_or_variable
```

其中 **GLOBAL** 关键词是用来避免其名称冲突：假如一个数据指针以 **LOCAL** 关键词来宣告，另一个以 **GLOBAL** 关键词来宣告，但是这两个数据指针却拥有相同的识别项时，除非您是使用 **GLOBAL** 关键词来进行宣告，否则将会引用定义为区域性的数据指针。一般而言，如果您开启一个全域数据指针，明确的宣告是比较保险的。

关闭数据指针

一旦您已经将开启的数据指针使用完毕之后，接下来的工作当然是将它关闭。您可以使用 **CLOSE** 陈述式将数据指针关闭，并且将数据指针所使用的资源也释放掉。假如您在使用 **DECLARE** 陈述式时也使用了 **SCROLLLOCKS** 参数，会同时释放任何锁定的数据列位置。**CLOSE** 命令的语法其实和 **OPEN** 命令是差不多的，仅有一个命令关键词要更改：

```
CLOSE [GLOBAL] cursor_or_variable
```

解除配置数据指针

在建立数据指针之后的最后一件事就是将它解除配置，您可以使用 **DEALLOCATE** 命令将数据指针解除配置。其语法的表示方式如下：

```
DEALLOCATE [GLOBAL] cursor_or_variable
```

DEALLOCATE 陈述式会移除数据指针识别项或数据指针变量，但是它不需要移除数据指针。数

据指针本身不需要被移除，除非所有引用到该数据指针的识别项被解除配置或超出有效范围。请

参考下列范例：

--数据指针

```
DECLARE myCursor CURSOR
```

```
    KEYSET
```

```
    READ_ONLY
```

```
    FOR SELECT *FROM Oils
```

--数据指针变量

```
DECLARE @cursorVariable CURSOR
```

--建立数据指针集

```
OPEN myCursor
```

--分配变量值为刚刚建立的数据指针

```
SET @cursorVariable =myCursor
```

--解除配置数据指针


```
DEALLOCATE myCursor
```

在您解除配置数据指针之后，使得识别项 **myCursor** 不再和数据指针集有任何相关性，但是数据指针集仍然由 **@cursorVariable** 变量所引用，此数据指针和数据指针集仍然还没有被释放。除非您也明确的解除配置数据指针变量，因此目前该数据指针和数据指针集仍然会一直存在，直到变量超过有效范围为止。

使用数据指针来操作数据列

假如您并不是做每一件事都使用数据指针时，那么使用数据指针就不是那样的有趣了。

Transact-SQL 提供 3 种不同的数据指针操作命令：**FETCH**、**UPDATE** 以及 **DELETE**。

FETCH 命令是用来从数据指针集中撷取指定的数据列。**FETCH** 命令的语法如下所示：

```
FETCH cursor_or_variable
```

这种方式会传回数据指针中的定位数据列（目前的数据列）。

使用简单的 **FETCH** 命令

1. 在 Query Analyzer 工具列上按一下 **新增查询** 按钮，以便开启一个新的查询窗口。
-



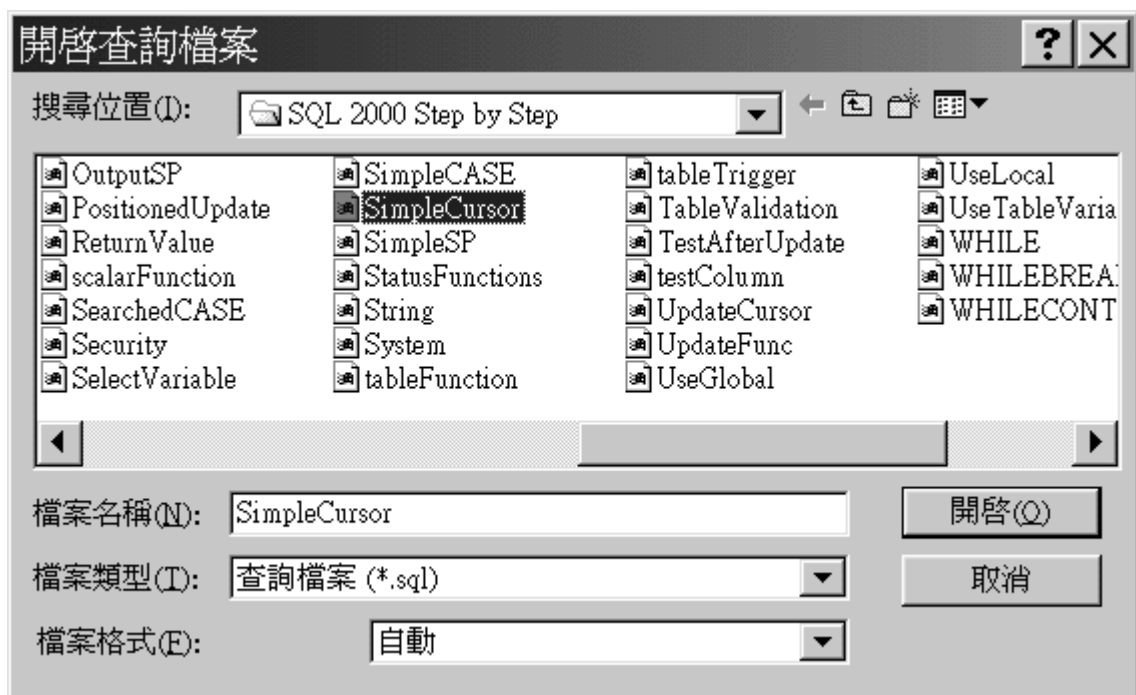
新增查询按钮

2. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。
-

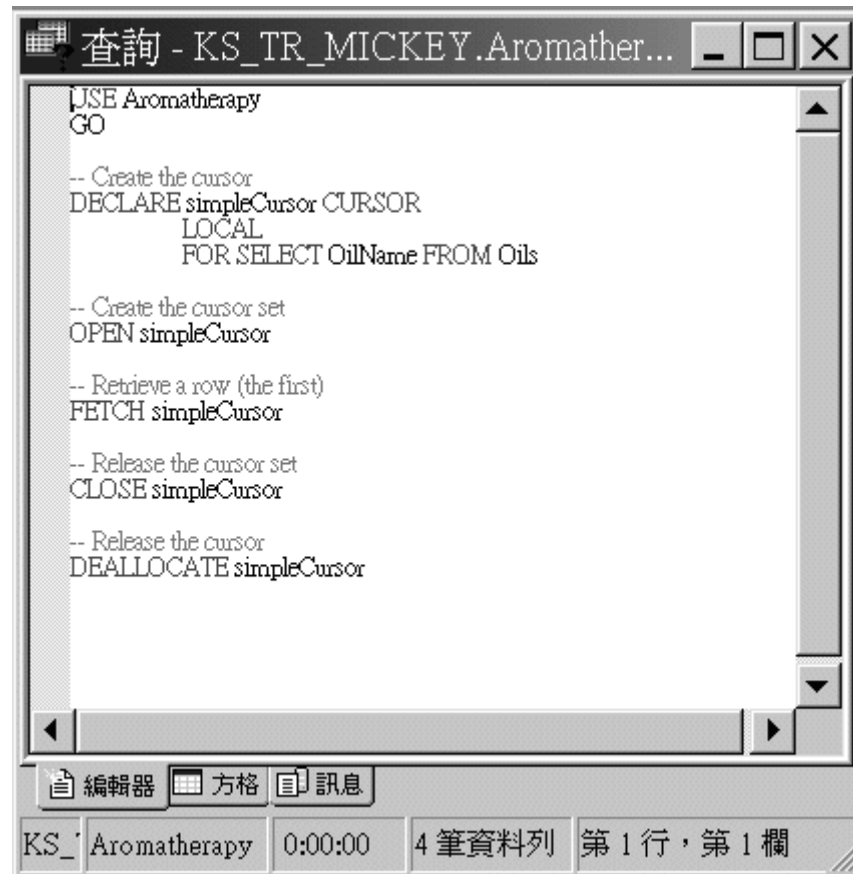


加载 SQL 指令码按钮

此 Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



3. 寻找档案数据夹 **SQL 2000 Step by Step**，并选取文件名称为 **SimpleCursor** 的指令码档案，然后按一下 **开启** 按钮。
4. 此 Query Analyzer 会将指令码档案加载到查询窗口中。



5. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



執行查詢按鈕

此 Query Analyzer 會執行此查詢。



提示

您可能会注意到这个冗长的指令码的执行时间比 **SELECT** 陈述式执行的时间还要长，这是因为在建立和开启数据指针的时间花费比较长时间的缘故。请您注意：假如您可以使用 **SELECT** 陈述式执行时，就不要使用数据指针。

FETCH 命令允许您将所传回的数据行值直接储存在变量中。为要将执行 **FETCH** 的结果储存在一个变量中，您可以使用如下所示的语法：

```
FETCH cursor_or_variable  
INTO variable_list
```

在 **variable_list** 中，您可以使用逗号「，」将每一个变量隔开。在执行 **FETCH** 命令之前，您必须宣告这些变量。**variable_list** 必须在 **SELECT** 陈述式中的每一个数据行包含一个变量，并且这些变量的数据类型必须兼容或相等于数据行的数据类型。

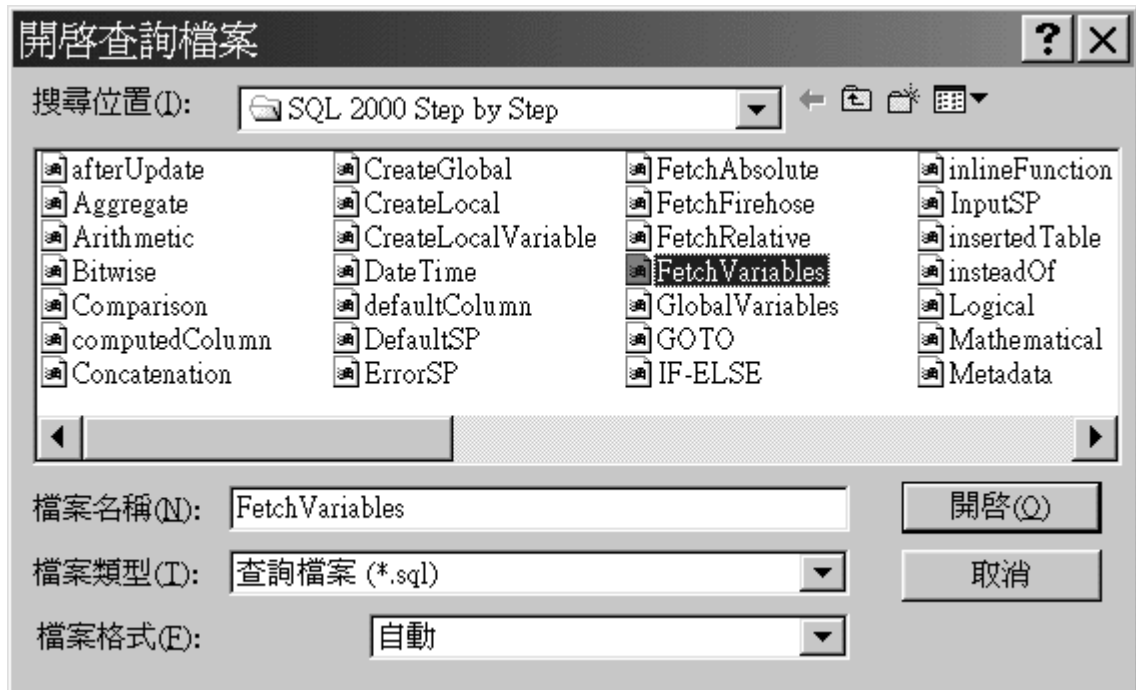
撷取一数据列至变量中

1. 在 Query Analyzer 工具列上按一下 **加载 SQL 指令码** 按钮。



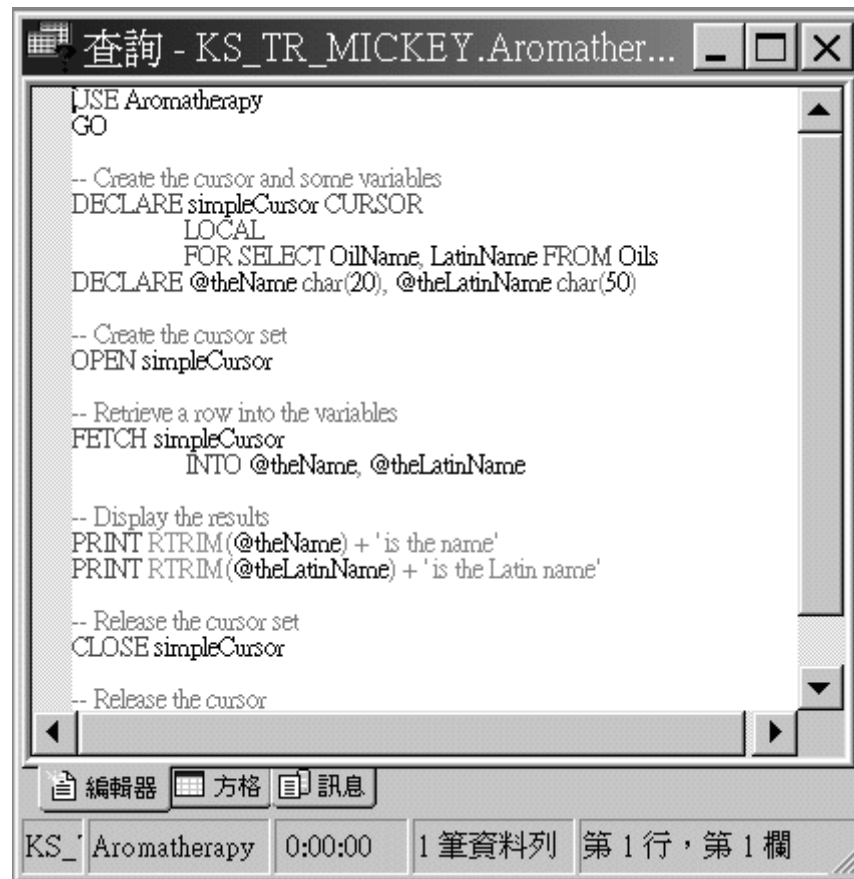
加载 SQL 指令码按钮

此 Query Analyzer 会显示 **开启查询档案** 的对话框。



2. 选取文件名称为 **FetchVariables** 的指令码档案，然后按一下 **开啓** 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



3. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



執行查詢按鈕

此 Query Analyzer 會執行此查詢。



在前一个范例中已经使用 **FETCH** 陈述式将目前的数据列传回。**FETCH** 陈述式语法也提供一些关键词来指定不同的数据列。当您使用其中之一关键词时，**FETCH** 陈述式将会传回您所指定的数据列，并且确保所传回的数据列是目前的数据列。

有 3 个关键词可以让您在数据指针集中指定绝对位置，**FIRST** 和 **LAST** 关键词会传回第一个和最后一个数据列，如果使用 **ABSOLUTE n** 时，如果 **n** 为正数，会从数据指针的前面算起传回第 **n** 列的数据列；假如 **n** 为负数，会从数据指针的后面算起传回第 **n** 列的数据列。您可以将 **n** 的值指定为一个常数（例如为 3）或一个变数（例如@theRow）。

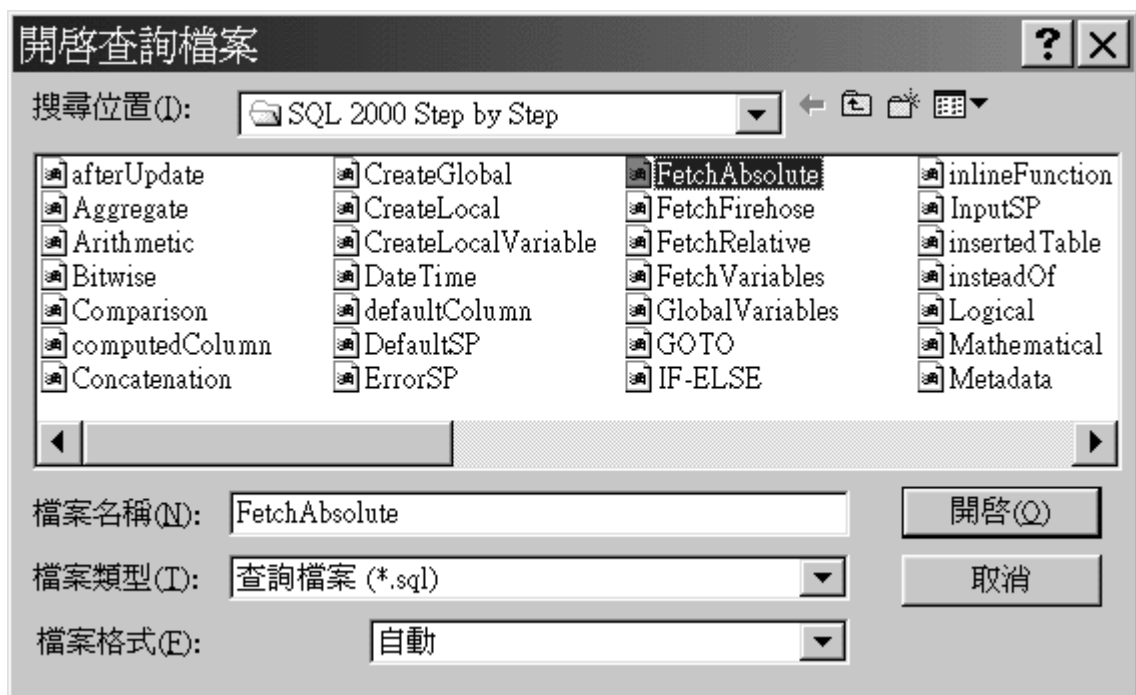
藉由绝对位置撷取数据列

1. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。



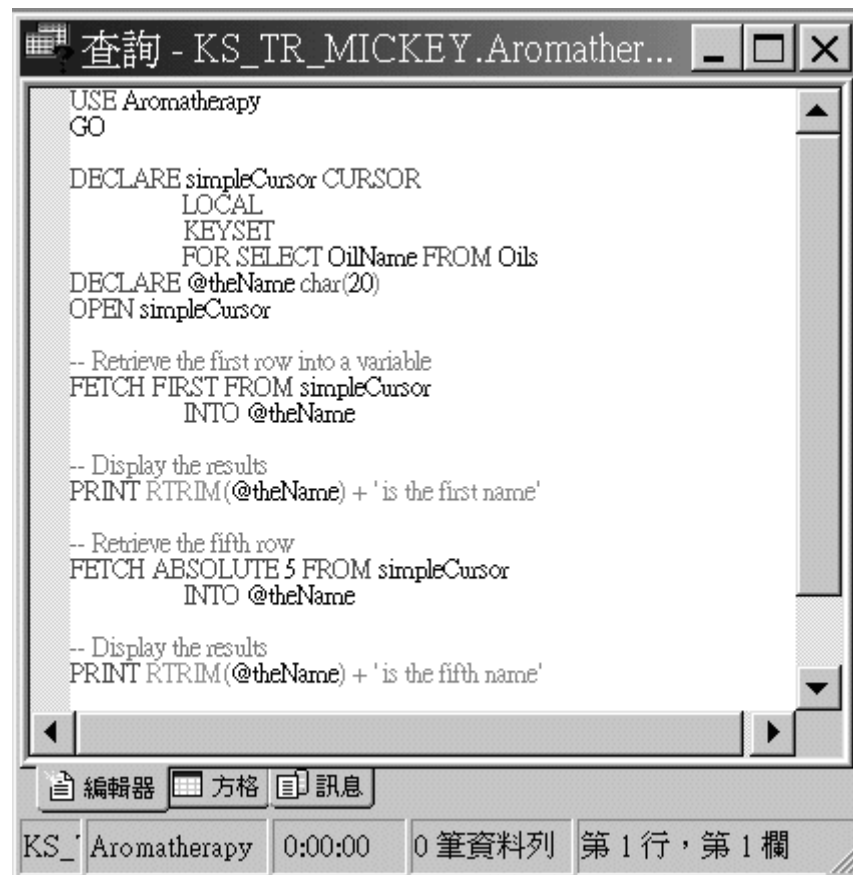
加载 SQL 指令码按钮

此 Query Analyzer 会显示 [开启查询档案](#) 的对话框。



2. 选取文件名称为 **FetchAbsolute** 的指令码档案，然后按一下 **开启** 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



3. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



執行查詢按鈕

此 Query Analyzer 會執行此查詢。



除了可以让您在绝对位置上取回数据列的关键词以外，**FETCH** 陈述式提供 3 个关键词让您基于目前的数据列的相对位置取回数据列。**FETCH NEXT** 会传回下一个资料列，而 **FETCH PRIOR** 会传回前一笔资料列，而 **FETCH RELATIVE n** 会传回从目前资料列算起 **n** 列的资料列，如同 **FETCH ABSOLUTE n** 一样，**FETCH RELATIVE n** 当 **n** 为负数时，可以传回目前数据列的前面 **n** 列的资料列；假如 **n** 为正数时，可以传回目前数据列后面 **n** 列的数据列。

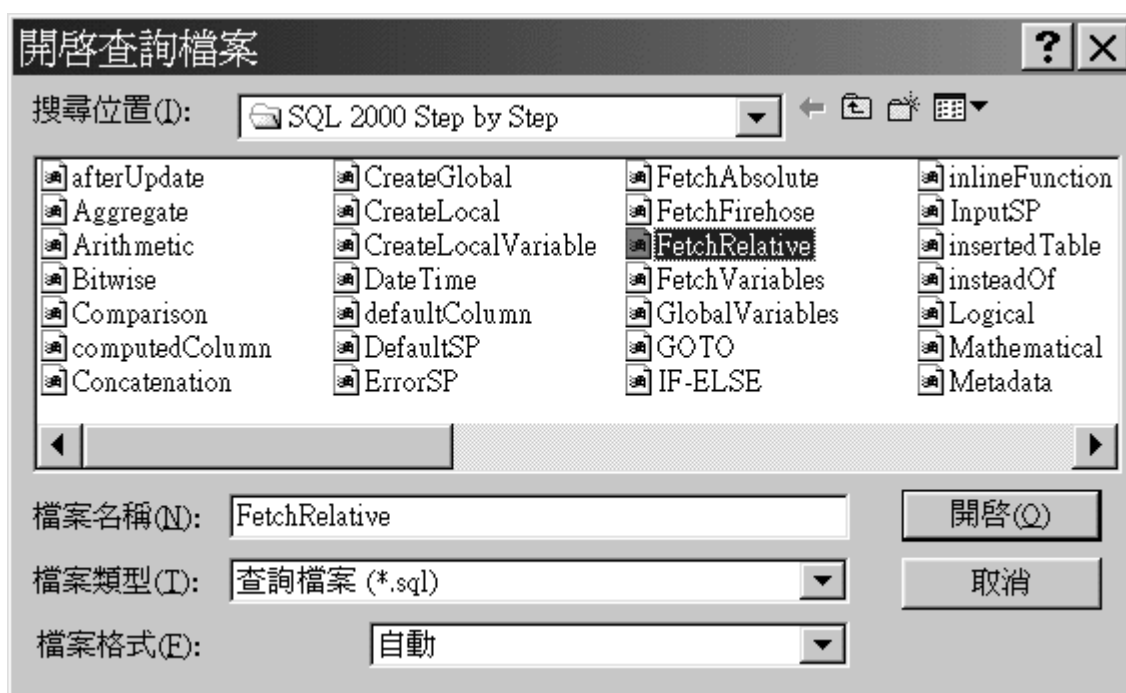
藉由相对位置撷取数据列

1. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。



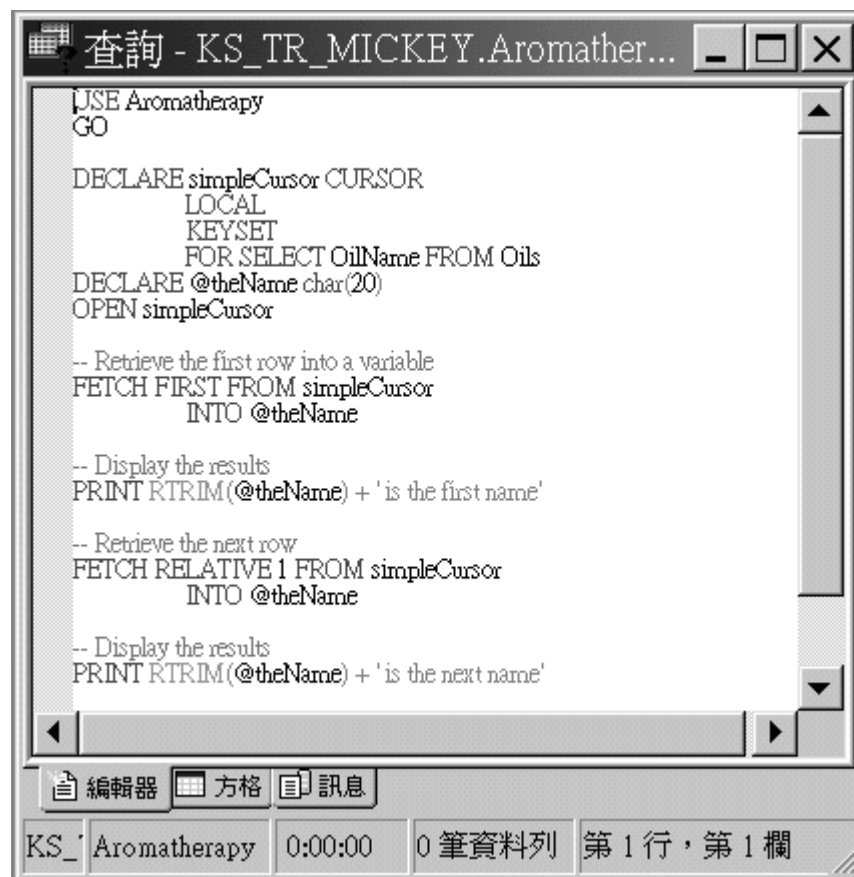
加载 SQL 指令码按钮

此 Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



2. 选取文件名称为 **FetchRelative** 的指令码档案，然后按一下 **开启** 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



3. 在 Query Analyzer 的工具列上按一下 **执行查询** 按钮。



执行查询按钮

此 Query Analyzer 会执行此查询。



假如您的数据指针是使用 FORWARD_ONLY 或 FAST_FORWARD 时，您唯一可以使用的移位方式是 NEXT。事实上，假如您的数据指针是这些型态的其中之一时，其 NEXT 就不需要。SQL Server 会将每一个 FETCH 都视为 FETCH NEXT。

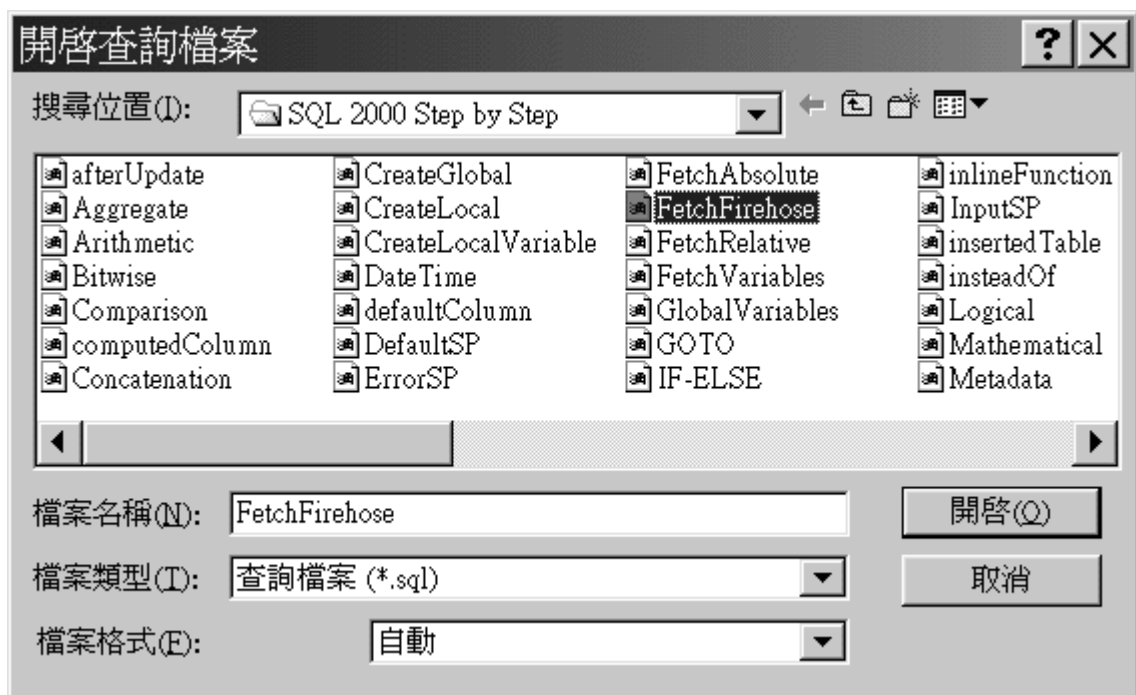
使用 **FETCH NEXT** 配合消防管数据指针

1. 在 Query Analyzer 工具列上按一下 **加载 SQL 指令码** 按钮。



加载 SQL 指令码按钮

此 Query Analyzer 会显示一个 **开启查询档案** 的对话框。



2. 选取文件名称为 **FetchFirehose** 的指令码档案，然后按一下 **开启** 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



3. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



執行查詢按鈕

此 Query Analyzer 會執行此查詢。



使用数据指针来更新及删除数据列

在数据指针集中将底层数据值进行更新或删除是相当简单的。有一个 **WHETE** 子句特殊的形式

可以藉由数据指针来进行更新：

```
UPDATE table_or_view
```

```
SET update_list
```

```
WHERE CURRENT OF cursor_or_variable
```

这个就叫做 [定位更新](#)（position update）。Trnsact-SQL 也提供一个 [定位删除](#)（position delete）

的形式：

```
DELETE table_or_view  
  
WHERE CURRENT OF cursor_or_variable
```

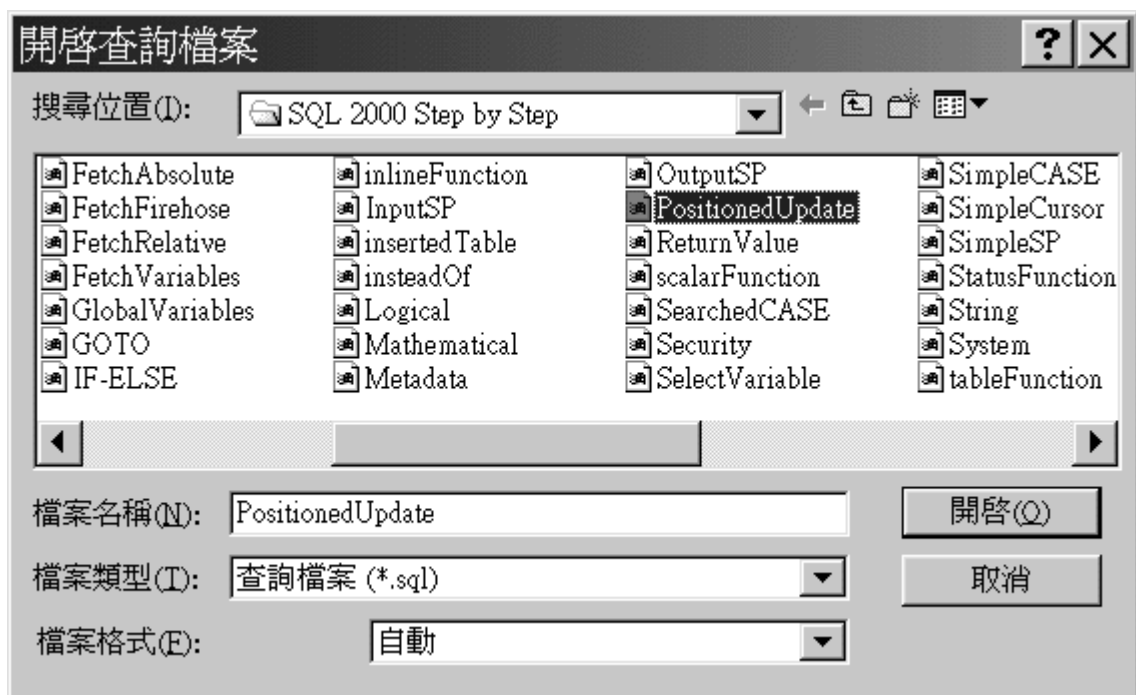
执行定位更新

1. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。



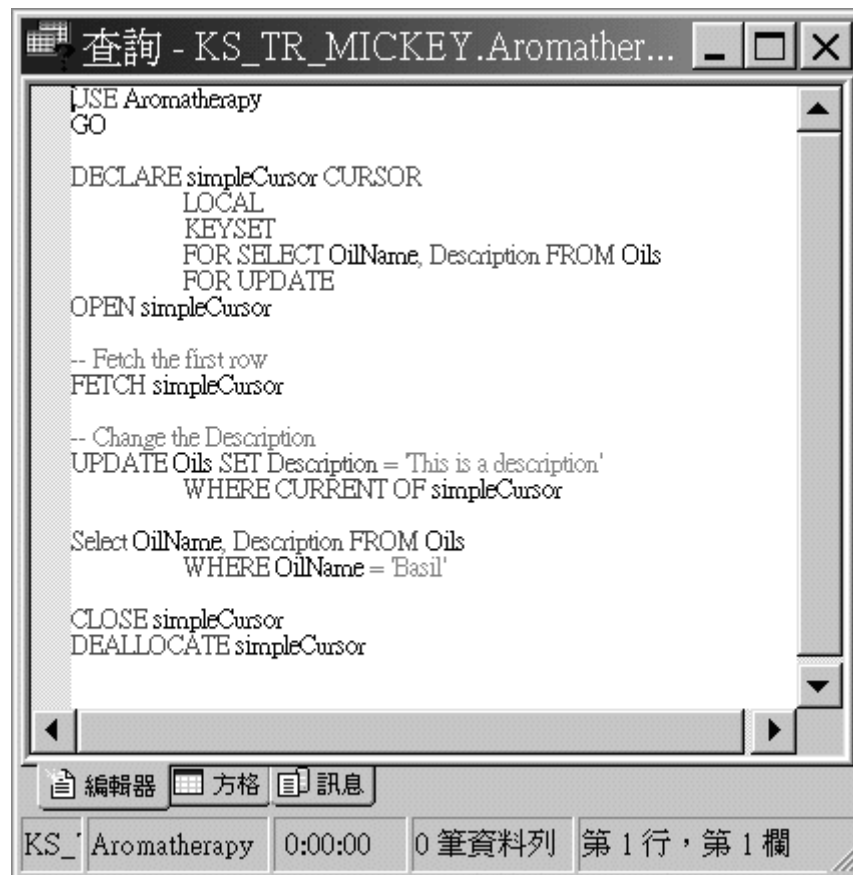
加载 SQL 指令码按钮

此 Query Analyzer 会显示 [开启查询档案](#) 的对话框。



2. 选取文件名称为 **PositionedUpdate** 的指令码档案，然后按一下 **开启** 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



3. 在 Query Analyzer 的工具列上按一下 [执行查询](#) 按钮。



执行查询按钮

Query Analyzer 会执行此查询。请注意窗口中会出现二个方格。第一个方格是由 **FETCH** 建立并且显示资料行的初始内容；第二个方格则是执行 **SELECT** 陈述式的执行结果，并且显示出其 **Description** 值已经被更新。



监控 Transact-SQL 数据指针

Transact-SQL 提供 2 个全域性变量以及一个函数来帮助您了解您的数据指针的进行状态。

@@CURSOR_ROWS 会传回联机时最近被开启的数据指针中的数据列数。表 27-1 列出

@@CURSOR_ROWS 所传回的值。

所传回的	说明
------	----

值	
-m	资料指针还没有被完全扩展（populate）；m 数据列目前位于数据指针集中。
-1	数据指针是动态的，并且任何在数据指针的数据列数目将会不断变更。
0	没有已经开启的数据指针，上次开启的数据指针已关闭或解除配置，或者数据指针内包含零笔数据列。
n	数据指针已完全扩展，n 是指数据指针中的数据列总数。

表 27-1 @@CURSOR_ROWS 所传回的值

@@FETCH_STATUS 是传回最后一个 FETCH 命令执行的状态。表 27-2 为

@@FETCH_STATUS 所传回的值。

所传回的值	说明
0	FETCH 陈述式顺利执行。
-1	FETCH 陈述式失败。
-2	所撷取的数据列已遗漏。

表 27-2 @@FETCH_STATUS 所传回的值

最后，Transact-SQL 提供 CURSOR_STATUS 函数。CURSOR_STATUS 函数的语法如下所示：

```
CURSOR_STATUS(type, cursor_or_variable )
```

在此 type 的参数值为 **local**、**global**、**variable**；cursor_or_variable 是指数据指针识别项或数据指针变量。表 27-3 为 CURSOR_STATUS 函数所传回的值。

所传回的值	说明
1	假如此函数是针对动态数据指针而被呼叫时，其动态数据指针的结果集可为 0、1 或多列资料列。假如此函数是针对其它型态数据指针时被呼叫时，其结果集至少会有一列资料列。
0	其数据指针结果集为空的。
-1	数据指针为关闭的。
-2	只传回数据指针变量。不是特定变量中的数据指针已经关闭，就是没有数据指针被指派给变量。
-3	特定数据指针或数据指针变量并不存在。

表 27-3 CURSOR_STATUS 函数所传回的值

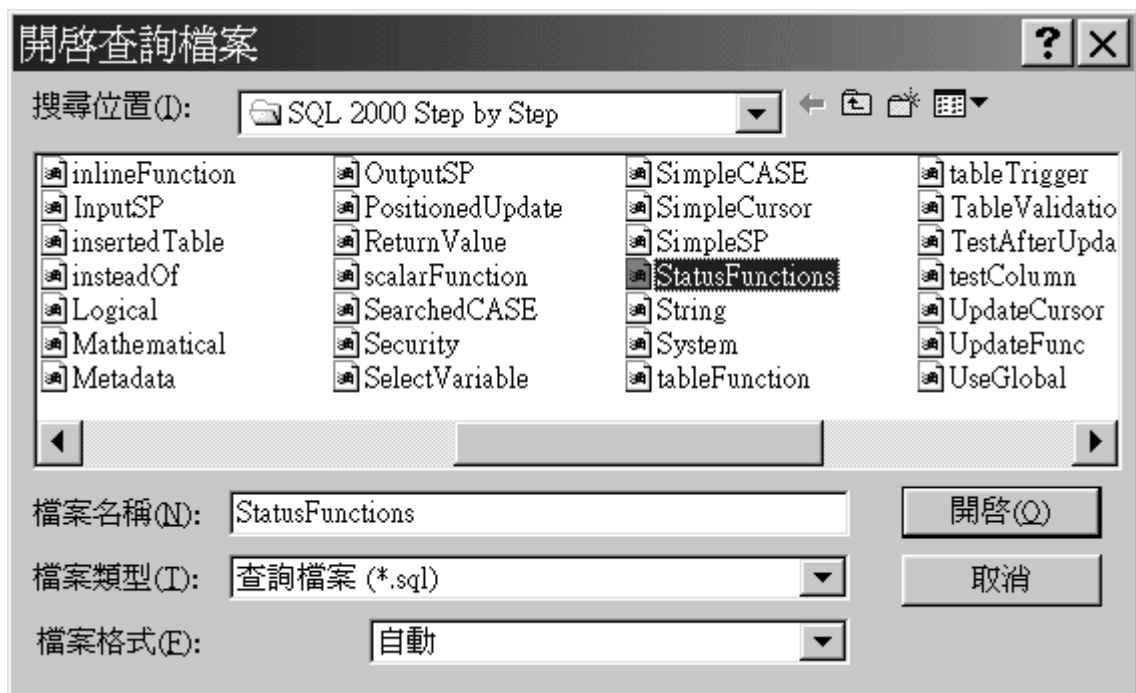
使用数据指针监控函数

1. 在 Query Analyzer 工具列上按一下 **加载 SQL 指令码** 按钮。



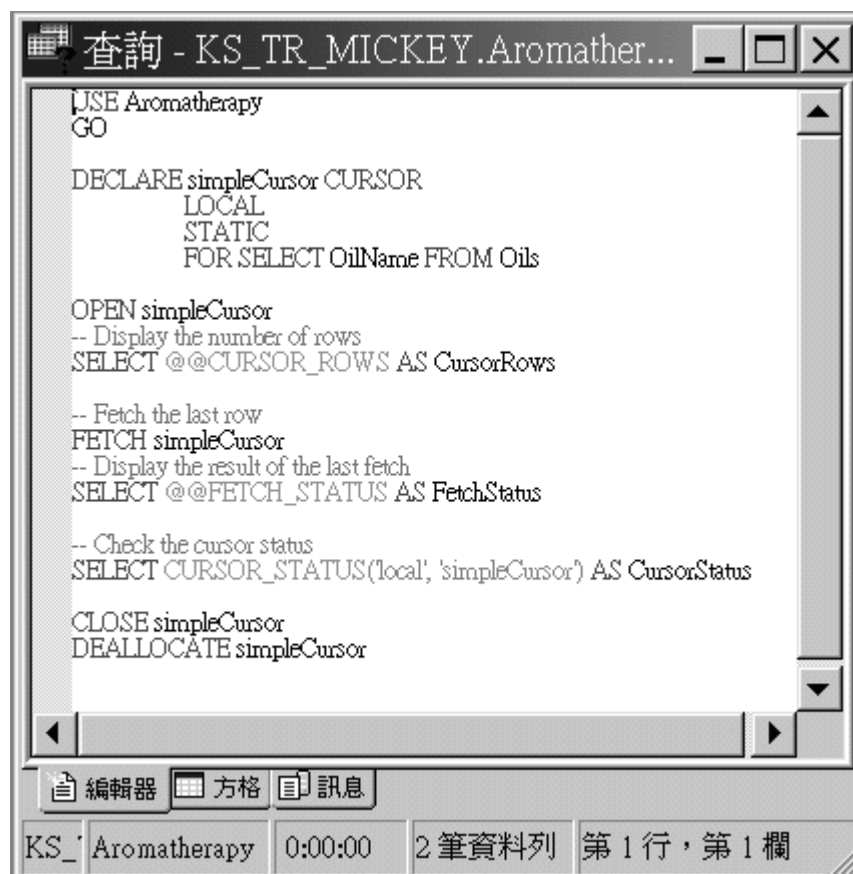
加载 SQL 指令码按钮

此 Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



2. 选取文件名称为 **StatusFunctions** 的指令码档案，然后按一下 [开启](#) 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。

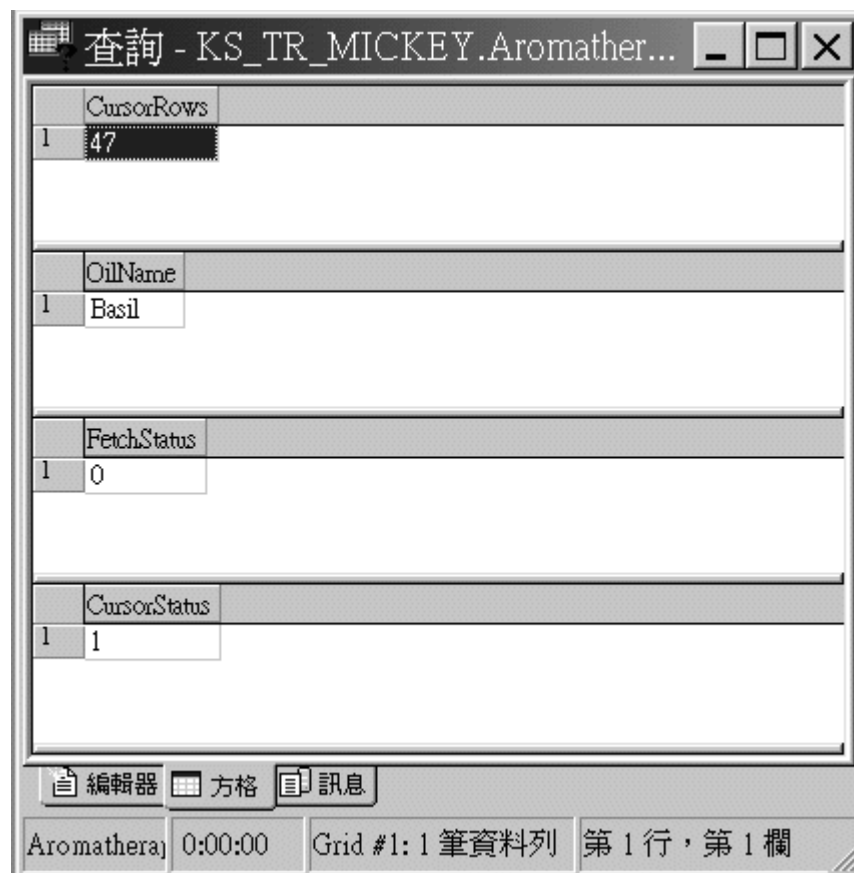


3. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按钮。



执行查询按钮

Query Analyzer 会执行此查询。请注意在窗口中会出现四个方格。第一个方格是执行 `SELECT @@CURSOR_ROWS` 陈述式所产生的结果、第二个方格是执行 `FETCH` 陈述式所产生的结果、第三个方格是执行 `@@FETCH_STATUS` 陈述式所产生的结果、第四个方格是执行 `SELECT CURSOR_STATUS('local', 'simpleCursor')` 陈述式所产生的结果。



本章总结

要执行的工作	SQL 语法
建立数据指针	<pre>DECLARE cursor_name CURSOR [visibility] [scroll] [type] [lock] [TYPE_WARNING] FOR select_statement [FOR UPDATE [OF column_names]]</pre>
开启数据指针	<pre>OPEN [GLOBAL] cursor_or_variable</pre>
关闭数据指针	<pre>CLOSE [GLOBAL] cursor_or_variable</pre>
解除配置数据指针	<pre>DEALLOCATE [GLOBAL] cursor_or_variable</pre>
使用简单的 FETCH 命令	<pre>FETCH cursor_or_variable</pre>
撷取数据列至变量中	<pre>FETCH cursor_or_variable INTO variable_list</pre>
藉由绝对位置撷取数据列	<pre>FETCH {FIRST LAST ABSOLUTE n} FROM</pre>

	cursor_or_variable
藉由相对位置撷取数据列	FETCH {NEXT PRIOR RELATIVE n} FROM cursor_or_variable
执行定位更新	UPDATE table_or_view SET update_list WHERE CURRENT OF cursor_or_variable
执行定位删除	DELETE table_or_view WHERE CURRENT OF cursor_or_variable
使用 CURSOR_STATUS 查询数据指针 的状态	CURSOR_STATUS ('local', 'local_cursor') 或 CURSOR_STATUS ('global', 'global_cursor') 或 CURSOR_STATUS('variable', 'cursor_variable')

28. 预存程序

在本章中，您将学习到：

- 执行简单的预存程序。
- 执行有输入参数的预存程序。
- 执行已经命名参数名称的预存程序。
- 执行使用 `DEFAULT` 参数关键词的预存程序。
- 执行有输出参数的预存程序。
- 执行有传回值的预存程序。
- 建立简单的预存程序。
- 建立有输入参数的预存程序。

- 建立有预设参数值的预存程序。
- 建立有输出参数的预存程序。
- 建立有传回值的预存程序。

在最后几章中，我们都使用 **SQL** 指令码（用以执行 **Transact-SQL** 陈述式批次操作并储存为文本文件）来执行操作。然而，**SQL Server** 也提供了预存程序来编写储存在服务器内的 **Transact-SQL** 批次操作。在本章中，我们将讨论如何使用以及建立预存程序。

认识预存程序

预存程序的用途并不是只有执行 **Transact-SQL** 陈述式而已。我们在前面已经使用过 **Transact-SQL** 指令码，一般我们都从应用程序直接传送指令码至数据库服务器。但是预存程序还有以下优点：

- 预存程序是数据库对象，它们是属于数据库档案。并且如果您进行卸离或复写数据库时，预存程序也会随档案一并被复制。

- 预存程序可以针对数据来作处理和接受，您可以将数据传递至程序中，当然也可以从程序中将结果传递回来。
- 预存程序以一个最佳化的形式储存，以便让它们能够快速地执行。

使用预存程序来交换数据

在前面章节中所编写和执行的 **SQL** 指令码并无法接收和传回任何信息，您只能从查询窗口内的 **方格** 或 **讯息** 窗格中显示它们所传回的资料。预存程序提供二种额外处理信息的方式：参数（**parameters**）和传回值（**return values**）。

参数是以比较特别的区域变量形式存在，并且为预存程序的一部份。您可以使用参数将信息传递至预存程序中（输入参数）或者是从预存程序中接受数据（输出参数）。

预存程序的传回值其实与函数的传回值十分相似，并且可以利用和函数相同的方式分配给区域变量。一般来说传回值通常是整数型态，在理论上它们可以用来传回任何的结果，但是一般习惯会将传回值用来传回预存程序的执行状态。举例来说，假如在执行的过程中没有产生任何错误，预存程序就会传回 **0**；而在执行的过程中若有产生任何错误，预存程序就会传回 **-1**。大多数功能比较强的预存程序可以传回不同的传回值，以便指示其发生的错误状态及种类。

请您不要将从预存程序所传回的参数和传回的任何结果集混淆了，这是很重要的。一个预存程序可以包含任意数目的 **SELECT** 陈述式，并且传回结果集。您不需要使用参数来接受这些结果集，它们会独自将数据传到应用程序中。

系统程序

预存程序可以包含二种类型：系统预存程序，它们是由 **SQL Server** 所建立；使用者自订的预存程序，它们是由您所建立的。系统预存程序是储存在 **master** 数据库内，并且所有的预存程序都是以 **sp_** 来作为起始字符。

提示

您可以使用 **sp_** 来当作预存程序的起始字符，但这不是一个好的做法，假如有一个系统预存程序的名称和您的自订预存程序相同，**SQL Server** 会在 **master** 数据库中寻找此系统预存程序，届时您所建立的预存程序将永远不会被执行。

在 **master** 数据库中存在许多系统预存程序，多数的系统预存程序都被计划用来执行一些管理工作，如同我们在本书第一篇所讨论的。举例来说，**sp_addlogin** 可以让您新增一个数据库登入账号，而 **sp_add_jobschedule** 可以让您新增一个排程来执行如数据库备份的工作。

提示

在 **SQL Server** 在线丛书中有所有系统预存程序的详细说明。

其它的系统预存程序可以帮您管理数据库对象。举例来说，**sp_rename** 可以帮您将数据库对象的名称更改，而 **sp_renamedb** 可以将数据库的名称更改。

提示

使用 **sp_renamedb** 系统预存程序是更改数据库名称的唯一方式，在 **Enterprise Manager** 中无法执行更改数据库的名称。

有一组重要的系统预存程序可以提供目前系统状态的相关数据：**sp_who** 提供关于目前系统的使用者和处理序的资料；**sp_cursor_list** 提供目前服务器联机所开启的数据指针清单；**sp_helpdb** 提供服务器内目前数据库的相关信息，对于任何单一的数据库来说，它会告诉您此数据库的实体位置数据以及交易记录档案的位置，您可以使用 **sp_help** 了解关于数据库对象的相关数据，它可

以列出数据库对象的型态、拥有者及名称，系统和使用者自订的数据型态以及预存程序的参数信息。

使用者自订的预存程序

就像数据库内的数据表一样，一个使用者自订预存程序是数据库对象的其中一种，您可以建立它，当然也可以移除它。在储存数据方面，它和其它数据库对象不同的地方是预存程序会储存 Transact-SQL 指令码。

提示

SQL Server 会在每一个数据库内建立一些预存程序，这些预存程序的名称都是以 `dt_` 为起始字符，它们是用来作为来源控制以及数据转换的服务。

执行一个预存程序的速度通常会比在应用程序中执行的批次操作的速度还要快，这是因为 SQL Server 可以在所执行的程序中执行一些复杂的步骤，并且将它们与预存程序储存在一起。

使用和建立预存程序

如同任何一个数据库对象一般，首先您必须要建立一个使用者自订预存程序，然后您才能使用它。

为了先让您了解如何使用预存程序，我们将要执行的二个步骤的顺序颠倒：首先我们先使用预存程序，然后再由您自己来建立。

使用预存程序

EXECUTE 陈述式是用来呼叫使用者自订及系统预存程序。假如预存程序没有要求参数，或者是它没有传回值时，其执行的语法是非常简单的：

```
EXECUTE procedure_name
```

提示

在批次操作中的第一个陈述式呼叫预存程序时，是不需要 **EXECUTE** 关键词的。然而，为了避免您忘记，最好使用 **EXECUTE**，或是它的缩写 **EXEC** 来执行预存程序。

执行简单的预存程序

1. 在 Query Analyzer 工具列上按一下 **新增查询** 按钮，以便开启一个新的查询窗口。



新增查询按钮

2. 在查询窗口中输入如下所列的陈述式：

```
EXECUTE sp_helpdb
```

3. 在 Query Analyzer 的工具列上按一下 **执行查询** 按钮。



执行查询按钮

此 Query Analyzer 会执行此预存程序并且在方格窗格中显示结果。

查詢 - KS_TR_MICKEY.Aro...				
	name	db_size	owner	dl
1	Aromatherapy	4.00 MB	KS_TR_MICKEY\Administrator	8
2	master	14.13 MB	sa	1
3	model	1.25 MB	sa	3
4	msdb	14.00 MB	sa	4
5	Northwind	4.63 MB	sa	6
6	pubs	2.50 MB	sa	5
7	tempdb	8.75 MB	sa	2

編輯器 方格 訊息

Ar 0:00:01 Grid #1: 7 筆資料列 第 1 行, 第 1 欄

重要

虽然您可以使用 `sp_help` 将目前服务器内所有的数据库显示出来,但是在您系统内的方格窗格中所显示的内容可能会和笔者所执行的结果不一样。

假如预存程序接收输入参数时,您可以藉由位置或名称来提供这些信息。要藉由位置以提供给预存程序参数,您只要在预存程序名称的后面加入此参数名称,并且以逗号「,」将每一个参数隔开:

```
EXECUTE procedure_name param [,param ...]
```

使用对象浏览器执行预存程序操作

在 [对象浏览器](#) 中会显示每一个数据库的 [预存程序](#) 数据夹，其中包含了 **master** 数据库在内，而在每个数据夹内也会列出预存程序的 [参数](#) 数据夹，因此您可以很快速地检查预存程序的参数名称以及它们的排列顺序。

执行有输入参数的预存程序

1. 在查询窗口内选取 [编辑器](#) 窗格，并且在 Query Analyzer 工具列上按一下 [清除窗口](#) 按钮。



清除窗口按钮

2. 在查询窗口中输入如下所示的陈述式：

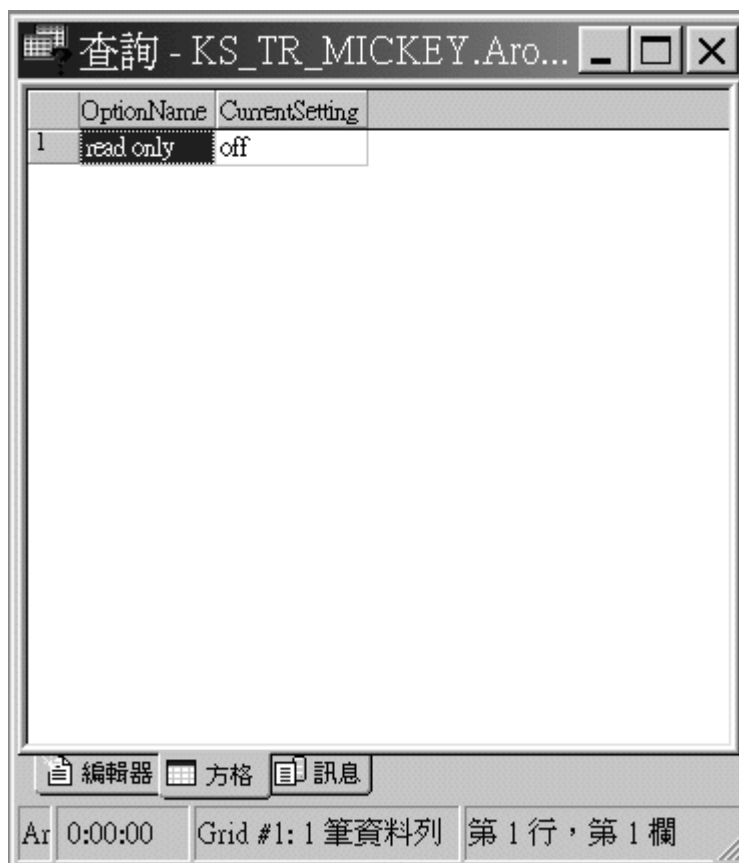
```
EXECUTE sp_dboption 'Aromatherapy ', 'read only '
```

3. 在 Query Analyzer 的工具列上按一下 [执行查询](#) 按钮。



执行查询按钮

此 Query Analyzer 会执行此预存程序并且在方格窗格中显示结果。



您也可以將已經命名的參數傳至預存程序中，假如您需要以這種方式輸入更多的參數時，您可以將這些參數以任何順序來傳遞，使用已經命名的參數的語法為：

```
EXECUTE stored_procedure @param_name = value[ ,@param_name =value...]
```

執行已經命名參數的預存程序

1. 在查詢窗口內選取 **編輯器** 窗格，並且在 Query Analyzer 工具列上按一下 **清除窗** 按钮。

2. 在查詢窗口中輸入如下所示的陳述式：

```
3. EXECUTE sp_dboption @optname ='read only ',
```

```
@dbname ='Aromatherapy '
```

4. 在 Query Analyzer 的工具列上按一下 [执行查询](#) 按钮。



执行查询按钮

此 Query Analyzer 会执行此预存程序并且在方格窗格中显示结果。



某些预存程序会以默认值来作为它们的参数，就像是数据表内其数据行的默认值一样。假如使用者没有指定值时，预存程序将会使用预设的参数值。假如您是使用已经命名的参数，那么使用默认值就更容易了，就像您没有为参数指定值一样。

假如您是以位置的方式来传递参数时，您可以依据参数列表内的所在位置来存取这些预设参数。

假如此参数是在列表的最后一个或者是只有一个参数时，您就可以忽略它。例如我们以没有指定数据库名称的方式来执行 `sp_helpdb`，假如参数并不是在参数列表的最后一个时，您可以使用特殊的关键词 `DEFAULT` 来告诉预存程序您要使用默认值。

执行使用 **DEFAULT** 参数关键词的预存程序

1. 在查询窗口内选取 [编辑器](#) 窗格，并且在 Query Analyzer 工具列上按一下 [清除窗口](#) 按钮。

2. 在查询窗口中输入如下所示的陈述式：

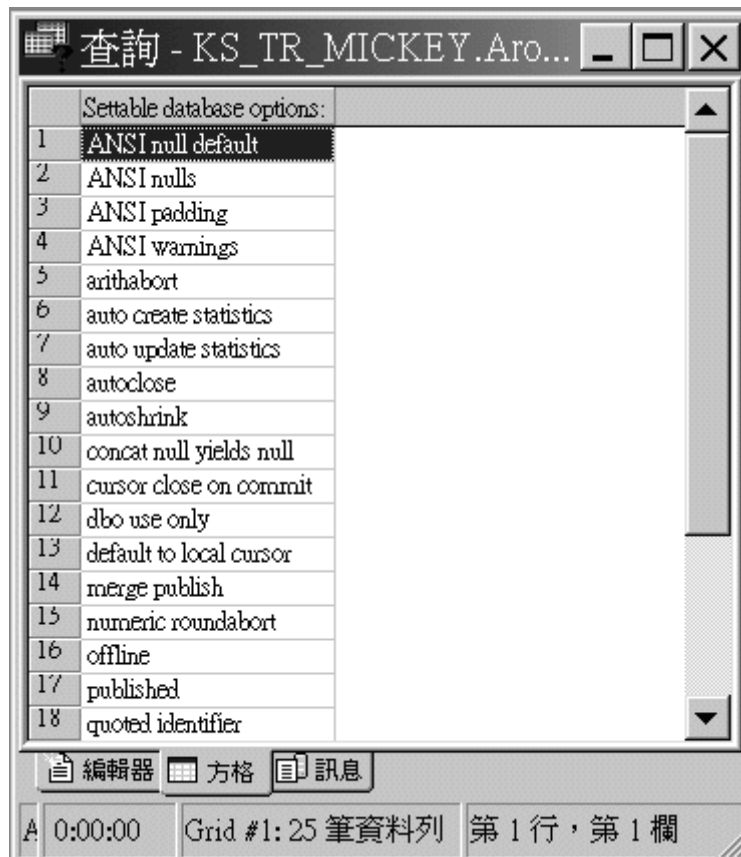
```
EXECUTE sp_dboption DEFAULT,'read only '
```

3. 在 Query Analyzer 的工具列上按一下 [执行查询](#) 按钮。



执行查询按钮

此 Query Analyzer 会执行此预存程序并且在方格窗格中显示结果。



除了接收传递给预存程序的数据，预存程序也可以使用输出参数将数据传送出来。

输出参数必须为区域变量，可以使用名称或位置指定，但是必须还要加入 **OUTPUT** 关键词。

执行有输出参数的预存程序

1. 在查询窗口内选取 **編輯器** 窗格，并且在 Query Analyzer 工具列上按一下 **清除窗**

□ 按钮。



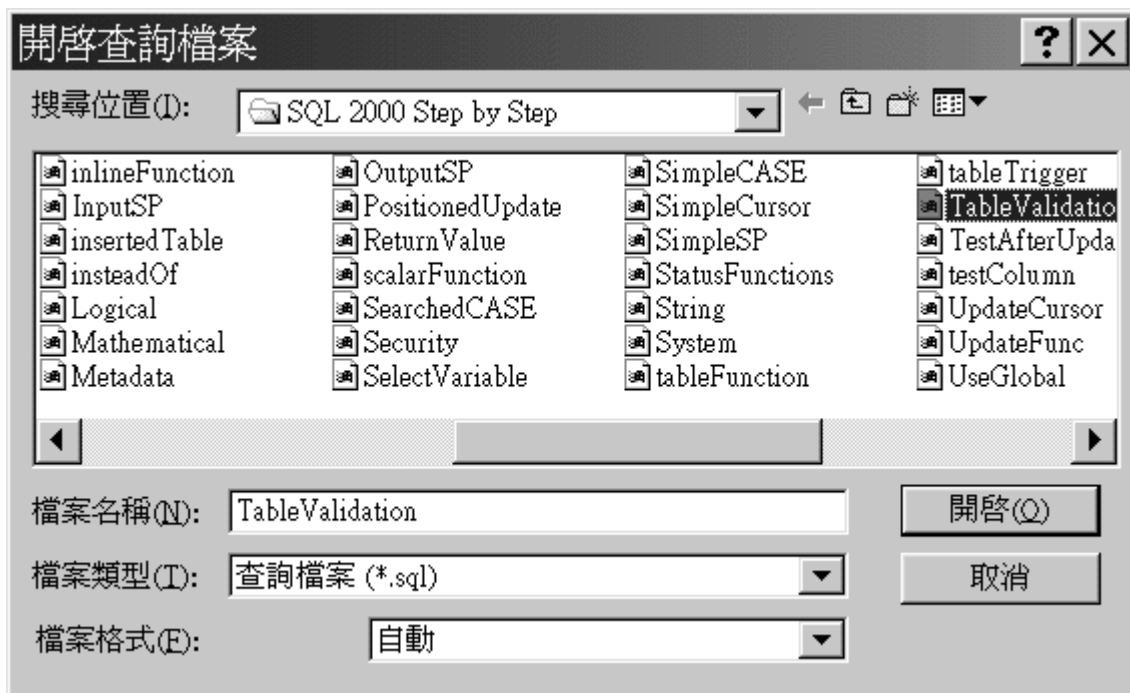
清除窗口按钮

2. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。
-



加载 SQL 指令码按钮

Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



3. 將檔案數據夾移至 **SQL 2000 Step by Step** 數據夾，並選取文件名稱

為 **TableValidation** 的指令碼檔案，然後按一下 **開啓** 按鈕。

此 Query Analyzer 會將指令碼檔案加載到查詢窗口中。



4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



執行查詢按鈕

此 Query Analyzer 會執行此預存程序並且在方格窗格中顯示結果。



指定预存程序的传回值的语法是混合 EXECUTE 陈述式与 SET 陈述式：

```
EXECUTE @variable_name = stored_procedure [,param [ ,param ... ] ]
```

大多数的系统预存程序都拥有传回值，但是就像预设参数一样，您可以忽略它们。假如您没有指定区域变量以接收传回值，SQL Server 将会扔掉它们。

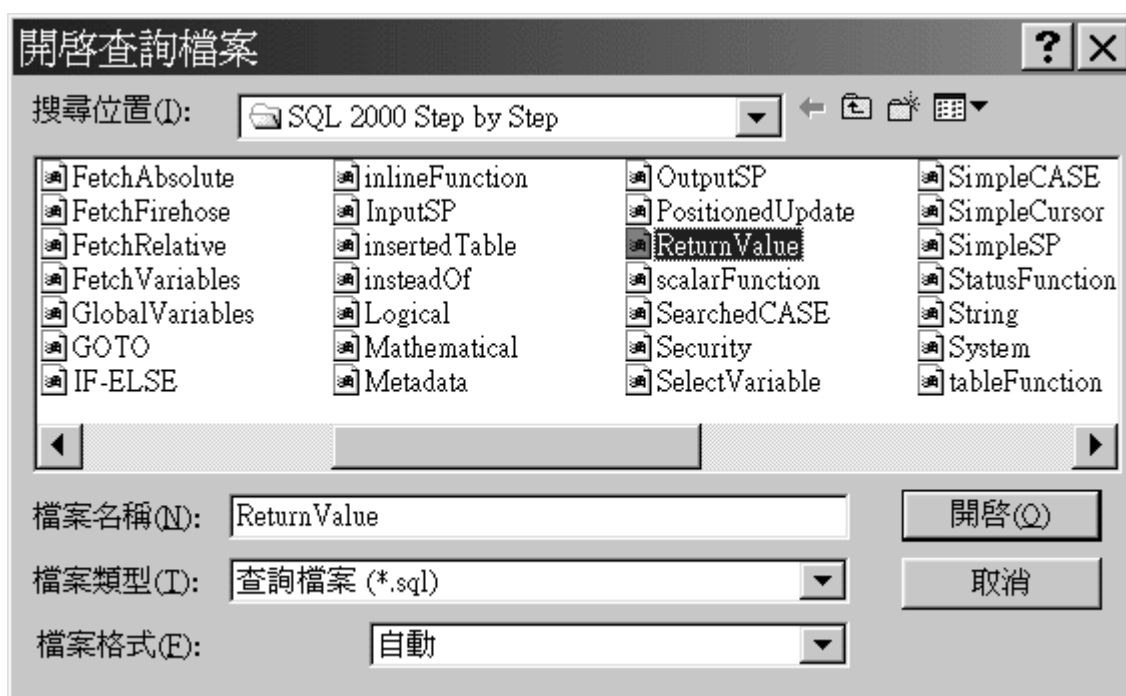
执行有传回值的预存程序

1. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。



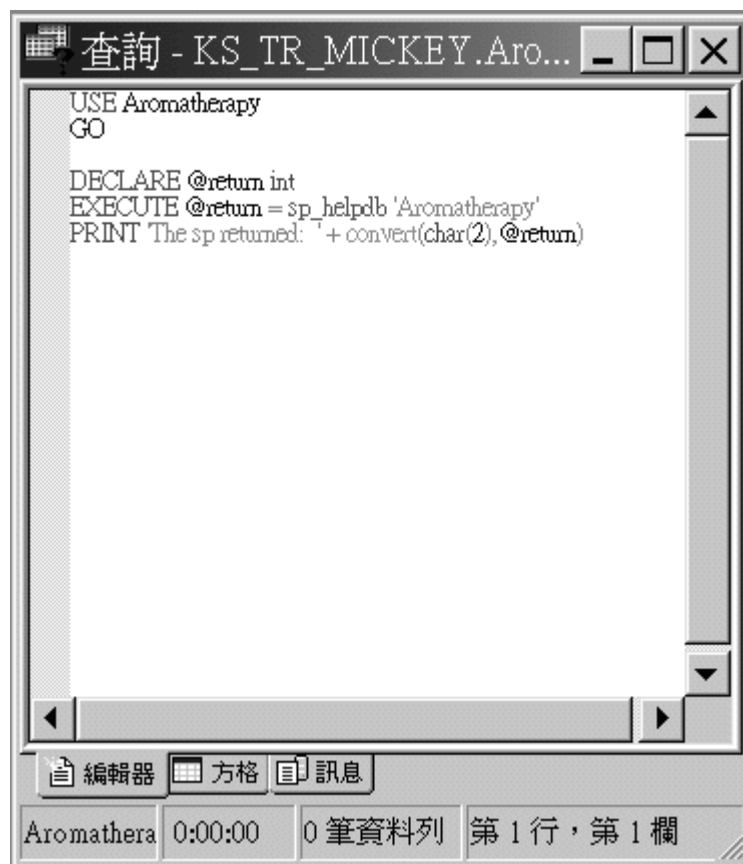
加载 SQL 指令码按钮

Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



2. 选取文件名称为 **ReturnValue** 的指令码档案，然后按一下 **开启** 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



3. 在 Query Analyzer 的工具列上按一下 **执行查询** 按钮。

此 Query Analyzer 会执行此预存程序并且在方格窗格中显示结果。

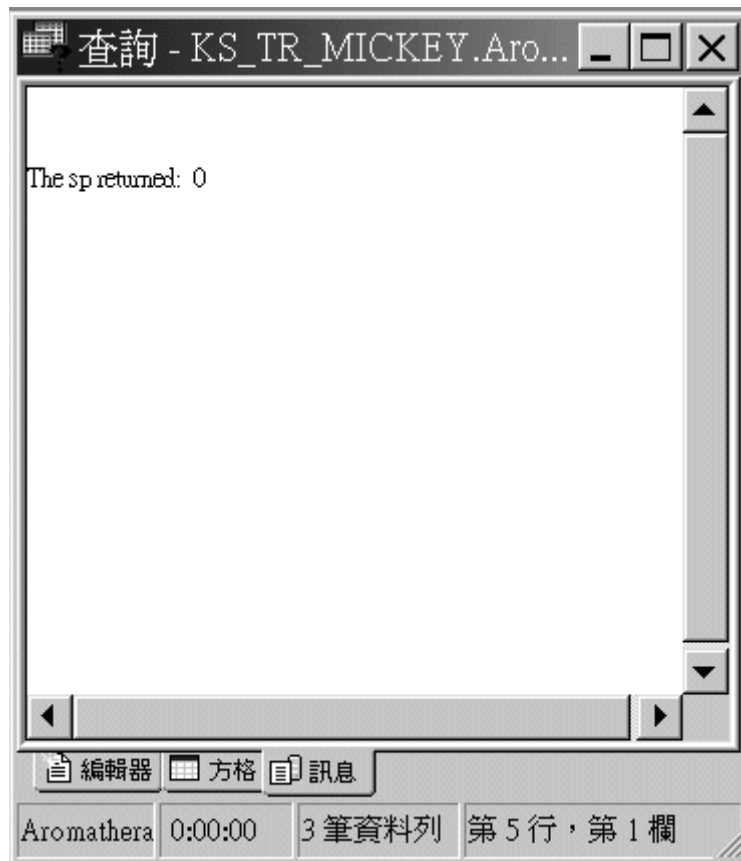


执行查询按钮



4. 选取 **訊息** 卷标页。

此时 Query Analyzer 会显示 PRINT 陈述式的执行结果，并且会显示所传回的值。



建立預存程序

也许您已经知道是以 **CREATE** 陈述式的相关方式来建立预存程序，建立预存程序的指令是

CREATE PROCEDURE，而 **CREATE PROCEDURE** 陈述式的语法为：

```
CREATE PROCEDURE procedure_name
```

```
[parameter_list ]
```

```
AS
```



```
procedure_statements
```

其中 `procedure_name` 是必要选项，当成识别项使用。

提示

您可以在预存程序名称的开头以「#」或「##」来代表您所建立的是暂时性的区域或全域预存程序。

其中 `procedure_statement` 是放置在 `CREATE` 陈述式内 `AS` 的后面，是用来执行预存程序的，它和我们所编写的指令码是一样的。事实上，您可以将 `AS` 关键词之前的内容当作是一个 `SQL` 指令码。

预存程序可以呼叫其它的预存程序，这样的呼叫方式会形成巢状（`nesting`）。事实上，一个预存程序可以呼叫其它预存程序，最多可呼叫达 32 层。

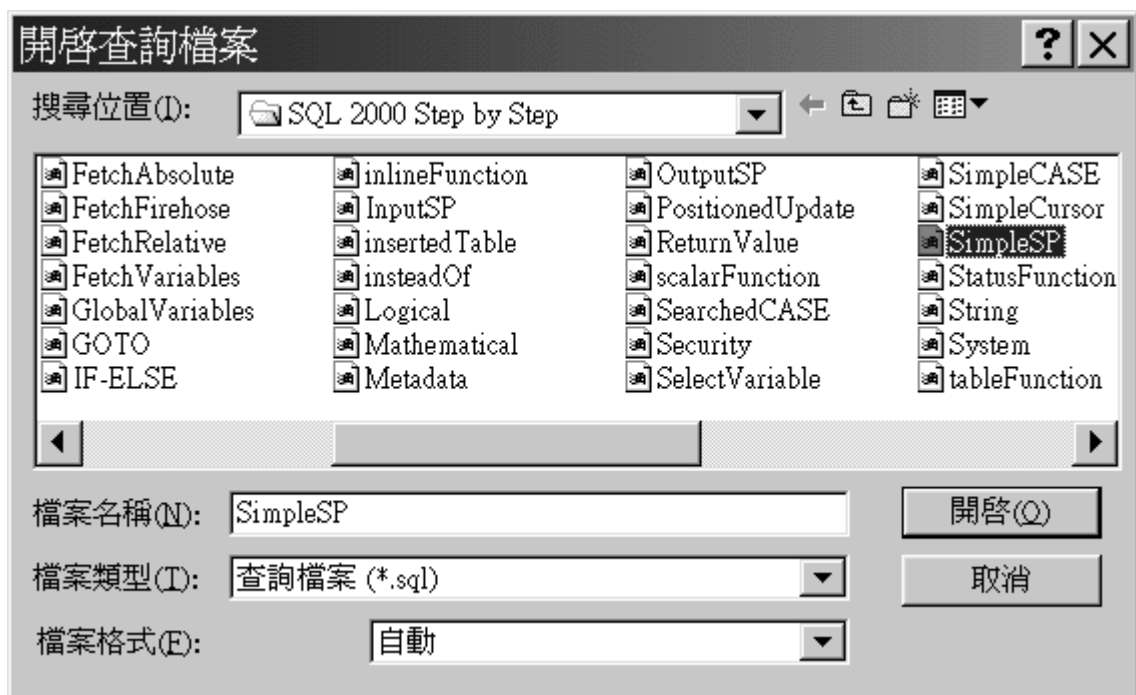
建立简单的预存程序

1. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。
-



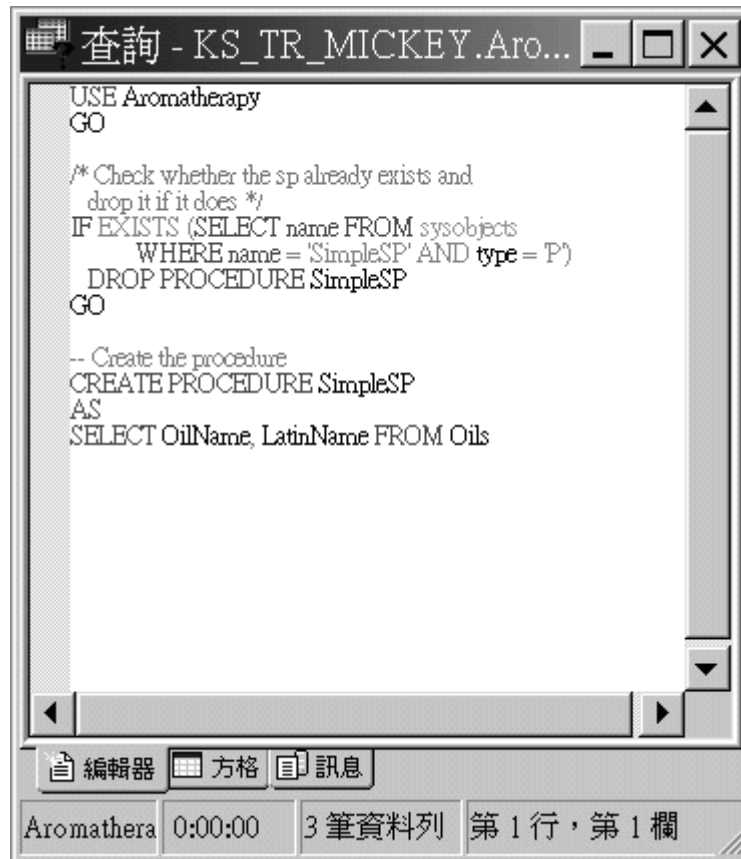
加载 SQL 指令码按钮

Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



2. 选取文件名称为 [SimpleSP](#) 的指令码档案，然后按一下 [开启](#) 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



3. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



執行查詢按鈕

此 Query Analyzer 会执行此预存程序并且在方格窗格中显示结果。



4. 在 Query Analyzer 工具列上按一下 [新增查詢](#) 按钮，以便开启一个新的查询窗口。



新增查询按钮

此时 Query Analyzer 会开启一新的查询窗口。

5. 在编辑器窗格中输入如下所示的陈述式：

```
EXECUTE SimpleSP
```

6. 在 Query Analyzer 的工具列上按一下 **执行查询** 按钮。



执行查询按钮

此 Query Analyzer 会执行此预存程序并且在方格窗格中显示结果。

查詢 - KS_TR_MICKEY.Aro...

	OilName	LatinName
1	Basil	Ocimum Basilicum
2	Bergamot	Citrus Bergamia
3	Black Pepper	Piper Nigrum
4	Cedarwood	Cedrus Atlantica, Juniperus Virginiana
5	German Chamomile	Matricaria Chamomilla
6	Roman Chamomile	Anthemis Nobilis
7	Cinnamon	Cinnamomum Zeylanicum
8	Citronella	Cymbopogon Nardus
9	Clary Sage	Salvia Sclarea
10	Coriander	Coriandrum Sativum
11	Cypress	Cupressus Sempervirens
12	Eucalyptus	Eucalyptus Globulus
13	Frankincense	Buswellia Thurifera
14	Geranium	Pelargonium Graveolens
15	Ginger	Zingiber Officinale
16	Grapefruit	Citrus Paradisi
17	Jasmine	Jasminum Officinale

編輯器 方格 訊息

A 0:00:00 Grid #1: 47 筆資料列 第 1 行, 第 1 欄

7. 关闭此查询窗口，假如有出现一个提示对话框要求您是否要储存更改的查询时，请您选择 **否**。

在 **parameter_lis** 参数列表内的每一个参数拥有下列的结构：

```
@parameter_name data_type [=default_value ] [OUTPUT ]
```

其中 **parameter_name** 必须要遵识别项的规则。参数的名称就如同是区域性变量一样，通常是以「@」为前置字。其实参数是一个区域性变量，它们仅能在预存程序内才可以存在，在每一个预存程序内最多可以有 2,100 个参数。

而 `default_value` 也是一样，也是存在预存程序内，而且是当您没有在预存程序内指定输入参数时才会被使用。`OUTPUT` 关键词是一个选择性选项，它是用来当预存程序结束执行时会将输出参数目前的值传回给呼叫程序。

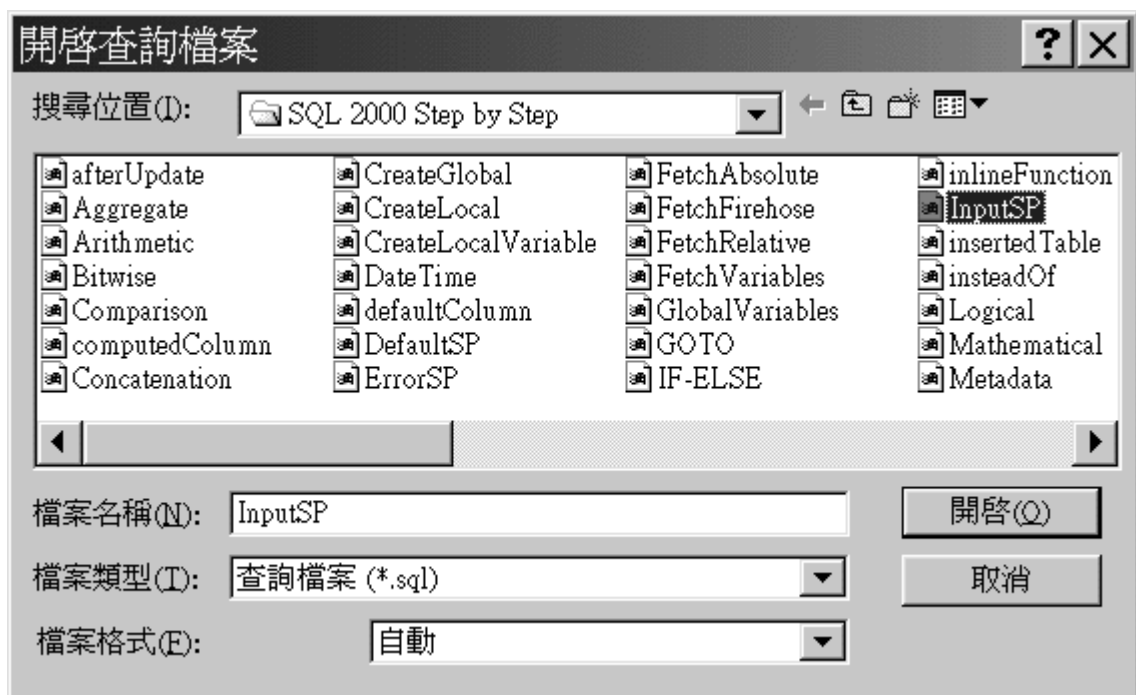
建立有输入参数的预存程序

1. 选取包含 `SimpleSP` 指令文件的查询窗口。
2. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。



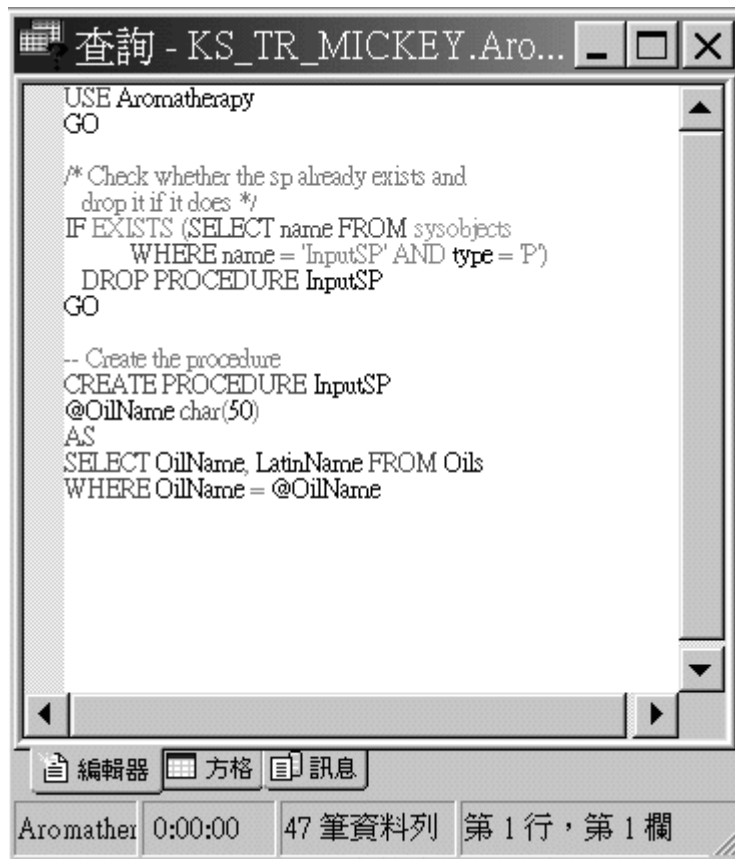
加载 SQL 指令码按钮

Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



3. 选取文件名称为 **InputSP** 的指令码档案，然后按一下 **开启** 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



執行查詢按鈕

此 Query Analyzer 會建立此預存程序。



5. 在 Query Analyzer 工具列上按一下 [新增查詢](#) 按钮，以便开启一个新的查询窗口。



新增查询按钮

此时 Query Analyzer 会开启一新的查询窗口。

6. 在编辑器窗格中输入如下所示的陈述式：

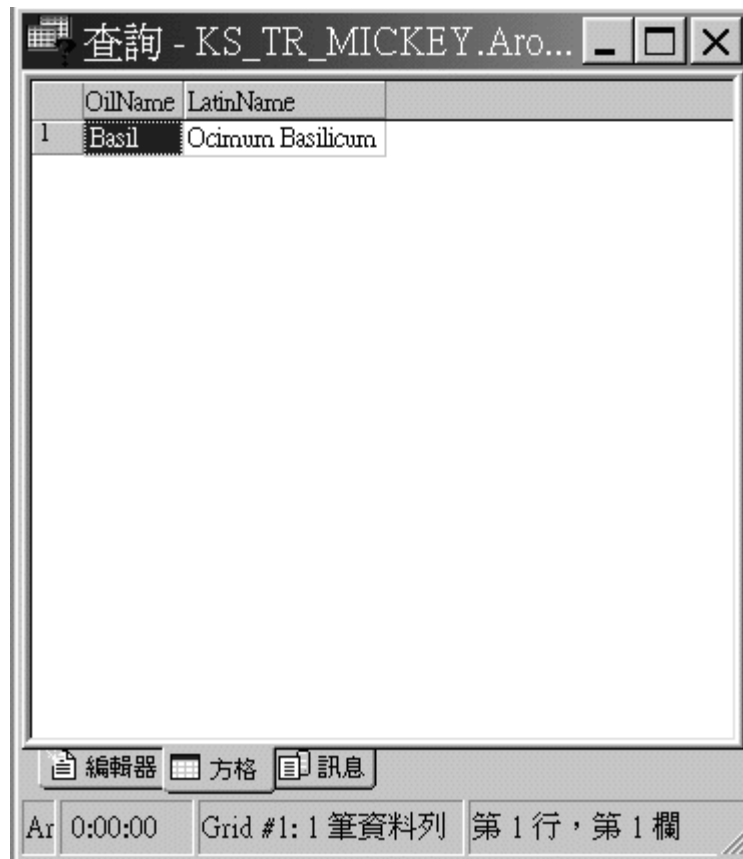
```
EXECUTE InputSP 'Basil '
```

7. 在 Query Analyzer 的工具列上按一下 [执行查询](#) 按钮。



执行查询按钮

此 Query Analyzer 会执行此预存程序并且在方格窗格中显示结果。



8. 关闭此查询窗口，假如有出现一个提示对话框要求您是否要储存更改的查询时，请您选择 [否](#)。

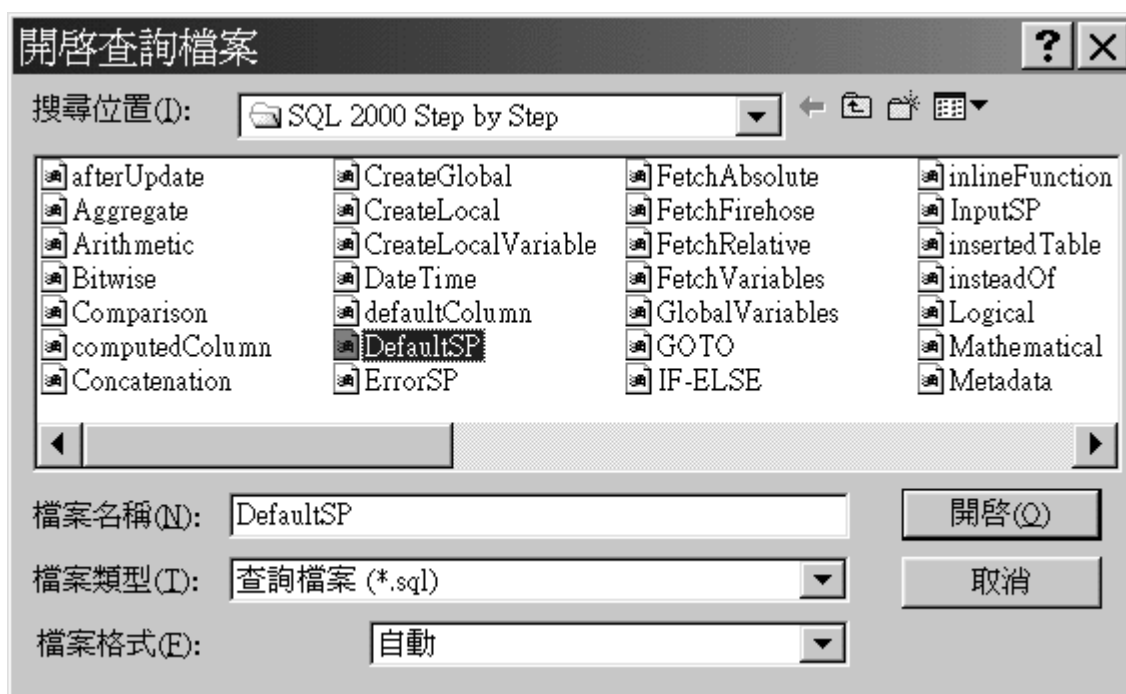
建立有预设参数值的预存程序

1. 选取包含有 InputSP 指令文件的查询窗口。
2. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。



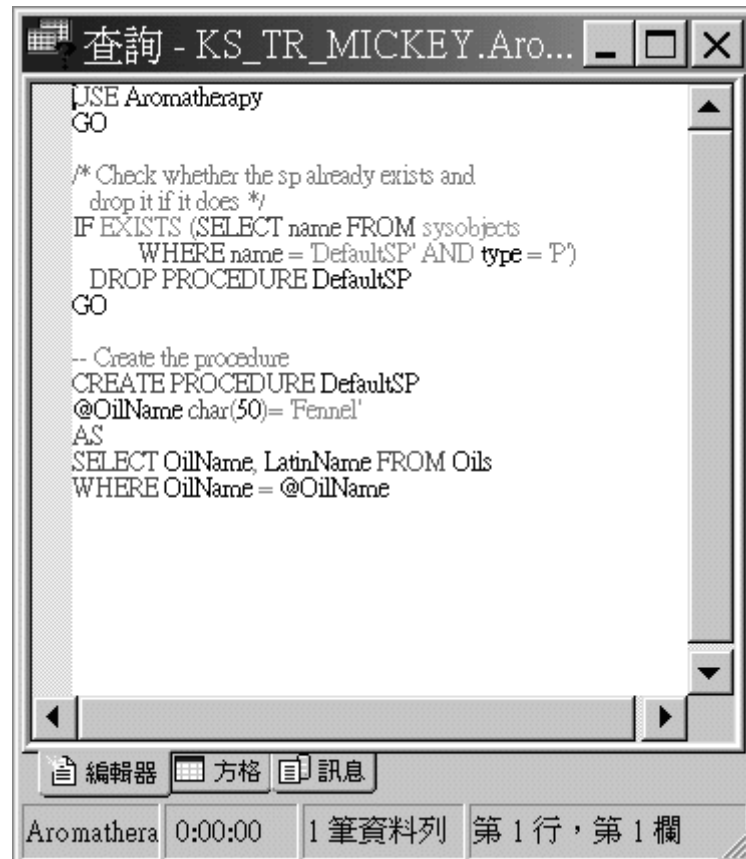
加载 SQL 指令码按钮

Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



3. 选取文件名称为 **DefaultSP** 的指令码档案，然后按一下 [开启](#) 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



執行查詢按鈕

此 Query Analyzer 会建立此预存程序。



5. 在 Query Analyzer 工具列上按一下 [新增查詢](#) 按钮，以便开启一个新的查询窗口。



新增查询按钮

此时 Query Analyzer 会开启一新的查询窗口。

6. 在编辑器窗格中输入如下所示的陈述式：

```
EXECUTE DefaultSP
```

7. 在 Query Analyzer 的工具列上按一下 **执行查询** 按钮。



执行查询按钮

此 Query Analyzer 会执行此预存程序并且在方格窗格中显示结果。



8. 关闭此查询窗口，假如有出现一个提示对话框要求您是否要储存更改的查询时，
请您选择 [否](#)。

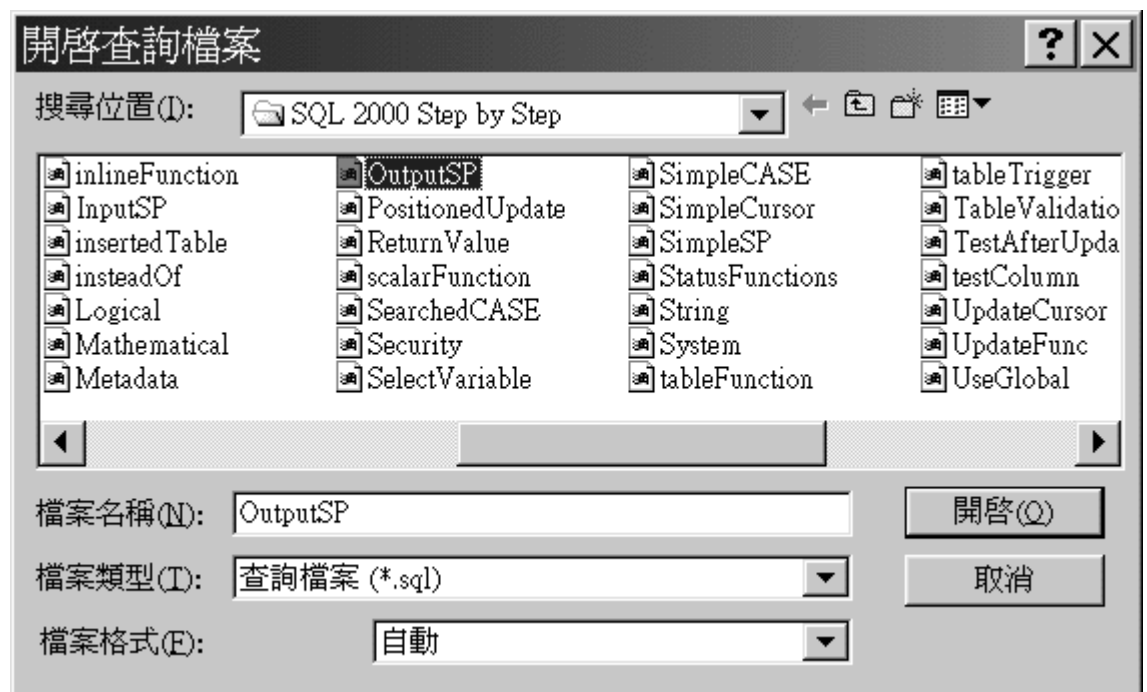
建立有输出参数的预存程序

1. 选取包含有 **DefaultSP** 指令文件的查询窗口。
2. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。



加载 SQL 指令码按钮

Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



3. 选取文件名称为 **OutputSP** 的指令码档案，然后按一下 [开启](#) 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



執行查詢按鈕

此 Query Analyzer 会建立此预存程序。



5. 在 Query Analyzer 工具列上按一下 **新增查詢** 按钮，以便开启一个新的查询窗口。



新增查询按钮

此时 Query Analyzer 会开启一新的查询窗口。

6. 在编辑器窗格中输入如下所示的陈述式：

```
7.      DECLARE @myOutput char(6)
8.      EXECUTE OutputSP @myOutput OUTPUT
SELECT @myOutput
```

9. 在 Query Analyzer 的工具列上按一下 [执行查询](#) 按钮。



执行查询按钮

此 Query Analyzer 会执行此预存程序并且在方格窗格中显示结果。



10. 关闭此查询窗口，假如有出现一个提示对话框要求您是否要储存更改的查询时，
请您选择 **否**。

传回值可以使用 **RETURN** 陈述式建置，利用下列形式：

```
RETURN(int)
```

在 **RETURN** 陈述式中，**int** 是指一个整数。就如同先前我们所看到的，通常传回值是指将预存程序执行后的执行状态传回，假如传回值是「0」时，就表示此预存程序的执行是成功的；假如传回其它的数值时，则表示此预存程序的执行是不成功的。您可以使用全域变量 **@@ERROR** 来测试最近所执行的 **T-SQL** 陈述式执行状态是否成功，「0」表示顺利；非零值表示有错误发生。

提示

SQL Server 将错误讯息字符串储存在 **master** 数据库内的 **sysmessages** 数据表中，您可以自己使用 **sp_addmessage** 系统预存程序来建立自己的错误讯息字符串，然后使用 **RAISERROR** 函数将此讯息传回客户端，以便当作服务器的错误讯息。

建立有传回值的预存程序

1. 选取包含 **OutputSP** 指令文件的查询窗口。
2. 在 **Query Analyzer** 工具列上按一下 **加载 SQL 指令码** 按钮。



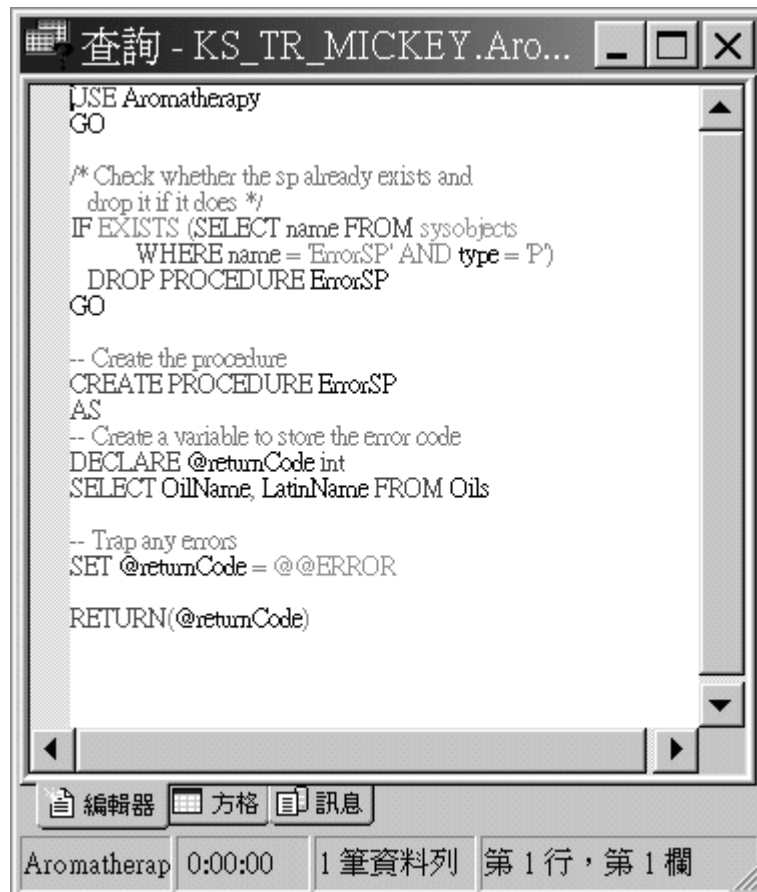
加载 SQL 指令码按钮

Query Analyzer 会显示一个 **开启查询档案** 的对话框。



3. 选取文件名称为 **ErrorSP** 的指令码档案，然后按一下 **开啓** 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



執行查詢按鈕

此 Query Analyzer 會建立此預存程序。



5. 在 Query Analyzer 工具列上按一下 [新增查詢](#) 按钮，以便开启一个新的查询窗口。



新增查询按钮

此时 **Query Analyzer** 会开启一新的查询窗口。

6. 在编辑器窗格中输入如下所示的陈述式：

```
7.      DECLARE @theError int
8.      EXECUTE @theError =ErrorSP
      SELECT @theError AS 'Return Value '
```

9. 在 **Query Analyzer** 的工具列上按一下 **执行查询** 按钮。



执行查询按钮

此 **Query Analyzer** 会执行此预存程序并且在方格窗格中显示结果。在第二个方格中所显示的「0」是表示此指令执行成功。



10. 关闭此查询窗口，假如有出现一个提示对话框要求您是否要储存更改的查询时，
请您选择 否。

本章总结

要执行的工作	SQL 语法
执行简单的预存程序	EXECUTE procedure_name
执行有输入参数的预存程序	EXECUTE procedure_name param[, param...]
执行有已经命名参数的预存程序	EXECUTE stored_procedure @param_name = value

	[, @param_name = value ...]
执行有输出参数的预存程序	在 EXECUTE 陈述式中的 @param_name 后面 使用 OUTPUT 关键词。
执行有传回值的预存程序	EXECUTE @variable_name = stored_procedure [, param [, param ...]]
建立预存程序	CREATE PROCEDURE procedure_name AS procedure_statements
从预存程序中传回值	RETURN (return_value)

29. 触发程序

在本章中，您将学习到：

- 建立 AFTER 触发程序。
- 建立 INSTEAD OF 触发程序。
- 使用 UPDATE 函数。
- 使用 inserted 及 deleted 数据表。

所谓的 **触发程序** (trigger) 是一种特殊形态的预存程序，当您使用 INSERT、UPDATE 或 DELETE 命令来修改数据列时，Microsoft SQL Server 会自动执行您所定义的触发程序。在本章中，我们将说明不同类型的触发程序，包括了功能强大的 INSTEAD OF 触发程序，同时也是 SQL Server 2000 的新功能。

认识触发程序

使用触发程序程序的好处是 **SQL Server** 会使用触发程序以自动执行该触发程序所定义的命令，这使得数据库功能性更为强大。因此只要您事先定义触发程序，您不需要担心数据库客户端（这包含了使用者或应用程序）是否了解所有的数据依存性和商业规则。

举例来说，一位业务员已经规划了一个订单总计等于或大于**\$10,000** 时，就必须要进行信用检查的规则。假如 **CreditApproved** 是指 **Customer** 数据表内的一个资料行，而现在有一笔订单的相关数据要新增至 **Order** 数据表中，此时您无法使用 **CHECK** 条件约束来执行此规则。但是更令人烦恼的是您要在数据库客户端强制这些规则更为困难，于是触发程序就可以提供在数据库本身进行建置此规则的机制。

触发程序也可以用于避免客户端变更数据库规则。继续前一个范例，假如将商业规则更改为金额大于或等于**\$15,000** 时才需要信用验证时，这个新的规则可以在数据库层级建置，而您不需要检查和更新所有客户端的应用程序。

触发程序最典型的使用方式是一商业规则无法使用数据表条件约束强制建置时，就使用触发程序。触发程序很早就已经需要串联参考完整性（**cascading referential integrity**）的功能，但是直到 **SQL Server 2000** 这个版本才支持。然而，现在我们可以资料表层级使用触发程序建置串联参考完整性，就像刚才的范例一样。大多数的商业规则会涉及多个数据表一例如计算总和需要使用到其它的数据表，这通常藉由触发程序会比较容易建置。

提示

纵使触发程序的执行效能并不是问题的重点所在（当它和数据指针一起使用时），您应该尽可能的使用较低阶的方式建置规则。假如您的商业规则可以利用 **CHECK** 条件约束建置时，您就不应该使用触发程序；而如果您可以使用 **UNIQUE** 条件约束时，您就不应该使用 **CHECK** 条件约束。

AFTER 触发程序

SQL Server 2000 支持了二种不同的触发程序型态：**AFTER** 触发程序和 **INSTEAD OF** 触发程序。

触发的情况分别如下：**AFTER** 触发程序将会在其所定义的命令执行后被触发；**INSTEAD OF** 触发程序将会代替其所定义的命令被触发。

您可以针对 **INSERT**、**UPDATE** 和 **DELETE** 命令建立 **AFTER** 触发程序。**AFTER** 触发程序只能在数据表上建立，而不能在检视表上建立。但是您可以针对这三个命令分别建立多个触发程序，相对地，单一的触发程序也可以套用这三个命令的任意组合。

提示

假如您针对单一个命令建立多个触发程序时，您可以使用系统预存程序 **sp_settriggerorder** 来设定该命令执行时的第一个和最后一个触发程序。

AFTER 触发程序会在所有的条件约束被处理之后再触发，假如有违反条件约束的情况发生时，该 **AFTER** 触发程序将不会被触发。

举例来说，假如在资料表中试图新增资料列而导致违反 **PRIMARY KEY** 条件约束时，该 **INSERT** 陈述式会在呼叫触发程序之前就发生执行失败。

INSTEAD OF 触发程序

INSTEAD OF 触发程序会取代其所定义的命令被触发。就像 **AFTER** 触发程序一样，您可以在 **INSERT**、**UPDATE** 或 **DELETE** 命令中定义 **INSTEAD OF** 触发程序。单一的触发程序可以套用多重命令的任意组合。

与 **AFTER** 触发程序不同的是，您可以针对资料表和检视表来建立 **INSTEAD OF** 触发程序，但是在数据表或检视表上的每一个动作，您只能建立单一个 **INSTEAD OF** 触发程序。

INSTEAD OF 触发程序不兼容于串联参考完整性。**INSTEAD OF DELETE** 与 **INSTEAD OF UPDATE** 触发程序不能定义于以 **DELETE** 或 **UPDATE** 动作定义外部索引键的数据表。

因为 **INSTEAD OF** 触发程序可以宣告在检视表中，所以它们非常地适用于建置检视表的功能性。

举例来说，针对一个包含有 **GROUP BY** 子句的检视表来说，**SQL Server** 会防止使用 **INSERT**

陈述式，但是却允许您针对该检视表定义一个 **INSTEAD OF INSERT** 触发程序。您可以使用该触发程序以将数据列插入定义检视表的底层数据表中。

BEFORE 触发程序

就本质上而言，是没有 **BEFORE** 触发程序的。但是 **INSTEAD OF** 触发程序可以代替其所定义的命令执行。如果没有 **INSTEAD OF** 触发程序存在，则该命令就会真的执行。

举例来说，假如您想要在 **INSERT** 之前检查某些条件时，您可以宣告一个 **INSTEAD OF INSERT** 触发程序。该 **INSTEAD OF** 触发程序将执行此检查，然后在数据表中执行 **INSERT**。该 **INSERT** 陈述式将会正常地执行，而不需要递归地呼叫 **INSTEAD OF** 触发程序。

建立触发程序

SQL Server 使用触发程序来处理一些条件约束。您无法使用触发程序 **CREATE**、**ALTER** 或 **DROP** 数据库、您无法使用触发程序将数据库或交易记录档案进行还原、并且您无法使用触发程序进行更改 **SQL Server** 设定的操作（您可以参考「**SQL Server** 在线丛书」所提供的完整说明）。

假如您在触发程序中更改数据库选项时，您所更改的选项只会在触发程序执行期间有效，但是在触发程序执行完毕之后，您所更改的值将会恢复到原来的值。

理论上，您可以使用 **RETURN** 陈述式自触发程序中传回一个值，但是您不应该依赖客户端应用程序来探知该触发程序是否存在或该触发程序作了什么。**RAISERROR** 命令提供了一个比较好的技术，因为大多数的应用程序的设计会掌控这些错误。

使用 **CREATE TRIGGER** 命令

就像其它的数据库对象一样，您可以使用 **CREATE** 陈述式的形式建立一个触发程序。**CREATE** 陈述式的基本语法为：

```
CREATE TRIGGER trigger_name  
  
ON table_or_view  
  
trigger_type command_list  
  
AS  
  
SQL_statements
```

其中 **trigger_name** 必须要遵守唯一性的规则，假如 **trigger_type** 是 **INSTEAD OF** 时，

table_or_view 可以是检视表名称，因为您只可以在检视表中定义 **INSTEAD OF** 触发程序。触发程序不能建立在暂存数据表或系统数据表中，但是它们可以引用暂存数据表。

`trigger_type` 关键词只能是 AFTER、FOR 或 INSTEAD OF 其中之一；`command_list` 可以联合使用任何 INSERT、UPDATE 或 DELETE 命令，假如您要使用一个以上的命令时，您可以使用逗号「，」将这些命令区隔开来。

提示

早期的 SQL Server 版本仅支持 AFTER 触发程序和 `trigger_type` FOR 语法，此语法在 SQL Server 2000 中也有支持，但是它与 AFTER 所表示的意思是相同的。

在 AS 关键词后面的 `SQL_statement` 所指的是触发程序所要执行的动作，触发程序除了不能包含参数之外，其余的与预存程序都相同。

建立 AFTER 触发程序

1. 在 Query Analyzer 工具列上按一下 **新增查询 按钮**，以便开启一个新的查询窗口。



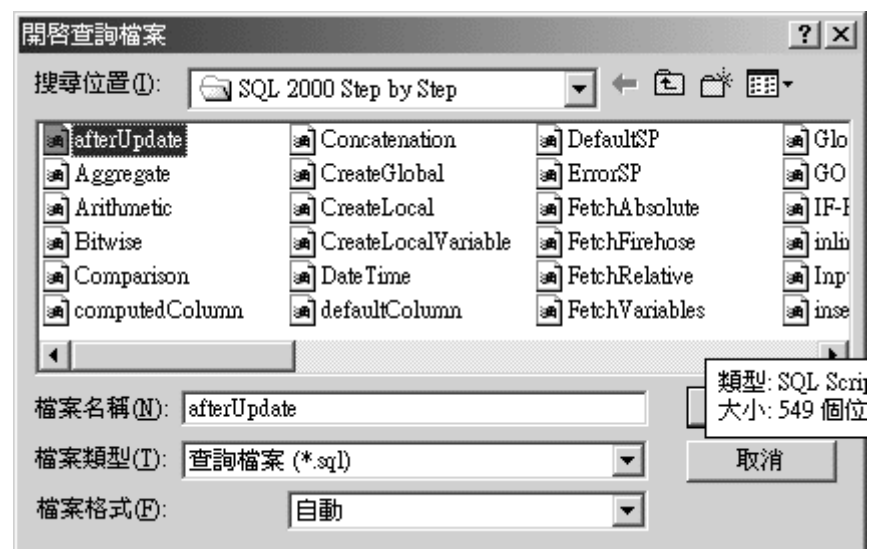
新增查询按钮

-
2. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。
-



加载 SQL 指令码按钮

此时 Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



3. 将档案数据夹移至 **SQL 2000 Step by Step** 数据夹中，并选取文件名称

为 **afterUpdate** 的指令码档案，然后按一下 **开启** 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



4. 在 Query Analyzer 的工具列上按一下 **执行查询** 按钮。



执行查询按钮

5. 在 Query Analyzer 工具列上按一下 [新增查询](#) 按钮，以便开启一个新的查询窗口。
-



新增查询按钮

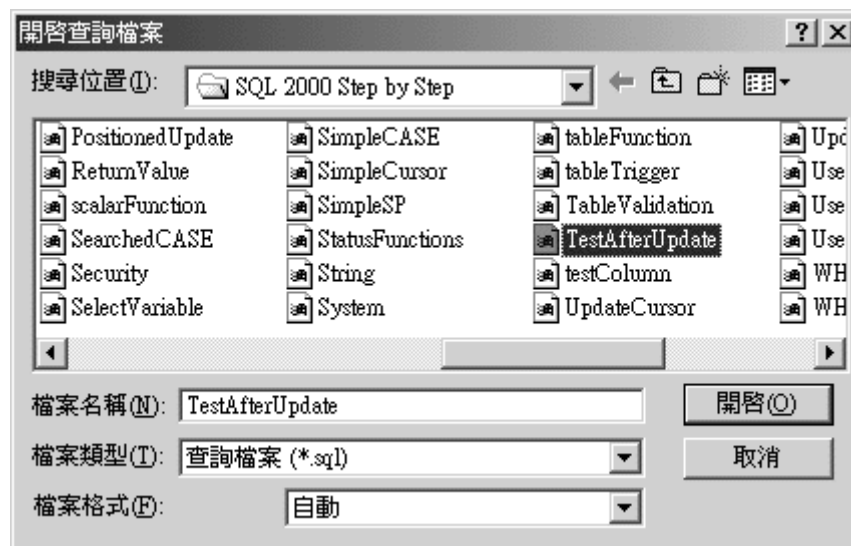
此时 Query Analyzer 会开启一新的查询窗口。

6. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。
-



加载 SQL 指令码按钮

此时 Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



7. 选取文件名称为 [TestAfterUpdate](#) 的指令码档案，然后按一下 [开启](#) 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



8. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



執行查詢按鈕

此 Query Analyzer 會執行此指令碼並且在方格窗格中顯示結果。

查詢 - MICKEY.Aromatherapy.MICKEY\Administrator - D:\SQL 2000 ...

	MessageID	TriggerName	MessageText
1	1	afterUpdate	Posted from the afterUpdate trigger

編輯器 方格 訊息

MICKEY\ Aromatherapy 0:00:00 Grid #1:0 筆資料列 第 1 行，第 1 欄

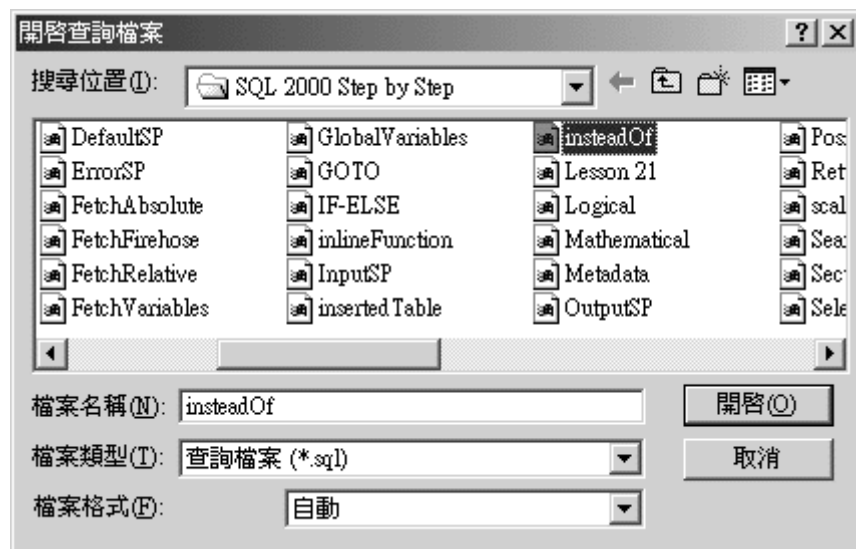
建立 INSTEAD OF 触发程序

1. 选取 afterUpdate 指令码的查询窗口。
 2. 在 Query Analyzer 工具列上按一下 **加载 SQL 指令码** 按钮。
-



加载 SQL 指令码按钮

此时 Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



3. 选取文件名称为 **insteadOf** 的指令码档案，然后按一下 [开启](#) 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



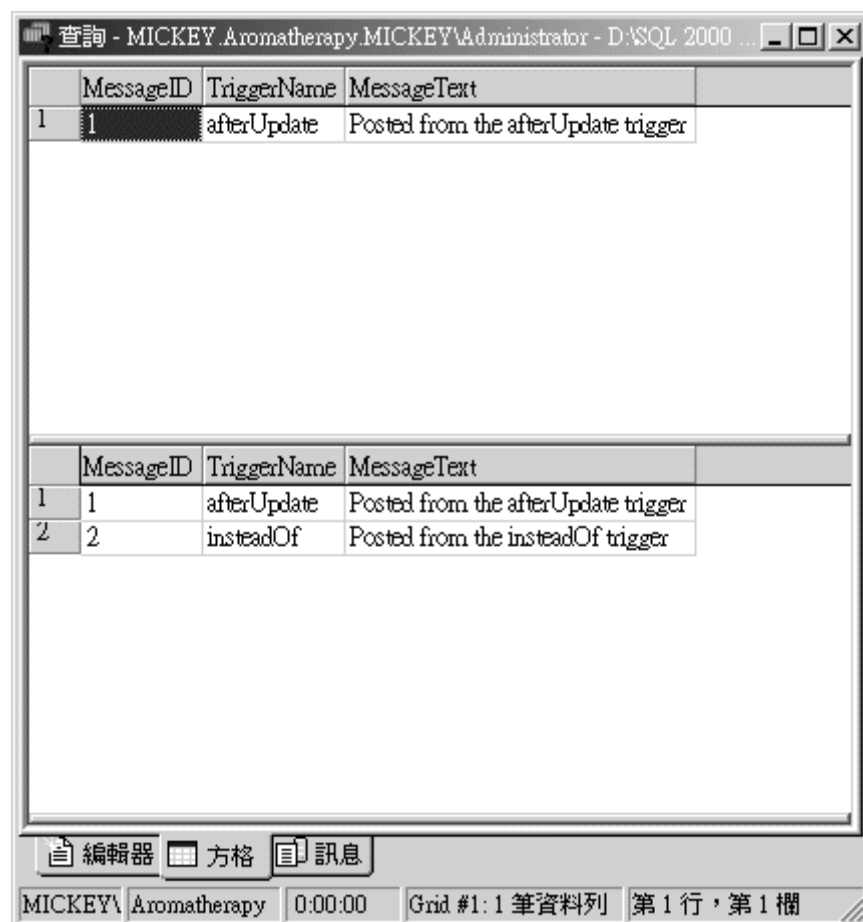
執行查詢按鈕

5. 选取 **TestAfterUpdate** 指令碼的查詢窗口。
6. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



执行查询按钮

此 Query Analyzer 会执行此指令码并且在方格窗格中显示结果。



使用 UPDATE 函数

SQL Server 提供了一个特殊的函数 UPDATE，它可以使用在触发程序之中以便决定在数据列中的特定数据行是否可以被更新。

```
UPDATE(column_name)
```

假如藉由 INSERT 或 UPDATE 命令将所指定的数据行内的数据值更改时，其 UPDATE 函数将会传回 TRUE 值。

提示

其它的 Transact-SQL 函数，COLUMNS_UPDATE 会传回一个将每一个已更新的数据行内的一个位设定的位屏蔽(bitmask)。假如您需要检查多个数据行的状态时，使用 COLUMNS_UPDATE 函数会比 UPDATE 函数还好用。

使用 UPDATE 函数

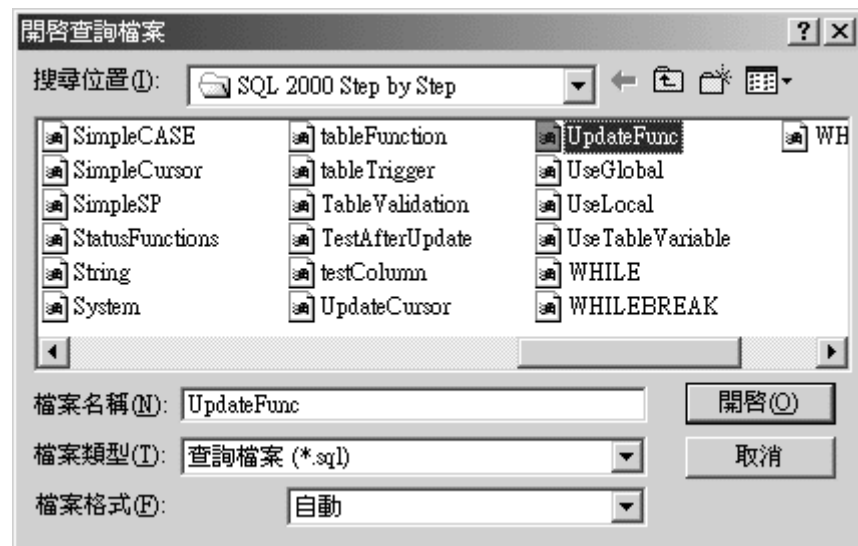
1. 选取 insteadOf 指令码的查询窗口。

2. 在 Query Analyzer 工具列上按一下 **加载 SQL 指令码** 按钮。



加载 SQL 指令码按钮

此时 Query Analyzer 会显示一个 **开启查询档案** 的对话框。



3. 选取文件名称为 **UpdateFunc** 的指令码档案，然后按一下 **开启** 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



執行查詢按鈕

此 Query Analyzer 会建立此触发程序。

5. 选取 **TestAfterUpdate** 指令文件的查询窗口。
6. 在 Query Analyzer 的工具列上按一下 **执行查询** 按钮。



执行查询按钮

此 Query Analyzer 会执行此指令码并且在方格窗格中显示结果。

查詢 - MICKEY.Aromatherapy.MICKEY\Administrator - D:\SQL 2000 ...

	MessageID	TriggerName	MessageText
1	1	afterUpdate	Posted from the afterUpdate trigger
2	2	insteadOf	Posted from the insteadOf trigger

	MessageID	TriggerName	MessageText
1	1	afterUpdate	Posted from the afterUpdate trigger
2	2	insteadOf	Posted from the insteadOf trigger
3	3	afterUpdate	Posted from the afterUpdate trigger
4	4	UpdateFunc	Description updated

編輯器 方格 訊息

MICKEY\ Aromatherapy 0:00:00 Grid #1: 2 筆資料列 第 1 行，第 1 欄

使用 inserted 及 deleted 数据表

SQL Server 建立二个数据表(inserted 和 deleted)以便在执行触发程序期间来操作数据。inserted 和 deleted 数据表是暂存性的内存常驻 (memory-resident) 数据表，其中包含触发程序的执行命令所影响的数据列。

当从 DELETE 命令呼叫一个触发程序时，deleted 数据表中将会包含自此数据表已经移除的数据列；对于 INSERT 命令来说，inserted 数据表中将会包含新增资料列的复本。一个 UPDATE 陈述式的执行方式其实是先执行一个 INSERT 陈述式然后再执行 DELETE 陈述式，因此 deleted

数据表将包含旧的数据值，而 **inserted** 数据表是包含新的资料值。您可以自从触发程序中引用这些数据表的内容，但是您不能修改它们。

请记得必须在数据表的变更完成后，才能呼叫 **AFTER** 触发程序，所以在底层数据表内的数据列将会先被更改。相反的，一个 **INSTEAD OF** 触发程序会取代其定义的动作执行，因此该数据表将不会被变更。事实上，除非 **INSTEAD OF** 触发程序执行了适当的命令，否则数据表的值将不会被变更。

重要

触发程序的呼叫是以命令为单位被呼叫一次（并非以数据列为单位），当您在撰写处理多个数据列的触发程序时必须很小心。当您呼叫触发程序时，如果在 **inserted** 或 **deleted** 中存在数据列，全域变量 **@@ROWCOUNT** 将会传回在这些数据表中的数据列数目。

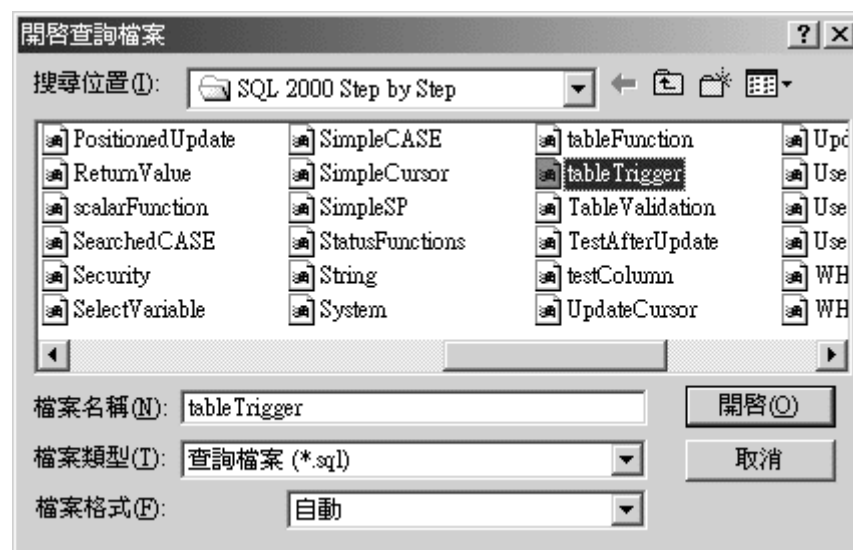
使用 **deleted** 数据表

1. 选取 **UpdateFunc** 指令码的查询窗口。
2. 在 **Query Analyzer** 工具列上按一下 **加载 SQL 指令码** 按钮。



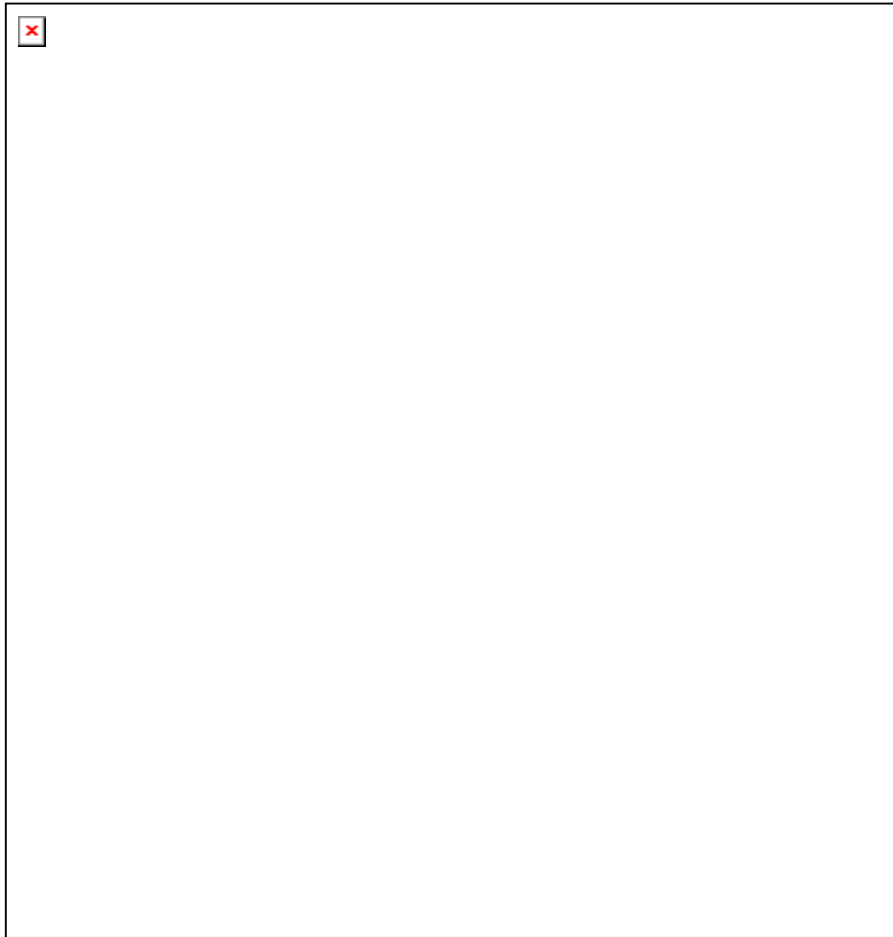
加载 SQL 指令码按钮

此时 Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



3. 选取文件名称为 **tableTrigger** 的指令码档案，然后按一下 [开启](#) 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



4. 在 Query Analyzer 的工具列上按一下 [执行查询](#) 按钮。



执行查询按钮

此 Query Analyzer 会建立此触发程序。

5. 选取 **TestAfterUpdate** 指令文件的查询窗口。
6. 在 Query Analyzer 的工具列上按一下 **执行查询** 按钮。



执行查询按钮

此 Query Analyzer 会执行此指令码并且在方格窗格中显示结果。

查詢 - MICKEY.Aromatherapy.MICKEY\Administrator - D:\SQL 2000 ...				
	MessageID	TriggerName	MessageText	
1	1	afterUpdate	Posted from the afterUpdate trigger	
2	2	insteadOf	Posted from the insteadOf trigger	
3	3	afterUpdate	Posted from the afterUpdate trigger	
4	4	UpdateFunc	Description updated	
	MessageID	TriggerName	MessageText	
1	1	afterUpdate	Posted from the afterUpdate trigger	
2	2	insteadOf	Posted from the insteadOf trigger	
3	3	afterUpdate	Posted from the afterUpdate trigger	
4	4	UpdateFunc	Description updated	
5	5	afterUpdate	Posted from the afterUpdate trigger	
6	6	UpdateFunc	Description updated	
7	7	tableTrigger	1 rows updated	
<div> <div>編輯器</div> <div>方格</div> <div>訊息</div> </div>				
MICKEY\Aromatherapy		0:00:00	Grid #1: 4 筆資料列	第 1 行，第 1 欄

本章总结

要执行的工作	SQL 语法
建立触发程序	<pre> CREATE TRIGGER trigger_name ON table_or_view trigger_type command_list AS SQL_statements </pre>

使用 UPDATE 函数

UPDATE (column_name)

30. 使用者自订函数

在本章中，您将学习到：

- 建立纯量函数。
- 建立内嵌数据表数值函数。
- 建立多重陈述式数据表数值函数。
- 在 Transact-SQL 陈述式内使用函数。
- 在 CREATE TABLE 陈述式内使用函数。

使用者自订函数（User-defined Function）是 SQL Server 2000 的新功能。使用者自订函数与预存程序的相似之处是它也接收输入参数和传回结果。然而，使用者自订函数比预存程序的功能还要强很多，并且有些功能甚至可以取代预存程序一例如用于数据表的定义，以及支持 **SELECT** 陈述式的 **FROM** 子句。

认识使用者自订函数

使用者自订函数可以依据它传回值的数据型别来分类，可以是纯量值（**scalar**，也称为数值类）或是数据表数值类（**table-valued**），以及由它们的 **决定机制**（**determinism**）来分类。函数的决定机制是取决于它传回结果值的一致性（**constancy**）。

如果您提供相同的输入值，函数就会传回相同的结果值，则该函数即为决定机制函数。举例来说，内建函数 **DATEADD**，假如您将 1958 年 4 月 20 日的资料加 3 天时，其所传回的结果值就是 1958 年 4 月 23 日。

如果您提供相同的输入值，而传回不同结果的输出值，则该函数便是非决定机制函数。举例来说，内建函数 **GETDATE** 就是非决定机制函数，每当它被呼叫时，每次它所传回的值都不一样。

使用者自订函数的决定机制是独立性的，无论数值型或数据表数值型，两者都可以是决定机制或非决定机制。假如使用者自订函数有引用到该函数有效范围以外的数据库对象时，您就可以将该使用者自订函数视为决定机制。

非决定机制的使用者自订函数不能使用于计算的数据行（**computed column**）上建立索引。丛集索引无法建立于引用任何非决定机制函数的检视表上。

数值类函数

一个数值类函数会传回一个数值结果值（单一值），例如一个字符串或一个数值。藉由数值类函数所传回的结果值其资料型别其实是有些限制的，非数值类型别例如数据指针和数据表很明显地

就被禁止。另外，数值类函数不能传回 **timestamp**、**text**、**ntext** 或 **image** 等数据类型别的结果值，也不可以传回使用者自订资料型别的结果值，就算它基本上是由数值类函数衍生而来的使用者自订数据类型别也不可以。

数据表数值函数

理所当然地，数据表数值的使用者自订函数可以传回数据表，而数据表数值函数不能放置在预存程序或检视表中。举例来说，在 **Aromatherapy** 范例数据库内的 **Oils** 数据表已经完整正规化，因此 **PlantParts** 和 **Cautions** 数据表的详细内容是储存在不同的数据表内。对于一般的使用者来说，这是没有多大意义的；并且对于大多数的人来说，使用 **SELECT** 陈述式将这三个数据表中的所有数据列联结在是一起十分困难的（反正规化）。

检视表可以隐蔽特定的信息不给客户端看见，但是检视表不支持参数的使用。因此无法提供在查询执行期间供客户端指出那些数据列要显示的方式；预存程序可以接收参数，但是预存程序无法用于 **SELECT** 陈述式中的 **FROM** 子句，因此预存程序在控制哪些数据列要显示的方面也十分拙劣。数据表数值的使用者自订函数可以克服这二个问題。

举例来说，当您呼叫使用者自订函数 **GetOilDetails** 时，它可以接收输入油品名称并且传回此油品的详细内容，您的客户端可以使用下列简单的 **Transact-SQL** 陈述式，并且接受其传回油品名称的详细内容来当作是结果值。

```
SELECT * FROM GetOilDetails(' Basil')
```

这对于简化客户端应用程序来说，这种功能明显地提供了许多好处。

建立使用者自订函数

与其它的数据对象一样，使用者自订函数可以使用 **CREATE** 陈述式来建立，而所建立的使用者自订函数的种类是依据语法的不同而不同。

使用者自订函数中的 **Transact-SQL** 陈述式在使用上有二个限制。第一个是函数不能有副作用（**side-effects**）——也就是它们不能让该函数有效范围以外的任何对象有永久性的变更。

举例来说，假如有一个暂存数据表是在函数内被建立，该函数中的陈述式可以新增、修改及删除此暂存数据表中数据列。然而，使用者自订函数不能在一个永久性的数据表内变更任何的数据列。

第二个限制是在使用者自订函数内的陈述式不能呼叫任何非决定机制的函数（不论是内建或使用者自订函数），或引用非决定机制的全域变量。例如 **@@TOTAL_ERRORS**，它会传回自 **SQL Server** 最后一次启动之后发生错误的数目。

建立数值类函数

数值类使用者自订函数的 **CREATE** 陈述式如下所示，数值类使用者自订函数可以用于任何地方（这些地方泛指其所传回的资料型别可以被使用的地方）。

```
CREATE FUNCTION function_name([parameter_list])

RETURNS data_type

AS

BEGIN

    [tsql_statements]

    RETURN (return_value)

END
```

其中，**function_name** 是指使用者自订函数的名称，并且其函数名称必须符合唯一性规则；

parameter_list 与预存程序内的参数清单的语法是一样的：

```
@parameter_name data_type [= default_value]
```

parameters_name 必须要遵守唯一性的规则，并且必须要以 **@** 作为起始字符。使用者自订函数最多可以有 **1024** 个输入参数，但是它们并不支持输出参数，这是因为它们所传回的值就是使用者自订函数本身的结果。请注意 **parameters_name** 是选择性的。

RETURNS 子句是用来定义函数传回值的型别。如同前面所述，数值类函数可以传回任意纯量系统数据型别的值（除了 **timestamp**、**text**、**ntext** 或 **image** 数据型别）。

BEGIN...END 中包含整个函数所需要的陈述式，并且是必备的。即使整个函数中只包含一个

RETURN 陈述式，您仍然需要使用 **BEGIN..END** 陈述式。

建立数值类函数

1. 在 Query Analyzer 工具列上按一下 [新增查询](#) 按钮，以便开启一个新的查询窗口。



新增查询按钮

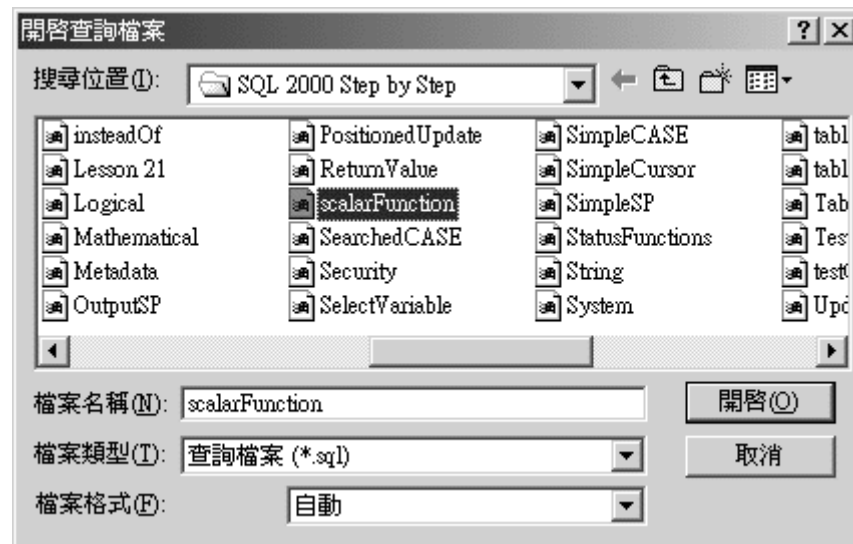
此时 Query Analyzer 会开启一个新的查询窗口。

2. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。



加载 SQL 指令码按钮

此时 Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



3. 将档案数据夹移至 **SQL 2000 Step by Step** 数据夹中，并且选取文件名称

为 **scalarFunction** 的指令码档案，然后按一下 **开启** 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



4. 在 Query Analyzer 的工具列上按一下 [执行查询](#) 按钮。

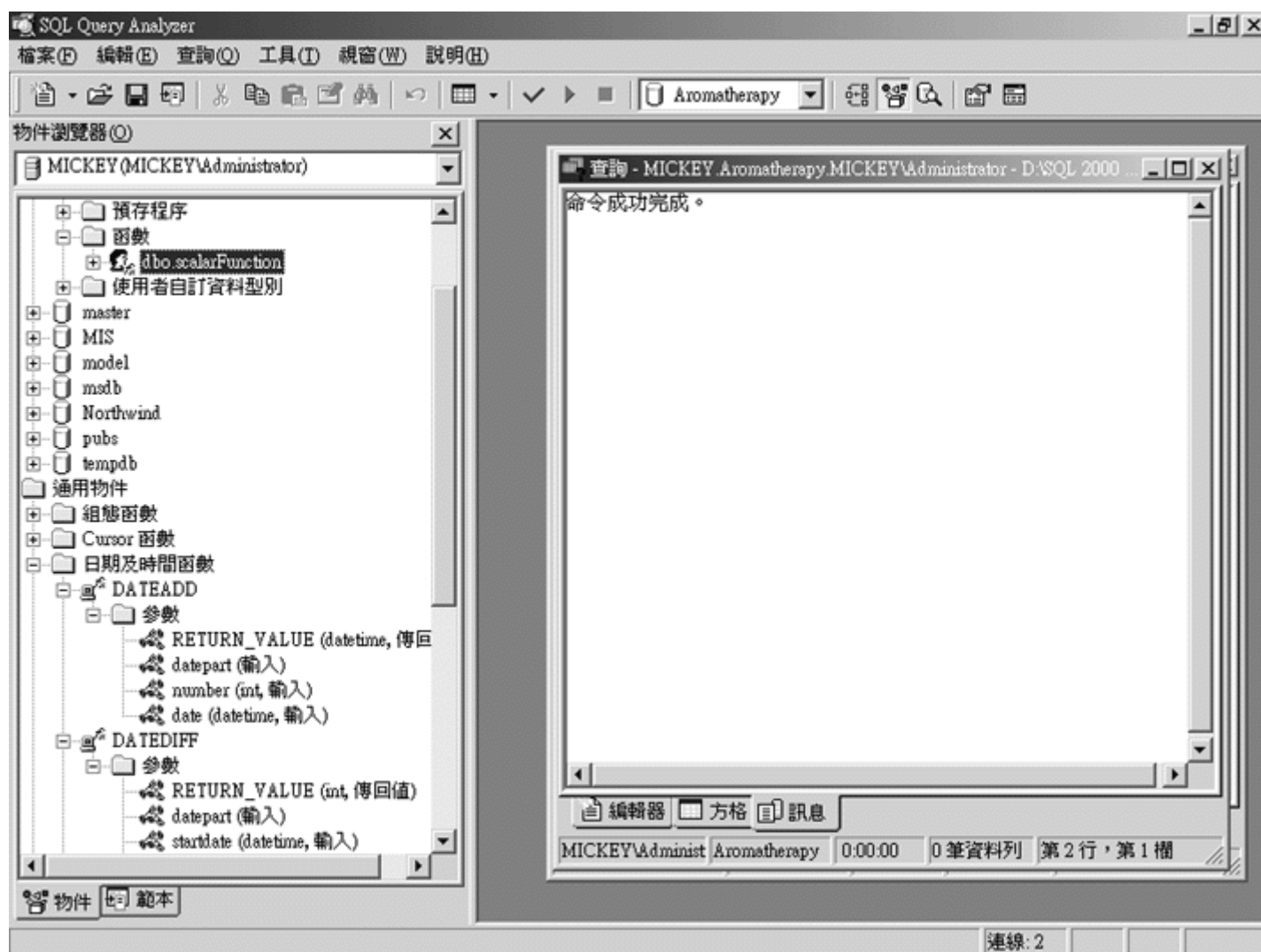


执行查询按钮

5. 将 [对象浏览器](#) 中的 Aromatherapy 数据库的 [函数](#) 数据夹展开, 并且按一下 **F5** 键

以便重新整理显示结果。

此时 对象浏览器 内会显示 dbo.scalarFunction 的函数名称。



说明

我们将在下一节中使用此使用者自订函数。

建立数据表数值函数

CREATE FUNCTION 陈述式支持二种不同型式的数据表数值函数：[内嵌](#)（inline）和 [多重陈述](#)

[式](#)（multistatement）。一个内嵌数据表数值函数的本身是由单一的 SELECT 陈述式所组成；

多重陈述式数据表数值函数本身是由任意数目的 Transact-SQL 陈述式所组成。

一个内嵌数据表数值函数的语法是 CREATE FUNCTION 陈述式的简短版本，它没有包含

EBEGIN...END 区块以及任何陈述式（除了 RETURN 之外）：

```
CREATE FUNCTION function_name (parameter_list)

RETURNS table

AS

RETURN (select_statement)
```

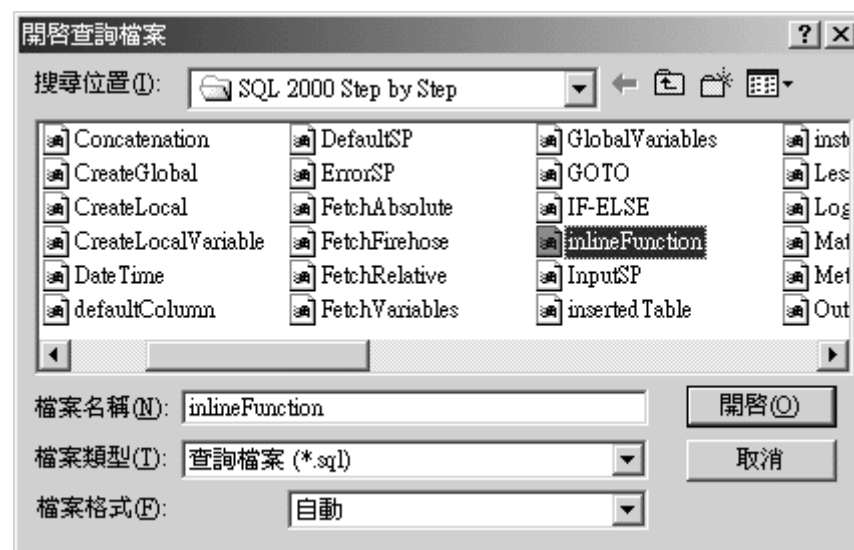
建立内嵌数据表数值函数

1. 选取 `scalarFunction` 指令码的查询窗口。
2. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。



加载 SQL 指令码按钮

此时 Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



3. 选取文件名称为 **inlineFunction** 的指令码档案，然后按一下 [开启](#) 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。

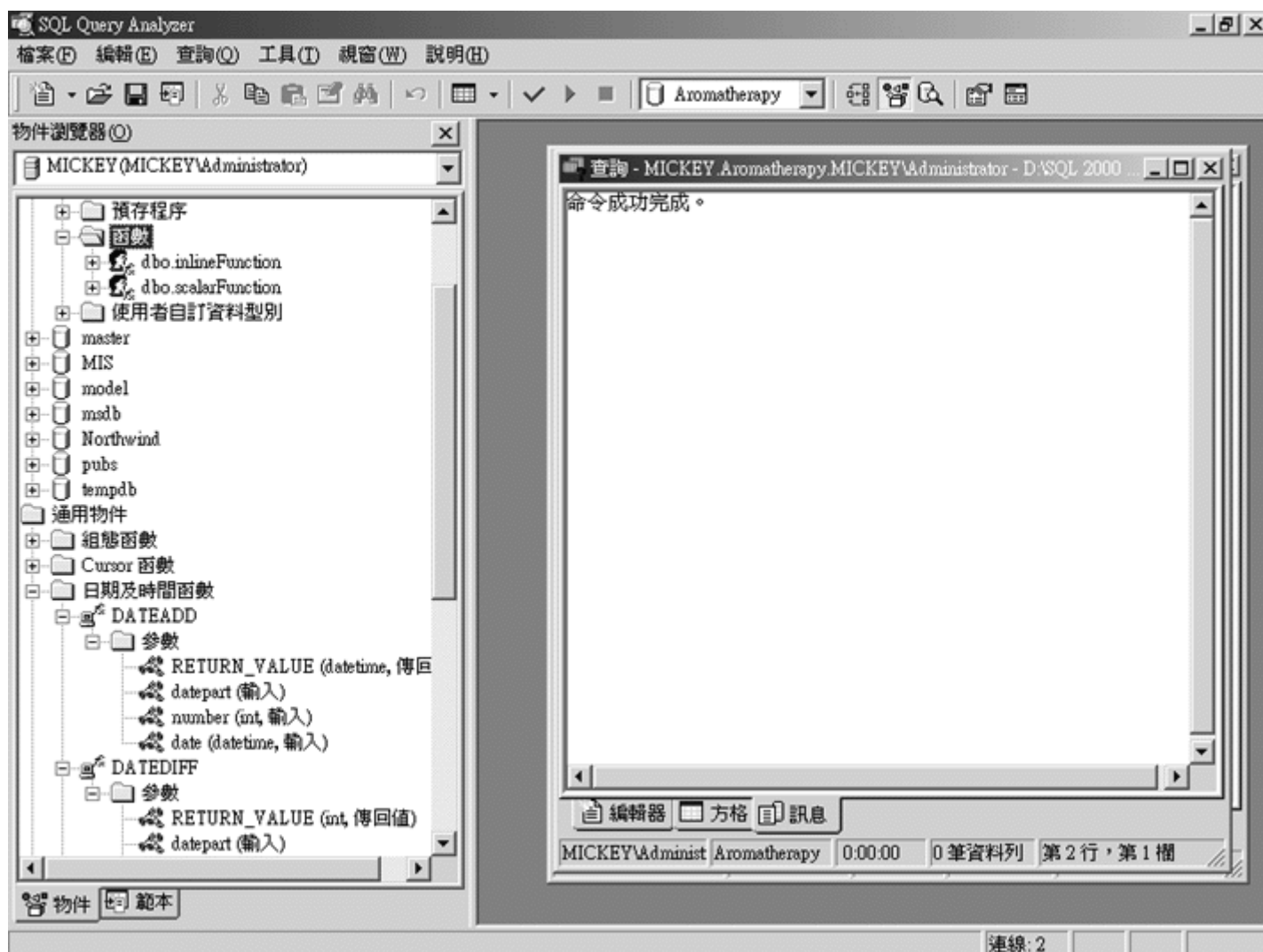


執行查詢按鈕

此时 Query Analyzer 会建立此使用者自订函数。

5. 选取 **对象浏览器** 内的 **函数** 数据夹，并且按一下 **F5** 键以便重新整理显示结果。

此时 **对象浏览器** 内会显示 `dbo.inlineFunction` 的函数名称。



针对多重陈述式数据表数值函数的 **CREATE FUNCTION** 语法来说，它有数值类函数及内嵌函数

组合后的语法：

```
CREATE FUNCTION function_name (parameter_list)

RETURNS @local_table_variable TABLE

    (table_definition)

AS

BEGIN

    tsql_statements

    RETURN

END
```

就像数值类函数一样，一个多重陈述式数据表数值类函数有 **BEGIN...END** 区块包含

Transact-SQL 陈述式，由于此区块中包含多行的 **SELECT** 陈述式，因此您必须在 **RETURNS**

子句中将会传回的数据表定义好。

因为在多重陈述式数据表函数中的 **RETURN** 子句将会传回在 **RETURNS** 子句中定义好的数据

表，因此 **RETURN** 子句执行时不能有参数。举例来说，我们要以 **RETURN** 的方式来执行，而

不是以 **RETURN @myTable** 的方式来执行。

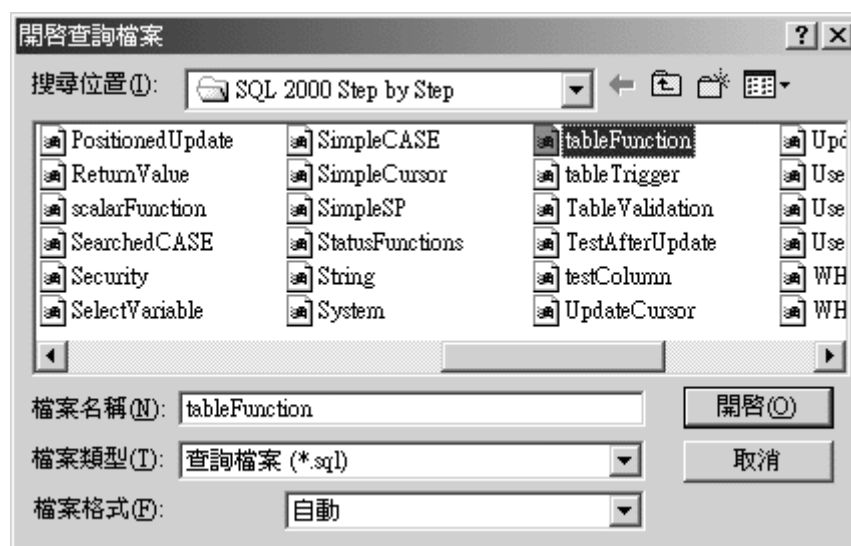
建立多重陈述式数据表数值函数

1. 选取 **inlineFunction** 指令文件的查询窗口。
2. 在 Query Analyzer 工具列上按一下 **加载 SQL 指令码** 按钮。



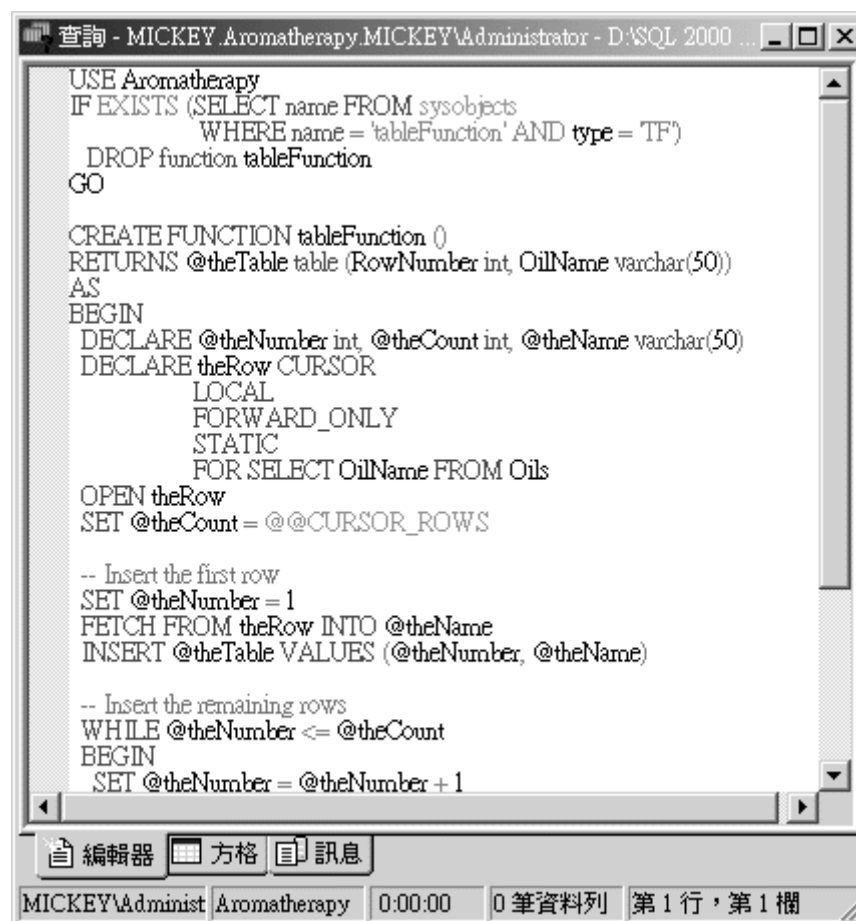
加载 SQL 指令码按钮

此时 Query Analyzer 会显示 **开启查询档案** 的对话框。



3. 选取文件名称为 **tableFunction** 的指令码档案，然后按一下 **开启** 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



4. 在 Query Analyzer 的工具列上按一下 [执行查询](#) 按钮。

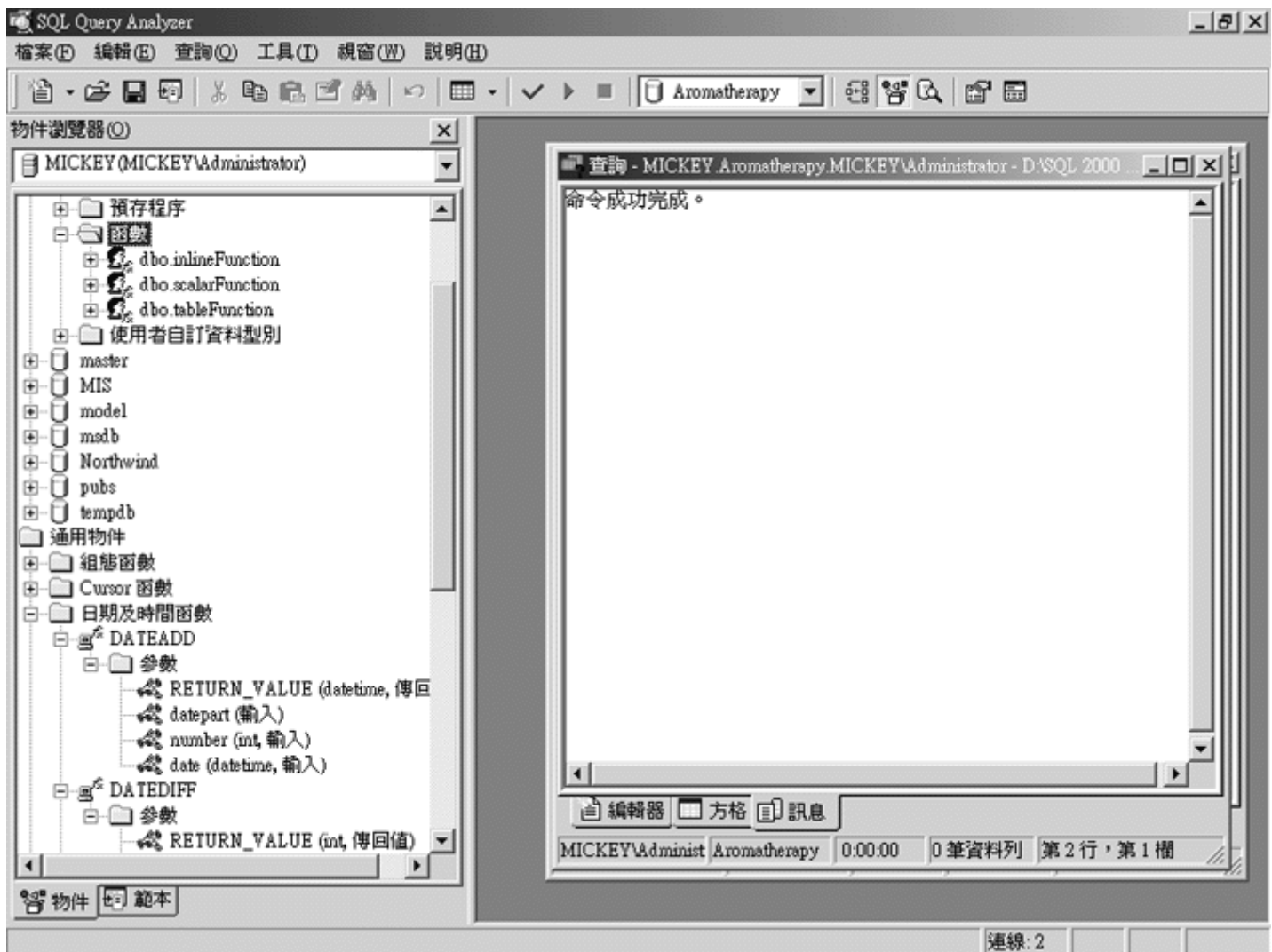


执行查询按钮

此时 Query Analyzer 会建立此使用者自订函数。

5. 选取 **对象浏览器** 内的 **函数** 数据夹，并且按一下 **F5** 键以便重新整理显示结果。

此时 **对象浏览器** 内会显示 `dbo.tableFunction` 的函数名称。



使用使用者自订函数

数值类函数的呼叫语法类似于呼叫 Transact-SQL 内建函数的语法:

```
owner_name.function_name ([parameter_list])
```

数值类函数的 **owner_name** 自变量并不是选择性的。您也不能使用命名的参数语法（举例来说，**@parameter_name = value**）或是忽略该参数。但是您可以使用 **DEFAULT** 关键词来指定一个默认值，就像您呼叫预存程序一样。

提示

SQL Server 提供一些内建的使用者自订函数，这些函数与内建函数是不一样的，它们是以 **fn_** 为起始字符，并且以 **::function_name([parameter_list])** 为呼叫语法。其中 **::** 是取代 **owner_name**，并且表示此函数为内建的使用者自订函数。

您也可以使用 **EXECUTE** 陈述式和数值类函数一起使用：

```
EXECUTE @return_variable = function_name (parameter_list)
```

当您使用 **EXECUTE** 陈述式和使用使用者自订函数一起时，您不需要包括 **owner_name**。使用这种语法时，您可以使用命名的参数选项：

```
EXECUTE @return_variable =function_name @parameter =value  
[, @parameter =value [...]]
```

如果您使用命名的参数时，这些参数就不需要依照该函数中定义的次序，但是您必须要包含所有的参数；您亦不能忽略引用默认值的参数。

数据表数值使用者自订函数—不论是内嵌或多重陈述式函数，都必须要使用与内建函数相同的语法：

```
function_name ([parameter_list])
```

您可以看出在上列语法中并没有使用 **owner_name**，但是您必须要包含所有已经定义的参数，就像您呼叫任何使用者自订函数的方式一样。

在 Transact-SQL 陈述式中使用使用者自订函数

数值类使用者自订函数可以用于任何地方（这些地方泛指其所传回的资料型别可以被使用的地方）。数据表数值使用者自订函数只可以用在 **SELECT** 陈述式的 **FROM** 子句中。

提示

如果该 **SELECT** 陈述式包含在 **DECLARE CURSOR** 陈述式中时，该数据指针必须要设定为 **STATIC** 和 **READ_ONLY**。

在 PRINT 陈述式中使用数值类函数

1. 在 Query Analyzer 工具列上按一下 **新增查询** 按钮，以便开启一个新的查询窗口。



新增查询按钮

此时 Query Analyzer 会开启一个新的查询窗口。

2. 在查询窗口中输入如下所示的陈述式内容：

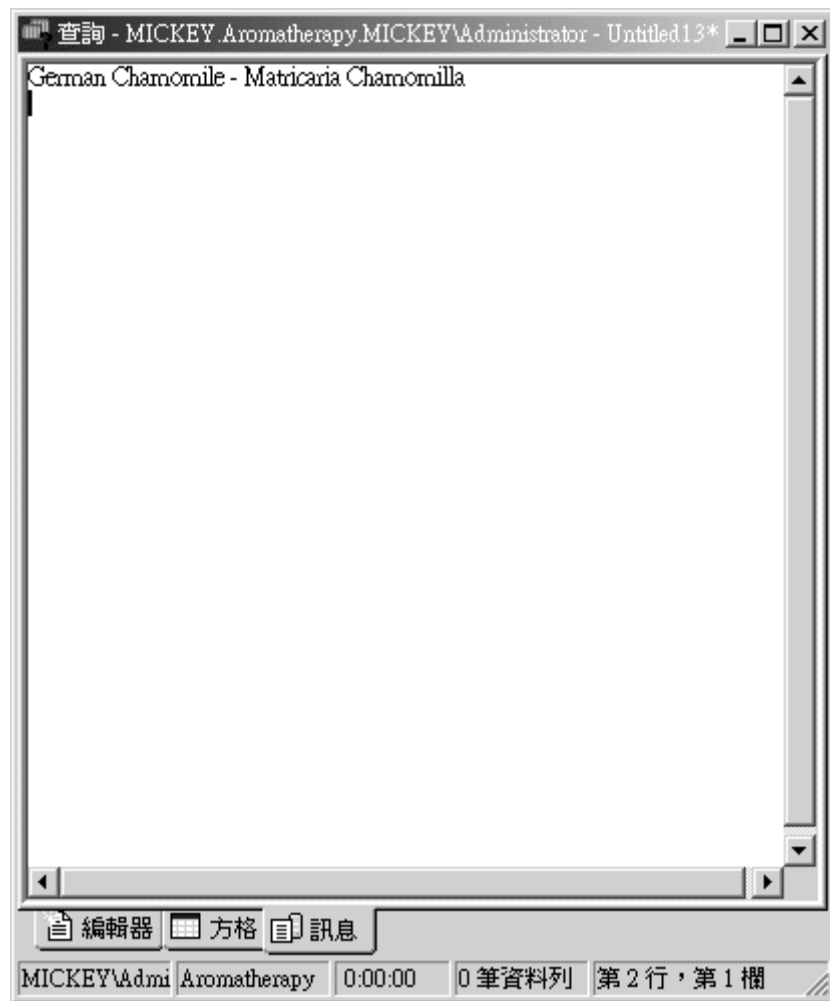
```
PRINT dbo.scalarFunction('German Chamomile')
```

3. 在 Query Analyzer 的工具列上按一下 **执行查询** 按钮。



执行查询按钮

此时 Query Analyzer 会执行此陈述式并且显示其执行结果。



在 **SELECT** 陈述式内使用数值类函数

1. 在 [查询](#) 窗口内选取 [编辑器](#) 标签页，并且在 Query Analyzer 工具列上按一下 [清除窗口](#) 按钮。



清除窗口按钮

此时 Query Analyzer 会清除窗口内容。

2. 在 [查询](#) 窗口中输入如下所示的陈述式内容：

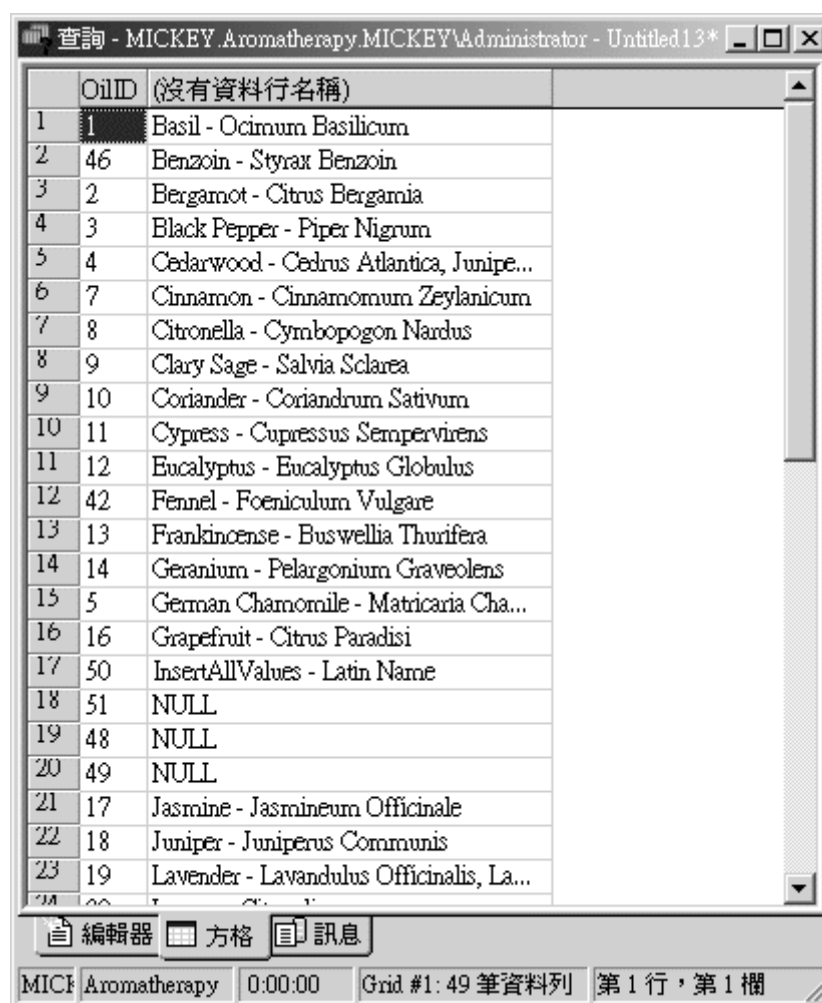
```
3.      SELECT OilID, dbo.scalarFunction(OilName)
FROM Oils
```

4. 在 Query Analyzer 的工具列上按一下 [执行查询](#) 按钮。



执行查询按钮

此时 Query Analyzer 会执行此陈述式并且显示其执行结果。



The screenshot shows a window titled "查詢 - MICKEY.Aromatherapy.MICKEY\Administrator - Untitled13*". The window contains a table with the following data:

	OilID	(沒有資料行名稱)
1	1	Basil - Ocimum Basilicum
2	46	Benzoin - Styrax Benzoin
3	2	Bergamot - Citrus Bergamia
4	3	Black Pepper - Piper Nigrum
5	4	Cedarwood - Cedrus Atlantica, Junipe...
6	7	Cinnamon - Cinnamomum Zeylanicum
7	8	Citronella - Cymbopogon Nardus
8	9	Clary Sage - Salvia Sclarea
9	10	Coriander - Coriandrum Sativum
10	11	Cypress - Cupressus Sempervirens
11	12	Eucalyptus - Eucalyptus Globulus
12	42	Fennel - Foeniculum Vulgare
13	13	Frankincense - Boswellia Thurifera
14	14	Geranium - Pelargonium Graveolens
15	5	German Chamomile - Matricaria Cha...
16	16	Grapefruit - Citrus Paradisi
17	50	InsertAllValues - Latin Name
18	51	NULL
19	48	NULL
20	49	NULL
21	17	Jasmine - Jasmineum Officinale
22	18	Juniper - Juniperus Communis
23	19	Lavender - Lavandulus Officinalis, La...

At the bottom of the window, there are buttons for "編輯器" (Editor), "方格" (Grid), and "訊息" (Messages). Below these buttons, a status bar shows "MICKEY.Aromatherapy", "0:00:00", "Grid #1: 49 筆資料列", and "第 1 行, 第 1 欄".

在 SELECT 陈述式内使用数据表数值函数

1. 在 查詢 窗口内选取 編輯器 标签页, 并且在 Query Analyzer 工具列上按一下 清除窗口 按钮。



清除窗口按钮

此时 Query Analyzer 会清除窗口内容。

2. 在查询窗口中输入如下所示的陈述式内容：

```
SELECT * FROM tableFunction()
```

3. 在 Query Analyzer 的工具列上按一下 [执行查询](#) 按钮。



执行查询按钮

此时 Query Analyzer 会执行此陈述式并且显示其执行结果。

查詢 - MICKEY.Aromatherapy.MICKEY\Administrator - Untitled13*

	RowNumber	OilName	
1	1	Basil	
2	2	Benzoin	
3	3	Bergamot	
4	4	Black Pepper	
5	5	Cedarwood	
6	6	Cinnamon	
7	7	Citronella	
8	8	Clary Sage	
9	9	Coriander	
10	10	Cypress	
11	11	Eucalyptus	
12	12	Fennel	
13	13	Frankincense	
14	14	Geranium	
15	15	German Chamomile	
16	16	Grapefruit	
17	17	InsertAllValues	
18	18	InsertDefault	
19	19	InsertFromGrid	
20	20	InsertFromSQL	
21	21	Jasmine	
22	22	Juniper	
23	23	Lavender	
24	24		

編輯器 方格 訊息

MICKEY.Aromatherapy 0:00:00 Grid #1: 50 筆資料列 第 1 行, 第 1 欄

4. 关闭此查询窗口，假如有出现一个提示对话框要求您是否要储存更改的查询时，

请您选择 **否**。

在数据表定义中使用使用者自订函数

使用者自订函数可以用于数据表定义中，数据表的拥有者也是使用者自订函数的拥有者，但是用于函数中的参数在使用上有某些限制。

如果使用者自订函数用于计算数据行时，该使用者自订函数的参数就必须是数据表中其它的数据行或是常数。如果使用者自订函数是当作一个 **CHECK** 条件约束使用时，也适用相同的规则。如果使用者自订函数被当作数据行的默认值在使用，此参数就必须是常数。

使用使用者自订函数为计算数据行



加载 SQL 指令码按钮

1. 选取 **tableFunction** 指令码的查询窗口。
2. 在 Query Analyzer 工具列上按一下 **加载 SQL 指令码** 按钮。

此时 Query Analyzer 会显示一个 **开启查询档案** 的对话框。



3. 选取文件名称为 **computedColumn** 的指令码档案，然后按一下 **开启** 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



4. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



執行查詢按鈕

此時 Query Analyzer 會建立此函數及數據表。

5. 在 Query Analyzer 工具列上按一下 [新增查询](#) 按钮，以便开启一个新的查询窗口。



新增查询按钮

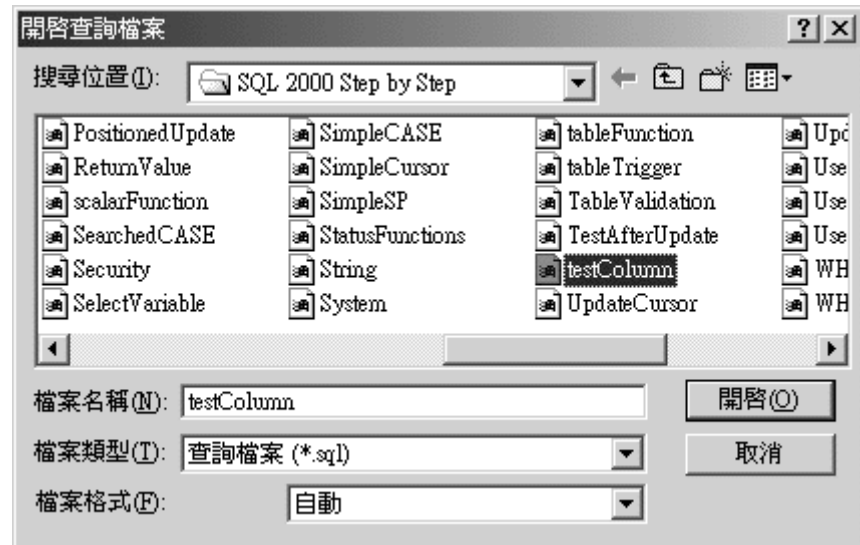
此时 Query Analyzer 会开启一个新的查询窗口。

6. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。



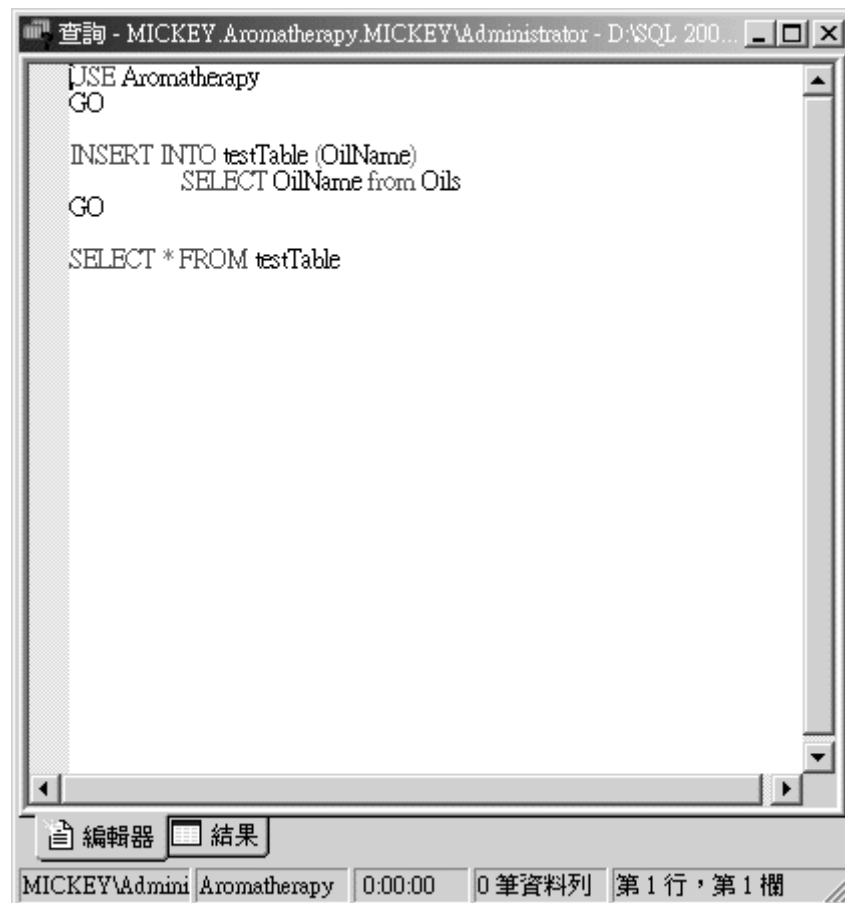
加载 SQL 指令码按钮

此时 Query Analyzer 会显示 [开启查询档案](#) 的对话框。



7. 选取文件名称为 **testColumn** 的指令码档案，然后按一下 **开啓** 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



8. 在 Query Analyzer 的工具列上按一下 [執行查詢](#) 按鈕。



执行查询按钮

此时 Query Analyzer 会执行此陈述式并且显示其执行结果。

查詢 - MICKEY.Aromatherapy.MICKEY\Administrator - D:\SQL 200...

	OilID	OilName	NewOilsID
1	1	Basil	2
2	2	Benzoin	3
3	3	Bergamot	4
4	4	Black Pepper	5
5	5	Cedarwood	6
6	6	Cinnamon	7
7	7	Citronella	8
8	8	Clary Sage	9
9	9	Coriander	10
10	10	Cypress	11
11	11	Eucalyptus	12
12	12	Fennel	13
13	13	Frankincense	14
14	14	Geranium	15
15	15	German Chamomile	16
16	16	Grapefruit	17
17	17	InsertAllValues	18
18	18	InsertDefault	19
19	19	InsertFromGrid	20
20	20	InsertFromSQL	21
21	21	Jasmine	22

編輯器 方格 訊息

MICKEY Aromatherapy 0:00:00 Grid #1: 49 筆資料列 第 1 行, 第 1 欄

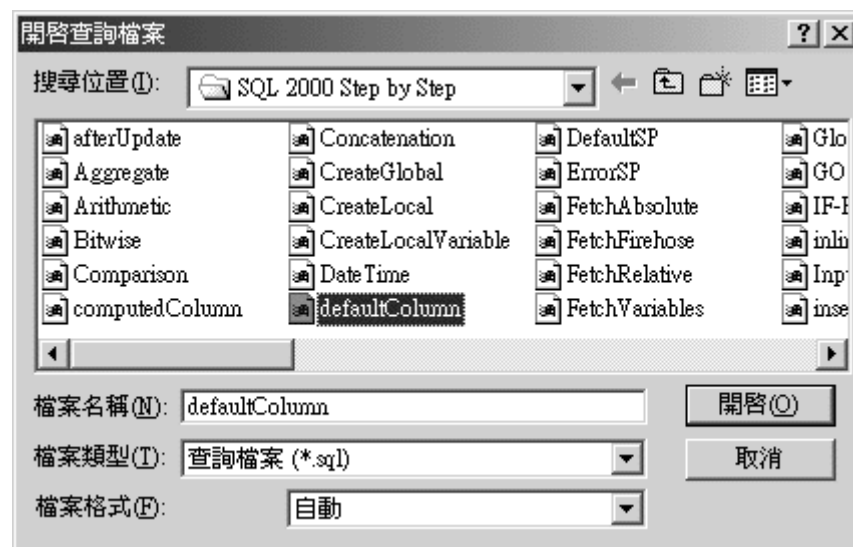
在 DEFAULT 定义中使用使用者自订函数

1. 选取 computedColumn 指令码的查询窗口。
2. 在 Query Analyzer 工具列上按一下 [加载 SQL 指令码](#) 按钮。



加载 SQL 指令码按钮

此时 Query Analyzer 会显示一个 [开启查询档案](#) 的对话框。



3. 选取文件名称为 [defaultColumn](#) 的指令码档案，然后按一下 [开启](#) 按钮。

此 Query Analyzer 会将指令码档案加载到查询窗口中。



4. 在 Query Analyzer 的工具列上按一下 [执行查询](#) 按钮。



执行查询按钮

此时 Query Analyzer 会建立此数据表。

5. 选取 testColumn 函数的查询窗口。



执行查询按钮

6. 在 Query Analyzer 的工具列上按一下 [执行查询](#) 按钮。

此时 Query Analyzer 会执行此陈述式并且显示其执行结果。

查詢 - MICKEY.Aromatherapy.MICKEY\Administrator - D:\SQL 2000 Ste...				
	OilID	OilName	NewOilsID	
1	1	Basil	6	
2	2	Benzoin	6	
3	3	Bergamot	6	
4	4	Black Pepper	6	
5	5	Cedarwood	6	
6	6	Cinnamon	6	
7	7	Citronella	6	
8	8	Clary Sage	6	
9	9	Coriander	6	
10	10	Cypress	6	
11	11	Eucalyptus	6	
12	12	Fennel	6	
13	13	Frankincense	6	
14	14	Geranium	6	
15	15	German Chamomile	6	
16	16	Grapefruit	6	
17	17	InsertAllValues	6	
18	18	InsertDefault	6	
19	19	InsertFromGrid	6	
20	20	InsertFromSQL	6	
21	21	Jasmine	6	
22	22	Juniper	6	

編輯器

方格

訊息

MICKEY\Aromatherapy

0:00:00

Grid #1: 49 筆資料列

第 1 行，第 1 欄

本章总结

要执行的工作	SQL 语法
建立数值类函数	<pre> CREATE FUNCTION function_name ([parameter_list]) RETURNS data_type AS BEGIN </pre>

	<pre> [tsql_statements] RETURN (return_value) END </pre>
建立内嵌数据表数值函数	<pre> CREATE FUNCTION function_name (parameter_list) RETURNS table AS RETURN (select_statement) </pre>
建立多重陈述式数据表数值函数	<pre> CREATE FUNCTION function_name (parameter_list) RETURNS @local_table_variable TABLE (table_definition) AS BEGIN tsql_statements RETURN END </pre>
使用数值类函数	<pre> owner_name.function_name([parameter_list]) </pre>
在 EXECUTE 陈述式内使用函数	<pre> EXECUTE @return_variable = </pre>

	<div>function_name(parameter_list)</div> <div>或</div> <div>EXECUTE @return_variable = function_name</div> <div>@parameter = value [, @parameter = alue [...]]</div>
<div>将函数用于计算的数据行</div>	<div>column_name AS function</div> <div>该参数必须为其它的数据行或常数</div>
<div>将函数用于 DEFAULT 数据行</div>	<div>Column_name data_type = function</div> <div>该参数必须是为常数</div>

