第1章　SQL Server 2005 关系数据库简介

1.1 关系数据库

1.1.1 关系模型的基本概念

1.1.2 关系模型的性质

1.1.3 关系数据库完整性

1.1.4 关系数据库的规范化

1.1.1 关系模型的基本概念

- 数据库技术是作为数据处理的一门技术而发展起来的，所研究的问题就是如何科学地组织和存储数据，如何高效地获取和处理数据。
- 在数据库中用数据模型来抽象、表示和处理现实世界中的数据。数据库即是模拟现实世界中某应用环境（一个企业、单位或部门）所涉及的数据的集合，它不仅要反映数据本身的内容，而且要反映数据之间的联系。
- 最常用的数据模型有层次模型（Hierarchical Model）、网状模型（Network Model）和关系模型（Relational Model）。
- 关系模型就是用二维表格结构来表示实体及实体之间联系的模型。

1.1.1 关系模型的基本概念

- 关系（Relation）：一个关系对应一张二维表，每个关系有一个关系名。在SQL Server中，一个关系就是一个表文件。
- 元组（Tuple）：二维表中水平方向的一行，有时也叫做一条记录。
- 属性（Attribute）：表格中的一列，相当于记录中的一个字段。
- 关键字（Key）：可唯一标识元组的属性或属性集，也称为关系键或主码。
- 域（Domain）：属性的取值范围，如性别的域是（男，女）。
- 分量：每一行对应的列的属性值，即元组中的一个属性值。
- 关系模式：对关系的描述，一般表示为：关系名（属性1，属性2，……属性n）。

1.1.2 关系模型的性质

（1）关系中不允许出现相同的元组。因为数学上集合中没有相同的元素，而关系是元组的集合，所以作为集合元素的元组应该是唯一的。

（2）关系中元组的顺序（即行序）是无关紧要的，在一个关系中可以任意交换两行的次序。因为集合中的元素是无序的，所以作为集合元素的元组也是无序的。根据关系的这个性质，可以改变元组的顺序使其具有某种排序，然后按照顺序查询数据，可以提高查询速度。

（3）关系中属性的顺序是无关紧要的，即列的顺序可以任意交换。交换时，应连同属性名一起交换，否则将得到不同的关系。

（4）同一属性名下的各个属性值必须来自同一个域，是同一类型的数据。

（5）关系中各个属性必须有不同的名字，不同的属性可来自同一个域，即它们的分量可以取自同一个域。

（6）关系中每一分量必须是不可分的数据项，或者说所有属性值都是原子的，是一个确定的值，而不是值的集合。

1.1.3 关系数据库完整性

1、实体完整性(Entity Integrity)

- 实体完整性是指主关系键的值不能为空或部分为空。

2、参照完整性(Referential integrity)

- 如果关系R2的外部关系键X与关系R1的主关系键相符,则X的每个值或者等于R1中主关系键的某一个值，或者取空值。

3、域完整性

- 域完整性是针对某一具体关系数据库的约束条件。它反映某一具体应用所涉及的数据必须满足的语义要求。

1.1.4　关系数据库的规范化

- 第一范式（1NF）：元组中每一个分量都必须是不可分割的数据项
- 第二范式（2NF）：不仅满足第一范式，而且所有非主属性完全依赖于其主码
- 第三范式（3NF）：不仅满足第二范式，而且它的任可一个非主属性都不传递于任何主关键字

1.2 SQL Server 2005的新特点

1、增强的通知服务。
2、增强的报表服务。
3、新增Service Broker技术。
4、增强的数据引擎。
5、增强的数据访问接口。
6、增强的分析服务。
7、增强的集成服务。
8、增强的数据复制服务。
9、改进的开发工具。

1.3 SQL Server2005的安装和配置

1.3.1 SQL Server2005环境需求

1.3.2 SQL Server2005的安装

1.3.3 SQL Server升级

1.3.4 SQL Server2005系统数据库简介

1.3.1 SQL Server2005环境需求

1. 硬件需求

（1）显示器：VGA或者分辨率至少在1,024x768像素之上的显示器。

（2）点触式设备：鼠标或者兼容的点触式设备。

（3）CD 或者 DVD驱动器。

1.3.1 SQL Server2005环境需求

1. 硬件需求

（4）处理器型号，速度及内存需求。SQL Server 2005不同的版本其对处理器型号，速度及内存的需求是不同的， 如下表

SQL Server 2005版本	处理器型号	处理器速度	内存(RAM)
SQL Server 2005企业版（Enterprise Edition） SQL Server 2005开发者版 （Developer Edition） SQL Server 2005标准版（Standard Edition） SQL Server 2005工作组版（Workgroup Edition）	Pentium III及其兼容处理器，或者更高型号。	至少600 MHz，推荐1GHz或更高。	至少512MB，推荐1GB或更大。
SQL Server 2005简化版（Express Edition）	Pentium III及其兼容处理器，或者更高型号。	至少600 MHz，推荐1GHz或更高。	至少192 MB，推荐512MB或更大

1.3.1 SQL Server2005环境需求

1. 硬件需求

（5）硬盘空间需求。实际的硬件需求取决于你的系统配置以及你所选择安装的SQL Server 2005服务和组件。如下表

服务和组件	硬盘需求
数据库引擎及数据文件，复制，全文搜索等	150 MB
分析服务及数据文件	35 KB
报表服务和报表管理器	40 MB
通知服务引擎组件，客户端组件以及规则组件	5 MB
集成服务	9 MB
客户端组件	12 MB
管理工具	70 MB
开发工具	20 MB
SQL Server联机图书以及移动联机图书	15 MB
范例以及范例数据库	390 MB

1.3.1 SQL Server2005环境需求

2. 软件需求

（1）浏览器软件。在装SQL Server 2005之前，需安装Microsoft Internet Explorer 6.0 SP1或者其升级版本。因为微软控制台以及HTML帮助都需要此软件。

（2）IIS软件。在装SQL Server 2005之前，需安装IIS5.0及其后续版本，以支持SQL Server 2005的报表服务。

（3）ASP.NET 2.0。当安装报表服务时，SQL Server 2005安装程序会检查ASP.NET是否已安装到本机上。

（4）还需要安装以下软件：Microsoft Windows .NET Framework 2.0；Microsoft SQL Server Native Client；Microsoft SQL Server Setup support files。

1.3.1 SQL Server2005环境需求

2. 软件需求

（5）下表列出常见的操作系统是否支持运行SQLServer 2005的各种不同版本。

	企业版	开发版	标准版	工作组版	简化版
Windows 2000	不支持	不支持	不支持	不支持	不支持
Windows 2000 Professional Edition SP4	不支持	支持	支持	支持	支持
Windows 2000 Server SP4	支持	支持	支持	支持	支持
Windows 2000 Advanced Server SP4	支持	支持	支持	支持	支持
Windows 2000 Datacenter Edition SP4	支持	支持	支持	支持	支持
Windows XP Home Edition SP2	不支持	支持	不支持	不支持	支持
Windows XP Professional Edition SP2	不支持	支持	支持	支持	支持
Windows 2003 Server SP1	支持	支持	支持	支持	支持
Windows 2003 Enterprise Edition SP1	支持	支持	支持	支持	支持

1.3.2 SQL Server2005的安装

- SQL Server 2005的安装过程与其它Microsoft Windows系列产品类似。 用户可根据向导提示，选择需要的选项一步一步地完成。

1.3.3 SQL Server 升级

- 可以将SQL Server 2000 Service Pack 3 (SP3)或更高版本的实例以及 SQL Server 7.0 SP4 或更高版本的实例直接升级到 SQL Server 2005。通过安装程序可以完成大多数升级操作；但是，某些组件支持或要求需要在在运行安装程序后迁移应用程序或解决方案。
- 在运行安装程序以升级到 SQL Server 2005 之前，应该首先检查系统要求和升级要求。检查系统要求和升级要求之后，运行 SQL Server 升级顾问以分析 SQL Server 2000 和 SQL Server 7.0 的实例。升级顾问针对您的安装生成问题列表，必须在升级之前或之后解决这些问题。SQL Server 安装程序将检测阻止升级到 SQL Server 2005 遇到的问题，但不会列出可能影响应用程序的问题。

1.3.4 SQL Server2005系统数据库简介

SQL Server 2005有4个系统数据库，它们分别为Master、Model、Msdb、Tempdb。

（1）Master数据库是SQL Server系统最重要的数据库，它记录了SQL Server系统的所有系统信息。这些系统信息包括所有的登录信息、系统设置信息、SQL Server的初始化信息和其他系统数据库及用户数据库的相关信息。因此，如果 master 数据库不可用，则 SQL Server 无法启动。在 SQL Server 2005 中，系统对象不再存储在 master 数据库中，而是存储在 Resource 数据库中。

（2）model 数据库用作在 SQL Server 实例上创建的所有数据库的模板。因为每次启动 SQL Server 时都会创建 tempdb，所以 model 数据库必须始终存在于 SQL Server 系统中。当发出 CREATE DATABASE（创建数据库）语句时，将通过复制 model 数据库中的内容来创建数据库的第一部分，然后用空页填充新数据库的剩余部分。 如果修改 model 数据库，之后创建的所有数据库都将继承这些修改。例如，可以设置权限或数据库选项或者添加对象，例如，表、函数或存储过程。

1.3.4 SQL Server2005系统数据库简介

（3）Msdb数据库是代理服务数据库，为其报警、任务调度和记录操作员的操作提供存储空间。

（4）Tempdb是一个临时数据库，它为所有的临时表、临时存储过程及其他临时操作提供存储空间。Tempdb数据库由整个系统的所有数据库使用，不管用户使用哪个数据库，他们所建立的所有临时表和存储过程都存储在tempdb上。SQL Server每次启动时，tempdb数据库被重新建立。当用户与SQL Server断开连接时，其临时表和存储过程自动被删除。

1.4 SQL Server2005 工具和实用程序

1.4.1 SQL Server 2005管理平台

1.4.2 商业智能开发平台（Business Intelligence Development Studio）

1.4.3 SQL Server 分析器

1.4.4 数据库引擎优化顾问

1.4.5 Analysis Services

1.4.6 SQL Server配置管理器

1.4.7 SQL Server文档和教程

1.4.1 SQL Server 2005管理平台

- SQL Server 2005管理平台（SQL Server Management Studio）包含了SQL Server 2000企业管理器（Enterprise Manager），以及查询分析器（Query Analyzer）等方面的功能。此外，SQL Server 2005管理平台还提供了一种环境，用于管理 Analysis Services（分析服务）、Integration Services（集成服务）、Reporting Services（报表服务）和 XQuery。

1.4.1 SQL Server 2005管理平台

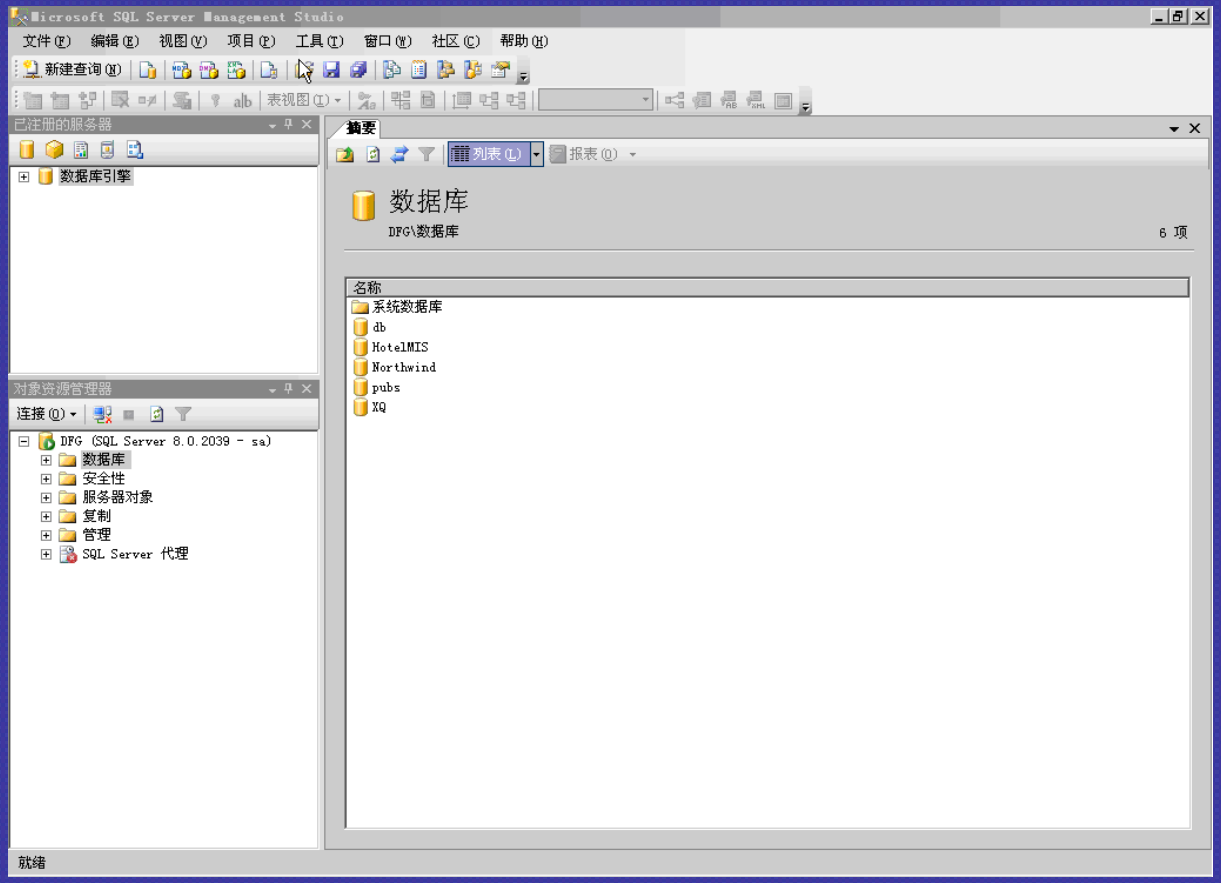


图 SQL Server 管理平台界面

1.4.1 SQL Server 2005管理平台

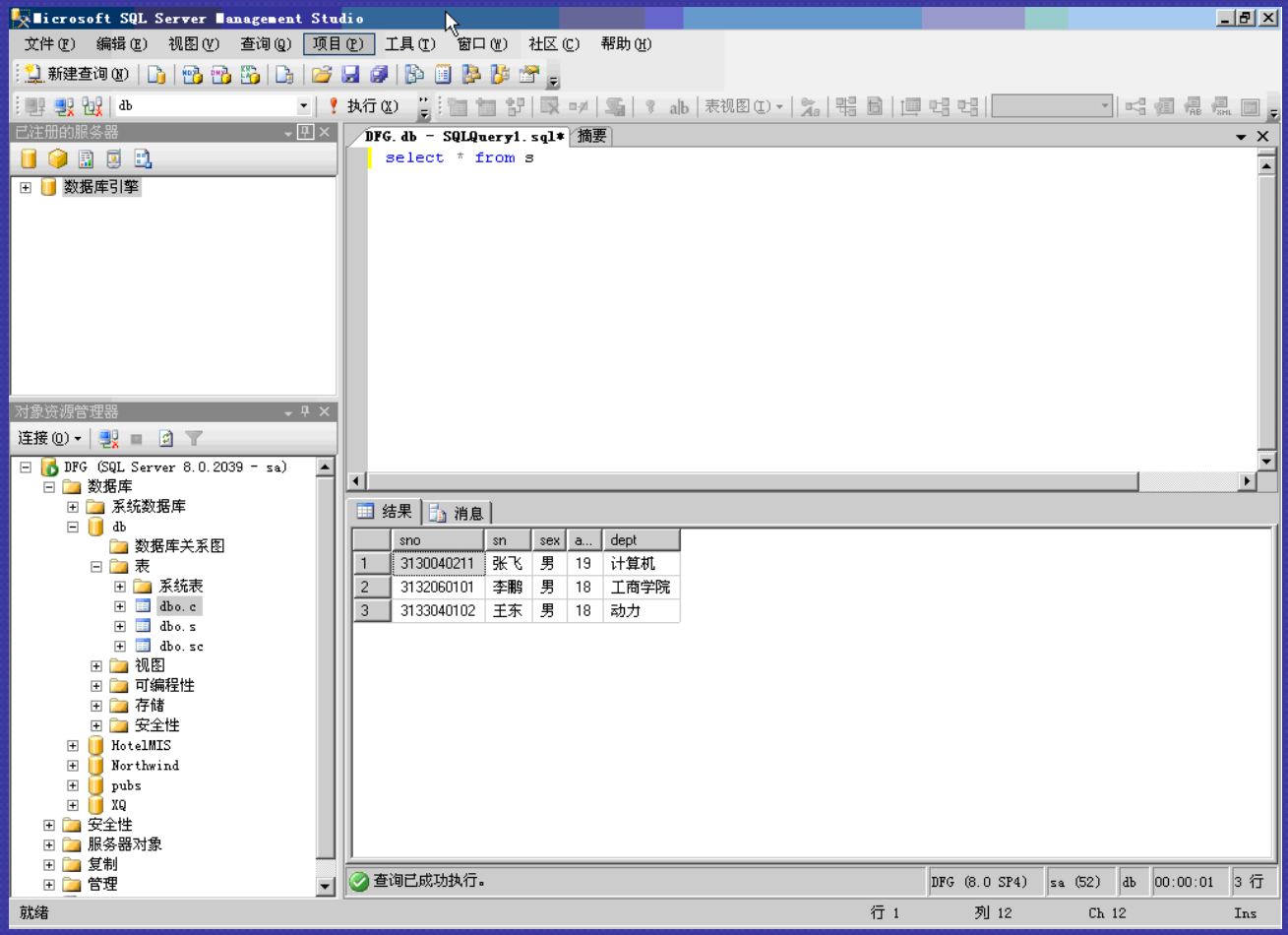
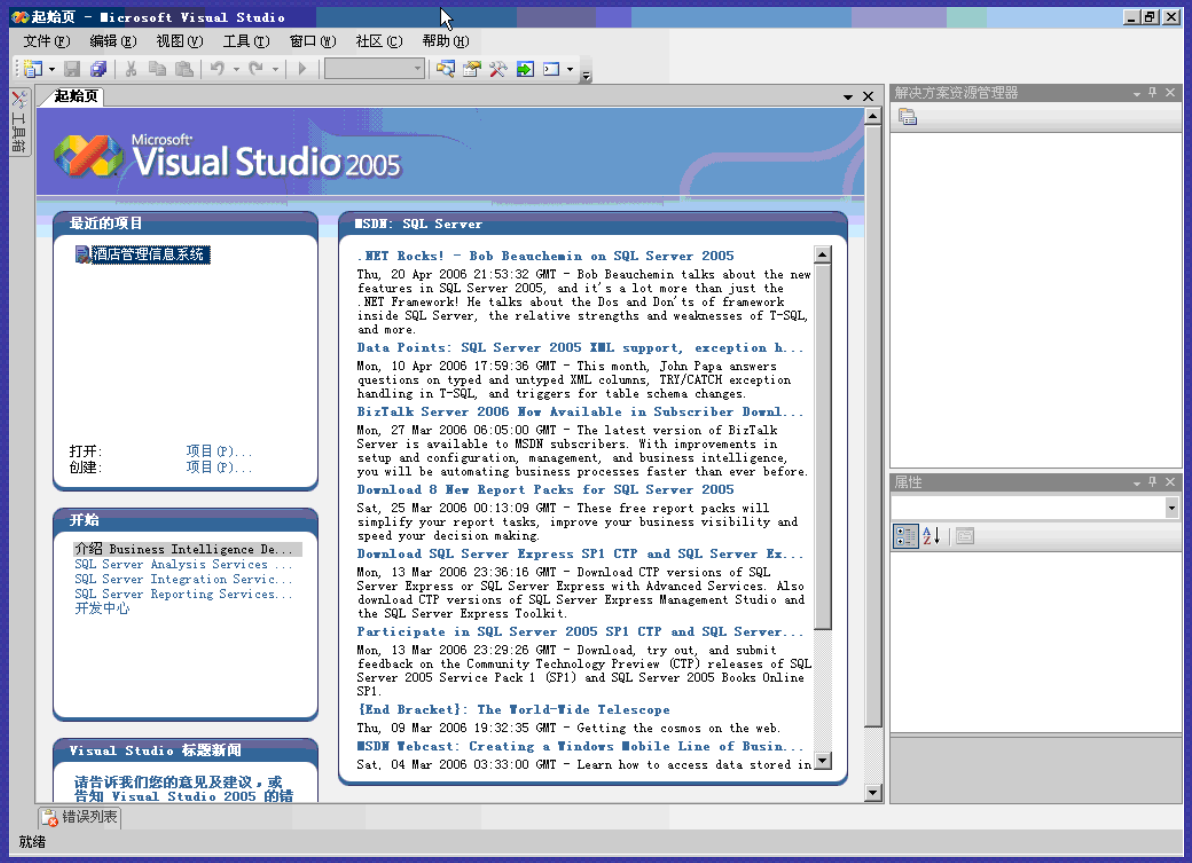


图 SQL Server2005查询分析器界面

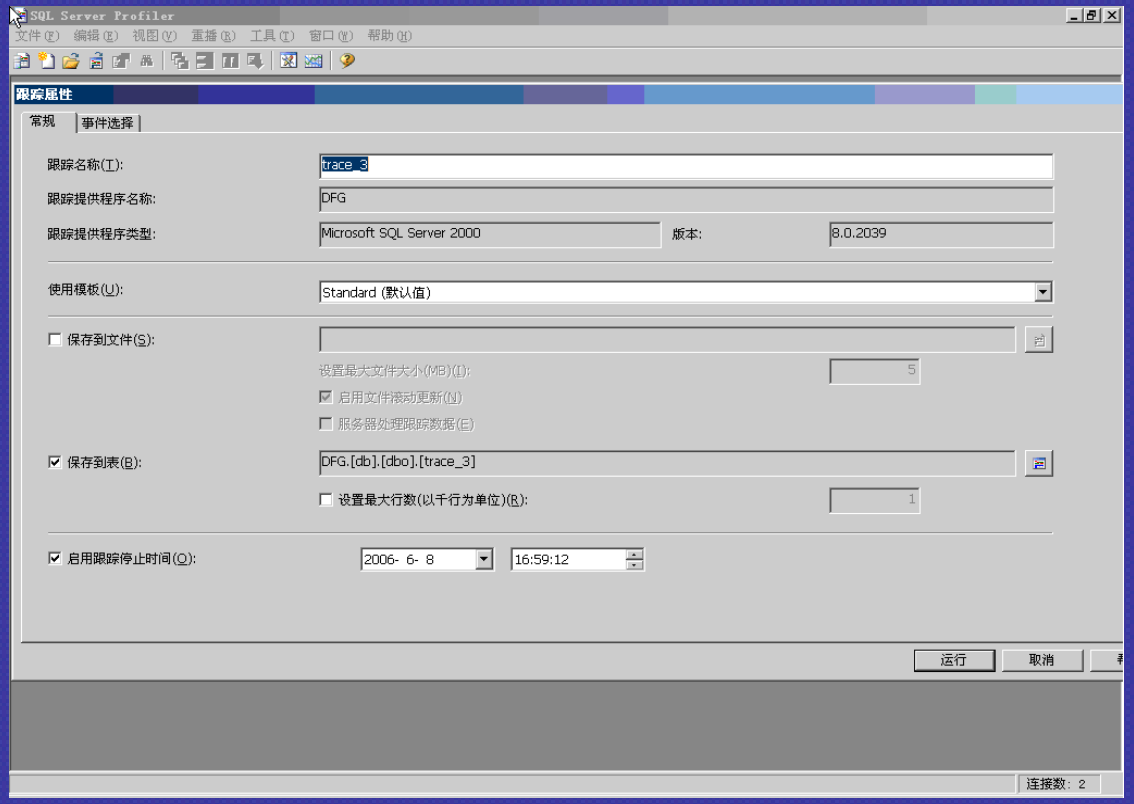
1.4.2 商业智能开发平台

- SQL Server商业智能开发平台(Business Intelligence Development Studio)是一个集成的环境，用于开发商业智能构造（如多维数据集、数据源、报告和Integration Services 软件包），如下图所示。



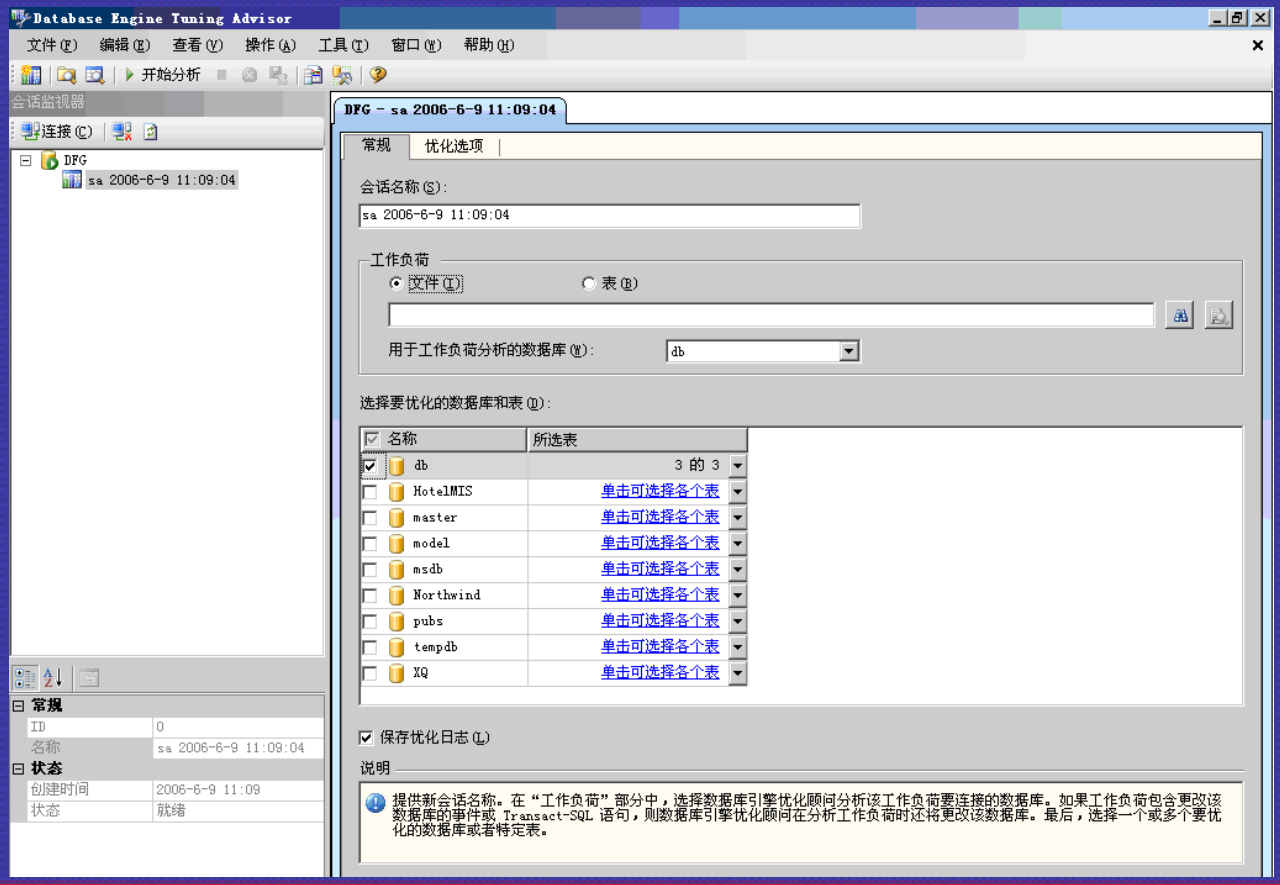
1.4.3 SQL Server分析器

- SQL Server分析器（Profiler）是一个图形化的管理工具，用于监督、记录和检查SQL Server 数据库的使用情况。对系统管理员来说，它是一个连续实时地捕获用户活动情况的间谍。可以通过多种方法启动 SQL Server Profiler，以支持在各种情况下收集跟踪输出。如下图所示。



1.4.4 数据库引擎优化顾问

- 企业数据库系统的性能依赖于组成这些系统的数据库中物理设计结构的有效配置。这些物理设计结构包括索引、聚集索引、索引视图和分区，其目的在于提高数据库的性能和可管理性。SQL Server 2005 提供了数据库引擎优化顾问，这是分析一个或多个数据库上工作负荷的性能效果的工具。如下图所示。

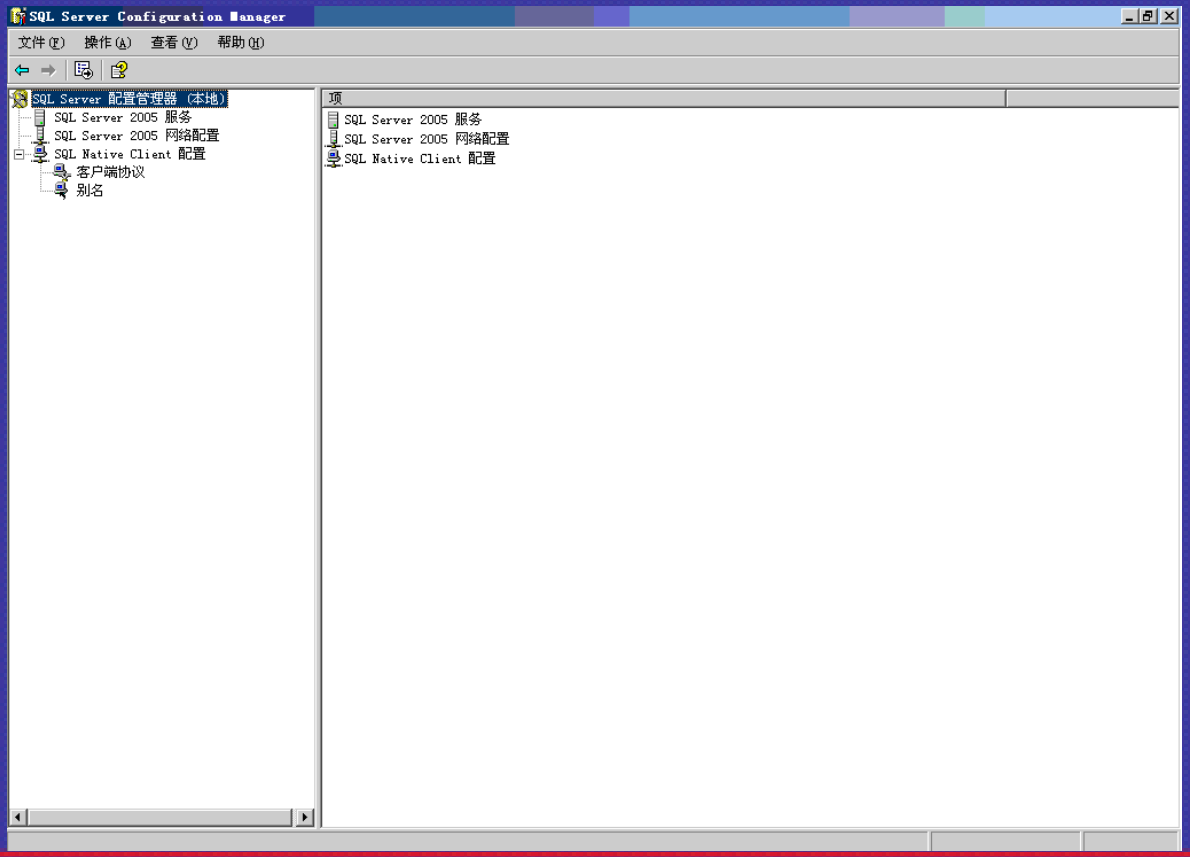


1.4.5 分析服务

- Microsoft SQL Server 2005 Analysis Services（SSAS）为商业智能应用程序提供联机分析处理（OLAP）和数据挖掘功能。Analysis Services 允许设计、创建和管理包含从其他数据源（如关系数据库）聚合的数据的多维结构，以实现对 OLAP 的支持。对于数据挖掘应用程序，分析服务允许设计、创建和可视化处理那些通过使用各种行业标准数据挖掘算法，并根据其他数据源构造出来的数据挖掘模型。

1.4.6 SQL Server配置管理器

- SQL Server 配置管理器是一种工具，用于管理与 SQL Server 相关联的服务、配置 SQL Server 使用的网络协议以及从 SQL Server 客户端计算机管理网络连接配置。如下图所示。



1.4.7 SQL Server文档和教程

- SQL Server 2005提供了大量的联机帮助文档（Books Online），它具有索引和全文搜索能力，可根据关键词来快速查找用户所需信息。SQL Server 2005 中提供的教程可以帮助了解 SQL Server 技术和开始项目，如下图所示。

第2章　服务器管理

2.1 服务器注册

2.1.1 创建服务器组

2.1.2 服务器注册与连接

2.1.1 创建服务器组

- 在一个网络系统中，可能有多个SQL Server服务器，可以对这些SQL Server服务器进行分组管理。分组的原则往往是依据组织结构原则，如将公司内一个部门的几个SQL Server服务器分为一组。SQL Server分组管理由SQL Server管理平台来进行。

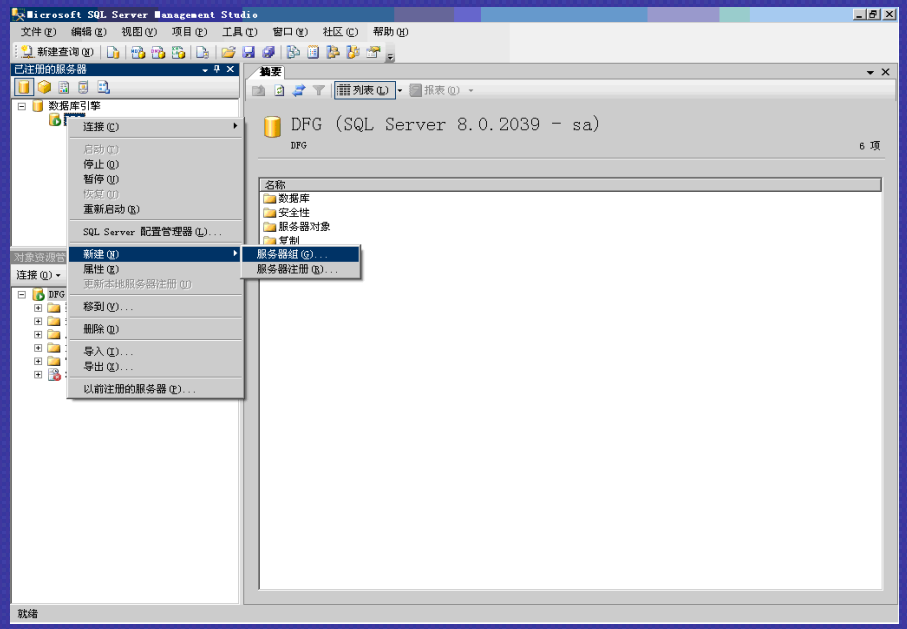


图2-1 打开新建服务器组对话框

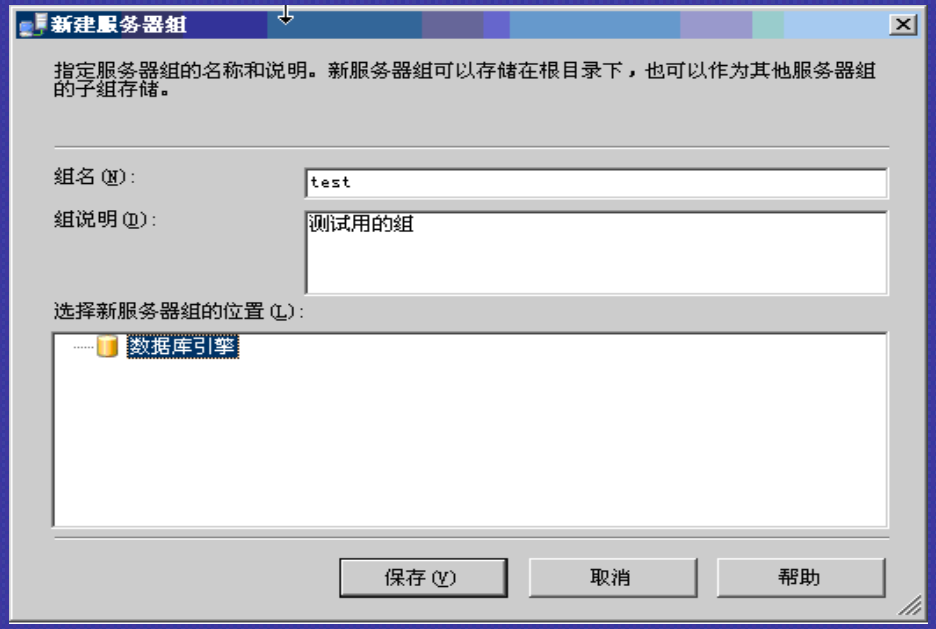


图2-2 新建服务器组窗口

2.1.2 服务器注册与连接

- 在**SQL Server**管理平台中注册服务器可以存储服务器连接信息，以供将来连接时使用。
- 有三种方法可以在**SQL Server**管理平台中注册服务器：

（1）在安装管理平台之后首次启动它时，将自动注册 **SQL Server** 的本地实例；

（2）可以随时启动自动注册过程来还原本地服务器实例的注册；

（3）可以使用 **SQL Server**管理平台的“已注册的服务器”工具注册服务器。

2.1.2 服务器注册与连接

在注册服务器时必须指定以下选项，如图2-3所示：

（1）服务器的类型。

（2）服务器的名称。

（3）登录到服务器时使用的身份验证的类型，以及登录名和密码（如果需要）。

（4）注册了服务器后要将该服务器加入到其中的组的名称。

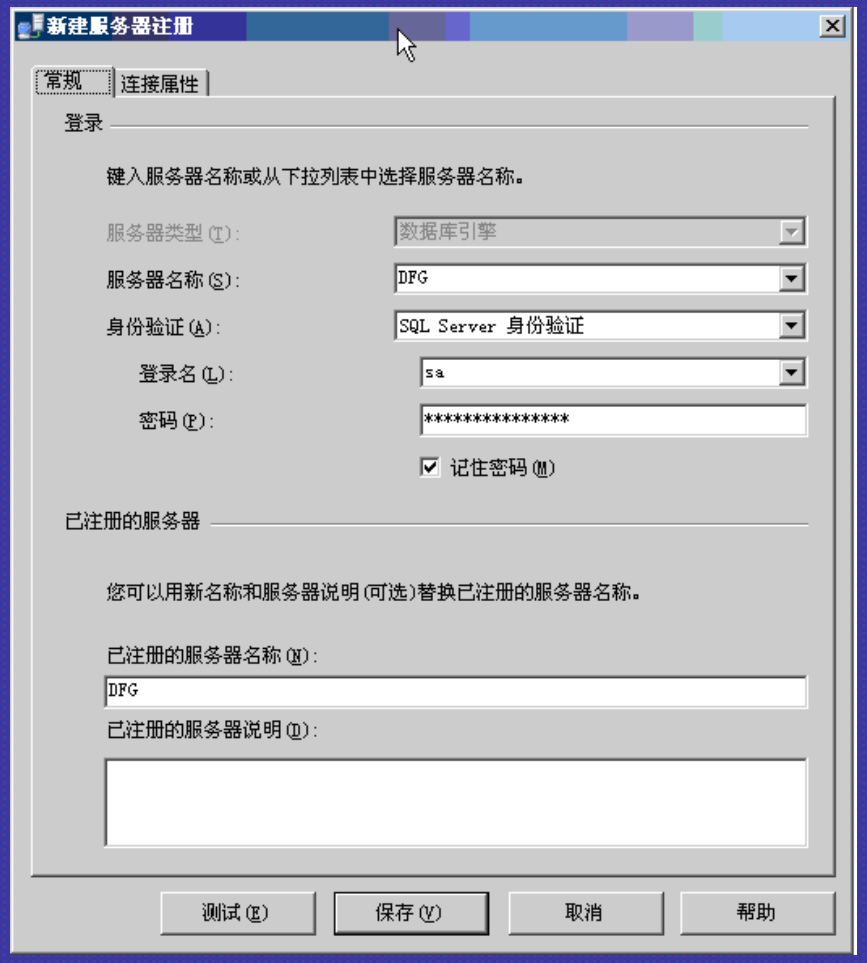


图2-3 新建服务器注册窗口

2.1.2 服务器注册与连接

选择服务器连接属性页框，还可以指定下列连接选项，如图2-4所示：

（1）服务器默认情况下连接到的数据库。

（2）连接到服务器时所使用的网络协议，要使用的网络数据包大小。

（3）连接超时值，执行超时值等。

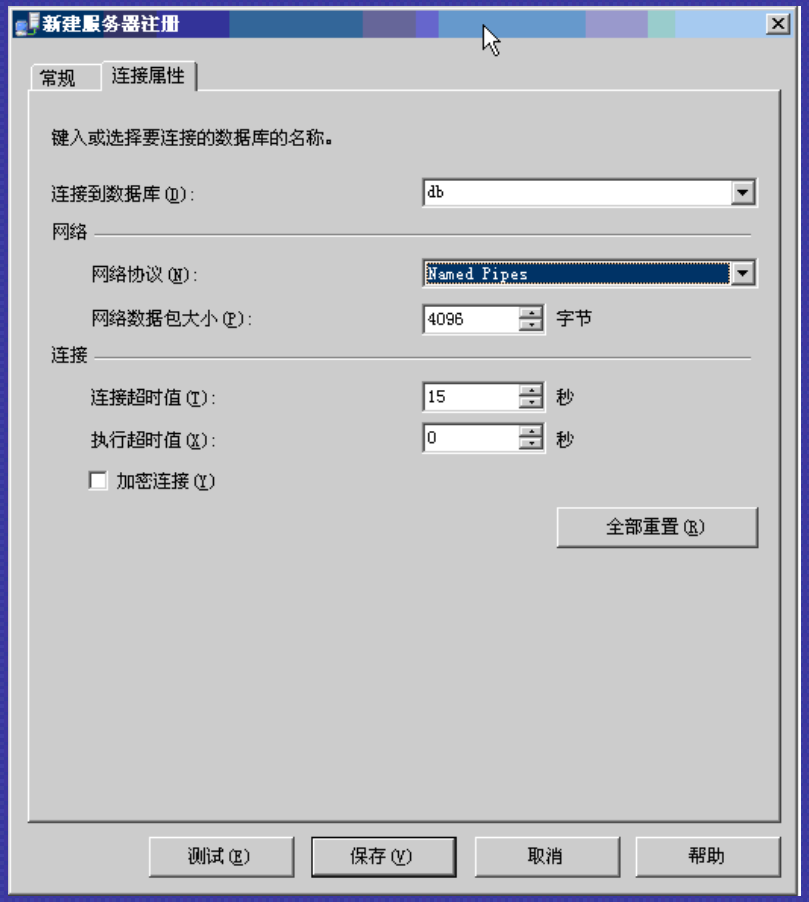


图2-4 服务器注册连接属性窗口

2.1.2 服务器注册与连接

- 要和已注册的服务器实现“连接”，则需要使用右键单击一个服务器，指向“连接”，然后单击“对象资源管理器”，如图2-5所示。
- 与连接服务器相反的是断开服务器，只要在所要断开的服务器上单击右键，选择“断开”即可。注意断开服务器并不是从计算机中将服务器删除，而只是从SQL Server管理平台中删除了对该服务器的引用。需要再次使用该服务器时，只需在SQL Server管理平台中重新连接即可。

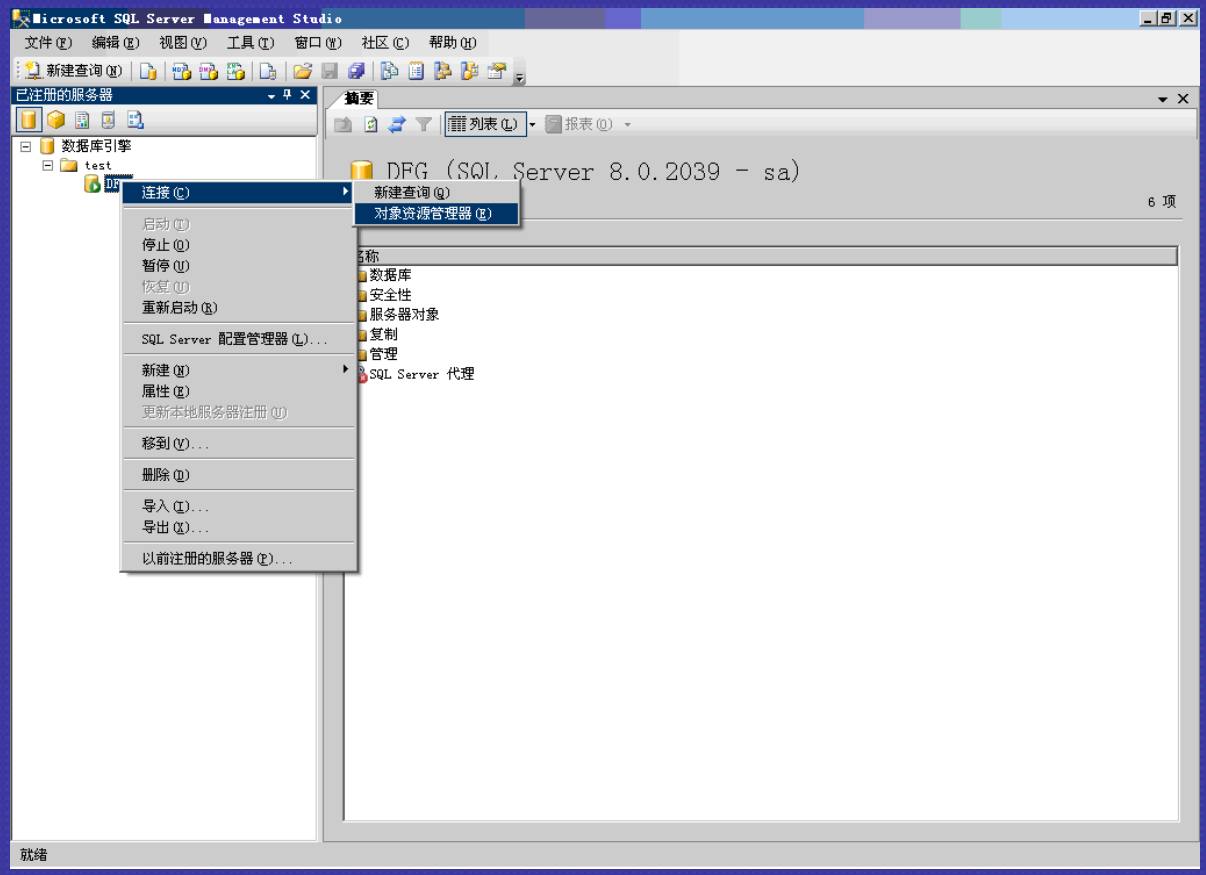


图2-5 连接已注册的服务器

2.2 服务器启动、暂停和停止

- 在SQL Server管理平台中，在所要启动的服务器上单击右键，从弹出的快捷菜单中选择“启动”选项，即可启动服务器。
- 暂停和关闭服务器的方法与启动服务器的方法类似，只需在相应的快捷菜单中选择“暂停（Pause）”或“停止（Stop）”选项即可，如图2-6所示。

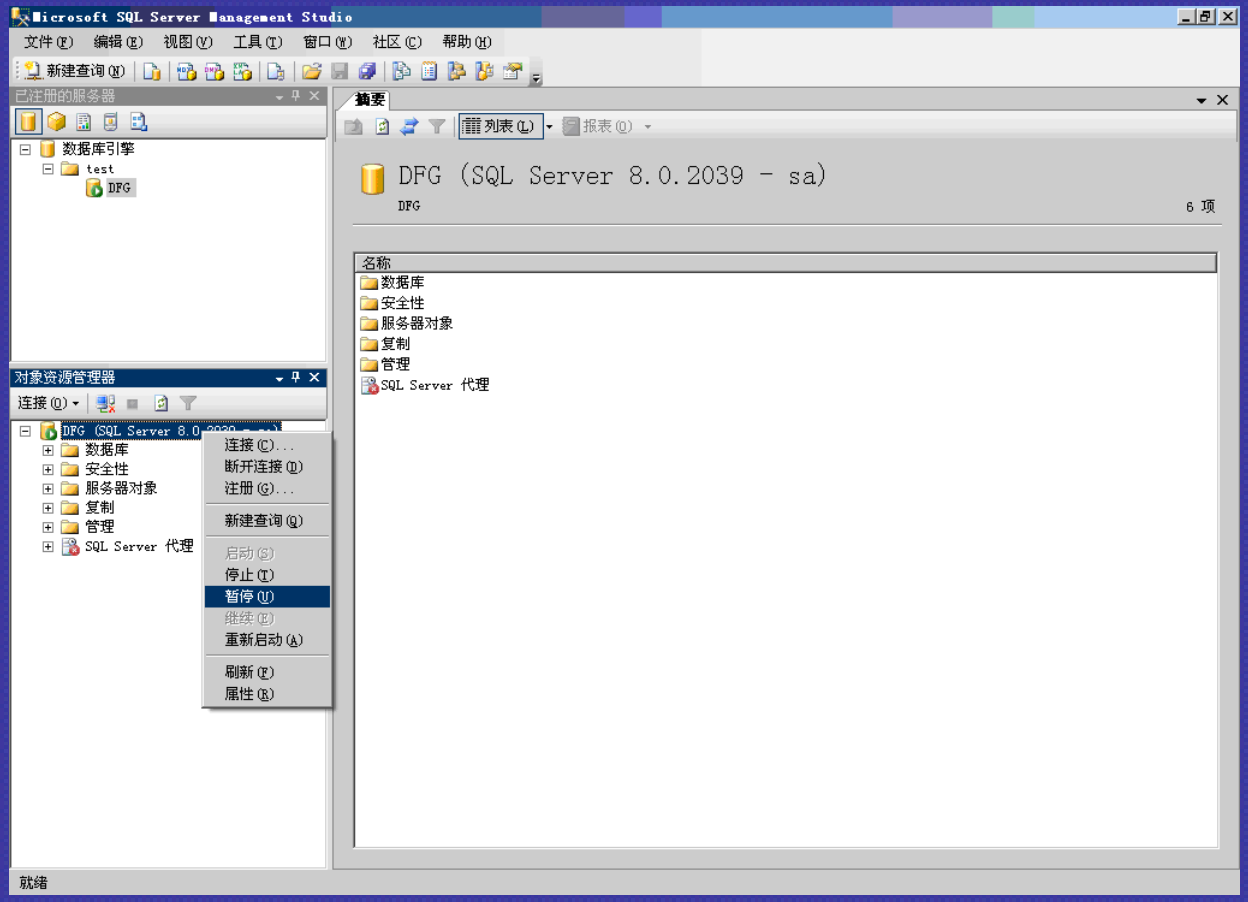


图2-6 暂停或关闭服务器选项

2.3 服务器配置选项设置

- 使用SQL Server管理平台配置服务器的操作方法为：在SQL Server管理平台中用右键单击所要进行配置的服务器，从快捷菜单中选择“属性（Properties）”选项，就会出现如图2-7所示的对话框，其中可以进行服务器的属性（配置选项）的设置。
- 在如图2-7所示的服务器属性对话框中共有7个选项。这7个选项分别是：常规选项、内存选项、处理器选项、安全性选项、连接选项、数据库设置选项、高级选项。

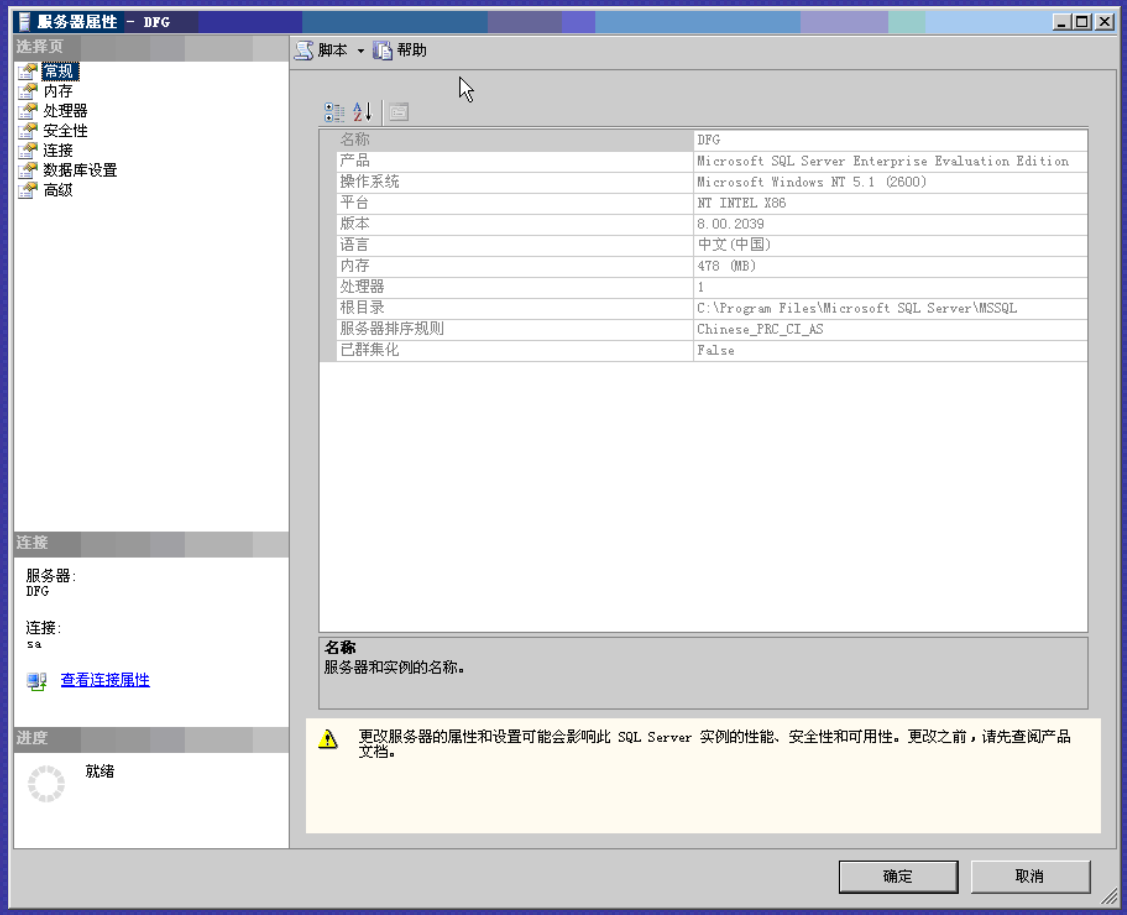


图2-7 服务器属性设置对话框

第3章 Transact-SQL语言

3.1 数据定义语言

数据定义语言（DDL）是指用来定义和管理数据库以及数据库中各种对象的语句，这些语句包括CREATE、ALTER和DROP等。在SQL Server 2005中，数据库对象包括表、视图、触发器、存储过程、规则、默认、用户自定义的数据类型等。这些对象的创建、修改和删除等都可以通过使用CREATE，ALTER，DROP等语句来完成。

3.1 数据定义语言

例3-1 创建数据库表

```
/*下面的例子将创建表S。*/
CREATE TABLE S
(
  sno  char(10)  NOT NULL      /*学号字段*/
       CONSTRAINT PK_sno PRIMARY KEY CLUSTERED/*主键约束*/
       CHECK (sno like '31300501[0-9][0-9]')/*检查约束*/,
  sname     char(8)    NULL, /*姓名字段*/
  sex     char(2)   NULL, /*性别字段*/
  age    int  NULL, /*年龄字段*/
  dept   varchar(20)   NULL/*系别字段*/
 )
```

3.1 数据定义语言

例3-2 修改S表，增加一个班号列

程序清单如下：

```
ALTER TABLE S
ADD
CLASS_NO CHAR(6)
```

3.1 数据定义语言

例3-3　删除S表

程序清单如下：

- DROP table S

3.2 数据操纵语言（DML）

- 数据操纵语言是指用来查询、添加、修改和删除数据库中数据的语句，这些语句包括SELECT、INSERT、UPDATE、DELETE等。
- 将在第6章详细讲解SELECT语句的语法及实例。

3.2.2 INSERT语句

- INSERT语句用于向数据库表或者视图中加入一行数据。INSERT语句的语法形式如下：

  INSERT [INTO] table_or_view [(column_list)] VALUES(data_values)

- 其中，table_or_view是指要插入新记录的表或视图；column_list是可选项，指定待添加数据的列； VALUES子句指定待添加数据的具体值。列名的排列顺序不一定要和表定义时的顺序一致。但当指定列名表时VALUES子句值的排列顺序必须和列名表中的列名排列顺序一致，个数相等，数据类型一一对应。

3.2.2 INSERT语句

在进行数据插入操作时须注意以下几点：

（1）必须用逗号将各个数据分开，字符型数据要用单引号括起来。

（2）INTO子句中没有指定列名，则新插入的记录必须在每个属性列上均有值，且VALUES子句中值的排列顺序要和表中各属性列的排列顺序一致。

（3）将VALUES子句中的值按照INTO子句中指定列名的顺序插入到表中。

（4）对于INTO子句中没有出现的列，则新插入的记录在这些列上将取空值，如上例的SCORE即赋空值。但在表定义时有NOT NULL约束的属性列不能取空值。

3.2.2 INSERT语句

例3-5 创建SC表（学生选课表），并向SC表中插入一条选课记录（'S7','C1'）。

程序清单如下：

```
CREATE TABLE SC
(
   sno  char(10)  NOT NULL,
   cno char(2)    NULL, /*课程编号字段*/
  score  numerical(4,1) NULL  /*成绩字段*/
  )
        Go
INSERT INTO SC (sno,cno) VALUES ('3130050101', 'c1')
Go
```

3.2.2 INSERT语句

下面是插入与列顺序不同的数据的例子。

例3-6 使用 column_list 及 VALUES 列表显式地指定将被插入每个列的值。

程序清单如下：

```
CREATE TABLE T1
 ( column_1 int,
column_2 varchar(30))
Go
INSERT T1 (column_2, column_1) VALUES ('This is a test',1)
```

3.2.2 INSERT语句

插入多行数据的语法格式为：

INSERT INTO table_or_view [(column_list)] 子查询

例3-7　求出各位学生的平均成绩，把结果存放在新表AVGSCORE中。

程序清单如下：

```
/*首先建立新表AVGSCORE，用来存放学号和学生的平均成绩。*/
CREATE TABLE AVGSCORE
(SNO CHAR(10),
AVGSCORE SMALLINT)
Go
/*利用子查询求出SC表中各位学生的平均成绩，把结果存放在新表
   AVGSCORE中。*/
INSERT INTO AVGSCORE
SELECT SNO,AVG(SCORE)
FROM SC
GROUP BY SNO
```

3.2.3 UPDATE语句

- UPDATE语句用于修改数据库表或视图中特定记录或者字段的数据，其语法形式如下：

  UPDATE table_or_view SET <column>=<expression>[,<column>=<expression>>]...

  [WHERE <search_condition>]

- 其中：table_or_view是指要修改的表或视图；SET子句给出要修改的列及其修改后的值，其中column为要修改的列名，expression为其修改后的值；WHERE子句指定待修改的记录应当满足的条件，WHERE子句省略时，则修改表中的所有记录。

3.2.3 UPDATE语句

- 例3-8 一个带有WHERE条件的修改语句。

程序清单如下：

```
use adventureworks
go
update person.address
set city='Boston' where  addressid=1
```

- 例3-9 将所有学生年龄增加1岁

程序清单如下：

```
UPDATE S SET AGE=AGE+1
```

3.2.3 UPDATE语句

- 例3-12 创建把讲授C5课程的教师的工资增加100元。

程序清单如下：

```
/*T表（教师基本情况表）的结构为
  T(TNO,TN,SEX,AGE,PROF,SAL,DEPT)分别表示教师的
  编号，姓名，性别，年龄，职称，工资，系别。TC表
  （教师授课表）的结构为TC(TNO,CNO)分别表示教师的
  编号，课程编号。*/
  UPDATE T  SET SAL=SAL+100
  WHERE TNO IN
  (SELECT T.TNO FROM T,TC
 WHERE T.TNO=TC.TNO AND TC.CNO='C5')
 /*通过连接查询找到讲授C5课程的教师编号。*/
```

3.2.4 DELETE语句

- 使用DELETE语句可以删除表中的一行或多行记录，其语法格式为：

  DELETE FROM table_or_view [WHERE <search_condition>]

- 其中：table_or_view是指要删除数据的表或视图；WHERE子句指定待删除的记录应当满足的条件，WHERE子句省略时，则删除表中的所有记录。
- 下面是删除一行记录的例子。

  例3-14 删除张益琳教师的记录。

  程序清单如下：

  DELETE FROM T WHERE TN=’张益琳’

3.2.4 DELETE语句

- 下面是删除多行记录的例子。
- 例 3-15 删除所有教师的授课记录。

  程序清单如下：

  DELETE FROM TC
- 执行此语句后，TC表即为一个空表，但其定义仍存在数据字典中。
- 例3-16 删除李明同学选课的记录。

  DELETE FROM SC WHERE SNO=
  (SELECT SNO FROM S WHERE SN=’李明’)

3.3 数据控制语言

- 数据控制语言（DCL）是用来设置或更改数据库用户或角色权限的语句，包括GRANT，DENY，REVOKE等语句。在默认状态下，只有sysadmin，dbcreator，db_owner或db_securityadmin等人员才有权力执行数据控制语言。

3.3.1 GRANT语句

1、语句权限与角色的授予

- SQL语言使用GRANT语句为用户授予语句权限的语法格式为：

  GRANT <语句权限>|<角色> [,<语句权限>|<角色>]…

  TO <用户名>|<角色>|PUBLIC[,<用户名>|<角色>]…

  [WITH ADMIN OPTION]

- 其语义为：将指定的语句权限授予指定的用户或角色。其中：

（1）PULBIC代表数据库中的全部用户；

（2）WITH ADMIN OPTION为可选项，指定后则允许被授权的用户将指定的系统特权或角色再授予其他用户或角色。

3.3.1 GRANT语句

1、语句权限与角色的授予

- 例3-17 给用户 Mary 和 John 以及 Windows NT 组 Corporate\BobJ 授予多个语句权限。

  程序清单如下：

```
GRANT CREATE DATABASE, CREATE TABLE
TO Mary, John, [Corporate\BobJ]
```

- 例3-18 为用户ZhangYiLin授予CREATE TABLE的语句权限。

  程序清单如下：

```
GRANT CREATE TABLE
TO ZhangYiLin
```

3.3.1 GRANT语句

2、对象权限与角色的授予

- 数据库管理员拥有系统权限，而作为数据库的普通用户，只对自己创建的基本表、视图等数据库对象拥有对象权限。如果要共享其他的数据库对象，则必须授予他一定的对象权限。
- 同语句权限的授予类似，SQL语言使用GRANT语句为用户授予对象权限，其语法格式为：

  GRANT ALL|<对象权限>[(列名[,列名]…)][,<对象权限>]…ON <对象名>

  TO <用户名>|<角色>|PUBLIC[,<用户名>|<角色>]…

  [WITH ADMIN OPTION]
- 其语义为：将指定的操作对象的对象权限授予指定的用户或角色。其中：

  （1）ALL代表所有的对象权限。

  （2）列名用于指定要授权的数据库对象的一列或多列。如果不指定列名，被授权的用户将在数据库对象的所有列上均拥有指定的特权。实际上，只有当授予INSERT、UPDATE权限时才需指定列名。

  （3）ON子句用于指定要授予对象权限的数据库对象名，可以是基本表名、视图名等。

  （4）WITH ADMIN OPTION为可选项，指定后则允许被授权的用户将权限再授予其他用户或角色。

3.3.1 GRANT语句

2、对象权限与角色的授予

- 例3-19 在权限层次中授予对象权限。首先，给所有用户授予 SELECT 权限，然后，将特定的权限授予用户 Mary，John 和 Tom。

程序清单如下：

```
GRANT SELECT
 ON s
 TO public
 GO
 GRANT INSERT, UPDATE, DELETE
 ON s
 TO Mary, John, Tom
 GO
```

3.3.1 GRANT语句

2、对象权限与角色的授予

- 例3-20 将查询T表和修改教师职称的权限授予USER3，并允许将此权限授予其他用户。

  程序清单如下：

```
GRANT SELECT,UPDATE(PROF)
ON T TO USER3
WITH ADMIN OPTION
```

- 上例中，USER3具有此对象权限，并可使用GRANT命令给其他用户授权，如下例，USER3将此权限授予USER4：

```
GRANT SELECT,UPDATE(PROF)
ON T
TO USER4
```

3.3.2 REVOKE语句

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

1、语句权限与角色的收回

数据库管理员可以使用REVOKE语句收回语句权限，其语法格式为：

REVOKE <语句权限>|<角色> [,<语句权限>|<角色>]…

FROM <用户名>|<角色>|PUBLIC[,<用户名>|<角色>]…

•例：收回用户ZHANGYILIN所拥有的CREATE TABLE的语句权限。

```
REVOKE CREATE TABLE
FROM ZHANGYILIN
```

2、对象权限与角色的收回

所有授予出去的权力在必要时都可以由数据库管理员和授权者收回，收回对象权限仍然使用REVOKE语句，其语法格式为：

REVOKE <对象权限>|<角色> [,<对象权限>|<角色>]…

FROM <用户名>|<角色>|PUBLIC[,<用户名>|<角色>]…

3.3.2 REVOKE语句

•例3-21 收回用户USER1对C表的查询权限。

程序清单如下：

```
REVOKE SELECT
ON C
FROM USER1
```

•例3-22 收回用户USER3查询T表和修改教师职称的权限。

程序清单如下：

```
REVOKE SELECT,UPDATE(PROF)
ON T
FROM USER3
```

•在上例中，USER3将对T表的权限授予了USER4，在收回USER3对T表的权限的同时，系统会自动收回USER4对T表的权限。

3.3.2 REVOKE语句

•例3-23 首先从 public 角色中收回 SELECT 权限，然后，收回用户 Mary，John 和 Tom 的特定权限。

程序清单如下：

```
USE pubs
GO
REVOKE  SELECT ON s FROM public
GO
REVOKE  INSERT, UPDATE, DELETE
ON s
FROM Mary, John, Tom
```

3.3.3 DENY语句

- DENY语句用于拒绝给当前数据库内的用户或者角色授予权限，并防止用户或角色通过其组或角色成员继承权限。
- 否定语句权限的语法形式为：

  DENY ALL|<语句权限>|<角色> [,<语句权限>|<角色>]…

  TO <用户名>|<角色>|PUBLIC[,<用户名>|<角色>]…
- 否定对象权限的语法形式为：

  DENY ALL|<对象权限>[(列名[,列名]…)][,<对象权限>]…ON <对象名>

  TO <用户名>|<角色>|PUBLIC[,<用户名>|<角色>]…

3.3.3 DENY语句

- 例3-24 首先给 public 角色授予 SELECT 权限，然后，拒绝用户 Mary，John 和 Tom 的特定权限。

程序清单如下：

```
USE pubs
GO
GRANT SELECT
ON s
TO public
GO
DENY SELECT, INSERT, UPDATE, DELETE
ON s
TO Mary,John,Tom
```

3.4 系统存储过程

- 系统存储过程是SQL Server系统创建的存储过程，它的目的在于能够方便地从系统表中查询信息，或者完成与更新数据库表相关的管理任务或其他的系统管理任务。系统存储过程可以在任意一个数据库中执行。系统存储过程创建并存放于系统数据库master中，并且名称以sp_或者xp_开头。一些系统过程只能由系统管理员使用，而有些系统过程通过授权可以被其他用户使用。

3.4 系统存储过程

系统存储过程的部分示例如下：

- sp_addtype：用于定义一个用户定义数据类型；
- sp_configure：用于管理服务器配置选项设置；
- xp_sendmail：用于发送电子邮件或寻呼信息；
- sp_stored_procedures：用于返回当前数据库中的存储过程的清单；
- sp_help：用于显示参数清单和其数据类型；
- sp_depends：用于显示存储过程依据的对象或者依据存储过程的对象；
- sp_helptext：用于显示存储过程的定义文本；
- sp_rename：用于修改当前数据库中用户对象的名称。

3.5 其他语言元素

3.5.1 注释

3.5.2 变量

3.5.3 运算符

3.5.4 函数

3.5.5 流程控制语句

3.5.1 注释

- 注释是程序代码中不执行的文本字符串（也称为注解）。使用注释对代码进行说明，不仅能使程序易读易懂，而且有助于日后的管理和维护。注释通常用于记录程序名称、作者姓名和主要代码更改的日期。注释还可以用于描述复杂的计算或者解释编程的方法。
- 在SQL Server中，可以使用两种类型的注释字符：一种是ANSI标准的注释符“--”，它用于单行注释；另一种是与C语言相同的程序注释符号，即“/* */”。“/*”用于注释文字的开头，“*/”用于注释文字的结尾，利用它们可以在程序中标识多行文字为注释。当然，单行注释也可以使用“/* */”，我们只需将注释行以“/*”开头并以“*/”结尾即可。反之，段落注释也可以使用“--”，只需使段落注释的每一行都以“--”开头即可。

3.5.1 注释

- 例3-25　使用两种注释类型的例子。
  程序清单如下：

```
USE AdventureWorks
GO
 -- First line of a multiple-line comment.
 -- Second line of a multiple-line comment.
SELECT * FROM person.address
GO
/* 注释语句的第一行.
注释语句的第二行. */
SELECT * FROM Production.Product
GO
-- 在Transact-SQL语言调试过程中使用注释语句。
```

3.5.2 变量

- 变量是一种语言中必不可少的组成部分。Transact-SQL语言中有两种形式的变量，一种是用户自己定义的局部变量，另外一种是系统提供的全局变量。
- 1. 局部变量

  局部变量是一个能够拥有特定数据类型的对象，它的作用范围仅限制在程序内部。局部变量被引用时要在其名称前加上标志“@”，而且必须先用DECLARE命令定义后才可以使用。
- 定义局部变量的语法形式如下：

  DECLAER {@local_variable data_type} […n]
- 其中，参数@local_variable用于指定局部变量的名称，变量名必须以符号@开头，并且局部变量名必须符合SQL Server的命名规则。参数data_type用于设置局部变量的数据类型及其大小。data_type可以是任何由系统提供的或用户定义的数据类型。但是，局部变量不能是text，ntext 或 image 数据类型。
- 使用DECLARE命令声明并创建局部变量之后，会将其初始值设为NULL，如果想要设定局部变量的值，必须使用SELECT命令或者SET命令。其语法形式为：

  SET { { @local_variable = expression } 或者 SELECT { @local_variable = expression } [,...n]
- 其中，参数@local_variable是给其赋值并声明的局部变量，参数expression是任何有效的SQL Server表达式。

3.5.2 变量

1. 局部变量

- 例3-26　创建一个@myvar 变量，然后将一个字符串值放在变量中，最后输出 @myvar 变量的值。

  程序清单如下：

```
DECLARE @myvar  char(20)
select  @myvar = 'This is a test'
SELECT @myvar
GO
```

- 例3-27　通过查询给变量赋值。

  程序清单如下：

```
USE adventureworks
GO
DECLARE @rows int
SET @rows = (SELECT COUNT(*)  FROM
humanresources.employee)
```

3.5.2 变量

- 2. 全局变量

  除了局部变量之外，SQL Server系统本身还提供了一些全局变量。全局变量是SQL Server系统内部使用的变量，其作用范围并不仅仅局限于某一程序，而是任何程序均可以随时调用。全局变量通常存储一些SQL Server的配置设定值和统计数据。用户可以在程序中用全局变量来测试系统的设定值或者是Transact-SQL命令执行后的状态值。在使用全局变量时应该注意以下几点：

  （1）全局变量不是由用户的程序定义的，它们是在服务器级定义的。

  （2）用户只能使用预先定义的全局变量。

  （3）引用全局变量时，必须以标记符“@@”开头。

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

3.5.2 变量

- 2. 全局变量
- 例3-29　显示到当前日期和时间为止试图登录SQL Server的次数。

  程序清单如下：

  SELECT GETDATE( ) AS '当前的时期和时间',

  @@CONNECTIONS AS '试图登录的次数'

3.5.3 运算符

- 运算符是一些符号，它们能够用来执行算术运算、字符串连接、赋值以及在字段、常量和变量之间进行比较。在SQL Server 2005中，运算符主要有以下六大类：算术运算符、赋值运算符、位运算符、比较运算符、逻辑运算符和字符串串联运算符。

1．算术运算符

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

2．赋值运算符

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

3.5.3 运算符

3. 位运算符

- 位运算符使我们能够在整型数据或者二进制数据（image 数据类型除外）之间执行位操作。此外，在位运算符左右两侧的操作数不能同时是二进制数据。表3-1列出了所有的位运算符及其含义。

表3-1　位运算符

运算符	含义
&（按位 AND）	按位 AND（两个操作数）
|（按位 OR）	按位 OR（两个操作数）
^（按位互斥 OR）	按位互斥 OR（两个操作数）

3.5.3 运算符

4. 比较运算符

比较运算符亦称为关系运算符，用于比较两个表达式的大小或是否相同，其比较的结果是布尔值，即TRUE（表示表达式的结果为真）、FALSE（表示表达式的结果为假）以及UNKNOWN。除了 text，ntext 或 image 数据类型的表达式外，比较运算符可以用于所有的表达式。

5. 逻辑运算符

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

三个运算符的优先级别为：NOT，AND，OR。

6. 字符串串联运算符

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

3.5.3 运算符

- 在SQL Server 2005中，运算符的优先等级从高到低如下所示，如果优先等级相同，则按照从左到右的顺序进行运算。

（1）括号：（）；
（2）乘、除、求模运算符：*，/，%；
（3）加减运算符：＋，－；
（4）比较运算符：=，>，<，>=，<=，<>，!=，!>，!<；
（5）位运算符：^，&，|；
（6）逻辑运算符：NOT；
（7）逻辑运算符：AND；
（8）逻辑运算符：OR。

3.5.4 函数

- 在Transact-SQL语言中，函数被用来执行一些特殊的运算以支持SQL Server的标准命令。SQL Server包含多种不同的函数用以完成各种工作，每一个函数都有一个名称，在名称之后有一对小括号，如：gettime( )。大部分的函数在小括号中需要一个或者多个参数。
- Transact-SQL 编程语言提供了四种函数：行集函数、聚合函数、Ranking函数、标量函数。

3.5.4 函数

1. 行集函数

- 行集函数可以在Transact-SQL语句中当作表引用。
- 例3-33　通过行集函数OPENQUERY()执行一个分布式查询，以便从服务器local中提取表department中的记录。

  程序清单如下：

  ```
  select * from  openquery(local,'select * from department')
  ```

3.5.4 函数

2. 聚合函数

- 聚合函数用于对一组值进行计算并返回一个单一的值。除COUNT 函数之外，聚合函数忽略空值。聚合函数经常与 SELECT 语句的 GROUP BY 子句一同使用。仅在下列项中聚合函数允许作为表达式使用：SELECT 语句的选择列表（子查询或外部查询）；COMPUTE 或 COMPUTE BY 子句；HAVING 子句。

3.5.4 函数

2. 聚合函数

计算 Adventure Works Cycles 的副总所用的平均休假小时数以及总的病假小时数。对检索到的所有行，每个聚合函数都生成一个单独的汇总值。

程序清单如下。

```
USE AdventureWorks;
GO
SELECT AVG(VacationHours)as 'Average vacation hours',
    SUM  (SickLeaveHours) as 'Total sick leave hours'
FROM HumanResources.Employee
WHERE Title LIKE 'Vice President%'
```

3.5.4 函数

3. Ranking函数

- Ranking函数为查询结果数据集分区中的每一行返回一个序列值。依据此函数，一些行可能取得和其他行一样的序列值。Transact-SQL提供以下一些Ranking函数：RANK；DENSE_RANK；NTILE；ROW_NUMBER。

3.5.4 函数

4. 标量函数

标量函数用于对传递给它的一个或者多个参数值进行处理和计算，并返回一个单一的值。标量函数可以应用在任何一个有效的表达式中。标量函数可分为如表3-4所示的几大类

表3-4 标量函数的分类

函数分类	解释
配置函数	返回当前的配置信息
游标函数	返回有关游标的信息
日期和时间函数	对日期和时间输入值进行处理
数学函数	对作为函数参数提供的输入值执行计算
元数据函数	返回有关数据库和数据库对象的信息
安全函数	返回有关用户和角色的信息
字符串函数	对字符串（char 或 varchar）输入值执行操作
系统函数	执行操作并返回有关SQL Server中的值、对象和设置的信息
系统统计函数	返回系统的统计信息
文本和图像函数	对文本或图像输入值或列执行操作，返回有关这些值的信息

3.5.4 函数

4. 标量函数

（1）字符串函数

●字符串函数可以对二进制数据、字符串和表达式执行不同的运算，大多数字符串函数只能用于char和varchar数据类型以及明确转换成char和varchar的数据类型，少数几个字符串函数也可以用于binary和varbinary数据类型。

●字符串函数可以分为以下几大类：

➢基本字符串函数：UPPER，LOWER，SPACE，REPLICATE，STUFF，REVERSE，LTRIM，RTRIM。

➢字符串查找函数：CHARINDEX，PATINDEX。

➢长度和分析函数：DATALENGTH，SUBSTRING，RIGHT。

➢转换函数：ASCH，CHAR，STR，SOUNDEX，DIFFERENCE。

3.5.4 函数

4. 标量函数

（1）字符串函数

例3-38　使用 LTRIM 函数删除字符变量中的起始空格。

程序清单如下：

```
DECLARE @string_to_trim varchar(60)
SET @string_to_trim = '    Five spaces are at the beginning
    of this    string.'
SELECT 'Here is the string without the leading spaces: ' +
LTRIM(@string_to_trim)
```

3.5.4 函数

4. 标量函数

（1）字符串函数

例3-39　使用可选的start_location参数从addressline1列的第2个字符开始查找“shoe”程序清单如下。

```
USE adventureworks
SELECT CHARINDEX('shoe', addressline1,2) FROM person.address
WHERE addressid = '5'
```

3.5.4 函数

4. 标量函数

（1）字符串函数

例3-40显示如何只返回字符串的一部分。该查询在一列中返回person.contact表中的姓氏，在另一列中返回person.contact表中的名字首字母。

程序清单如下：

```
USE adventureworks
SELECT lastname, SUBSTRING(firstname, 1, 1)
FROM person.contact
ORDER BY lastname
```

例3-41　在第一个字符串（abcdef）中删除从第二个位置（字符 b）开始的三个字符，然后在删除的起始位置插入第二个字符串，创建并返回一个字符串。

程序清单如下：

```
SELECT STUFF('abcdef', 2, 3, 'ijklmn')
```

3.5.4 函数

4. 标量函数

（2）日期和时间函数

日期和时间函数用于对日期和时间数据进行各种不同的处理和运算，并返回一个字符串、数字值或日期和时间值。与其他函数一样，可以在SELECT语句的SELECT和WHERE子句以及表达式中使用日期和时间函数。

表3-6　日期和时间函数的类型

函数	参数	功能
DATEADD	（datepart,number,date）	以datepart指定的方式，返回date加上number之和
DATEDIFF	（datepart,date1,date2）	以datepart指定的方式，返回date2与date1之差
DATENAME	（datepart,date）	返回日期date中datepart指定部分所对应的字符串
DATEPART	（datepart,date）	返回日期date中datepart指定部分所对应的整数值
DAY	（date）	返回指定日期的天数
GETDATE	（）	返回当前的日期和时间
MONTH	（date）	返回指定日期的月份数
YEAR	（date）	返回指定日期的年份数

3.5.4 函数

4. 标量函数

（2）日期和时间函数

例3-42　显示在humanresources.employee 表中雇用日期到当前日期间的天数。

程序清单如下。

```
USE adventureworks
SELECT DATEDIFF(day, hiredate, getdate())  AS diffdays
FROM humanresources.employee
```

例3-43　从GETDATE函数返回的日期中提取月份名。

程序清单如下：

```
SELECT DATENAME(month, getdate())  AS  'Month Name'
```

3.5.4 函数

4. 标量函数

（3）数学函数

- 数学函数用于对数字表达式进行数学运算并返回运算结果。数学函数可以对SQL Server提供的数字数据（decimal、integer、float、real、money、smallmoney、smallint 和 tinyint）进行处理。在SQL Server中，常用的数学函数如表3-8所示。

3.5.4 函数

4. 标量函数

（3）数学函数

例3-45　在同一表达式中使用CEILING()，FLOOR()，ROUND()函数。

程序清单如下：

```
select  ceiling(13.4)，floor(13.4)，
  round(13.4567,3)
```

3.5.4 函数

4. 标量函数

（4）系统函数

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

3.5.4 函数

4. 标量函数

（4）系统函数

- 转换函数有两个：CONVERT和CAST。
- CAST函数允许把一个数据类型强制转换为另一种数据类型，其语法形式为：

  CAST(expression AS data_type)
- CONVERT函数允许用户把表达式从一种数据类型转换成另一种数据类型，还允许把日期转换成不同的样式，其语法形式为：

  CONVERT (data_type[(length)],expression [,style])

3.5.4 函数

4. 标量函数

（4）系统函数

- 转换函数有两个：CONVERT和CAST。
- CAST函数允许把一个数据类型强制转换为另一种数据类型，其语法形式为：

  CAST(expression AS data_type)
- CONVERT函数允许用户把表达式从一种数据类型转换成另一种数据类型，还允许把日期转换成不同的样式，其语法形式为：

  CONVERT (data_type[(length)],expression [,style])

3.5.4 函数

4. 标量函数

（4）系统函数

例3-46 示例检索列表价格的第一位是 3 的产品的名称，并将ListPrice转换为int。

程序清单如下。

```
USE AdventureWorks
GO
SELECT SUBSTRING(Name, 1, 30) AS ProductName,
ListPrice
FROM Production.Product
WHERE CAST(ListPrice AS int) LIKE '3%'
GO
```

3.5.4 函数

4. 标量函数

（4）系统函数

例3-47　用style 参数将当前日期转换为不同格式的字符串。

程序清单如下：

```
SELECT '101'=CONVERT(char, GETDATE(), 101),
'1'=CONVERT(char, GETDATE(), 1),
'112'=CONVERT(char, GETDATE(), 112)
```

3.5.4 函数

4. 标量函数

（4）系统函数

例3-48　从adventureworks数据库中返回person.contact表的首列名称

程序清单如下。

```
USE adventureworks
SELECT COL_NAME(OBJECT_ID('person.contact'), 1)
```

例3-49　检查 sysdatabases 中的每一个数据库，使用数据库标识号来确定数据库名称。

程序清单如下：

```
USE master
SELECT dbid, DB_NAME(dbid)  AS  DB_NAME
FROM sysdatabases
ORDER BY dbid
```

3.5.5 流程控制语句

- 流程控制语句是指那些用来控制程序执行和流程分支的语句，在SQL Server 2005中，流程控制语句主要用来控制SQL语句、语句块或者存储过程的执行流程。

3.5.5 流程控制语句

1. IF…ELSE语句

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

```
IF Boolean_expression
    { sql_statement | statement_block }
[ ELSE
    { sql_statement | statement_block } ]
```

3.5.5 流程控制语句

1. IF…ELSE语句

例3-51 显示带有语句块的IF条件。如果产品的平均价格不低于$15，那么就显示文本：Average price is more than $15

程序清单如下。

```
USE adventureworks
IF (SELECT AVG(listprice) FROM production.product WHERE color = 'blue') <
$15
BEGIN
   PRINT 'The following product are excellent:'
   PRINT ' '
   SELECT SUBSTRING(name, 1, 15) AS name
   FROM production.product
   WHERE color = 'blue'
END
ELSE
   PRINT 'Average price is more than $15.'
```

3.5.5 流程控制语句

2. BEGIN...END语句

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

```
BEGIN
    { sql_statement
        | statement_block
    }
END
```

3.5.5 流程控制语句

2. BEGIN…END语句

例3-52 利用 BEGIN 和 END 语句使得 IF 语句在取值为 FALSE 时跳过语句块。

程序清单如下：

```
IF (@@ERROR <> 0)
BEGIN
   SET @ErrorSaveVariable = @@ERROR
   PRINT 'Error encountered, ' +
         CAST(@ErrorSaveVariable AS VARCHAR(10))
END
```

3.5.5 流程控制语句

3. GO 语句

Go 语句是批的结束语句。批是一起提交并作为一个组执行的若干SQL语句。

例3-53 用Go 语句作为批的结束语句。

程序清单如下：

```
•USE adventureworks
GO
DECLARE @MyMsg VARCHAR(50)
SELECT @MyMsg = 'Hello, World.'
GO -- @MyMsg 在Go语句后失效。
```

3.5.5 流程控制语句

4．CASE语句

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

•它们的语法形式分别为：

```
CASE input_expression
    WHEN when_expression THEN result_expression
        [ ...n ]
    [
        ELSE else_result_expression
    ENDCASE

 WHEN Boolean_expression THEN result_expression
        [ ...n ]
    [
        ELSE else_result_expression
    END
```

3.5.5 流程控制语句

4．CASE语句

例3-54　使用CASE函数去更改产品的分类显示，以使其更易于理解

程序清单如下。

```
USE adventureworks
SELECT   Category =
      CASE color
        WHEN 'red' THEN 'red color'
        WHEN 'black' THEN 'black color'
        WHEN 'silver' THEN 'silver color'
        WHEN 'yellow' THEN 'yellow color'
        WHEN 'blue' THEN 'blue color'
      END,
   CAST(name AS varchar(25)) AS 'Shortened name',
   listprice AS Price
FROM production.product
WHERE listprice IS NOT NULL
ORDER BY color, price
COMPUTE AVG(listprice) BY color
```

3.5.5 流程控制语句

4．CASE语句

例3-55　使用搜索CASE函数，根据产品的价格范围将价格显示为文本注释程序清单如下。

```
USE adventureworks
SELECT    'Price Category'=
     CASE
        WHEN listprice IS NULL THEN 'Not yet priced'
         WHEN listprice < 10 THEN 'Very Reasonable Title'
         WHEN listprice >= 10 and listprice < 20 THEN 'Coffee Table Title'
         ELSE 'Expensive!'
      END,
   CAST(name AS varchar(20)) AS 'Shortened name'
FROM production.product
ORDER BY listprice
GO
```

3.5.5 流程控制语句

5. WHILE…CONTINUE…BREAK语句

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

•其语法形式为：

```
WHILE Boolean_expression
   { sql_statement | statement_block }
   [ BREAK ]
   { sql_statement | statement_block }
   [ CONTINUE ]
```

3.5.5 流程控制语句

5．WHILE…CONTINUE…BREAK语句

例3-56　在嵌套的 IF...ELSE 和 WHILE 中使用 BREAK 和 CONTINUE。如果平均价格少于 $30，WHILE 循环就将价格加倍，然后选择最高价。如果最高价少于或等于 $50，WHILE 循环重新启动并再次将价格加倍。该循环不断地将价格加倍直到最高价格超过 $50，然后退出 WHILE 循环并打印一条消息。

程序清单如下：

```
USE adventureworks
WHILE (SELECT AVG(listprice)  FROM production.product) < $30
BEGIN
   UPDATE production.product
      SET listprice = listprice * 2
   SELECT MAX(listprice) FROM production.product
   IF (SELECT MAX(listprice) FROM production.product) > $50
      BREAK
   ELSE
      CONTINUE
END
PRINT 'Too much for the market to bear'
```

3.5.5 流程控制语句

6. GOTO语句

•GOTO语句可以使程序直接跳到指定的标有标识符的位置处继续执行，而位于GOTO语句和标识符之间的程序将不会被执行。GOTO语句和标识符可以用在语句块、批处理和存储过程中，标识符可以为数字与字符的组合，但必须以“:”结尾。如：‘a1:’。在GOTO语句行，标识符后面不用跟“:”。GOTO语句的语法形式为：

```
GOTO label
......
label:
```

例3-57　利用GOTO语句求出从1加到5的总和。

程序清单如下：

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

3.5.5 流程控制语句

7．WAITFOR语句

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

•WAITFOR语句的语法形式为：

```
WAITFOR { DELAY 'time' | TIME 'time' }
```

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

例3-58　使用WAITFOR TIME语句，以便在晚上10:20执行存储过程update_all_stats。

程序清单如下：

```
BEGIN
   WAITFOR TIME '22:20'
   EXECUTE update_all_stats
END
```

3.5.5 流程控制语句

8. RETURN语句

- RETURN语句用于无条件地终止一个查询、存储过程或者批处理，此时位于RETURN语句之后的程序将不会被执行。
- RETURN语句的语法形式为：

RETURN [integer_expression]

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

3.5.5 流程控制语句

8. RETURN语句

例3-59　显示如果在执行 findjobs 时没有给出用户名作为参数，RETURN 则将一条消息发送到用户的屏幕上然后从过程中退出。如果给出用户名，将从适当的系统表中检索由该用户在当前数据库内创建的所有对象名。

程序清单如下：

```
CREATE PROCEDURE findjobs @nm sysname = NULL
AS
IF @nm IS NULL
  BEGIN
    PRINT 'You must give a username'
     RETURN
  END
ELSE
  BEGIN
    SELECT o.name, o.id, o.uid
    FROM sysobjects o INNER JOIN master..syslogins l
       ON o.uid = l.sid
    WHERE l.name = @nm
  END
```

第4章 数据库管理

4.1 数据库存储结构

数据库的存储结构分为逻辑存储结构和物理存储结构两种。

1、数据库的逻辑存储结构指的是数据库是由哪些性质的信息所组成，SQL Server的数据库不仅仅只是数据的存储，所有与数据处理操作相关的信息都存储在数据库中。实际上，SQL Server的数据库是由诸如表、视图、索引等各种不同的数据库对象所组成，它们分别用来存储特定信息并支持特定功能，构成数据库的逻辑存储结构。

2、数据库的物理存储结构则是讨论数据库文件是如何在磁盘上存储的。数据库在磁盘上是以文件为单位存储的，由数据库文件和事务日志文件组成，一个数据库至少应该包含一个数据库文件和一个事务日志文件。

4.1.1 数据库文件

1. 主数据库文件（Primary Database File）

2. 辅助数据库文件（Secondary Database File）

3. 事务日志文件

4.1.2 数据库文件组

- 为了便于分配和管理，SQL Server允许将多个文件归纳为同一组，并赋予此组一个名称，这就是文件组。
- 与数据库文件一样，文件组也分为主文件组（Primary File Group）和次文件组（Secondary File Group）。

4.2 创建、修改和删除数据库

4.2.1 创建数据库

4.2.2 修改数据库

4.2.3 删除数据库

4.2.1 创建数据库

每个数据库都由以下几个部分的数据库对象所组成：

关系图、表、视图、存储过程、用户、角色、规则、默认、用户自定义数据类型和用户自定义函数。

4.2.1　创建数据库

创建数据库的方法有以下三种（实际为两种）：

1、使用模板创建数据库（实际上也是使用Transact-SQL语言创建数据库）

2、使用SQL Server管理平台创建数据库

4.2.1　创建数据库

1、使用模板创建数据库

- 以下图4-1 到图4-2是使用模板创建数据库，用户根据提示操作，即可创建数据库。

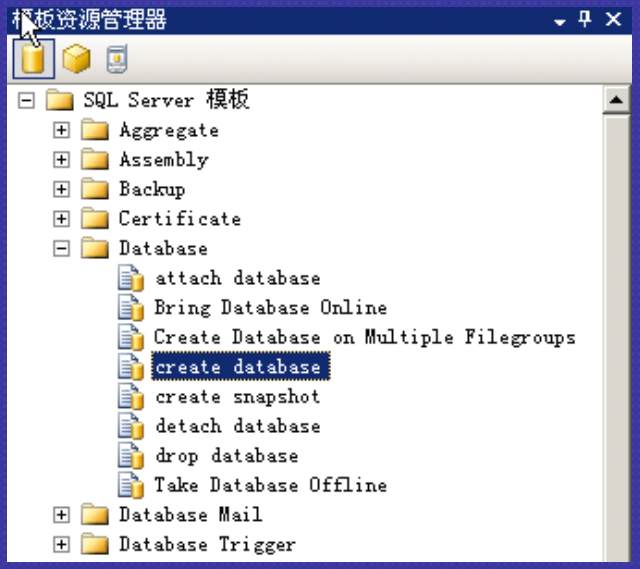


图4-1　选择创建数据库模板对话框

4.2.1 创建数据库

1、使用模板创建数据库

- 在图4-1中双击“创建数据库”命令，就会出现创建数据库的SQL语言模板，如图4-2所示。

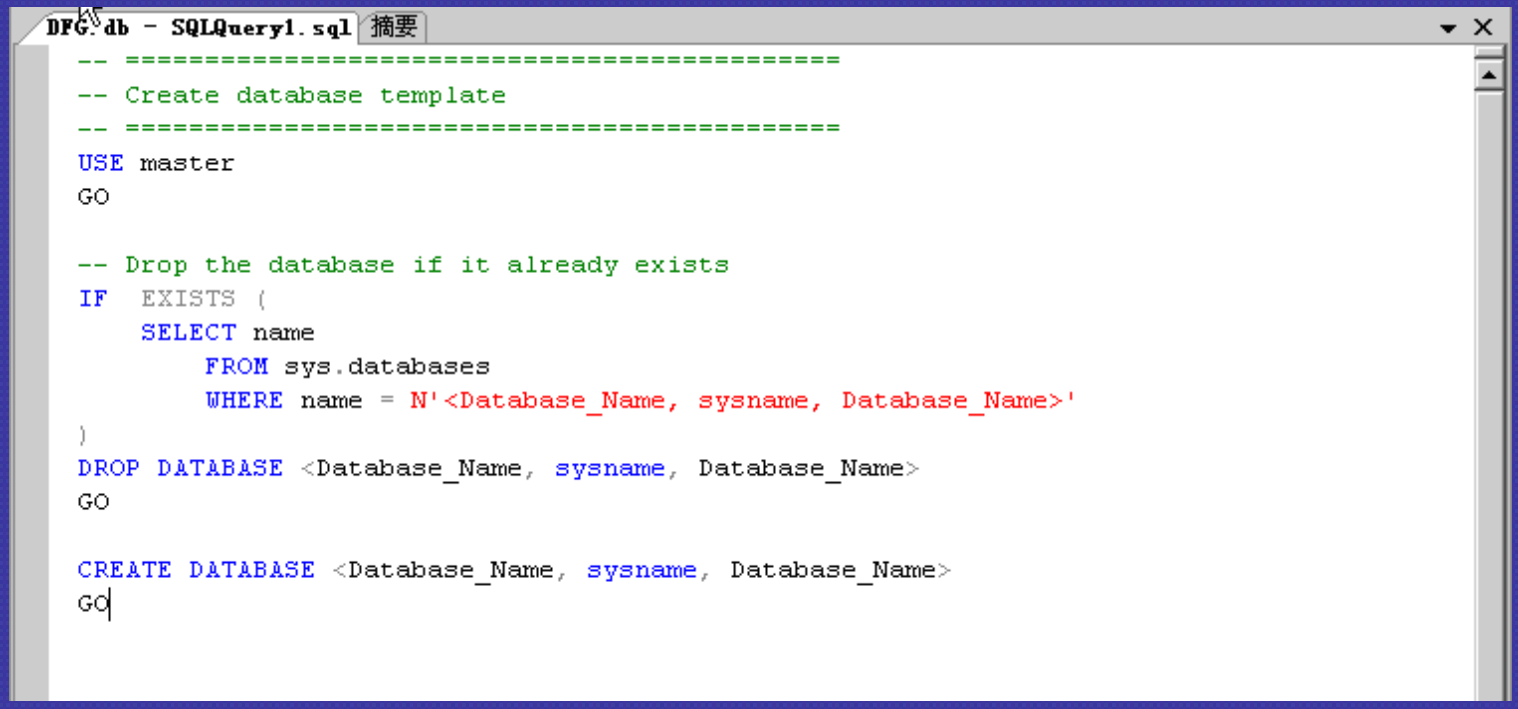


图4-2 创建数据库的SQL语言模板

4.2.1 创建数据库

1、使用模板创建数据库

- Transact-SQL语言使用CREATE DATABASE命令来创建数据库。该命令的语法如下：

CREATE DATABASE database_name

[ON [PRIMARY] [<filespec> [，…n] [，<filegroupspec> [，…n]]]

[LOG ON {<filespec> [，…n]}]

[FOR RESTORE]

<filespec>::=（[NAME=logical_file_name，]

FILENAME='os_file_name'

[，SIZE=size]

[，MAXSIZE={max_size|UNLIMITED}]

[，FILEGROWTH=growth_increment]） [，…n]

<filegroupspec>::=FILEGROUP filegroup_name <filespec> [，…n]

4.2.1 创建数据库

1、使用模板创建数据库

各参数说明如下：

- database_name：数据库的名称，最长为128个字符。
- PRIMARY：该选项是一个关键字，指定主文件组中的文件。
- LOG ON：指明事务日志文件的明确定义。
- NAME：指定数据库的逻辑名称，这是在SQL Server系统中使用的名称，是数据库在SQL Server中的标识符。
- FILENAME：指定数据库所在文件的操作系统文件名称和路径，该操作系统文件名和NAME的逻辑名称一一对应。
- SIZE：指定数据库的初始容量大小。
- MAXSIZE：指定操作系统文件可以增长到的最大尺寸。如果没有指定，则文件可以不断增长直到充满磁盘。
- FILEGROWTH：指定文件每次增加容量的大小，当指定数据为0时，表示文件不增长。

4.2.1　创建数据库

2、使用SQL Server管理平台创建数据库

（1）在SQL Server管理平台中，在数据库文件夹或其下属任一用户数据库图标上右击，从弹出的快捷菜单中选择新建数据库选项，出现如图4-3所示的对话框。

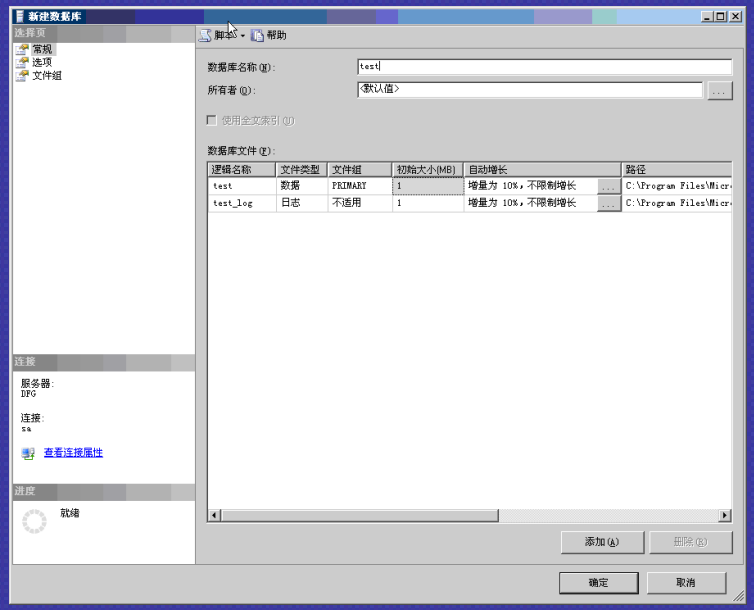


图4-3　创建数据库对话框

4. 2. 1　创建数据库

2、使用SQL Server管理平台创建数据库

（2）在选项页框中，如图4-4所示，可设置数据库的排序规则，恢复模式，兼容级别以及其他一些选项的设置。

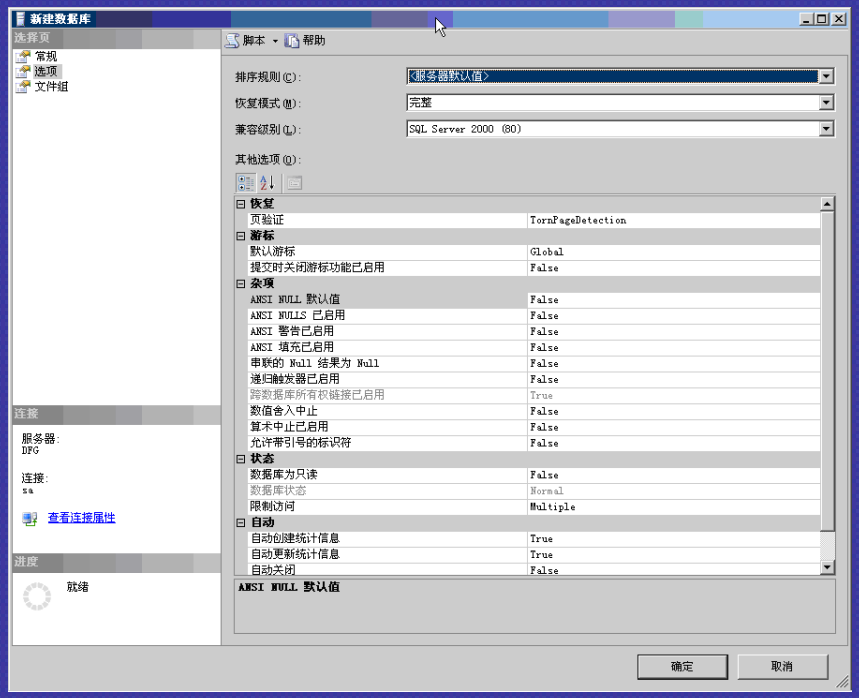


图4-4　创建数据库的选项页框

4.2.1　创建数据库

2、使用SQL Server管理平台创建数据库

（3）在文件组页框中，如图4-5所示，可设置或添加数据库文件和文件组的属性，如是否只读，是否为默认值等。

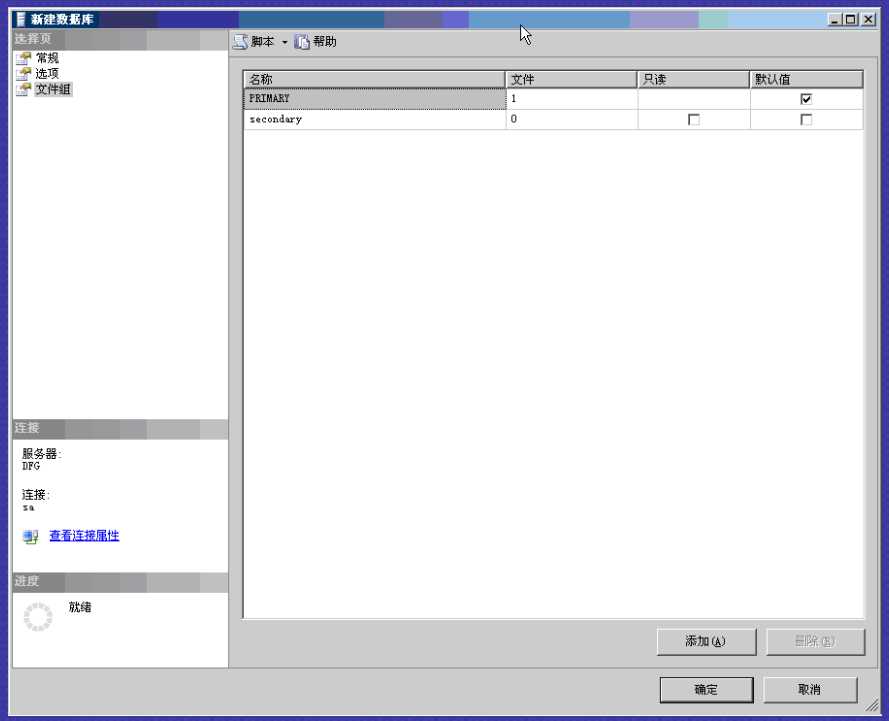


图4-5　创建数据库的文件组页框

4.2.1 创建数据库

例4-1 创建了一个Test数据库，该数据库的主数据文件逻辑名称为Test_data，物理文件名为Test.mdf，初始大小为10MB，最大尺寸为无限大，增长速度为10%；数据库的日志文件逻辑名称为Test_log，物理文件名为Test.ldf，初始大小为1MB，最大尺寸为5MB，增长速度为1MB。

程序清单如下：

```
CREATE DATABASE test
 ON  PRIMARY
( NAME = 'test',
FILENAME='C:\Program Files\Microsoft SQL Server\MSSQL\data\test.mdf',
SIZE=10240KB,
MAXSIZE = UNLIMITED,
FILEGROWTH = 10%)
 LOG ON
( NAME='test_log',
FILENAME='C:\Program Files\Microsoft SQL Server\MSSQL\data\test_log.ldf',
SIZE=1024KB,
MAXSIZE = 5120KB,
FILEGROWTH = 1024KB )
GO
```

4.2.2 修改数据库

1.利用SQL Server管理平台修改数据库

在SQL Server管理平台中，右击所要修改的数据库，从弹出的快捷菜单中选择“属性”选项，出现如图4-6所示的数据库属性设置对话框。可以看到，修改或查看数据库属性时，属性页框比创建数据库时多了两个，即选项和权限页框。

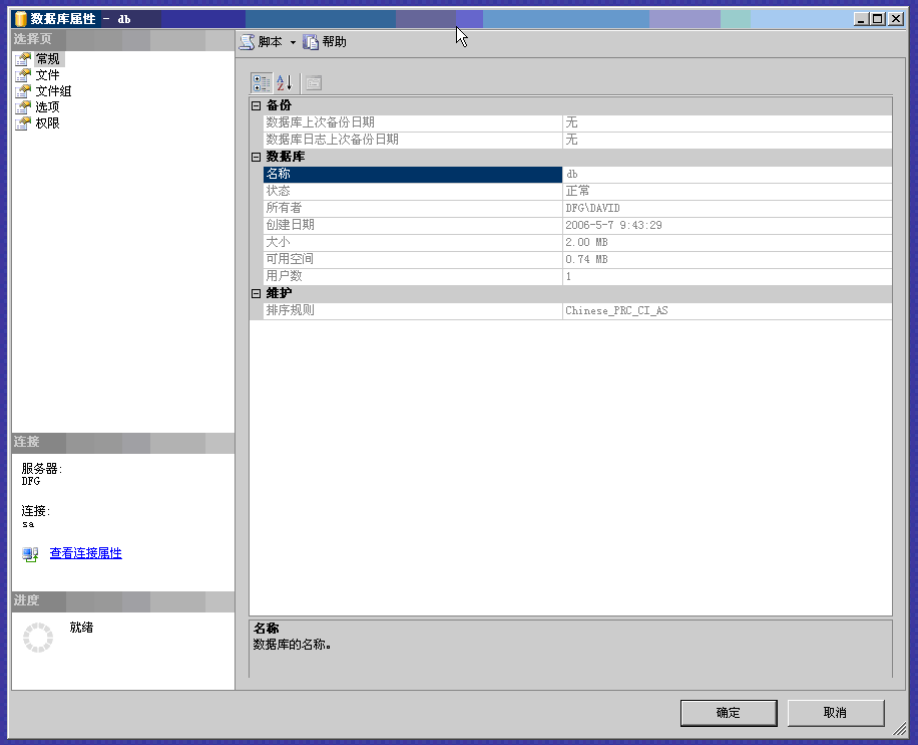


图4-6 数据库属性设置对话框

4.2.2 修改数据库

1.利用SQL Server管理平台修改数据库

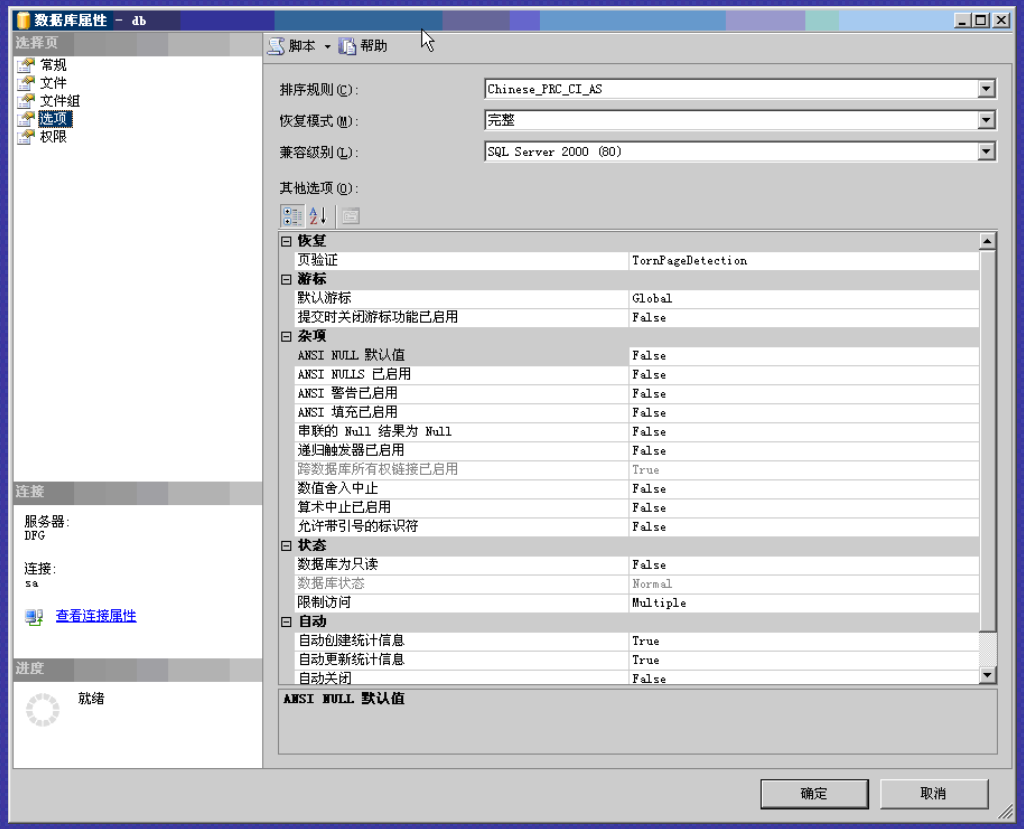


图4-7 数据库选项设置对话框

4.2.2 修改数据库

1.利用SQL Server管理平台修改数据库

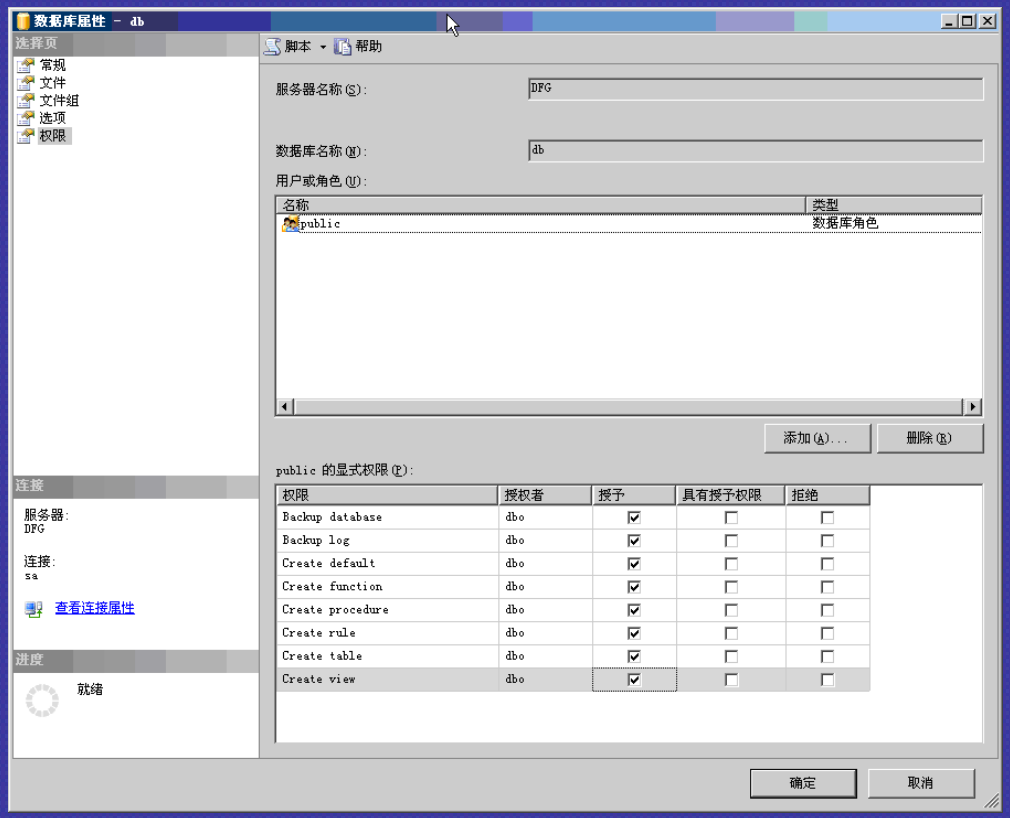


图4-8 数据库权限设置对话框

4.2.2 修改数据库

2. 使用ALTER DATABASE语句修改数据库

- ALTER DATABASE语法形式如下：

```
Alter database databasename
{add file<filespec>[,…n] [to filegroup filegroupname]
|add log file <filespec>[,…n]
|remove file logical_file_name [with delete]
|modify file <filespec>
|modify name=new_databasename
|add filegroup filegroup_name
|remove filegroup filegroup_name
|modify filegroup filegroup_name
{filegroup_property|name=new_filegroup_name}}
```

图4-6　数据库属性设置对话框

4.2.2 修改数据库

2. 使用ALTER DATABASE语句修改数据库

•例4-3　将两个数据文件和一个事务日志文件添加到test数据库中。

程序清单如下：

```
ALTER DATABASE Test
ADD FILE
(NAME = Test1, FILENAME='c:\Program Files\Microsoft SQL Server\MSSQL\Data\test1.ndf', SIZE = 5MB, MAXSIZE = 100MB, FILEGROWTH = 5MB),
(NAME = Test2, FILENAME='c:\Program Files\Microsoft SQL Server\MSSQL\Data\test2.ndf', SIZE = 3MB, MAXSIZE = 10MB, FILEGROWTH = 1MB)
GO
ALTER DATABASE Test
 ADD LOG FILE ( NAME = testlog1, FILENAME='c:\Program Files\Microsoft SQL Server\MSSQL\Data\testlog1.ldf',  SIZE = 5MB,  MAXSIZE = 100MB, FILEGROWTH = 5MB)
GO
```

4.2.3 删除数据库

1.利用SQL Server管理平台删除数据库

在SQL Server管理平台中，右击所要删除的数据库，从弹出的快捷菜单中选择“删除”选项即可删除数据库。系统会弹出确认是否要删除数据库对话框，如图4-9所示，单击“确定”按钮则删除该数据库。

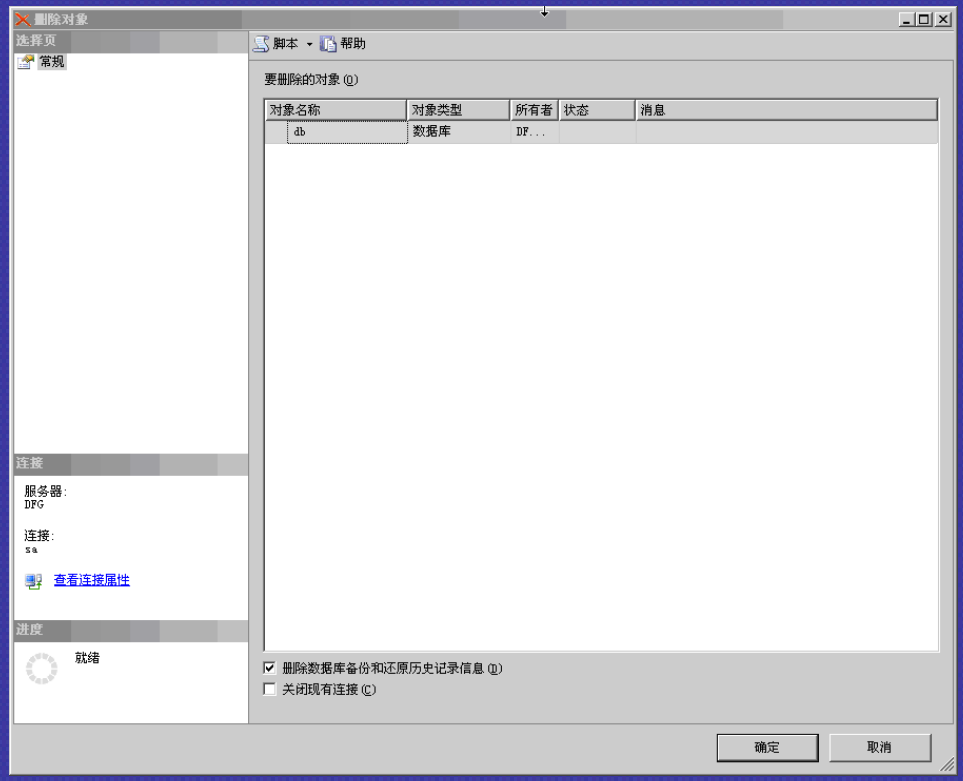


图4-9　确认删除数据库对话框

4.2.3 删除数据库

2.利用Drop语句修改数据库

• Drop语句可以从SQL Server中一次删除一个或多个数据库。其语法如下：

```
Drop database database_name[,…n]
```

•例4-4　删除创建的数据库Test。

程序清单如下：

```
drop database Test
```

图4-6　数据库属性设置对话框

4.3 数据库备份

4.3.1 备份概述

4.3.2 创建备份设备

4.3.3 备份的执行

4.3.1 备份概述

- **Microsoft SQL Server 2005**提供了高性能的备份和还原机制。数据库备份可以创建备份完成时数据库内存在的数据的副本，这个副本能在遇到故障时恢复数据库。这些故障包括：媒体故障，硬件故障（例如，磁盘驱动器损坏或服务器报废），用户操作错误（例如，误删除了某个表），自然灾害等。此外，数据库备份对于例行的工作（例如，将数据库从一台服务器复制到另一台服务器、设置数据库镜像、政府机构文件归档和灾难恢复）也很有用。
- 对**SQL Server**数据库或事务日志进行备份时，数据库备份记录了在进行备份这一操作时数据库中所有数据的状态，以便在数据库遭到破坏时能够及时地将其恢复。**SQL Server**备份数据库是动态的，在进行数据库备份时，**SQL Server**允许其他用户继续对数据库进行操作。执行备份操作必须拥有对数据库备份的权限许可，**SQL Server**只允许系统管理员、数据库所有者和数据库备份执行者备份数据库。备份是数据库系统管理的一项重要内容，也是系统管理员的日常工作。

4.3.1 备份概述

- **SQL Server 2005**提供了四种不同的备份方式，它们分别为：

（1）完整备份和完整差异备份

（2）部分备份和部分差异备份

（3）事务日志备份

（4）数据库文件和文件组备份。

4.3.2 创建备份设备

- 备份或还原操作中使用的磁带机或磁盘驱动器称为“备份设备”。
- 在创建备份时，必须选择要将数据写入的备份设备。**Microsoft SQL Server 2005** 可以将数据库、事务日志和文件备份到磁盘和磁带设备上。

4.3.2 创建备份设备

1. 使用SQL Server 管理平台创建备份设备

在**SQL Server** 管理平台中，选择想要创建备份设备的服务器，打开服务器对象文件夹，在备份设备图标上右击，从弹出的快捷菜单中选择“新建备份设备”选项，如图**4-10**所示。然后弹出备份设备对话框，如图**4-11**所示。

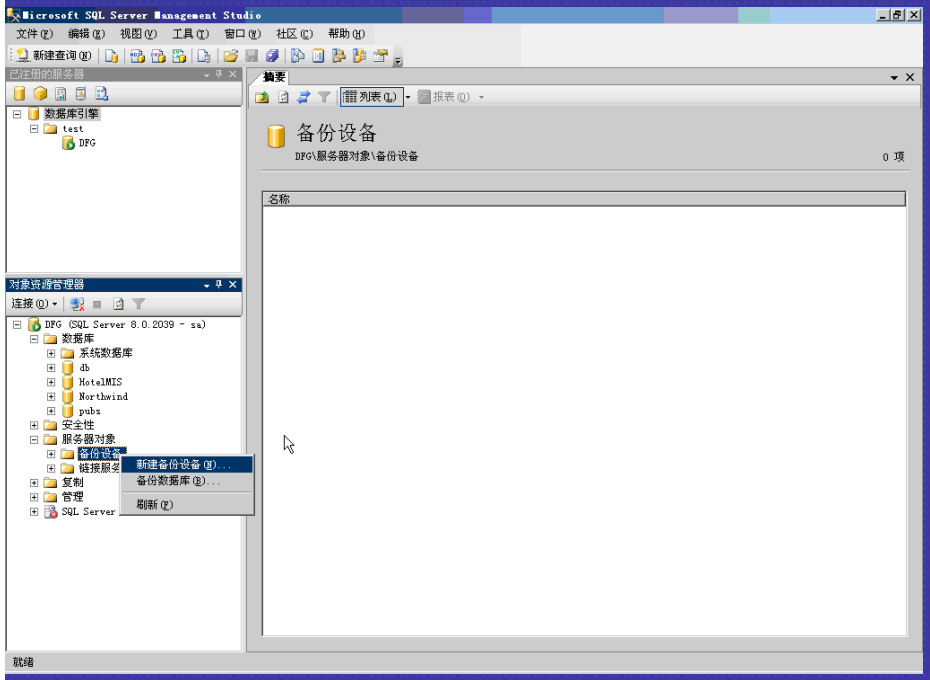


图4-10　使用SQL Server 管理平台创建备份设备

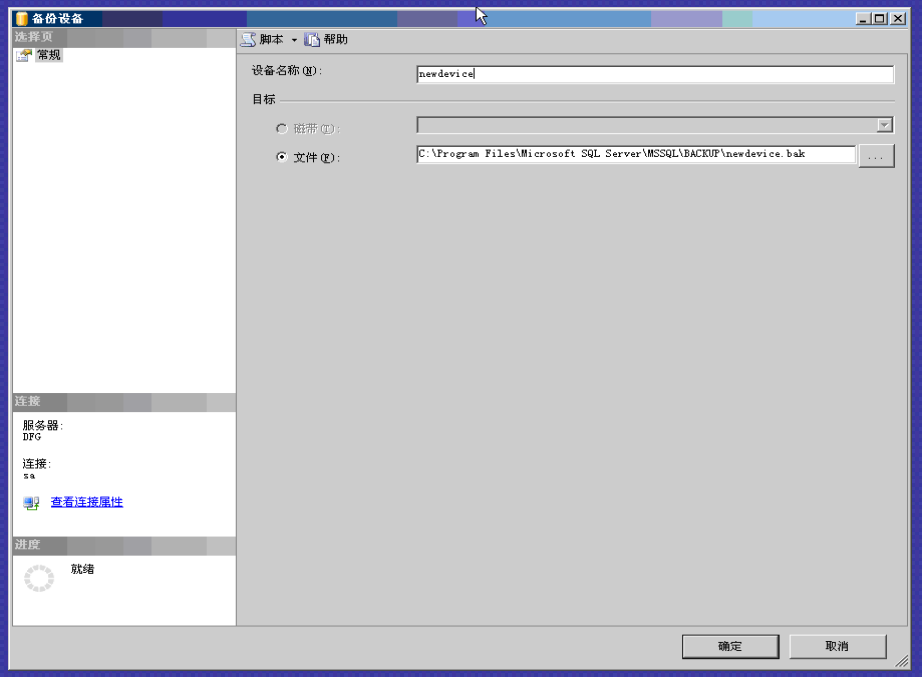


图4-11　输入备份设备属性对话框

4.3.2 创建备份设备

2. 使用系统存储过程创建备份设备

在**SQL Server** 中，可以使用**sp_addumpdevice**语句创建备份设备，其语法形式如下：

sp_addumpdevice {'device_type'}

[,'logical_name'][,'physical_name'][,{{controller_type|'device_status'}}]

其中，**device_type**表示设备类型，其值可为**disk**和**tape**。**logical_name**表示设备的逻辑名称。**physical_name**表示设备的实际名称。**controller_type**和**device_status**可以不必输入。

- 例**4-5**　在磁盘上创建了一个备份设备。

程序清单如下：

```
use master
exec sp_addumpdevice 'disk', 'test_backup', 'C:\Program Files\Microsoft SQL Server\MSSQL\BACKUP\test_backup.bak'
```

4.3.3 备份的执行

1.使用SQL Server 管理平台进行备份

（1）在SQL Server 管理平台中，打开数据库文件夹，右击所要进行备份的数据库图标，在弹出的快捷菜单中选择“任务”选项，再选择备份数据库，如图4-12所示。

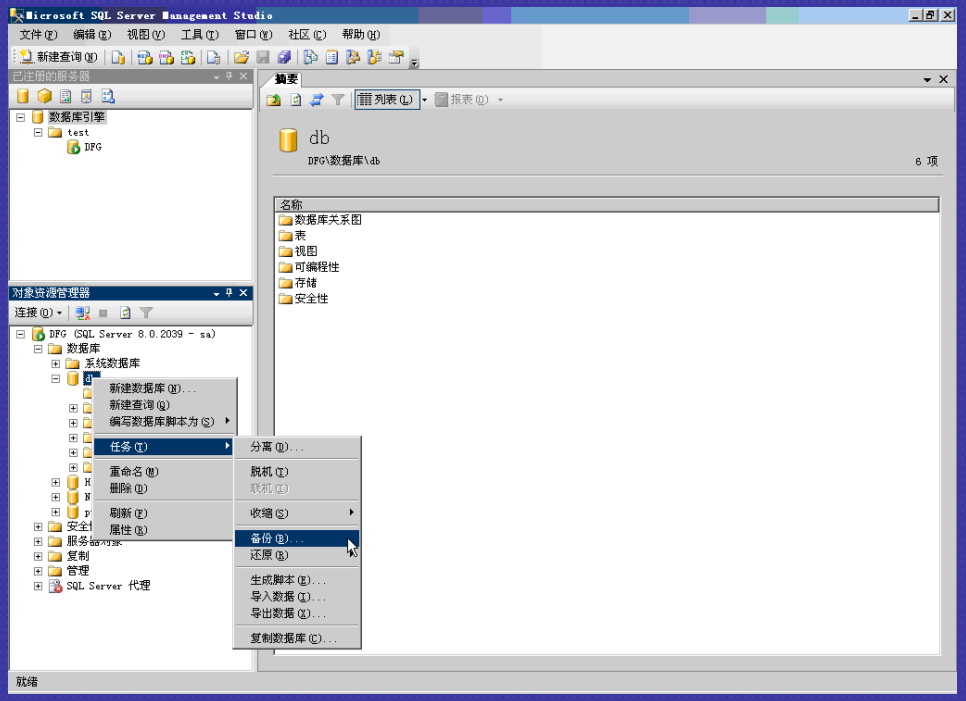


图4-12　SQL Server备份操作窗口

4.3.3 备份的执行

1.使用SQL Server 管理平台进行备份

（2）出现SQL Server备份对话框，如图4-13所示。图4-13中有两个页框，即“常规”和“选项”页框。

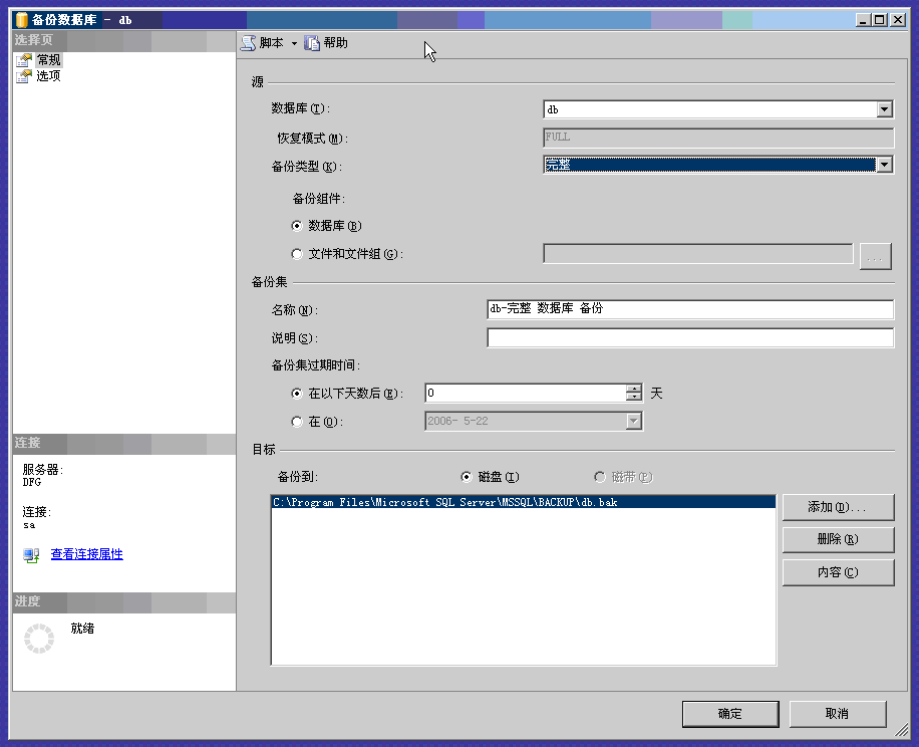


图4-13　SQL Server备份对话框

4.3.3 备份的执行

1.使用SQL Server 管理平台进行备份

（3）单击“添加”按钮可以选择将备份添加备份文件还是设备，如图4-14所示。

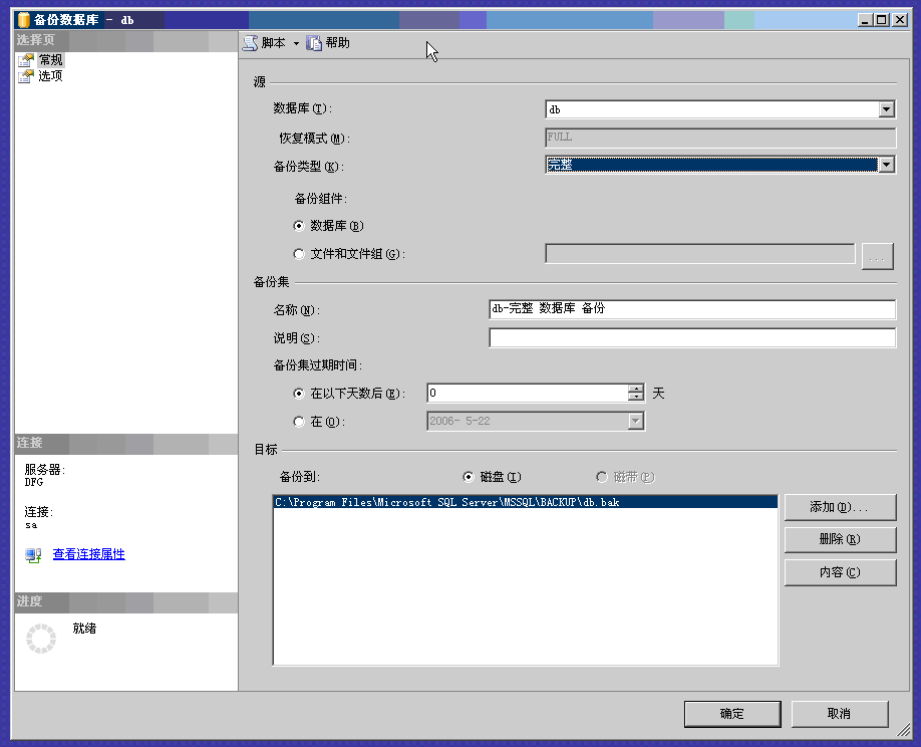


图4-13　SQL Server备份对话框

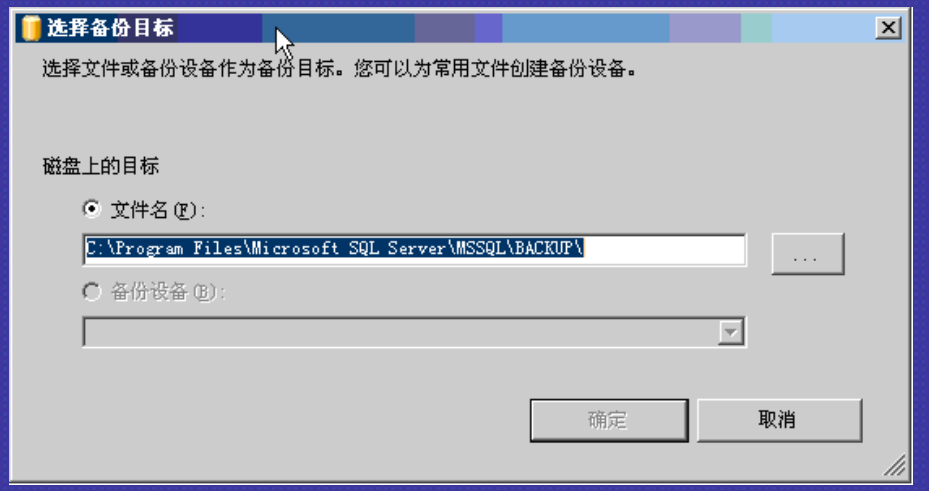


图4-14　选择备份目的对话框

4.3.3 备份的执行

1.使用SQL Server 管理平台进行备份

（4）备份数据库的选项页框

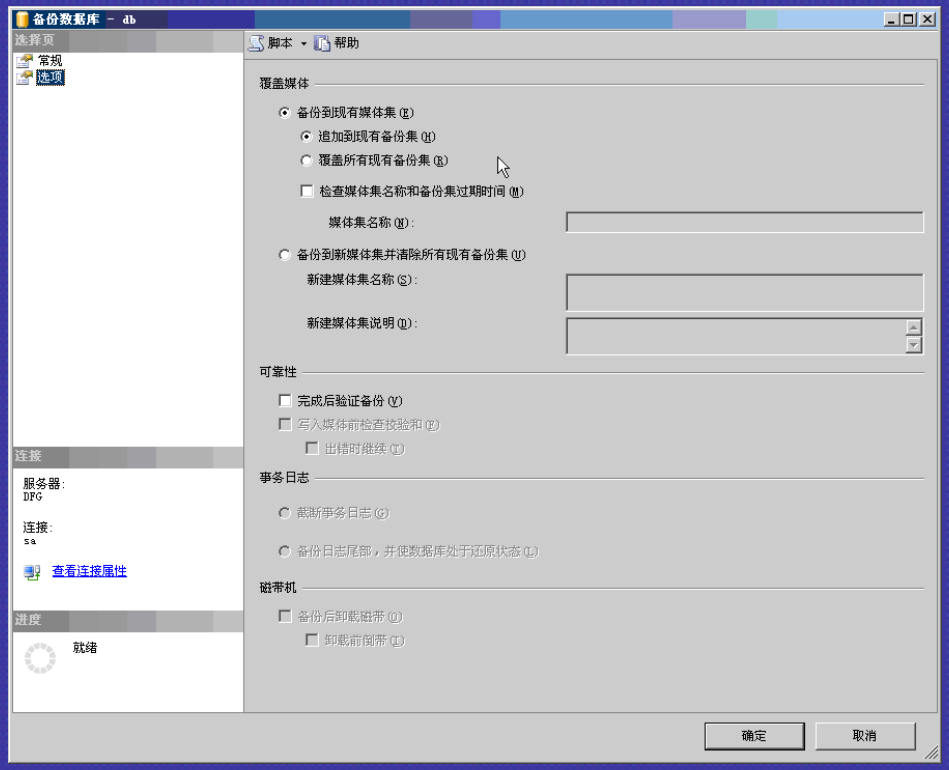


图4-15　备份数据库的选项页框

4.3.3 备份的执行

2.使用备份向导

（1）在SQL Server管理平台中，点击视图菜单中的模板资源管理器。

（2）模板资源管理器中的模板是分组列出的。展开“backup”，再双击“backup database”。 在“连接到数据库引擎”对话框中，填写连接信息，再单击“连接”。此时将打开一个新查询编辑器窗口，其中包含“备份数据库”模板的内容，如图4-16所示。

DFG.db - SQLQuery3.sql 摘要

```
-- ============================
-- Backup Database Template
-- ============================
BACKUP DATABASE <Database_Name, sysname, Database_Name>
    TO  DISK = N'<Backup_Path,,C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Backup\>
WITH
    NOFORMAT,
    NOINIT,
    NAME = N'<Database_Name, sysname, Database_Name>-Full Database Backup',
    SKIP,
    STATS = 10;
GO
```

图4-16　备份数据库模板

4.3.3 备份的执行

2.使用备份向导

（3）按照backup database的语法规则，书写数据库备份的sql语句，完成后执行此语句，即可完成数据库备份的操作。 Backup语句的语法形式如下：

Backup database
{database_name|@database_name_var}
to
<backup_file>[,…n]
[with
[[,]format]
[[,]{init|noinit}]
[[,]restart]
]
<backup_file>::={backup_file_name|@backup_file_evar}|{disk|tape}
={temp_file_name|@temp_file_name_evar}

4.3.3 备份的执行

2.使用备份向导

例4-7　将数据库db备份到备份设备db.bak上，使用WITH FORMAT子句初始化备份设备。

程序清单如下：

```
BACKUP DATABASE db
TO DISK=' C:\Program Files\Microsoft SQL Server\MSSQL\BACKUP\db.bak'
WITH FORMAT
```

4.4 还原数据库

4.4.1 还原概述

4.4.2 还原数据库

4.4.1 还原概述

数据库备份后，一旦系统发生崩溃或者执行了错误的数据库操作，就可以从备份文件中还原数据库。数据库还原是指将数据库备份加载到系统中的过程。系统在还原数据库的过程中，自动执行安全性检查、重建数据库结构以及完成填写数据库内容。安全性检查是还原数据库时必不可少的操作。这种检查可以防止偶然使用了错误的数据库备份文件或者不兼容的数据库备份覆盖已经存在的数据库。SQL Server还原数据库时，根据数据库备份文件自动创建数据库结构，并且还原数据库中的数据。

4.4.2 还原数据库

•由于数据库的还原操作是静态的，所以在还原数据库时，必须限制用户对该数据库进行其他操作，因而在还原数据库之前，首先要设置数据库访问属性。如图4-17所示。

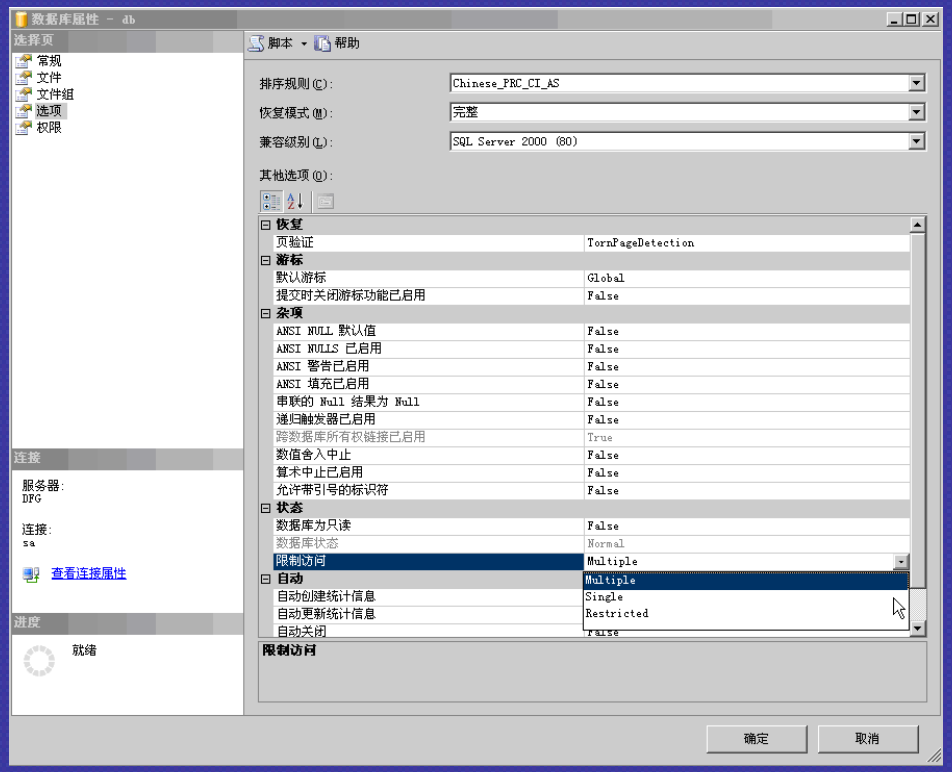


图4-17 设置数据库访问属性对话框

4.4.2 还原数据库

1. 使用SQL Server管理平台还原数据库

（1）打开SQL Server管理平台，在数据库上单击鼠标右键，从弹出的快捷菜单中选择“任务”选项，再选择“还原数据库”命令，弹出还原数据库对话框，如图4-18所示。

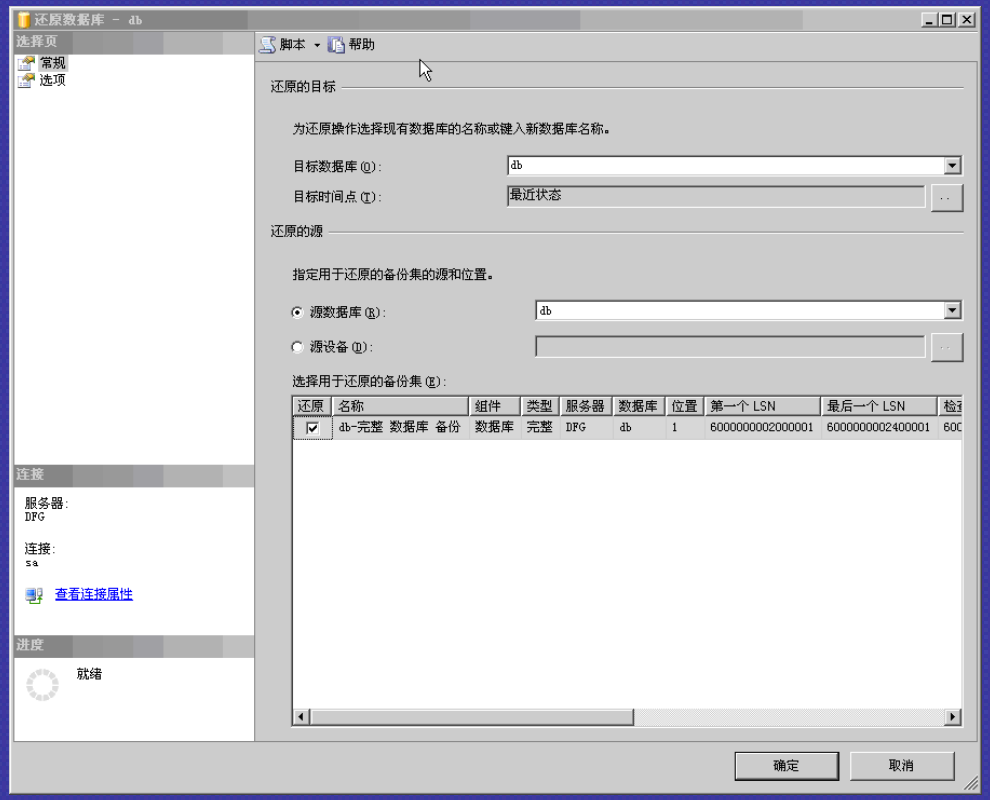


图4-18 还原数据库对话框

4.4.2　还原数据库

1. 使用SQL Server管理平台还原数据库

（2）选中“选项”页框，进行其他选项的设置，如图4-19所示

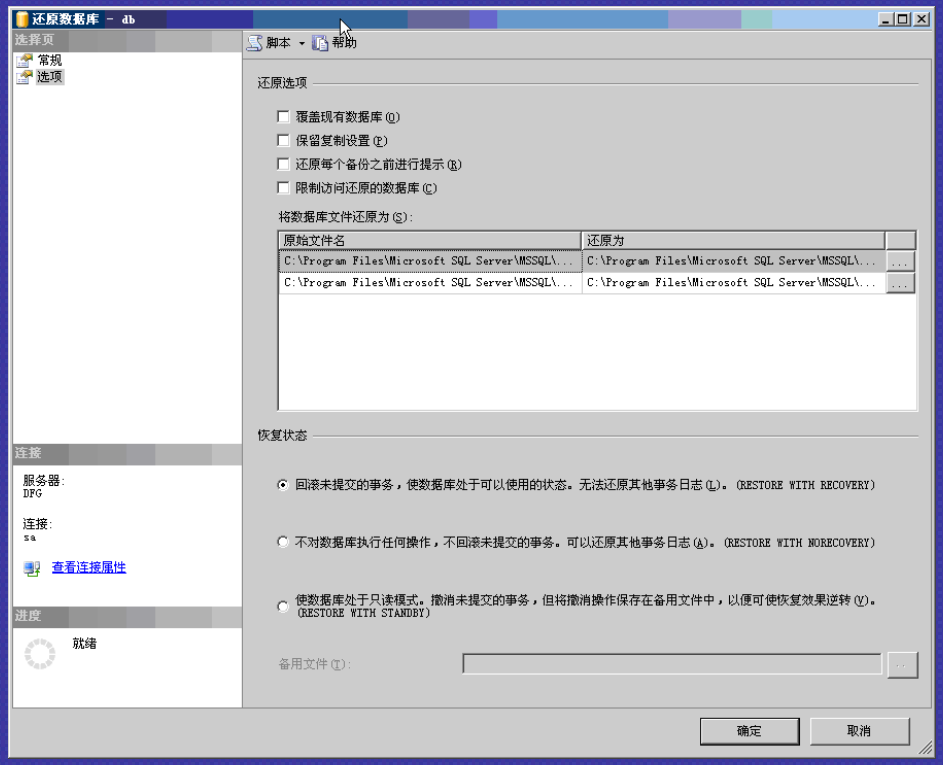


图4-19　还原数据库对话框-选项页框

4.4.2 还原数据库

2.使用Transact-SQL语句还原数据库

•Transact-SQL提供了restore语句还原数据库，其语法形式如下：

```
Restore  database
[from <backup_device[],…n>]
[with
[[,]file=file_number]
[[,]move ‘logical_file_name’ to ‘operating_system_file_name’]
[[,]replace]
[[,]{norecovery|recovery|standby=undo_file_name}]
]
<backup_device>::={{backup_device_name|@backup_device_name_eva
r}
|{disk|tape|pipe}
={temp_backup_device|@temp_backup_device_var}
```

4.4.2　还原数据库

2.使用Transact-SQL语句还原数据库

例4-8　从backup_company备份设备中还原数据库company。

程序清单如下：

```
use master
restore database company
from backup_company
```

4.5 数据库维护

4.5.1 数据库维护概述

4.5.2 数据库维护计划向导

4.5.1 数据库维护概述

- 数据库创建后，所有的对象和数据均已添加且都在使用中，需要对其进行维护，数据库的维护可以使它保持运行的最佳状态。例如，定期备份数据库是很重要的。创建数据库维护计划可以让**SQL Server**自动而有效地维护数据库，为系统管理员节省大量时间，也可以防止延误数据库的维护工作。
- 在 **SQL Server 2005**数据库引擎中，维护计划可创建一个作业以按预定间隔自动执行这些维护任务。
- 维护计划向导可以用于设置核心维护任务，从而确保数据库执行良好，做到定期备份数据库以防系统出现故障，对数据库实施不一致性检查。维护计划向导可创建一个或多个 **SQL Server** 代理作业，代理作业将按照计划的间隔自动执行这些维护任务。

4.5.2 数据库维护计划向导

•利用数据库的维护计划向导可以方便地设置数据库的核心维护任务，以便于定期地执行这些任务，其创建数据库维护计划的步骤如下：

（1）展开服务器。

（2）展开“管理”文件夹，右键单击“维护计划”，然后选择“维护计划向导”。

（3）单击“下一步”按钮，则会出现选择目标服务器对话框，如图4-21所示，选择服务器名称，身份验证模式。

（4）单击“下一步”按钮，就会出现“选择维护任务”对话框，如图4-22所示。

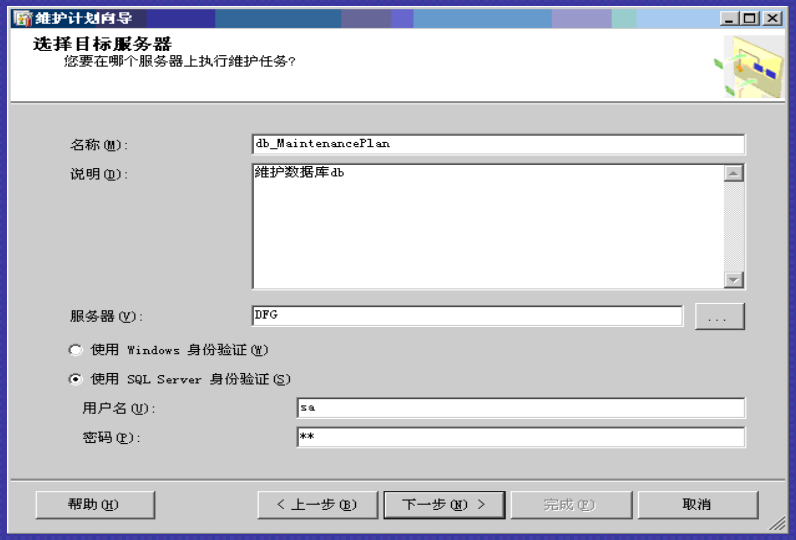


图4-21 选择目标服务器对话框

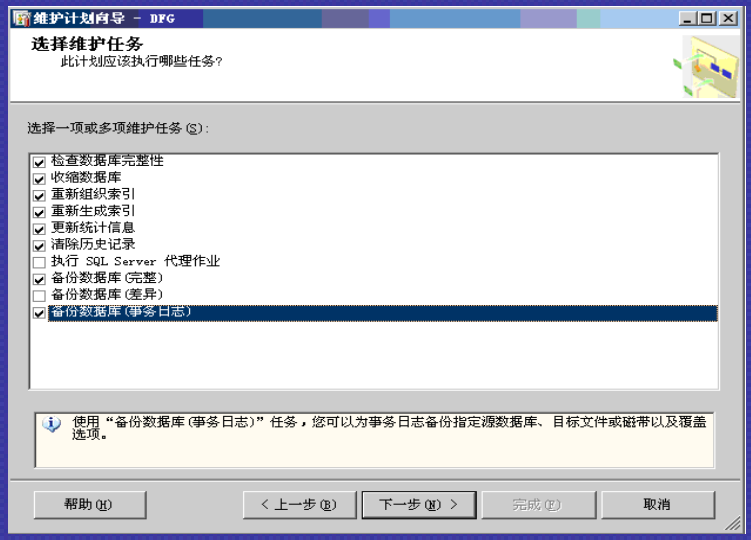


图4-22 选择维护任务对话框

4.5.2 数据库维护计划向导

（5）单击“下一步”按钮，则会出现“选择维护任务顺序”对话框，在此对话框中可以改变执行任务的顺序，如图4-23所示。

（6）单击“下一步”按钮，如图4-24所示，出现定义“数据库检查完整性”对话框，选择进行维护的数据库。

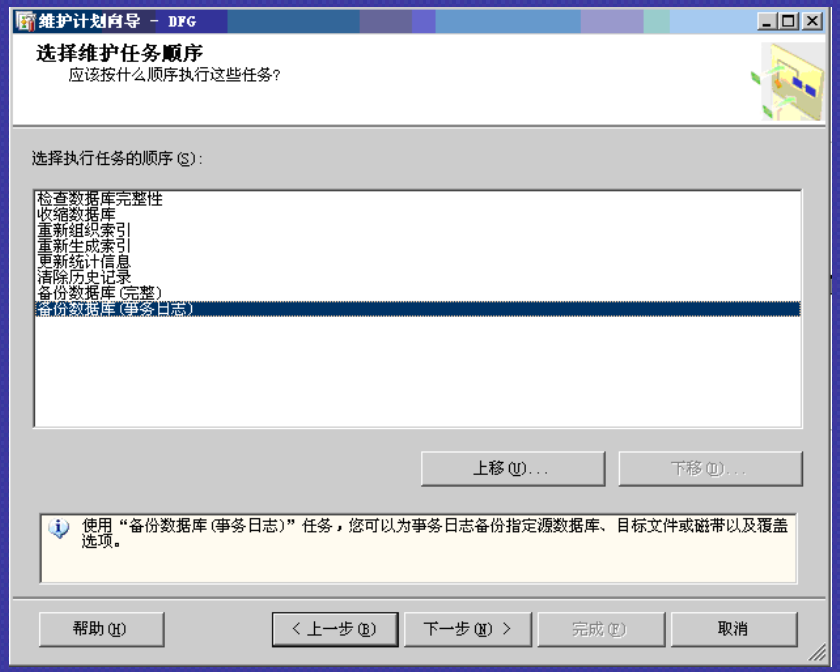


图4-23 选择维护任务顺序对话框

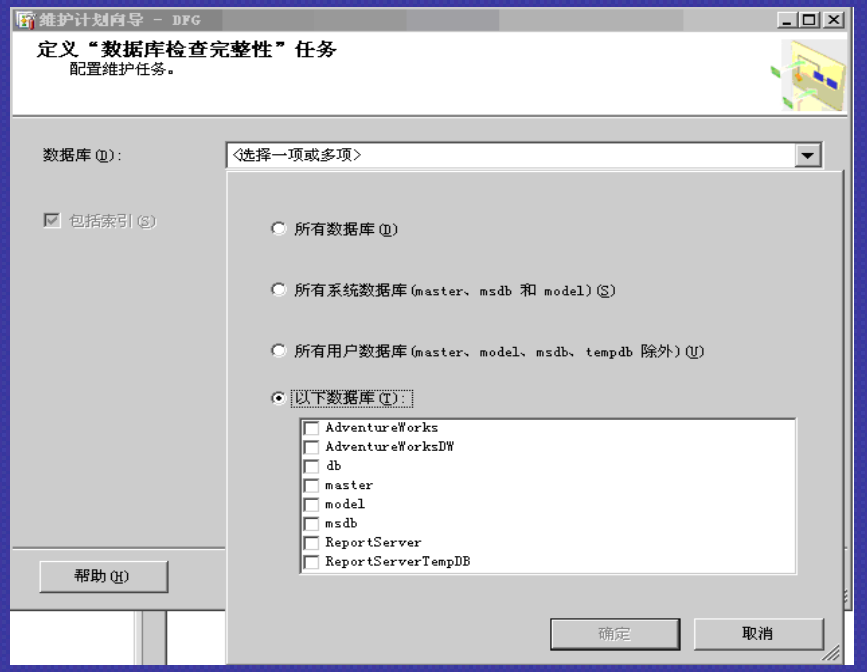


图4-24 定义“数据库检查完整性”对话框

4.5.2 数据库维护计划向导

（7）单击“下一步”按钮，出现定义“收缩数据库”对话框，如图4-25所示。可以确定收缩数据库的条件，收缩后保留的可用空间，释放后的空间存放位置。

（8）单击“下一步”按钮，则会出现定义“重新组织索引”任务对话框，如图4-26所示。其中可以选择数据库，数据库对象类型（表，视图），具体的表或视图。

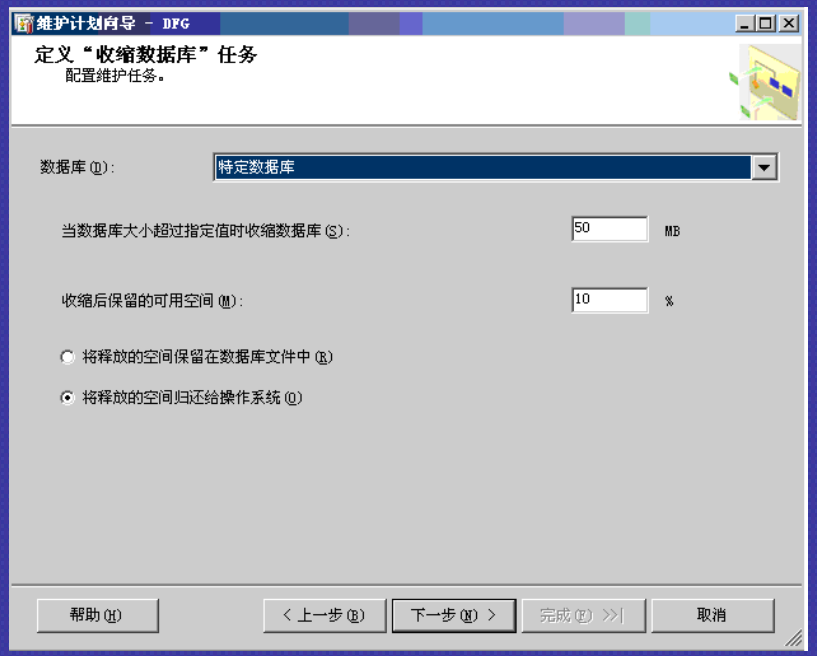


图4-25 定义“收缩数据库”对话框

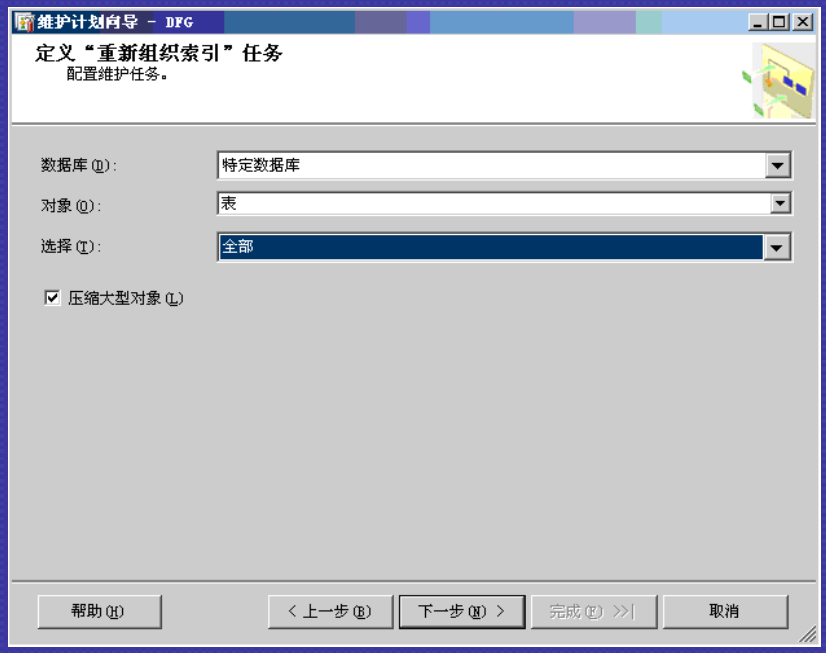


图4-26 定义“重新组织索引”对话框

4.5.2 数据库维护计划向导

（9）单击“下一步”按钮，则会出现定义“重新生成索引”任务对话框，如图4-27所示。其中可以设定可用空间选项及高级选项。

（10）单击“下一步”按钮，则出现定义“更新统计信息”任务对话框，如图4-28所示。

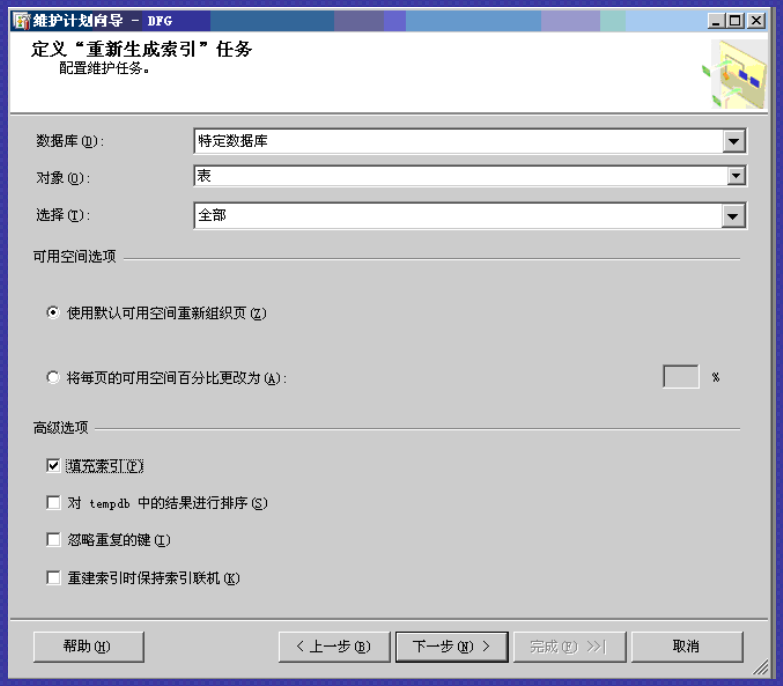


图4-27 定义“重新生成索引”对话框

图4-28 定义“重新统计信息”对话框

4.5.2 数据库维护计划向导

（11）单击“下一步”按钮，出现定义“清除历史记录”任务对话框，如图4-29所示。其中可以选择要删除的历史数据。

（12）单击“下一步”按钮，出现定义“备份数据库（完整）”任务对话框，如图4-30所示。其中可以设定备份组件，备份目标等。

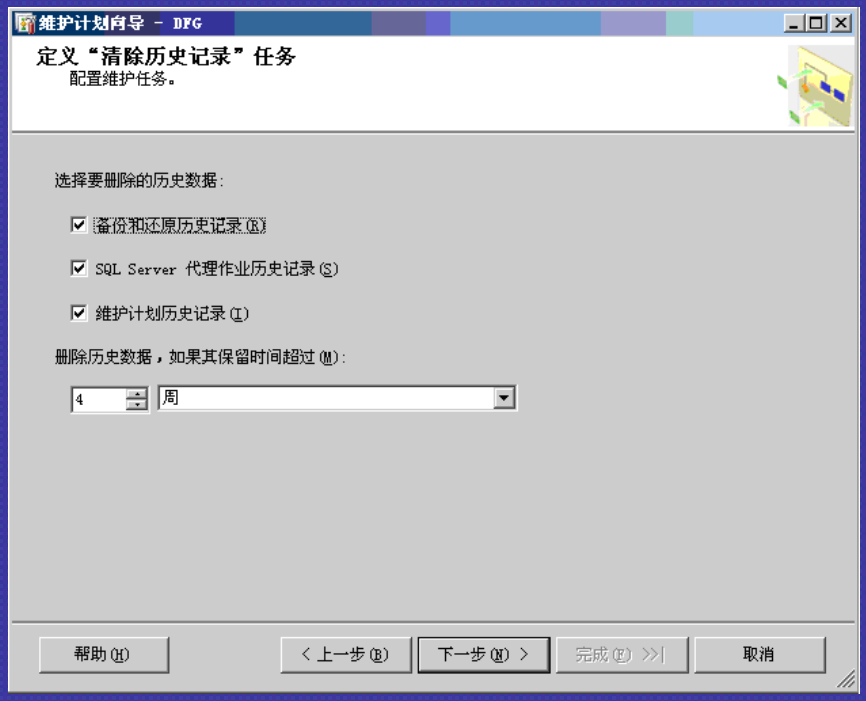


图4-29 定义“清除历史记录”任务对话框

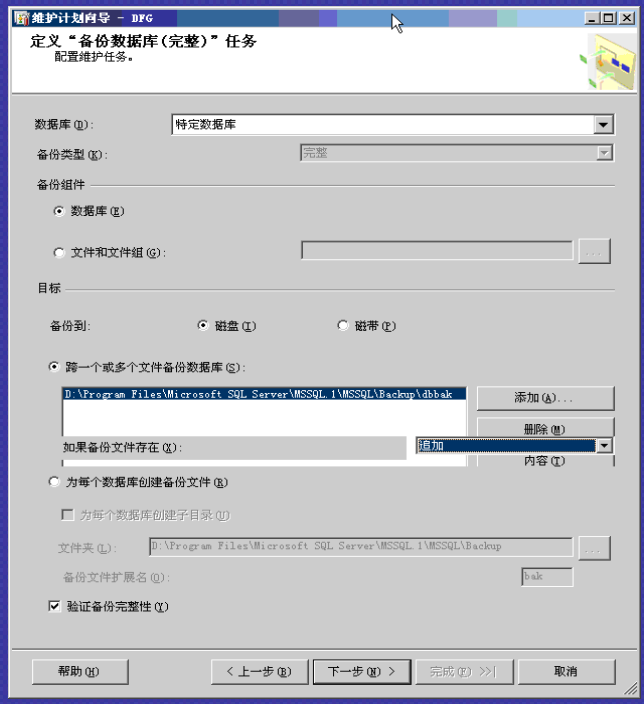


图4-30 定义“备份数据库（完整）”任务对话框

4.5.2 数据库维护计划向导

（13）单击“下一步”按钮，出现定义“备份数据库（事务日志）”任务对话框，如图4-31所示。

（14）单击“下一步”按钮，出现“选择计划属性”对话框，如图4-32所示。其中可设定执行的频率和起止时间。

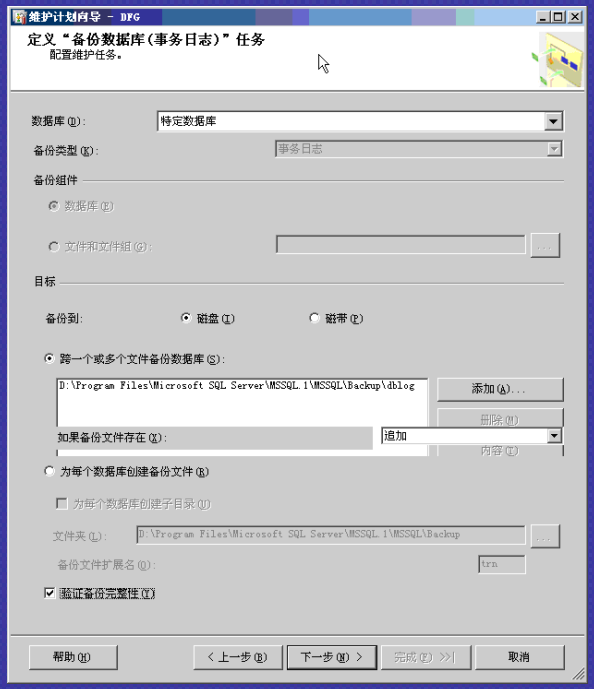


图4-31　定义“备份数据库（事务日志）”任务对话框

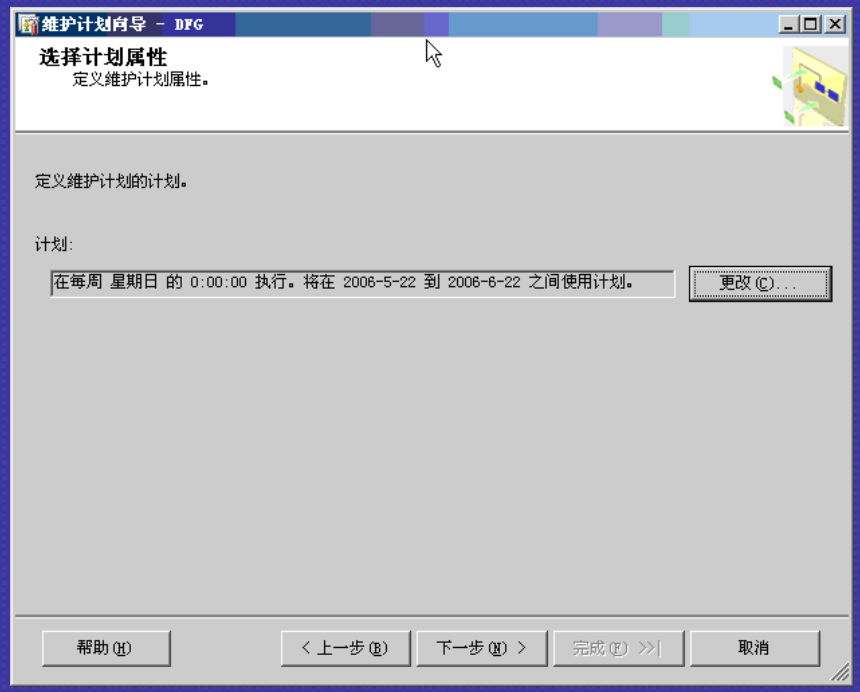


图4-32　“选择计划属性”对话框

4.5.2 数据库维护计划向导

（15）单击“下一步”按钮，出现“选择报告”选项对话框，如图4-33所示。（16）单击“下一步”按钮，出现“完成该向导”对话框，如图4-34所示。

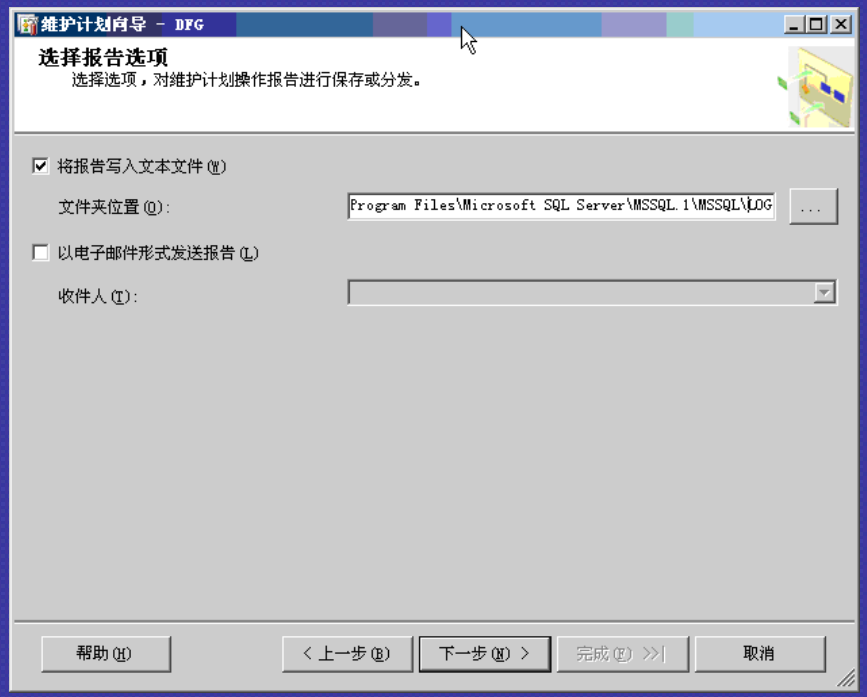


图4-33　维护计划操作报告保存或分发对话框　　图4-34　指定数据库维护计划名称对话框

4.5.2 数据库维护计划向导

（17）执行的进度及结果如图**4-35**所示。

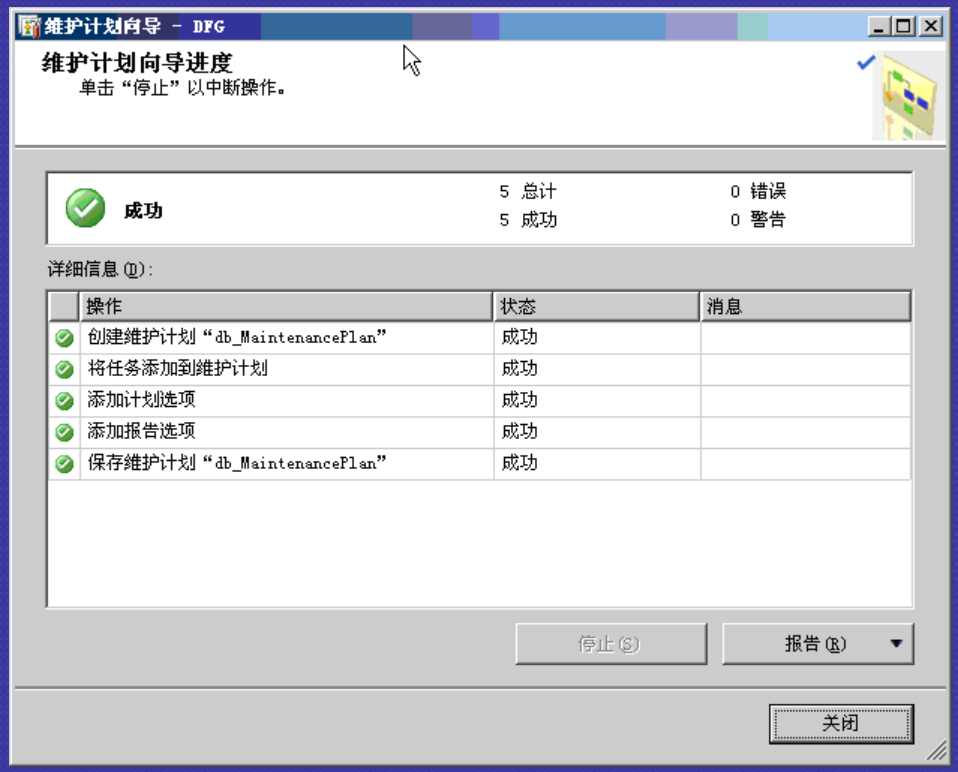


图4-35 “维护计划向导进度”对话框

第5章 表的操作与管理

5.1 数据类型

- 在SQL Server2005 中，每个列、局部变量、表达式和参数都有其各自的数据类型。指定对象的数据类型相当于定义了该对象的四个特性：

（1）对象所含的数据类型，如字符、整数或二进制数。

（2）所存储值的长度或它的大小。

（3）数字精度（仅用于数字数据类型）。

（4）小数位数（仅用于数字数据类型）。

- SQL Server提供系统数据类型集，定义了可与SQL Server一起使用的所有数据类型；另外用户还可以使用Transact-SQL或.NET框架定义自己的数据类型，它是系统提供的数据类型的别名。每个表可以定义至多250个字段，除文本和图像数据类型外，每个记录的最大长度限制为1962个字节。

5.1.1 系统数据类型

1、精确数字类型；
2、近似数字类型；
3、日期和时间类型；
4、字符数据类型；
5、Unicode 字符数据类型；
6、二进制字符数据类型；
7、其它数据类型。

5.1.1 系统数据类型

1、精确数字类型

(1)精确数字类型包括：

- 整数类型;
- Bit（位类型）;
- Decimal和Numeric（数值类型）;
- Money和SmallMoney（货币类型）。

5.1.1 系统数据类型

1、精确数字类型

(2)整数类型：

整数类型是最常用的数据类型之一，它主要用来存储数值，可以直接进行数据运算，而不必使用函数转换。整数类型包括以下四类：

（1）Bigint：Bigint数据类型可以存储从 (-9223372036854775808) 到 (9223372036854775807) 范围之间的所有整型数据。每个Bigint数据类型值存储在8个字节中。

（2）Int（Integer）：Int（或integer）数据类型可以存储从（-2147483648）到（2147483647）范围之间的所有正负整数。每个Int数据类型值存储在4个字节中。

（3）Smallint：可以存储从（-32768）到32767范围之间的所有正负整数。每个smallint类型的数据占用2个字节的存储空间。

（4）Tinyint：可以存储从0~255范围之间的所有正整数。每个tinyint类型的数据占用1个字节的存储空间。

5.1.1 系统数据类型

1、精确数字类型

(3)位数据类型

•Bit称为位数据类型，其数据有两种取值：0和1，长度为1字节。在输入0以外的其他值时，系统均把它们当1看待。这种数据类型常作为逻辑变量使用，用来表示真、假或是、否等二值选择。

(4)Decimal数据类型和Numeric数据类型

Decimal数据类型和Numeric数据类型完全相同，它们可以提供小数所需要的实际存储空间，但也有一定的限制，可以用2~17个字节来存储-1038+1到1038-1之间的固定精度和小数位的数字。也可以将其写为Decimal（p，s）的形式，p和s确定了精确的总位数和小数位。其中p表示可供存储的值的总位数，默认设置为18；s表示小数点后的位数，默认设置为0。例如：decimal（10，5），表示共有10位数，其中整数5位，小数5位。

5.1.1 系统数据类型

1、精确数字类型

(5)货币数据类型

货币数据类型包括Money和SmallMoney两种：

（1）Money：用于存储货币值，存储在money数据类型中的数值以一个正数部分和一个小数部分存储在两个4字节的整型值中，存储范围为（-922337213685477580 8）到（922337213685477580 7），精确到货币单位的千分之十。

（2）Smallmoney：与money数据类型类似，但范围比money数据类型小，其存储范围为-2147483468到2147483467之间，精确到货币单位的千分之十。

当为money或smallmoney的表输入数据时，必须在有效位置前面加一个货币单位符号。

5.1.1 系统数据类型

2、近似数字类型

•近似数字类型包括**Real**和**Float**两大类。

（**1**）**Real**：可以存储正的或者负的十进制数值，最大可以有**7**位精确位数。它的存储范围从**-3.40E-38~3.40E+38**。每个**Real** 类型的数据占用**4**个字节的存储空间。

（**2**）**Float**：可以精确到第**15**位小数，其范围从**-1.79E-308~1.79E+308**。如果不指定**Float** 数据类型的长度，它占用**8**个字节的存储空间。**Float**数据类型也可以写为**Float**（**n**）的形式，**n**指定**Float**数据的精度，**n**为**1~15**之间的整数值。当**n**取**1~7**时，实际上是定义了一个**Real** 类型的数据，系统用**4**个字节存储它；当**n**取**8~15**时，系统认为其是**Float**类型，用**8**个字节存储它。

5.1.1 系统数据类型

3. 日期和时间数据类型

（1）**Datetime**：用于存储日期和时间的结合体，它可以存储从公元**1753**年**1**月**1**日零时起~公元**9999**年**12**月**31**日**23**时**59**分**59**秒之间的所有日期和时间，其精确度可达三百分之一秒，即**3.33**毫秒。**Datetime**数据类型所占用的存储空间为**8**个字节，其中前**4**个字节用于存储基于**1900**年**1**月**1**日之前或者之后日期数，数值分正负，负数存储的数值代表在基数日期之前的日期，正数表示基数日期之后的日期，时间以子夜后的毫秒存储在后面的**4**个字节中。当存储**Datetime**数据类型时，默认的格式是**MM DD YYYY hh:mm A.M./P.M**，当插入数据或者在其他地方使用**Datetime**类型时，需要用单引号把它括起来。默认的时间日期是**January 1,1900 12:00 A.M**。可以接受的输入格式如下：**Jan 4 1999**、**JAN 4 1999**、**January 4 1999**、**Jan 1999 4**、**1999 4 Jan**和**1999 Jan 4**。

（2）**Smalldatetime**：与**Datetime**数据类型类似，但其日期时间范围较小，它存储从**1900**年**1**月**1**日~**2079**年**6**月**6**日内的日期。**SmallDatetime**数据类型使用**4**个字节存储数据，**SQL Server 2000**用**2**个字节存储日期**1900**年**1**月**1**日以后的天数，时间以子夜后的分钟数形式存储在另外两个字节中，**SmallDatetime**的精度为**1**分钟。

5.1.1 系统数据类型

4. 字符数据类型

•字符数据类型也是**SQL Server**中最常用的数据类型之一，它可以用来存储各种字母、数字符号和特殊符号。在使用字符数据类型时，需要在其前后加上英文单引号或者双引号。

（**1**）**Char**：其定义形式为**Char**（**n**），当用**Char**数据类型存储数据时，每个字符和符号占用一个字节的存储空间。**n**表示所有字符所占的存储空间，**n**的取值为**1~8000**。若不指定**n**值，系统默认**n**的值为**1**。若输入数据的字符串长度小于**n**，则系统自动在其后添加空格来填满设定好的空间；若输入的数据过长，将会截掉其超出部分。如果定义了一个**Char**数据类型，而且允许该列为空，则该字段被当作**Varchar**来处理。

（**2**）**Varchar**：其定义形式为**Varchar**（**n**）。用**Char**数据类型可以存储长达**255**个字符的可变长度字符串，和**Char**类型不同的是**Varchar**类型的存储空间是根据存储在表的每一列值的字符数变化的。例如定义**Varchar**（**20**），则它对应的字段最多可以存储**20**个字符，但是在每一列的长度达到**20**字节之前系统不会在其后添加空格来填满设定好的空间，因此使用**Varchar**类型可以节省空间。

（**3**）**Text**：用于存储文本数据，其容量理论上为**1~231-1**（**2**，**147**，**483**，**647**）个字节，但实际应用时要根据硬盘的存储空间而定。

5.1.1 系统数据类型

5. Unicode 字符数据类型

•Unicode 字符数据类型包括Nchar、Nvarchar、Ntext三种：

（1）Nchar：其定义形式为Nchar（n）。它与Char数据类型类似，不同的是Nchar数据类型n的取值为1~4000。Nchar数据类型采用Unicode标准字符集，Unicode标准用两个字节为一个存储单位，其一个存储单位的容纳量就大大增加了，可以将全世界的语言文字都囊括在内，在一个数据列中就可以同时出现中文、英文、法文等，而不会出现编码冲突。

（2）Nvarchar：其定义形式Nvarchar（n）。它与Varchchar数据类型相似，Nvarchar数据类型也采用Unicode标准字符集，n的取值范围为1~4000。

（3）Ntext：与Text数据类型类似，存储在其中的数据通常是直接能输出到显示设备上的字符，显示设备可以是显示器、窗口或者打印机。Ntext数据类型采用Unicode标准字符集，因此其理论上的容量为230-1（1，073，741，823）个字节。

5.1.1 系统数据类型

6. 二进制字符数据类型

- 二进制数据类型包括**Binary**、**Varbinary**、**Image**三种：

（1）**Binary**：其定义形式为**Binary**（n），数据的存储长度是固定的，即**n+4**个字节，当输入的二进制数据长度小于**n**时，余下部分填充**0**。二进制数据类型的最大长度（即**n**的最大值）为**8000**，常用于存储图像等数据。

（2）**Varbinary**：其定义形式为**Varbinary**（n），数据的存储长度是变化的，它为实际所输入数据的长度加上**4**字节。其他含义同**Binary**。

（3）**Image**：用于存储照片、目录图片或者图画，其理论容量为**231-1**（**2，147，483，647**）个字节。其存储数据的模式与**Text**数据类型相同，通常存储在**Image**字段中的数据不能直接用**Insert**语句直接输入。

5.1.1 系统数据类型

7. 其它数据类型

（1）**Sql_variant**：用于存储除文本、图形数据和**Timestamp**类型数据外的其他任何合法的**SQL Server**数据。此数据类型极大地方便了**SQL Server**的开发工作。

（2）**Table**：用于存储对表或者视图处理后的结果集。这种新的数据类型使得变量可以存储一个表，从而使函数或过程返回查询结果更加方便、快捷。

（3）**Timestamp**：亦称时间戳数据类型，它提供数据库范围内的惟一值，反应数据库中数据修改的相对顺序，相当于一个单调上升的计数器。当它所定义的列在更新或者插入数据行时，此列的值会被自动更新，一个计数值将自动地添加到此**Timestamp**数据列中。如果建立一个名为**“Timestamp”**的列，则该列的类型将自动设为**Timestamp**数据类型。

（4）**Uniqueidentifier**：用于存储一个**16**字节长的二进制数据类型，它是**SQL Server**根据计算机网络适配器地址和**CPU**时钟产生的全局惟一标识符代码（**Globally Unique Identifier**，简写为**GUID**）。此数字可以通过调用**SQL Server**的 **newid**（）函数获得，在全球各地的计算机经由此函数产生的数字不会相同。

（5）**XML**：可以存储**XML**数据的数据类型。利用它可以将**XML**实例存储在字段中或者**XML**类型的变量中。注意存储在**XML**中的数据不能超过**2GB**。

（6）**Cursor**：这是变量或存储过程**OUTPUT** 参数的一种数据类型，这些参数包含对游标的引用。使用 **Cursor** 数据类型创建的变量可以为空。注意：对于 **CREATE TABLE** 语句中的列，不能使用**Cursor** 数据类型。

5.1.2 自定义数据类型

- **SQL Server**允许用户自定义数据类型，用户自定义数据类型是建立在**SQL Server**系统数据类型基础上的，当用户定义一种数据类型时，需要指定该类型的名称、建立在其上的系统数据类型以及是否允许为空等。
- **SQL Server**为用户提供了两种方法来创建自定义数据类型：

（1）使用**SQL Server**管理平台创建用户自定义数据类型；

（**2**）利用系统存储过程创建用户自定义数据类型

5.1.2 自定义数据类型

（1）使用SQL Server管理平台创建用户自定义数据类型

在SQL Server管理平台中，打开指定的服务器和数据库项，如图5-1所示，选择并展开“程序→类型”项，接下来用右键单击“用户自定义数据类型”选项，从弹出的快捷菜单中选择“新建”命令，出现用户定义的数据类型属性对话框，如图5-2所示。

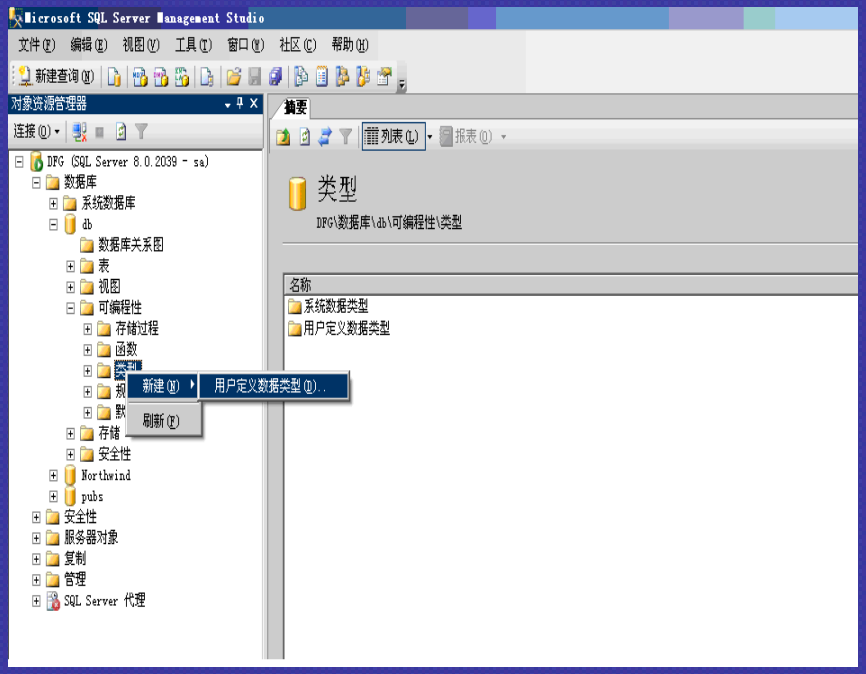


图5-1 打开用户定义的数据类型窗口

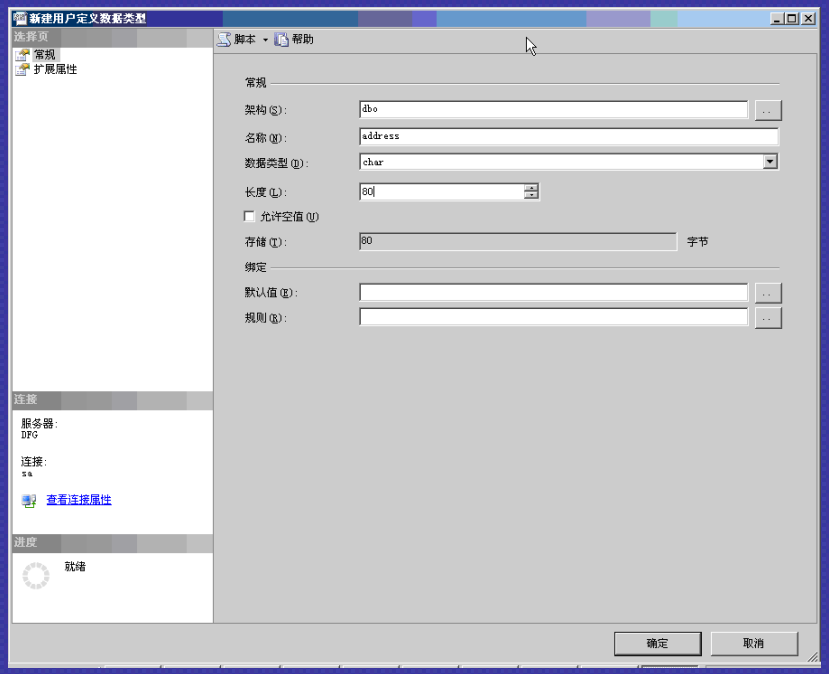


图5-2 用户定义的数据类型属性对话框

5.1.2 自定义数据类型

（2）利用系统存储过程创建用户自定义数据类型

- 系统存储过程sp_addtype为用户提供了用T_SQL语句创建自定义数据类型的途径，其语法形式如下：

```
sp_addtype [@typename=] type,
[@phystype=] system_data_type
[, [@nulltype=] 'null_type']
[, [@owner=] 'owner_name']
```

5.1.2 自定义数据类型

（2）利用系统存储过程创建用户自定义数据类型

•例5-1　自定义一个地址（address）数据类型。程序清单如下：

```
exec sp_addtype address, ‘varchar（80）’, ‘not null’
```

5.2 表操作

表是包含数据库中所有数据的数据库对象。表定义为列的集合，数据在表中是按行和列的格式组织排列的，每行代表惟一的一条记录，而每列代表记录中的一个域。

5.2.1 创建表

5.2.2 创建约束

5.2.3 修改表

5.2.4 查看表

5.2.5 删除表

5.2.1 创建表

1.利用SQL Server管理平台创建表

在SQL Server管理平台中，展开指定的服务器和数据库，打开想要创建新表的数据库，右击表对象，并从弹出的快捷菜单中选择“新建表”选项，如图5-4所示。在图5-4的对话框中，可以对表的结构进行更改，设置主键及字段属性，使用SQL Server管理平台可以非常直观地修改数据库结构和添加数据。在表中任意行上右击，则弹出一个快捷菜单，如图5-6所示。

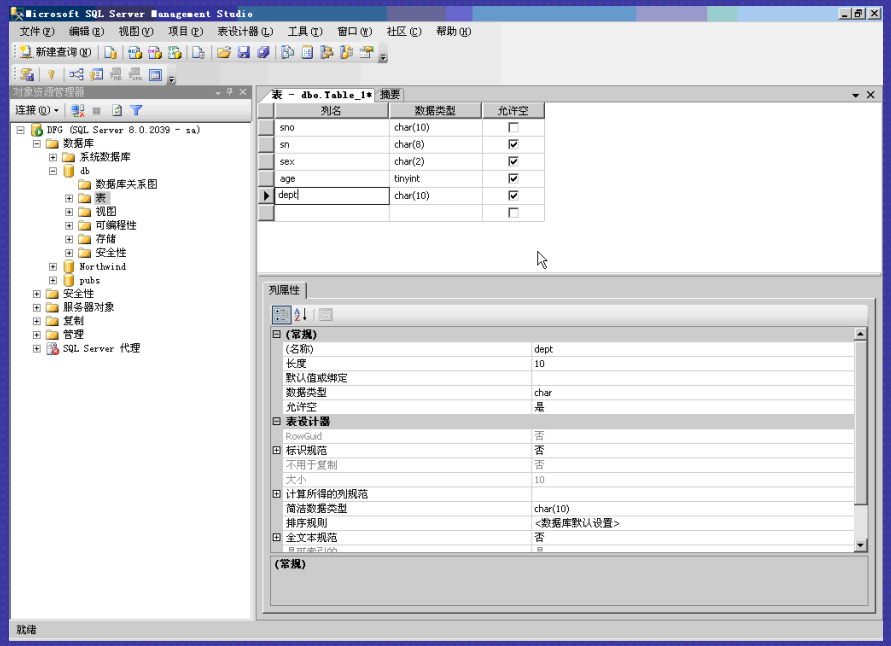


图5-4 新建表对话框

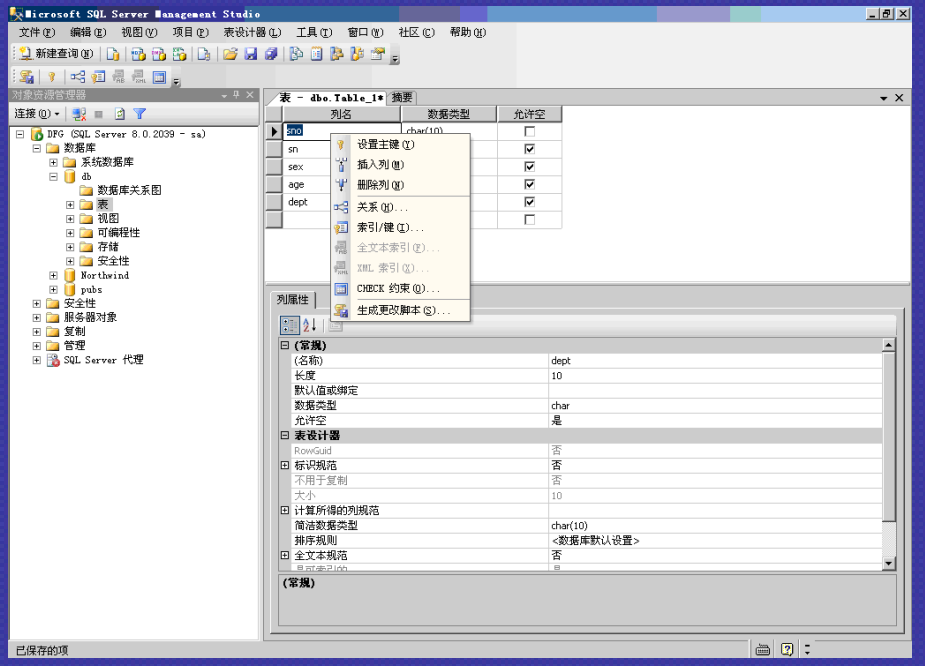


图5-6 设置字段属性对话框

5.2.1 创建表

2. 利用create命令创建表

- 使用create命令创建表非常灵活，它允许对表设置几种不同的选项，包括表名、存放位置和列的属性等。
- 其完整语法形式如下：

```
CREATE TABLE
  [database_name.[owner].|owner.]table_name
 （{<column_definition>|column_name AS computed_column_expression|
<table_constraint>}[，…n]）
        [ON{ filegroup|DEFAULT}]
[TEXTIMAGE_ON { filegroup|DEFAULT}]
        <column_definition>::={column_name data_type}
  [COLLATE <collation_name>]
  [[DEFAULT constant_expression]
   |[IDENTITY[（seed,increment ）[NOT FOR REPLICATION]]]]
     [ROWGUIDCOL]
  [<column_constraint>][...n]
```

5.2.1 创建表

2. 利用create命令创建表

其中，各参数的说明如下：

- database_name：用于指定所创建表的数据库名称。
- owner：用于指定新建表的所有者的用户名。
- table_name：用于指定新建表的名称。
- column_name：用于指定新建表的列名。
- computed_column_expression：用于指定计算列的列值表达式。
- ON {filegroup | DEFAULT}：用于指定存储表的文件组名。
- TEXTIMAGE_ON：用于指定 text、ntext 和 image 列的数据存储的文件组。
- data_type：用于指定列的数据类型。
- DEFAULT：用于指定列的默认值。
- constant_expression：用于指定列的默认值的常量表达式、可以为一个常量或NULL或系统函数。
- IDENTITY：用于将列指定为标识列。Seed：用于指定标识列的初始值。Increment：用于指定标识列的增量值。
- NOT FOR REPLICATION：用于指定列的IDENTITY属性，在把从其他表中复制的数据插入到表中时不发生作用，即不生成列值，使得复制的数据行保持原来的列值。
- ROWGUIDCOL：用于将列指定为全局惟一标识行号列（row global unique identifier column）。
- COLLATE：用于指定表的校验方式。
- column_constraint和table_constraint：用于指定列约束和表约束。

5.2.1 创建表

2. 利用**create**命令创建表

- 例5-3 创建了一个工人信息表，它包括工人编号、姓名、性别、出生日期、职位、工资和备注信息。
- SQL语句的程序清单如下：

```
CREATE TABLE worker
（number  char(8)  not null,
 name  char（8） NOT NULL,
 sex  char（2） NULL,
 birthday  datetime  null,
 job_title  varchar（10） null,
 salary  money  null,
memo  ntext  null
）
```

5.2.2 创建约束

- 约束是**SQL Server**提供的自动保持数据库完整性的一种方法，它通过限制字段中数据、记录中数据和表之间的数据来保证数据的完整性。在**SQL SERVER**中，对于基本表的约束分为列约束和表约束。
- 列约束是对某一个特定列的约束，包含在列定义中，直接跟在该列的其他定义之后，用空格分隔，不必指定列名；表约束与列定义相互独立，不包括在列定义中，通常用于对多个列一起进行约束，与列定义用’,’分隔，定义表约束时必须指出要约束的那些列的名称。
- 完整性约束的基本语法格式为：

  [CONSTRAINT constraint_name（约束名）**] <**约束类型**>**

  约束不指定名称时，系统会给定一个名称。
- 在**SQL Server 2005**中有**6**种约束：主键约束（**primary key constraint**）、惟一性约束（**unique constraint**）、检查约束（**check constraint**）、默认约束（**default constraint**）、外部键约束（**foreign key constraint**）和空值（**NULL**）约束。

5.2.2 创建约束

1.主键（PRIMARY KEY）约束

- PRIMARY KEY约束用于定义基本表的主键，它是惟一确定表中每一条记录的标识符，其值不能为NULL，也不能重复，以此来保证实体的完整性。PRIMARY KEY与UNIQUE约束类似，通过建立唯一索引来保证基本表在主键列取值的唯一性，但它们之间存在着很大的区别：

①在一个基本表中只能定义一个PRIMARY KEY约束，但可定义多个UNIQUE约束；

②对于指定为PRIMARY KEY的一个列或多个列的组合，其中任何一个列都不能出现空值，而对于UNIQUE所约束的唯一键，则允许为空。

- 注意：不能为同一个列或一组列既定义UNIQUE约束，又定义PRIMARY KEY约束。
- PRIMARY KEY既可用于列约束，也可用于表约束。

5.2.2 创建约束

1.主键（PRIMARY KEY）约束

主键的创建操作方法有两种：SQL Server管理平台操作法和Transact-SQL语句操作法。

（1） SQL Server管理平台操作法，如图5-7所示。

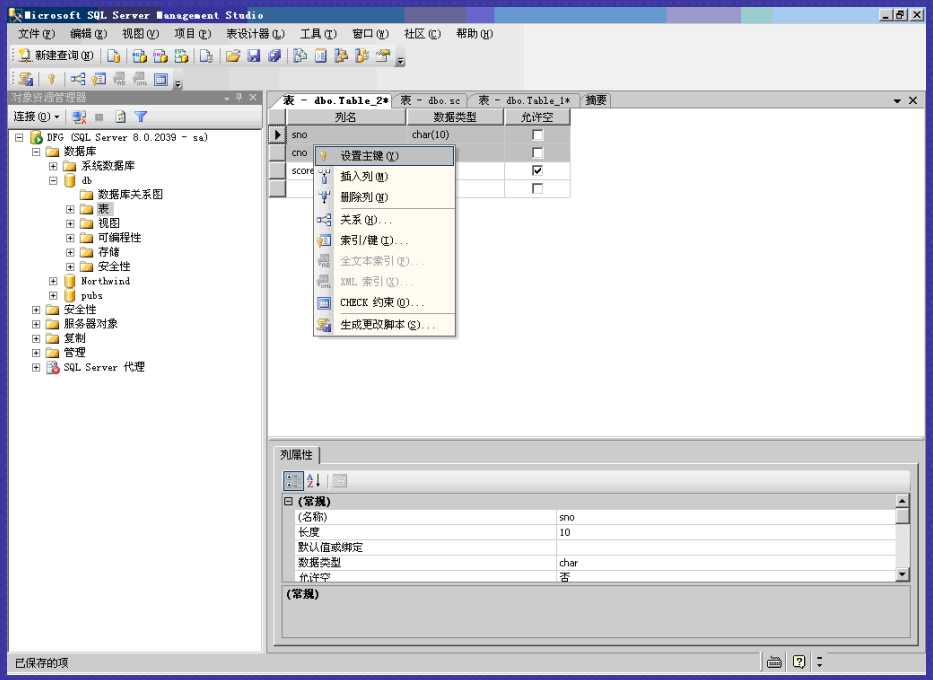


图5-7 选择多个字段共同做为主键对话框

5.2.2 创建约束

1.主键（PRIMARY KEY）约束

（2）使用Transact-SQL语句操作法设置主键约束，其语法形式如下：

CONSTRAINT constraint_name
PRIMARY KEY [CLUSTERED|NONCLUSTERED]
（column_name）

•PRIMARY KEY用于定义表约束时，即将某些列的组合定义为主键，其语法格式如下：

CONSTRAINT constraint_name
PRIMARY KEY [CLUSTERED|NONCLUSTERED]
（column_name[,…n]）

5.2.2 创建约束

1.主键（PRIMARY KEY）约束

例5-5 建立一个SC表，定义SNO，CNO共同组成SC的主键
程序清单如下：

```
CREATE TABLE SC
(SNO CHAR(5) NOT NULL,
CNO CHAR(5) NOT NULL,
SCORE NUMERIC(3),
CONSTRAINT SC_PRIM PRIMARY KEY(SNO,CNO))
```

5.2.2 创建约束

2. 惟一性约束

惟一性约束用于指定一个或者多个列的组合值具有惟一性，以防止在列中输入重复的值。定义了UNIQUE约束的那些列称为唯一键，系统自动为唯一键建立唯一索引，从而保证了唯一键的唯一性。

当使用惟一性约束时，需要考虑以下几个因素：

- 使用惟一性约束的字段允许为空值；
- 一个表中可以允许有多个惟一性约束；
- 可以把惟一性约束定义在多个字段上；
- 惟一性约束用于强制在指定字段上创建一个惟一性索引；
- 默认情况下，创建的索引类型为非聚集索引。

5.2.2 创建约束

2. 惟一性约束

创建惟一性约束的方法有两种：通过SQL Server管理平台可以完成创建和修改惟一性约束的操作；使用Transact-SQL语句完成惟一性约束的操作。

（1）通过SQL Server管理平台可以完成创建和修改惟一性约束的操作，如图5-8所示。

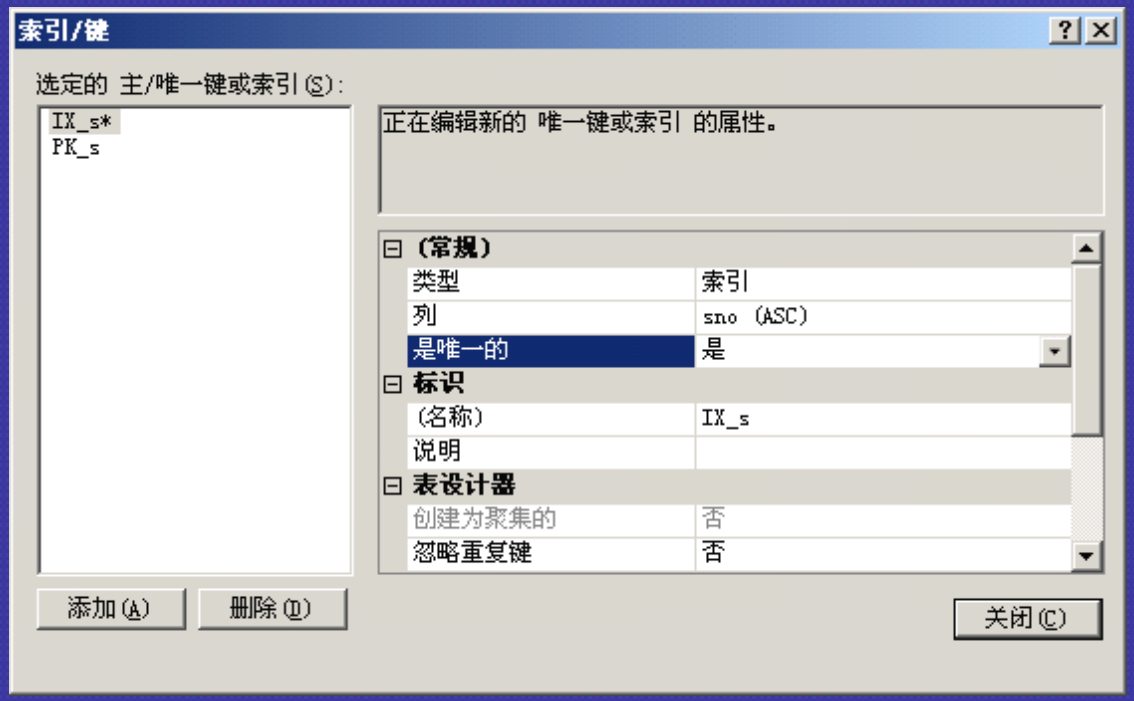


图5-8 创建惟一性约束对话框

5.2.2 创建约束

2. 惟一性约束

（2）使用Transact-SQL语句完成惟一性约束的操作，其语法形式如下：

```
CONSTRAINT constraint_name
 UNIQUE [CLUSTERED|NONCLUSTERED]
（column_name[,…n]）
```

例5-6　创建一个学生信息表，其中name字段具有惟一性。

程序清单如下：

```
Create table student（
id char（8）,
name char（10）,
sex char（2）,
 constraint pk_id primary key（id）,
 constraint uk_identity unique（name）
）
```

5.2.2 创建约束

3. 检查约束

检查约束对输入列或者整个表中的值设置检查条件，以限制输入值，保证数据库数据的完整性。

当使用检查约束时，应该考虑和注意以下几点：

- 一个列级检查约束只能与限制的字段有关；一个表级检查约束只能与限制的表中字段有关；
- 一个表中可以定义多个检查约束；
- 每个CREATE TABLE语句中每个字段只能定义一个检查约束；
- 在多个字段上定义检查约束，则必须将检查约束定义为表级约束；
- 当执行INSERT语句或者UPDATE语句时，检查约束将验证数据；
- 检查约束中不能包含子查询。

5.2.2 创建约束

3. 检查约束

创建检查约束常用的操作方法有如下两种 ：使用SQL Server管理平台创建检查约束；用Transact-SQL语句创建检查约束。

（1）使用SQL Server管理平台创建检查约束，如图5-9所示。

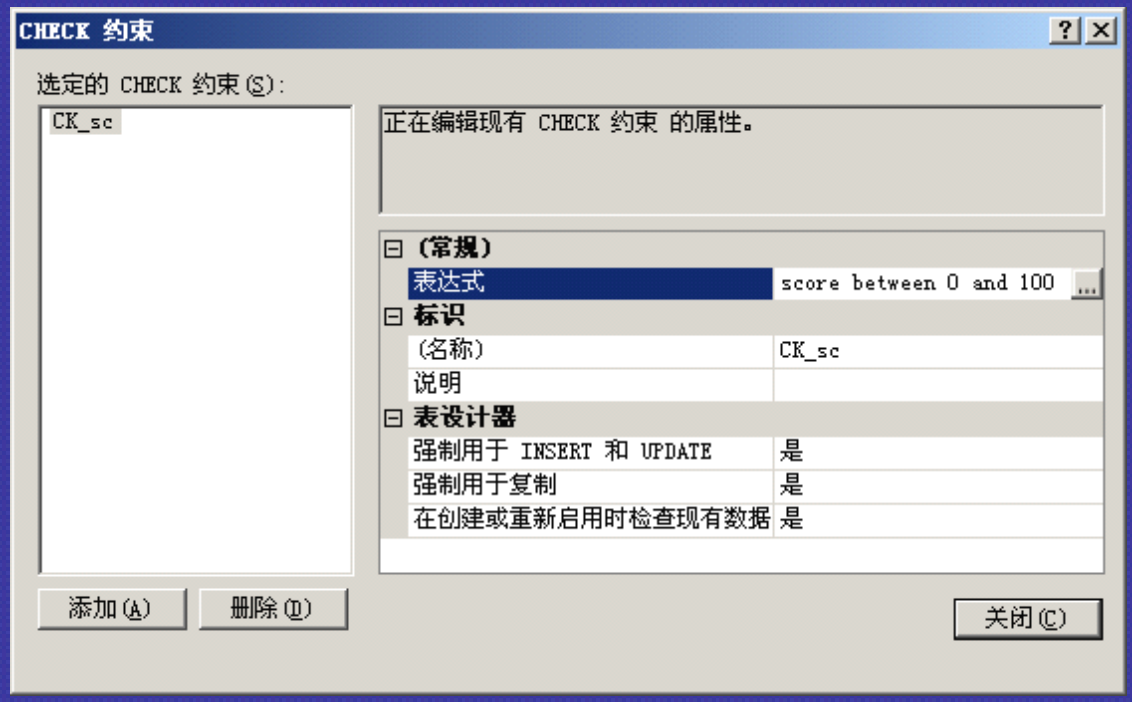


图5-9 创建检查约束对话框

5.2.2 创建约束

3. 检查约束

（2）用Transact-SQL语句创建检查约束。

用Transact-SQL语句创建检查约束。其语法形式如下：

```
CONSTRAINT  constraint_name
CHECK  [NOT FOR REPLICATION]
(logical_expression)
```

例5-8 建立一个SC表，定义SCORE 的取值范围为0到100之间。

程序清单如下：

```
CREATE TABLE SC
(SNO CHAR(5),
CNO CHAR(5),
SCORE NUMERIC(5,1) CONSTRAINT SCORE_CHK
CHECK(SCORE>=0 AND SCORE <=100))
```

5.2.2 创建约束

4. 默认（DEFAULT）约束

默认约束指定在插入操作中如果没有提供输入值时，则系统自动指定值。默认约束可以包括常量、函数、不带变元的内建函数或者空值。

使用默认约束时，应该注意以下几点：

（1）每个字段只能定义一个默认约束；

（2）如果定义的默认值长于其对应字段的允许长度，那么输入到表中的默认值将被截断；

（3）不能加入到带有IDENTITY属性或者数据类型为timestamp的字段上；

（4）如果字段定义为用户定义的数据类型，而且有一个默认绑定到这个数据类型上，则不允许该字段有默认约束。

5.2.2 创建约束

4. 默认（DEFAULT）约束

创建默认约束常用的操作方法有如下两种：使用SQL Server管理平台创建默认约束；创建默认约束的Transact-SQL语句操作法。

（1）使用SQL Server管理平台创建默认约束，如图5-10所示。

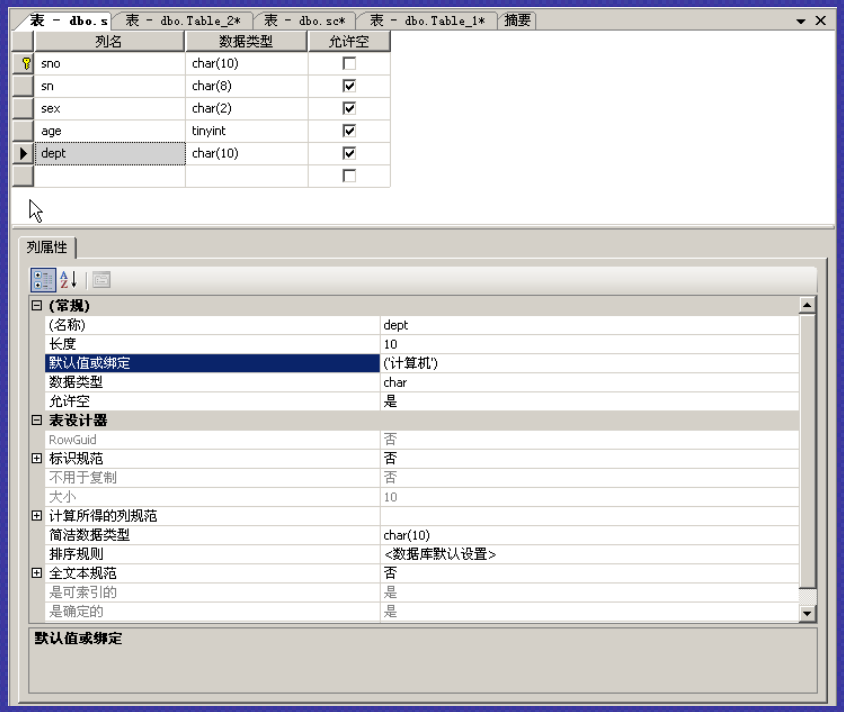


图5-10 创建默认约束对话框

5.2.2 创建约束

4. 默认（**DEFAULT**）约束

（2）创建默认约束的Transact-SQL语句操作法。其语法形式如下：

CONSTRAINT constraint_name

DEFAULT constraint_expression [FOR column_name]

例5-10 为 dept字段创建默认约束。

程序清单如下：

constraint con_dept default ‘计算机’ for dept

5.2.2 创建约束

5. 外部键约束

外键 (FOREIGN KEY) 是用于建立和加强两个表数据之间的链接的一列或多列。外部键约束用于强制参照完整性。

当使用外部键约束时，应该考虑以下几个因素：

- 外部键约束提供了字段参照完整性；
- 外部键从句中的字段数目和每个字段指定的数据类型都必须和REFERENCES从句中的字段相匹配；
- 外部键约束不能自动创建索引，需要用户手动创建；
- 用户想要修改外部键约束的数据，必须有对外部键约束所参考表的SELECT权限或者REFERENCES权限；
- 参考同一表中的字段时，必须只使用REFERENCES子句，不能使用外部键子句；
- 一个表中最多可以有31个外部键约束；
- 在临时表中，不能使用外部键约束；
- 主键和外部键的数据类型必须严格匹配

5.2.2 创建约束

5. 外部键约束

创建外部键约束常用的操作方法有如下两种：在SQL Server管理平台中添加外部键约束；使用Transact-SQL语句设置外部键约束。

（1）在SQL Server管理平台中添加外部键约束，在SQL Server管理平台中添加外部键约束。如图5-11，5-12所示。

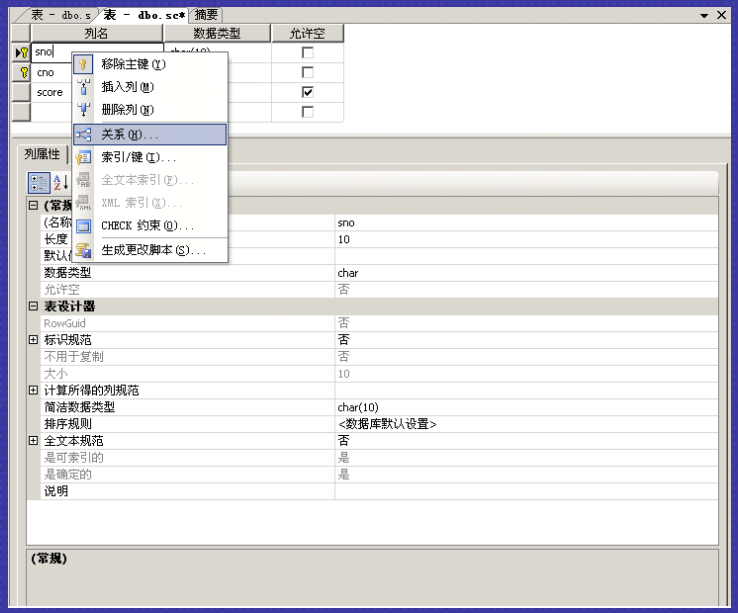


图5-11 选择创建外键约束的字段

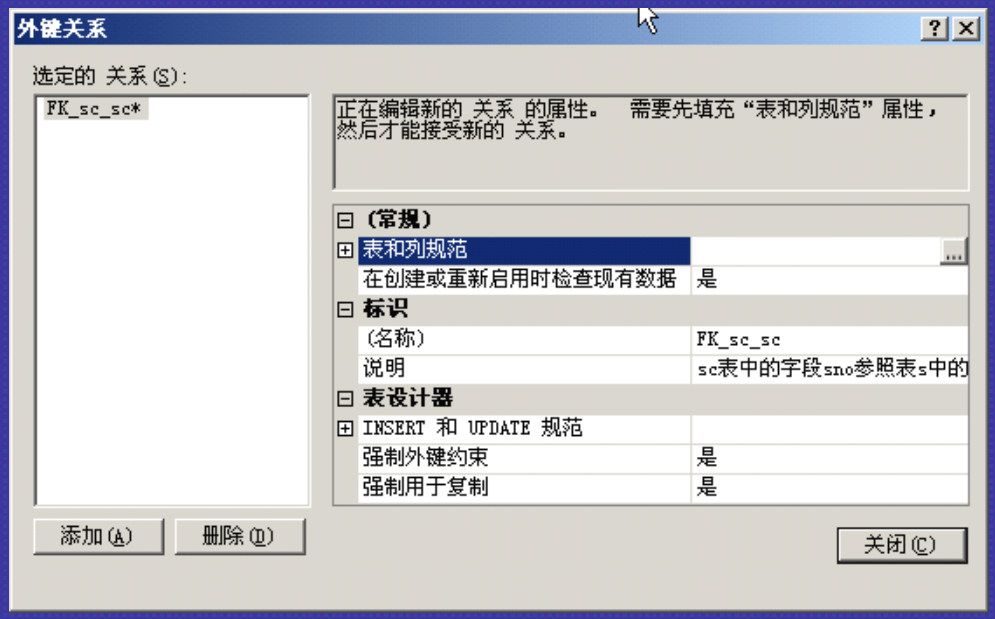


图5-12 创建外键约束对话框

5.2.2 创建约束

5. 外部键约束

（2）使用Transact-SQL语句设置外部键约束，其语法形式如下：

```
CONSTRAINT  constraint_name
FOREIGN  KEY  （column_name[,…n]）
REFERENCES  ref_table  [（ref_column[,…n]）]
```

例5-11 建立一个SC表，定义SNO,CNO为SC的外部键。

程序清单如下：

```
CREATE TABLE SC
(SNO CHAR(5) NOT NULL
 CONSTRAINT S_FORE FOREIGN KEY REFERENCES S(SNO),
CNO CHAR(5) NOT NULL
 CONSTRAINT C_FORE FOREIGN KEY REFERENCES C(CNO),
SCORE NUMERIC(3),
CONSTRAINT S_C_PRIM PRIMARY KEY (SNO,CNO))
```

5.2.2 创建约束

6. 空值（NULL）约束

•空值（NULL）约束用来控制是否允许该字段的值为NULL。NULL值不是0也不是空白，更不是填入字符串的“NULL”字符串，而是表示“不知道”、“ 不确定”或“没有数据”的意思。

•当某一字段的值一定要输入才有意义的时候，则可以设置为NOT NULL。如主键列就不允许出现空值，否则就失去了唯一标识一条记录的作用。空值（NULL）约束只能用于定义列约束。

•创建空值（NULL）约束常用的操作方法有如下两种：

（1）在SQL Server管理平台中添加空值（NULL）约束；

（2）使用Transact-SQL语句设置空值（NULL）约束。

5.2.2 创建约束

6. 空值（NULL）约束

（1）在SQL Server管理平台中添加空值（NULL）约束。如图5-14所示。

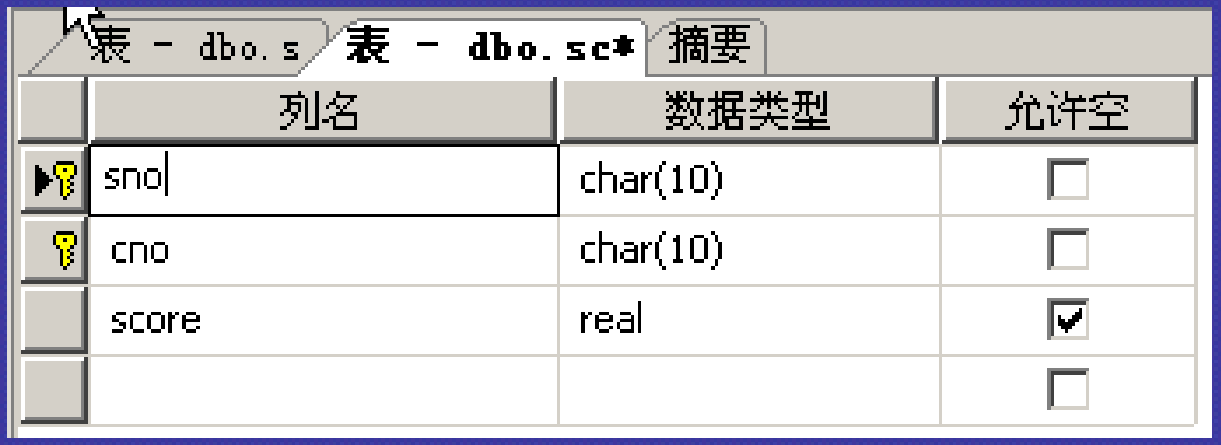


图5-14设置空值（NULL）约束对话框

5.2.2 创建约束

6. 空值（NULL）约束

（2）使用Transact-SQL语句设置空值（NULL）约束， 其语法形式如下：

[CONSTRAINT <约束名>][NULL|NOT NULL]

例5-13 建立一个S表，对SNO字段进行NOT NULL约束。

程序清单如下：

```
CREATE TABLE S
(SNO CHAR(10) CONSTRAINT S_CONS NOT NULL,
SN VARCHAR(20),
AGE INT,
SEX CHAR(2) DEFAULT '男',
DEPT VARCHAR(20))
```

5.2.3 修改表

当数据库中的表创建完成后，可以根据需要改变表中原先定义的许多选项，以更改表的结构。用户可以增加、删除和修改列，增加、删除和修改约束，更改表名以及改变表的所有者等。

1、修改列属性

修改列属性包括以下一些内容：

（1）修改列的数据类型；

（2）修改列的数据长度；

（3）修改列的精度；

（4）修改列的小数位数；

（5）修改列的为空性。

5.2.3 修改表

2、添加和删除列

在SQL Server 2005中，如果列允许空值或对列创建 DEFAULT 约束，则可以将列添加到现有表中。将新列添加到表时，SQL Server 2005数据库引擎在该列为表中的每个现有数据行插入一个值。因此，在向表中添加列时向列添加DEFAULT定义会很有用。如果新列没有 DEFAULT定义，则必须指定该列允许空值。数据库引擎将空值插入该列，如果新列不允许空值，则返回错误。

反之，可以删除现有表中的列，但具有下列特征的列不能被删除：

（1）用于索引；

（2）用于CHECK、FOREIGN KEY、UNIQUE 或PRIMARY KEY约束；

（3）与DEFAULT 定义关联或绑定到某一默认对象；

（4）绑定到规则；

（5）已注册支持全文；

（6）用作表的全文键。

5.2.3 修改表

3、增加、修改和删除约束

（1）增加、修改和删除PRIMARY KEY 约束。

（2）增加、修改和删除UNIQUE约束。

（3）增加、修改和删除CHECK约束。

（4）增加、修改和删除DEFAULT约束。

（5）增加、修改和删除FOREIGN KEY约束。

（6）增加和修改标识符列。只能为每个表创建一个标识符列和一个 GUID 列。

5.2.3 修改表

例 5-14　创建一个雇员信息表，然后在表中增加一个salary字段，删除表中的age字段，并且修改memo字段的数据类型。

SQL语句的程序清单如下：

```
create table employees（
id  char（8）  primary key,
name  char（20）  not null,
department  char（20）  null,
memo  char（30）  null,
age  int  null,
）
alter  table  employees
  add  salary  int  null,
  drop  column  age,
alter  column  memo  varchar（200）  null
```

5.2.3 修改表

例 5-15 在S表中增加一个班号列和住址列。

SQL语句的程序清单如下：

```
ALTER TABLE S
ADD
CLASS_NO CHAR(6),
ADDRESS CHAR(40)
```

注意：使用此方式增加的新列自动填充NULL值，所以不能为增加的新列指定NOT NULL约束。

例5-16 在SC表中增加完整性约束定义，使SCORE在0-100之间。

SQL语句的程序清单如下：

```
ALTER TABLE SC
ADD
CONSTRAINT SCORE_CHK CHECK(SCORE BETWEEN 0 AND 100)
```

5.2.4 查看表

当在数据库中创建了表后，有时需要查看表的有关信息。比如表的属性、定义、数据、字段属性和索引等。尤其重要的是查看表内存放的数据，另外有时需要查看表与其他数据库对象之间的依赖关系。

1. 查看表的定义，如图5-15，5-16所示。

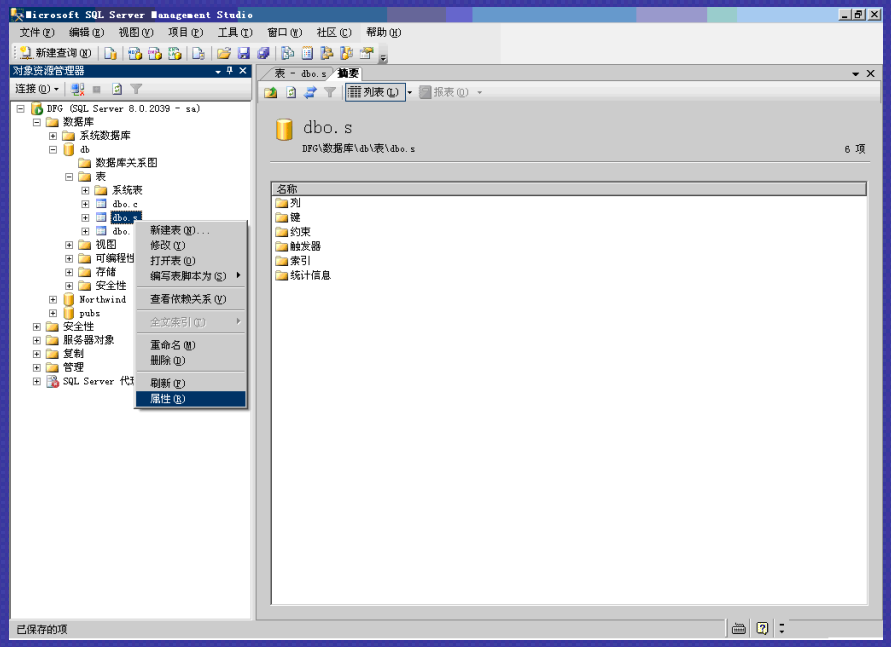


图5-15 选择表格属性对话框

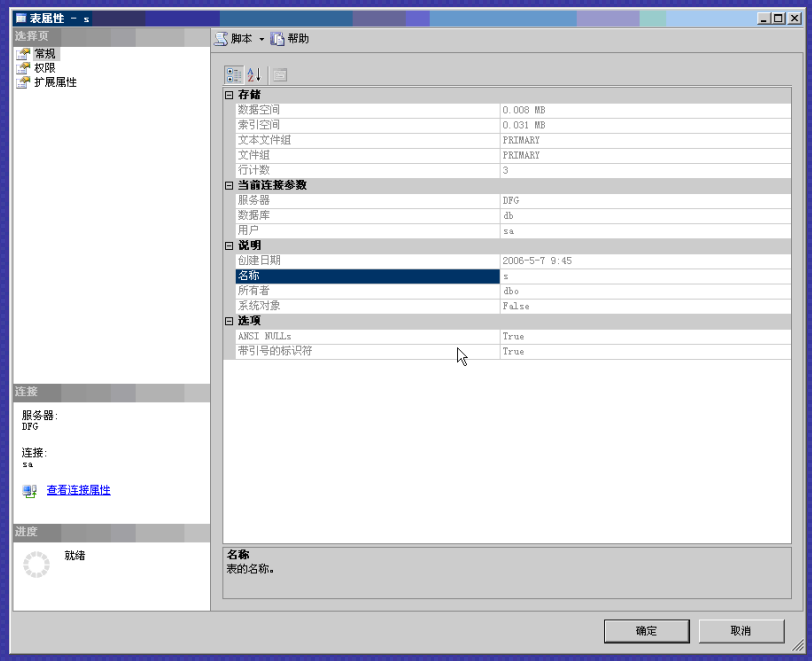


图5-16 表格属性对话框

5.2.4 查看表

2. 查看表中存储的数据，如图5-17，5-18所示。

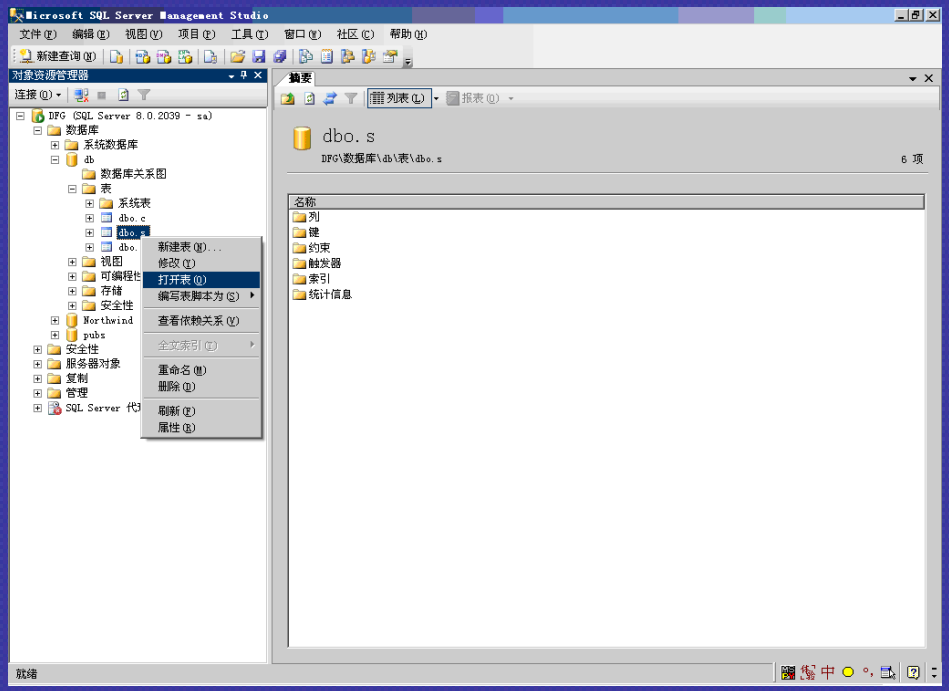
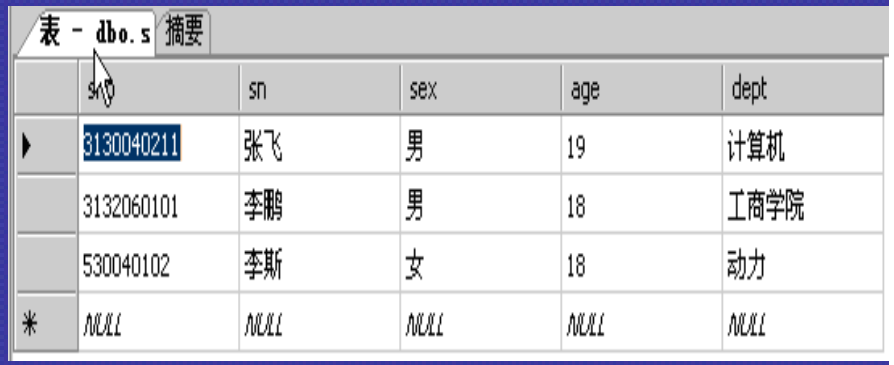


图5-17 选择打开表对话框



sno	sn	sex	age	dept
3130040211	张飞	男	19	计算机
3132060101	李鹏	男	18	工商学院
530040102	李斯	女	18	动力
NULL	NULL	NULL	NULL	NULL

图5-18 显示表格数据对话框

5.2.4 查看表

3. 查看表与其他数据库对象的依赖关系，如图5-19所示。

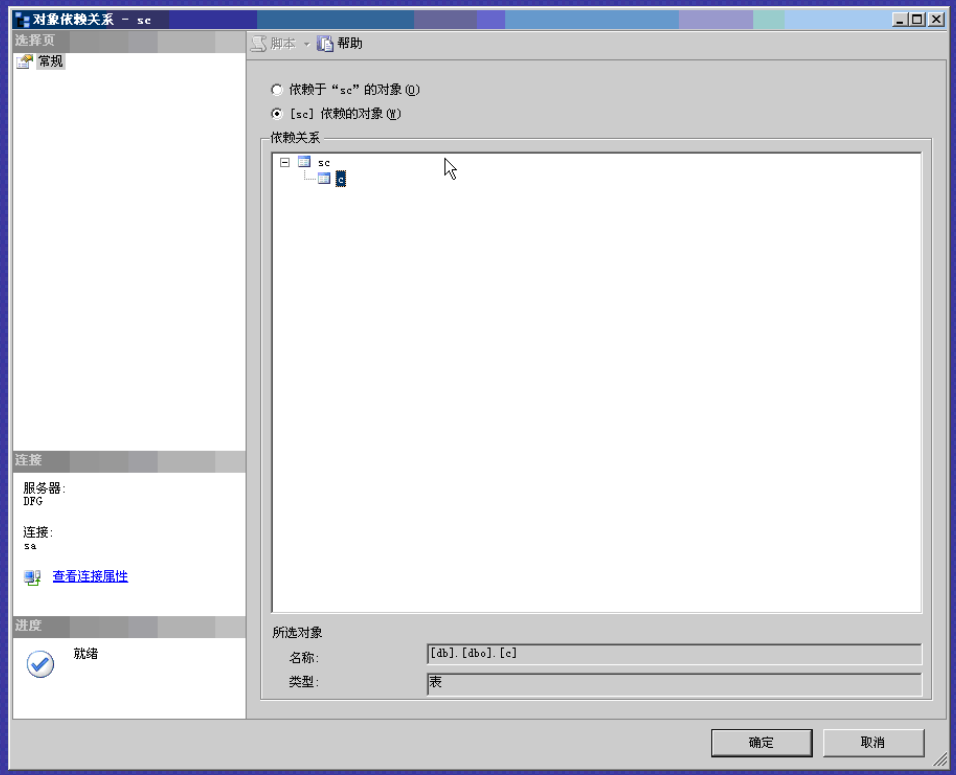


图5-19 显示相关性对话框

5.2.4 查看表

4. 利用系统存储过程查看表的信息

系统存储过程Sp_help可以提供指定数据库对象的信息，也可以提供系统或者用户定义的数据类型的信息，其语法形式如下：

sp_help [[@objname=]name]

例5-17 （1）显示当前数据库中所有对象的信息；（2）显示表Person.Contact 的信息。在SQL Server管理平台的查询窗口中，它们对应的语句和运行结果如图5-20和图5-21所示

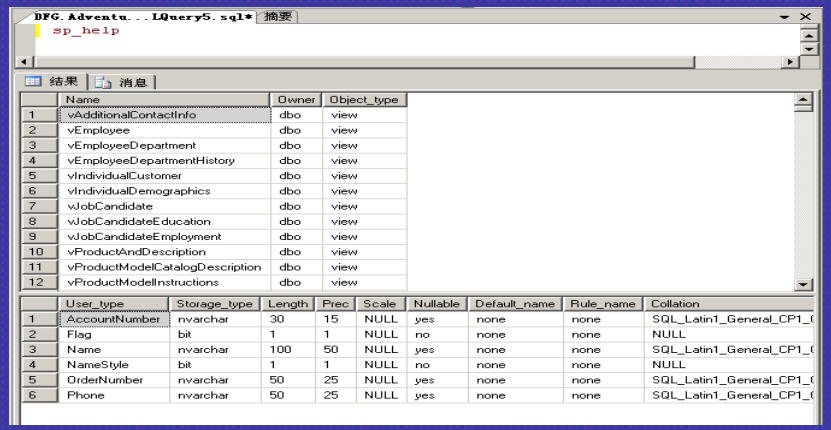


图5-20 所有数据库对象显示窗口

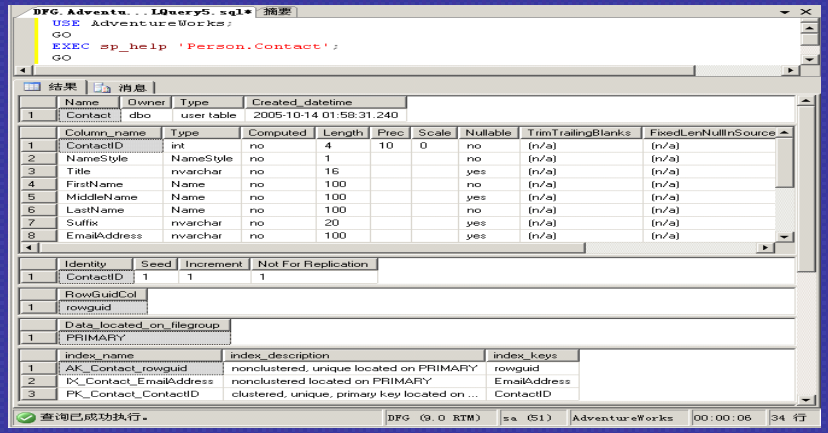


图5-21 当前数据库对象显示窗口

5.2.5 删除表

1. 利用管理平台删除表

在SQL Server管理平台中，展开指定的数据库和表，右击要删除的表，从弹出的快捷菜单中选择“删除”选项，则出现除去对象对话框，如图5-25所示。

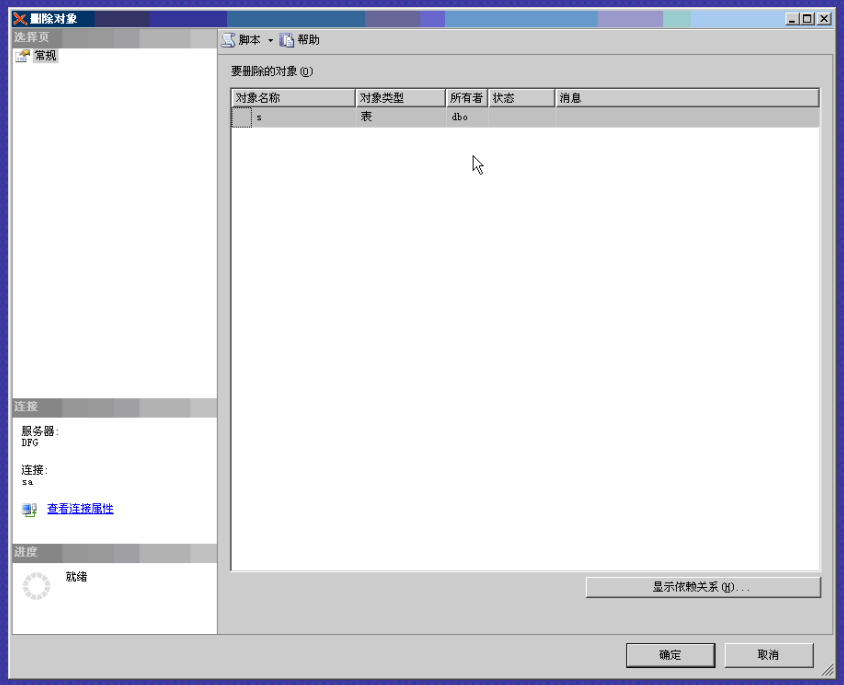


图5-25 删除表对话框

5.2.5 删除表

2. 利用DROP TABLE语句删除表

- DROP TABLE语句可以删除一个表和表中的数据及其与表有关的所有索引、触发器、约束、许可对象。
- DROP TABLE语句的语法形式如下：

  DROP TABLE table_name

- 要删除的表如果不在当前数据库中，则应在table_name中指明其所属的数据库和用户名。在删除一个表之前要先删除与此表相关联的表中的外部关键字约束。当删除表后，绑定的规则或者默认值会自动松绑。
- 例5-18　删除company数据库中的表employee。

程序如下：

```
drop  table  company.dbo.employee
```

5.3 索引操作

索引是数据库随机检索的常用手段，它实际上就是记录的关键字与其相应地址的对应表。通过索引可大大提高查询速度。此外，在SQL SERVER中，行的唯一性也是通过建立唯一索引来维护的。

使用索引可以大大提高系统的性能，其具体表现在：

（1）通过创建惟一索引，可以保证数据记录的惟一性。

（2）可以大大加快数据检索速度。

（3）可以加速表与表之间的连接，这一点在实现数据的参照完整性方面有特别的意义。

（4）在使用ORDER BY和GROUP BY子句进行检索数据时，可以显著减少查询中分组和排序的时间。

（5）使用索引可以在检索数据的过程中使用优化隐藏器，提高系统性能

5.3.1 创建索引

SQL Server 2005提供了如下几种创建索引的方法：

1. 利用SQL Server管理平台创建索引；

2. 利用Transact-SQL语句中的CREATE INDEX命令创建索引。

另外，可以在创建表的PRIMARY KEY或UNIQUE约束时自动创建索引。

5.3.1 创建索引

1.利用SQL Server管理平台创建索引。

(1)展开指定的服务器和数据库，选择要创建索引的表，展开该表，选择“索引”选项（如图5-26所示），右键单击索引，从弹出的快捷菜单中选择“新建索引”，就会出现新建索引对话框，如图5-27所示。

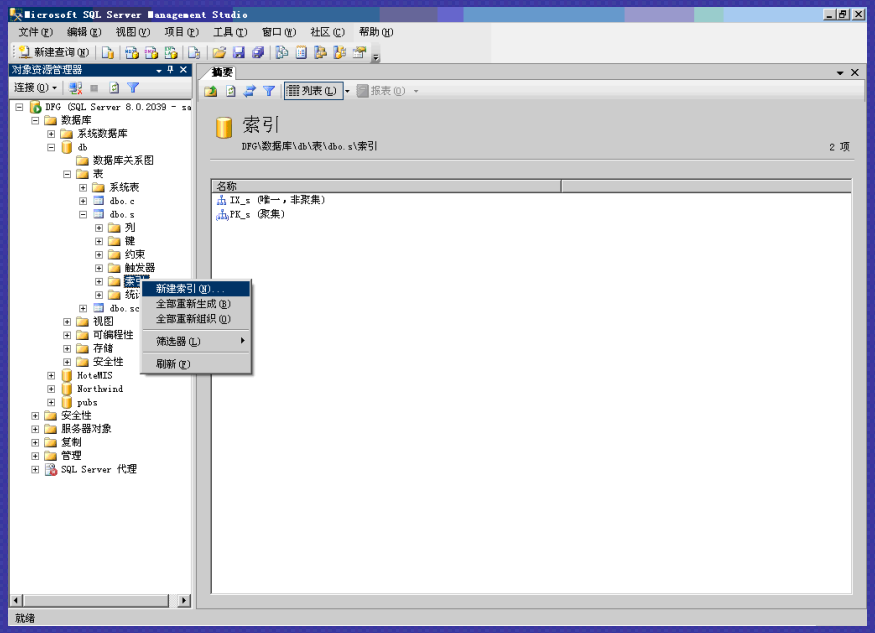


图5-26 选择新建索引选项对话框

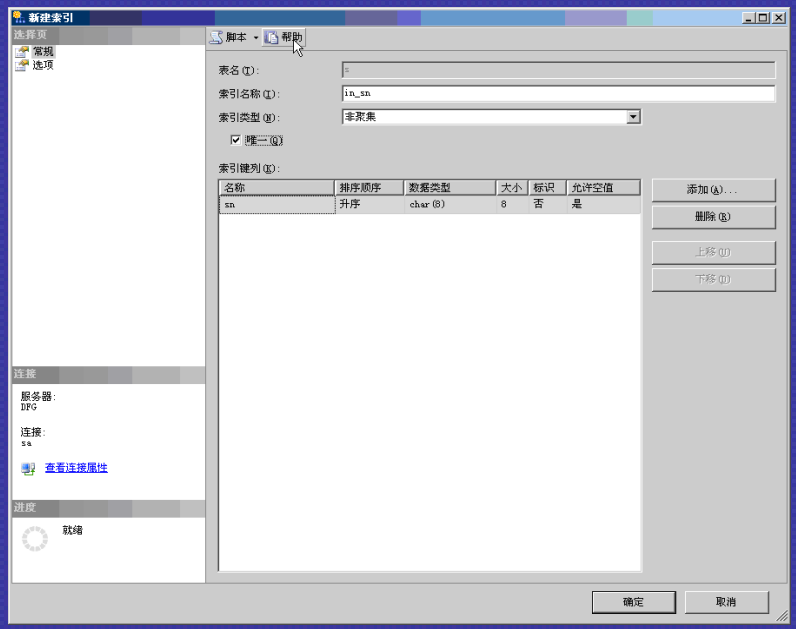


图5-27 创建索引对话框

5.3.1 创建索引

1.利用SQL Server管理平台创建索引。

（2）点击“添加”按钮，可选择用于创建索引的字段，如图5-28所示。

（3）打开创建索引对话框的选项页框，在此还可以设定索引的属性,如图5-29所示。

图5-28 选择用于创建索引的字段

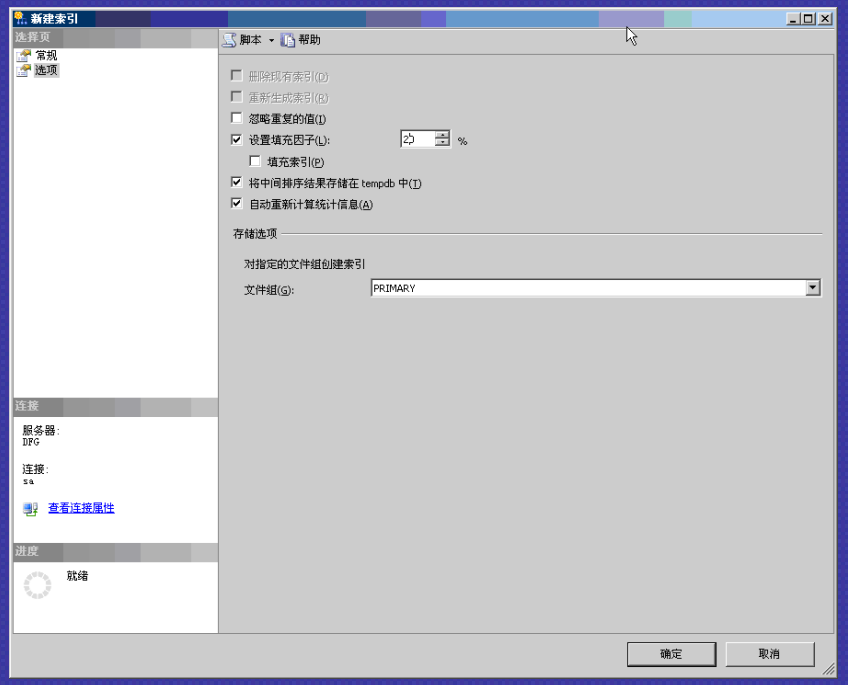


图5-29 索引对话框的选项页框

5.3.1 创建索引

2.利用Transact-SQL语句中的CREATE INDEX命令创建索引

CREATE INDEX命令既可以创建一个可改变表的物理顺序的聚集索引，也可以创建提高查询性能的非聚集索引，其语法形式如下：

```
CREATE [ UNIQUE ] [ CLUSTERED | NONCLUSTERED ]
INDEX index_name ON { table | view } （ column [ ASC | DESC ]
[ ,...n ] ）
[with
[PAD_INDEX]
      [[,]FILLFACTOR=fillfactor]
      [[,]IGNORE_DUP_KEY]
      [[,]DROP_EXISTING]
    [[,]STATISTICS_NORECOMPUTE]
[[,]SORT_IN_TEMPDB]
                ]
[ ON filegroup ]
```

5.3.1　创建索引

2.利用Transact-SQL语句中的CREATE INDEX命令创建索引

例5-19　为表employees创建了一个惟一聚集索引。

程序清单如下：

```
CREATE UNIQUE CLUSTERED INDEX number_ind
    ON employees （number）
with
    pad_index,
    fillfactor=20,
    ignore_dup_key,
    drop_existing,
    statistics_norecompute
```

5.3.2 查看、修改和删除索引

1.利用SQL Server管理平台查看、修改和删除索引。

（1）在SQL Server管理平台中，展开指定的服务器和数据库项，并展开要查看的表，从选项中选择“索引”选项，则会出现表中已存在的索引列表。双击某一索引名称，则出现索引属性对话框，如图5-30所示。索引碎片管理页框如图5-31所示。

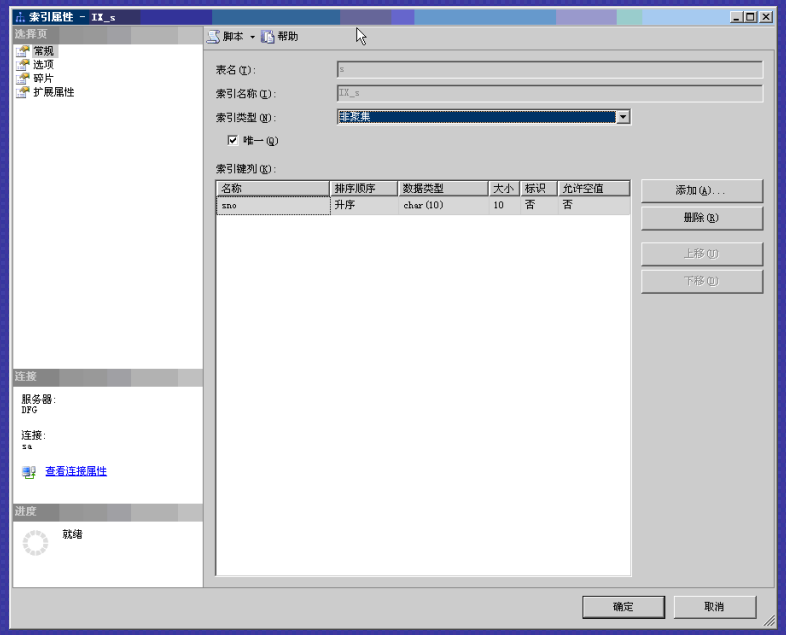


图5-30 索引属性对话框

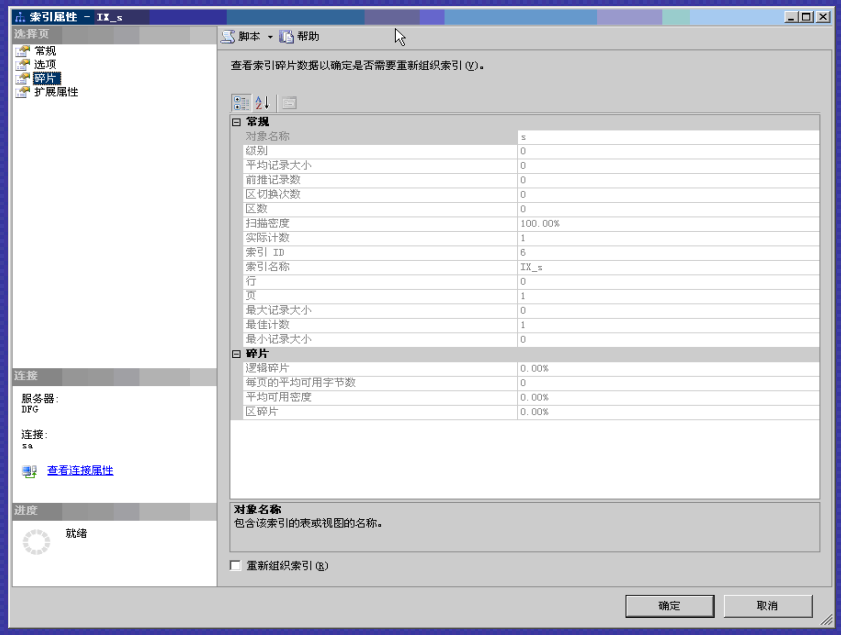


图5-31 索引碎片数据管理对话框

5.3.2 查看、修改和删除索引

1.利用SQL Server管理平台查看、修改和删除索引。

（2）扩展属性对话框，如图5-32所示，主要包含数据库名称，校对模式等。

通过右键单击索引名称，选择“创建索引脚本到新的查询分析器窗口”，则可以查看创建索引的SQL脚本语句，如图5-33所示。

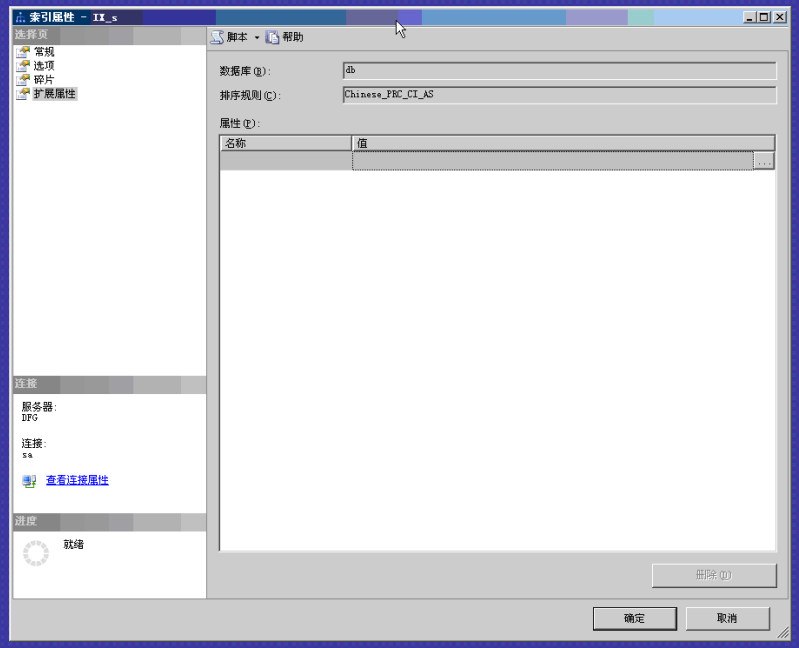


图5-32 索引扩展属性对话框

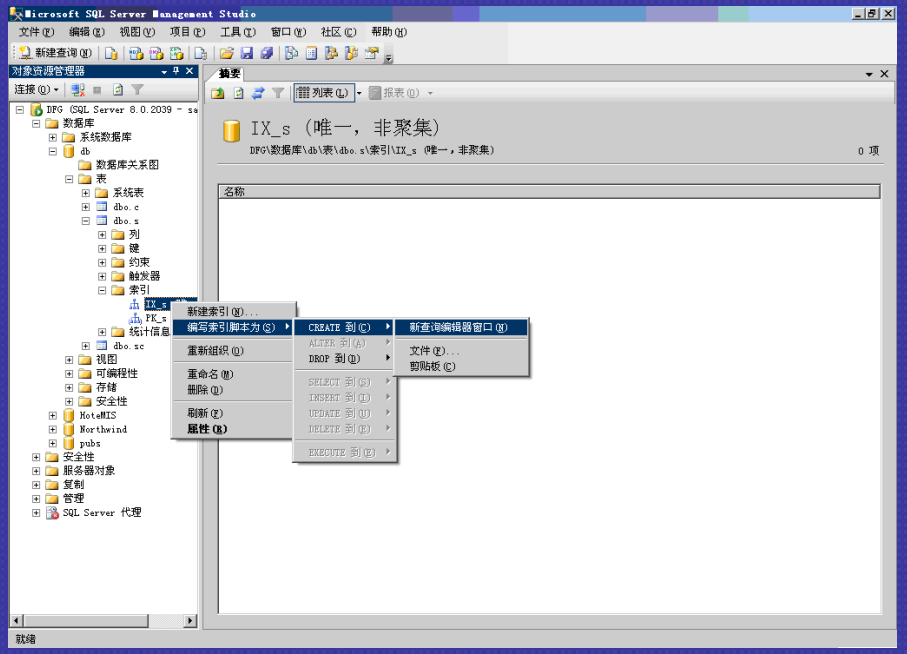


图5-33 索引的SQL脚本对话框

5.3.2 查看、修改和删除索引

2. 用系统存储过程查看和更改索引名称

系统存储过程sp_helpindex可以返回表的所有索引信息，其语法形式如下；

sp_helpindex [@objname=]’name’

其中，[@objname=]’name’参数用于指定当前数据库中的表的名称。

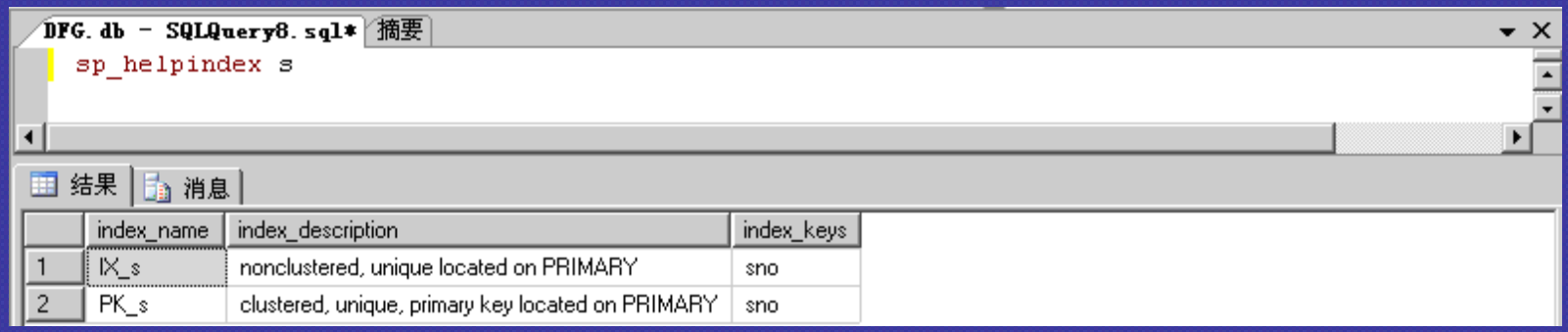
另外，系统存储过程sp_rename可以用来更改索引的名称，其语法形式如下：

sp_rename[@objname=]'object_name',
[@newname=]'new_name'
[, [@objtype =] 'object_type']

5.3.2 查看、修改和删除索引

2. 用系统存储过程查看和更改索引名称

例5-21 使用系统存储过程sp_helpindex 来查看表authors的索引信息。运行结果如下图所示。



	index_name	index_description	index_keys
1	IX_s	nonclustered, unique located on PRIMARY	sno
2	PK_s	clustered, unique, primary key located on PRIMARY	sno

例5-22 将employees表中的索引名称employees_name_index名称更改为employees _name_ind。

程序清单如下：

```
exec sp_rename ‘employees.[employees_name_index]’,
‘employees_name_ ind’, ‘index’
```

5.3.2 查看、修改和删除索引

3.使用Transact-SQL语句中的DROP INDEX命令删除索引

•当不再需要某个索引时，可以将其删除，DROP INDEX命令可以删除一个或者多个当前数据库中的索引，其语法形式如下：

DROP INDEX 'table.index | view.index' [,...n]

•其中，table | view用于指定索引列所在的表或索引视图；index用于指定要删除的索引名称。

例5-23　删除表employees中的索引employees_name_index。

程序清单如下：

```
drop index employees.employees_name_index
```

第6章 查询技术

SELECT 主要子句

```
SELECT [ ALL | DISTINCT ]
  [TOP expression [PERCENT] [WITH TIES ]]
  < select_list >
  [ INTO new_table ]
  [ FROM { <table_source> } [ ,...n ] ]
  [ WHERE <search_condition> ]
  [ GROUP BY [ ALL ] group_by_expression [ ,...n ]
  [ WITH { CUBE | ROLLUP } ]
  [ HAVING < search_condition > ]
[ORDER BY order_expression [ASC|DESC]]
  [ COMPUTE  {{AVG|COUNT|MAX|MIN|SUM}
  (expression)} [ ,...n ]
  [ BY expression [ ,...n ] ]
```

SELECT 主要子句

参数说明如下：

- SELECT子句用于指定所选择的要查询的特定表中的列，它可以是星号（*）、表达式、列表、变量等。
- INTO子句用于指定所要生成的新表的名称。
- FROM子句用于指定要查询的表或者视图，最多可以指定16个表或者视图，用逗号相互隔开。
- WHERE子句用来限定查询的范围和条件。
- GROUP BY子句是分组查询子句。
- HAVING子句用于指定分组子句的条件。
- GROUP BY子句、HAVING子句和集合函数一起可以实现对每个组生成一行和一个汇总值。
- ORDER BY子句可以根据一个列或者多个列来排序查询结果，在该子句中，既可以使用列名，也可以使用相对列号。
- ASC表示升序排列，DESC表示降序排列。
- COMPUTE子句使用集合函数在查询的结果集中生成汇总行。
- COMPUTE BY子句用于增加各列汇总行。

6.1 基本SELECT语句

6.1.1 投影查询

6.1.2 条件查询

6.1.1 投影查询

- 最基本的 SELECT 语句仅有两个部分：要返回的列，和这些列源于的表。也就是说查询均为不使用WHERE子句的无条件查询，也称作投影查询。
- 例6-1 查询全体学生的学号、姓名和年龄。

  程序清单如下：

  SELECT SNO, SN, AGE FROM S
- 例6-2 查询学生的全部信息。

  程序清单如下：

  SELECT * FROM S
- 注意：用‘ * ’表示表的全部列名，而不必逐一列出。

6.1.1 投影查询

- 例6-3 查询选修了课程的学生号。

  程序清单如下：

  SELECT DISTINCT SNO FROM SC

  注意：应用DISTINCT消除查询结果以某列为依据的重复行。上例中，sc表中相同学号（SNO）的纪录只保留第一行，余下的具有相同学号的记录将从查询结果中清除。也就是每个同学保留一条选课纪录。

- 另外，利用投影查询可控制列名的顺序，并可通过指定别名改变查询结果的列标题的名字，如下例。
- 例6-4 查询全体学生的姓名、学号和年龄。

  程序清单如下：

  SELECT SN NAME, SNO, AGE FROM S
- 注意：NAME为SN的别名，这里我们改变了列的显示顺序。

6.1.2 条件查询

- 当要在表中找出满足某些条件的行时，则需使用WHERE子句指定查询条件。WHERE子句中，条件通常通过三部分来描述：列名；比较运算符；列名、常数。
- 条件查询又可分为以下几方面内容：

1、比较大小和确定范围；

2、部分匹配查询；

3、空值查询；

4、查询的排序

6.1.2 条件查询

1、比较大小和确定范围

- 例6-5 查询选修课程号为‘C1’的学生的学号和成绩。

  程序清单如下：

  SELECT SNO,SCORE FROM SC WHERE CNO=’C1’

- 例6-6 查询成绩高于85分的学生的学号、课程号和成绩。

  程序清单如下：

  SELECT SNO,CNO,SCORE FROM SC WHERE SCORE>85

6.1.2 条件查询

1、比较大小和确定范围

- 当WHERE子句需要指定一个以上的查询条件时，则需要使用逻辑运算符AND、OR和NOT将其连结成复合的逻辑表达式。其优先级由高到低为：NOT、AND、OR，用户可以使用括号改变优先级。
- 例6-7 查询选修C1或C2且分数大于等于85分学生的的学号、课程号和成绩。

程序清单如下：

```
SELECT SNO，CNO，SCORE FROM SC
WHERE（CNO='C1' OR CNO='C2'） AND SCORE>=85
```

SQL语句中也有一个特殊的 BETWEEN 运算符，用于检查某个值是否在两个值之间（包括等于两端的值）。

- 例6-8 查询工资在1000至1500之间的教师的教师号、姓名及职称。

程序清单如下：

```
SELECT TNO,TN,PROF FROM T
WHERE SAL BETWEEN 1000 AND 1500
```

- 上面SQL语句等价于以下语句：

```
SELECT TNO,TN,PROF FROM T
WHERE SAL>=1000 AND SAL<=1500
```

6.1.2 条件查询

1、比较大小和确定范围

- 注意：在SELECT语句中可利用“IN”操作来查询属性值属于指定集合的元组。利用“NOT IN”可以查询指定集合外的元组。如下面两个例子。
- 例6-10 查询选修C1或C2的学生的学号、课程号和成绩。

  程序清单如下：

```
SELECT SNO, CNO, SCORE
FROM SC
WHERE CNO IN('C1', 'C2')
```

- 此语句也可以使用逻辑运算符“OR”实现。相应的程序清单如下：

```
SELECT SNO, CNO, SCORE
FROM SC
WHERE CNO='C1' OR CNO= 'C2'
```

6.1.2 条件查询

2、部分匹配查询

- 当不知道完全精确的值时，用户还可以使用LIKE或NOT LIKE进行部分匹配查询（也称模糊查询）。LIKE运算使我们可以使用通配符来执行基本的模式匹配。
- 使用LIKE运算符的一般格式为：

  <属性名> LIKE <字符串常量>

  字符串常量的字符可以包含如表6-2所示的通配符。

表6-2 通配符及说明

通配符	说明
_	表示任意单个字符
%	表示任意长度的字符串
[ ]	与特定范围（例如，[a-f]）或特定集（例如，[abcdef]）中的任意单字符匹配。
[^]	与特定范围（例如，[^a-f]）或特定集（例如，[^abcdef]）之外的任意单字符匹配。

6.1.2 条件查询

2、部分匹配查询

- 例6-12 查询所有姓张的教师的教师号和姓名。

  程序清单如下：

```
SELECT TNO, TN
FROM T
WHERE TN LIKE '张%'
```

- 例6-13 查询姓名中第二个汉字是“力”的教师号和姓名。

  程序清单如下：

```
SELECT TNO, TN
FROM T
WHERE TN LIKE '_力%'
```

6.1.2 条件查询

3、空值查询

- 某个字段没有值称之为具有空值（NULL）。通常没有为一个列输入值时，该列的值就是空值。空值不同于零和空格，它不占任何存储空间。例如，某些学生选课后没有参加考试，有选课记录，但没有考试成绩，考试成绩为空值，这与参加考试，成绩为零分的不同。
- 例6-15 查询没有考试成绩的学生的学号和相应的课程号。

  程序清单如下：

```
SELECT SNO, CNO
FROM SC
WHERE SCORE IS NULL
```

- 注意：这里的空值条件为IS NULL，不能写成SCORE=NULL。

6.1.2 条件查询

4. 查询的排序

- 当需要对查询结果排序时，应该在SELECT语句中使用ORDER BY子句。ORDER BY 子句包括了一个或多个用于指定排序顺序的列名，排序方式可以指定，DESC为降序，ASC为升序，缺省时为升序。ORDER BY子句必须出现在其他子句之后。
- ORDER BY 子句支持使用多列。可以使用以逗号分隔的多个列作为排序依据：查询结果将先按指定的第一列进行排序，然后再按指定的下一列进行排序。
- 例6-16 查询选修C1 的学生学号和成绩，并按成绩降序排列。

程序清单如下：

```
SELECT SNO, SCORE
FROM SC
WHERE CNO='C1'
ORDER BY SCORE DESC
```

6.2 分组查询

6.2.1 聚合函数和GROUP BY子句

6.2.2 GROUP BY 和 WHERE 子句、HAVING 子句

6.2.1 聚合函数和GROUP BY子句

GROUP BY子句可以将查询结果按属性列或属性列组合在行的方向上进行分组，每组在属性列或属性列组合上具有相同的聚合值。如果聚合函数没有使用 GROUP BY 子句，则只为 SELECT 语句报告一个聚合值。常用的聚合函数，如表6-3所示。

表6-3常用的聚合函数

函数名称	MIN	MAX	SUM	AVG	COUNT	COUNT(*)
功能	求一列中的最小值	求一列中的最大值	按列计算值的总和	按列计算平均值	按列值计个数	返回表中的所用行数

6.2.1 聚合函数和GROUP BY子句

•例6-18　通过查询求学号为S1学生的总分和平均分。

程序清单如下：

```
SELECT SUM(SCORE) AS TotalScore, AVG(SCORE) AS AveScore
FROM SC
WHERE SNO = 'S1'
```

•注意：函数SUM和AVG只能对数值型字段进行计算。

•例6-19 通过查询求选修C1号课程的最高分、最低分及之间相差的分数

程序清单如下：

```
SELECT MAX(SCORE) AS MaxScore, MIN(SCORE) AS MinScore,
MAX(SCORE)- MIN(SCORE) AS Diff
FROM SC
WHERE (CNO = 'C1')
```

6.2.1 聚合函数和GROUP BY子句

•例6-20 通过查询求管理系学生的总数。

程序清单如下：

```
SELECT COUNT(SNO) FROM S
WHERE DEPT='管理'
```

•例6-21 通过查询求学校中共有多少个系。

程序清单如下：

```
SELECT COUNT(DISTINCT DEPT) AS DeptNum
FROM S
```

•注意：加入关键字DISTINCT后表示消去重复行，可计算字段“DEPT“不同值的数目。COUNT函数对空值不计算，但对零进行计算。

•例6-22 统计有成绩的学生的人数。

程序清单如下：

```
SELECT COUNT (SCORE)
FROM SC
```

•注意：上例中成绩为零的同学计算在内，没有成绩（即为空值）的不计算。

6.2.1 聚合函数和GROUP BY子句

•例6-23 利用特殊函数COUNT(*)求计算机系学生的总数

程序清单如下：

```
SELECT COUNT(*) FROM S WHERE DEPT='计算机'
```

•注意：上例中，COUNT（*）用来统计元组的个数。此函数不消除重复行，也不允许使用DISTINCT关键字。

•在分组查询中，只要表达式中不包括聚合函数，就可以按该表达式分组。如下例所示。

•例6-24 查询每位学生的学号及其选课的门数。

程序清单如下：

```
SELECT CNO,COUNT(*) AS C_NUM  FROM SC GROUP BY CNO
```

• GROUP BY子句按CNO的值分组，所有具有相同CNO的元组为一组，对每一组使用函数COUNT进行计算，统计出各位学生选课的门数。

•例6-25统计各年度出生的雇员人数

程序清单如下。

```
Use adventureworks
SELECT DATEPART(year, birthdate ) AS Year,
COUNT(*) AS NumberOfemployees
FROM humanresources.employee
GROUP BY DATEPART(year, birthdate)
```

6.2.2 GROUP BY 和 WHERE 子句、HAVING 子句

- 可以在包含 GROUP BY 子句的查询中使用 WHERE 子句。在完成任何分组之前，将消除不符合 WHERE 子句中的条件的行。若在分组后还要按照一定的条件进行筛选，则需使用HAVING子句。
- 例6-26 在分组查询中使用WHERE条件，查询计算机系的学生学号及平均成绩。

  程序清单如下：

```
SELECT sno, AVG(score) AS 'AverageScore'
FROM sc
WHERE sno=(select sno from s where dept='计算机')
GROUP BY sno
ORDER BY sno
GO
```

6.2.2 GROUP BY 和 WHERE 子句、HAVING 子句

- 例6-27 在分组查询中使用HAVING条件，查询平均成绩大于85的学生学号及平均成绩。

  程序清单如下：

```
SELECT sno, AVG(score) AS 'AverageScore'
FROM sc
GROUP BY sno
HAVING AVG(score) >85
GO
```

- 注意：如果 HAVING 中包含多个条件，那么这些条件将通过 AND、OR 或 NOT 组合在一起

6.2.2 GROUP BY 和 WHERE 子句、HAVING 子句

•例6-28 查询选课在三门以上且各门课程均及格的学生的学号及其总成绩，查询结果按总成绩降序列出。

程序清单如下：

```
SELECT SNO,SUM(SCORE) AS TotalScore
FROM SC
WHERE SCORE>=60
GROUP BY SNO
HAVING COUNT(*)>=3
ORDER BY SUM(SCORE) DESC
```

6.3 连接查询

数据表之间的联系是通过表的字段值来体现的，这种字段称为连接字段。连接操作的目的就是通过加在连接字段的条件将多个表连接起来，以便从多个表中查询数据。前面的查询都是针对一个表进行的，当查询同时涉及两个以上的表时，称为连接查询。

6.3.1 等值连接与非等值连接

6.3.2自身连接

6. 3. 1 等值连接与非等值连接

- 连接条件的一般格式为：

[<表名1>.] <列名1> <比较运算符> [<表名2>.] <列名2>

- 其中，比较运算符主要有：＝、＞、＜、＞＝、＜＝、！＝。当比较运算符为“＝”时，称为等值连接，其他情况为非等值连接。
- 例6-29 查询张飞同学所选修的课程。

```
SELECT S.SNO ,SN,CNO
FROM S,SC
WHERE (S.SNO = SC. SNO) AND (SN='张飞')
```

6.3.1 等值连接与非等值连接

•例6-30 查询所有选课学生的学号、姓名、选课名称及成绩。

程序清单如下：

```
SELECT S.SNO,SN,CN,SCORE
FROM S,C,SC
WHERE  S.SNO=SC.SNO
AND SC.CNO=C.CNO
```

•注意：本例涉及三个表，WHERE子句中有两个连接条件。当有两个以上的表进行连接时，称为多表连接。

6.3.2 自身连接

•当一个表与其自已进行连接操作时，称为表的自身连接。要查询的内容均在同一表中，可以将表分别取两个别名，一个是X，一个是Y。将X，Y中满足查询条件的行连接起来。这实际上是同一表的自身连接。

•例6-31 查询所有比李明工资高的教师姓名、性别、工资和刘伟的工资。
程序清单如下：

```
SELECT X.TN,X.SAL AS SAL_a,Y.SAL AS SAL_b
FROM T AS X ,T AS Y
WHERE X.SAL>Y.SAL AND Y.TN='李明'
```

6.3.2 自身连接

例6-32 检索所有学生姓名，年龄和选课名称。

程序清单如下：

```
SELECT SN,AGE,CN
FROM S,C,SC
WHERE S.SNO=SC.SNO AND
SC.CNO=C.CNO
```

6.4 子查询

在WHERE子句中包含一个形如SELECT-FROM-WHERE的查询块，此查询块称为子查询或嵌套查询，包含子查询的语句称为父查询或外部查询。

6.4.1返回一个值的子查询

6.4.1返回一个值的子查询

6.4.1 返回一个值的子查询

- 当子查询的返回值只有一个时，可以使用比较运算符（=, >, <, >=, <=, !=）将父查询和子查询连接起来。
- 例6-33 查询与李明教师职称相同的教师号、姓名。

程序清单如下：

```
SELECT TNO,TN
FROM T
WHERE PROF=(SELECT PROF
FROM T
WHERE TN='李明')
```

6.4.2 返回一组值的子查询

如果子查询的返回值不止一个，而是一个集合时，则不能直接使用比较运算符，可以在比较运算符和子查询之间插入ANY或ALL。

1. 使用ANY
2. 使用IN
3. 使用ALL
4. 使用EXISTS

6.4.2 返回一组值的子查询

1.使用ANY

例6-34 查询讲授课程号为C5的教师姓名。

程序清单如下：

```
SELECT TN
FROM T
WHERE TNO=ANY
   (SELECT TNO
FROM TC
    WHERE CNO='C5')
```

6.4.2 返回一组值的子查询

2. 使用IN

可以使用IN代替“=ANY”。

例6-36 查询讲授课程号为C5的教师姓名

程序清单如下：

```
SELECT TN
  FROM T
  WHERE TNO IN
    (SELECT TNO
    FROM TC
   WHERE CNO='C5')
```

6.4.2 返回一组值的子查询

3. 使用ALL

ALL的含义为全部。

例6-37 查询其他系中比电力系所有教师工资都高的教师的姓名和工资。

程序清单如下：

```
SELECT TN,SAL
FROM T
WHERE SAL>ALL
      (SELECT SAL
      FROM T
       WHERE DEPT='电力')
      AND DEPT!= '电力'
```

6. 4. 2 返回一组值的子查询

4. 使用EXISTS

EXISTS表示存在量词，带有EXISTS的子查询不返回任何实际数据，它只得到逻辑值“真”或“假”。当子查询的的查询结果集合为非空时，外层的WHERE子句返回真值，否则返回假值。

•例6-38 查询讲授课程号为C5的教师姓名

程序清单如下：

```
SELECT TN
FROM T
WHERE EXISTS
        (SELECT *
FROM TC
       WHERE TNO=T.TNO
AND CNO='C5')
```

第7章 视图的操作与管理

视图概述

视图是个虚表，是从一个或者多个表或视图中导出的表，其结构和数据是建立在对表的查询基础上的。

使用视图的优点和作用主要有：

（1）视图可以使用户只关心他感兴趣的某些特定数据和他们所负责的特定任务，而那些不需要的或者无用的数据则不在视图中显示。

（2）视图大大地简化了用户对数据的操作。

（3）视图可以让不同的用户以不同的方式看到不同或者相同的数据集。

（4）在某些情况下，由于表中数据量太大，因此在表的设计时常将表进行水平或者垂直分割，但表的结构的变化对应用程序产生不良的影响。而使用视图可以重新组织数据，从而使外模式保持不变，原有的应用程序仍可以通过视图来重载数据。

（5）视图提供了一个简单而有效的安全机制。

7.1创建视图

SQL Server 2005提供了如下几种创建视图的方法：

（1）用SQL SERVER管理平台创建视图；

（2）用Transact-SQL语句中的CREATE VIEW命令创建视图；

（3）利用SQL SERVER管理平台的视图模板来创建视图。

创建视图时应该注意以下情况：

（1）只能在当前数据库中创建视图，在视图中最多只能引用1024列，视图中记录的数目限制只由其基表中的记录数决定。

（2）如果视图引用的基表或者视图被删除，则该视图不能再被使用，直到创建新的基表或者视图。

（3）如果视图中某一列是函数、数学表达式、常量或者来自多个表的列名相同，则必须为列定义名称。

（4）不能在视图上创建索引，不能在规则、默认、触发器的定义中引用视图。

（5）当通过视图查询数据时，SQL Server要检查以确保语句中涉及的所有数据库对象存在，每个数据库对象在语句的上下文中有效，而且数据修改语句不能违反数据完整性规则。

（6）视图的名称必须遵循标识符的规则，且对每个用户必须是惟一的。此外，该名称不得与该用户拥有的任何表的名称相同。

7.1.1利用SQL SERVER管理平台创建视图

利用SQL SERVER管理平台创建视图的具体操作步骤如下：

（1）在SQL SERVER管理平台中，展开指定的服务器，打开要创建视图的数据库文件夹，选择指定的数据库，右击该数据库图标，从弹出的快捷菜单中依次选择“新建（New）→视图”选项，如图7-1所示。接着就出现添加表、视图、函数对话框。如图7-2所示。

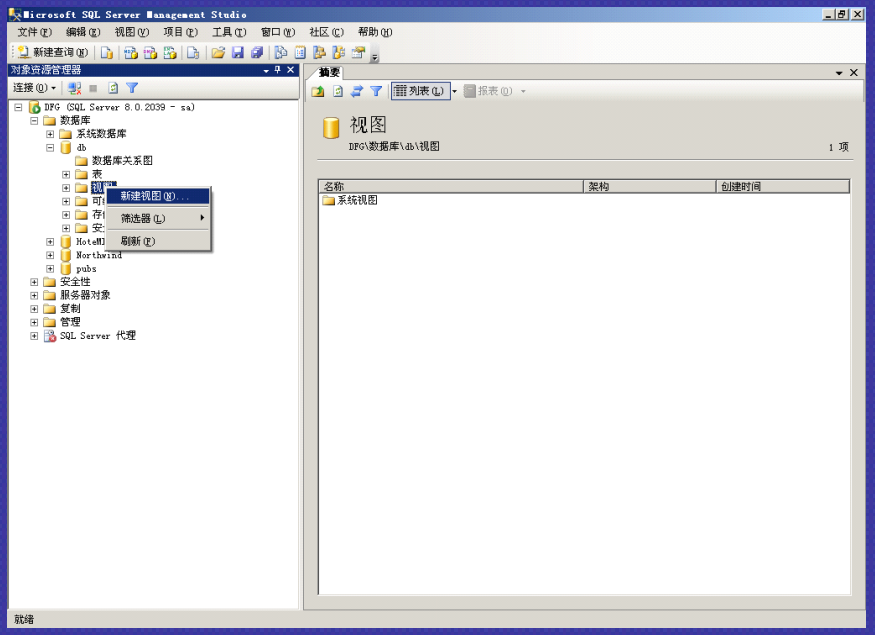


图7-1 选择新建视图选项对话框

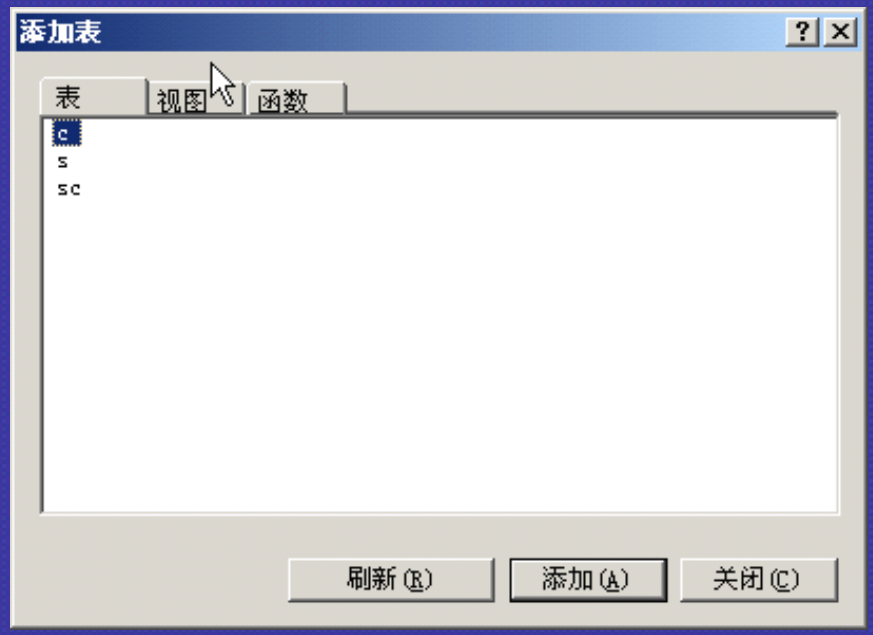


图7-2 添加表、视图、函数对话框

7.1.1利用SQL SERVER管理平台创建视图

（2）选择好创建视图所需的表、视图、函数后，通过单击字段左边的复选框选择需要的字段，如图7-3所示。单击工具栏中的“保存”按钮，或者单击鼠标右键，从快捷菜单中选择保存选项保存视图，输入视图名，即可完成视图的创建。

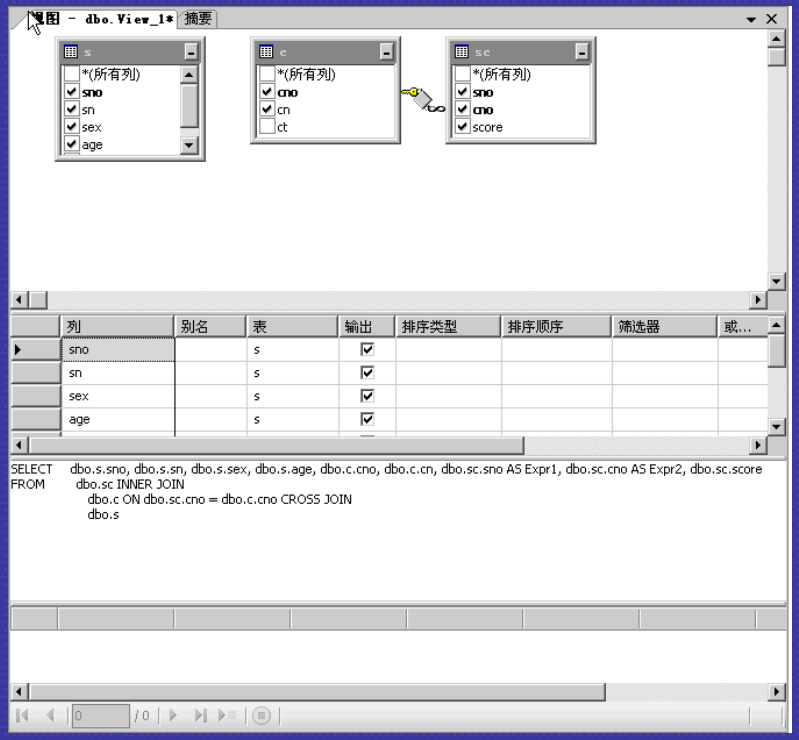


图7-3 选择视图字段对话框

7.1.2利用Transact-SQL语句中的CREATE VIEW命令创建视图

使用Transact-SQL语句中的CREATE VIEW创建视图，其语法形式如下：

```
CREATE VIEW [schema_name.] view_name [(column [,...n])]
[WITH <view_attribute> [,...n]]
AS
select_statement
[WITH CHECK OPTION]
< view_attribute > ::=
 {ENCRYPTION|SCHEMABINDING|VIEW_METADATA}
```

7.1.2利用Transact-SQL语句中的CREATE VIEW命令创建视图

例7-1　选择表s和sc中的部分字段和记录来创建一个视图，并且限制表s中的记录只能是计算机系的记录集合，视图定义为view_s。

程序清单如下：

```
create view view_s
as
select s.name,s.age,s.sex,
sc.cno,sc.score from s,sc
where s.sno=sc.sno and s.dept='计算机'
```

例7-2　创建一个视图，使之包含复杂的查询。

程序清单如下：

```
CREATE VIEW ExampleView
WITH SCHEMABINDING
AS
SELECT sno, SUM(score) AS Sumscore, COUNT(*) AS CountCol FROM sc
 GROUP BY sno
```

7.1.2利用Transact-SQL语句中的CREATE VIEW命令创建视图

例7-3　创建一个视图，使之包含字符串的运算。

程序清单如下：

```
CREATE VIEW v_shyjl(shyxh, shj, xm, jglb, phr, bzh) AS
Select
distinct shyxh,substring(shj,1,10)+""+zhi,
t01_shbshy.xm,t012_shyjl.jglb,t012_shyjl.phr,t012_shyjl.bzh
from t01_shbshy, t012_shyjl
where t012_shyjl.xm=t01_shbshy.xm
```

其中shj列是从基表中取出两列并和一空字符串相加而得到的。这些都为基表中字段间的灵活运算提供了极大方便，从而使我们可以随心所欲地定制符合自己要求的数据。

7.1.3 利用模板创建视图

使用视图模板可以很容易地创建视图，其具体操作步骤如下；

（1）在SQL SERVER管理平台中，选择view菜单中的“模板资源管理器 ”选项，如图7-4所示。

（2）在出现的“模板资源管理器”选项中选择“创建视图”选项，如图7-5所示。

（3）按照提示输入视图名称，select语句后，执行此语句，即可创建视图。

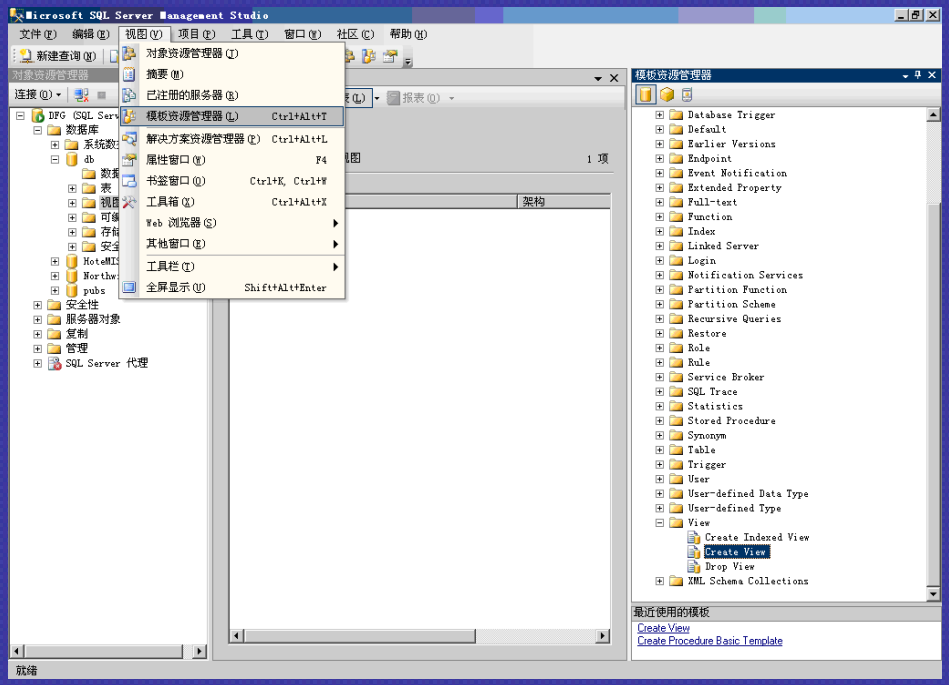


图7-4 选择工具菜单中的向导命令

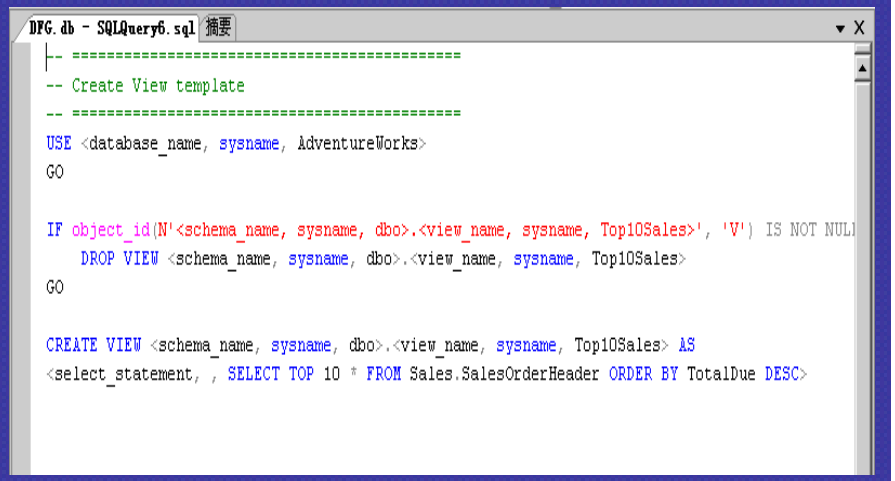


图7-5 创建视图模板

7.2 修改、删除和重命名视图

7.2.1 修改视图

7.2.2 重命名视图

7.2.3 查看视图信息、删除视图

7.2.1 修改视图

修改视图的方法有以下两种方法：

（1）在SQL SERVER管理平台中，右击要修改的视图，从弹出的快捷菜单中选择“设计视图”选项，出现视图修改对话框。该对话框与创建视图时的对话框相同，可以按照创建视图的方法修改视图。

（2）使用ALTER VIEW语句修改视图，但首先必须拥有使用视图的权限，然后才能使用ALTER VIEW语句，该语句的语法形式如下：

```
ALTER VIEW view_name
[（column[,...n]）]
[WITH ENCRYPTION]
AS
select_statement
[WITH CHECK OPTION
```

7.2.1 修改视图

例7-4　修改了视图V_employees，在该视图中增加了新的字段employees.salary，并且定义一个新的字段名称e_salary。

程序清单如下：

```
alter  view  dbo.employees
（number,name,age,e_salary）
as
select  number,name,age,salary
from  employees
where  name=’张三’
```

7.2.2 重命名视图

重命名视图方法有以下两种：

1. 在SQL SERVER管理平台中，选择要修改名称的视图，并右击该视图，从弹出的快捷菜单中选择“重命名”选项。或者在视图上再次单击，也可以修改视图的名称。接着该视图的名称变成可输入状态，可以直接输入新的视图名称。

2. 使用系统存储过程sp_rename来修改视图的名称，该过程的语法形式如下：

```
sp_rename  old_name,new_name
```

例7-5　把视图v_all重命名为v_part。

程序清单如下：

```
sp_rename  v_all,v_part
```

7.2.3 查看视图信息、删除视图

1. 查看视图信息

可以使用系统存储过程sp_help显示视图特征，使用sp_helptext显示视图在系统表中的定义，使用sp_depends显示该视图所依赖的对象。使用SQL Server 查询分析器可以方便地显示视图属性信息，如图7-7所示。图7-8显示了使用sp_helptext存储过程显示视图的创建语句。

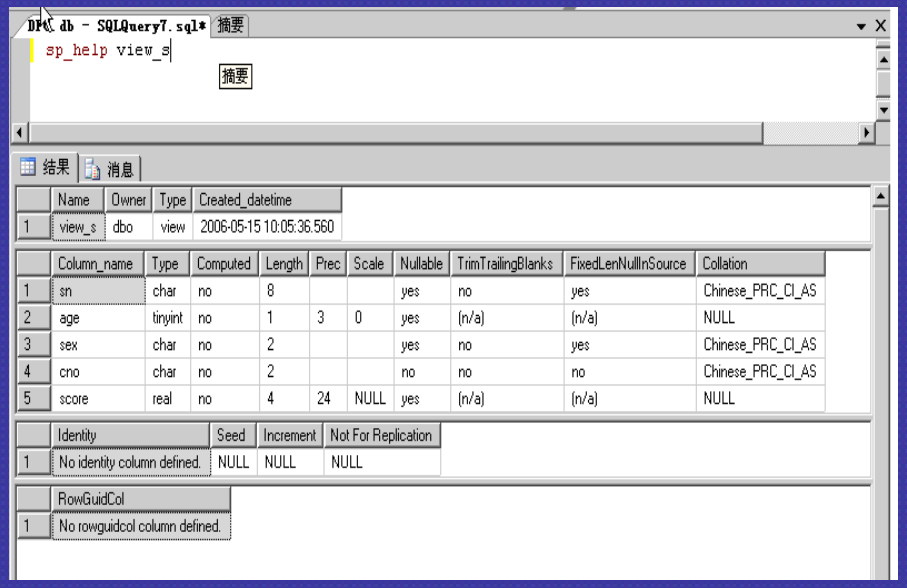


图7-7 显示视图属性对话框

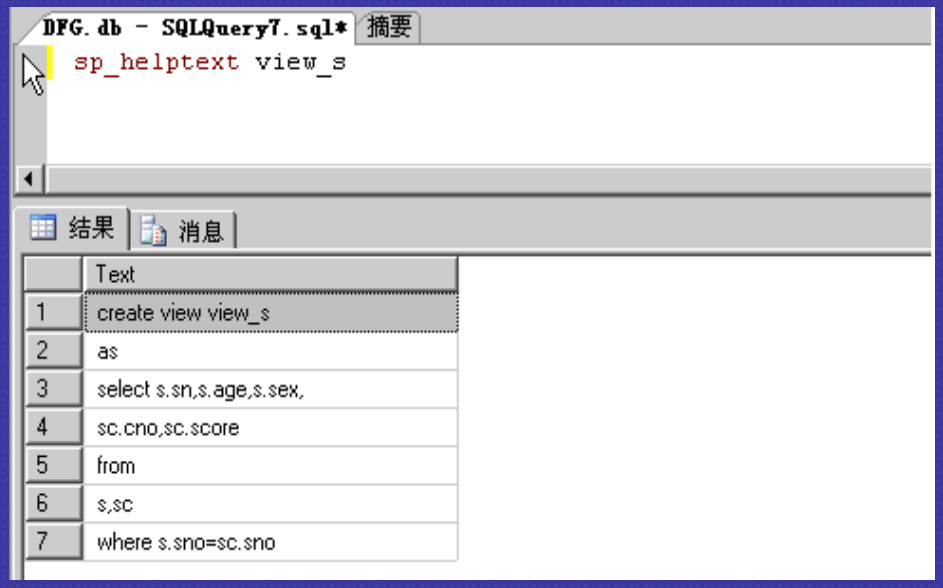


图7-8 显示视图创建语句

7.2.3 查看视图信息、删除视图

1. 查看视图信息

利用select语句或SQL SERVER管理平台可以查看视图的输出数据。在SQL SERVER管理平台中，右击某个视图的名称，从弹出的快捷菜单中选择“打开视图”选项，在SQL SERVER管理平台中就会显示该视图的输出数据，如图7-9所示。

视图 - dbo.view_s | 摘要

	s	age	sex	cno	score
	张飞	19	男	c1	89
	李鹏	18	男	c1	78
	王东	18	男	c1	67
▶	王东	18	男	c2	78
*	NULL	NULL	NULL	NULL	NULL

图7-9 视图输出数据窗口

7.2.3 查看视图信息、删除视图

2. 删除视图

对于不再使用的视图，可以使用SQL SERVER管理平台或者Transact-SQL语句中的DROP VIEW命令删除它。

使用Transact-SQL语句DROP VIEW删除视图，其语法形式如下：

DROP VIEW {view_name} [,…n]

可以使用该命令同时删除多个视图，只需在要删除的各视图名称之间用逗号隔开即可。

例7-6 同时删除视图v_student和v_teacher。

程序清单如下：

```
drop view v_student,v_teacher
```

7.3 通过视图修改记录

使用视图修改数据时，需要注意以下几点：

（1）修改视图中的数据时，不能同时修改两个或者多个基表，可以对基于两个或多个基表或者视图的视图进行修改，但是每次修改都只能影响一个基表。

（2）不能修改那些通过计算得到的字段，例如包含计算值或者合计函数的字段。

（3）如果在创建视图时指定了WITH CHECK OPTION选项，那么使用视图修改数据库信息时，必须保证修改后的数据满足视图定义的范围。

（4）执行UPDATE、DELETE命令时，所删除与更新的数据必须包含在视图的结果集中。

（5）如果视图引用多个表时，无法用DELETE命令删除数据，若使用UPDATE命令则应与INSERT操作一样，被更新的列必须属于同一个表。

7.3.1 插入数据记录

例7-7　创建一个基于表employees的新视图v_employees。
程序清单如下：

```
create  view  v_employees（number, name, age, sex, salary）
as
select  number, name, age, sex, salary
from  employees
where  name=’张三’
```

执行以下语句可向表employees中添加一条新的数据记录：

```
Insert  into  v_employees
Values（001,’李力’,22,’m’,2000）
```

7.3.1 插入数据记录

例7-8　首先创建一个包含限制条件的视图v_employee2，限制条件为工资>2000，然后插入了一条不满足限制条件的记录，再用SELECT语句检索视图和表。

程序清单如下：

```
create  view  v_employee2
as
select *  from  employee
where  工资>2000
go
insert  into  v_employee2
values（002,’王则’,30,’f’,1000）
go
select *  from  employee
go
select *  from  v_employee2
go
```

7.3.1 插入数据记录

例7-9　在例子7-8的基础上添加WITH CHECK OPTION选项。

程序清单如下：

```
create view v_employee3
as
select * from employee
where 工资>2000
with check option
go
insert into v_employee3
values（002,’王则’,30,’f’,1000）
go
select * from v_employee3
go
```

运行该程序将显示如下出错信息：

Server: Msg 550, Level 16, State 1, Line 1

The attempted insert or update failed because the target view either specifies WITH CHECK OPTION or spans a view that specifies WITH CHECK OPTION and one or more rows resulting from the operation did not qualify under the CHECK OPTION constraint.

The statement has been terminated.

7.3.2 更新和删除数据记录

使用视图可以更新数据记录，但应该注意的是，更新的只是数据库中的基表。使用视图删除记录，可以删除任何基表中的记录，直接利用DELETE语句删除记录即可。但应该注意，必须指定在视图中定义过的字段来删除记录。

- 例7-10　创建了一个基于表employees的视图v_employees，然后通过该视图修改表employees中的记录。

程序清单如下：

```
create  view  v_employees
as
select  *  from  employees
update  v_employees
set  name='张然'
where  name='张三'
```

- 例7-11　利用视图v_employees删除表employees中姓名为张然的记录。

程序清单如下：

```
delete  from  v_employees
where  name='张然'
```

第8章 存储过程的操作与管理

存储过程概述

存储过程是为完成特定的功能而汇集在一起的一组SQL程序语句，经编译后存储在数据库中的SQL程序。

在 SQL Server 中使用存储过程而不使用存储在客户端计算机本地的 Transact-SQL 程序的优点包括：

（1）存储过程已在服务器注册。

（2）存储过程具有安全特性（例如权限）和所有权链接，以及可以附加到它们的证书。

（3）存储过程可以强制应用程序的安全性。

（4）存储过程允许模块化程序设计。

（5）存储过程是命名代码，允许延迟绑定。

（6）存储过程可以减少网络通信流量。

8.1 创建存储过程

在SQL Server中，可以使用两种方法创建存储过程：

（1）使用创建存储过程模板创建存储过程；

（2）利用SQL Server 管理平台创建存储过程。

当创建存储过程时，需要确定存储过程的三个组成部分：

（1）所有的输入参数以及传给调用者的输出参数。

（2）被执行的针对数据库的操作语句，包括调用其他存储过程的语句。

（3）返回给调用者的状态值，以指明调用是成功还是失败。

8.1 创建存储过程

•CREATE PROCEDURE的语法形式如下：

```
CREATE {PROC|PROCEDURE}
[schema_name.]procedure_name[;number]
    [{@parameter[type_schema_name.] data_type}
    [VARYING][=default][[OUT[PUT]][,...n]
 [WITH <procedure_option> [,...n]
 [FOR REPLICATION]
AS
 {<sql_statement>[;][...n]|<method_specifier>}[;]
<procedure_option> ::=
[ENCRYPTION][RECOMPILE] EXECUTE_AS_Clause]
 <sql_statement> ::=
{[BEGIN] statements [END]}
 <method_specifier> ::= EXTERNAL NAME
assembly_name.class_name.method_name
```

8.1.1 使用模板创建存储过程

（1）在SQL Server 管理平台中，选择“视图（View）”菜单中的“模板资源资源管理器(Template Explorer)”，出现“模板资源管理器(Template Explorer)”窗口，选择“存储过程”中的“创建存储过程”选项，如图8-1所示。

（2）在文本框中可以输入创建存储过程的Transact_SQL语句，单击“执行”按钮，即可创建该存储过程。

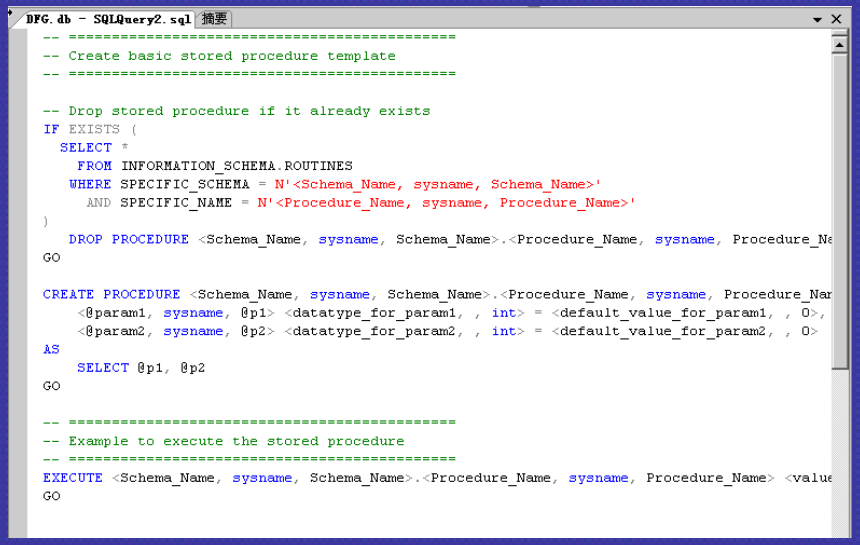


图8-1 创建存储过程模板

8.1.2使用管理平台创建存储过程

（1）在SQL Server管理平台中，展开指定的服务器和数据库，然后展开程序，右击存储过程选项，在弹出的快捷菜单中依次选择“新建→存储过程...”选项，如图8-2所示，出现创建存储过程窗口。

（2）在文本框中可以输入创建存储过程的Transact_SQL语句，单击“执行”按钮，即可创建该存储过程。

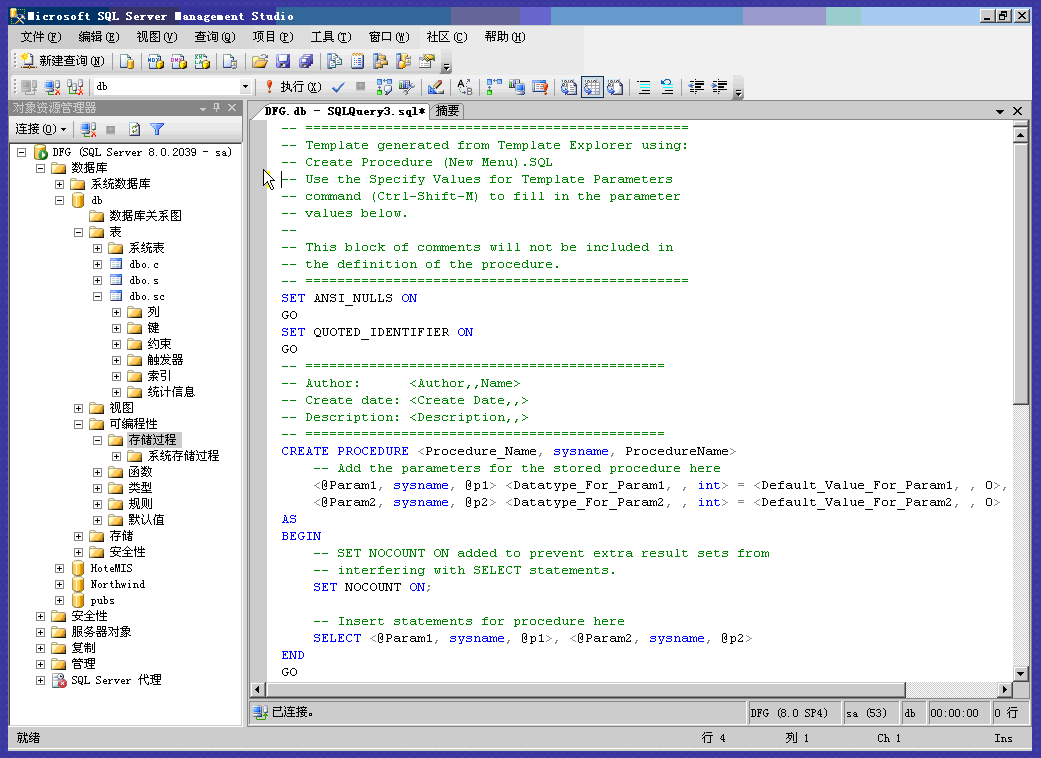


图8-2 新建存储过程

8.1.2使用管理平台创建存储过程

例8-1　创建一个带有SELECT语句的简单过程，该存储过程返回所有员工姓名，Email地址，电话等。该存储过程不使用任何参数

程序清单如下。

```
USE adventureworks
GO
CREATE PROCEDURE au_infor_all
AS
SELECT lastname, firstname, emailaddress, phone
    FROM person.contact
GO
```

8. 1. 2使用管理平台创建存储过程

例8-2 创建一个存储过程，以简化对sc表的数据添加工作，使得在执行该存储过程时，其参数值作为数据添加到表中。

程序清单如下：

```
CREATE PROCEDURE [dbo].[ pr1_sc_ins]
@Param1 char(10),@Param2 char(2),@Param3 real
AS
BEGIN
        insert into sc(sno,cno,score)
values(@Param1,@Param2,@Param3)
END
```

8.1.2使用管理平台创建存储过程

例8-3　创建一个带有参数的简单存储过程，从视图中返回指定的雇员（提供名和姓）及其职务和部门名称，该存储过程接受与传递的参数精确匹配的值

程序清单如下。

```
USE AdventureWorks;
GO
CREATE PROCEDURE GetEmployees
    @lastname varchar(40),
    @firstname varchar(20)
AS
    SELECT LastName, FirstName, JobTitle, Department
    FROM HumanResources.vEmployeeDepartment
    WHERE FirstName = @firstname AND LastName = @lastname;
GO
```

8.1.2使用管理平台创建存储过程

例8-4下面的存储过程从表person.contact中返回指定的一些员工姓名及其电话。该存储过程对传递的参数进行模式匹配。如果没有提供参数，则使用预设的默认值（姓氏以字母D开头）

程序清单如下。

```
USE AdventureWorks;
GO
CREATE PROCEDURE au_infor2
@lastname varchar(40) = 'D%', @firstname varchar(20) = '%'
AS
SELECT firstname, lastname, phone
FROM person.contact
WHERE firstname LIKE @firstname AND lastname LIKE @lastname
GO
```

8.1.2使用管理平台创建存储过程

例8-5以下示例显示有一个输入参数和一个输出参数的存储过程。存储过程中的第一个参数@sname将接收由调用程序指定的输入值(学生姓名)，第二个参数@sscore（成绩）将用于将该值返回调用程序。SELECT 语句使用@sname参数获取正确的@sscore值，并将该值分配给输出参数。

程序清单如下：

```
CREATE PROCEDURE s_score
@sname char(8),@sscore real output
AS
SELECT @sscore =score from sc join s on s.sno=sc.sno
where  sn=@sname
GO
```

8.1.3 执行存储过程

- 可以使用 Transact-SQL EXECUTE 语句来运行存储过程。存储过程与函数不同，因为存储过程不返回取代其名称的值，也不能直接在表达式中使用。
- 执行存储过程必须具有执行存储过程的权限许可，才可以直接执行存储过程，直接执行存储过程可以使用EXECUTE命令来执行，语法形式如下：

```
[[EXEC[UTE]]
        {
            [@return_status=]
                {procedure_name[;number]|@procedure_name_var}
     [[@parameter=]{value|@variable[OUTPUT]|[DEFAULT]}
          [,...n]
    [ WITH RECOMPILE ]
```

8.1.3 执行存储过程

例8-6 执行存储过程au_infor_all。

au_infor_all 存储过程可以通过以下方法执行：

EXECUTE（EXEC） au_infor_all

例8-7 使用 EXECUTE 命令传递参数，执行例8-2定义的存储过程pr1_sc_ins。

sc_ins存储过程可以通过以下方法执行：

EXEC pr1_sc_ins ‘3130040101’,’c1’,85

当然，在执行过程中变量可以显式命名：

EXEC sc_ins @Param1=’ 3130040101’,@Param2=’c1’, @Param3=85

例8-8 执行例8-3定义的存储过程GetEmployees 。

GetEmployees存储过程可以通过以下方法执行：

EXECUTE（EXEC） GetEmployees 'Dull', 'Ann' 或者

EXECUTE（EXEC） GetEmployees @lastname = 'Dull', @firstname = 'Ann' 或者

EXECUTE（EXEC） GetEmployees @firstname = 'Ann', @lastname = 'Dull'

8. 2查看、修改和删除存储过程

8.2.1 查看存储过程

8.2.2 修改存储过程

8.2.3 重命名和删除存储过程

8.2.1 查看存储过程

（1）使用SQL Server管理平台查看用户创建的存储过程。

在SQL Server管理平台中，展开指定的服务器和数据库，选择并依次展开“程序→存储过程”，然后右击要查看的存储过程名称，如图8-3所示，从弹出的快捷菜单中，选择“创建存储过程脚本为→CREATE到→新查询编辑器窗口”，则可以看到存储过程的源代码。

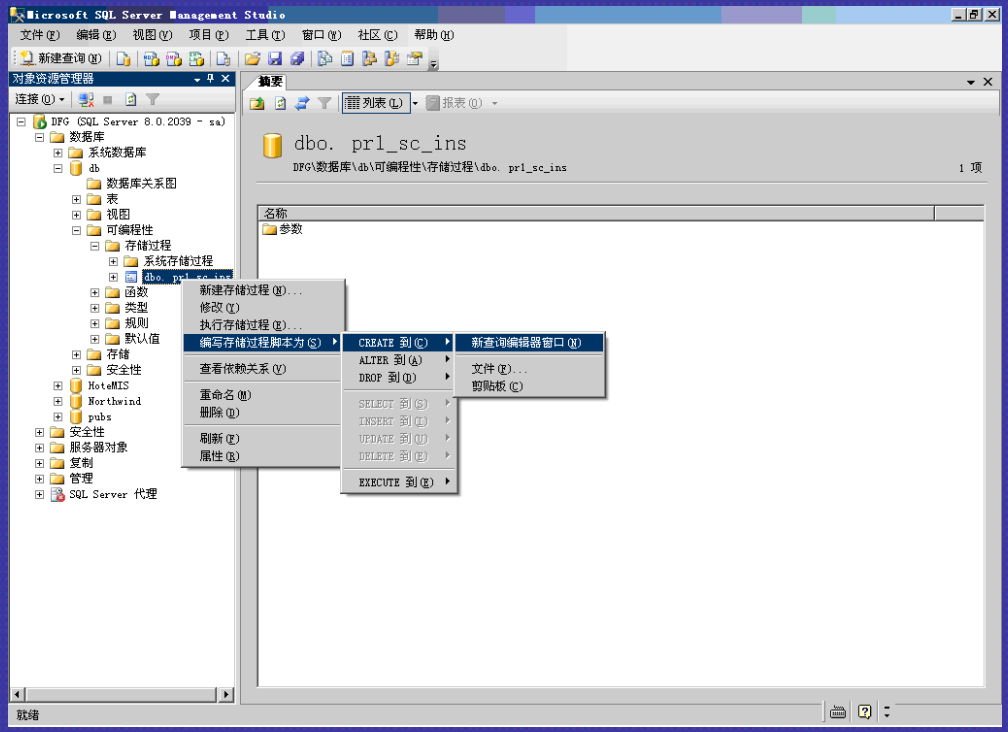


图8-3 查看存储过程

8.2.1 查看存储过程

（2）使用系统存储过程来查看用户创建的存储过程 。

可供使用的系统存储过程及其语法形式如下：

•sp_help，用于显示存储过程的参数及其数据类型，其语法为：

sp_help [[@objname=] name]，参数name为要查看的存储过程的名称。

•sp_helptext，用于显示存储过程的源代码，其语法为：

sp_helptext [[@objname=] name]，参数name为要查看的存储过程的名称。

•sp_depends，用于显示和存储过程相关的数据库对象，其语法为：

sp_depends [@objname=]’object’，参数object为要查看依赖关系的存储过程的名称。

•sp_stored_procedures，用于返回当前数据库中的存储过程列表，其语法为：

```
sp_stored_procedures[[@sp_name=]'name']
        [,[@sp_owner=]'owner']
        [,[@sp_qualifier =] 'qualifier']
```

其中，[@sp_name =] 'name' 用于指定返回目录信息的过程名；[@sp_owner =] 'owner' 用于指定过程所有者的名称；[@qualifier =] 'qualifier' 用于指定过程限定符的名称。

8.2.2 修改存储过程

•存储过程可以根据用户的要求或者基表定义的改变而改变。使用ALTER PROCEDURE语句可以更改先前通过执行 CREATE PROCEDURE 语句创建的过程，但不会更改权限，也不影响相关的存储过程或触发器。

•修改存储过程语法形式如下：

```
ALTER PROC[EDURE] procedure_name[;number]
  [{@parameter data_type}
    [VARYING][=default][OUTPUT]][,...n]
[WITH
 {RECOMPILE|ENCRYPTION|RECOMPILE,ENCRYPTION}]
[FOR REPLICATION]
 AS
 sql_statement [ ...n ]
```

8.2.2 修改存储过程

例8-9　创建了一个名为proc_person 的存储过程，该存储过程包含姓名和Email地址信息。然后，用ALTER PROCEDURE重新定义了该存储过程，使之只包含姓名信息，并使用ENCRYPTION关键字使之无法通过查看syscomments表来查看存储过程的内容。

程序清单如下。

```
USE adventureworks
GO
/*创建一个存储过程，该存储过程包含姓名和Email地址信息*/
CREATE PROCEDURE proc_person
AS
SELECT firstname, lastname, emailaddress
FROM person.contact
ORDER BY lastname, firstname
GO
```

8.2.2 修改存储过程

下面对该存储过程进行重新定义。使之只包含姓名信息，并使用ENCRYPTION关键字使之无法通过查看syscomments表来查看存储过程的内容。

程序清单如下：

```
ALTER PROCEDURE proc_person
WITH ENCRYPTION
AS
SELECT firstname, lastname
FROM person.contact
ORDER BY lastname, firstname
GO
```

8.2.3 重命名和删除存储过程

1. 重命名存储过程

修改存储过程的名称可以使用系统存储过程sp_rename，其语法形式如下：

sp_rename 原存储过程名称，新存储过程名称

另外，通过SQL Server管理平台也可以修改存储过程的名称。在SQL Server管理平台中，右击要操作的存储过程名称，从弹出的快捷菜单中选择“重命名”选项，当存储过程名称变成可输入状态时，就可以直接修改该存储过程的名称。

2. 删除存储过程

删除存储过程可以使用DROP命令，DROP命令可以将一个或者多个存储过程或者存储过程组从当前数据库中删除，其语法形式如下：

drop procedure {procedure}[,…n]

当然，利用SQL Server管理平台也可以很方便地删除存储过程。在SQL Server管理平台中，右击要删除的存储过程，从弹出的快捷菜单中选择“删除”选项，则会弹出除去对象对话框，在该对话框中，单击“确定”按钮，即可完成删除操作。单击“显示相关性”按钮，则可以在删除前查看与该存储过程有依赖关系的其他数据库对象名称。

第9章 触发器的操作与管理

触发器概述

•触发器是一种特殊的存储过程，它在执行语言事件时自动生效。

•SQL Server2005 包括两大类触发器：DML 触发器和 DDL 触发器。

（1）DML 触发器在数据库中发生数据操作语言 (DML) 事件时将启用。DML 事件包括在指定表或视图中修改数据的 INSERT 语句、UPDATE 语句或 DELETE 语句。DML 触发器可以查询其他表，还可以包含复杂的 Transact-SQL 语句。将触发器和触发它的语句作为可在触发器内回滚的单个事务对待。如果检测到错误（例如，磁盘空间不足），则整个事务即自动回滚。

（2）DDL 触发器是 SQL Server 2005 的新增功能。当服务器或数据库中发生数据定义语言 (DDL) 事件时将调用这些触发器。

9.1 DML触发器的创建和应用

- 当数据库中发生数据操作语言 (DML) 事件时将调用 DML 触发器。从而确保对数据的处理必须符合由这些SQL语句所定义的规则。

- DML 触发器的主要优点如下：

（1）DML 触发器可通过数据库中的相关表实现级联更改。例如，可以在 titles 表的 title_id 列上写入一个删除触发器，以使其他表中的各匹配行采取删除操作。该触发器用 title_id 列作为惟一键，在 titleauthor、sales 及 roysched 表中对各匹配行进行定位。

（2）DML 触发器可以防止恶意或错误的 INSERT、UPDATE 以及 DELETE 操作，并强制执行比 CHECK 约束定义的限制更为复杂的其他限制。与 CHECK 约束不同，DML 触发器可以引用其他表中的列。

（3）DML 触发器可以评估数据修改前后表的状态，并根据该差异采取措施。

9.1.1 DML触发器创建

当创建一个触发器时必须指定如下选项：

（1）名称；

（2）在其上定义触发器的表；

（3）触发器将何时激发；

（4）激活触发器的数据修改语句，有效选项为INSERT、UPDATE 或 DELETE，多个数据修改语句可激活同一个触发器；

（5）执行触发操作的编程语句。

9.1.1 DML触发器创建

- DML 触发器使用 deleted 和 inserted 逻辑表。它们在结构上和触发器所在的表的结构相同，SQL Server会自动创建和管理这些表。可以使用这两个临时的驻留内存的表测试某些数据修改的效果及设置触发器操作的条件。
- Deleted表用于存储delete，update语句所影响的行的副本。在执行delete或update语句时，行从触发器表中删除，并传输到deleted表中。
- Inserted表用于存储Insert或update语句所影响的行的副本，在一个插入或更新事务处理中，新建的行被同时添加到Inserted表和触发器表中。Inserted表中的行是触发器表中新行的副本。

9.1.1 DML触发器创建

使用SQL Server管理平台创建触发器的过程如下：

在SQL Server管理平台中，展开指定的服务器和数据库项，然后展开表，选择并展开要在其上创建触发器的表，如图9-1所示，右击触发器选项，从弹出的快捷菜单中选择“新建触发器”选项，则会出现触发器创建窗口，如图9-2所示。最后，单击“执行”按钮，即可成功创建触发器。

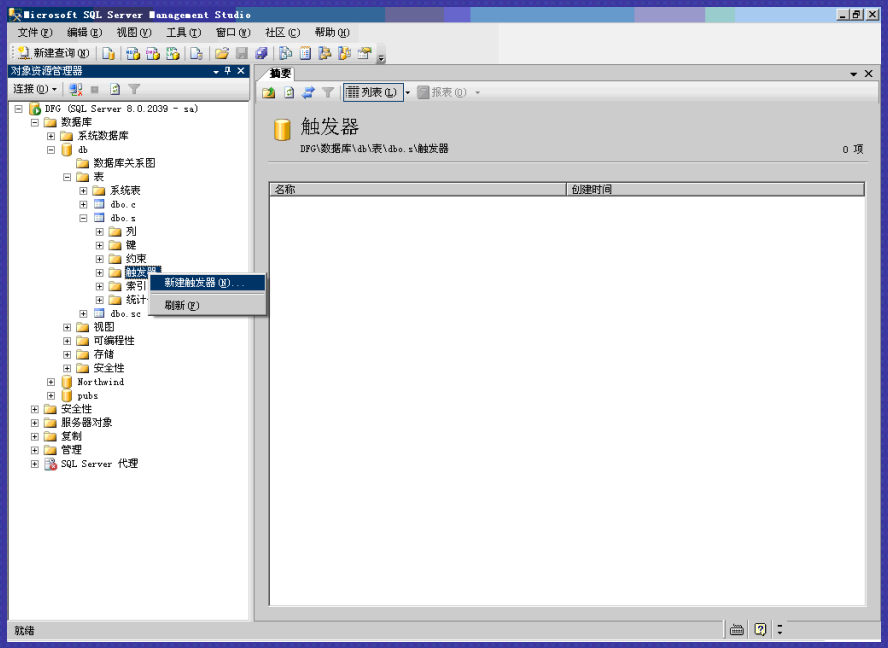


图9-1 新建触发器对话框

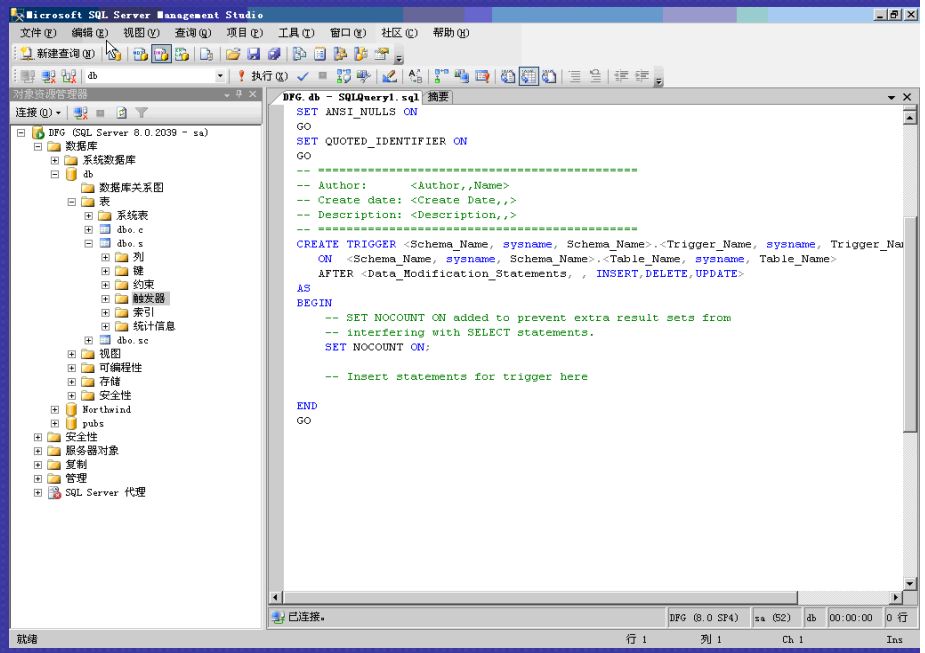


图9-2 新建触发器窗口

9.1.1 DML触发器创建

•使用**CREATE TRIGGER**命令创建**DML**触发器的语法形式如下：

```
CREATE TRIGGER [schema_name.]trigger_name
ON {table|view}
[WITH [ENCRYPTION] EXECUTE AS Clause][,...n]]
 {FOR|AFTER|INSTEAD OF} {[INSERT] [,] [UPDATE] [,] [DELETE]}
  [WITH APPEND]
[NOT FOR REPLICATION]
AS
 {sql_statement [;] [...n]|EXTERNAL NAME <method specifier [;]>}
<method_specifier> ::= assembly_name.class_name.method_name
```

9.1.1 DML触发器创建

•例9-1　示例说明inserted，deleted表的作用。执行结果如右图。

程序清单如下：

```
create table sc
 (sno  char(10),
  cno  char(2),
  score  real)
Go
CREATE TRIGGER tr1
ON sc
FOR INSERT, UPDATE, DELETE
AS
PRINT ‘inserted表：’
Select * from inserted
PRINT ‘deleted表：’
Select * from deleted
Go
```

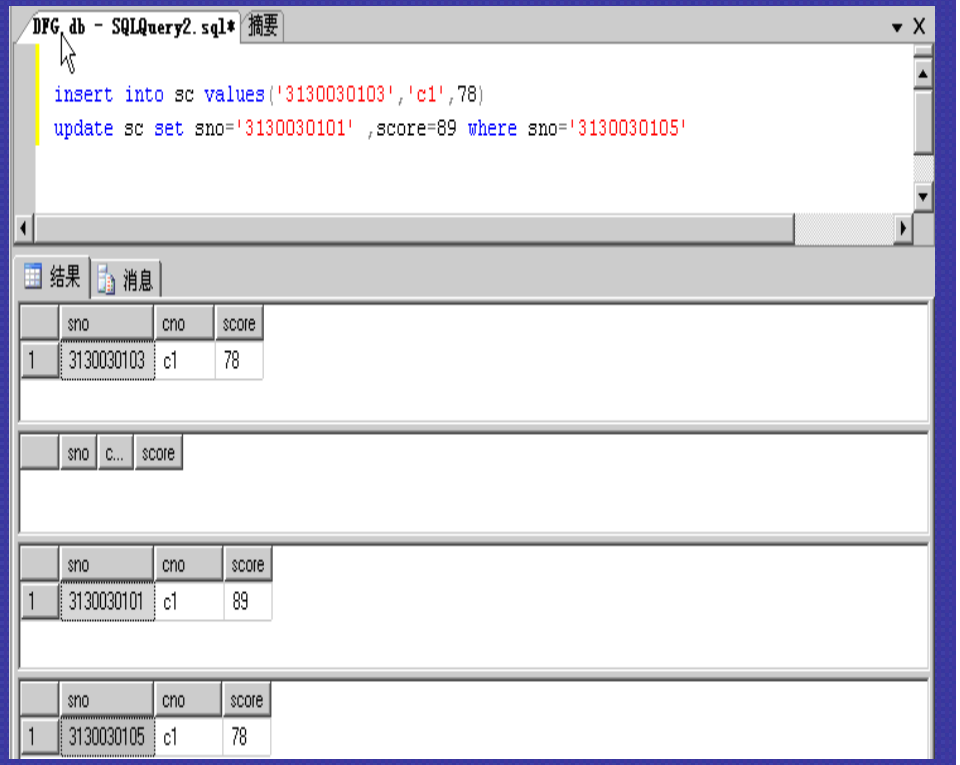


图9-3 触发器的执行结果

9.1.1 DML触发器创建

•例**9-2** 创建一个触发器，在 **s** 表上创建一个插入、更新类型的触发器。

程序清单如下：

```
CREATE TRIGGER tr_s
ON s
FOR INSERT, UPDATE
AS
Begin
DECLARE @bh varchar(6)
SELECT @bh =inserted.sno FROM inserted /*获取插入或更新操作时的新值（学号）*/
End
```

9.1.2 DML触发器的应用

1. 使用INSERT触发器

INSERT触发器通常被用来更新时间标记字段，或者验证被触发器监控的字段中数据满足要求的标准，以确保数据的完整性。

例9-3建立一个触发器，当向sc表中添加数据时，如果添加的数据与s表中的数据不匹配（没有对应的学号），则将此数据删除。

程序清单如下：

```
CREATE TRIGGER sc_ins ON sc
FOR INSERT
AS
BEGIN
DECLARE @bh char(5)
Select @bh=Inserted.sno from Inserted
If not exists(select sno from s where s.sno=@bh)
Delete sc where sno=@bh
END
```

9.1.2 DML触发器的应用

1. 使用**INSERT**触发器

例**9-4**创建一个触发器，当插入或更新成绩列时，该触发器检查插入的数据是否处于设定的范围内。

程序清单如下：

```
CREATE TRIGGER sc_insupd
ON sc
FOR INSERT, UPDATE
AS
DECLARE @cj int,
SELECT @cj=inserted.score from inserted
IF （@cj<0 or @cj > 100）
BEGIN
  RAISERROR （'成绩的取值必须在0到100之间', 16, 1）
  ROLLBACK TRANSACTION
END
```

9.1.2 DML触发器的应用

2. 使用UPDATE触发器

•当在一个有**UPDATE**触发器的表中修改记录时，表中原来的记录被移动到删除表中，修改过的记录插入到了插入表中，触发器可以参考删除表和插入表以及被修改的表，以确定如何完成数据库操作。

•例**9-5**　创建一个修改触发器，该触发器防止用户修改表**s**的入学成绩。

程序清单如下：

```
create trigger tri_s_upd
on s
for update
as
if update（escore）
begin
raiserror（‘不能修改入学成绩’,16,10）
rollback transaction
end
go
```

9.1.2 DML触发器的应用

2.使用UPDATE触发

•例9-6 DAS数据库由存放实时数据的数据表以及存放历史数据的历史表组成。由于存放实时数据的数据表不断更新，为了保存更新过的数据，在实时表和历史表之间建立了触发器。程序清单如下：

```
CREATE TRIGGER DasD_UTRIGGER ON DasD FOR UPDATE AS
BEGIN
IF Update(TV)  /*数据更新*/
BEGIN
   UPDATE DasD
      SET UT=getdate() /*更新时间*/
      FROM DasD,inserted
      WHERE DasD.ID=inserted.ID
   INSERT DasDHis(ID,TV,UT)
      SELECT inserted.ID,inserted.TV,DasD.UT FROM DasD,inserted
      WHERE  DasD.ID=inserted.ID    /*将更新过的数据送入历史库*/
END
END
```

9.1.2 DML触发器的应用

3. 使用DELETE触发器

•DELETE触发器通常用于两种情况，第一种情况是为了防止那些确实需要删除但会引起数据一致性问题的记录的删除，第二种情况是执行可删除主记录的子记录的级联删除操作。

•例9-8 建立一个与s表结构一样的表s1，当删除表s中的记录时，自动将删除掉的记录存放到s1表中。

程序清单如下：

```
CREATE TRIGGER tr_del ON s /*建立触发器
FOR DELETE        /*对表删除操作
AS insert s1 select * from deleted /*将删除掉的数据送入表s1中*/
GO
```

9.1.2 DML触发器的应用

3. 使用DELETE触发器

•例9-9当删除表s中的记录时，自动删除表sc中对应学号的记录。

程序清单如下：

```
CREATE TRIGGER tr_del_s ON s
FOR DELETE
BEGIN
DECLARE @bh char(5)
Select @bh=deleted.sno from deleted
Delete sc where sno=@bh
END
```

9.2 DDL触发器的创建和应用

- **DDL** 触发器会为响应多种数据定义语言 **(DDL)** 语句而激发。这些语句主要是以 **CREATE**、**ALTER** 和 **DROP** 开头的语句。**DDL** 触发器可用于管理任务，例如审核和控制数据库操作。
- **DDL** 触发器一般用于以下目的：

（1）防止对数据库架构进行某些更改；

（2）希望数据库中发生某种情况以响应数据库架构中的更改；

（3）要记录数据库架构中的更改或事件。

- 仅在运行触发 **DDL** 触发器的 **DDL** 语句后，**DDL** 触发器才会激发。**DDL** 触发器无法作为 **INSTEAD OF** 触发器使用。

9.2.1 创建DDL触发器

- 使用CREATE TRIGGER命令创建DDL触发器的语法形式如下：

```
CREATE TRIGGER trigger_name
ON {ALL SERVER|DATABASE}[WITH
<ddl_trigger_option> [ ,....n ]]
 {FOR|AFTER} {event_type|event_group}[,...n]
 AS {sql_statement[;] [...n]|EXTERNAL NAME <method
specifier>[;]}
```

- 其中：

```
<ddl_trigger_option>::=[ENCRYPTION] EXECUTE AS
Clause]
 <method_specifier> ::=
assembly_name.class_name.method_name
```

9.2.2 DDL触发器的应用

- 在响应当前数据库或服务器中处理的 Transact-SQL 事件时，可以激发 DDL 触发器。触发器的作用域取决于事件。
- 例9-11 使用 DDL 触发器来防止数据库中的任一表被修改或删除。

程序清单如下：

```
CREATE TRIGGER safety
ON DATABASE
FOR DROP_TABLE, ALTER_TABLE
AS
 PRINT 'You must disable Trigger "safety" to drop or alter tables!'
ROLLBACK
```

9.2.2 DDL触发器的应用

•例9-12 使用 DDL 触发器来防止在数据库中创建表。
程序清单如下：

```
CREATE TRIGGER safety
ON DATABASE
FOR CREATE_TABLE
 AS
PRINT 'CREATE TABLE Issued.'
SELECT

 EVENTDATA().value('(/EVENT_INSTANCE/TSQLCommand/Command
dText)[1]','nvarchar(max)')
RAISERROR ('New tables cannot be created in this database.', 16, 1)
ROLLBACK
```

9.3 DDL查看、修改和删除触发器

9.3.1 查看触发器

9.3.2 修改触发器

9.3.3 删除触发器

9.3.1 查看触发器

如果要显示作用于表上的触发器究竟对表有哪些操作，必须查看触发器信息。在SQL Server中，有多种方法可以查看触发器信息，其中最常用的有如下两种：

（1）使用SQL Server管理平台查看触发器信息；

（2）使用系统存储过程查看触发器。

9.3.1 查看触发器

（1）使用SQL Server管理平台查看触发器信息。

在SQL Server管理平台中，展开服务器和数据库，选择并展开表，然后展开触发器选项，右击需要查看的触发器名称，如图9-4所示，从弹出的快捷菜单中，选择“编写触发器脚本为→create到→新查询编辑器窗口”，则可以看到触发器的源代码。

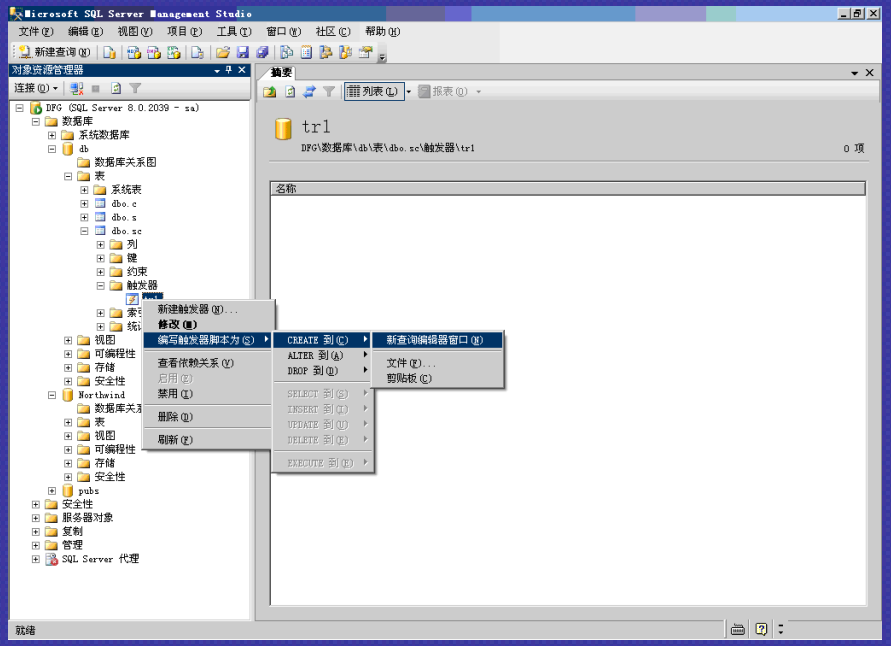


图9-4 查看触发器

9.3.1 查看触发器

（2）使用系统存储过程查看触发器。

•系统存储过程sp_help、sp_helptext和sp_depends分别提供有关触发器的不同信息。其具体用途和语法形式如下。

•sp_help：用于查看触发器的一般信息，如触发器的名称、属性、类型和创建时间。

sp_help ‘触发器名称’

•sp_helptext：用于查看触发器的正文信息。

sp_helptext ‘触发器名称’

•sp_depends：用于查看指定触发器所引用的表或者指定的表涉及到的所有触发器。

sp_depends ‘触发器名称’

sp_depends ‘表名’

9.3.2 修改触发器

•通过SQL Server管理平台、存储过程，可以修改触发器的正文和名称。

1．使用SQL Server管理平台修改触发器正文。

在管理平台中，展开指定的表，右击要修改的触发器，从弹出的快捷菜单中选择“修改”选项，则会出现触发器修改窗口，如图9-5所示。在文本框中修改触发器的SQL语句，单击“语法检查”按钮，可以检查语法是否正确，单击“执行”按钮，可以成功修改此触发器。

图9-5 触发器修改窗口

9.3.2 修改触发器

- 修改**DML**触发器的语法形式如下：

```
ALTER TRIGGER schema_name.trigger_name
 ON (table|view)
[WITH <dml_trigger_option>[,...n]]
 (FOR|AFTER|INSTEAD OF)
 {[DELETE][,][INSERT][,][UPDATE]}
[NOT FOR REPLICATION]
 AS {sql_statement[;][...n]|EXTERNAL NAME <method specifier>[;]}
 <dml_trigger_option>::=[ENCRYPTION][&IEXECUTE AS Clause >]
 <method_specifier> ::=assembly_name.class_name.method_name
```

- 修改**DDL**触发器的语法形式如下：

```
 ALTER TRIGGER trigger_name
 ON {DATABASE|ALL SERVER}[WITH <ddl_trigger_option> [,...n]]
 {FOR|AFTER}{event_type[,...n]|event_group}
AS {sql_statement[;]|EXTERNAL NAME <method specifier> [;]}
 <ddl_trigger_option>::=[ENCRYPTION][&IEXECUTE AS Clause > ]
 <method_specifier> ::=assembly_name.class_name.method_name
```

9.3.2 修改触发器

•例9-13 修改触发器。

程序清单如下：

```
CREATE TRIGGER s_reminder
ON S
 WITH ENCRYPTION
 AFTER INSERT, UPDATE
AS
RAISERROR ('不能对该表执行添加、更新操作', 16, 10)
ROLLBACK
GO
```

• -- 下面修改触发器.

```
ALTER TRIGGER s_reminder
 ON S
 AFTER INSERT
 AS
 RAISERROR ('不能对该表执行添加操作', 16, 10)
 ROLLBACK
GO
```

9.3.2 修改触发器

2. 使用**sp_rename**命令修改触发器的名称。

sp_rename命令的语法形式如下：

sp_rename oldname,newname

9.3.3 删除触发器

•由于某种原因，需要从表中删除触发器或者需要使用新的触发器，这就必须首先删除旧的触发器。只有触发器所有者才有权删除触发器。删除已创建的触发器有三种方法：

（1）使用系统命令**DROP TRIGGER**删除指定的触发器。其语法形式如下：

DROP TRIGGER { trigger } [,...n]

（2）删除触发器所在的表。删除表时，**SQL Server**将会自动删除与该表相关的触发器。

（3）在**SQL Server**管理平台中，展开指定的服务器和数据库，选择并展开指定的表，右击要删除的触发器，从弹出的快捷菜单中选择“删除”选项，即可删除该触发器。

第10章 SQL SERVER权限管理

10.1 SQL Server权限管理策略

- 对于一个数据库管理员来说，安全性就意味着必须保证那些具有特殊数据访问权限的用户能够登录到SQL Server，并且能够访问数据以及对数据库对象实施各种权限范围内的操作；同时，他还要防止所有的非授权用户的非法操作。
- SQL Server提供了既有效又容易的安全管理模式，这种安全管理模式是建立在安全身份验证和访问许可两者机制上的。

10.1.1 安全身份验证

- 安全身份验证用来确认登录SQL Server的用户的登录帐号和密码的正确性，由此来验证该用户是否具有连接SQL Server的权限。任何用户在使用SQL Server数据库之前，必须经过系统的安全身份验证。
- SQL Server 2005提供了两种确认用户对数据库引擎服务的验证模式：
  （1）Windows身份验证
  （2）SQL Server身份验证。

10.1.1 安全身份验证

（1）SQL Server数据库系统通常运行在Windows服务器上，而Windows作为网络操作系统，本身就具备管理登录、验证用户合法性的能力，因此Windows验证模式正是利用了这一用户安全性和帐号管理的机制，允许SQL Server可以使用Windows的用户名和口令。在这种模式下，用户只需要通过Windows的验证，就可以连接到SQL Server，而SQL Server本身也就不需要管理一套登录数据。

（2）SQL Server身份验证模式允许用户使用SQL Server安全性连接到SQL Server。在该认证模式下，用户在连接SQL Server时必须提供登录名和登录密码，这些登录信息存储在系统表syslogins中，与Windows的登录帐号无关。SQL Server自身执行认证处理，如果输入的登录信息与系统表syslogins中的某条记录相匹配，则表明登录成功。

10.1.1 安全身份验证

利用SQL Server管理平台可以进行认证模式的设置，步骤如下：

（1）打开Server管理平台，右击要设置认证模式的服务器，从弹出的快捷菜单中选择“属性”选项，则出现SQL Server属性对话框。

（2）在SQL Server属性对话框中选择“安全性”选项页，如图10-1所示。

（3）在“服务器身份验证”选项栏中，可以选择要设置的认证模式，同时在“登录审核”中还可以选择跟踪记录用户登录时的哪种信息，例如登录成功或登录失败的信息等。

（4）在“服务器代理帐户”选项栏中设置当启动并运行SQL Server时，默认的登录者中哪一位用户。

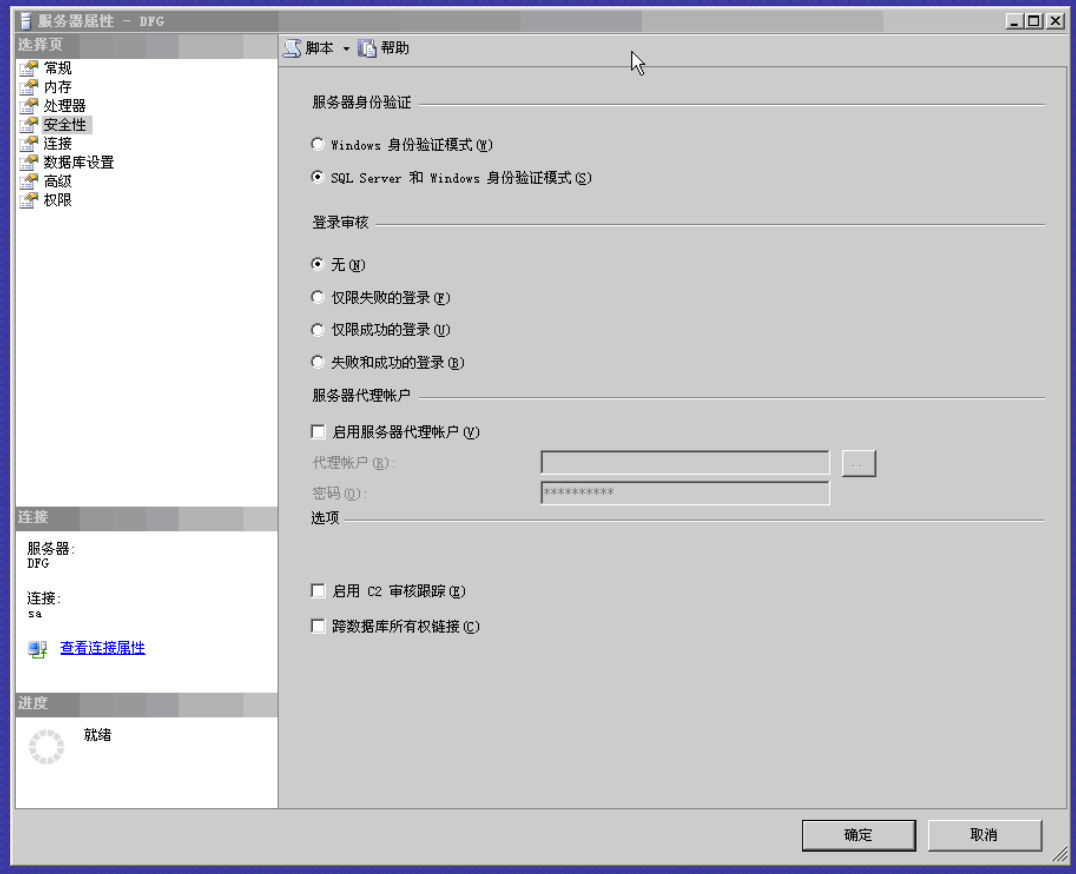


图10-1 SQL Server服务器属性页框

10.1.2 访问许可确认

通过了认证并不代表用户就能访问SQL Server中的数据，同时他还必须通过许可确认。用户只有在具有访问数据库的权限之后，才能够对服务器上的数据库进行权限许可下的各种操作，这种用户访问数据库权限的设置是通过用户帐号来实现的 。

10.2 用户权限管理

10.2.1 服务器登录帐号和用户帐号管理

10.2.2 许可（权限）管理

10.2.3 角色管理

10.2.1服务器登录帐号和用户帐号管理

1.SQL Server服务器登录管理

- 利用SQL Server管理平台可以创建、管理SQL Server登录帐号。其具体执行步骤如下：

（1）打开SQL Server管理平台，单击需要登录的服务器左边的“+”号，然后展开安全性文件夹。

（2）右击登录名（login）图标，从弹出的快捷菜单中选择“新建登录名”选项，则出现SQL Server“登录名—新建”对话框，如图10-2所示。

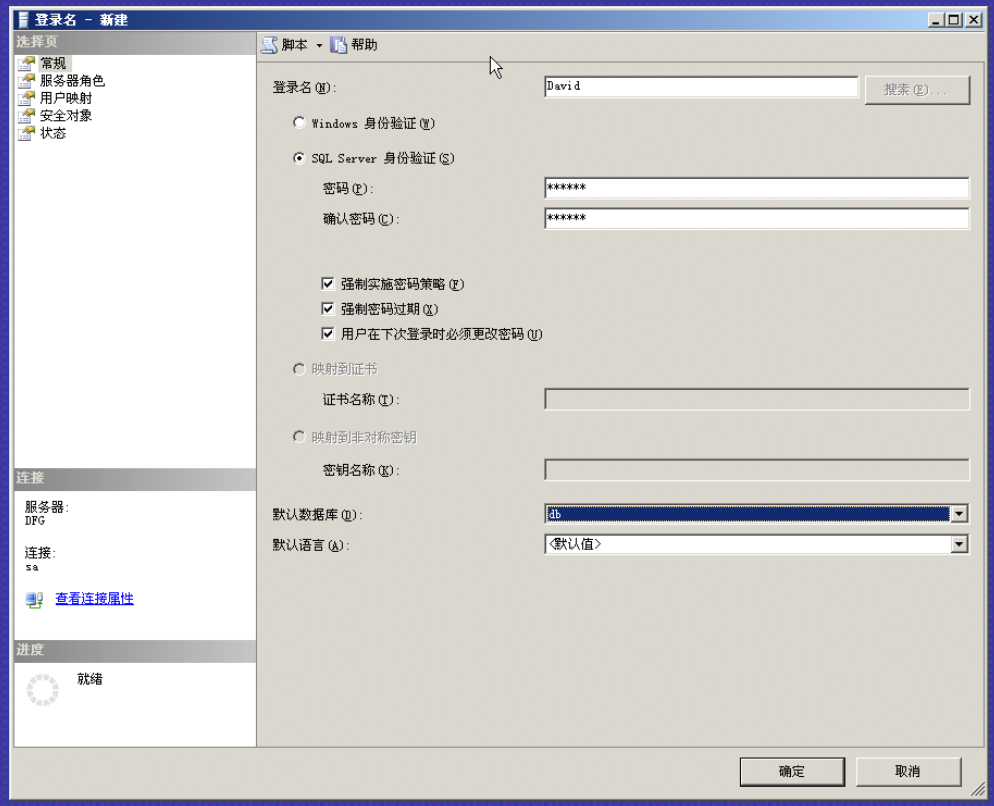


图10-2 新建登录名对话框

10.2.1服务器登录帐号和用户帐号管理

1.SQL Server服务器登录管理

（3）在“名称”文本框中输入登录名，在身份验证选项栏中选择新建的用户帐号是Windows 认证模式，或是SQL Server认证模式。

（4）选择“服务器角色”页框，如图10-3所示。在服务器角色列表框中，列出了系统的固定服务器角色。在这些固定服务器角色的左端有相应的复选框，打勾的复选框表示该登录帐号是相应的服务器角色成员。

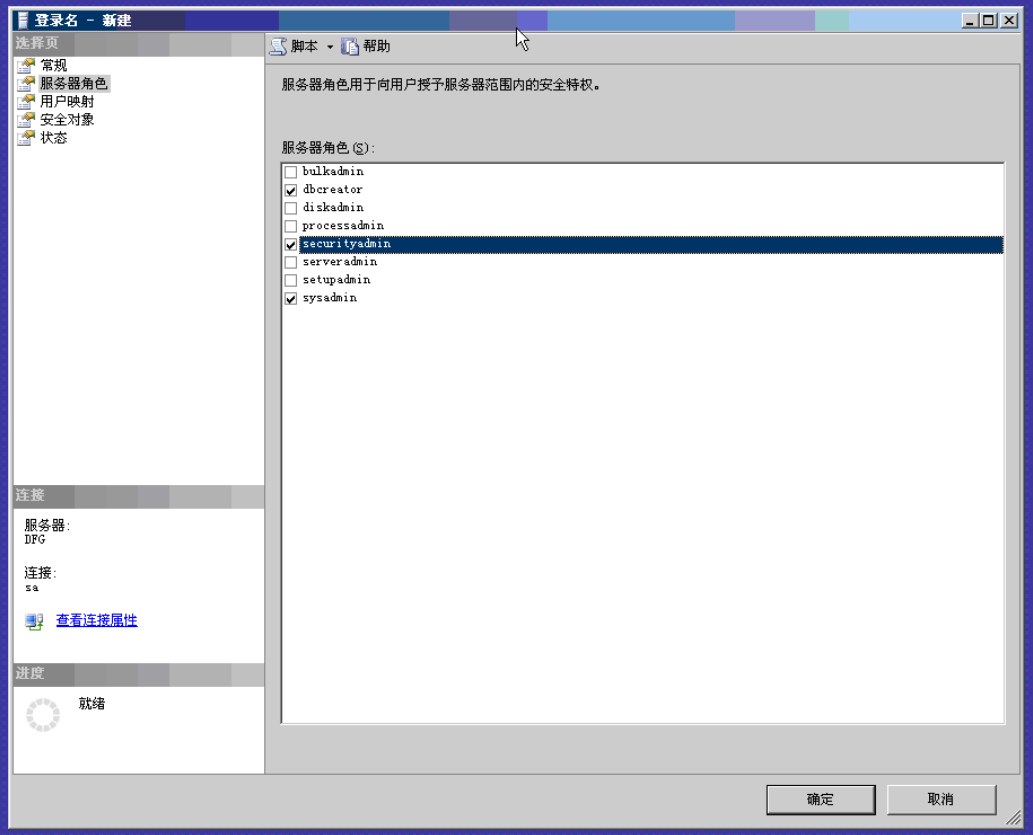


图10-3　服务器角色对话框

10.2.1服务器登录帐号和用户帐号管理

1.SQL Server服务器登录管理

（5）选择“用户映射”页框，如图10-4所示。上面的列表框列出了“映射到此登录名的用户”，单击左边的复选框设定该登录账号可以访问的数据库以及该帐号在各个数据库中对应的用户名。下面的列表框列出了相应的“数据库角色成员身份”清单，从中可以指定该帐号所属的数据库角色。

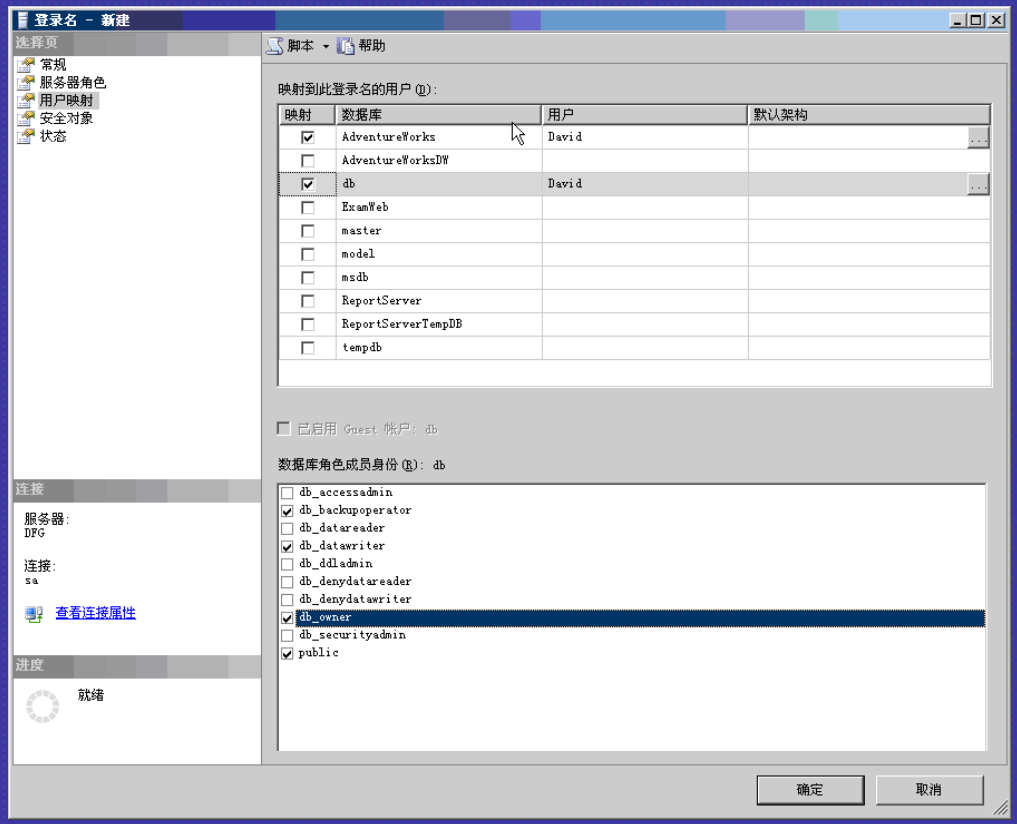


图10-4 用户映射对话框

10.2.1服务器登录帐号和用户帐号管理

1.SQL Server服务器登录管理

（6）选择“安全对象”页框，如图10-5所示。安全对象是SQL Server数据库引擎授权系统控制对其进行访问的资源。点击“添加...”按钮，可对不同类型的安全对象进行安全授予或拒绝。

（7）设置完成后，单击“确定”按钮即可完成登录帐号的创建。

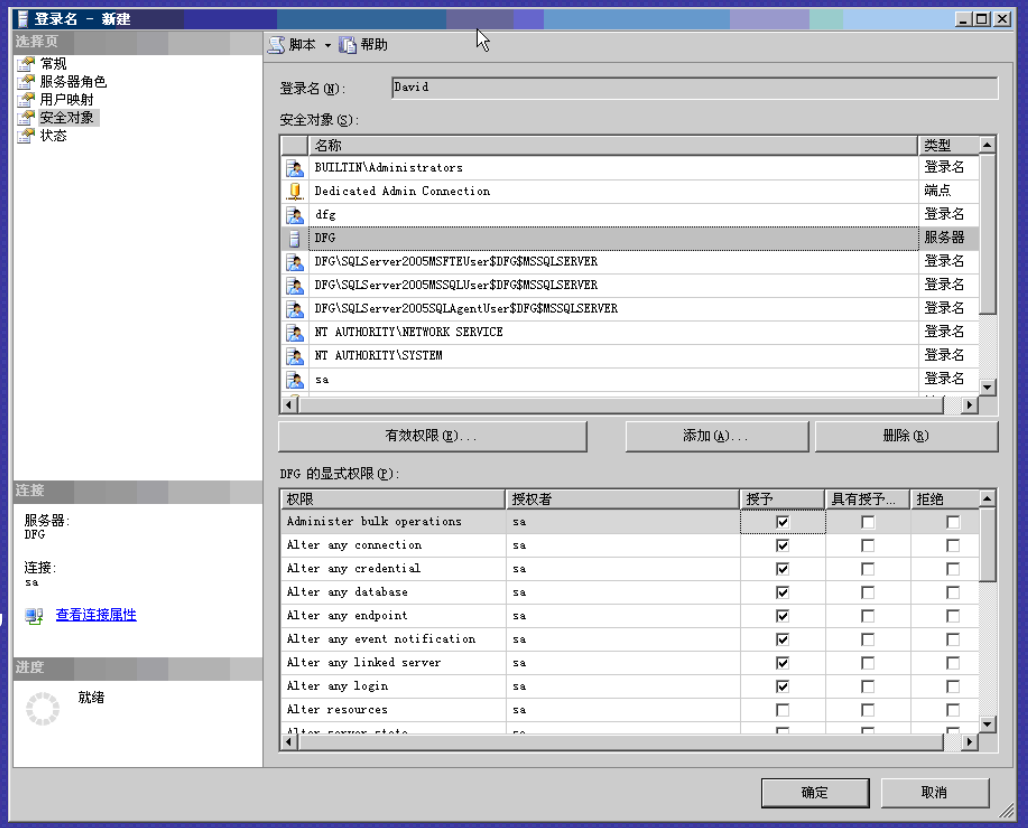


图10-5 安全对象对话框

10.2.1服务器登录帐号和用户帐号管理

2. 用户帐号管理

- 在一个数据库中，用户帐号惟一标识一个用户，用户对数据库的访问权限以及对数据库对象的所有关系都是通过用户帐号来控制的。
- 利用SQL Server管理平台可以授予SQL Server登录访问数据库的许可权限。

10.2.1服务器登录帐号和用户帐号管理

2. 用户帐号管理

- 利用SQL Server管理平台创建一个新数据库用户帐号的过程如下：
- 打开SQL Server管理平台，展开要登录的服务器和数据库文件夹，然后展开要创建用户的数据库及安全文件夹，右击用户图标，从快捷菜单中选择“新建用户”选项，则出现“数据库用户—新建”对话框，如图10-6所示。
- 在用户名框内输入数据库用户名称，在登录名选择框内选择已经创建的登录帐号，然后在下面的数据库角色成员选择框中为该用户选择数据库角色，最后单击“确定”按钮即可完成数据库用户的创建。

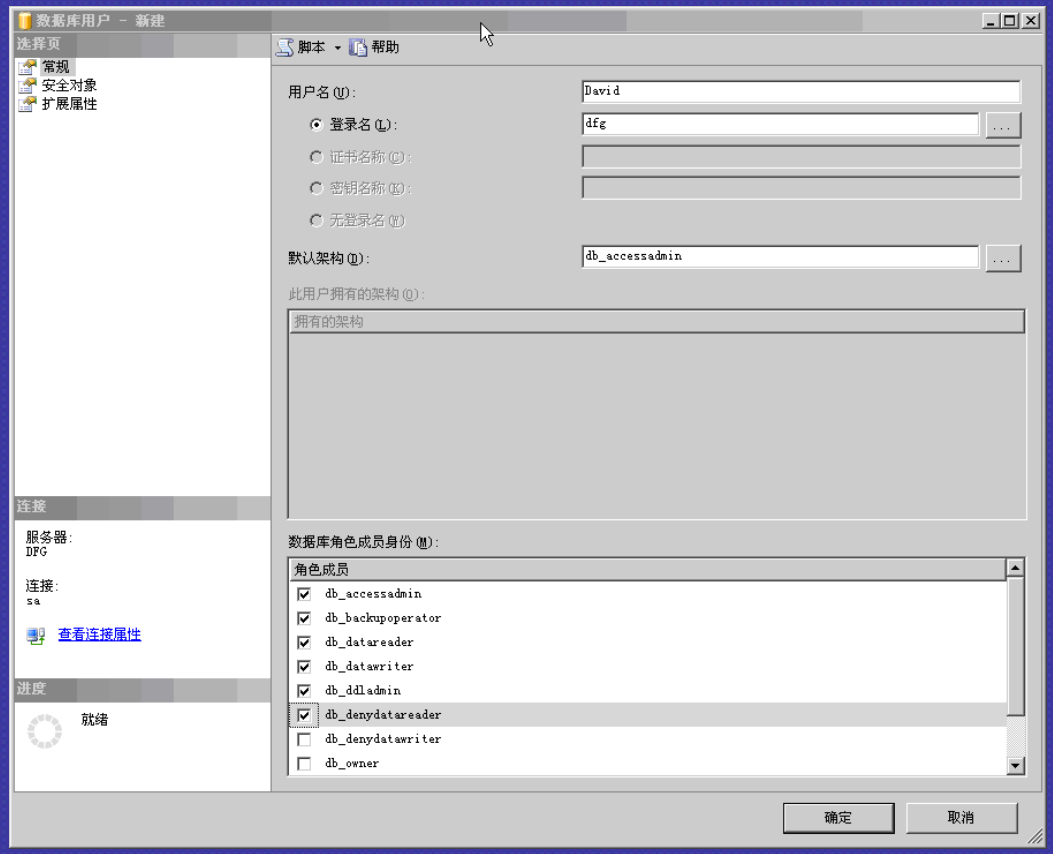


图10-6　新建数据库用户对话框

10.2.1服务器登录帐号和用户帐号管理

2. 用户帐号管理

- 同样，在SQL Server 管理平台中，也可以查看或者删除数据库用户，方法是：展开某一数据库，选中用户图标，则在右面的页框中显示当前的数据库的所有用户，如图10-7所示。要删除数据库用户，则在右面的页框中右击所要删除的数据库用户，从弹出的快捷菜单中选择“删除”选项，则会从当前的数据库中删除该数据库用户。

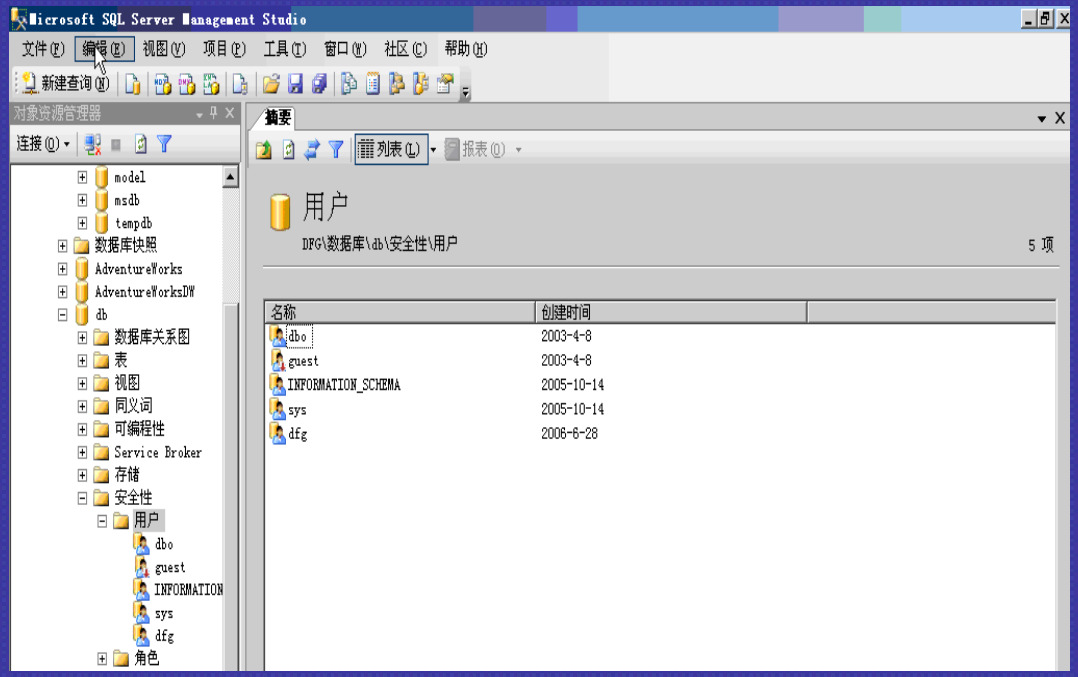


图10-7　管理数据库用户对话框

10.2.2 许可（权限）管理

- 许可用来指定授权用户可以使用的数据库对象和这些授权用户可以对这些数据库对象执行的操作。用户在登录到SQL Server之后，其用户帐号所归属的Windwos组或角色所被赋予的许可（权限）决定了该用户能够对哪些数据库对象执行哪种操作以及能够访问、修改哪些数据。在每个数据库中用户的许可独立于用户帐号和用户在数据库中的角色，每个数据库都有自己独立的许可系统。
- 在SQL Server中包括三种类型的许可：即对象许可、语句许可和预定义许可。

10.2.2 许可（权限）管理

- 对象许可表示对特定的数据库对象（即表、视图、字段和存储过程）的操作许可，它决定了能对表、视图等数据库对象执行哪些操作。如果用户想要对某一对象进行操作，其必须具有相应的操作的权限。表和视图许可用来控制用户在表和视图上执行SELECT，INSERT，UPDATE和DELETE语句的能力。字段许可用来控制用户在单个字段上执行SELECT，UPDATE和REFERENCES操作的能力。存储过程许可用来控制用户执行EXECUTE语句的能力。
- 语句许可表示对数据库的操作许可，也就是说，创建数据库或者创建数据库中的其他内容所需要的许可类型称为语句许可。这些语句通常是一些具有管理性的操作，如创建数据库、表和存储过程等。这种语句虽然仍包含有操作的对象，但这些对象在执行该语句之前并不存在于数据库中。因此，语句许可针对的是某个SQL语句，而不是数据库中已经创建的特定的数据库对象。
- 预定义许可是指系统安装以后有些用户和角色不必授权就有的许可。其中的角色包括固定服务器角色和固定数据库角色，用户包括数据库对象所有者。只有固定角色或者数据库对象所有者的成员才可以执行某些操作。执行这些操作的许可就称为预定义许可。

10.2.2 许可（权限）管理

- 许可的管理包括对许可的授权、否定和收回。在SQL Server中，可以使用SQL Server管理平台和Transaction_SQL 语句两种方式来管理许可。

1. 使用SQL Server 管理平台管理许可

- SQL Server 可通过两种途径实现对用户许可的设定：

（1）面向单一用户

（2）面向数据库对象的许可设置。

10.2.2 许可（权限）管理

1. 使用SQL Server 管理平台管理许可

（1）面向单一用户的许可设置。其具体过程如下：

- 在SQL Server管理平台中，展开服务器和数据库，单击用户图标，此时在右面的页框中将显示数据库的所有用户。在数据库用户清单中，右击要进行许可设置的用户，从弹出的快捷菜单中选择“属性”选项，则出现数据库用户属性对话框，选择“安全对象”页框，如图10-8所示。

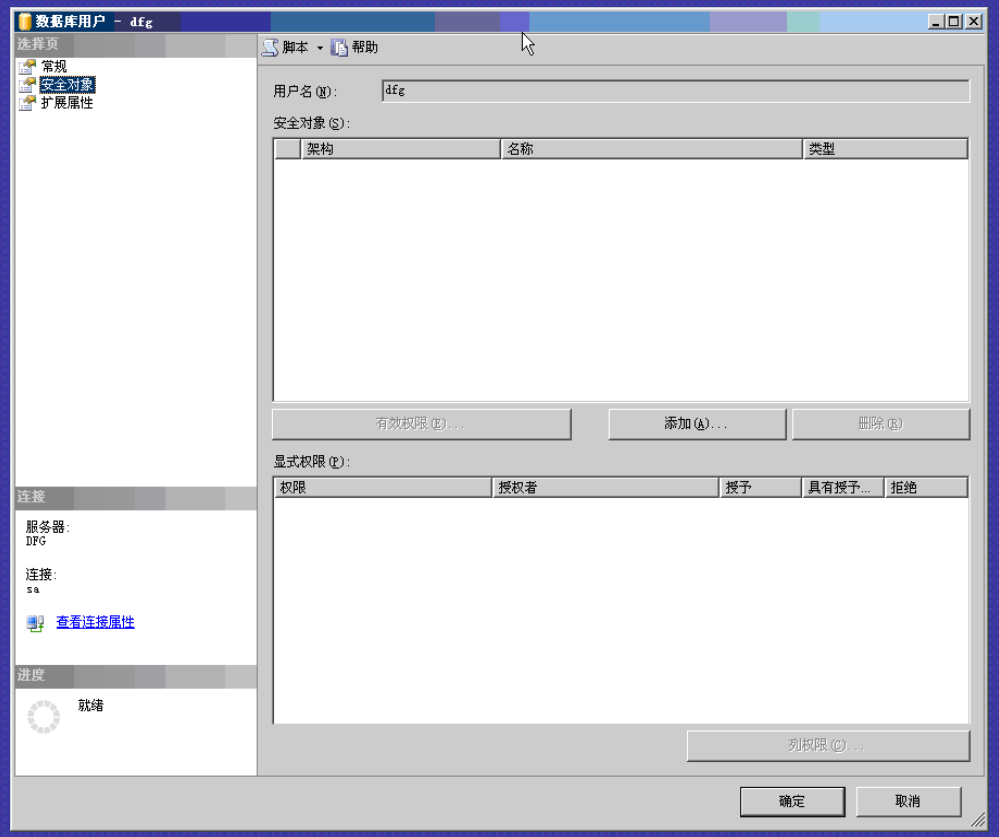


图10-8 数据库用户属性对话框

10.2.2 许可（权限）管理

1. 使用SQL Server 管理平台管理许可

（1）面向单一用户的许可设置。

•在上页对话框中单击“添加”按钮，则弹出“添加对象”对话框，如图10-9所示。选择“特定对象”单选钮后，出现如图10-10所示的对话框

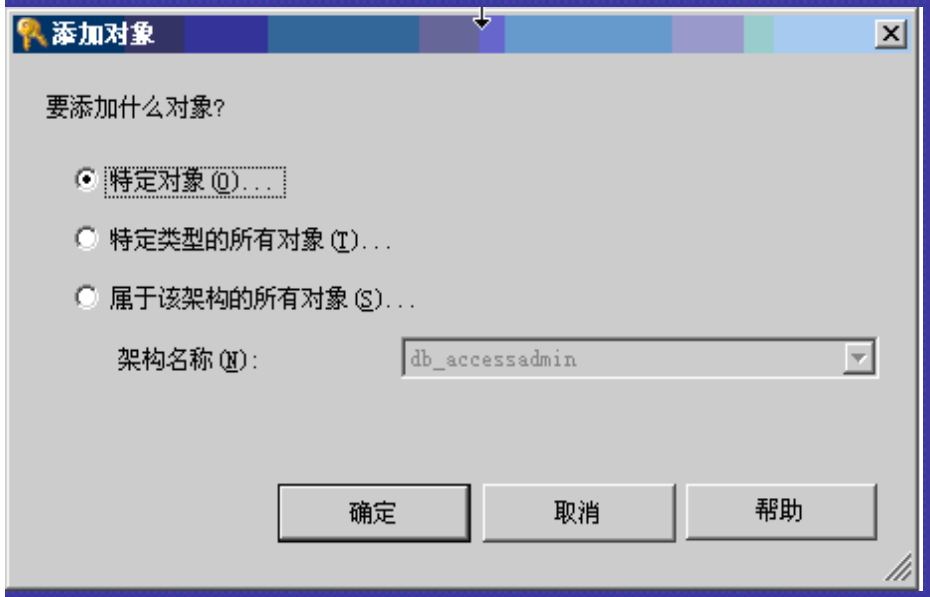


图10-9 添加对象对话框

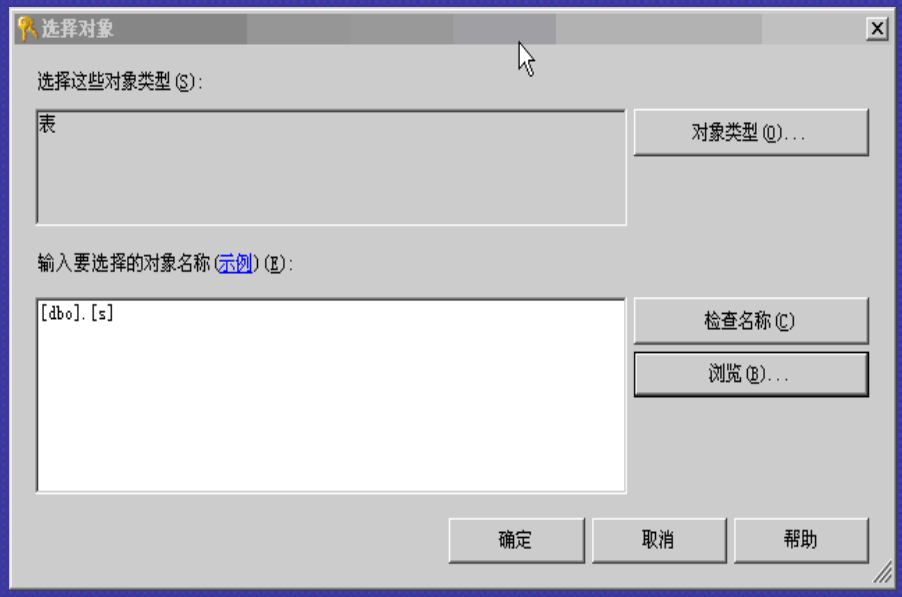


图10-10 选择对象对话框

10.2.2 许可（权限）管理

1. 使用SQL Server 管理平台管理许可

（1）面向单一用户的许可设置。

- 点击“确定”后则出现图10-11所示的对话框。在该对话框中可以进行对象许可的设置。点击对话框底部“列权限”按钮，出现如图10-12所示对话框，在该对话框中可以选择用户对哪些列具有哪些权限。最后单击“确定”按钮即可完成许可的设置。

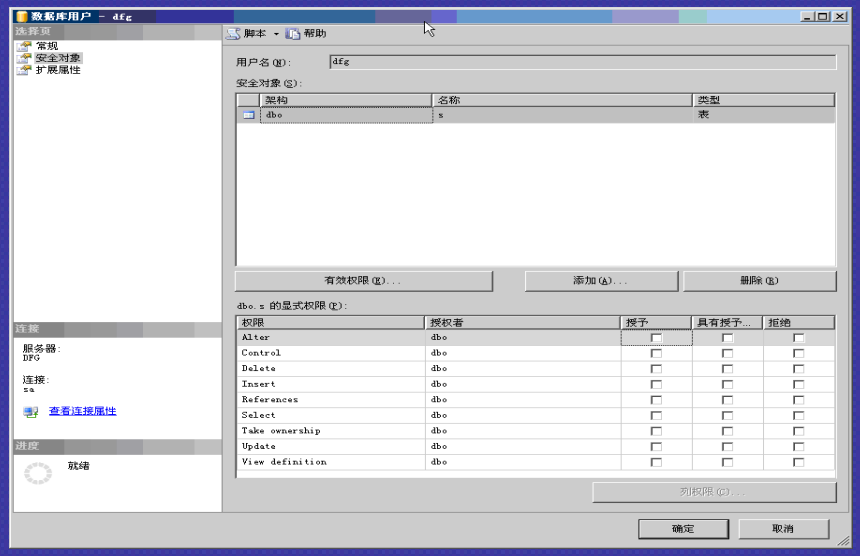


图10-11 设置对象权限对话框

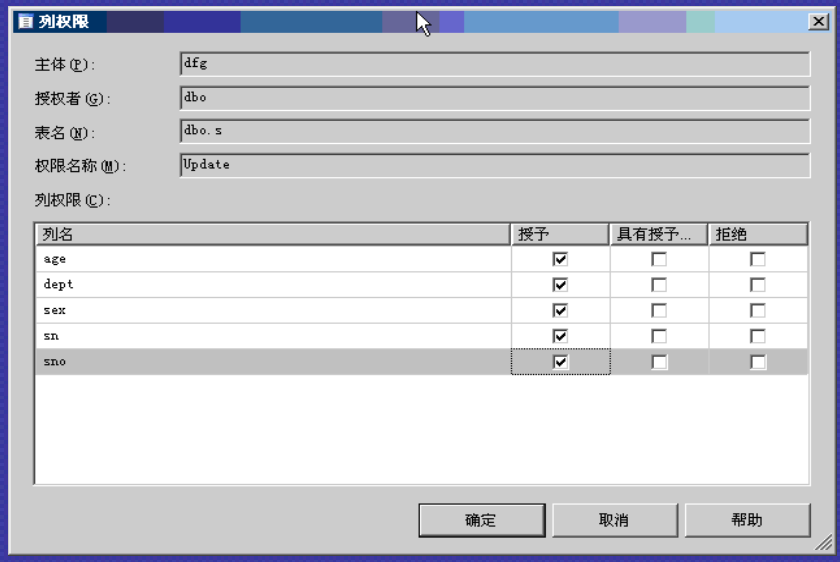


图10-12 设置列权限对话框

10.2.2 许可（权限）管理

1. 使用SQL Server管理平台管理许可

（1）面向单一用户的许可设置。

- 在如图10-13所示的数据库用户常规选项页中，如果在“数据库角色成员身份”选项栏中选择一个数据库角色，实际上就完成了数据库用户语句许可的设置。因为对于这些数据库固定角色，SQL Server已经定义了其具有哪些语句许可。

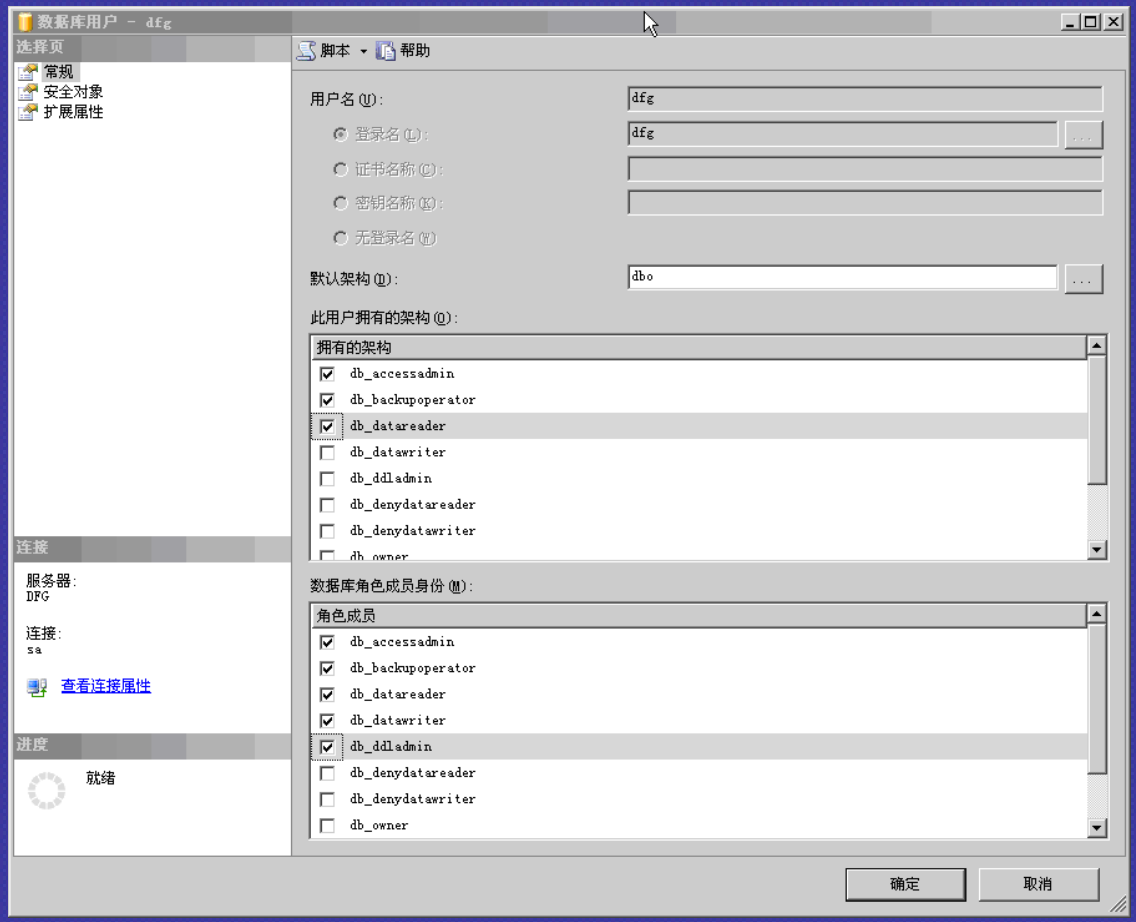


图10-13　数据库用户属性常规页框

10.2.2 许可（权限）管理

1. 使用SQL Server管理平台管理许可

（2）面向数据库对象的许可设置。

- 在SQL Server 管理平台中，展开服务器和数据库，然后选择需要设置的用户对象，即表、视图、存储过程等，在右面的页框中选择要进行许可设置的对象，右击该对象，从弹出的快捷菜单中选择“属性”选项，出现对象属性对话框，在该对话框中选择“权限”页框，单击“添加”按钮设置好相应的对象许可后，如图10-14所示，单击“确定”按钮即可完成数据库对象的许可设置。

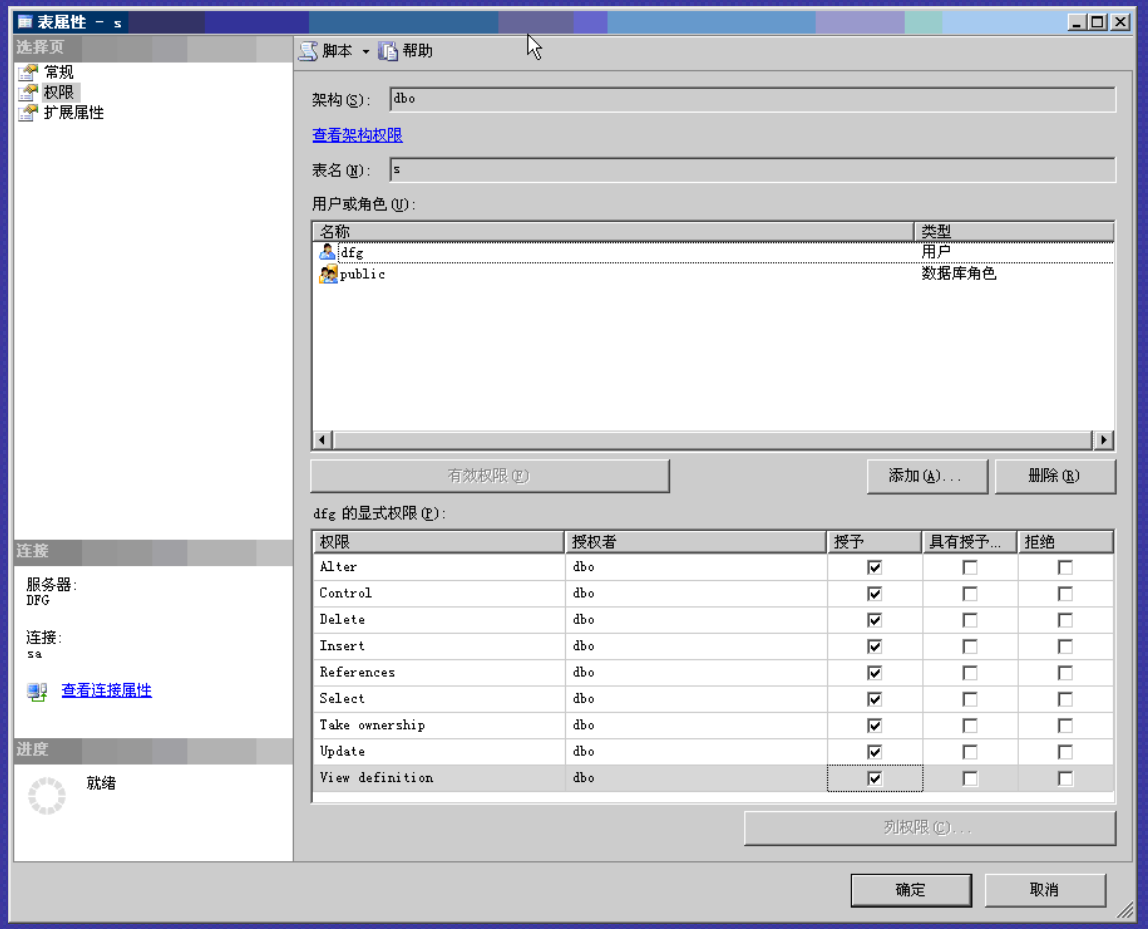


图10-14 表属性中的权限对话框

10.2.2 许可（权限）管理

2.使用Transaction_SQL 语句

Transaction-SQL 语句使用grant、revoke和deny三种命令来管理权限，相关的语法和实例可参照第3章数据控制语言部分。

10.2.3 角色管理

- 角色定义了常规的 SQL Server 用户类别。每种角色将该类别的用户与其使用 SQL Server时执行的任务集以及成功完成这些任务所需的知识相关联。利用角色，SQL Server 管理者可以将某些用户设置为某一角色，这样只要对角色进行权限设置便可以实现对所有用户权限的设置，大大减少了管理员的工作量。
- SQL Server提供了用户通常管理工作的预定义服务器角色和数据库角色。用户还可以创建自己的数据库角色，以便表示某一类进行同样操作的用户。当用户需要执行不同的操作时，只需将该用户加入不同的角色中即可，而不必对该用户反复授权许可和收回许可。

10.2.3 角色管理

1. 服务器角色

•服务器角色是指根据SQL Server的管理任务，以及这些任务相对的重要性等级来把具有SQL Server管理职能的用户划分为不同的用户组，每一组所具有的管理SQL Server的权限都是SQL Server内置的。服务器角色存在于各个数据库之中，要想加入用户，该用户必须有登录帐号以便加入到角色中。

•SQL Server2005提供了八种常用的固定服务器角色，其具体含义如下所示：

系统管理员（sysadmin）：拥有SQL Server所有的权限许可；

服务器管理员（Serveradmin）：管理SQL Server服务器端的设置；

磁盘管理员（diskadmin）：管理磁盘文件；

进程管理员（processadmin）：管理SQL Server系统进程；

安全管理员（securityadmin）：管理和审核SQL Server系统登录；

安装管理员（setupadmin）：增加、删除连接服务器，建立数据库复制以及管理扩展存储过程；

数据库创建者（dbcreator）：创建数据库，并对数据库进行修改。

批量数据输入管理员（bulkadmin）：管理同时输入大量数据的操作。

10.2.3 角色管理

2. 数据库角色

- 数据库角色是为某一用户或某一组用户授予不同级别的管理或访问数据库以及数据库对象的权限，这些权限是数据库专有的，并且还可以使一个用户具有属于同一数据库的多个角色。
- SQL Server提供了两种类型的数据库角色：
  （1）固定的数据库角色；
  （2）用户自定义的数据库角色。

10.2.3 角色管理

2. 数据库角色

（1）固定的数据库角色

固定的数据库角色是指SQL Server已经定义了这些角色所具有的管理、访问数据库的权限，而且SQL Server管理者不能对其所具有的权限进行任何修改。SQL Server中的每一个数据库中都有一组固定的数据库角色，在数据库中使用固定的数据库角色可以将不同级别的数据库管理工作分给不同的角色，从而有效地实现工作权限的传递。

10.2.3 角色管理

2. 数据库角色

（1）固定的数据库角色

•SQL Server提供了十种常用的固定数据库角色来授予组合数据库级管理员权限：

public：每个数据库用户都属于 public 数据库角色，当尚未对某个用户授予或拒绝对安全对象的特定权限时，则该用户将继承授予该安全对象的 public 角色的权限；

db_owner：可以执行数据库的所有配置和维护活动；

db_accessadmin：可以增加或者删除数据库用户、工作组和角色；

db_ ddladmin：可以在数据库中运行任何数据定义语言 (DDL) 命令；

db_securityadmin：可以修改角色成员身份和管理权限；

db_backupoperator：可以备份和恢复数据库；

db_datareader：能且仅能对数据库中的任何表执行select操作，从而读取所有表的信息；

db_datawriter：能够增加、修改和删除表中的数据，但不能进行SELECT操作；

db_denydatareader：不能读取数据库中任何表中的数据；

db_denydatawriter：不能对数据库中的任何表执行增加、修改和删除数据操作。

10.2.3 角色管理

2. 数据库角色

（2）用户自定义角色

•创建用户定义的数据库角色就是创建一组用户，这些用户具有相同的一组许可。如果一组用户需要执行在SQL Server中指定的一组操作并且不存在对应的Windows组，或者没有管理Windows用户帐号的许可，就可以在数据库中建立一个用户自定义的数据库角色。用户自定义的数据库角色有两种类型：即标准角色和应用程序角色。

•标准角色通过对用户权限等级的认定而将用户划分为不用的用户组，使用户总是相对于一个或多个角色，从而实现管理的安全性。所有的固定的数据库角色或SQL Server管理者自定义的某一角色都是标准角色。

•应用程序角色是一种比较特殊的角色。当我们打算让某些用户只能通过特定的应用程序间接地存取数据库中的数据而不是直接地存取数据库数据时，就应该考虑使用应用程序角色。当某一用户使用了应用程序角色时，他便放弃了已被赋予的所有数据库专有权限，他所拥有的只是应用程序角色被设置的角色。通过应用程序角色，能够以可控制方式来限定用户的语句或者对象许可。

10.2.3 角色管理

3. 使用SQL Server管理平台管理角色

（1）管理服务器角色。

打开SQL Server管理平台，展开指定的服务器，单击安全性文件夹，然后单击服务器角色图标，在右边的页框中右击所要的角色，从弹出的快捷菜单中选择“属性”选项，则出现服务器角色属性对话框，如图10-15所示。在该对话框中我们可以看到属于该角色的成员。单击“添加”按钮则弹出添加成员对话框，其中可以选择添加新的登录帐号作为该服务器角色成员，单击删除按钮则可以从服务器角色中“删除”选定的帐号。

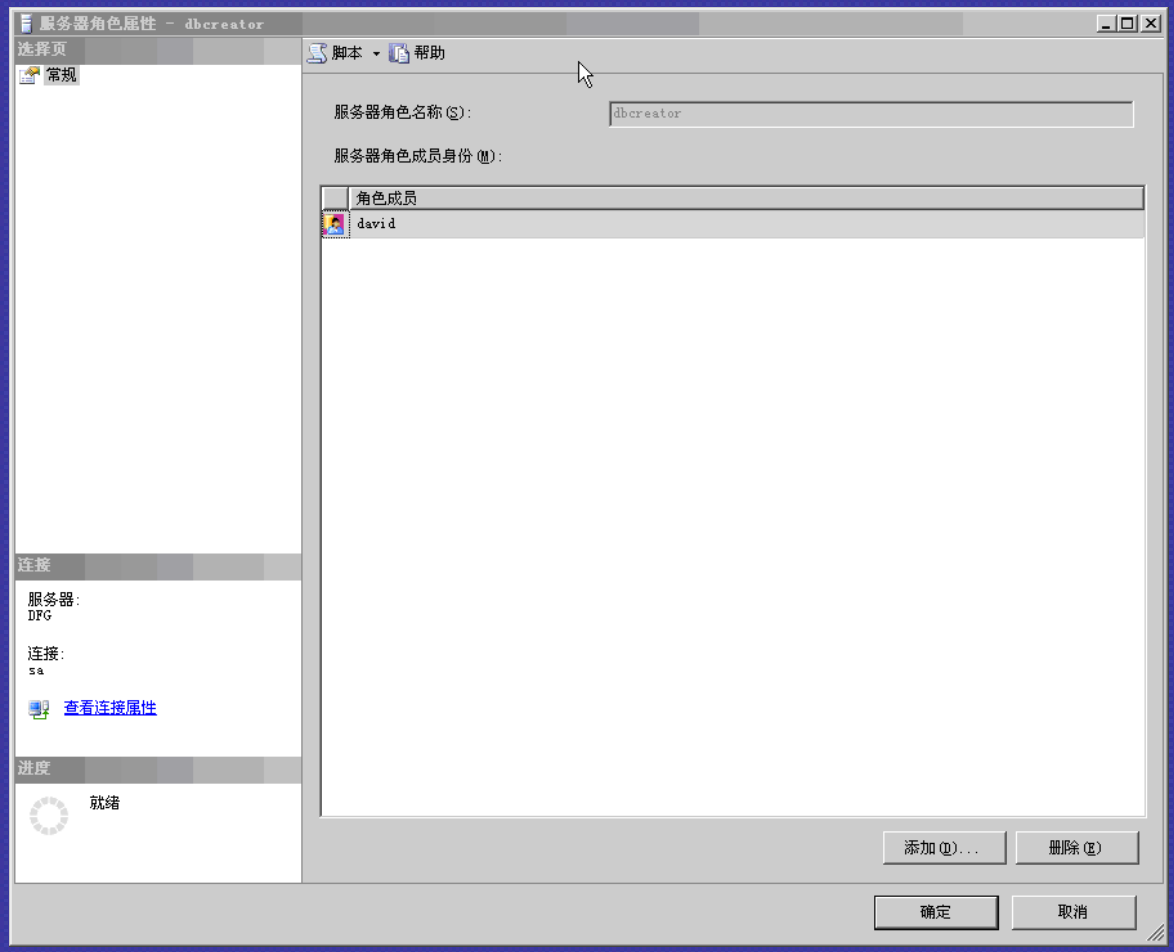


图10-15 服务器角色属性对话框

10.2.3 角色管理

3. 使用SQL Server管理平台管理角色

（2）管理数据库角色。

•在SQL Server管理平台中，展开指定的服务器以及指定的数据库，然后展开安全性文件夹，右击数据库角色图标，从弹出的快捷菜单中选择“新建数据库角色”选项，则出现新建数据库角色对话框，如图10-16所示。在名称文本框中输入该数据库角色的名称；点击架构前的复选框，可设定此角色拥有的架构；单击“添加”按钮，可将数据库用户增加到新建的数据库角色中；最后单击“确定”按钮即可完成新的数据库角色的创建。

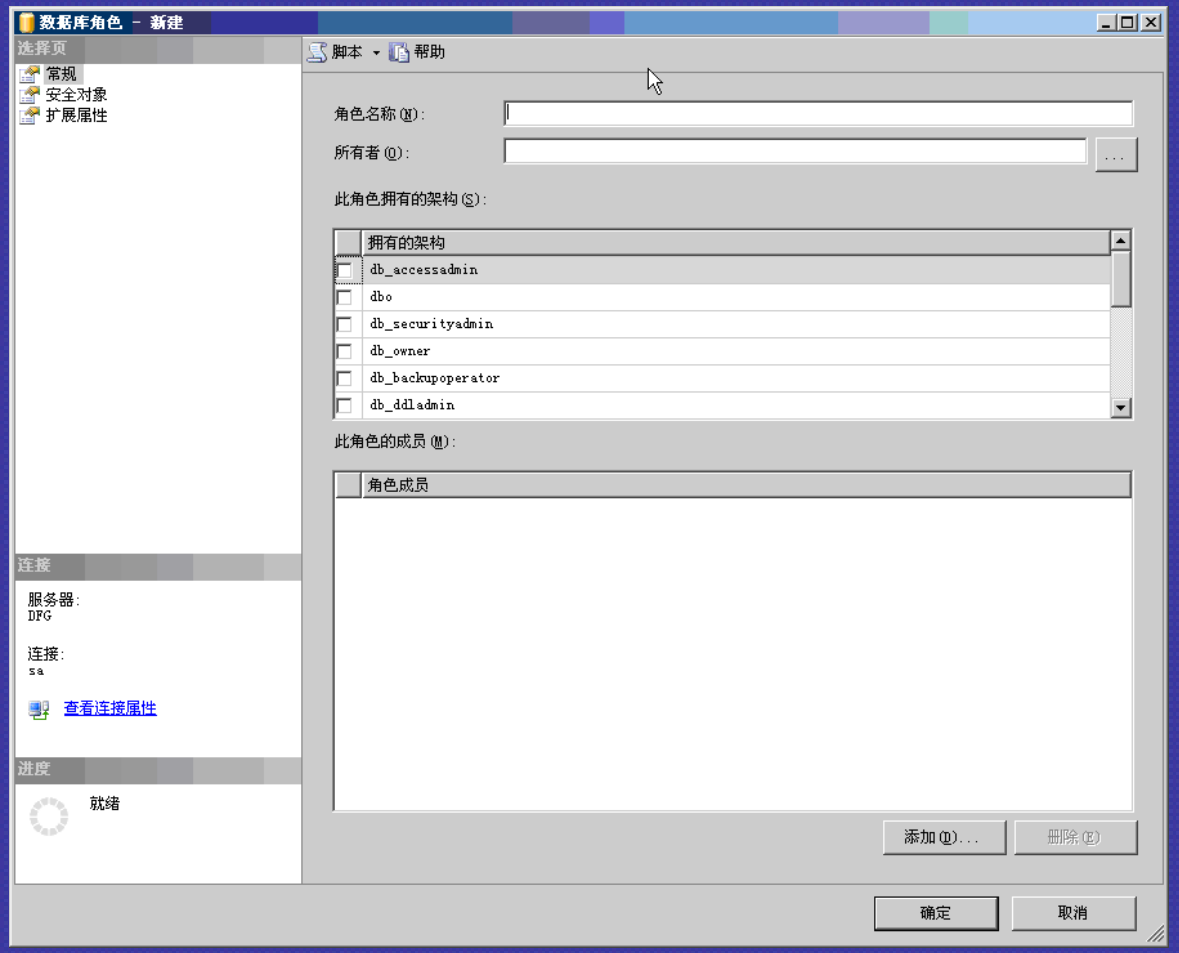


图10-16 “数据库角色—新建”对话框

10.2.3 角色管理

4. 使用存储过程管理角色

（1）管理服务器角色

•在SQL Server中，管理服务器角色的存储过程主要有两个：sp_addsrvrolemember和sp_dropsrvrolemember。

•系统存储过程sp_addsrvrolemember可以将某一登录帐号加入到服务器角色中，使其成为该服务器角色的成员。其语法形式如下：

sp_addsrvrolemember login，role

•系统存储过程sp_dropsrvrolemember可以将某一登录者从某一服务器角色中删除，当该成员从服务器角色中被删除后，便不再具有该服务器角色所设置的权限。其语法形式如下：

sp_dropsrvrolemember [@loginame=]'login'，[@rolename=]'role'

其中，@loginame为登录者名称；@rolename为服务器角色。

10.2.3 角色管理

4. 使用存储过程管理角色

（2）管理数据库角色

在SQL Server中，支持数据库管理的存储过程主要有六种：

•create role：用来创建一个新的数据库角色，create role语法形式如下：

create role role_name [AUTHORIZATION owner_name]

其中：role_name 为待创建角色的名称；AUTHORIZATION owner_name为拥有新角色的数据库用户或角色。如果未指定用户，则执行 CREATE ROLE 的用户将拥有该角色。

•droprole：用于从当前数据库角色中删除一个数据库角色，Droprole的语法形式如下：

Drop role role_name

•sp_helprole：用来显示当前数据库中所有数据库角色的全部信息。其语法形式如下：

sp_helprole ['role']

•sp_addrolemember：用来向数据库某一角色中添加数据库用户，这些角色可以是用户自定义的标准角色，也可以是固定的数据库角色，但不能是应用程序角色。其语法形式如下：

sp_addrolemember role，security_account

•sp_droprolemember：用来删除某一角色的用户。其语法形式如下：

sp_droprolemember role，security_account

•sp_helprolemember：用于显示某一数据库角色的所有成员。其语法形式如下

sp_helprolemember ['role']

第11章 SQL Server 2005 集成服务

集成服务概述

- 集成服务（Integration Services）是用于生成高性能数据集成和工作流解决方案（包括针对数据仓库的提取、转换和加载 (ETL) 操作）的平台。集成服务包括生成并调试包的图形工具和向导；执行如数据导入、导出， FTP 操作，SQL 语句执行和电子邮件消息传递等工作流功能的任务等。
- 数据转换服务是一个功能非常强大的组件。其中，导入和导出向导提供了把数据从一个数据源转换到另一个数据目的地的简单方法，该工具可以在异构数据环境中拷贝数据、拷贝整个表或者查询结果，并且可以交互式地定义数据转换方式。SQL Server商务智能开发平台是一个图形工具，它使创建和编辑集成服务包（SSIS包）的工作变得更加简单和轻松，而且它提供了比导入，导出向导更为强大的功能。可以向SSIS包中添加控制流、数据流任务和事件处理程序。

11.1 数据的导入和导出

11.1.1 数据的导入

11.1.2 数据的导出

11.1.1 数据的导入

1.导入Access数据库

利用导入、导出向导导入Access数据库的步骤如下：

（1）打开SQL Server管理平台，展开服务器和数据库，右击该数据库图标，从弹出的快捷菜单中选择“任务→导入数据”选项，如图11-1所示。启动数据导入向导工具，就会出现欢迎使用向导对话框，对话框中列出了导入向导能够完成的操作。

（2）单击“下一步”按钮，则出现选择数据源对话框，如图11-2所示。在该对话框中，可以选择数据源类型、文件名、用户名和密码等选项

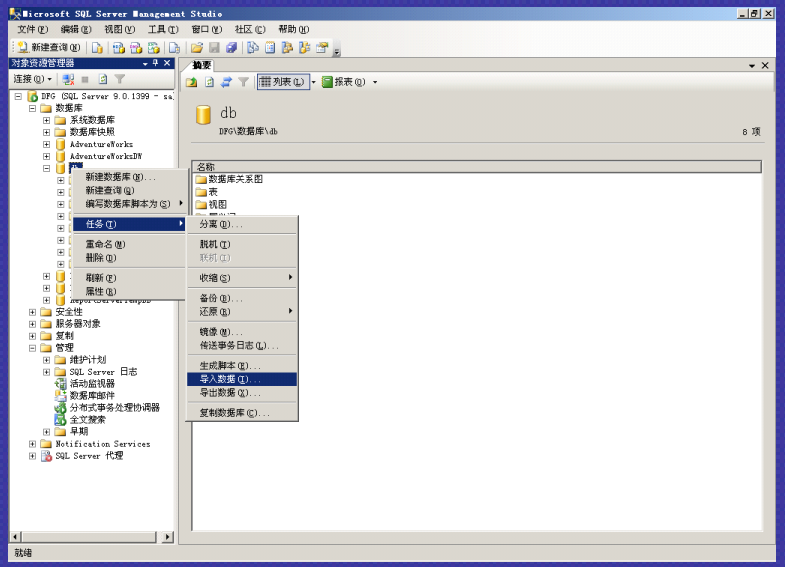


图11-1 打开导入向导

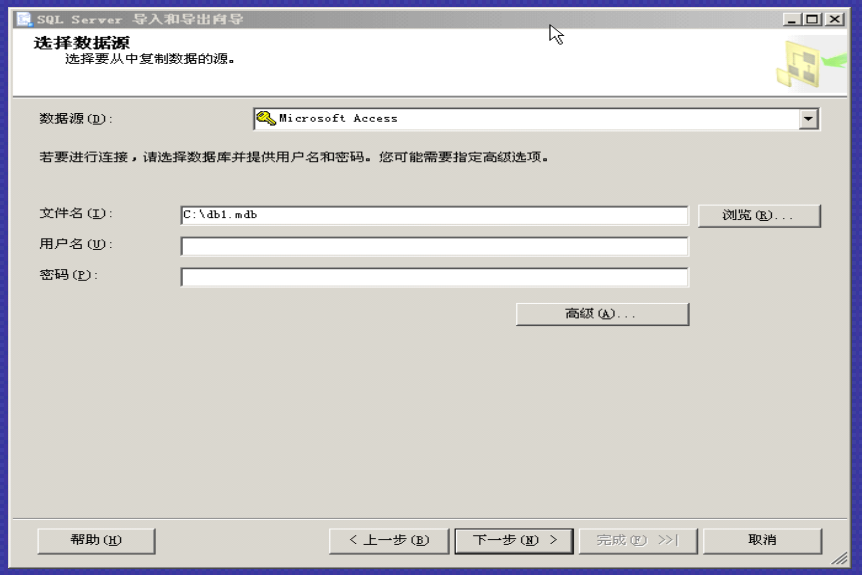


图11-2 选择数据源对话框

11.1.1 数据的导入

1.导入Access数据库

（3）单击“下一步”按钮，则出现选择导入的目标数据库类型对话框，如图11-3所示。本例使用SQL Server数据库作为目标数据库，在目标对话框中选择SQL Native Client，在服务器名称框中输入目标数据库所在的服务器名称。下方需要设定连接服务器的安全模式以及目标数据库的名称。设定完成后，单击“下一步”按钮，则出现指定表复制或者查询对话框，如图11-4所示。

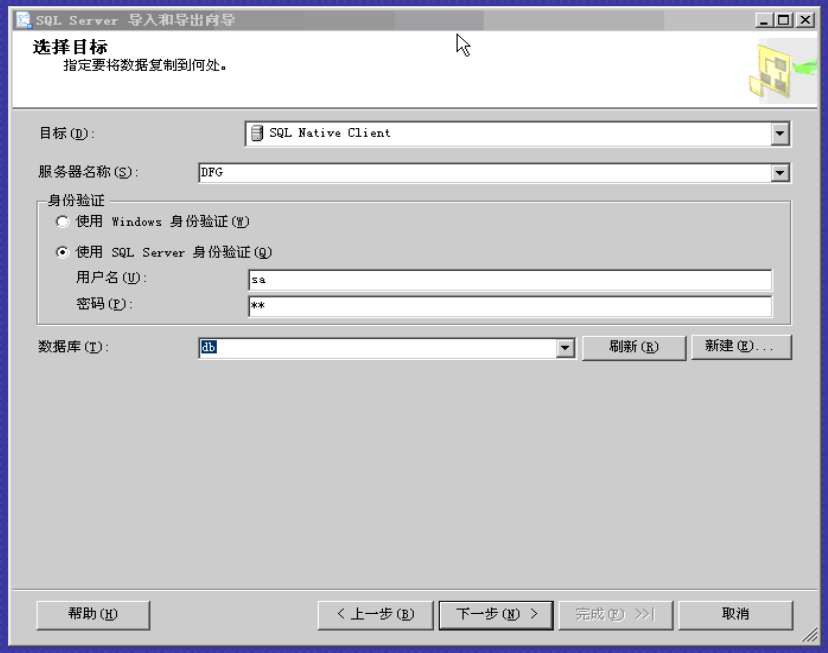


图11-3　选择目标对话框

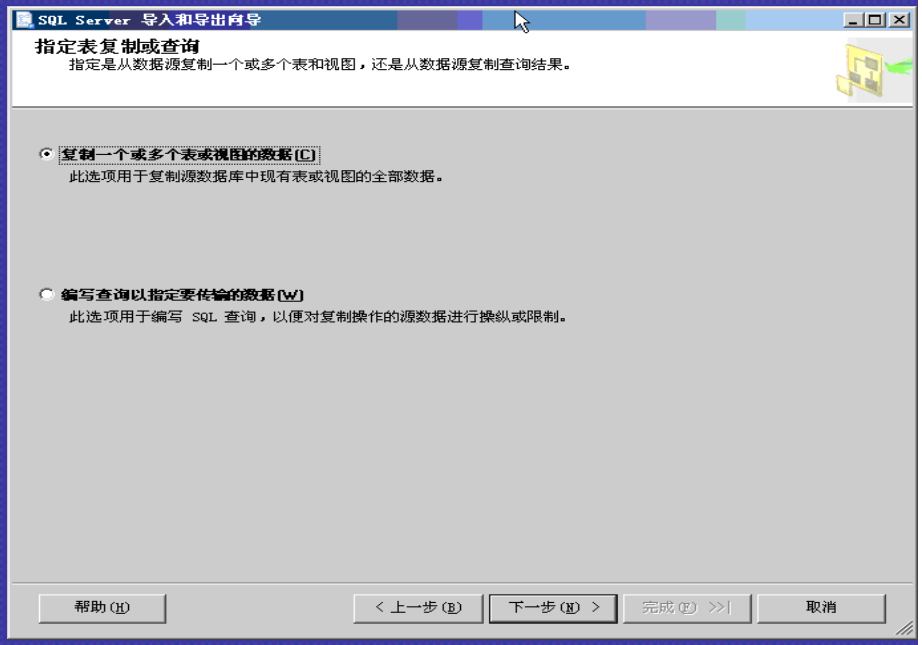


图11-4　指定表复制或查询对话框

11.1.1 数据的导入

1.导入Access数据库

（4）单击“下一步”按钮，就会出现选择源表和视图对话框，如图11-5所示。在该对话框中，可以设定需要将源数据库中的哪些表格传送到目标数据库中去。单击表格名称左边的复选框，可以选定或者取消对该表格的复制。如果想编辑数据转换时源表格和目标表格之间列的对应关系，可单击表格名称右边的“编辑...”按钮，则出现列映射对话框，如图11-6所示。

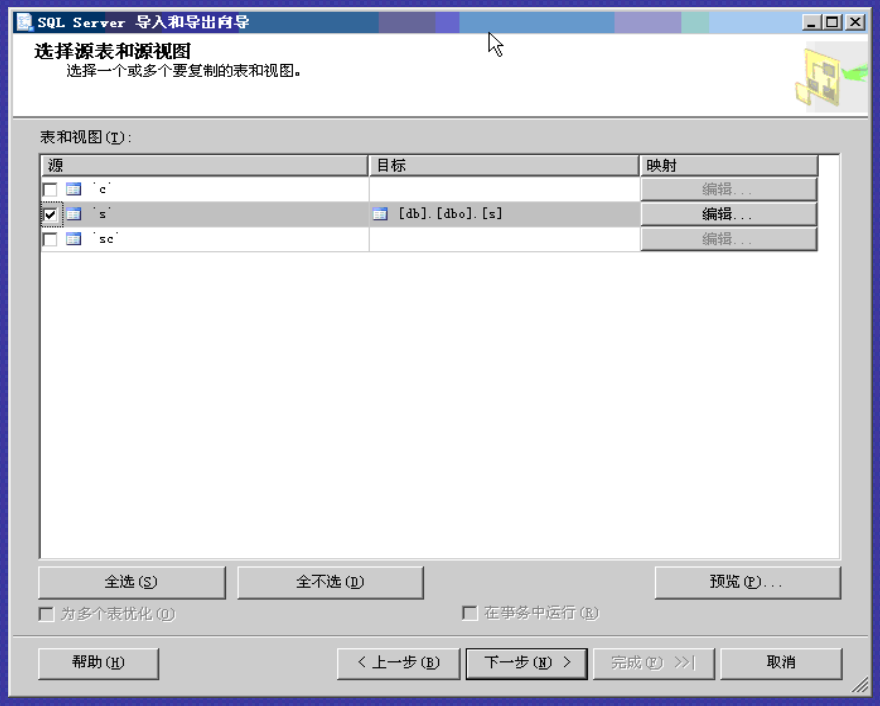


图11-5　选择源表和视图对话框

图11-6　列映射对话框

11.1.1 数据的导入

1.导 入 Access 数据库

（5）在图11-5中单击“下一步”按钮，则会出现“保存并执行包”对话框，如图11-7所示。在该对话框中，可以指定是否希望保存SSIS包，也可以立即执行导入数据操作。

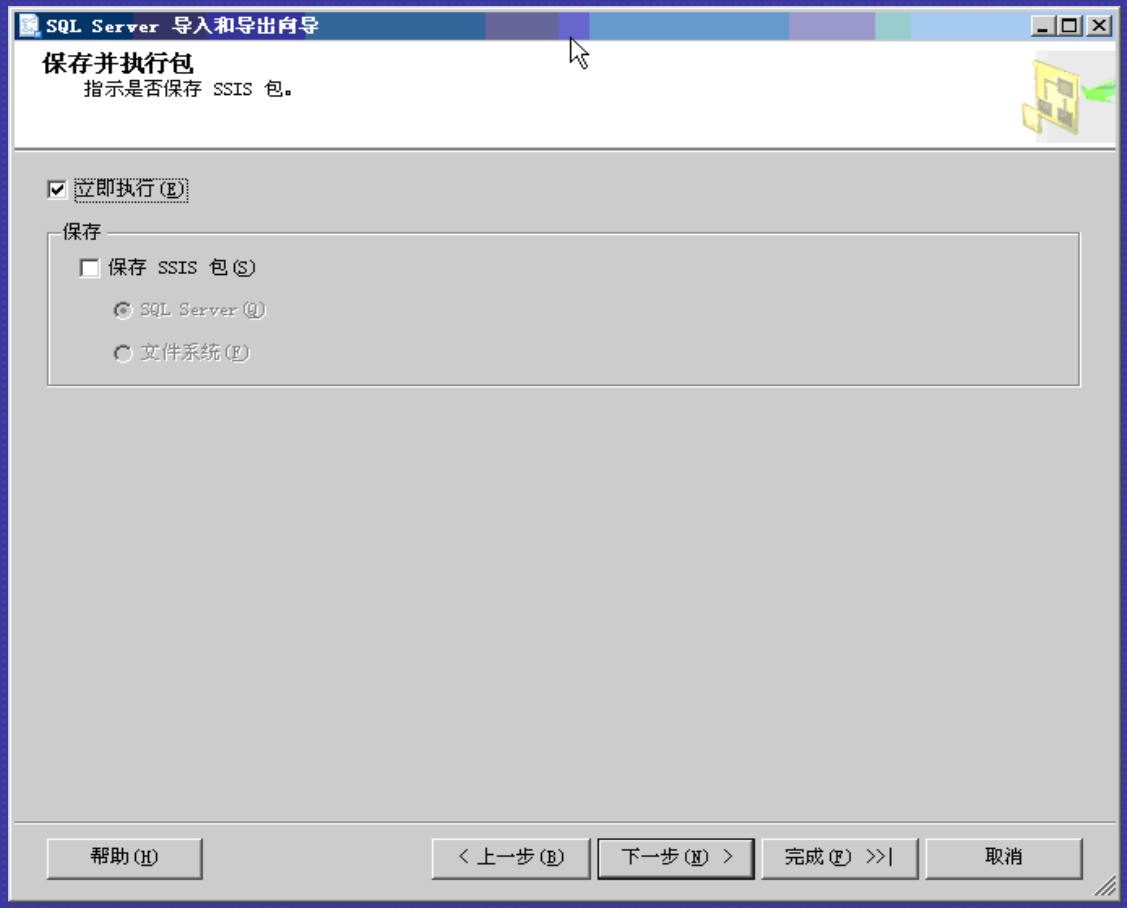


图11-7 “保存并执行包”对话框

11.1.1 数据的导入

1.导入Access数据库

（6）单击“下一步”按钮，则出现“包保护级别”对话框，如图11-8所示。点击“确定”按钮可完成包保护级别设定，并打开“保存SSIS包”页框，如图11-9所示。

图11-8 “包保护级别”对话框

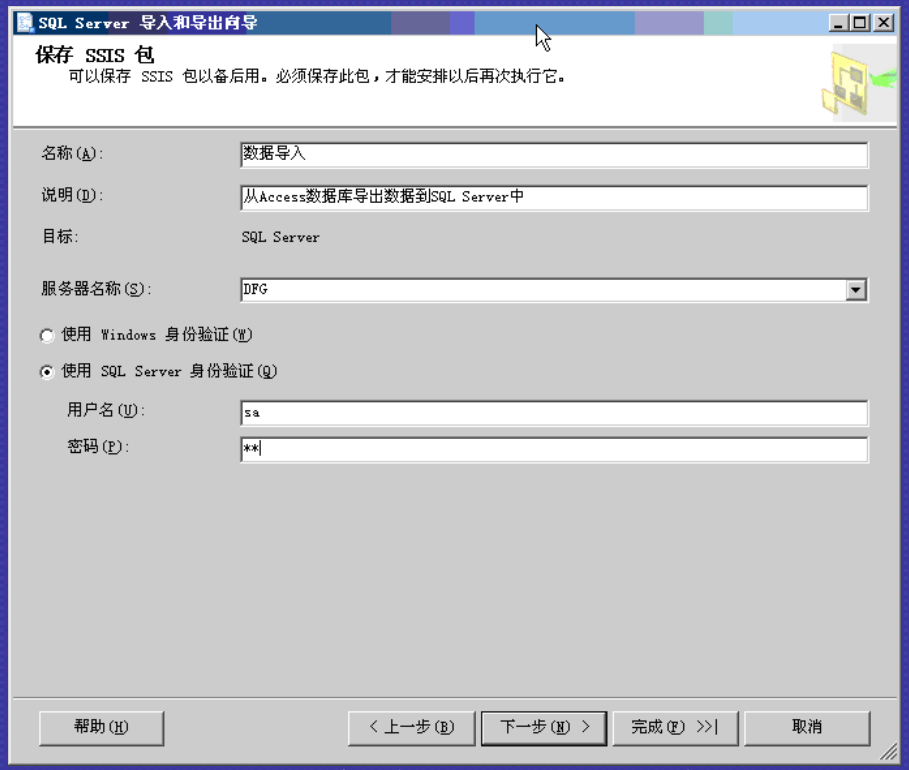


图11-9 保存SSIS包对话框

11.1.1 数据的导入

1.导 入 Access 数据库

（7）单击“下一步”按钮，则出现向导完成确认对话框，如图11-10所示。其中显示了在该向导中进行的设置，如果确认前面的操作正确，单击“完成”按钮后进行数据导入操作，否则，单击“上一步”按钮返回修改。

图11-10 完成向导对话框

11.1.1 数据的导入

2. 导入文本文件

（1）打开SQL Server管理平台，展开选定的服务器和数据库，右击该数据库图标，从弹出的快捷菜单中选择“任务→导入数据”选项，如图11-1所示。启动数据导入向导工具，就会出现欢迎使用向导对话框，对话框中列出了导入向导能够完成的操作。（2）单击“下一步”按钮，则出现选择数据源对话框，如图11-11所示。这里在数据源栏中选择平面文件源，即文本文件。

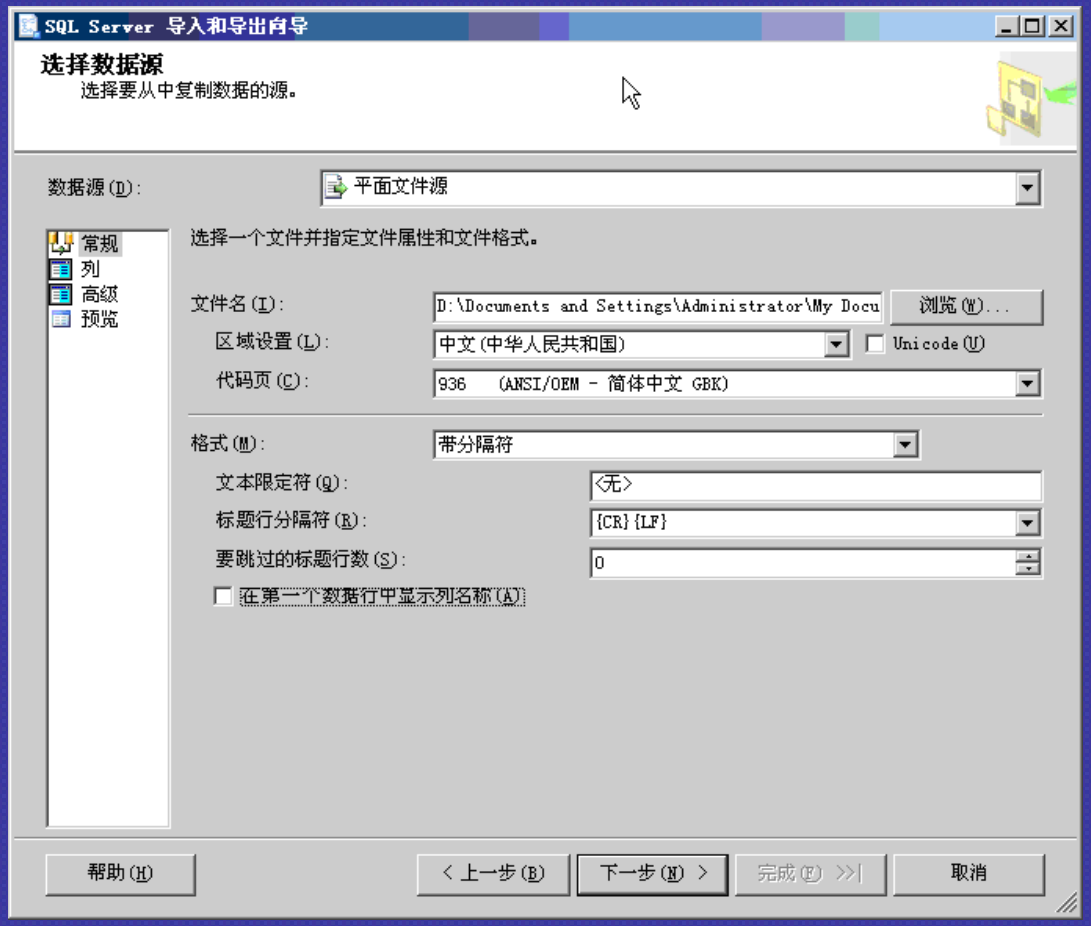


图11-11　选择文本数据源对话框

11.1.1 数据的导入

2. 导入文本文件

（3）单击“下一步”按钮，就会出现选择目的数据库类型对话框，如图11-12所示。这里选择为SQL Server，选定服务器名称和数据库名称后，单击“下一步”按钮，则出现选择源表和视图对话框，如图11-13所示。

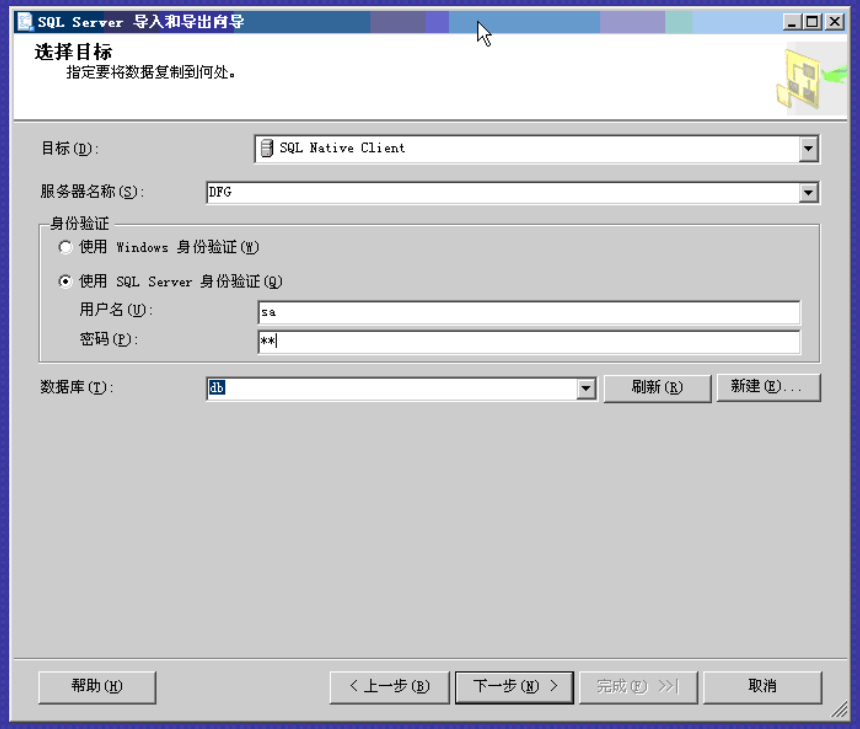


图11-12　选择目的数据库对话框

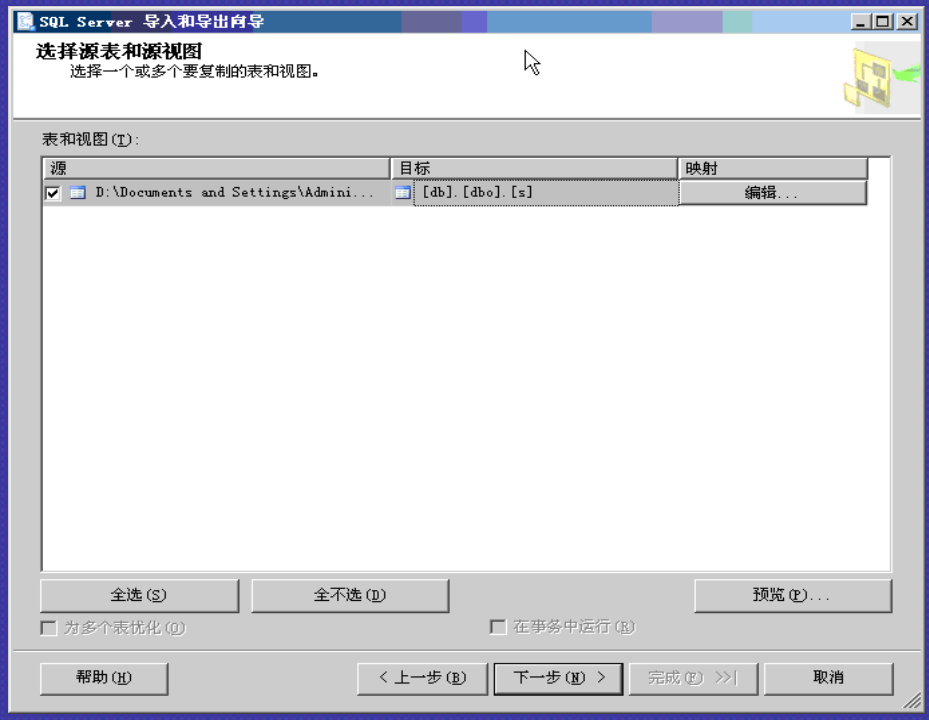


图11-13　选择源表和视图对话框

11.1.1 数据的导入

2. 导入文本文件

（4）单击映射“编辑...”按钮，则出现列映射对话框，如图11-14所示，点击“确定”按钮保存所作设置。单击“下一步”按钮，则出现“保存并执行包”对话框。在该对话框中，可以设置立即执行或者保存SSIS包以备以后执行。

（5）单击“下一步”按钮，则出现确认导入数据对话框，如图11-15所示。

图11-14　选择列映射对话框

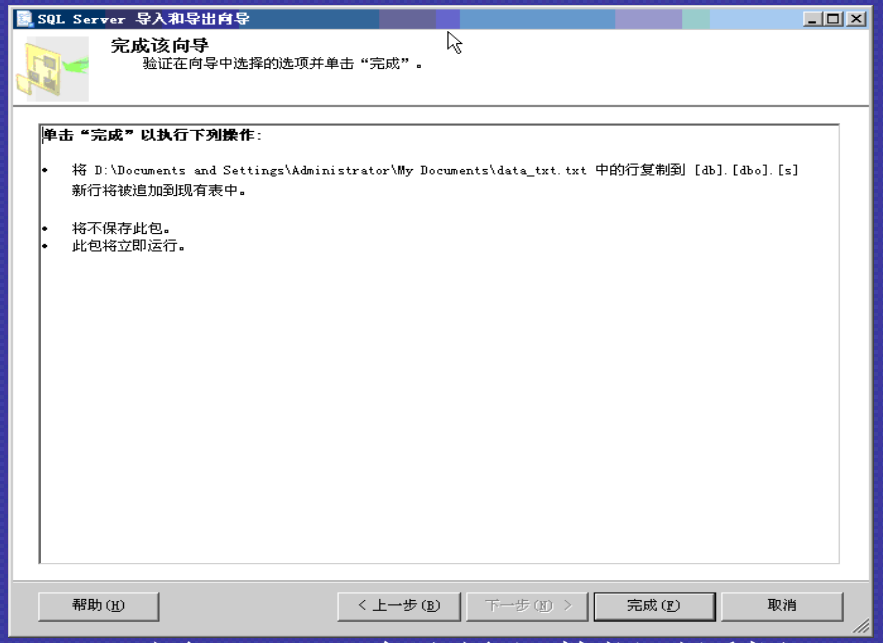


图11-15　确认导入数据对话框

11.1.1 数据的导入

2. 导入文本文件

（6）如果在向导中设定了立即执行，在向导结束后，则会出现数据导入对话框，如图11-16所示。该对话框中执行向导中定义的复制操作。

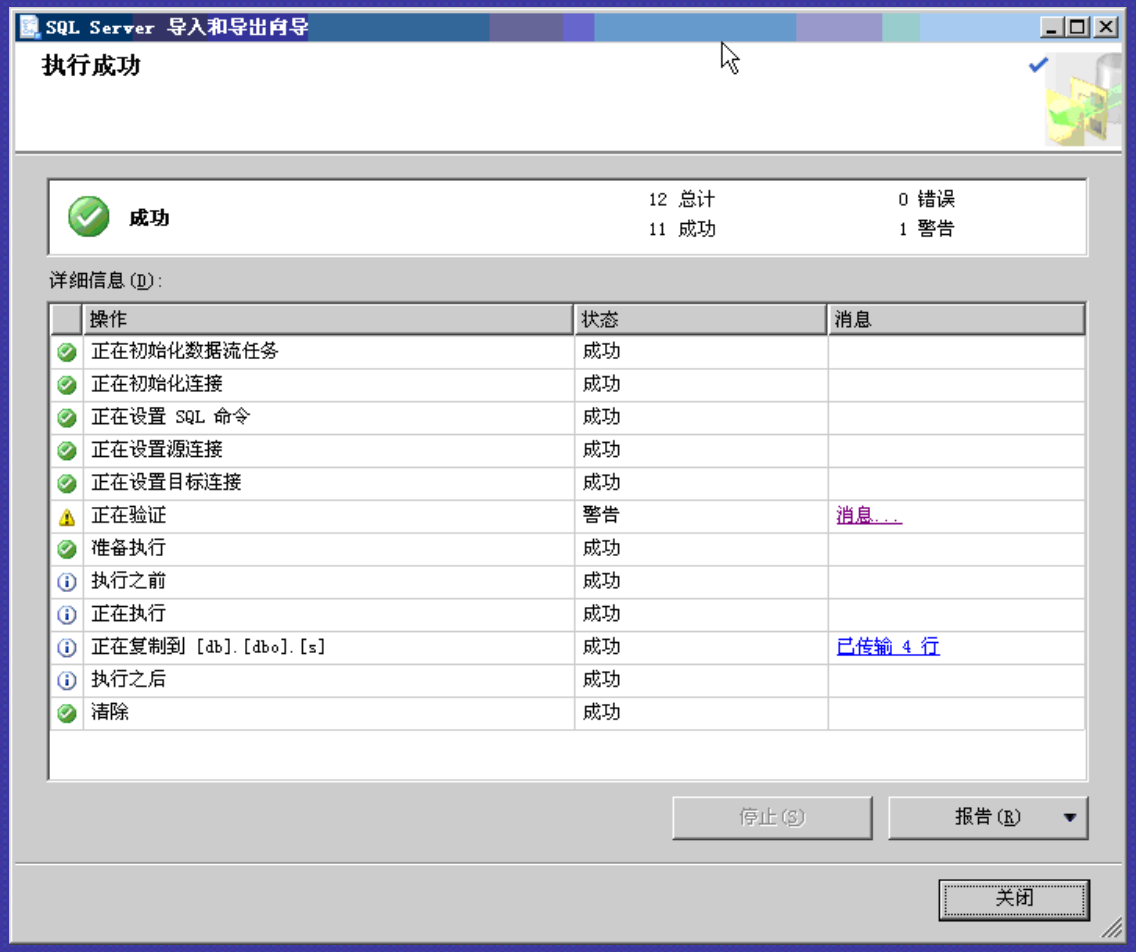


图11-16　进行数据导入对话框

11.1.2 数据的导出

1. 导出数据库至Access

（1）打开SQL Server管理平台，右击服务器图标，从弹出的快捷菜单中选择“所有任务→导出数据”选项，则会出现数据转换服务导入和导出向导对话框，它显示了导出向导所能完成的操作。

（2）单击“下一步”按钮，就会出现选择导出数据的数据源对话框，如图11-17所示。这里在数据源栏中选择“Microsoft OLE DB Provider for SQL Server”选项，然后选择身份验证模式以及数据库的名称。

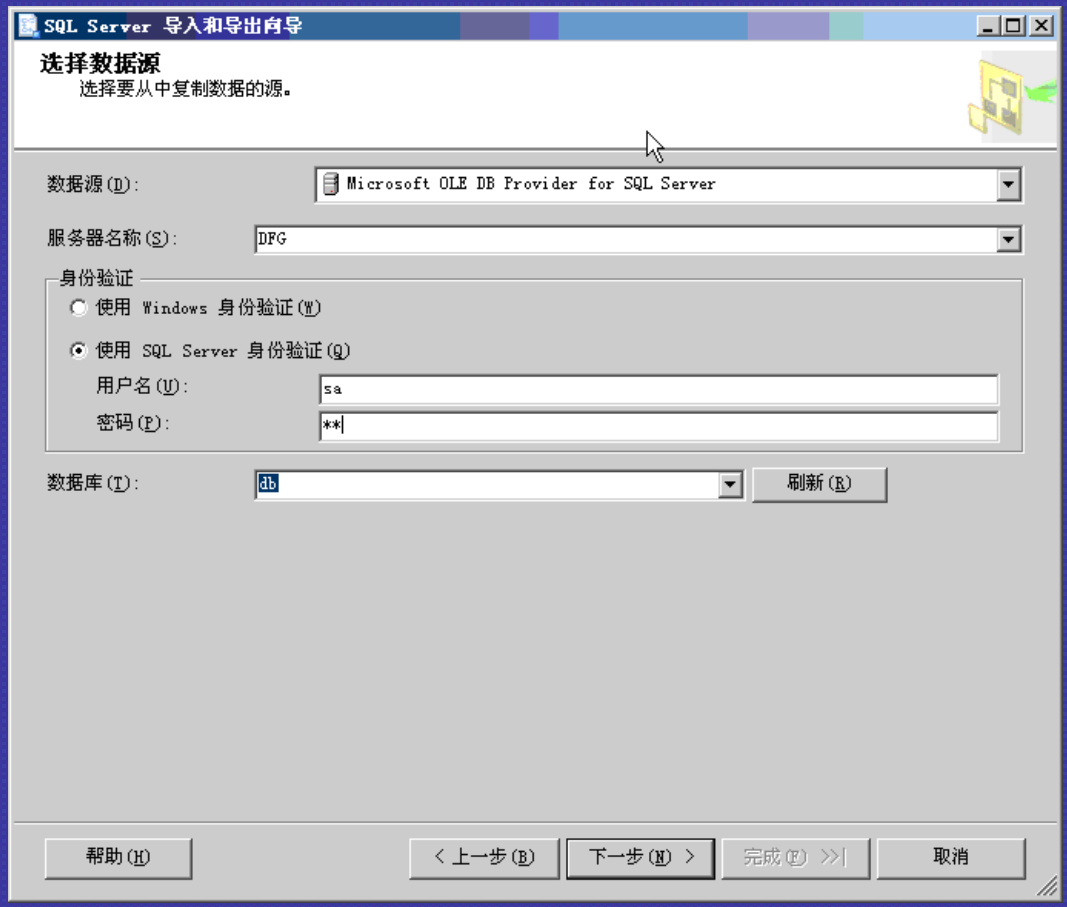


图11-17　选择数据源对话框

11.1.2 数据的导出

1. 导出数据库至Access

（3）单击“下一步”按钮，则会出现选择目的对话框，如图11-18所示。

（4）选定目标数据库后，单击“下一步”按钮，则出现指定表复制或查询对话框，如图11-19所示。

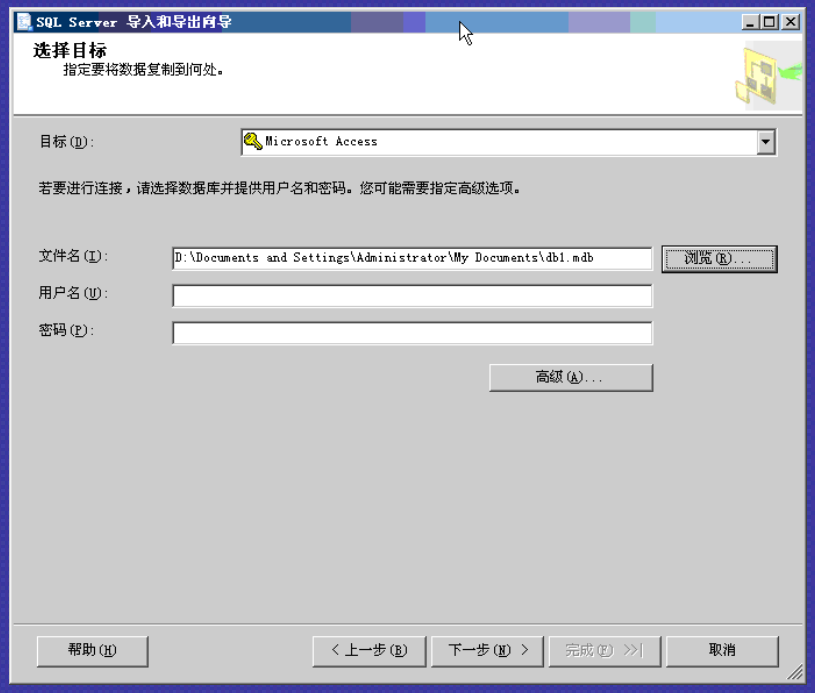


图11-18　选择目的数据库对话框

图11-19　指定表复制或查询对话框

11.1.2 数据的导出

1. 导出数据库至Access

（5）单击“下一步”按钮，则出现选择源表和视图对话框，如图11-20所示。其中可以选定将源数据库中的哪些表格或视图复制到目标数据库中，只需单击表格名称左边的复选框，即可选定或者取消删除复制该表格或视图。单击“编辑...”按钮，就会出现列映射对话框，如图11-21所示。

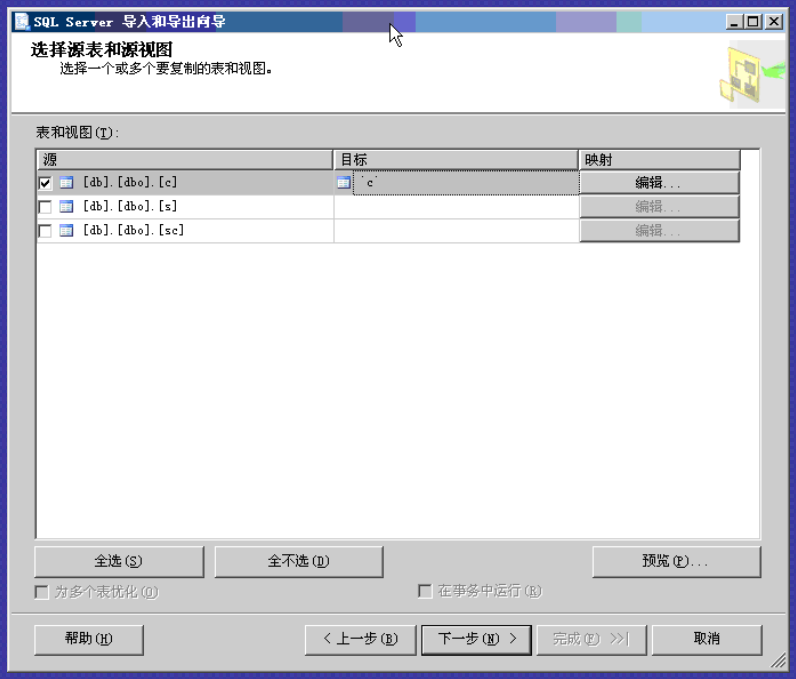


图11-20　选择源表和视图对话框

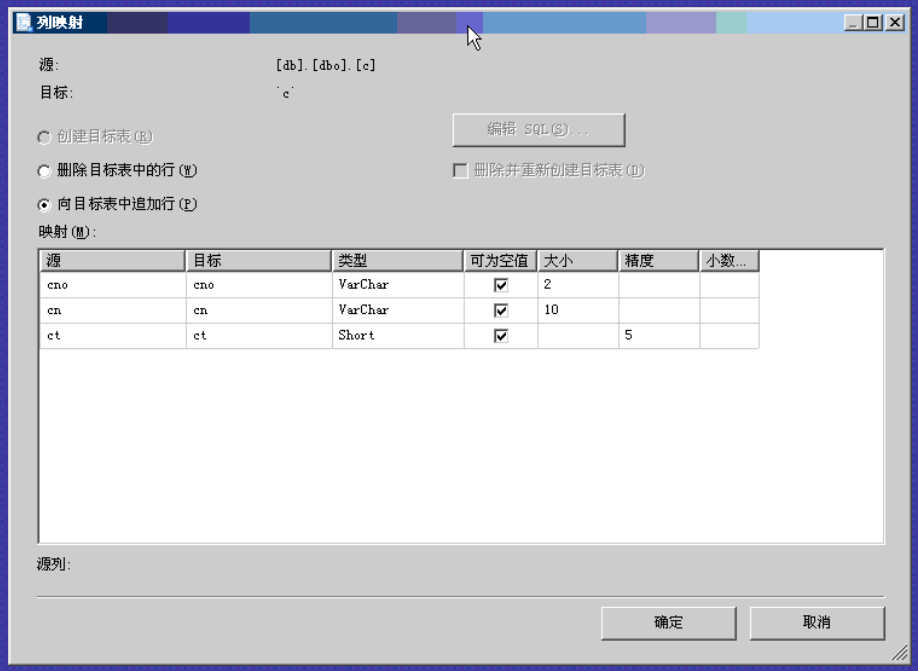


图11-21　列映射对话框

11.1.2 数据的导出

1. 导出数据库至Access

（6）选定某个表格后，单击“预览”按钮，就会出现查看数据对话框，如图11-22所示，在该对话框中可以预览表格内的数据。单击“下一步”按钮，则会出现“保存并执行包”对话框。在该对话框中，可以设定立即执行还是保存包以备以后执行。

（7）单击“下一步”按钮，就会出现导出向导结束对话框，如图11-23所示。

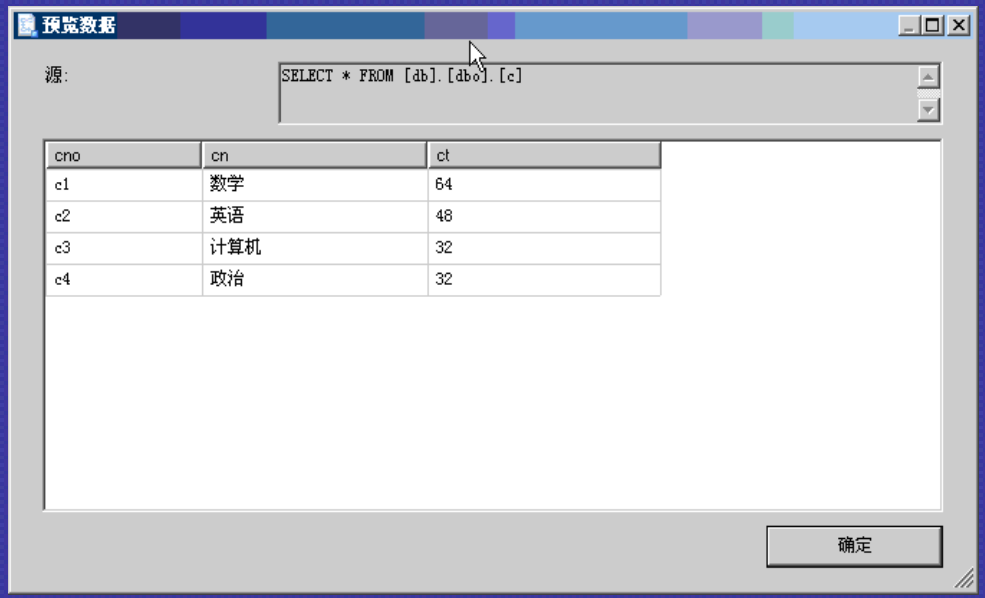


图11-22　查看数据对话框

图**11-23**　确认导出数据对话框

11.1.2 数据的导出

2. 导出数据库至文本文件

（1）打开SQL Server管理平台，右击服务器图标，从弹出的快捷菜单中选择“所有任务→导出数据”选项，则会出现数据转换服务导入和导出向导对话框，它显示了该导出向导所能完成的操作。

（2）单击“下一步”按钮，则会出现选择数据源对话框。

（3）单击“下一步”按钮，就会出现选择目标对话框，如图11-24。在“目标”栏中选择“平面文件目标”选项。单击“浏览”按钮，则会出现选择文件对话框，如图11-25所示，可以设定目标文件的文件名。

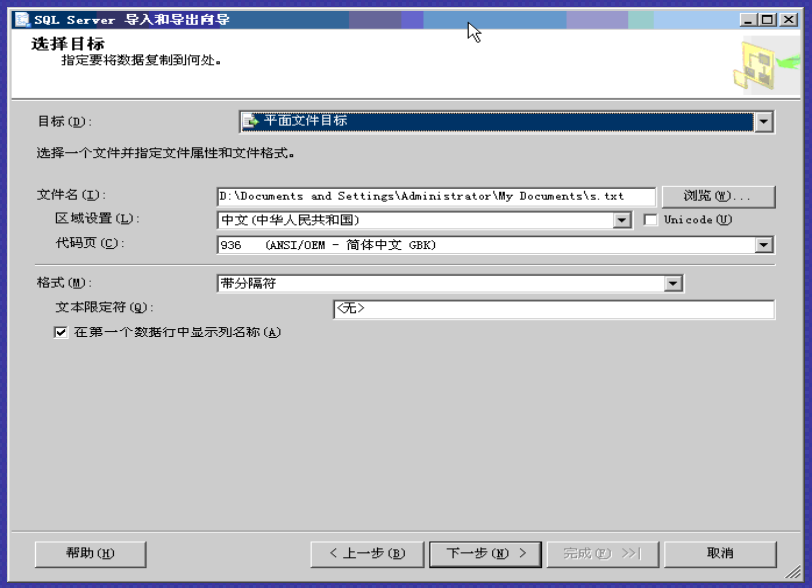


图11-24 选择目标对话框

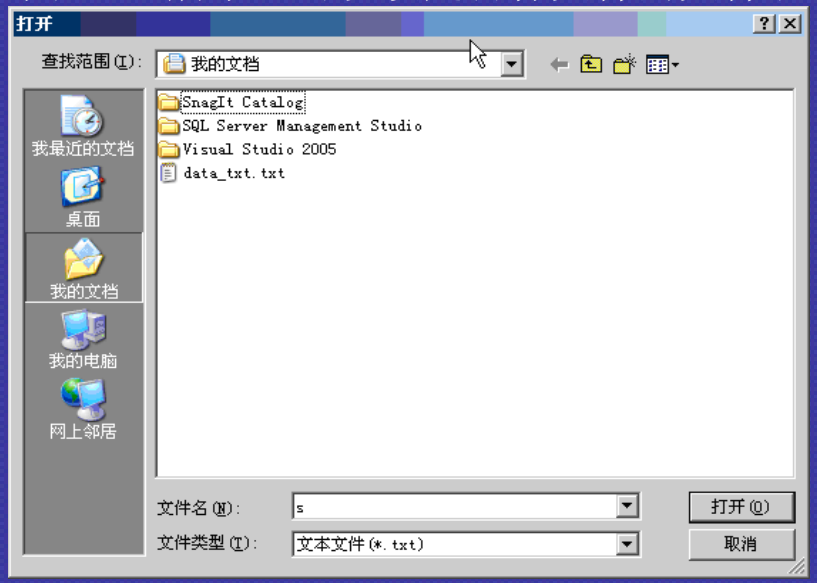


图11-25 选择文件对话框

11.1.2 数据的导出

2. 导出数据库至文本文件

（4）单击“下一步”按钮，就会出现指定表复制或查询对话框。其中可以选定将源数据库中的表格或视图复制到文本文件，还是将满足查询结果的记录复制到文本文件。

（5）单击“下一步”按钮，则出现“配置平面文件目标”对话框，如图11-26所示。如图11-27所示在该对话框中单击“编辑转换”按钮，则出现列映射对话框.

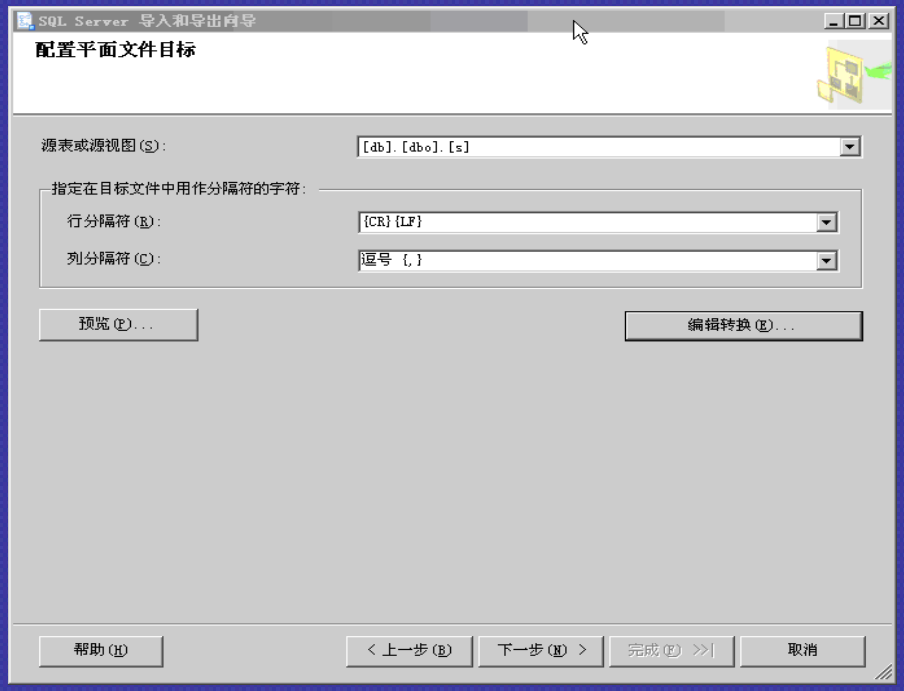


图11-26 配置平面文件目标对话框

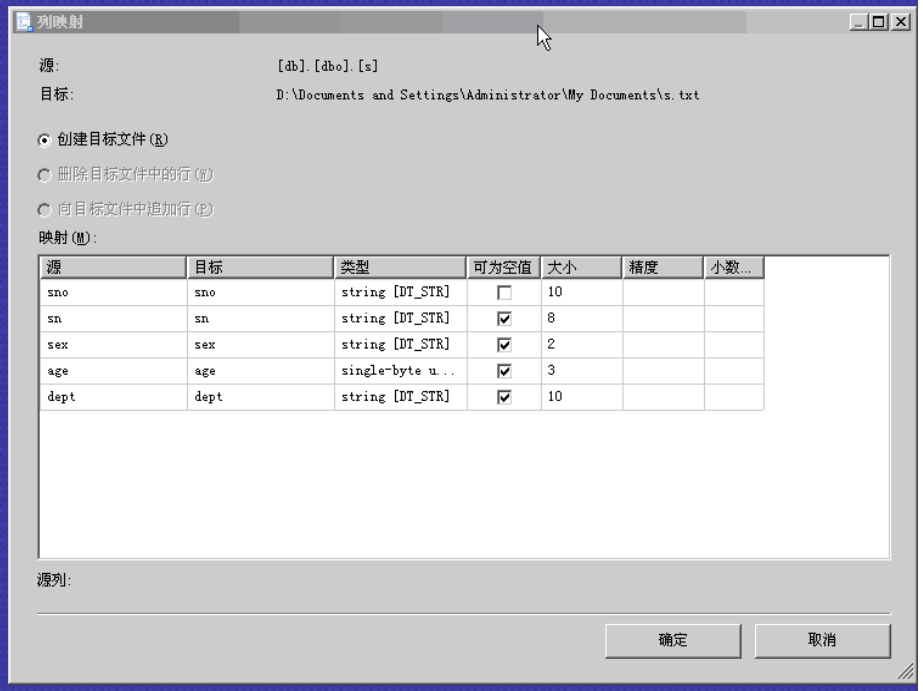


图11-27 列映射对话框

11.1.2 数据的导出

2. 导出数据库至文本文件

（6）在该对话框中单击“预览”按钮，可查看要导出的表中的数据，如图11-28所示。 单击“下一步”按钮，就会出现“保存并执行包”对话框。单击“下一步”按钮，则出现“完成该向导”对话框，如图11-29

图11-28 预览数据对话框

图11-29 完成向导对话框

11.2 使用图形设计界面来创建SSIS包

- SQL Server商务智能开发平台是一个图形工具，它使创建和编辑SSIS包的工作变得更加简单和轻松，而且它提供了比导入，导出向导更为强大的功能。可以向SSIS包中添加控制流、数据流任务和事件处理程序。
- SQL Server 2005集成服务包（SSIS包）中的控制流由不同类型的控制流元素构造而成：容器、任务和优先约束。容器提供包中的结构并给任务提供服务，任务在包中提供功能，优先约束将容器和任务连接成一个控制流。
- SQL Server 2005 SSIS包中的数据流由下列不同类型的数据流元素构造而成：提取数据的源、修改和聚合数据的转换、加载数据的目标以及将数据流组件的输出和输入连接为数据流的路径。
- 事件处理程序与包类似。事件处理程序可以像包一样为变量提供作用域，并且包含控制流和可选数据流。

11.2 使用图形设计界面来创建SSIS包

1.创建包

创建SSIS包的步骤如下：

（1）在SQL Server商务智能开发平台中，在文件菜单中选择“新建　项目”，如图11-31所示，则打开“新建项目”对话框，如图11-32所示。

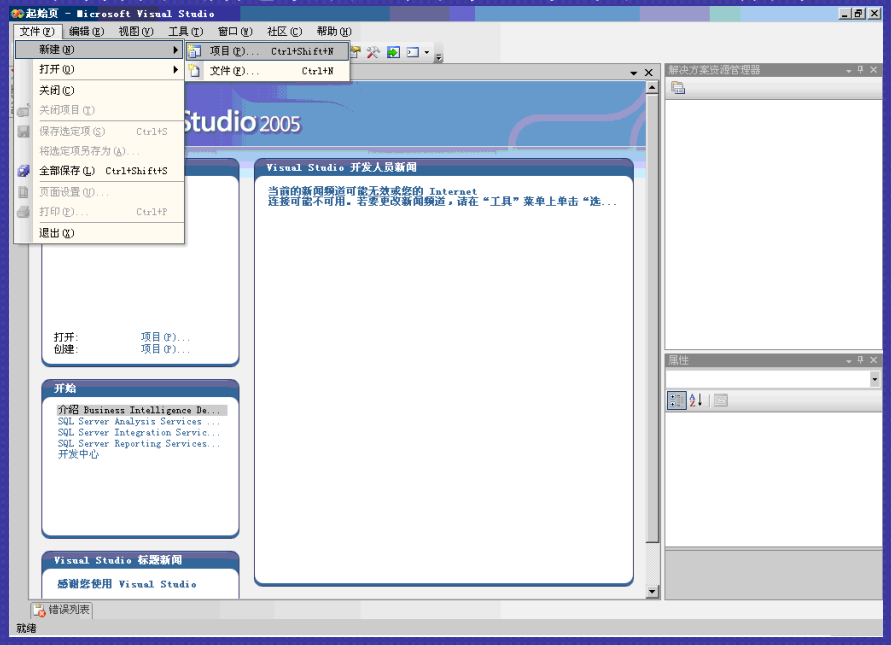


图11-31　打开“新建项目”对话框

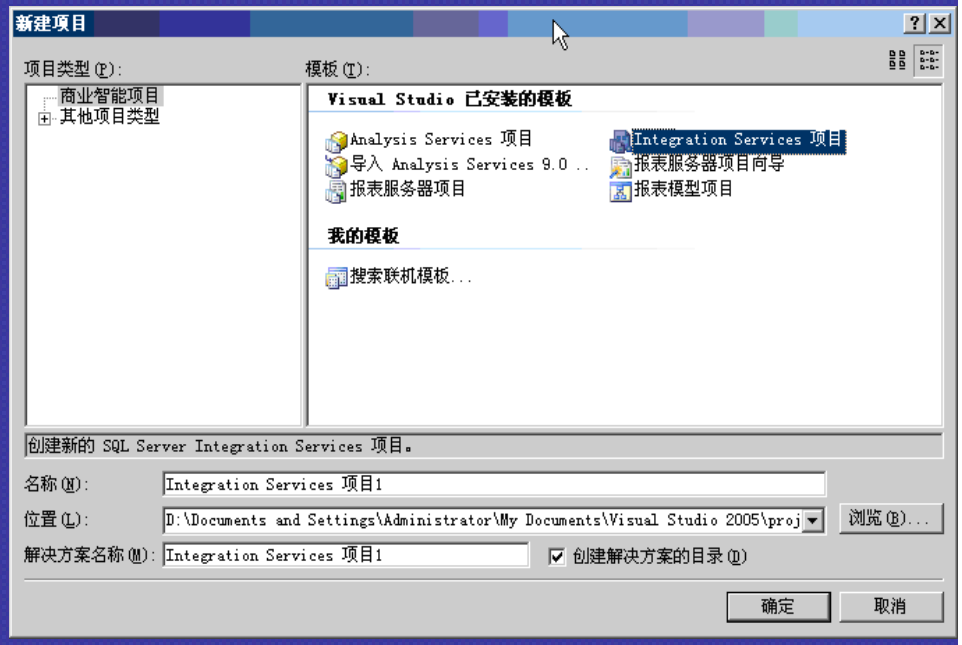


图11-32　“新建集成服务项目”对话框

11.2 使用图形设计界面来创建SSIS包

1.创建包

（2）在集成服务项目对话框中，如图11-33所示。可以向包中添加控制流、数据流任务和事件处理程序。控制流设计器用来创建包中的控制流。工具箱的“控制流项”节点列出多种类型的任务和容器，如图11-34所示。

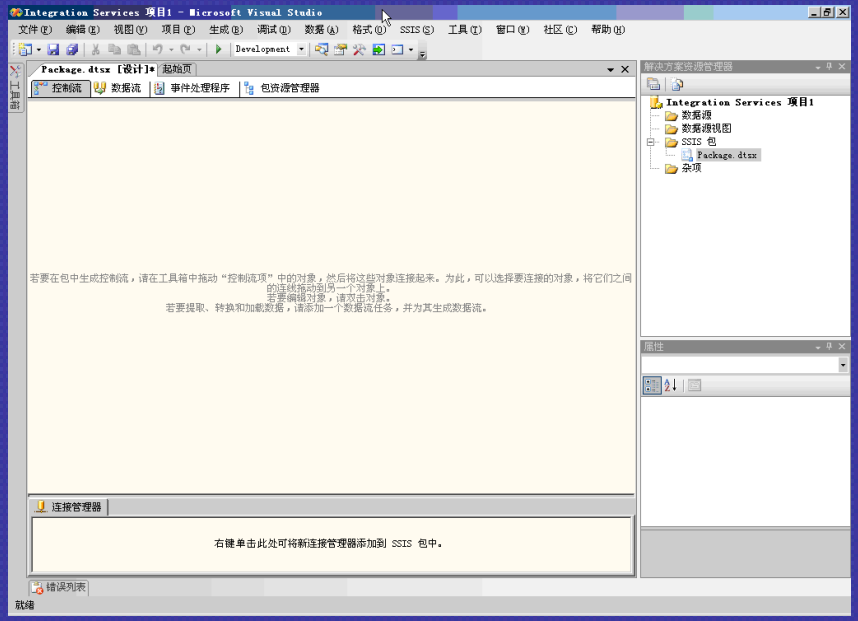


图11-33 “创建包”对话框

图11-34 “工具箱”中的控制流项对话框

11.2 使用图形设计界面来创建SSIS包

2. 定义和设置数据转换任务

（1）右键单击“连接管理器”区域中的任意位置，再单击“新建平面文件连接”，如图11-35所示。在“平面文件连接管理器编辑器”对话框的“连接管理器名称”字段中，键入名称text。单击“浏览”。在“打开”对话框中，浏览并找到数据文件夹，再打开相应的文件，如图11- 36所示。

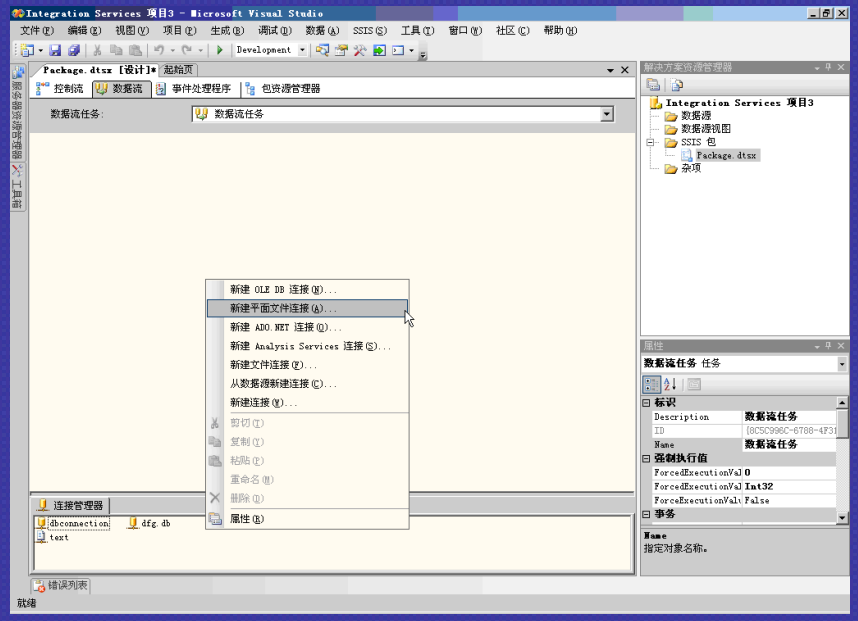


图11-35 打开“新建平面文件连接”对话框

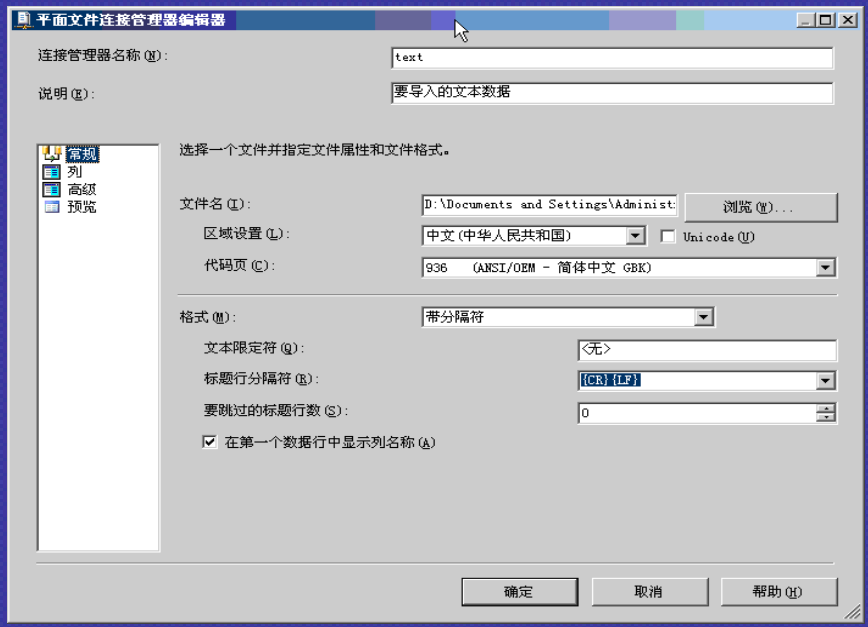


图11-36 “平面文件连接管理器”对话框

11.2 使用图形设计界面来创建SSIS包

2. 定义和设置数据转换任务

（2）右键单击连接管理器区域中的任意位置，再单击“新建 OLE DB 连接”。在“配置OLE DB连接管理器”对话框中，单击“新建”。在“服务器名称”中，输入本地服务器名称或输入localhost，如图11-37。在“配置OLE DB连接管理器”对话框的“数据连接”窗格中，如图11-38所示，确认选择了相应的服务器及数据库

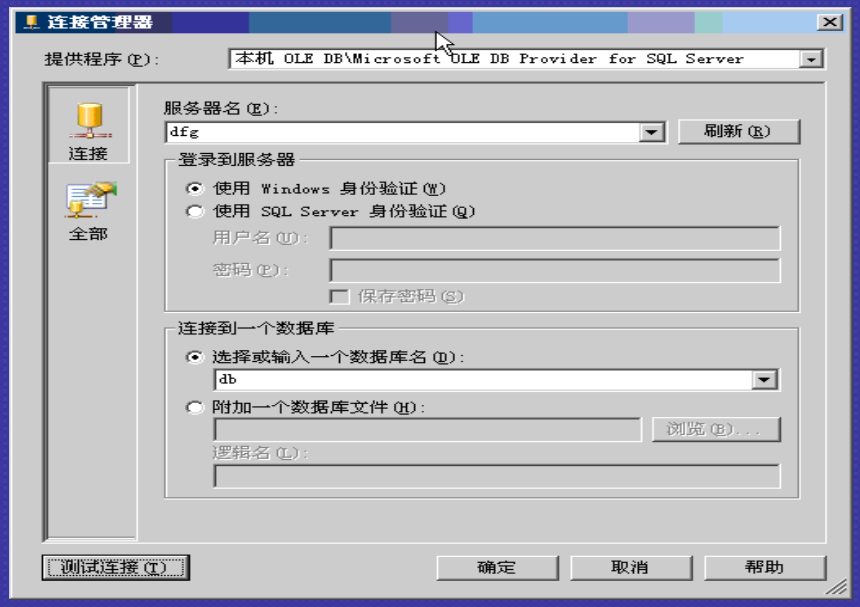


图11-37 “SQL Server连接管理器”对话框

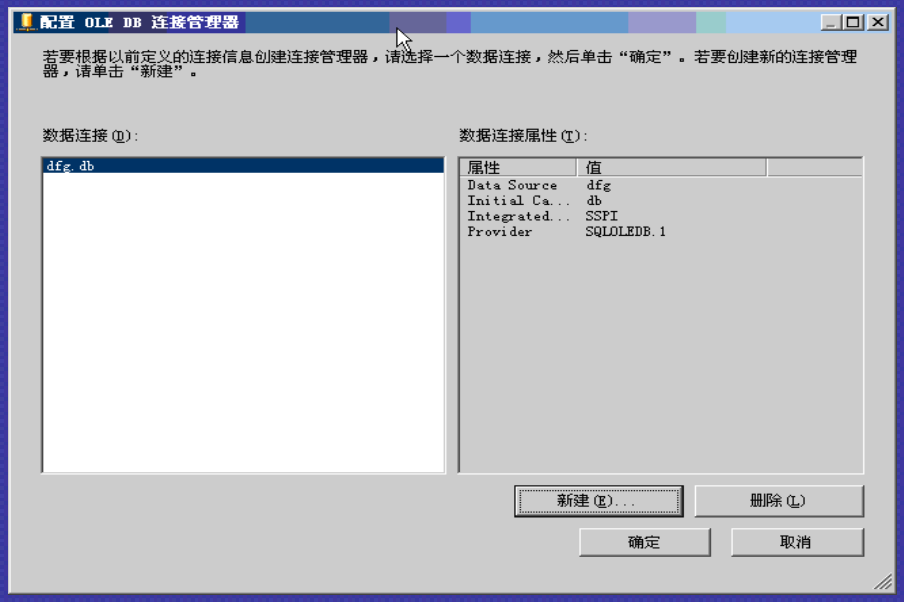


图11-38 配置OLE DB连接管理器窗口

11.2 使用图形设计界面来创建SSIS包

2. 定义和设置数据转换任务

（3）为源数据和目标数据创建了连接管理器后，下一个任务是在包中添加一个数据流任务。接下来向包中添加一个平面文件源并对其进行配置。如图11-39所示 。

（4）接下来，将目标数据库添加到数据流中。则数据流任务可表示为如图11-40所示。

（5）最后对包进行调试和运行。

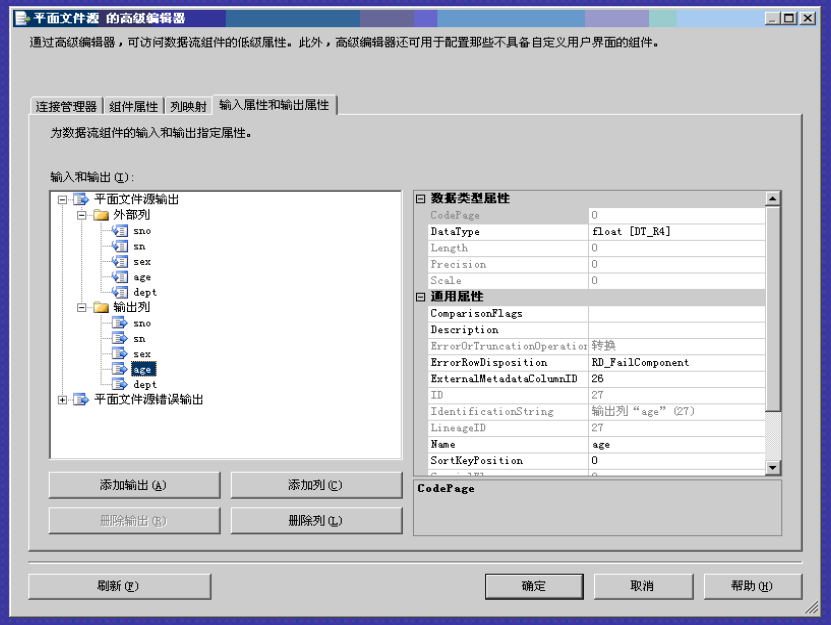


图11-39 平面文件源的高级编辑器窗口

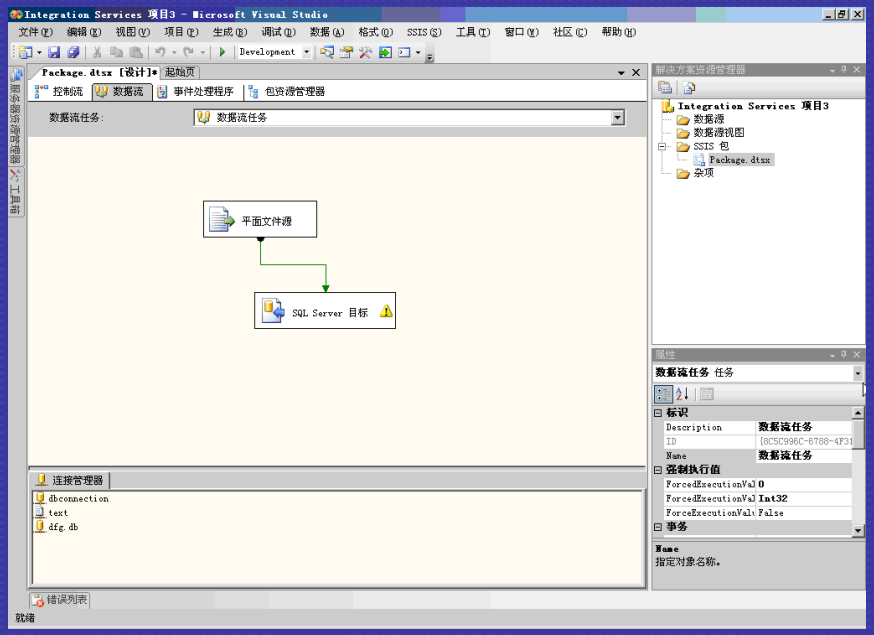


图11-40 数据流编辑器窗口

11.2 使用图形设计界面来创建SSIS包

3.定义和设置多任务

- 如果包中包含两个或更多任务，则可以通过将它们的连接线从一项拖动到其他项而将它们连接成控制流。两个项之间的连接器表示优先约束。优先约束定义了两个连接项之间的关系。它指定了运行时任务的执行顺序以及任务的运行条件。例如，优先约束可以指定某任务必须成功，才能运行控制流中的下一个任务。
- 如果在前面的数据转换任务成功完成后，需要完成另一个任务“执行T-SQL语句”，则在控制流界面中，可将这两个任务连接成控制流，如图11-41所示。

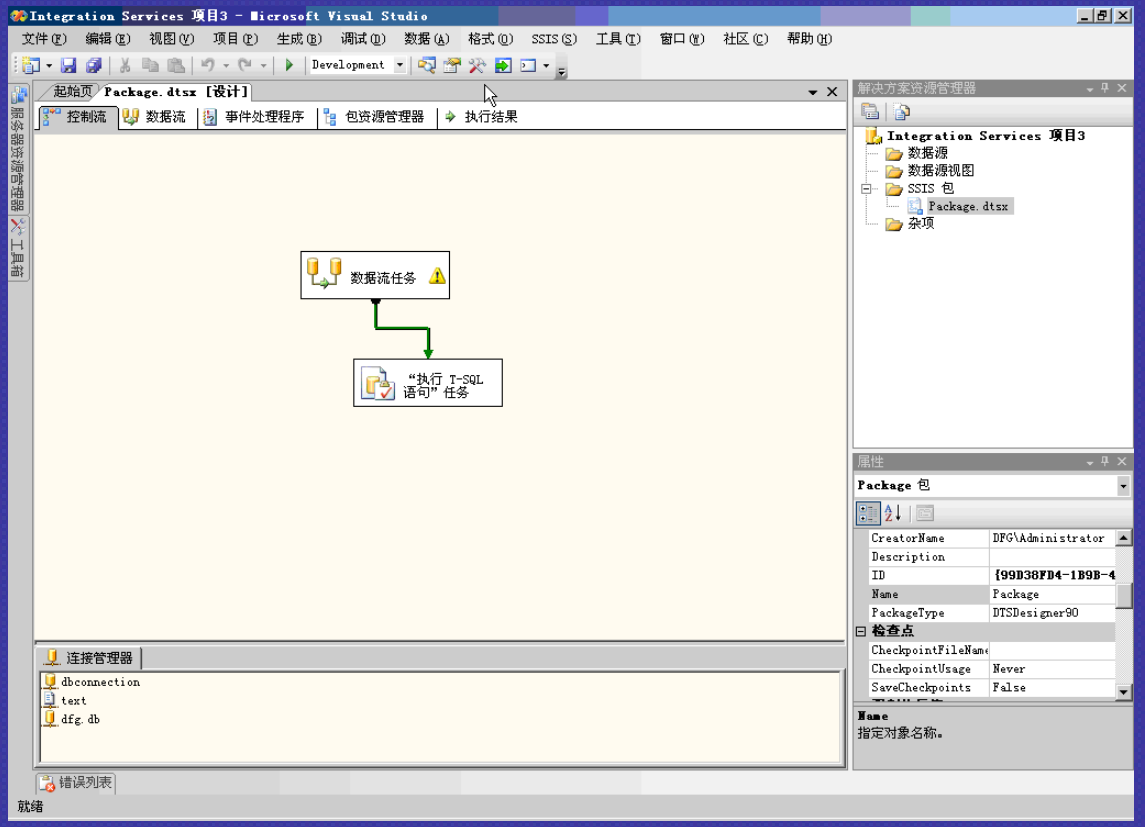


图11-41 数据流编辑器窗口

11.2 使用图形设计界面来创建SSIS包

3.定义和设置多任务

- 双击两个任务之间的连接线，打开“优先约束编辑器”窗口，如图11-42所示。在此窗口中可定义两个连接项之间的关系，例如前一个任务执行成功后，可执行后一个任务。
- 双击“执行T-SQL语句”任务图标，可编辑要执行的T-SQL语句，如图11-43所示。点击“确定”按钮完成编辑。

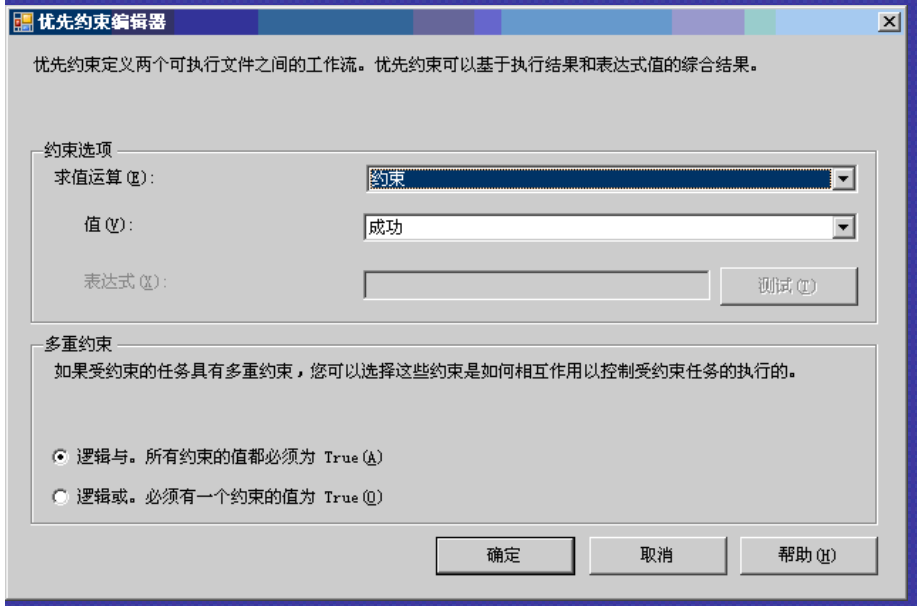


图11-42 “优先约束编辑器”窗口

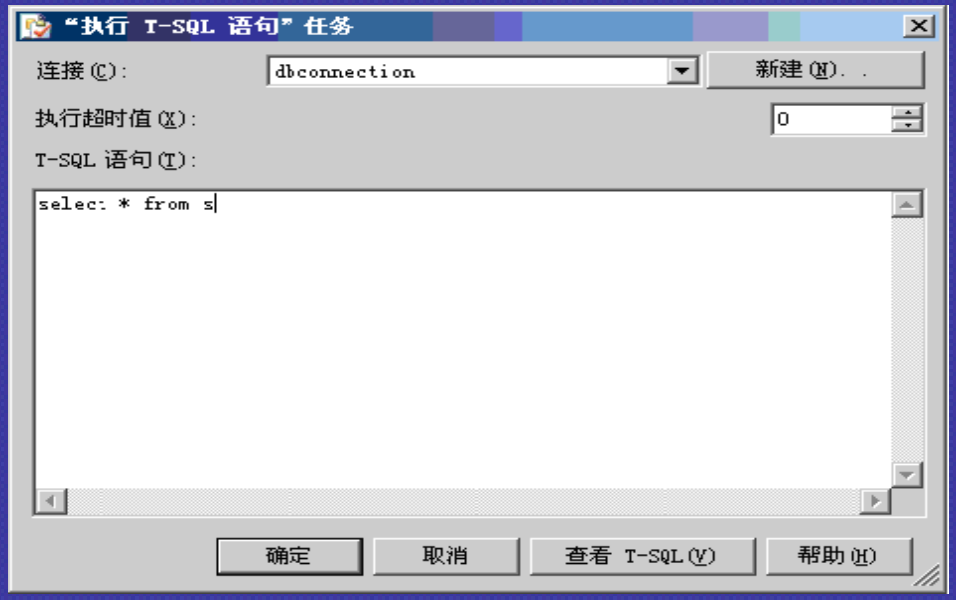


图11-43 编辑“执行T-SQL语句”任务窗口

第12章 代理服务

代理服务概述

- SQL Server 代理是一个任务规划器和警报管理器，在实际应用时，可以将那些周期性的工作定义成一个任务，在SQL Server代理的帮助下自动执行；在自动执行时，若出现故障，则SQL Server代理自动通知操作员，操作员获得通知后及时排除故障。如此在任务、操作员、警报三者之间既相互独立，又相互联系、相互补充，构成了自动完成某些任务的有机整体。

12.1 SQL Server代理服务配置

SQL Server 代理允许自动处理不同的管理任务，启动后可以利用SQL Server 管理平台对其进行配置，其具体步骤如下：

1.打开SQL Server 管理平台，展开指定的服务器，用右键单击SQL Server 代理图标，从快捷菜单中选择属性选项，则出现SQL Server 代理属性对话框，选择常规页框，如图12-1所示。

2.选择高级页框，如图12-2所示。

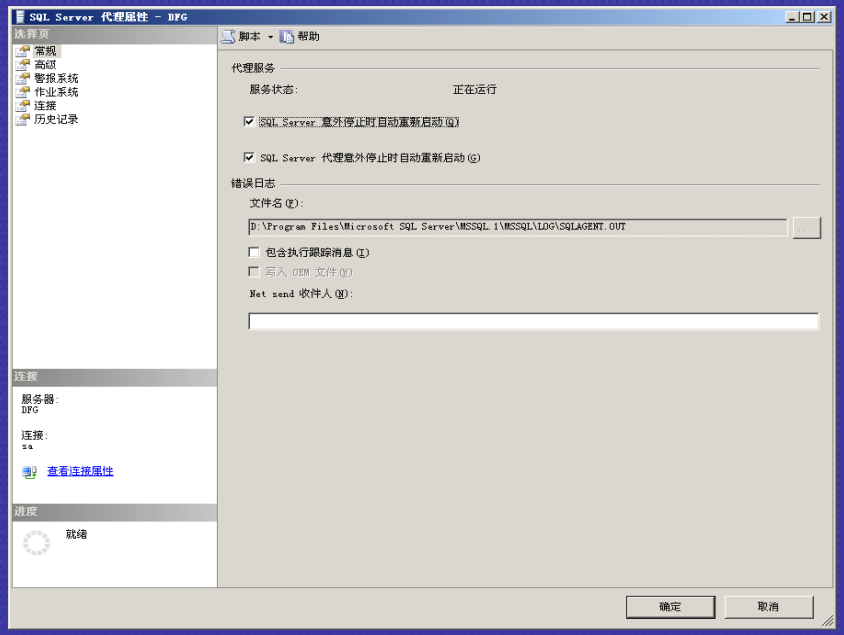


图12-1 SQL Server 代理属性—常规页框

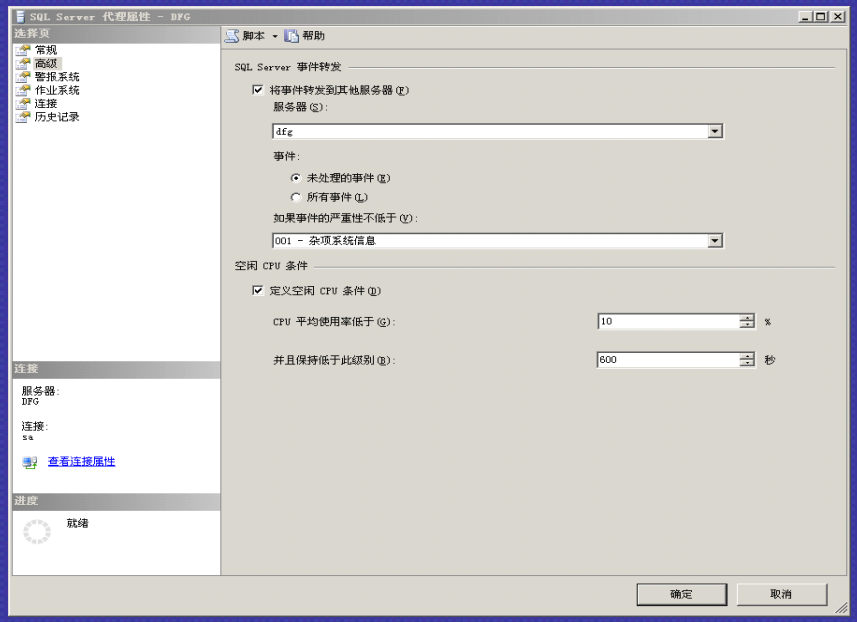


图12-2 SQL Server 代理属性—高级页框

12.1 SQL Server代理服务配置

3. 选择警报系统页框，如图12-3所示。使用此页可以查看和修改由SQL Server代理警报所发送的消息的设置。

4. 选择作业系统页框，如图12-4所示。

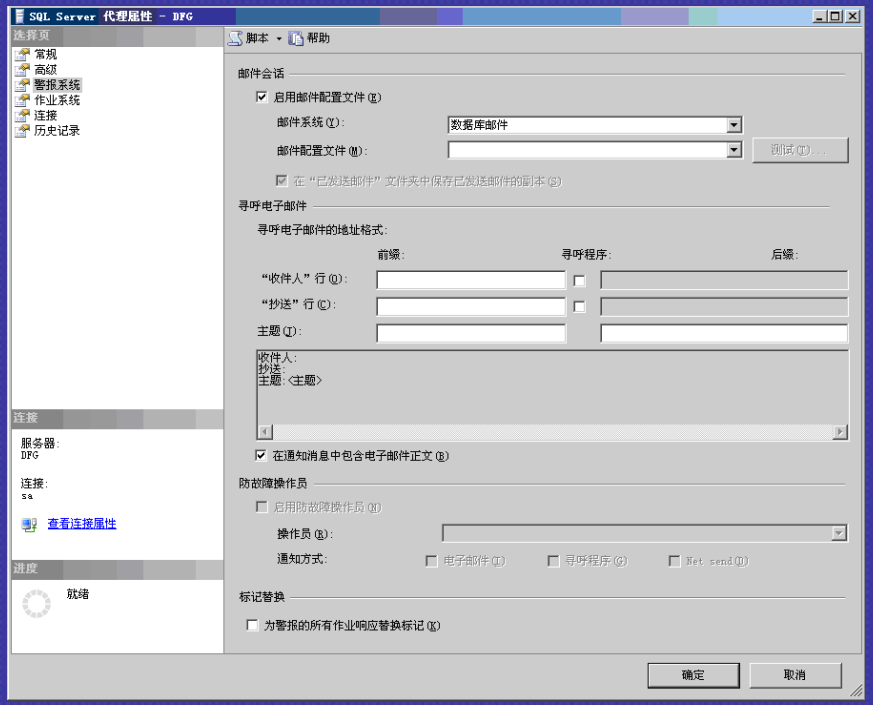


图12-3 SQL Server 代理属性—警报系统页框

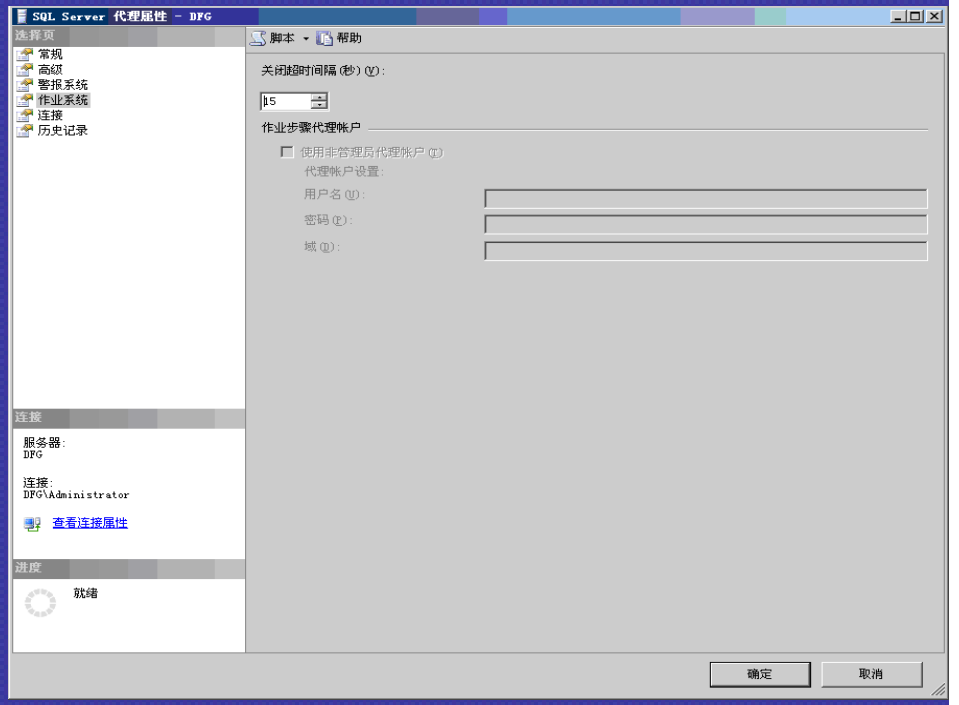


图12-4 SQL Server 代理属性—作业系统页框

12.1 SQL Server代理服务配置

5. 选择连接页框，如图12-5所示。使用此页可查看和修改 SQL Server代理服务与SQL Server 之间的连接设置。

6. 选择历史记录页框，如图12-6所示。使用此页可以查看和修改用于管理SQL Server代理服务历史记录日志的设置。

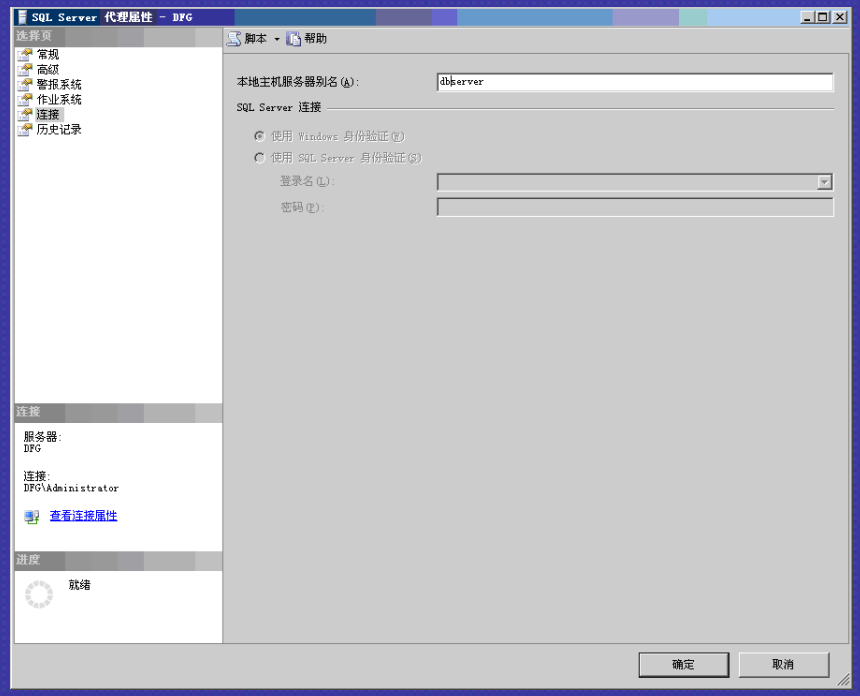


图12-5 SQL Server 代理属性—连接页框

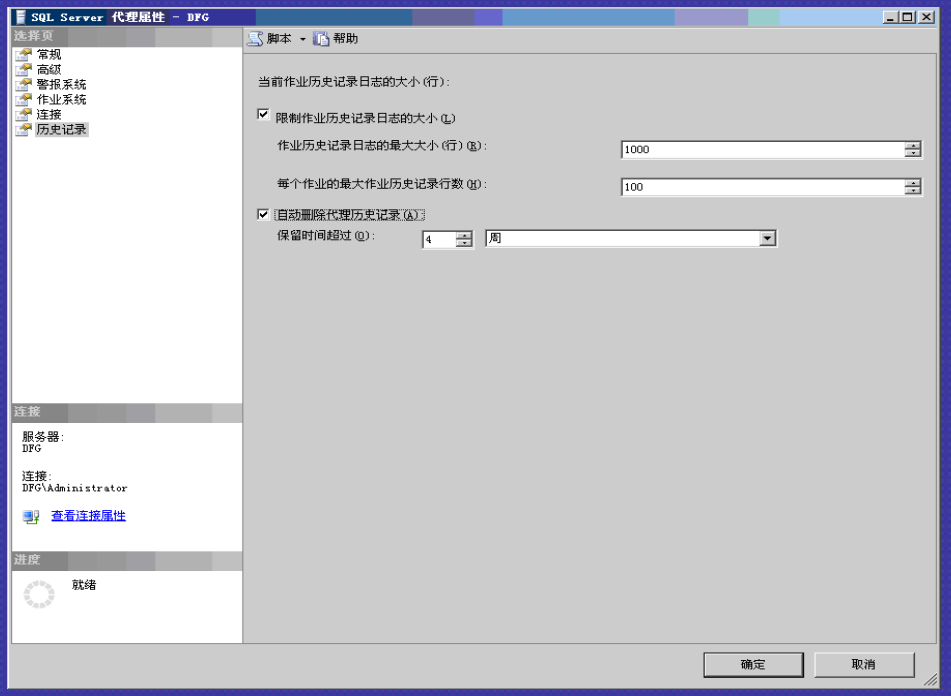


图12-6 SQL Server 代理属性—历史记录页框

12.2 定义操作员

使用SQL Server 管理平台可以创建一个操作员，其具体步骤如下：

1. 打开SQL Server管理平台，展开SQL Server 代理，右击操作员图标，从快捷菜单中选择新建操作员选项，则出现新建操作员属性对话框，如图12-7所示。
2. 选择通知页框，如图12-8所示，使用此页可设置向操作员通知的警报和作业。

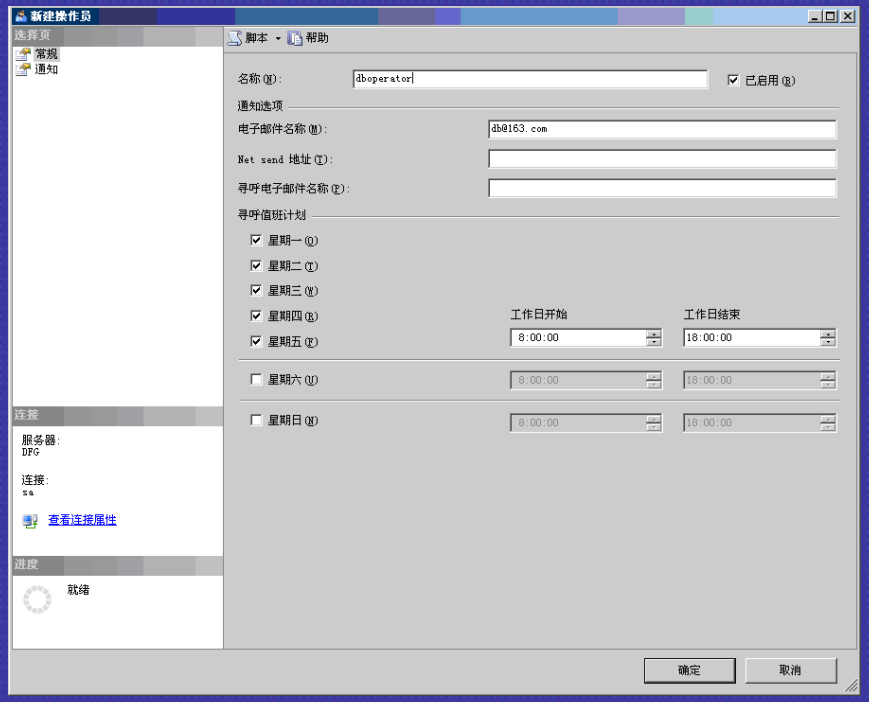


图12-7 新建操作员属性—常规页框

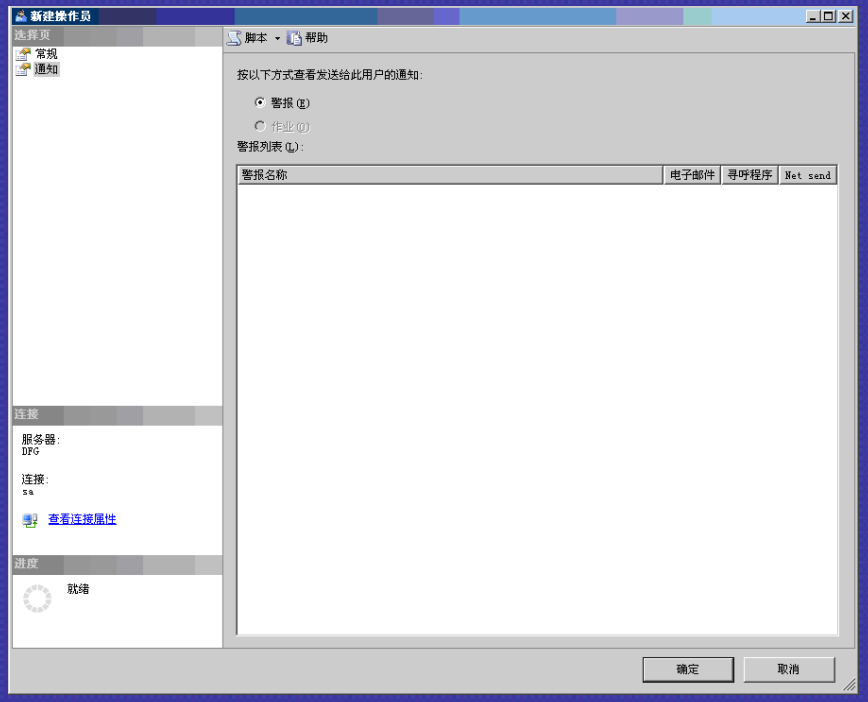


图12-8 新建操作员属性—通知页框

12.3 作业管理

- 自动处理一个任务的第一步是创建对应的作业。一般来说，如果要创建作业，必须执行以下三个步骤：

  （1）定义作业步

  （2）如果该作业不是用户指定执行，创建作业执行的计划时间

  （3）通知操作员作业的状态

12.3.1 作业步骤定义

- 作业步骤是作业对数据库或服务器执行的操作。每个作业必须至少有一个作业步骤。作业步骤可以为：

  （1）可执行程序和操作系统命令；

  （2）Transact-SQL 语句，包括存储过程和扩展存储过程；

  （3）Microsoft ActiveX 脚本；

  （4）复制任务；

  （5）Analysis Services 任务；

  （6）Integration Services 包。

12.3.2 创建作业

- 这里使用SQL Server 管理平台创建作业，其具体步骤如下：

1.打开已经启动的SQL Server代理，用右键单击作业图标，从快捷菜单中选择新建作业选项，则出现新建作业属性对话框，如图12-9所示。

2.每个作业必须有一个或者多个步骤，所以，除了定义作业属性外，在保存作业前，还至少要定义一个作业步骤。如图12-10所示。

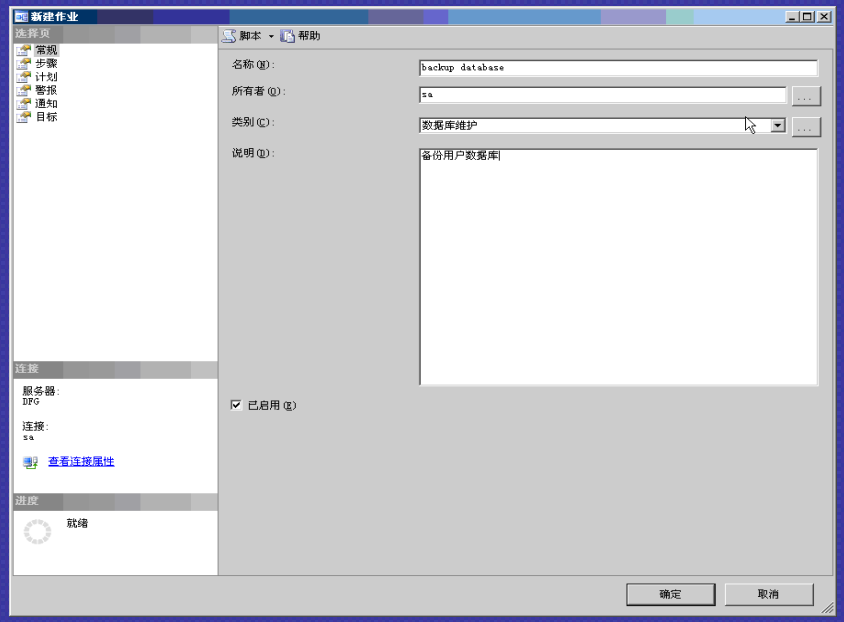


图12-9 新建作业属性对话框

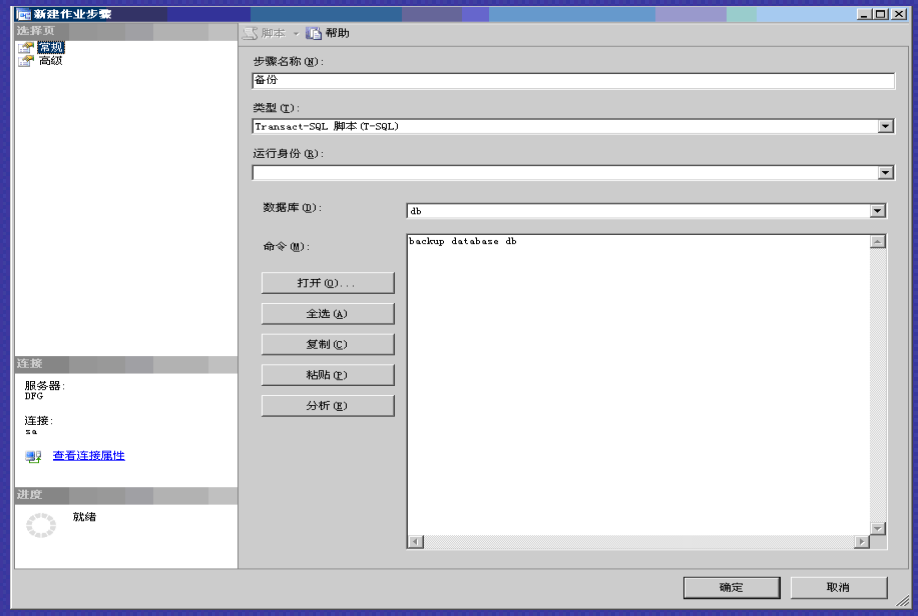


图12-10 新建作业步骤—常规页框

12.3.2 创建作业

3. 选择新建作业步骤对话框中的高级页框，如图12-11所示。
4. 选择新建作业中的计划页框，如图12-12所示。

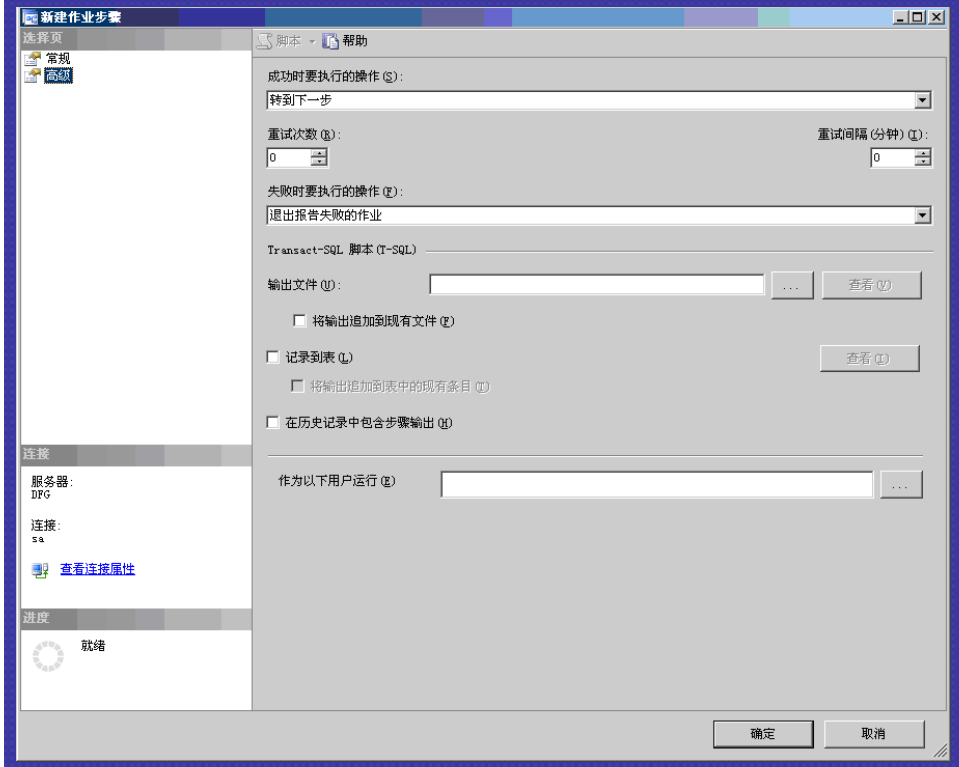


图12-11 新建作业步骤—高级页框

图12-12 新建作业计划对话框

12.3.2 创建作业

5. 选择新建作业中的通知页框，如图12-13所示，可以为现有的作业设置作业执行状态通知。

有关新建作业“警报”页框的内容将在下节介绍。

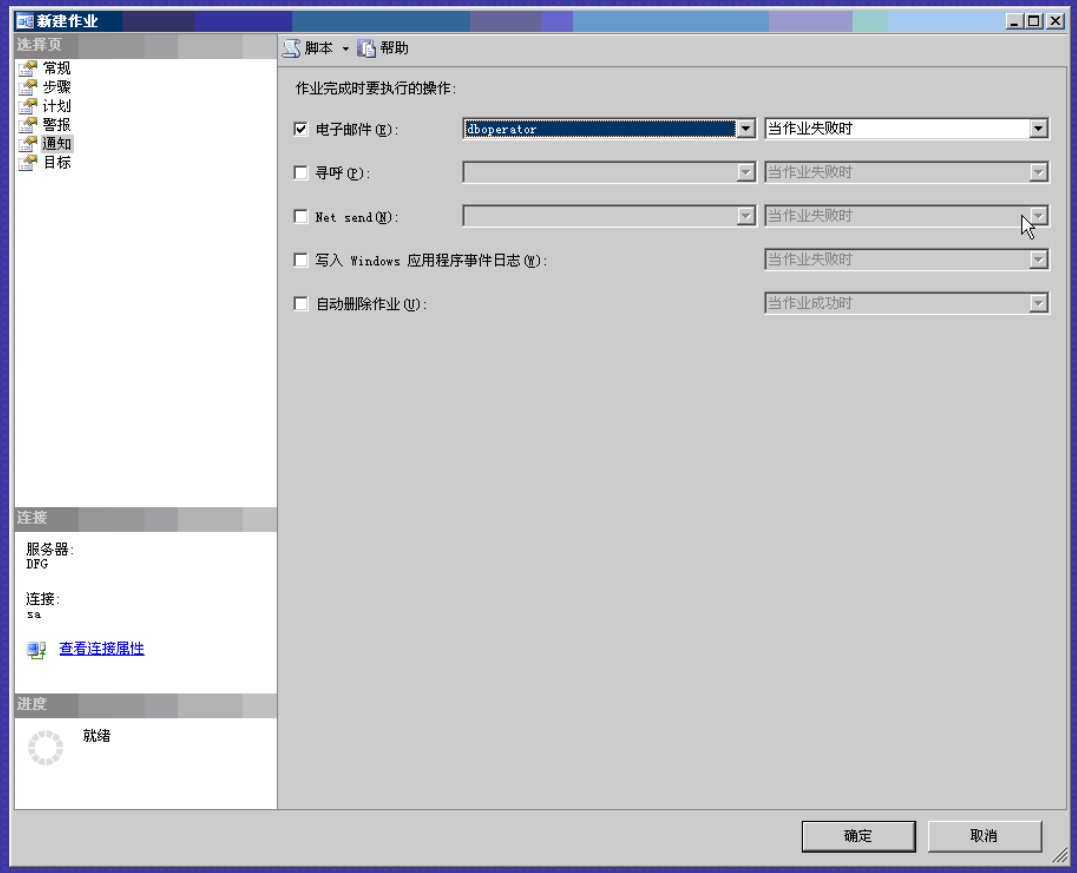


图12-13 新建作业属性—通知页框

12.4 警报管理

- 警报可以用于响应潜在的问题（如填满事务日志）。当警报被触发时，通过电子邮件，寻呼或者Net send通知操作员，从而让操作员了解系统中发生了什么事件。
- 可以定义一个警报，以便激活对特定的错误号或者属于特定严重级别的错误组的响应。警报可以使用SQL Server 管理平台定义。

12.4.1 创建事件警报

12.4.2 创建性能警报

12.4.1 创建事件警报

创建事件警报步骤如下：

（1）打开SQL Server管理平台，展开指定的服务器，然后展开启动的SQL Server 代理，用右键单击警报图标，从快捷菜单中选择新建警报选项，则出现新建警报属性对话框，从中选择常规页框，如图12-14所示。

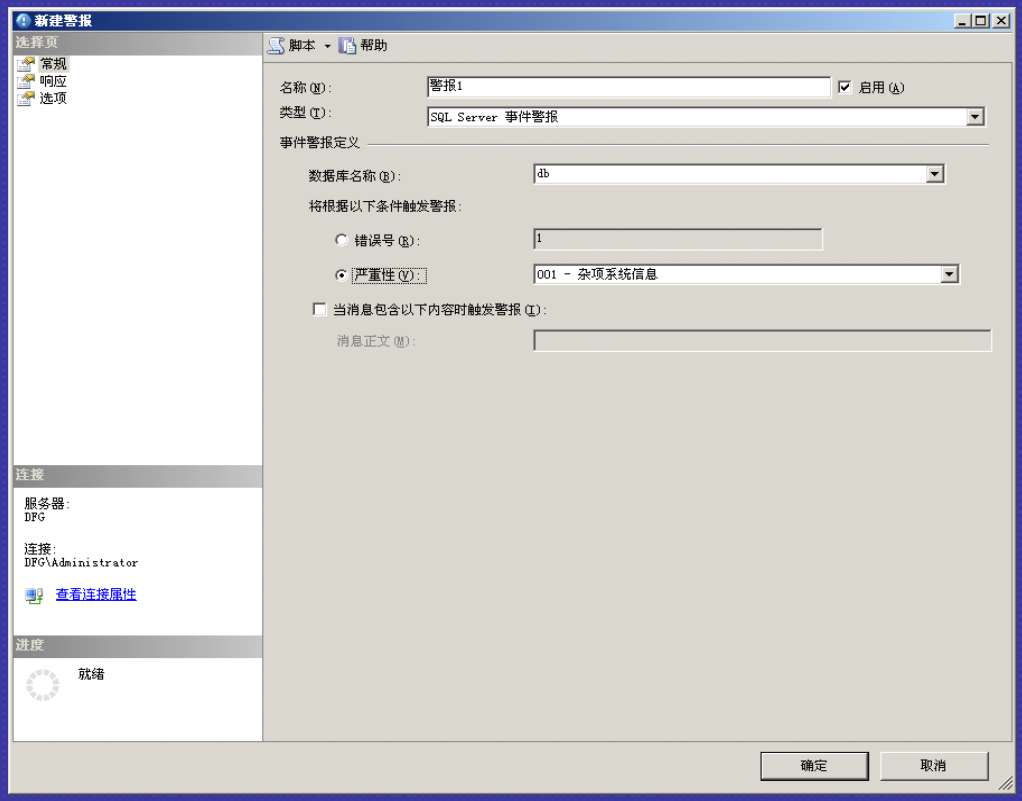


图12-14　新建警报属性—常规页框

12.4.1 创建事件警报

（2）选择响应页框，如图12-15所示。其中，“执行作业”下拉框用于选择出现警报时执行的作业；在要通知的操作员项下的表格中，用于显示把警报送给哪些操作者，并定义以哪种方式（电子邮件、寻呼、Net send）传送。

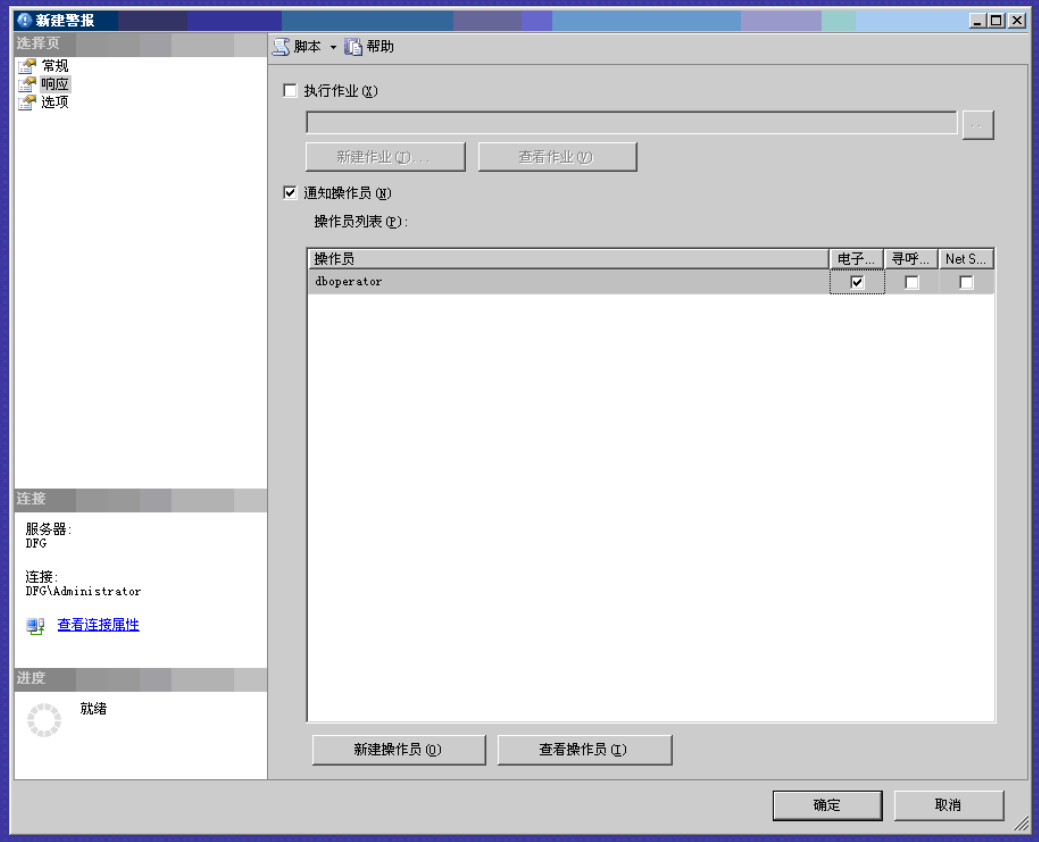


图12-15 新建警报属性—响应页框

12.4.1 创建事件警报

（3）选择选项页框，如图12-16所示。其中，“警报错误文本发送方式”用于选择把警报写入哪种（电子邮件、寻呼程序、Net send）通知当中；“要发送的其他通知消息”文本框用于输入传送给操作员的附加消息；“两次响应之间的延迟时间”表示警报连续两次响应的时间间隔。

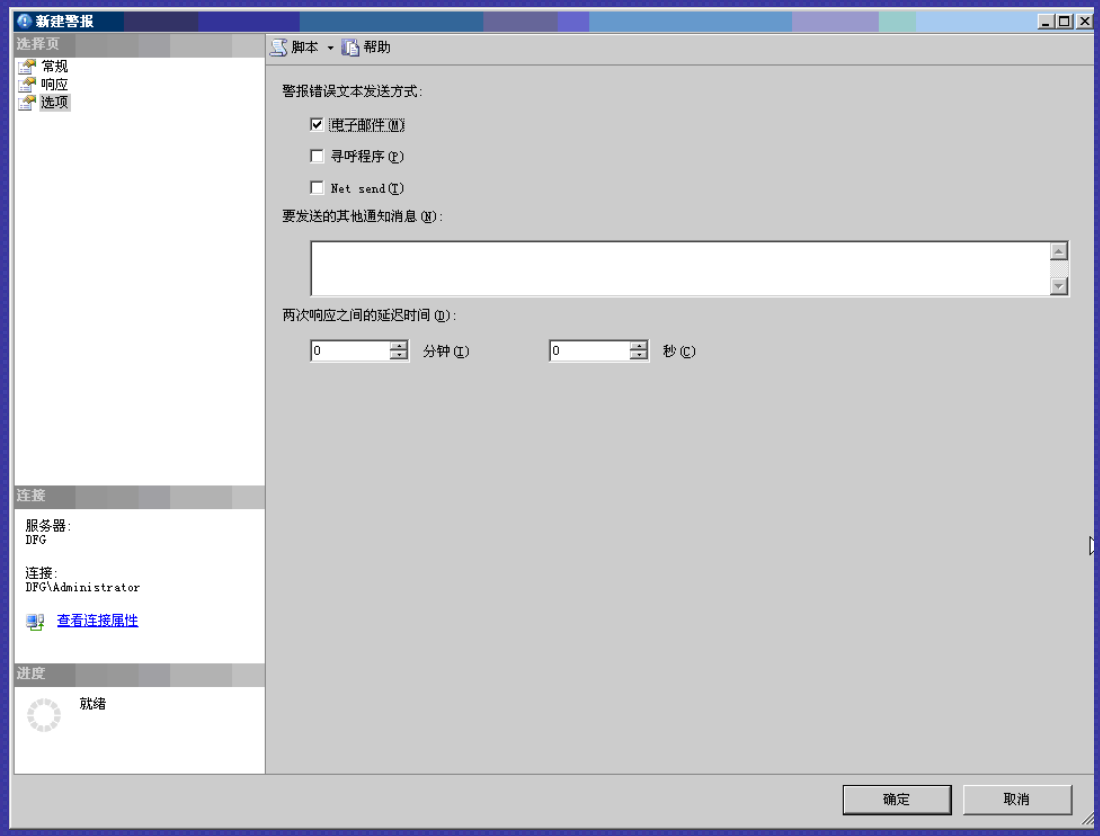


图12-16 新建警报属性—选项页框

第13章 报表服务管理

报表服务管理概述

- SQL Server 2005报表服务管理（SQL Server 2005 Reporting Services）是基于服务器的报表平台，可以用来创建和管理包含关系数据源和多维数据源中的数据的表格、矩阵、图形和自由格式的报表。
- 报表服务管理是一个基于服务器的企业级报表环境，可借助Web Services进行管理。报表可以用不同的格式发布，并可带多种交互和打印功能。可以通过把报表作为更进一步的商业智能数据源来分发，还可以包含复杂的分析来被更多的用户使用。
- 报表服务提供了以下功能：

  （1）用来处理和格式化报表的一个高性能引擎；

  （2）用来创建、管理和查看报表的一个完整的工具集；

  （3）可将报表解决方案嵌入或集成到不同IT环境中的一个可扩展架构和开放式接口。

13.1 报表服务配置

报表服务最终是作为一个Web服务实现的。报表服务配置的具体步骤如下：

（1）用鼠标点击“开始→程序”，然后选择“Microsoft SQL Server 2005”，接下来选择“配置工具”，然后选择报表服务配置。如下图13-1所示。接下来会出现如图13-2所示的选择报表服务器安装实例对话框。SQL Server 2005会根据系统相关配置给出相应的默认值。

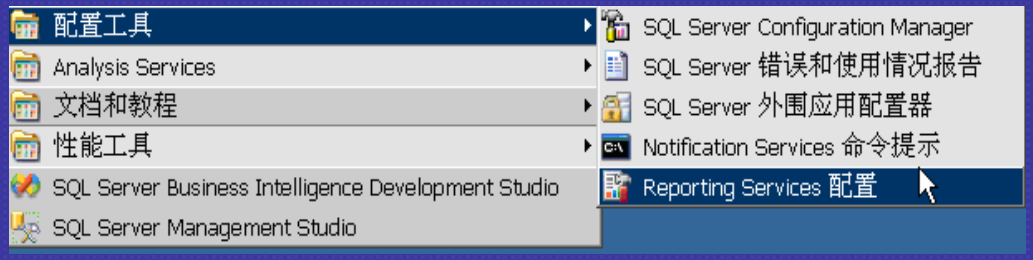


图13-1 进入Reporting Services 配置环境

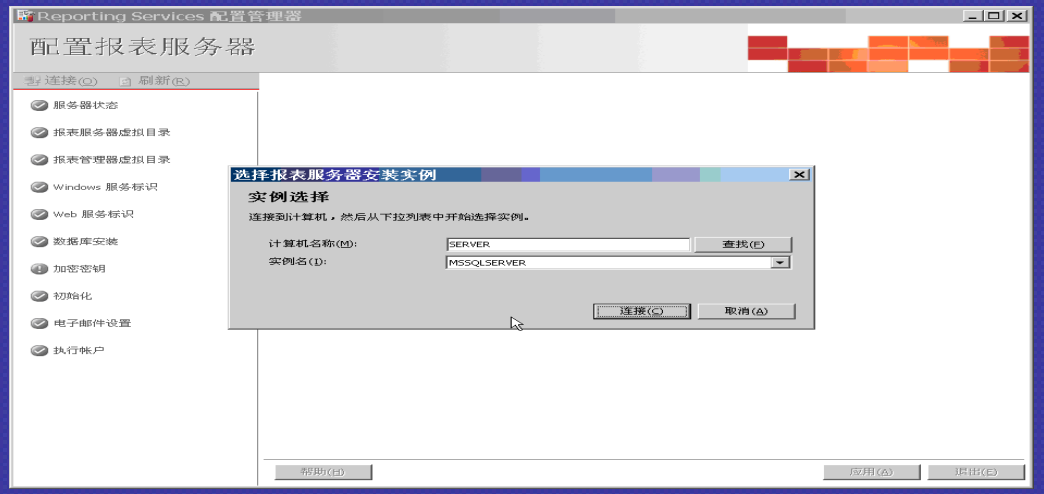


图13-2 选择报表服务器安装实例对话框

13.1 报表服务配置

报表服务最终是作为一个Web服务实现的。报表服务配置的具体步骤如下：

（1）用鼠标点击“开始→程序”，然后选择“Microsoft SQL Server 2005”，接下来选择“配置工具”，然后选择报表服务配置。如下图13-1所示。接下来会出现如图13-2所示的选择报表服务器安装实例对话框。SQL Server 2005会根据系统相关配置给出相应的默认值。

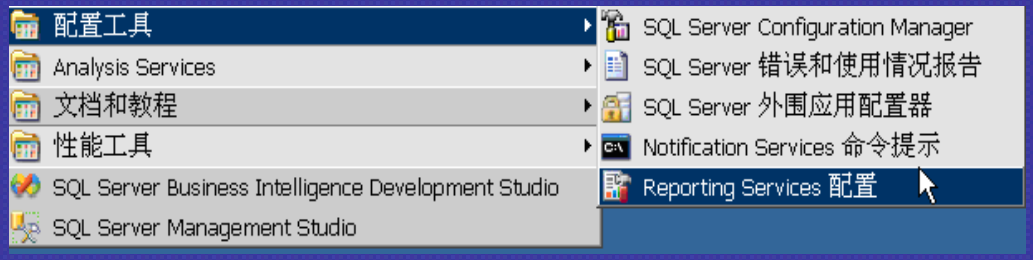


图13-1 进入Reporting Services 配置环境

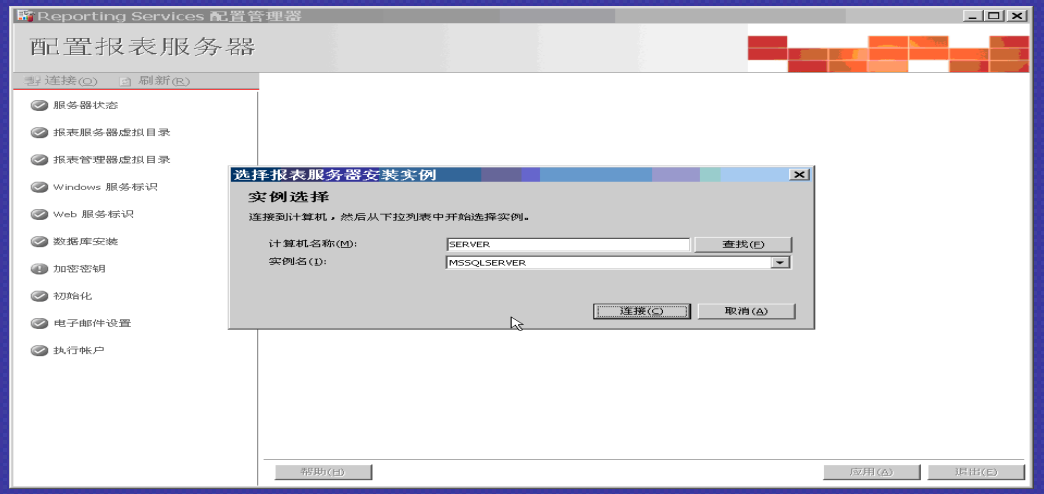


图13-2 选择报表服务器安装实例对话框

13.1 报表服务配置

（2）确定后点击连接即可进入到图13-3所示的配置报表服务器窗口。

图13-3 配置报表服务器窗口

13.1 报表服务配置

（3）点击左边功能选项中的第二项报表服务器虚拟目录配置选项，会出现如图13-4所示的配置选项界面。点击右面名称后面的新建按钮，则会出现如图13-5所示的创建新的虚拟目录配置选项对话框。

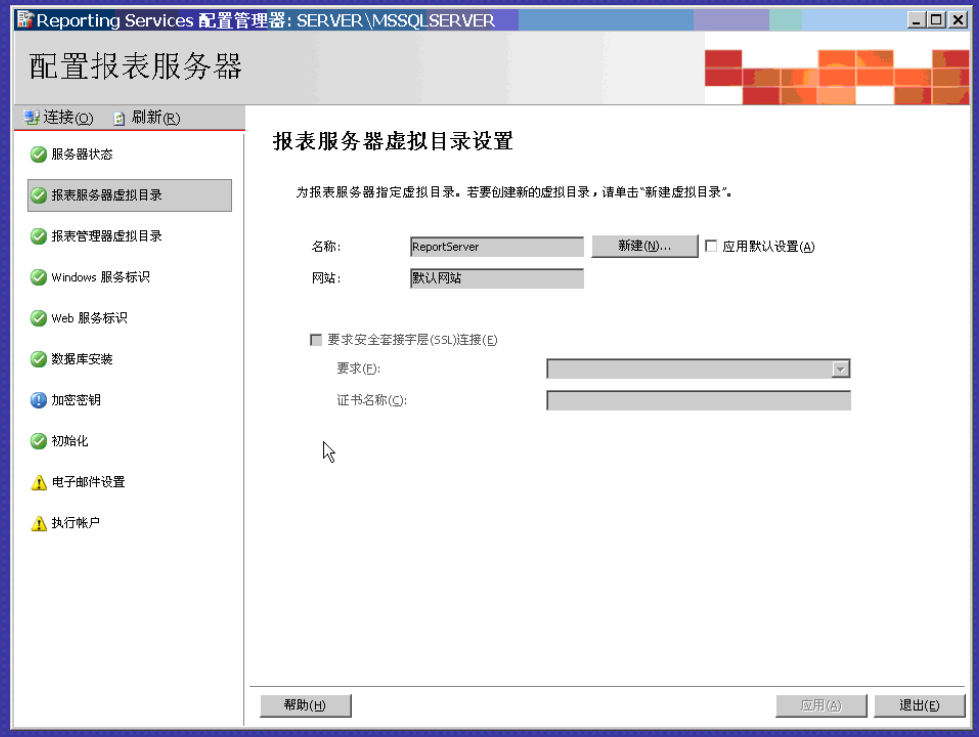


图13-4 报表服务器虚拟目录配置选项界面

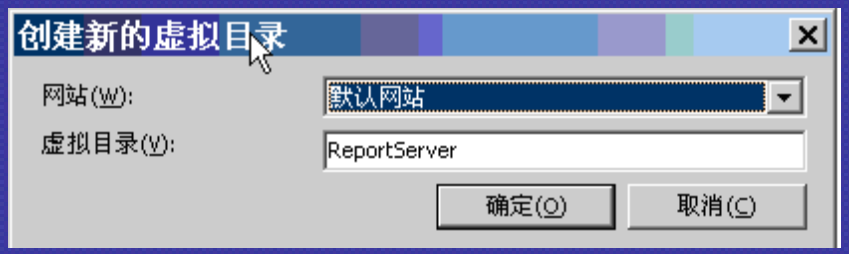


图13-5 创建新的虚拟目录配置选项对话框

13.1 报表服务配置

（4）点击数据库安装配置选项，则会出现如图13-6的数据库连接配置界面，点击服务器名称旁边的连接按钮，出现如图13-7所示的SQL Server连接对话框。

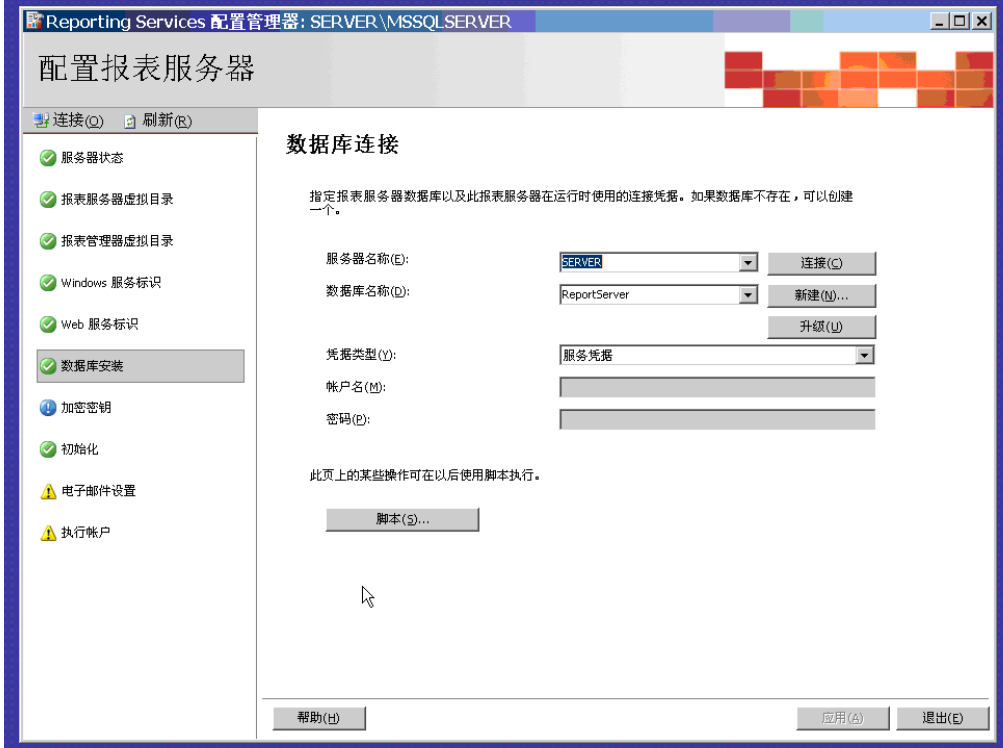


图13-6 数据库连接界面

图13-7 SQL Server连接对话框

13.1 报表服务配置

（5）点击左边配置选项中的初始化配置选项。可以进入到如图13-9所示的初始化界面。确认以上的选项都配置好之后，点击界面上的初始化按钮即可将上述一系列的配置选项进行初始化成实例供报表项目发布使用。至此已经完成了最基本的报表服务配置。其余的几项均为可选配置，如有需要可以参考前面的功能配置说明进行相应的配置。

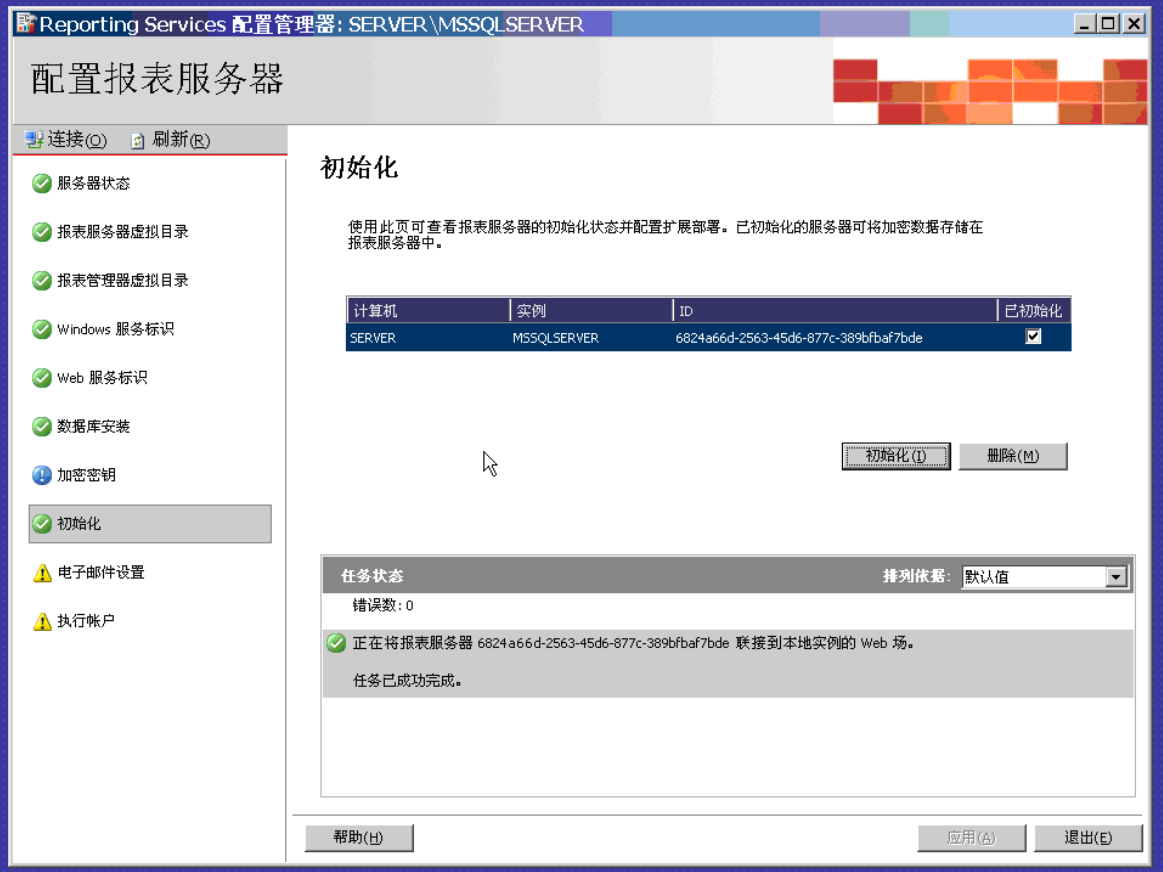


图13-9　配置报表服务器初始化界面

13.2 创建报表服务器项目

- 配置好报表服务环境之后，就可以进行报表服务器项目的开发了。报表服务器项目就是将报表先在本地机上创建之后再在服务器上发布的一系列过程。
- 利用SQL Server 2005 进行报表项目的设计有两种方式：

（1）通过报表设计器创建报表并发布；

（2）通过报表生成器生成报表后发布。

- 在报表设计器中，创建报表的方法又主要有以下三种：

（1）创建空白报表，然后手动添加查询和布局；

（2）使用报表向导，根据提供的信息自动创建表或矩阵报表；

（3）从Microsoft Access导入现有的报表。

13.2.1 通过报表设计器创建报表

1.创建空白报表，然后手动添加查询和布局。这种方式是创建报表最主要的方式，具有很强的应用性和灵活性。

（1）首先点击开“开始→程序”，如图13-10所示。然后选择“Microsoft SQL Server 2005”，接下来选择“SQL Server Business Intelligence Development Studio”，可以打开如图13-11所示的Microsoft Visual Studio开发环境，并且显示开发环境默认的起始页。点击菜单栏上的“文件”，然后选择“新建”，再选择“项目”。

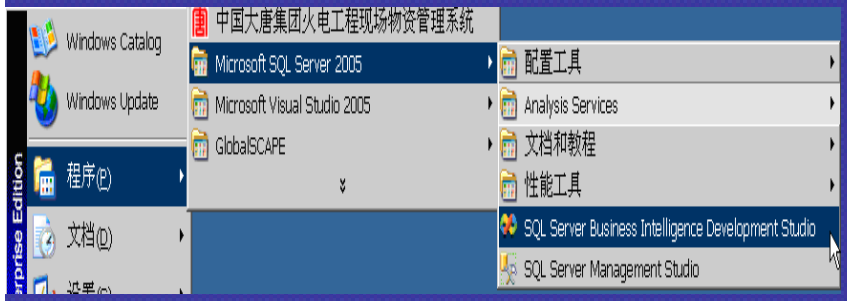


图13-10 进入报表设计器环境

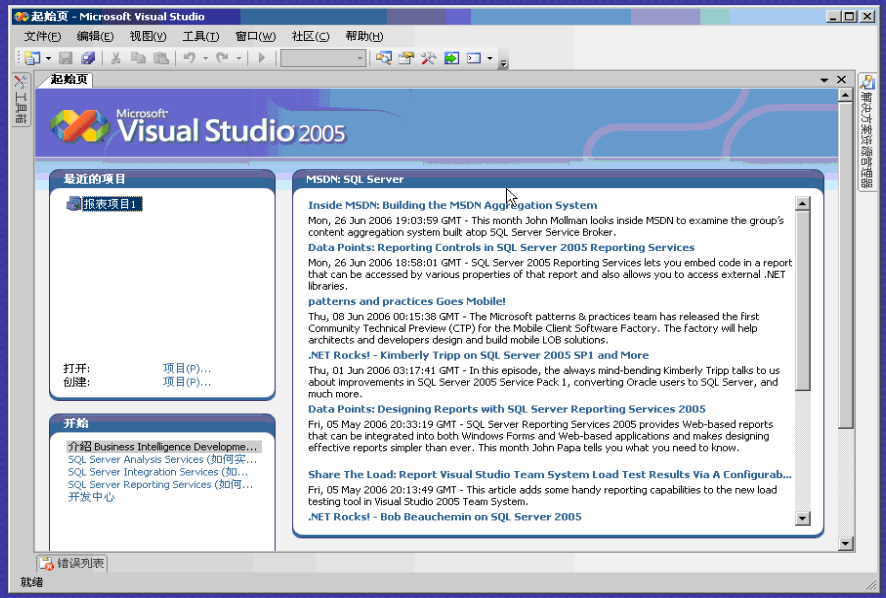


图13-11 Microsoft Visual Studio开发环境

13.2.1 通过报表设计器创建报表

1.创建空白报表，然后手动添加查询和布局。

（2）打开如图13-12所示的新建项目对话框。在左边的项目类型中选择“商业智能项目”，在右边的“模板”列表中选择“报表服务器项目”，然后在下面的名称框中输入报表项目的名称。点击右边“解决方案管理器”可以显示如图13-13的example项目的解决方案。

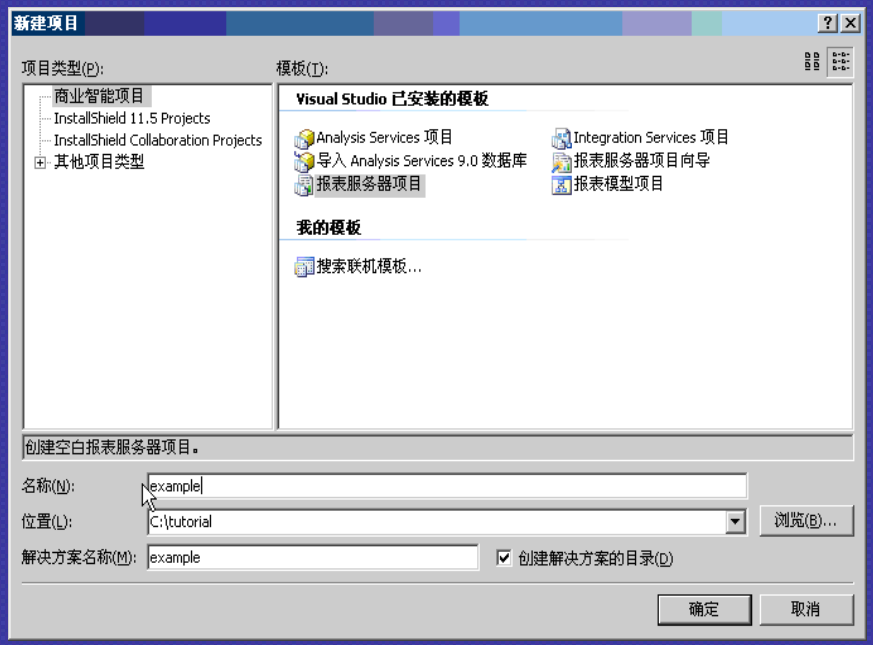


图13-12 新建项目对话框

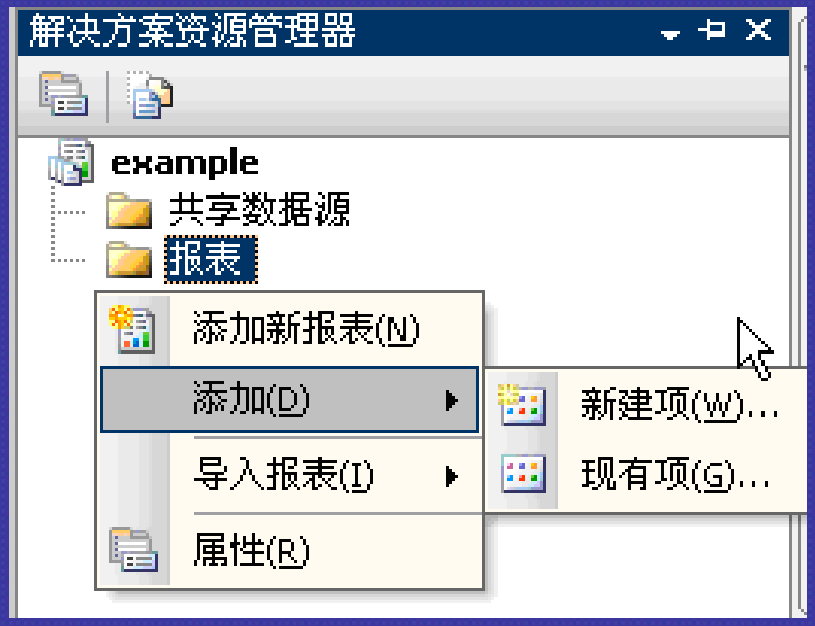


图13-13 解决方案资源管理器界面

13.2.1 通过报表设计器创建报表

1.创建空白报表，然后手动添加查询和布局。

（3）在报表上单击右键弹出的快捷菜单中选择“添加”，再选择“新建项”则会出现如图13-14所示的添加新项的对话框。在模板中单击“报表”选项，在名称中输入报表模板的名称。点击添加按钮后系统将打开一个包含“数据”、“布局”和“预览”选项卡的视图。系统将在“数据”视图中打开此报表，如图13-15所示。

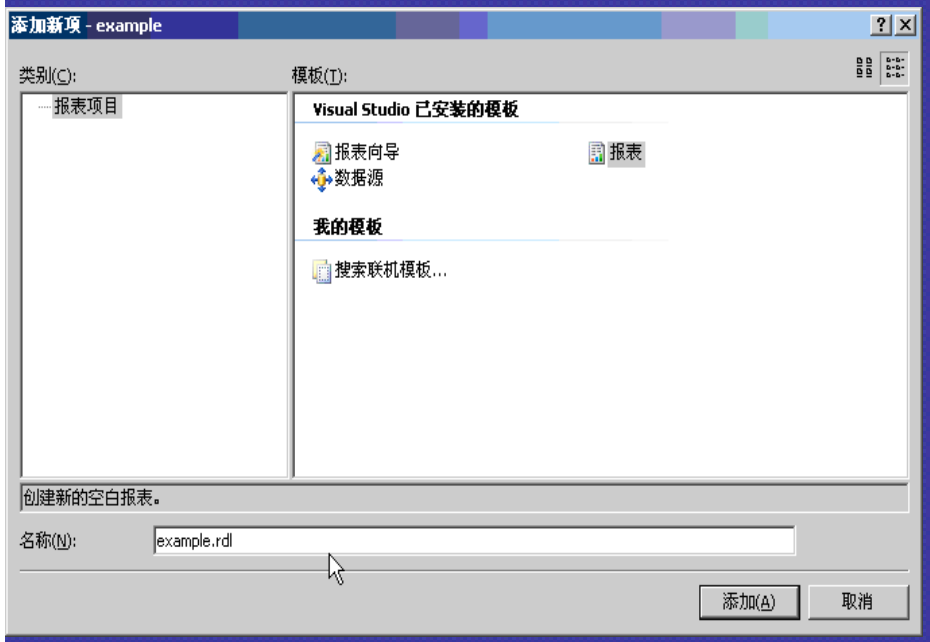


图13-14 添加新项对话框

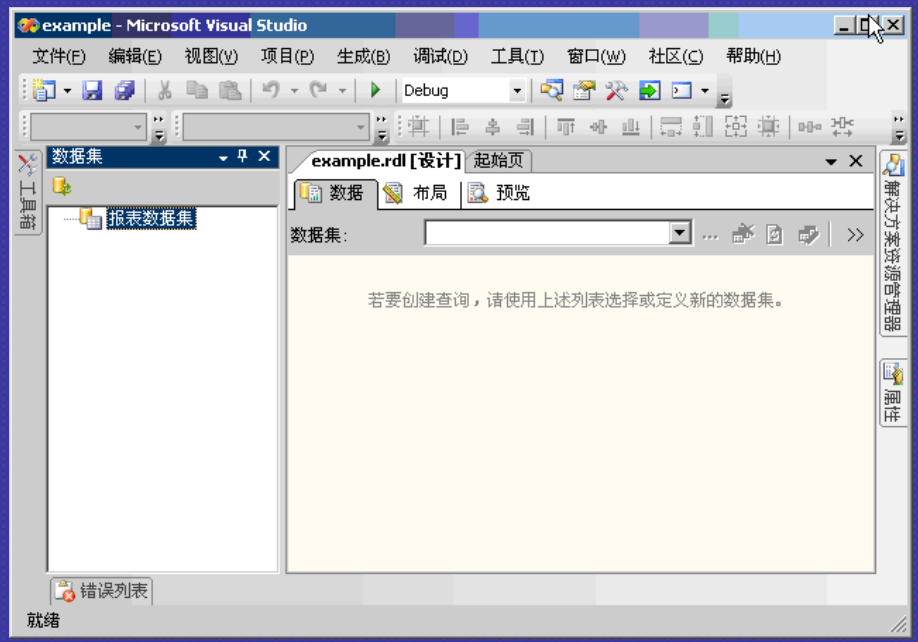


图13-15 在数据视图中显示的报表设计界面

13.2.1 通过报表设计器创建报表

1.创建空白报表，然后手动添加查询和布局。

（4）点击数据集旁边的下拉列表框，并选择“新建数据集”。此时，系统将显示如图13-16所示的“数据源”对话框。点击连接字符串旁边的编辑按钮，则弹出如图13-17所示的连接属性选择框。

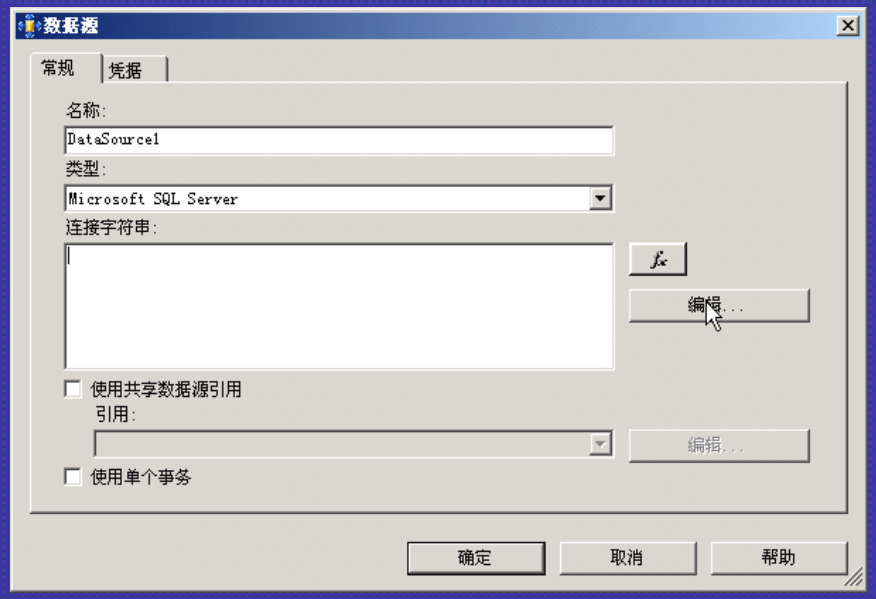


图13-16 数据源设置对话框

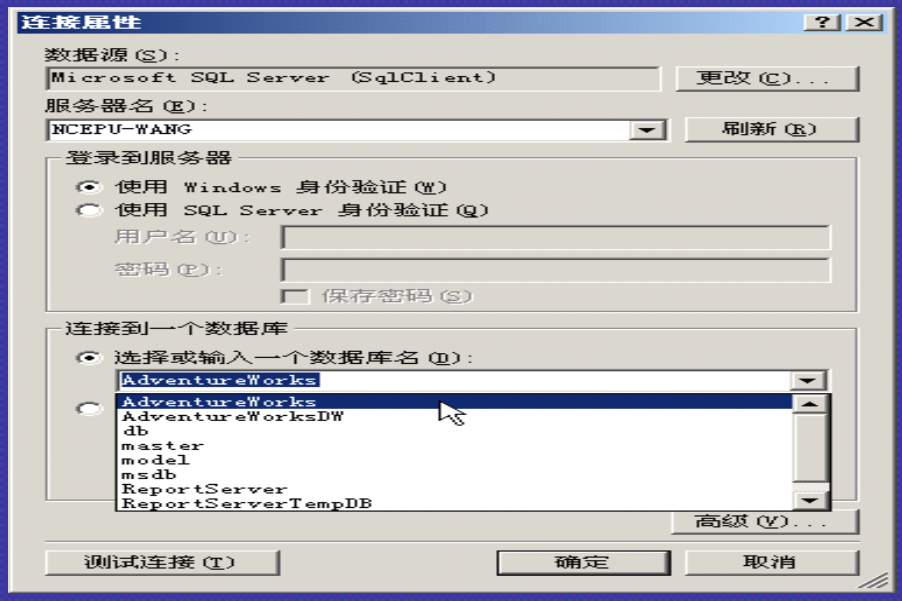


图13-17 连接属性设置对话框

13.2.1 通过报表设计器创建报表

1.创建空白报表，然后手动添加查询和布局。

（5）图13-18上面的窗格称为通用查询设计器的 SQL 窗格；下面的表格称为查询结果显示窗格。接下来点击图13-18中的布局选项卡切换到布局设计选项界面中。报表设计器将在设计图面的宽度内绘制一个具有三列的基本表，如图13-19所示。

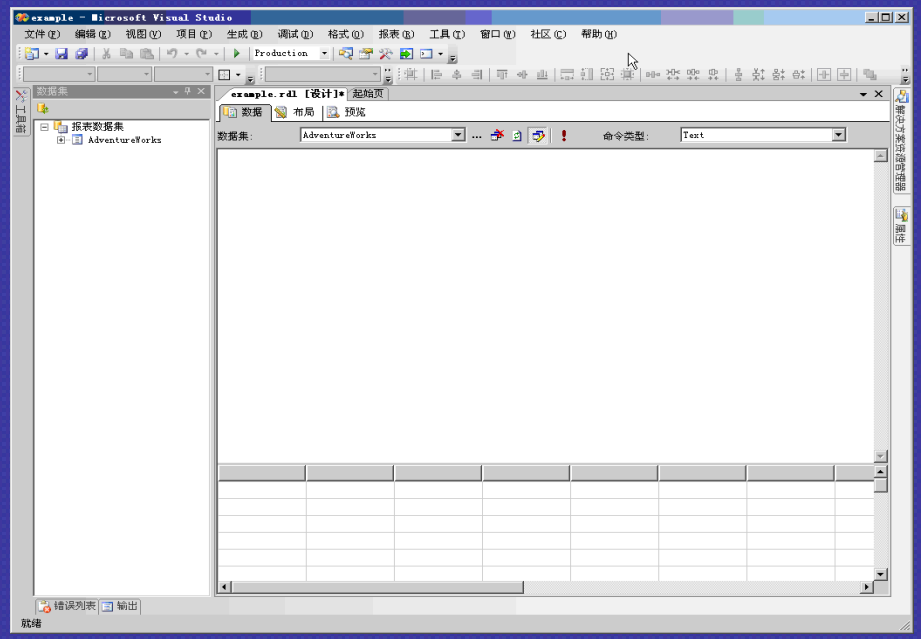


图13-18 设置数据源之后的报表数据视图界面

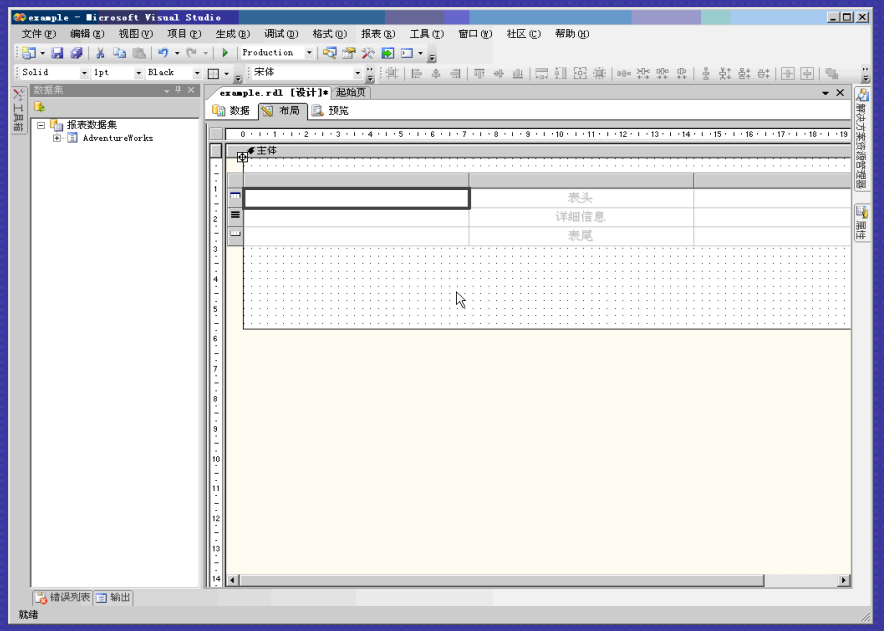


图13-19 报表布局设计界面

13.2.1 通过报表设计器创建报表

1.创建空白报表，然后手动添加查询和布局。

（6）点击图13-19左侧“报表数据集”窗口中的报表数据集下的AdventureWorks数据库以显示字段。设置好的结果如图13-20所示。此外，还可以对单个单元格的格式进行设置。如图13-21所示的“文本框属性”对话框。

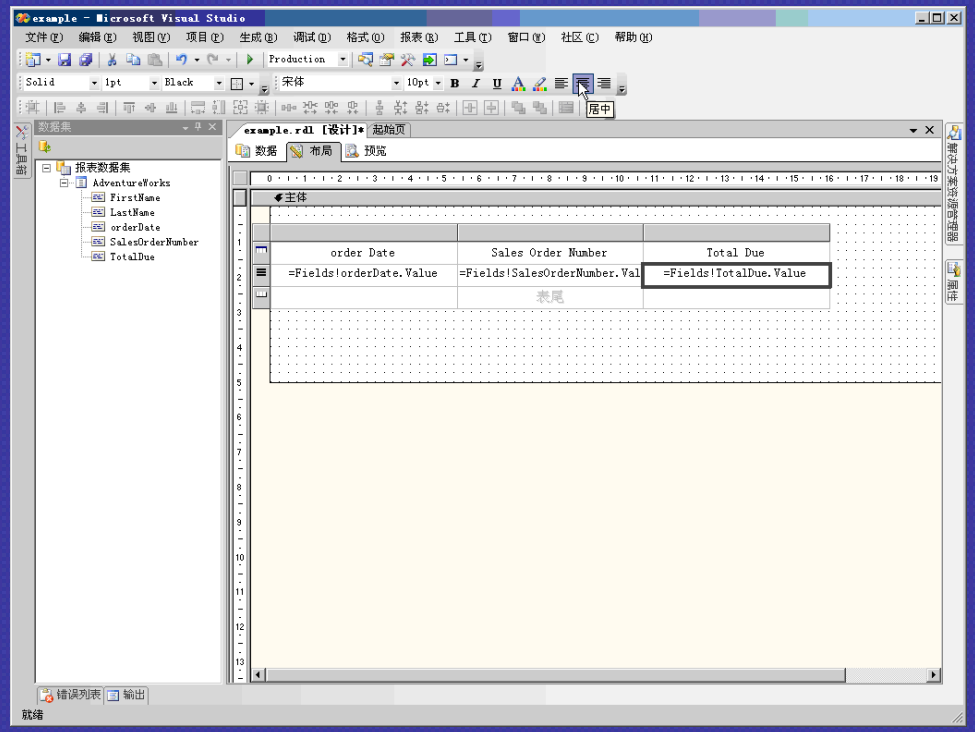


图13-20 添加三列之后的报表布局设计界面

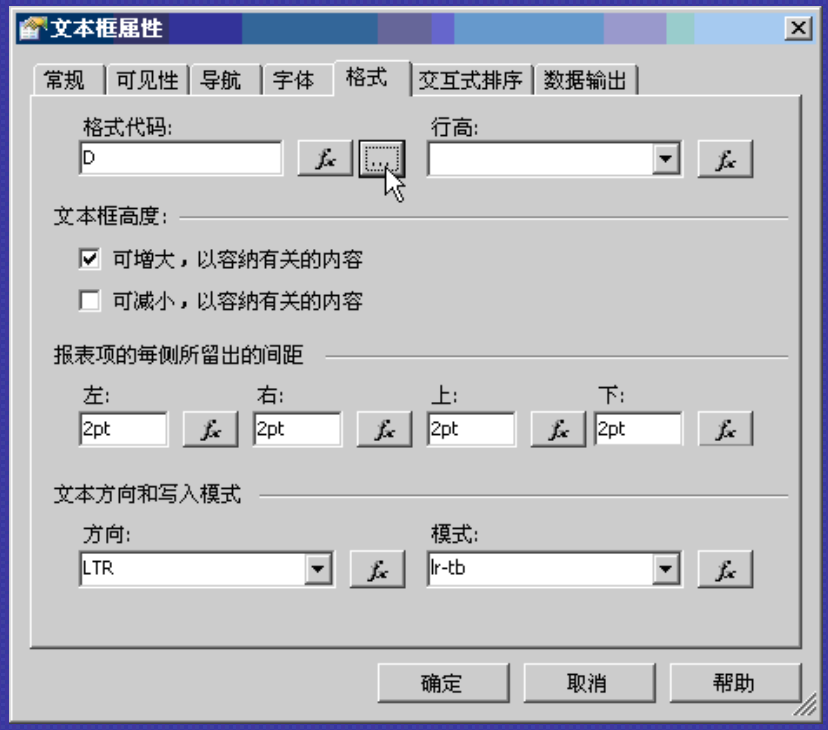


图13-21 文本框属性对话框

13.2.1 通过报表设计器创建报表

1.创建空白报表，然后手动添加查询和布局。

（7）点击图13-21所示格式代码选项旁边的浏览按钮，可以弹出如图13-22所示的选择格式对话框。此外，可以单击该单元格，点击右键，在弹出的快捷菜单中选择“表达式”，则出现如图13-23所示的编辑表达式对话框。

图13-22 选择格式对话框

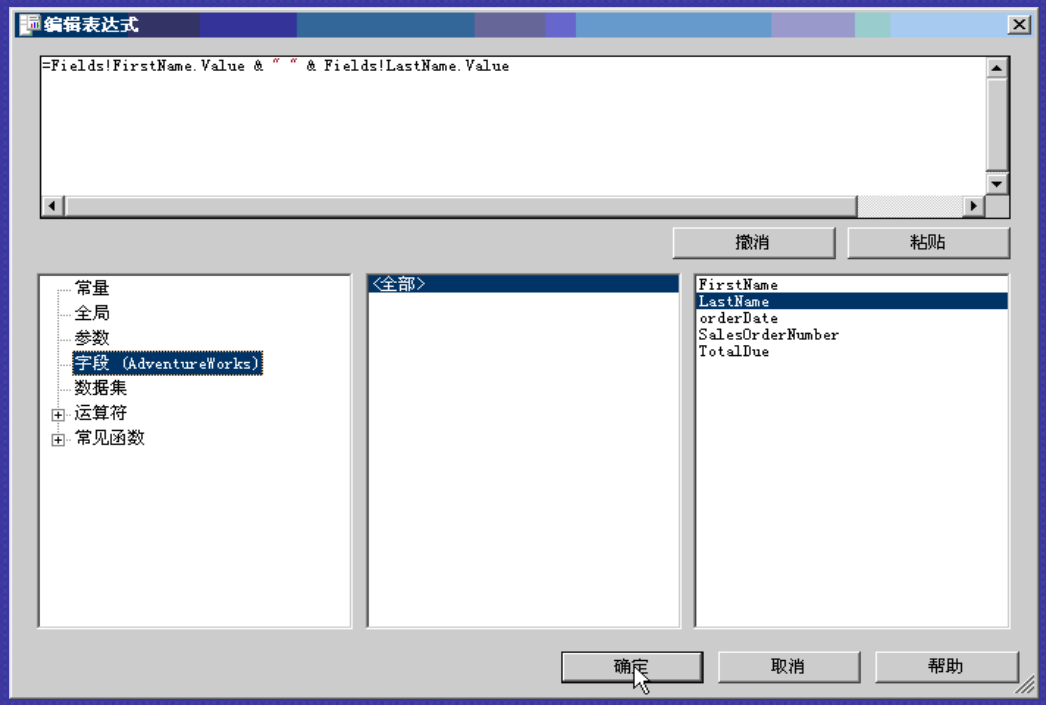


图13-23 编辑表达式对话框

13.2.1 通过报表设计器创建报表

1.创建空白报表，然后手动添加查询和布局。

（8）如果想对报表中的合同签订的雇员进行排序，可以用右键单击角部句柄，再单击“属性”按钮。可以弹出如图13-24所示的表属性对话框。点击预览选项页面，等待系统处理一段时间之后可以生成如图13-25所示的最终预览界面。

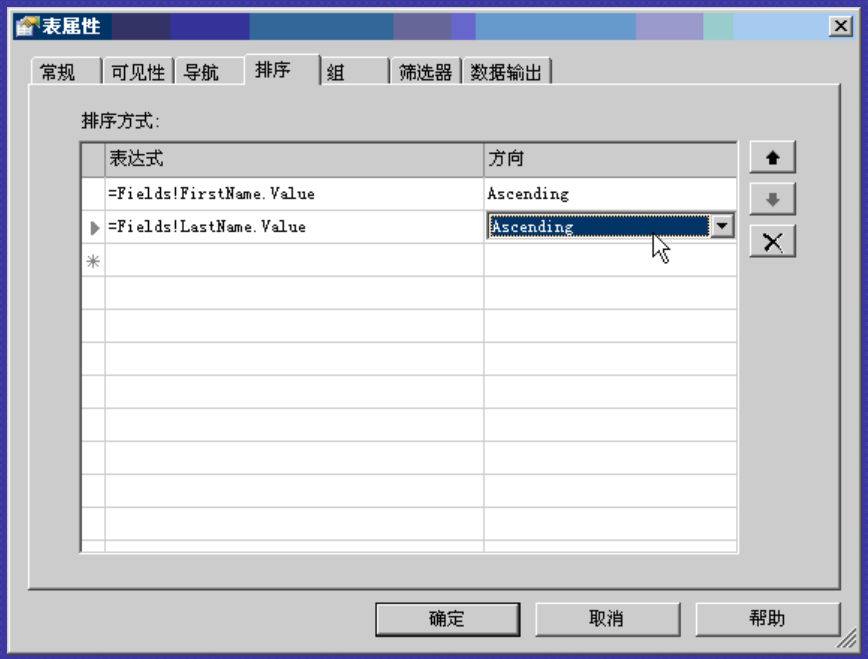


图13-24 表属性对话框

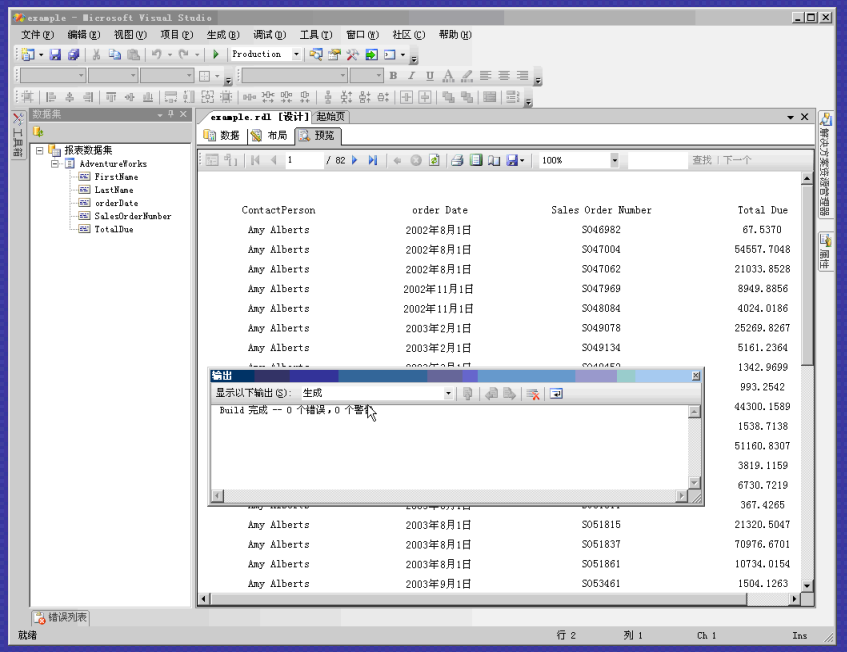


图13-25 生成报表预览界面

13.2.1 通过报表设计器创建报表

1.创建空白报表，然后手动添加查询和布局。

（9）如果预览后确认无误，可以进行报表的发布。如图13-26所示的 example 报表项目属性页对话框，单击上面的“配置管理器...”按钮，打开如图13-27所示的配置管理器对话框。

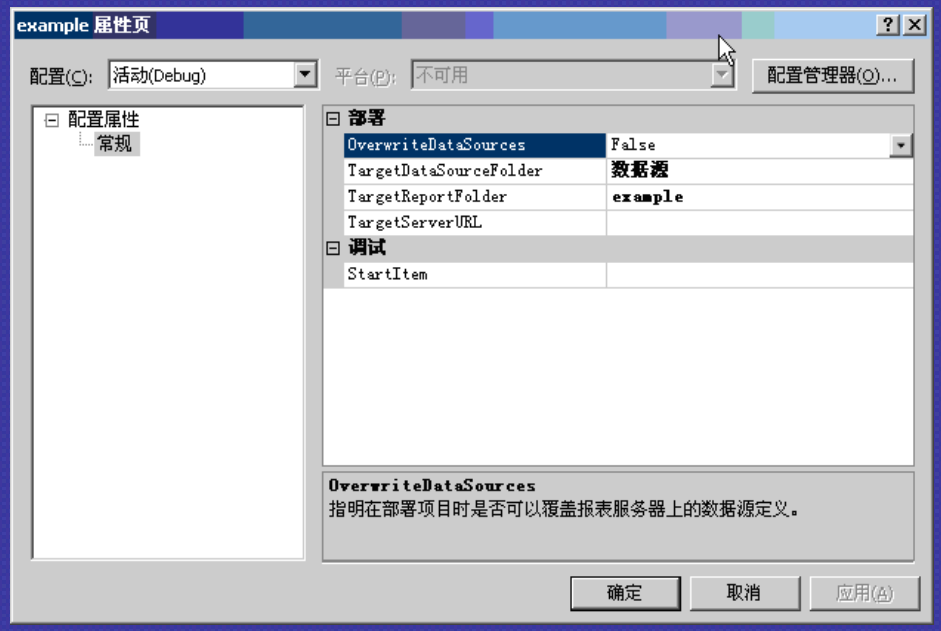


图13-26 example 报表项目属性页对话框

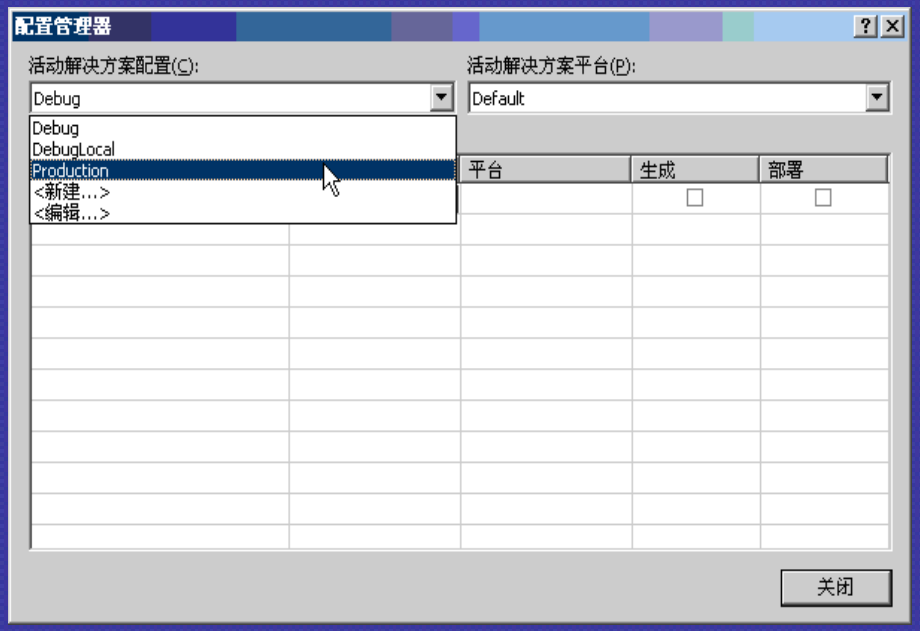


图13-27 配置管理器对话框

13.2.1 通过报表设计器创建报表

1.创建空白报表，然后手动添加查询和布局。

（10）等待发布完成后如果成功将会看到如图13-28所示的输出说明信息并且弹出浏览器显示生成的报表，否则在输出框中会给出提示性的错误信息。

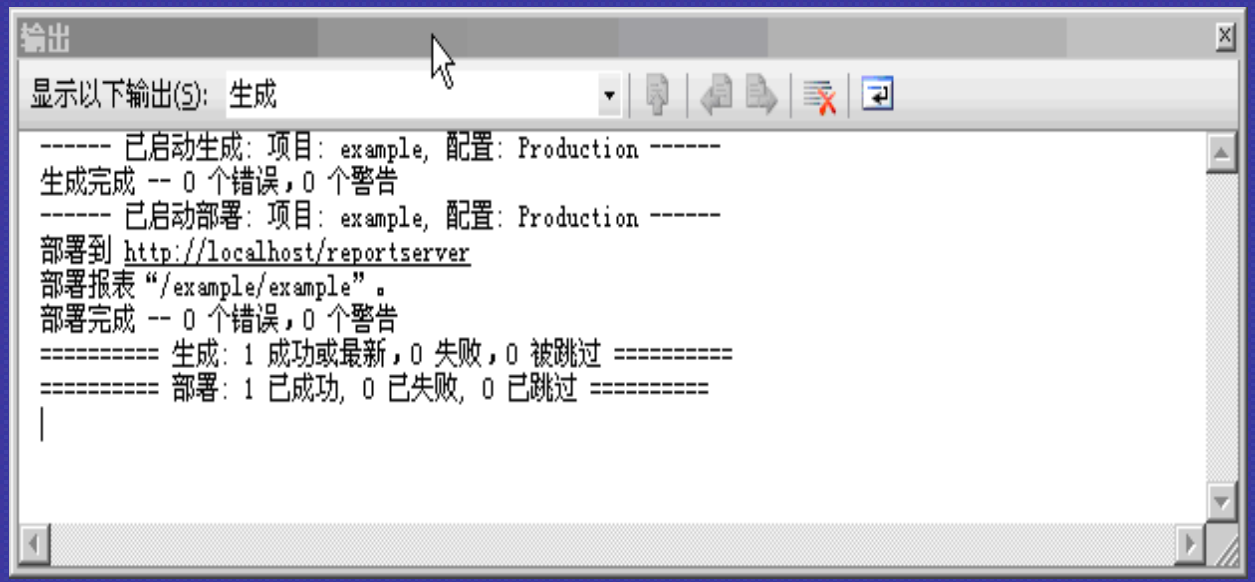


图13-28 发布完成后的输出信息说明

13.2.1 通过报表设计器创建报表

2. 使用报表向导，根据提供的信息自动创建表或矩阵报表

（1）如同前一种方法一样，首先点击“开始→程序”，然后选择“Microsoft SQL Server 2005”，接下来选择“SQL Server Business Intelligence Development Studio”，可以打开Microsoft Visual Studio开发环境。

（2）点击菜单栏上的“文件”，然后选择“新建”，再选择“项目”。可以打开新建项目对话框。在左边的项目类型中选择“商业智能项目”，在右边的“模板”列表中选择“报表服务器项目向导”，然后在下面的名称框中输入报表项目的名称。这里输入 Guidexample 作为新建报表的名称。点击确定后则出现如图13-29所示的报表向导开始对话框

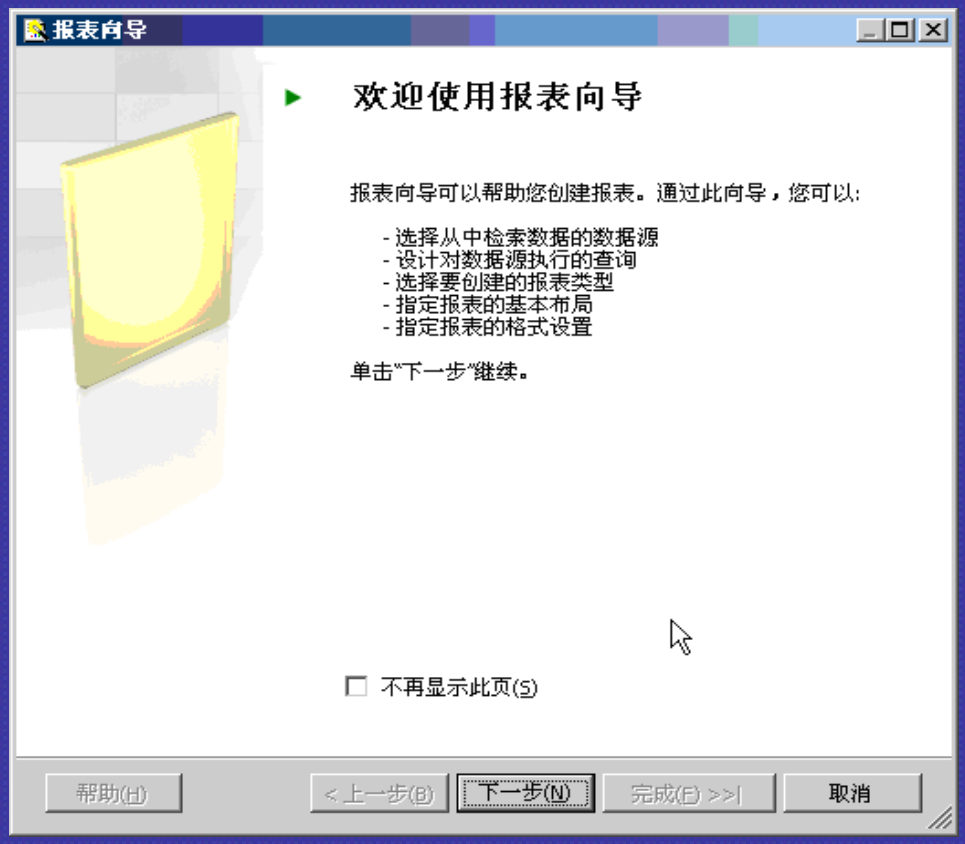


图13-29 报表向导开始对话框

13.2.1 通过报表设计器创建报表

2. 使用报表向导，根据提供的信息自动创建表或矩阵报表

（3）点击下一步则出现如图13-30所示的选择数据源对话框，在名称中可以修改新建数据源的名称。点击下一步，则出现如图13-31所示的设计查询页面。

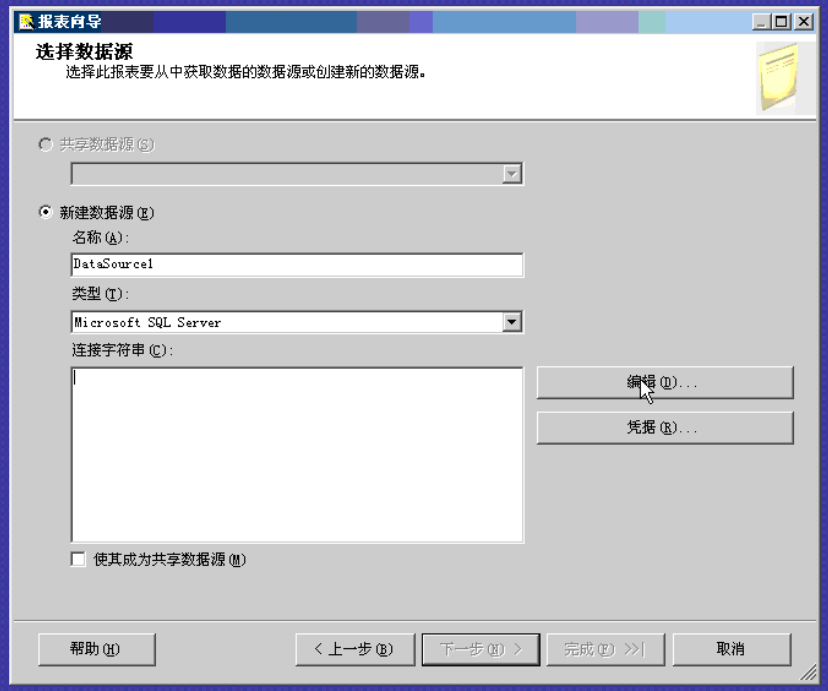


图13-30 选择数据源对话框

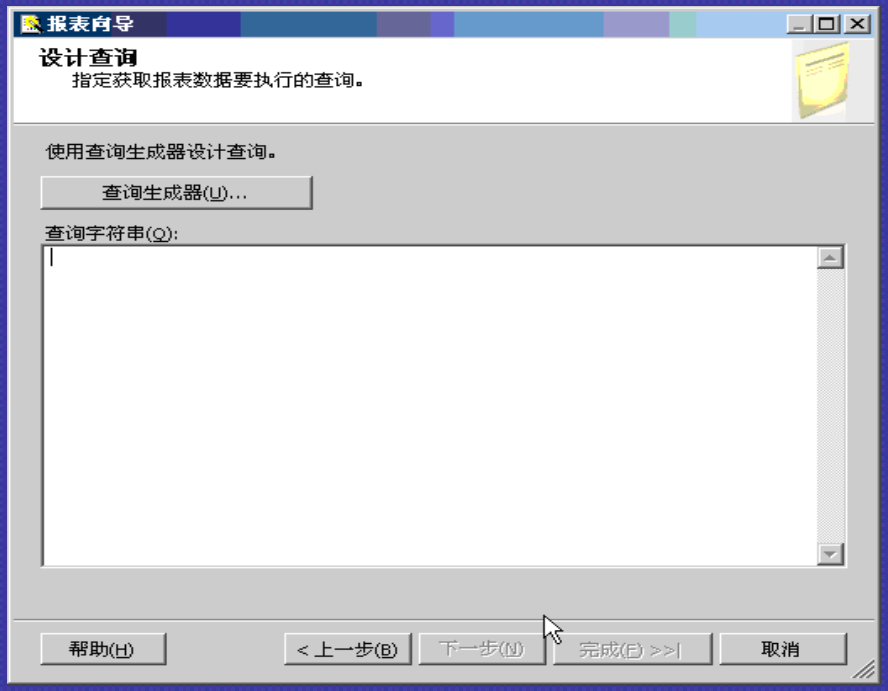


图13-31 设计查询页面

13.2.1 通过报表设计器创建报表

2. 使用报表向导，根据提供的信息自动创建表或矩阵报表

（4）点击查询生成器可以弹出如图13-32所示的查询生成器的界面，在上面输入查询语句，点击下一步，则出现如图13-33所示的选择报表类型对话框。

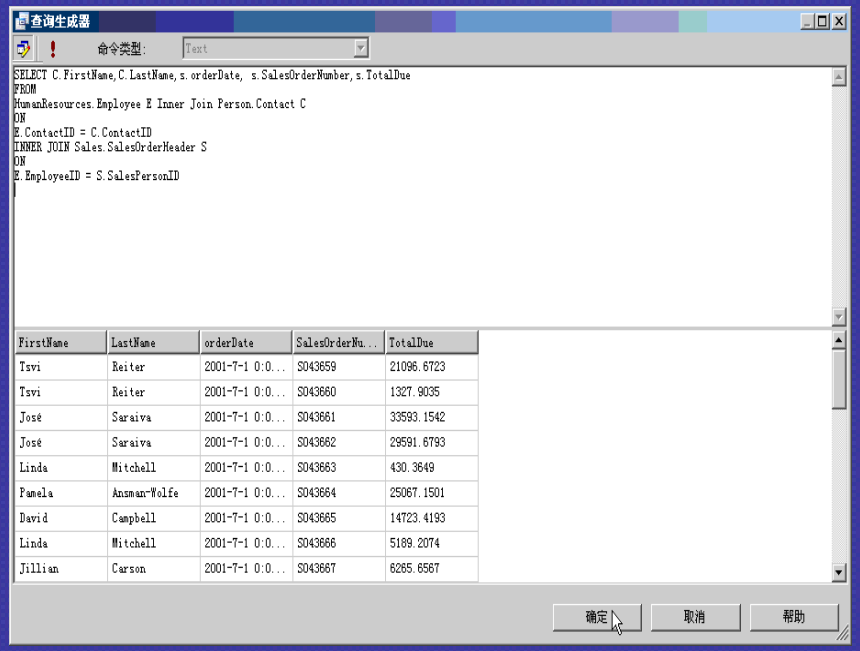


图13-32 查询生成器界面

图13-33 选择报表类型对话框

13.2.1 通过报表设计器创建报表

2. 使用报表向导，根据提供的信息自动创建表或矩阵报表

（5）点击下一步则出现如图13-34所示的设计表对话框，这里可以选择页面上要从上向下逐行进行特殊显示的字段，也可以选择用那些字段进行分组，以及选择显示详细信息的字段。点击下一步则出现如图13-35所示的选择表样式对话框。

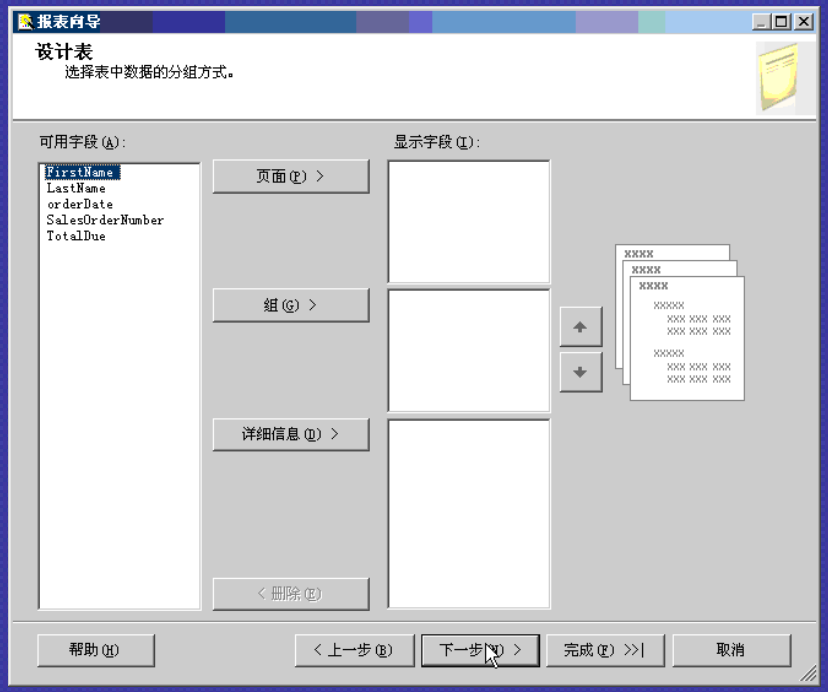


图13-34　设计表对话框

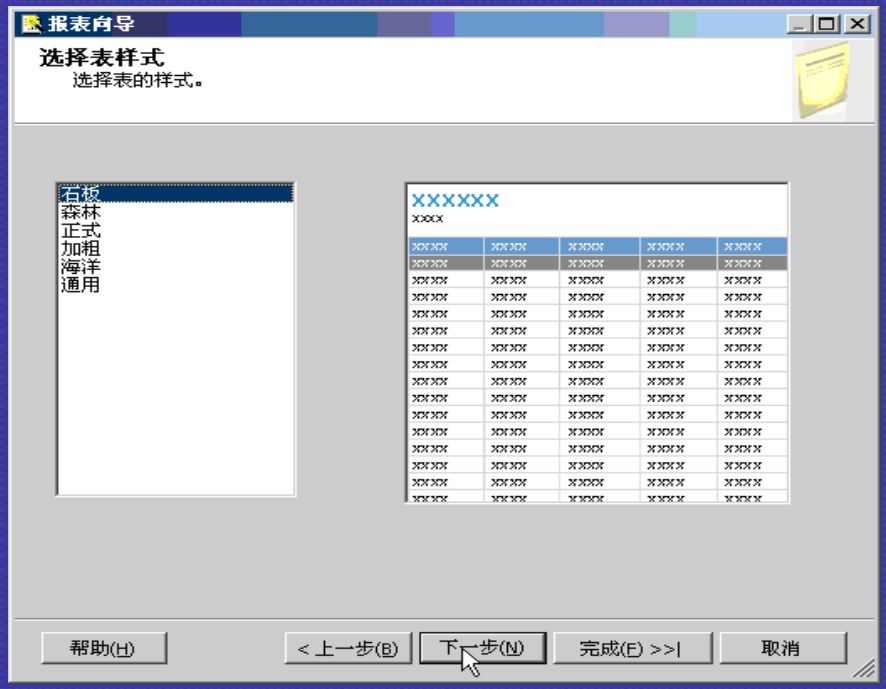


图13-35　选择表样式对话框

13.2.1 通过报表设计器创建报表

2. 使用报表向导，根据提供的信息自动创建表或矩阵报表

（6）点击下一步进入到如图13-36所示的选择部署位置对话框，SQL Server 2005将会根据系统自动做出相应的配置，点击下一步或完成皆可跳到如图13-37所示的最后一页的完成向导页面。

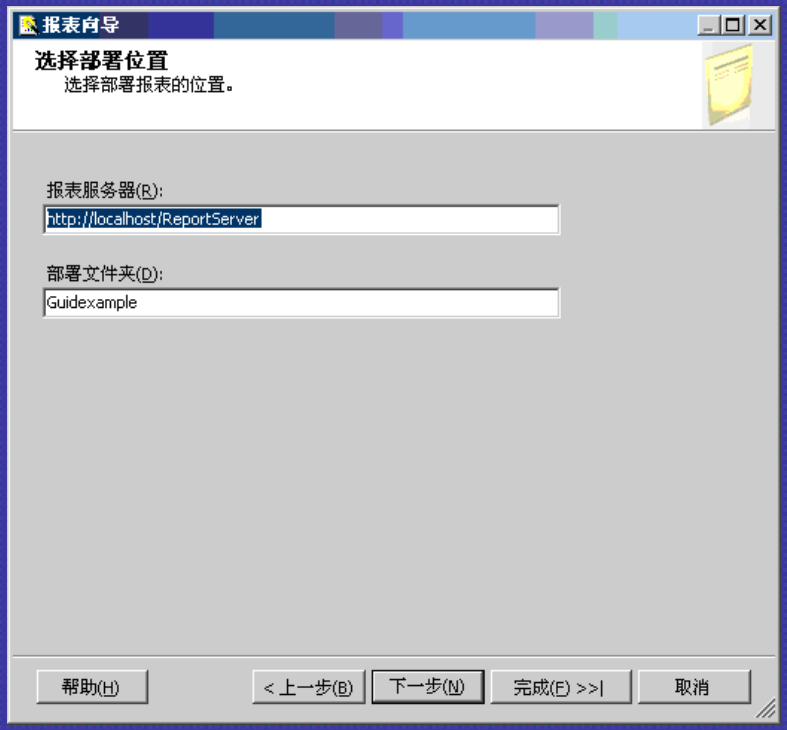


图13-36　选择部署位置对话框

图13-37　完成向导页面

13.2.1 通过报表设计器创建报表

3. 从 Microsoft Access 导入现有的报表

（1）点击“开始→程序”，然后选择“Microsoft SQL Server 2005”，接着选择“SQL Server Business Intelligence Development Studio”，可以打开Microsoft Visual Studio开发环境。然后打开或创建一个项目，以便向其中导入报表。在“项目”菜单上，指向“导入报表”，再单击 Microsoft Access；或者，也可以在解决方案资源管理器中右键单击项目，指向“导入报表”，再单击 Microsoft Access。执行完以上操作后，会弹出熟悉的Windows的打开文件对话框。如图13-38所示。

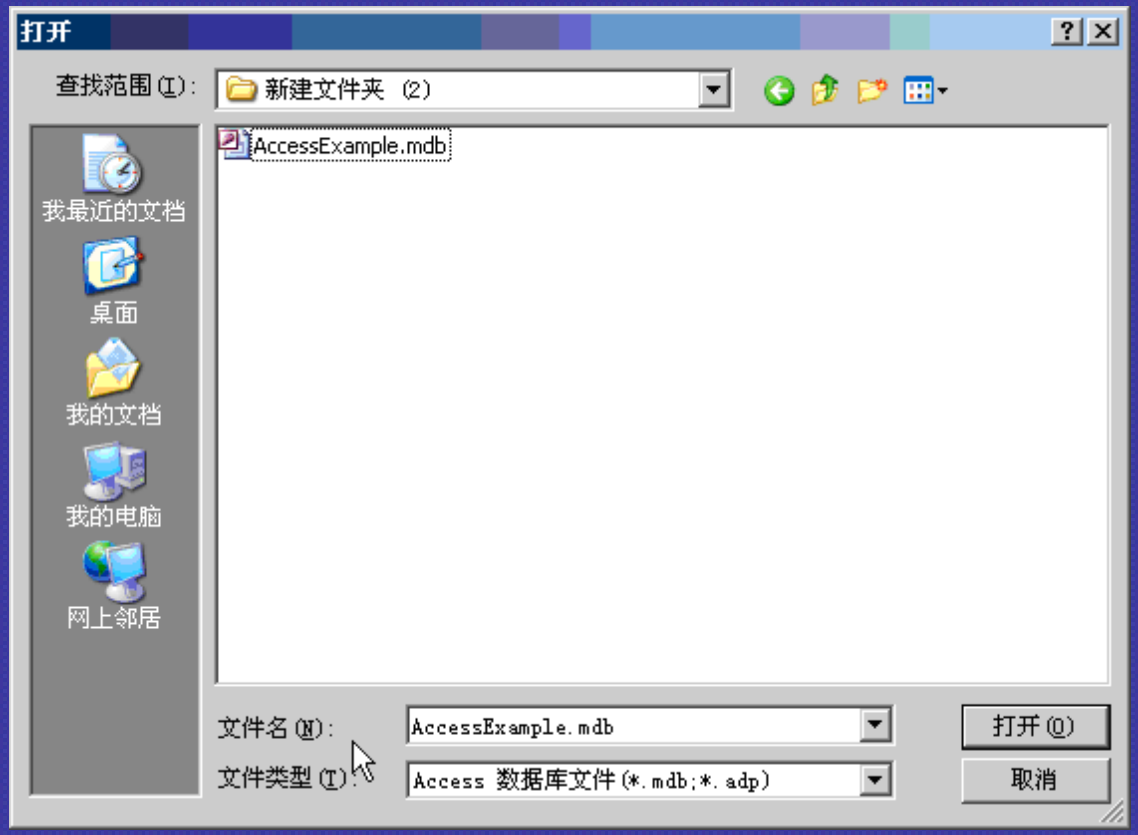


图13-38 导入Microsoft Access报表的选择文件对话框

13.2.1 通过报表设计器创建报表

3. 从 Microsoft Access 导入现有的报表

（2）在“打开”对话框中，选择包含相应报表的 Microsoft Access 数据库(.mdb)或项目(.adp)，再单击“打开”。则可以在如第一种方法介绍的Microsoft Visual Studio开发环境的布局设计页面中查看报表。所不同的是Microsoft Access 中的报表的页面布局是使用“带区”在页上排列报表项，所谓“带区”是指页上垂直排列的区域。这些区域可以包含报表表头、报表表尾、页眉、页脚、组和详细信息。所以在将报表从Access导入到报表设计器时，Access报表的页眉和页脚将转换为Reporting Services报表的页眉和页脚。组和详细信息将转换到列表数据区域中。报表表头和表尾将置于表体中，而不是位于单独的带区中。这可能会导致项的位置与其在Access报表中的位置稍有不同。然后可以对报表进行修改、调整、美化等设计，调整好之后可以对报表用第一种的发布方法进行发布。

13.2.1 通过报表设计器创建报表

3. 从 Microsoft Access 导入现有的报表

（3）如果在导入过程中遇到不支持的项（如模块和某些控件）时，这些项将作为生成错误显示在“任务列表”窗口中，若要查看“任务列表”窗口，可选择“视图”菜单，然后选择 “任务列表”即可。一般说来，除了某些模块和控件以及其它字符会出现问题外，较多的失败问题会出现在数据源的发布和凭证的认证上。如果是从 Access 项目 (.adp) 文件导入报表，则数据源的连接字符串将从 .adp 文件的连接字符串中获取。如果是从 Access 数据库 (.mdb) 文件导入报表，则连接字符串可能会指向 Access 数据库，并且在导入报表后可能需要更正该字符串，由于Access数据库需要用OLE DB相连接，其连接字符串的形式形如“Provider=Microsoft.Jet.OLEDB.4.0;Data Source=”数据库文件所在位置“”，该字符串可以在导入后的数据选项卡中点击数据集旁边的浏览按钮，在弹出的对话框中的查询选项页中点击数据源文本框旁边的按钮进行查看和修改。如果遇到发布报表后出现“执行此报表所需的用户数据源凭据未存储在报表服务器数据库中”的错误字样，则可以打开浏览器，转入到http://localhost/reports报表管理器的界面中，选择生成报表的文件夹，单击选择生成的报表，选择属性页面，点击在属性页面的左侧数据源的选项卡，在连接方式中选择相应的数据源即可，一般选择windows集成安全性选项即可解决。

13.2.2 通过报表生成器生成报表

- 报表生成器是一种ClickOnce Windows窗体应用程序，该程序由用户从报表服务器下载到本地计算机来生成即时报表。然后用户通过将字段从预定义的报表模型拖到预设计的报表布局模板上来创建报表，此外还可以编辑或定义公式。
- 通过报表生成器生成报表首先需要有预定义的报表模型，这需要从网络上下载或者自己在报表设计器中设计报表模板模型。在Web浏览器的地址栏中，键入13.1节中设置的报表服务器URL地址，默认的地址是http://localhost/reports (localhost表示服务器是本机，如果不是本机则需输入服务器名称)。这时会出现如图13-39所示的报表管理器的浏览页面。点击报表生成器即可出现报表生成器（Microsoft Report Bulider）窗口。或者在浏览器中直接键入地址：
- http://localhost/reportserver/reportbuilder/reportbuilder.application也可启动报表生成器窗口(localhost意义同上)。

13.2.2 通过报表生成器生成报表

1.创建报表模型

（1）通过报表管理器上传报表模板

上传文件的方法是打开图13-39所示的报表管理器页面，可以新建文件夹或者使用已有的文件夹用来管理模板文件，点击进入想要将模板上传进入的文件夹，然后点击上载文件则会转到如图13-40所示的上载文件页面。

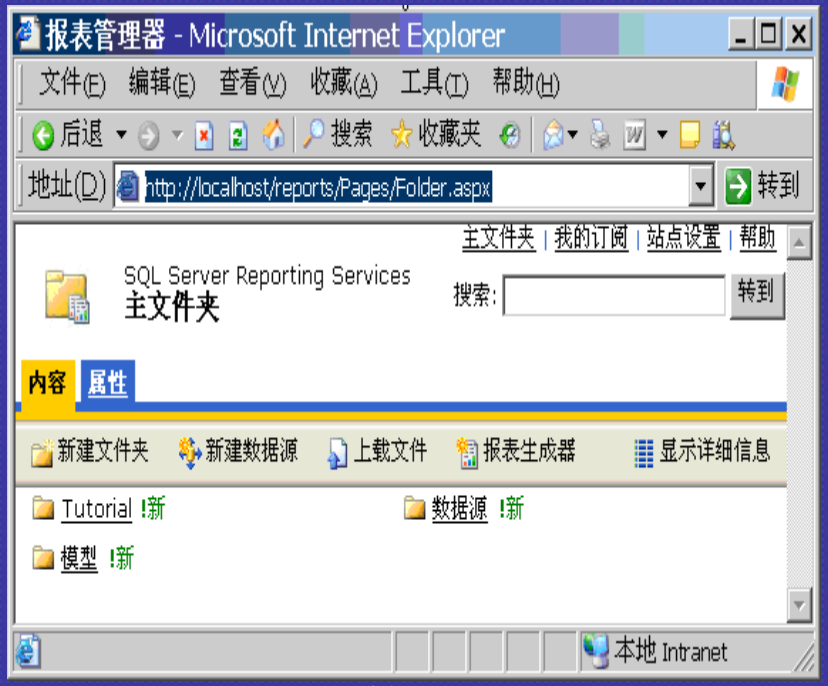


图13-39　报表管理器的浏览页面

图13-40　上载文件页面

13.2.2 通过报表生成器生成报表

1.创建报表模型

（2）通过报表设计器制作所需要的报表模板

如果没有现成的报表模板，或者是想制作所需要的报表模板，可以用报表设计器进行制作。点击“解决方案资源管理器”，则会出现如图13-41所示的页面。右键点击数据源，选择添加新数据源，则会弹出如图13-42所示的数据源向导对话框。

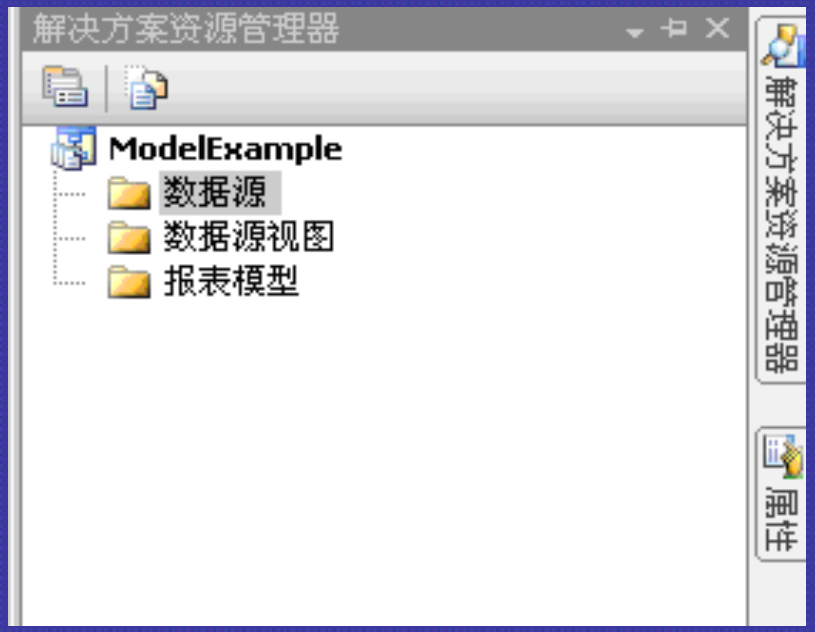


图13-41　解决方案资源管理器页面

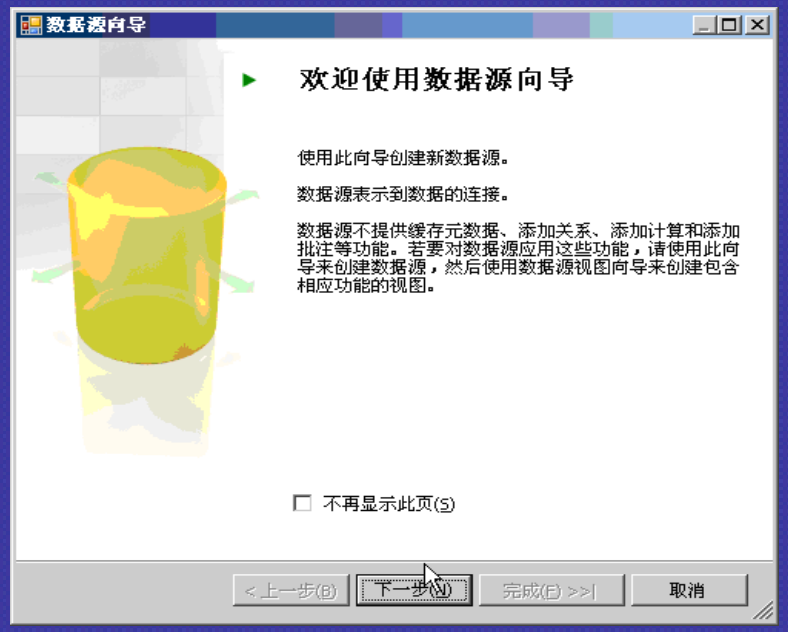


图13-42　数据源向导对话框

13.2.2 通过报表生成器生成报表

1.创建报表模型

（2）通过报表设计器制作所需要的报表模板

点击下一步进入到如图13-43所示的选择如何定义连接对话框，在这里可以看到已有的可用连接。建立好数据源后点击下一步进入到数据源向导的最后一个页面，即完成向导页面。如图13-44所示。

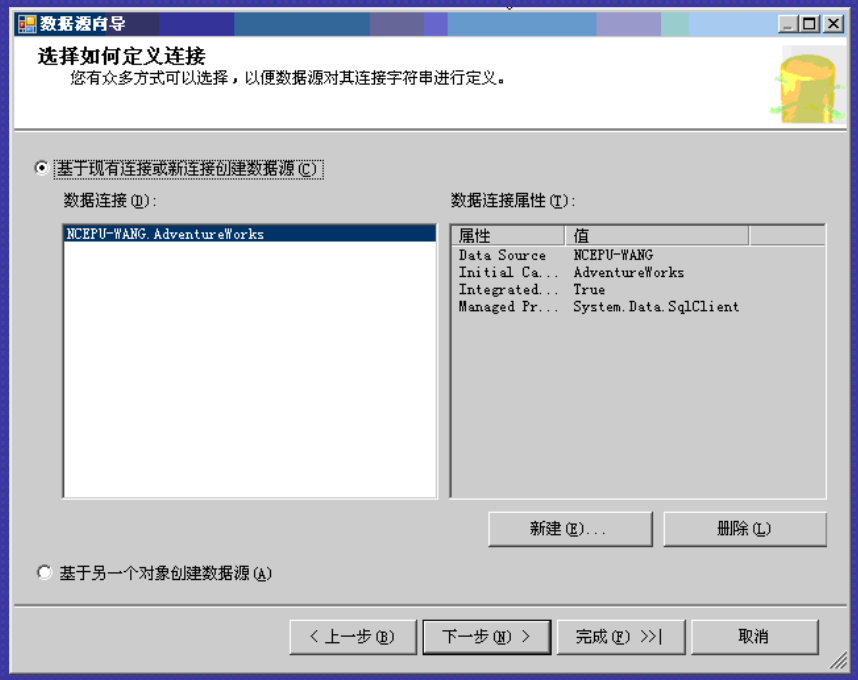


图13-43　选择如何定义连接对话框

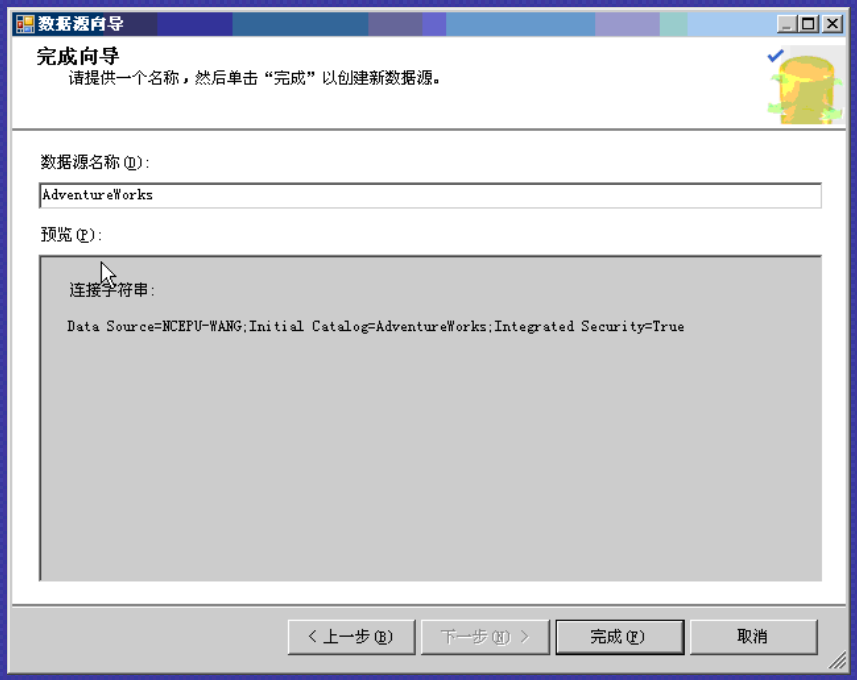


图13-44　完成向导对话框

13.2.2 通过报表生成器生成报表

1.创建报表模型

（2）通过报表设计器制作所需要的报表模板

接下来要添加数据源视图，点击图13-41右边“解决方案资源管理器”。右键点击数据源视图，选择添加新数据源视图，则会弹出如图13-45所示的数据源视图向导对话框。点击下一步，进入到如图13-46所示的选择数据源对话框。

图13-45　数据源视图向导对话框

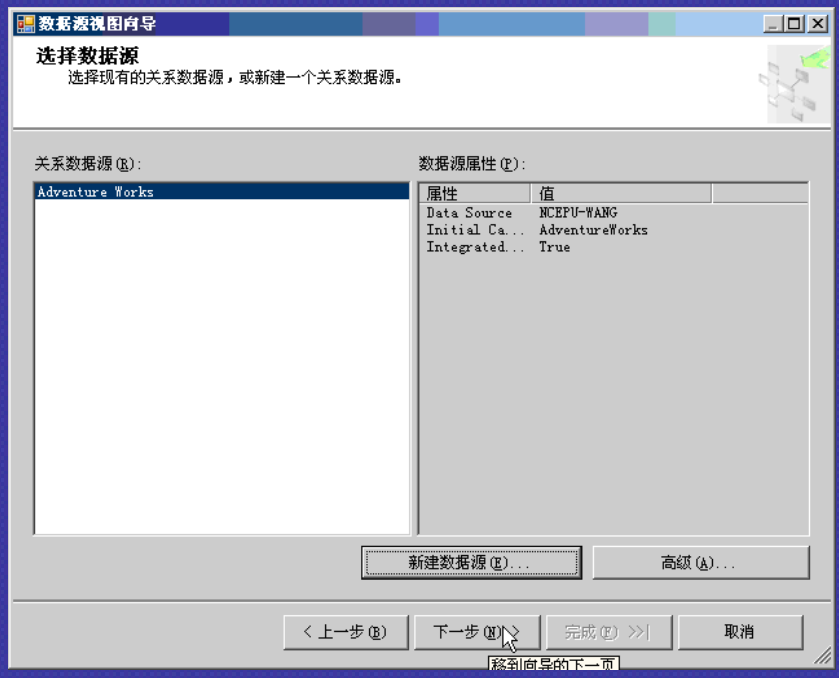


图13-46　选择数据源对话框

13.2.2 通过报表生成器生成报表

1.创建报表模型

（2）通过报表设计器制作所需要的报表模板

点击下一步，进入到如图13-47所示的选择表和视图对话框，在左边的可用对象中列出了数据源中所有可用的表和视图，示例中选择HumanResources.Employee表，然后点击下一步，进入到如图13-48所示的完成向导对话框。

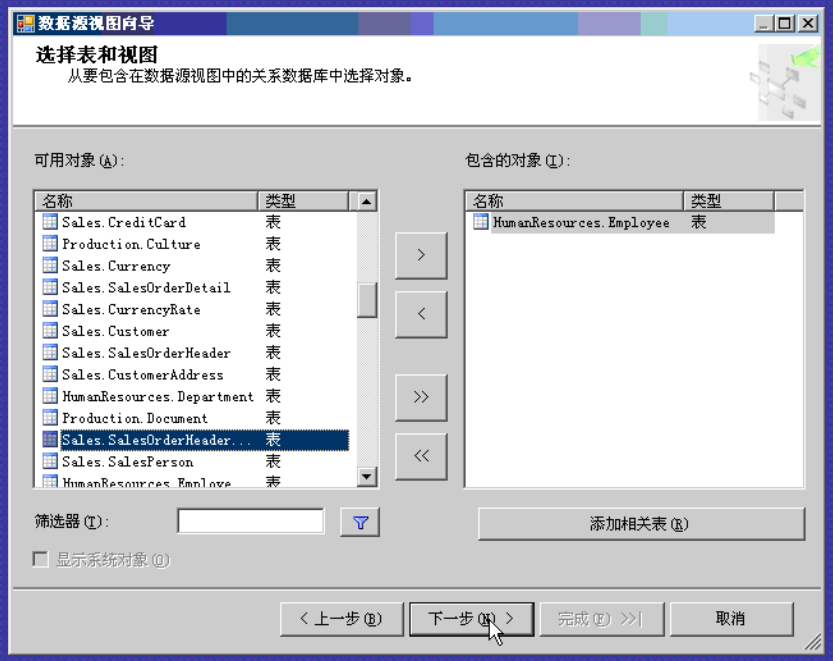


图13-47　选择表和视图对话框

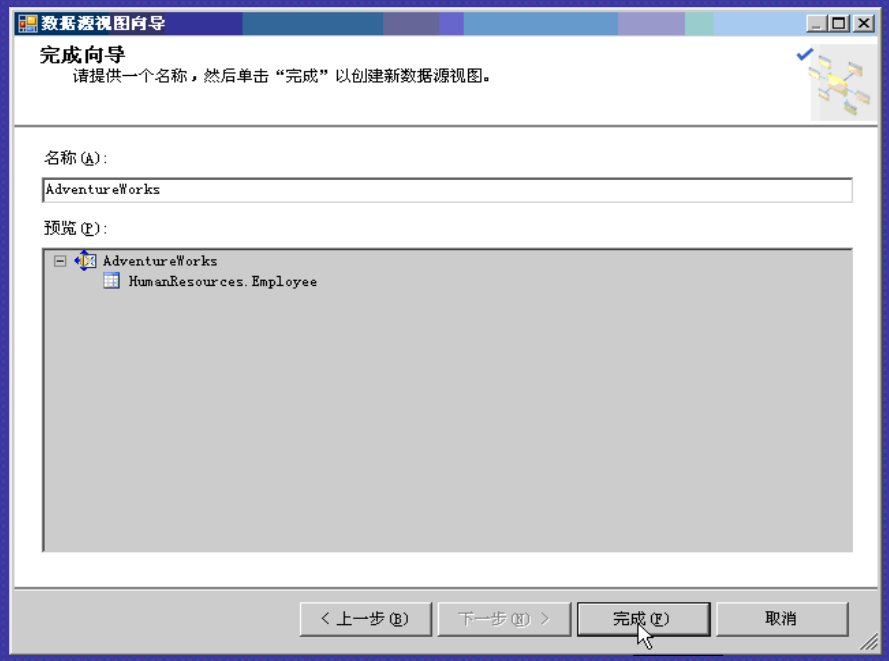


图13-48　完成向导对话框

13.2.2 通过报表生成器生成报表

1.创建报表模型

（2）通过报表设计器制作所需要的报表模板

接下来要制作报表模型，点击图13-41右边“解决方案资源管理器”，右键点击报表模型，选择添加报表模型，则会弹出如图13-49所示的报表模型向导对话框。点击下一步，进入到如图13-50所示的选择数据源视图对话框。

图13-49　报表模型向导对话框

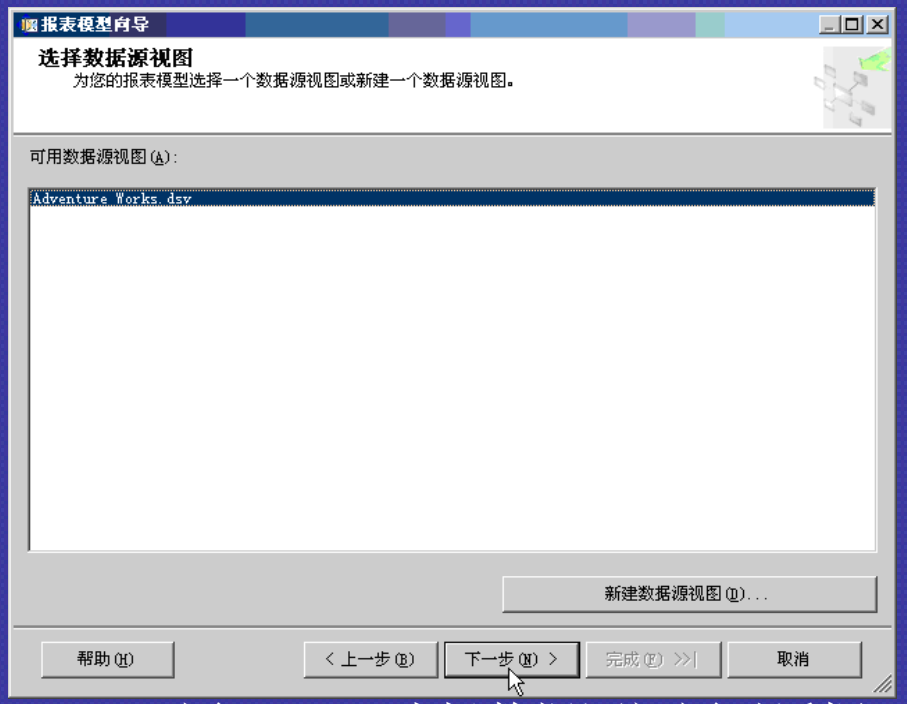


图13-50　选择数据源视图对话框

13.2.2 通过报表生成器生成报表

1.创建报表模型

（2）通过报表设计器制作所需要的报表模板

点击下一步，进入到如图13-51所示的选择报表模型生成规则对话框。点击下一步，进入到如图13-52收集模型统计信息对话框。

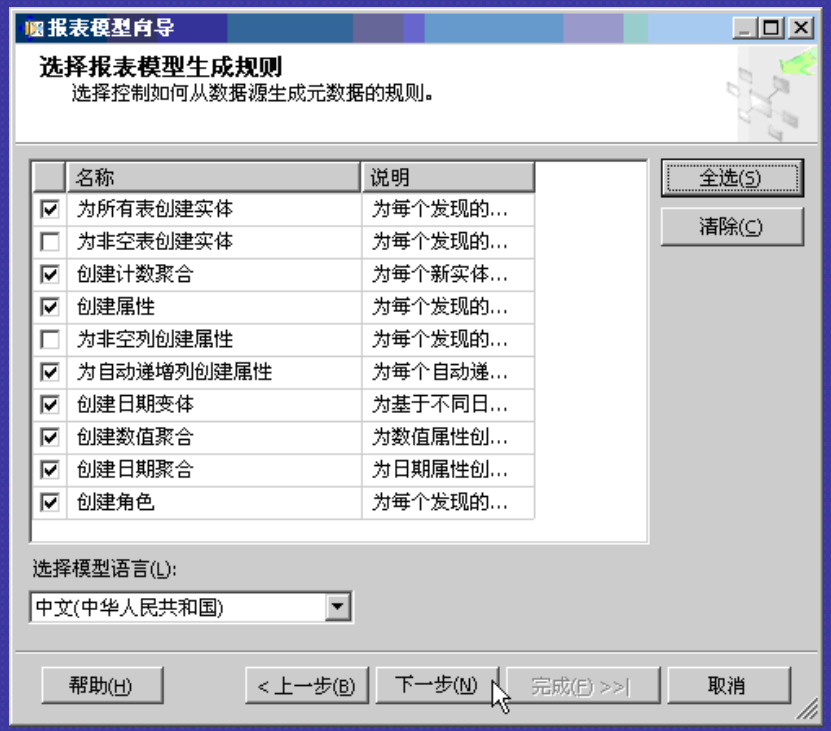


图13-51　选择报表模型生成规则对话框

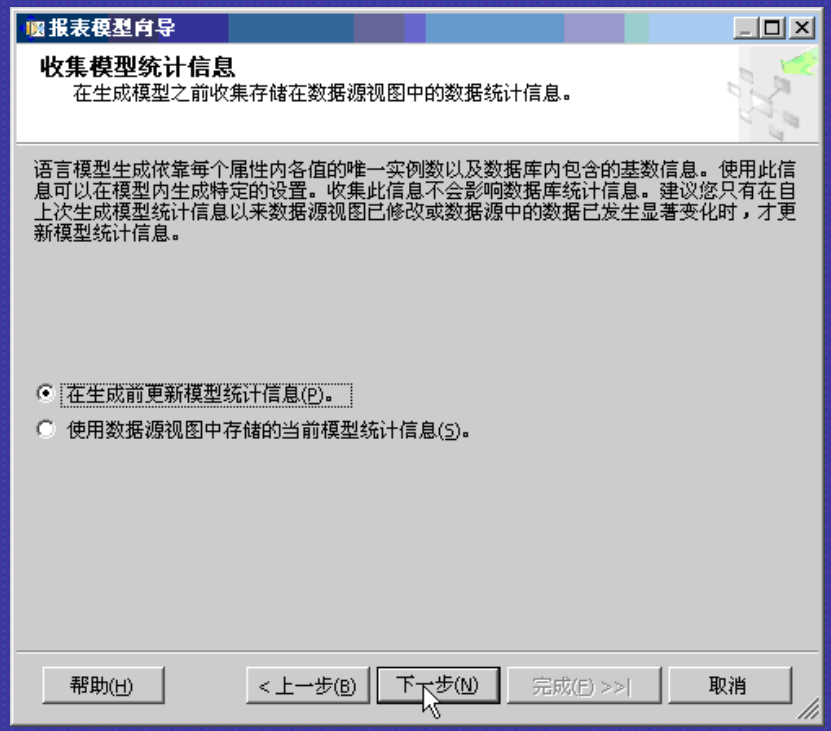


图13-52　收集模型统计信息

13.2.2 通过报表生成器生成报表

1.创建报表模型

（2）通过报表设计器制作所需要的报表模板

点击下一步，进入到如图13-53所示的完成向导对话框，在名称文本框中输入报表模型名称后点击运行即可对图13-51中所选的规则进行生成规则操作，规则全部生成后单击完成按钮可以完成报表模型的制作。

图13-53　完成向导对话框

13.2.2 通过报表生成器生成报表

2. 通过报表生成器创建报表

按照本节开头介绍的方法打开如图13-39所示的报表管理器的浏览页面，点击报表生成器进入到如图13-54所示的报表生成器设计界面，可以在此界面中点击文件菜单选择打开相应的报表模板来进行创建报表的工作。刚才建立的Adventure Works项目就显示在右边的数据源下，双击Adventure Works，报表生成器会显示出如图13-55所示的报表生成器设计界面。

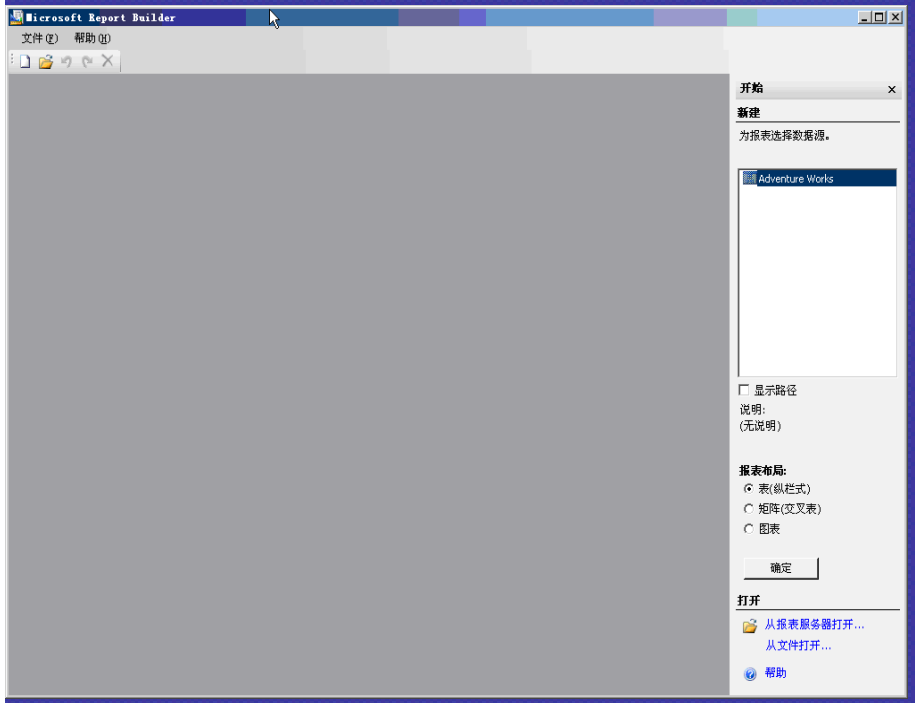


图13-54 报表生成器界面

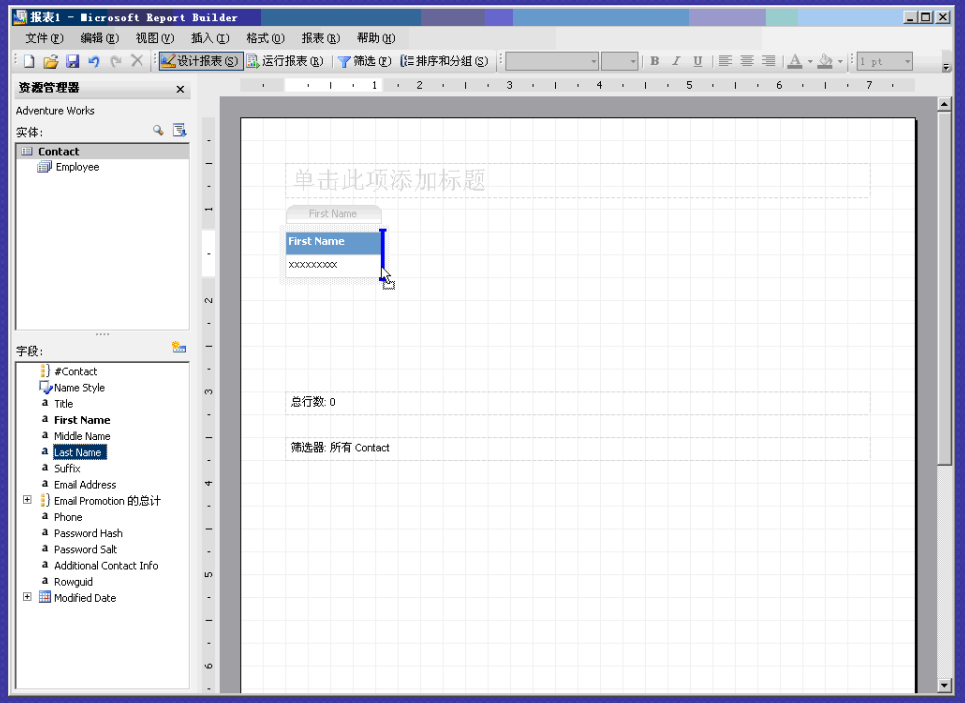


图13-55 报表生成器设计界面

SQL2005 恢复数据库

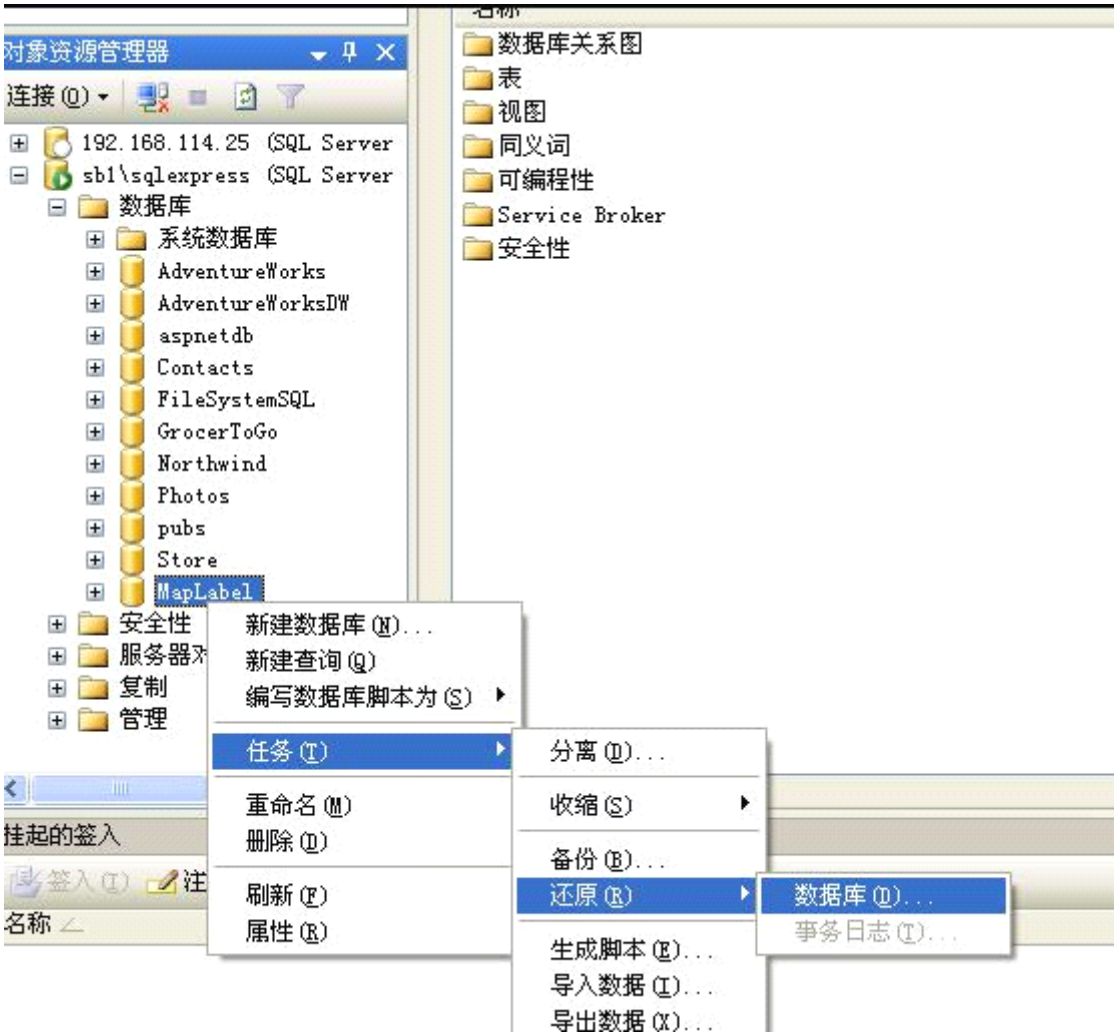
[code:tsql]
RESTORE DATABASE Enterprise
   FROM DISK = 'D:\Program Files\Microsoft SQL Server\MSSQL.4\MSSQL\Backup\Enterprise-20080403.bak'
   WITH MOVE 'Enterprise' TO 'D:\Program Files\Microsoft SQL Server\MSSQL.4\MSSQL\Data\Enterprise.mdf',
   MOVE 'Enterprise_log' TO 'D:\Program Files\Microsoft SQL Server\MSSQL.4\MSSQL\Data\Enterprise_log.ldf',
STATS = 10, REPLACE
GO
[/code]

10 percent processed.
20 percent processed.
30 percent processed.
40 percent processed.
50 percent processed.
60 percent processed.
70 percent processed.
80 percent processed.
90 percent processed.
100 percent processed.
Processed 8760 pages for database 'Enterprise', file 'Enterprise' on file 1.
Processed 2 pages for database 'Enterprise', file 'Enterprise_log' on file 1.
RESTORE DATABASE successfully processed 8762 pages in 4.801 seconds (14.949 MB/sec).

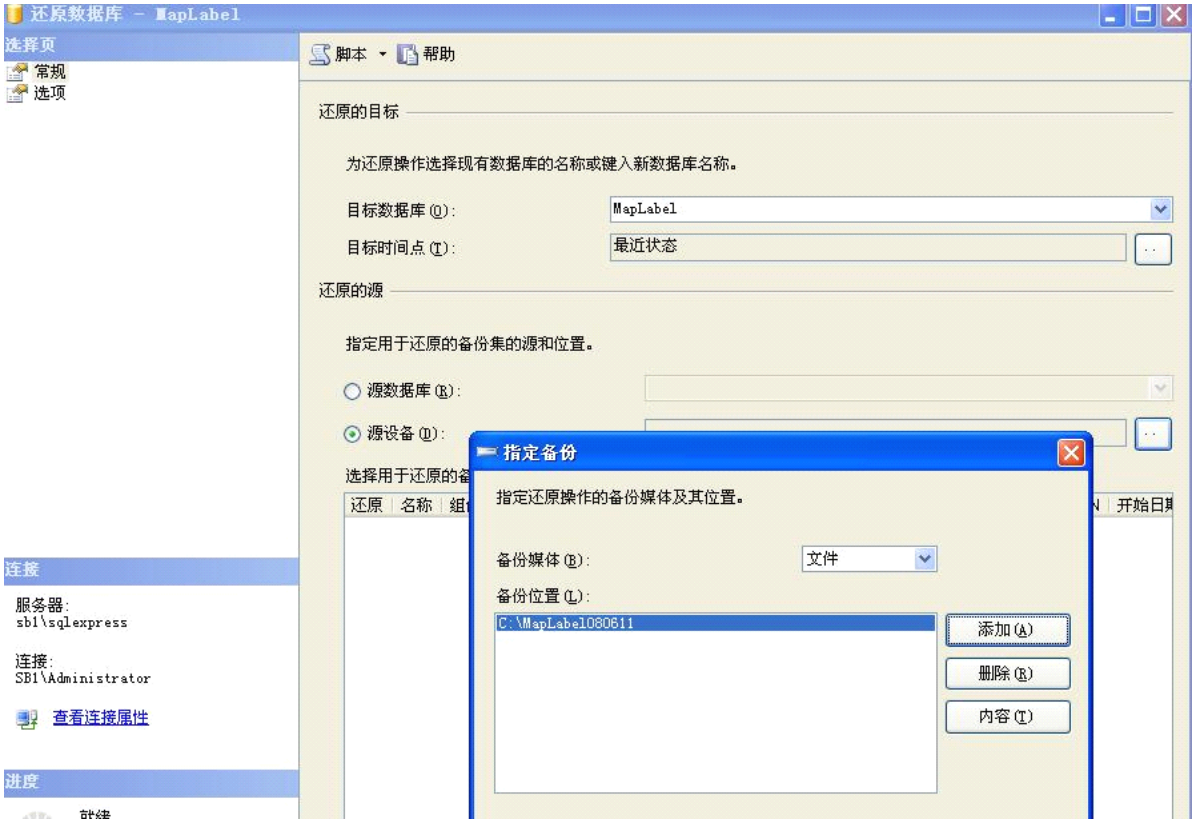
SQL Server2005 还原数据库详细

对于在 SQL Server2000 中的还原数据库，很多朋友都是使用过的，一起来也很简单，选择文件后，选择强制还原，问题即可解决，然而在 2005 中却不行了，原因是：2005 中数据库的备份中记录了备份数据库的地址，在你还原的过程中，你必须将此地址换成你电脑上要还原的数据库的地址。详细过程如下：

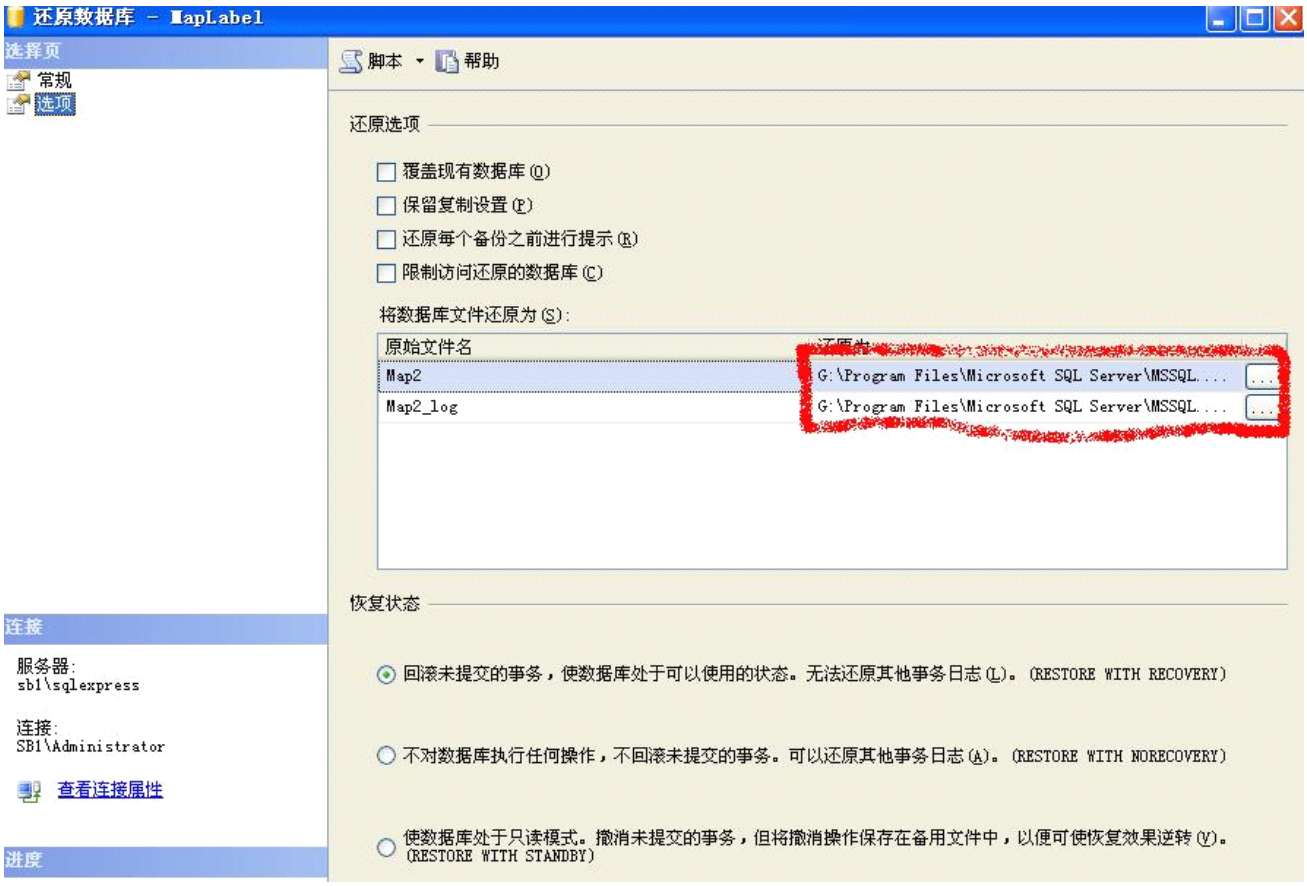
第一步：选择要还原的数据库



第二步：选择备份文件。选择"源设备"，点击文本框右边的按钮"..."，在弹出的对话框中选择数据库的备份.注意：文件格式要选择"所有文件"。



第三步：选择图片左上角的“常规”，出席下面的对话框，红线包围处的路径是备份文件数据库的位置，必须换成你的数据库所在的位置。



第四步： 更改了数据库的位置后，就可以更新了！

点击图片可以放大

呵呵，很简单吧，可我就是因为不知道要做第三步，结果从昨天中午整到了今天，都快崩溃了，希望朋友们不在在这里花太多的时间，将时间用在更有意义的事上不是更好吗....

```
//C连接SQL SERVER

#include <stdio.h>

#include <stdlib.h>

void main()

{

  CString SQL;
     ::CoInitialize(NULL);
     _RecordsetPtr PRs("ADODB.Recordset");
     _ConnectionPtr PConn("ADODB.Connection");
     _variant_t RecordsAffected;

     try
  {
   PConn.CreateInstance("ADODB.Connection");
      PRs.CreateInstance(__uuidof(Recordset));

SQL.Format("provider=SQLOLEDB;server=(%s);database=s;Uid=s;pwd=s;",ServerName,DbName,UserID,Pass
word);
PConn->Open((_bstr_t)SQL,"","",adModeUnknown);//这里用到了变量来代替连接参数,根据自己改动.
  if(PConn==NULL)
   {
       printf("Connect Sqlserver error!\n");
   }
         else
```

```
    {
        printf("Connect Sqlserver success!\n");

}

        PConn->Close;
        PRs->Close;

}

      catch(_com_error e)  //异常
   {
        printf("Connect Error: %s\n",(char *)e.ErrorMessage());
   }

}
```