

第 14 章 SQL Server 数据库的保护

SQL Server 数据库的保护分为三个方面介绍：14.1 节介绍 SQL Server 数据库的安全性机制；为了防止因软硬件障而导致的数据丢失或数据库的崩溃，14.2 节介绍了 SQL Server 的备份和恢复策略；SQL Server 支持并发操作，其并发操作的机制将在 14.3 节中介绍。

本章要点：

- * SQL Server 的安全机制
- * 登录和用户
- * 权限、角色管理
- * 备份和恢复的种类及方法
- * SQL Server 的并发机制

14.1 SQL Server 的安全性

对于一个数据库而言，安全性是指保护数据库不被破坏、偷窃和非法使用的性能。一个设计良好的安全模式能使用户的合法操作很容易，同时使非法操作和意外破坏很难或不可能发生。数据库的安全性和计算机系统的安全性（包括操作系统、网络系统的安全性）是紧密联系、相互支持的。

14.1.1 SQL Server 的安全机制

SQL Server 的安全管理机制包括验证（authentication）和授权（authorization）两种类型。验证是指检验用户的身份标识；授权是指允许用户做些什么。验证过程在用户登录操作系统和 SQL Server 的时候出现，授权过程在用户试图访问数据或执行命令的时候出现。SQL Server 的安全机制分为四级，其中第一层和第二层属于验证过程，第三层和第四层属于授权过程，如图 14-1 所示。

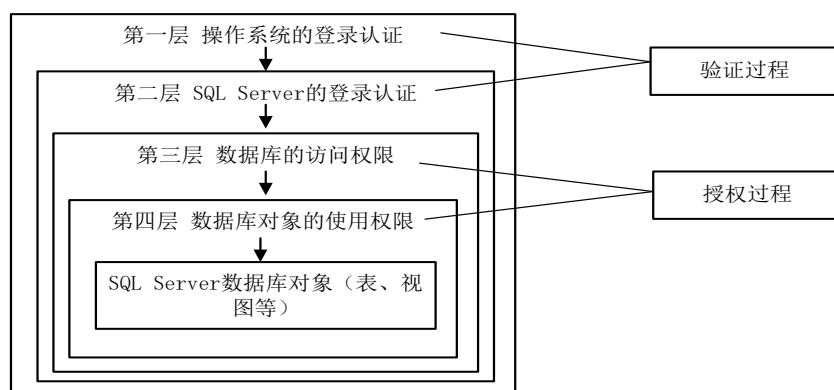


图 14-1 SQL Server 的四级安全机制

用户首先必须登录到操作系统，如果用户要访问数据库中的数据，还需要登录到 SQL Server。用户进入数据库系统通常是通过数据库应用程序实现的，这就需要用户向应用程序提供其身份，由应用程序将用户的身份递交给 DBMS 进行验证，只有合法的用户才能进入下一步操作。合法用户进行数据库操作时，DBMS 还有验证该用户是否具有这种操作权限，

如果有对应的操作权限，才可以进行操作，否则拒绝执行用户的操作。

因此，下面主要介绍服务器、数据库和数据库对象的安全性问题。

14.1.2 管理 SQL Server 服务器安全性

要想保证数据库数据的安全，必须搭建一个相对安全的运行环境，因此对服务器安全管理至关重要。在 SQL Server 中，对服务器安全管理主要通过身份验证模式来实现。身份验证是指当用户访问系统时，系统对该用户账户和口令的确认过程，内容包括用户账户是否有效、能否访问系统以及能访问系统的哪些数据等。

1. 身份验证模式

SQL Server 提供了 Windows 身份和混合身份两种验证模式。无论哪种模式，SQL Server 都需要对用户的访问进行两个阶段的检验：验证阶段和许可确认阶段。

(1) 验证阶段。

用户在 SQL Server 获得对任何数据库的访问权限之前，必须登录到 SQL Server 上，并且被认为是合法的。SQL Server 或者 Windows 要求对用户进行验证，如果验证通过，用户就可以连接到 SQL Server 上；否则，服务器将拒绝用户登录。

(2) 许可确认阶段。

用户验证通过后会登录到 SQL Server 上，此时系统将检查用户是否有访问服务器上数据的权限。

Windows 身份验证模式（Windows Authentication）会启用 Windows 身份验证并禁用 SQL Server 身份验证。混合模式（Mixed Authentication）会同时启用 Windows 身份验证和 SQL Server 身份验证。

1) Windows 身份验证

使用 Windows 身份验证模式是默认的身份验证模式。SQL Server 服务器如果选择采用 Windows 身份验证，就表明服务器将客户机的身份验证任务完全交给了 Windows 操作系统。

当用户通过 Windows 用户账户连接时，SQL Server 使用操作系统中的 Windows 主体标记验证账户名和密码，即用户身份由 Windows 进行确认。SQL Server 不要求提供密码，也不执行身份验证。Windows 身份验证使用 Kerberos 安全协议，提供有关强密码复杂性验证的密码策略强制，还提供账户锁定支持，并且支持密码过期。通过 Windows 身份验证完成的连接有时也称为可信连接，这是因为 SQL Server 信任由 Windows 提供的凭据。

(1) SQL Server 系统处理 Windows 身份验证方式中的登录账号的步骤如下：

① 当用户连接到 Windows 系统上时，客户机打开一个到 SQL Server 系统的委托连接。该委托连接将 Windows 的组和用户账号传送到 SQL Server 系统中。因为客户机打开了一个委托连接，所以 SQL Server 系统知道 Windows 已经确认该用户有效。

② 如果 SQL Server 系统在系统视图 sys.syslogins 中找到该用户的 Windows 用户账号或组账号，就接收这次身份验证连接。这时，SQL Server 系统不需要重新验证口令是否有效，因为 Windows 已经验证用户的口令是有效的。该用户可以是 Windows 用户账号，也可以是 Windows 组账号。当然，这些账号都已定义为 SQL Server 系统登录账号。

③ 如果多个 SQL Server 计算机在同一个域或一组信任域中，那么登录到单个网络域上就可以访问全部的 SQL Server 计算机。

(2) Windows 身份验证模式的优点如下：

① 数据库管理员的工作可以集中在管理数据库上面，而不是管理用户账户，对用户账户的管理可以交给 Windows 去完成。

② Windows 有更强的用户账户管理工具，可以设置账户锁定、密码期限等。

③ Windows 的组策略支持多个用户同时被授权访问 SQL Server。

2) 混合模式

使用混合安全的身份验证模式，可以同时使用 Windows 身份验证和 SQL Server 登录。使用该模式，SQL Server 首先确定用户的连接是否使用有效的 SQL Server 用户账户登录。如果用户有有效的登录和使用正确的密码，则接受用户的连接。仅当用户没有有效的登录时，SQL Server 才检查 Windows 账户的信息。在这种情况下，SQL Server 将会确定 Windows 账户是否有连接到服务器的权限。如果账户有权限，连接被接受；否则，连接被拒绝。

使用混合模式中的 SQL Server 身份验证时，系统管理员创建一个登录账号和口令，并把它们存储在 SQL Server 中。当用户连接到 SQL Server 时，必须提供 SQL Server 登录账号和口令。

(1) 混合模式身份验证的缺点如下：

- ① SQL Server 身份验证无法使用 Kerberos 安全协议。
- ② SQL Server 登录名不能使用 Windows 提供的其他密码策略。

(2) 混合模式身份验证的优点如下：

① 允许 SQL Server 支持那些需要进行 SQL Server 身份验证的旧版应用程序和由第三方提供的应用程序。

② 允许 SQL Server 支持具有混合操作系统的环境，在这种环境中并不是所有用户均由 Windows 域进行验证。

③ 允许用户从未知的或不可信的域进行连接。例如，既定客户使用指定的 SQL Server 登录名进行连接以接收其订单状态的应用程序。

④ 允许 SQL Server 支持基于 Web 的应用程序，在这些应用程序中用户可创建自己的标识。

⑤ 允许软件开发人员通过使用基于已知的预设 SQL Server 登录名的复杂权限层次结构来分发应用程序。

说明：

- 如果在安装过程中选择混合模式身份验证，则必须为名为 **sa** 的内置 SQL Server 系统管理员账户提供一个密码并确认该密码。**sa** 帐户通过使用 SQL Server 身份验证进行连接；
- 如果在安装过程中选择 Windows 身份验证，则安装程序会为 SQL Server 身份验证创建 **sa** 账户，但会禁用该账户。

3) 配置身份验证模式

在第一次安装 SQL Server 或使用 SQL Server 连接其他服务器时，需要指定验证模式。对于已指定验证模式的 SQL Server 服务器还可以进行修改，修改的具体操作步骤如下：

(1) 在“对象资源管理器”窗口中，右击当前服务器名称，在弹出的快捷菜单中，选择“属性”命令，打开“服务器属性”对话框，在左侧的选项卡列表框中，选择“安全性”选项卡，展开安全性选项内容，如图 14-2 所示。在此选项卡中即可设置身份验证模式。

(2) 通过在“服务器身份验证”选项区域下，选择相应的单选按钮，可以确定 SQL Server 的服务器身份验证模式。无论使用哪种模式，都可以通过审核来跟踪访问 SQL Server 的用户，默认时仅审核失败的登录。

当启用审核后，用户的登录被记录于 Windows 应用程序日志、SQL Server 错误日志，这取决于如何配置 SQL Server 的日志。可用的审核选项如下：

- ① 无：禁止跟踪审核
- ② 仅限失败的登录：默认设置，选择后仅审核失败的登录尝试。
- ③ 仅限成功的登录：仅审核成功的登录尝试。

最后，单击“确定”按钮，完成登录验证模式的设置。

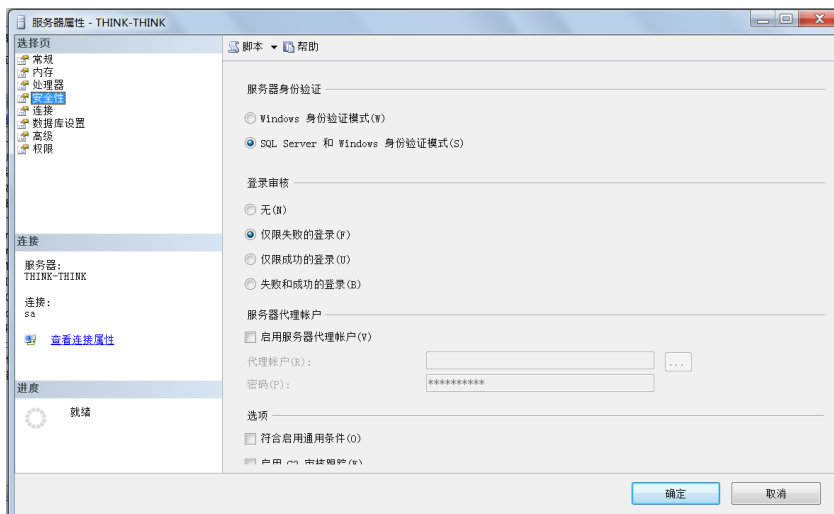


图 14-2 “安全性”选项卡

更改了服务器的登录模式后，需要重新启动 SQL Server 使新的身份验证模式生效。

2. 管理登录账号

登录属于服务器级的安全策略，要连接到数据库，首先要存在一个合法的登录。除了使用系统内置的登录账户外，用户经常需要自己创建登录账号。用户可以将 Windows 账号添加到 SQL Server 中，也可以新建 SQL Server 账号。具体操作过程如下。

(1) 打开 SSMS，并连接到服务器。依次展开“服务器”→“安全性”节点，在“登录名”节点上右击，在弹出的快捷菜单中选择“新建登录名”命令，打开“登录名-新建”窗口，如图 14-3 所示。

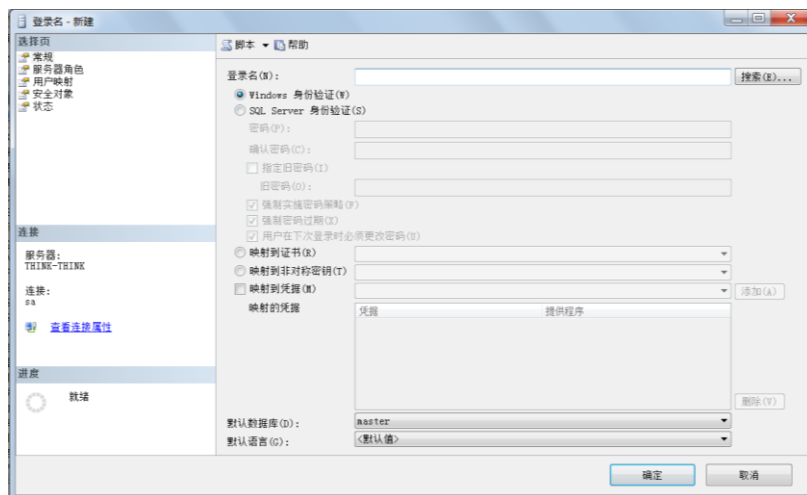


图 14-3 新建登录对话框

(2) 在“登录名”栏中,如果选择“Windows 身份验证”单选按钮,需要把 Windows 账号添加到 SQL Server 中,那么单击“登录名”行右端的“搜索”按钮,出现如图 14-4 所示的“选择用户和组”对话框。将已存在的 Windows 账号添加到 SQL Server 中即可。

说明： 对于 Windows 账号，账号名采用“域名（或计算机名）\用户（或组）名”的形式。

（3）如果希望新建一个 SQL Server 登录名，那么在新建登录对话框中，依次完成登录名、密码、确认密码和其他参数的设置。本例中新建一个 Music，选择采用 SQL Server 身份验证，强制实施密码策略，取消强制密码过期，选择默认数据库为 Music 数据库，如图 14- 5 所示。

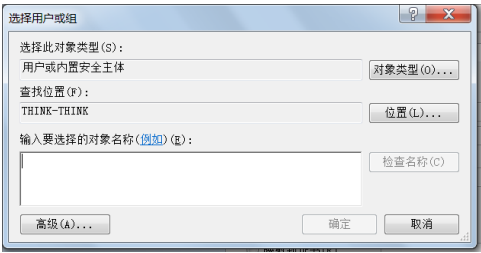


图 14- 4 “选择用户和组”对话框

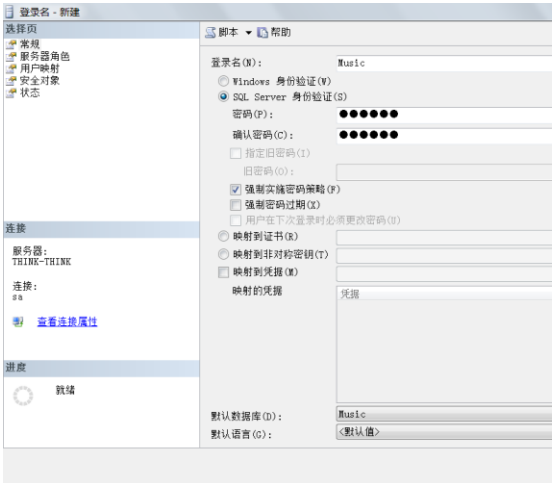


图 14- 5 新建 Music 登录名

（4）选择“选择页”中的“服务器角色”项，出现服务器角色设定页面，可以为此登录名设定服务器角色，如图 14- 6 所示。

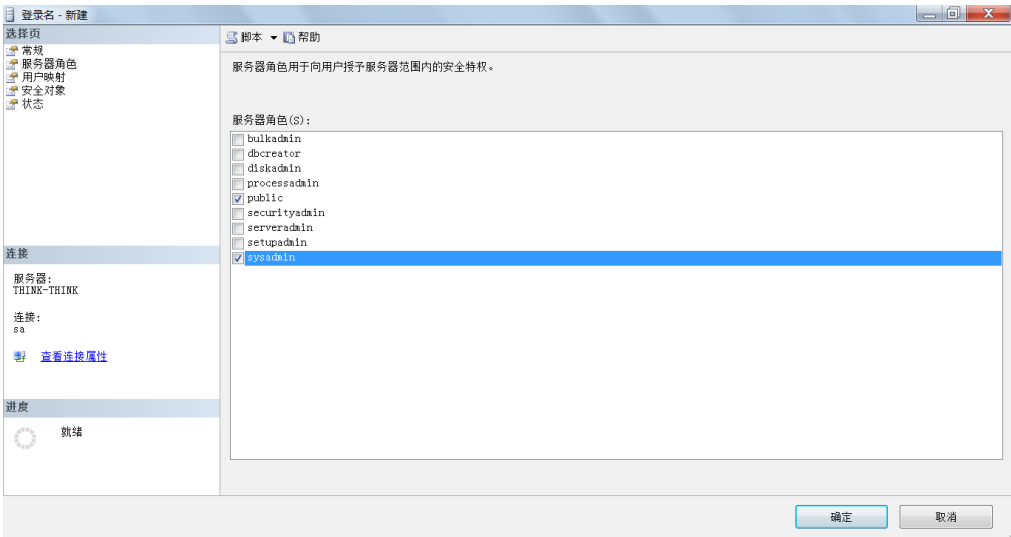


图 14- 6 服务器角色页面

（5）选择“选择页”中的“用户映射”项，进入映射设置页面。可以为这个新建的登录添加一个映射到此登录名的用户，并添加数据库角色，从而使得该用户获得数据库相应角色对应的数据库权限，如图 14- 7 所示。

说明: 自动映射的用户名和登录名相同。

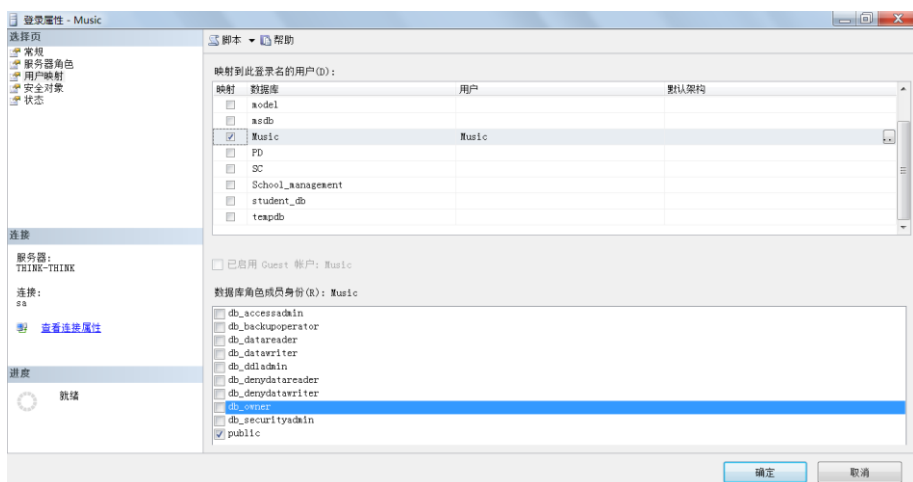


图 14-7 用户映射页面

(6) 单击“确定”按钮,完成 SQL Server 登录账户的创建。

下面来测试新的登录名 **Music** 能否成功连接到服务器。

(1) 在 SSMS 中, 选择“连接”→“数据库引擎”命令, 将打开“连接到服务器”对话框。

(2) 从“身份验证”下拉列表中, 选择“SQL Server 身份验证”选项, “登录名”文本框中输入 Music, “密码”文本框输入相应的密码, 如图 14-8 所示。

(3) 单击“连接”按钮，登录到服务器，如图 14-9 所示。



图 14-8 连接服务器

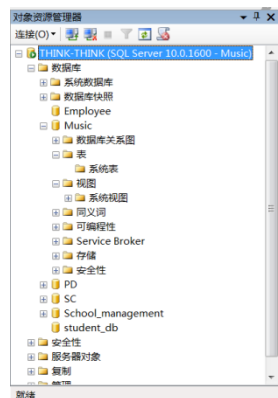


图 14-9 Music 登录连接服务器成功

(4) 由于 Music 登录名的默认数据库是 Music 数据库, 因此当访问其他数据库 (如 school_management 数据库) 时, 会出现如图 14-10 所示的错误提示信息。

3. 管理服务器角色

角色是对权限的集中管理机制。数据库管理员将操作数据库的权限赋予角色，然后数据库管理员再将角色赋给数据库用户或登录账户，从而使数据库用户或者登录账户拥有了相应

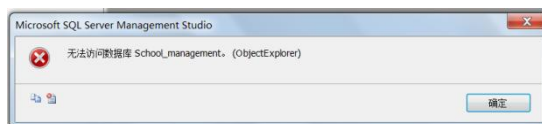


图 14-10 无法访问数据库

的权限。当角色的权限变更了，这些相关的用户权限都会发生变更。因此，角色可以方便管理员对用户权限的集中管理。

1) 固定服务器角色

为便于管理服务器上的权限，SQL Server 提供了若干“角色”。服务器级角色也称为“固定服务器角色”，因为不能创建新的服务器级角色。

服务器级角色的权限作用域为服务器范围。可以向服务器级角色中添加 SQL Server 登录名、Windows 账户和 Windows 组。服务器级角色的每个成员都可以向其所属角色添加其他登录名，新建的登录可以指派给这些服务器角色之中的任意一个角色。SQL Server 的服务器角色如表 14- 1 所示。

表 14- 2 服务器角色及说明

服务器级角色名称	说明
系统管理员（sysadmin）	可以在服务器上执行任何活动。默认情况下，Windows BUILTIN \Administrators 组（本地管理员组）的所有成员都是 sysadmin 的成员。通常情况下，这个角色仅适合数据库管理员（DBA）
服务器管理员（serveradmin）	可以更改服务器范围的配置选项和关闭服务器
安全管理员（securityadmin）	可以管理登录名及其属性，它们可以授予服务器级别和数据库级别的权限；此外，它们还可以重置 SQL Server 登录名的密码
进程管理员（processadmin）	可以终止在 SQL Server 实例中运行的进程
安装管理员（setupadmin）	可以添加和删除链接服务器
批量管理员（bulkadmin）	可以运行 BULK INSERT 语句，这条语句允许他们从文本文件中将数据导入到 SQL Server 数据库中
磁盘管理员（diskadmin）	用于管理磁盘文件，如添加备份设备等
数据库创建者（dbcreator）	可以创建、更改、删除和还原任何数据库
public	在 SQL Server 中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。这个数据库角色不能被删除

2) 查看服务器角色成员

（1）打开 SSMS，并连接到服务器。依次展开“服务器”→“安全性”→“服务器角色”选项，可以看到所有的服务器角色的列表，如图 14-11 所示。

（2）选择拟添加登录名的服务器角色，右击，在弹出的快捷菜单中选择“属性”命令选项，出现“服务器角色属性”对话框，如图 14-12 所示。然后单击“添加”按钮，打开“选择登录名”窗口，单击“浏览”按钮，打开“查找对象”对话框，如图 14-13 所示，选择拟添加的登录名。

（3）在“服务器角色属性”窗口中单击“确定”按钮，完成添加操作。

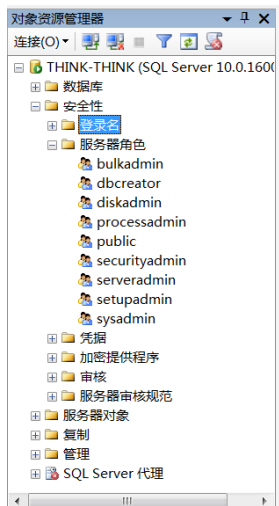


图 14-11 服务器角色

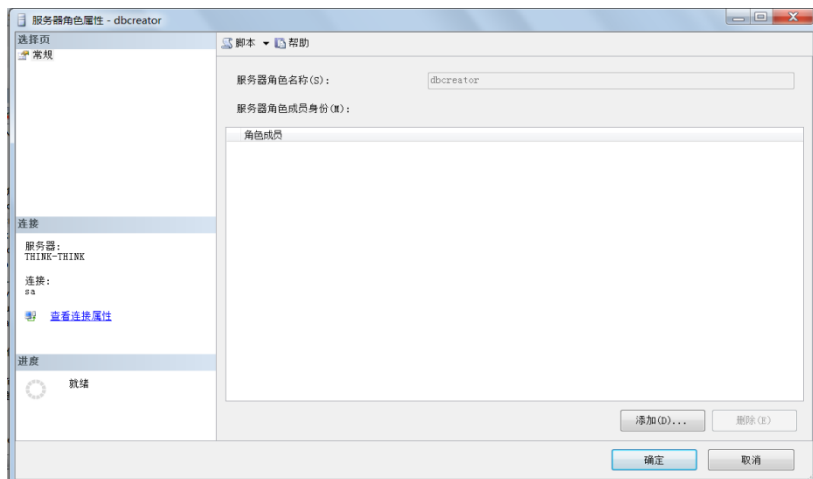


图 14-12 服务器角色属性对话框

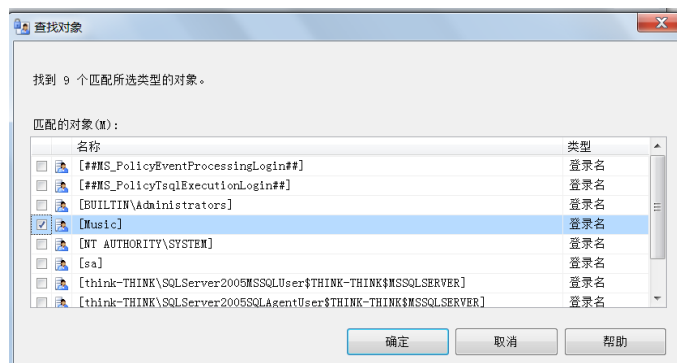


图 14-13 查找对象对话框

14.1.3 管理数据库的安全性

1. 管理数据库用户

用户是数据库级的安全策略，在为数据库创建新的用户前，必须存在创建用户的一个登录或者使用已经存在的登录创建用户。

无论是 Windows 账户还是 SQL Server 账户，登录数据库服务器后的其他操作都是相同的。首要任务都是要获得对数据库的访问权，在 SQL Server 中通过为登录账户指派数据库用户来使其获得对数据库的访问权。

一个数据库服务器可能有多个用户创建的多个数据库，除系统管理员外，普通的登录者一般不可能有权访问所有的数据库，因此，SQL Server 使每个数据库都具有自行创建数据库用户的功能，以达到仅指定登录名能够访问本数据库的目的。

2. 管理数据库角色

数据库角色是对数据库对象操作的权限的集合，数据库级角色的权限作用域为数据库范围。

SQL Server 中有两种类型的数据库级角色：固定数据库角色（数据库中预定义的）和用户自定义的数据库角色。

1) 固定数据库角色

固定数据库角色是在数据库级别定义的，并且存在于每个数据库中。**db_owner** 和 **db_securityadmin** 数据库角色的成员可以管理固定数据库角色成员身份。但是，只有 **db_owner** 数据库角色的成员能够向 **db_owner** 固定数据库角色中添加成员。固定数据库角色的每个成员都可向同一个角色添加其他登录名。

表 14- 3 显示了固定数据库级角色及其能够执行的操作。所有数据库中都有这些角色。

表 14- 3 固定数据库角色及权限

数据库级别的角色名称	说明
db_owner	可以执行数据库的所有配置和维护活动，还可以删除数据库。
db_securityadmin	可以修改角色成员身份和管理权限。
db_accessadmin	可以为 Windows 登录名、Windows 组和 SQL Server 登录名添加或删除数据库访问权限。
db_backupoperator	可以备份数据库。
db_ddladmin	可以在数据库中运行任何数据定义语言 (DDL) 命令。
db_datawriter	可以在所有用户表中添加、删除或更改数据。
db_datareader	可以从所有用户表中读取所有数据。
db_denydatawriter	不能添加、修改或删除数据库内用户表中的任何数据。
db_denydatareader	不能读取数据库内用户表中的任何数据。
public	每个数据库用户都属于 public 数据库角色。如果未向某个用户授予或拒绝对安全对象的特定权限时，该用户将继承授予该对象的 public 角色的权限。

2) 用户自定义角色

用户也可以创建新角色，使这个角色拥有某个或某些权限，创建的角色还可以修改其对应的权限。

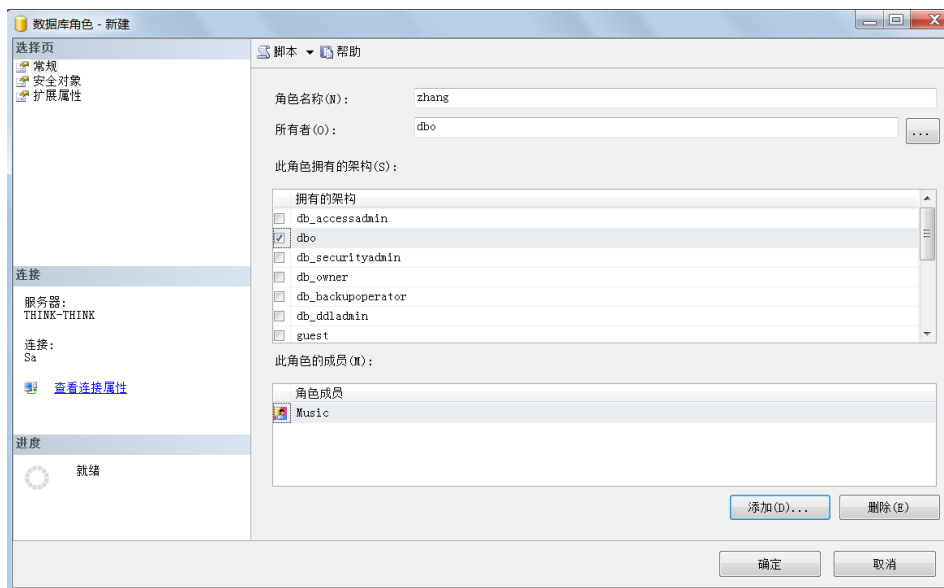


图 14- 16 数据库角色-新建对话框

用户可以使用 SSMS 图形界面工具来创建新角色，即需要完成以下任务：

- (1) 创建新的数据库角色；
- (2) 分配权限给创建的角色；
- (3) 将这个角色授予某个用户。

这不同于固定数据库角色，因为在固定角色中不需要指派权限，只需要添加用户。

步骤如下：展开要添加新角色的目标数据库的“安全性”节点，在“角色”上单击右键，弹出快捷菜单，选择“新建”→“新建数据库角色”命令。出现“数据库角色-新建”对话框，如图 14- 16 所示。在“常规”页面中，添加角色名称和所有者，选择所拥有的架构,也可以点击“添加”按钮为新角色添加用户。

14.1.4 管理数据库对象的安全性

1. 权限概述

权限用来控制用户如何访问数据库对象。一个用户可以直接分配到权限，也可以作为一个角色中的成员来间接得到权限。一个用户还可以同时属于不同权限的多个角色，这些不同的权限提供了对同一数据库对象的不同的访问级别。

SQL Server 中的权限分为 3 种：对象权限、语句权限和隐含权限。

1) 对象权限是用来控制一个用户如何与一个数据库对象进行交互操作的，有 5 个不同的权限：查询 (Select)、插入 (Insert)、修改 (Update)、删除 (Delete) 和执行 (Execute)。前 4 个用户表和视图，执行只用于存储过程。

2) 语句权限授予用户执行相应的语句命令的能力。包括 BACKUP DATABASE, BACKUP LOG, CREATE DATABASE, CREATE DEFAULT, CREATE FUNCTION, CREATE PROCEDURE, CREATE RULE, CREATE TABLE, CREATE VIEW。

3) 隐含权限是指系统预定义的服务器角色或数据库所有者和数据库对象所有者所拥有的权限。隐含权限不能明确的赋予和撤销。

2. 给数据库用户授予对象权限

对数据对象的安全性管理，SQL Server 通过对象权限来进行，比如对表的安全性的管

理，除了对表设置数据库用户的权限，还可以对列设置权限。

SQL Server 的权限控制操作可以通过在 SSMS 中操作，也可以使用 T-SQL 中的 GRANT、REVOKE 等语句来完成。

在 14.1.2 节中通过登录名 Music 的用户映射在 Music 数据库上，为数据库添加了一个用户，以该用户登录服务器，再对 Music 数据库进行操作，如添加表或查看表，都会显示如图所示的错误信息。原因在于没有为该用户设置权限，那么默认情况下，该用户属于 public 角色，但是 public 角色仅仅能“看到”数据库，不能操作数据库。如果希望该用户能够操作数据库对象，比如对“Singers”表进行 SELECT、INSERT 操作，可按以下方法进行。

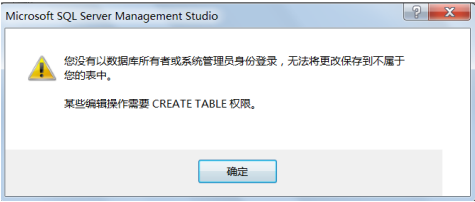


图 14- 17 用户没有相应操作权限的示例

1) 在 SSMS 中设置权限

- (1) 在“对象资源管理器”中，依次展开“Music 数据库”→“安全性”→“用户”，找到用户 Music，右击，出现“数据库用户-Music”属性编辑对话框。
- (2) 在“选项页”中选择“安全对象”选项，打开“安全对象”页面，如图 14- 18 所示，在此页面中，可以编辑 Music 的权限。

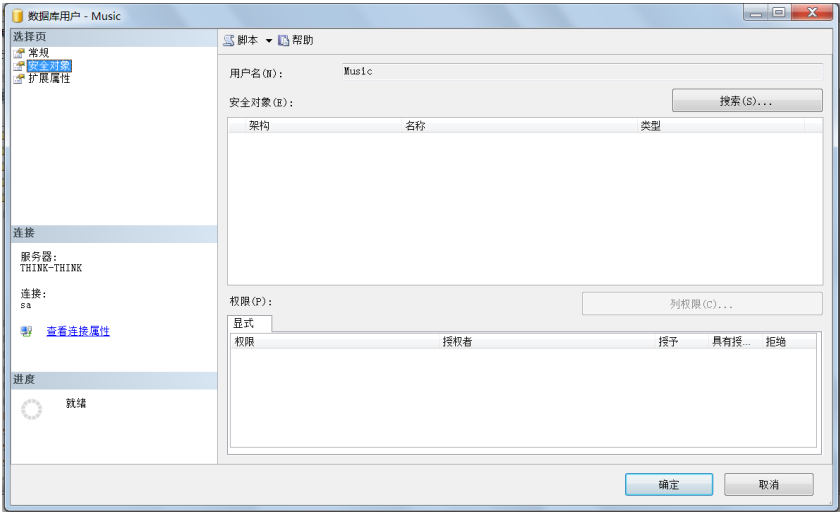


图 14- 18 安全对象页面

- (3) 点击“搜索”按钮，出现“添加对象”对话框，如图 14- 19 所示，单击要添加的对象类别前的单选按钮，单击“确定”按钮。
- (4) 出现“选择对象”对话框，如图 14- 20 所示，从中单击“对象类型”按钮。

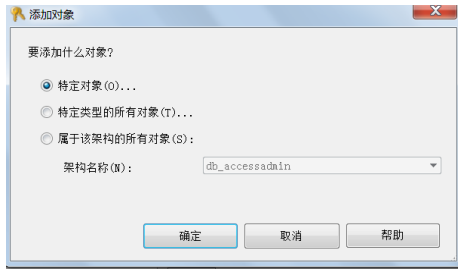


图 14- 19 添加对象

(5) 出现“选择对象类型”对话框，依次选择需要添加权限的对象类型前的复选框，如图 14- 21 所示，单击“确定”按钮。

(6) 回到“选择对象”对话框，此时在“选择这些对象类型”栏下出现了刚才选择的对象类型，如图 14-22 所示，单击“浏览...”按钮。

(7) 出现“查找对象”对话框，依次选中要添加权限的对象前的复选框，如图 14-23 所示。单击“确定”按钮。

(8) 回到“选择对象”对话框，这时在“输入要选择的对象名称”栏下已出现了刚才选择的对象，如图 14-24 所示。单击“确定”按钮。

(9) 回到“数据库用户”对话框，此时已包含了用户添加的对象，依次选择每一个对象，并在下面该对象的“权限”窗口中根据需要选择“授予”、“拒绝”以及“具有授予权限”。设置完一个对象的访问权限，单击“确定”按钮，完成给用户添加数据库对象权限的所有操作，如图 14-25 所示。

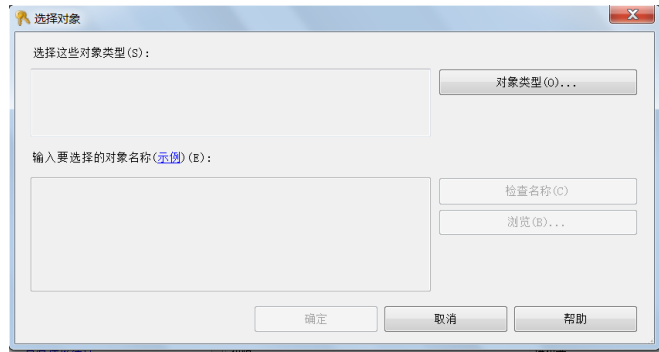


图 14- 20 选择对象

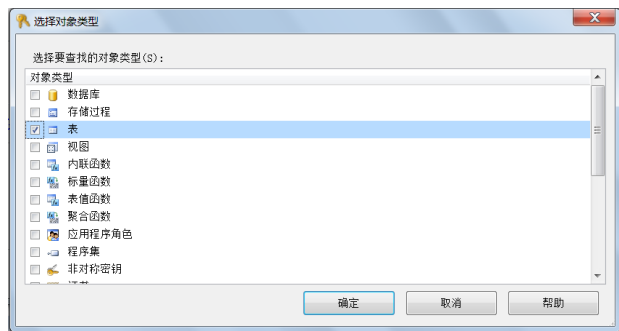


图 14- 21 选择对象类型

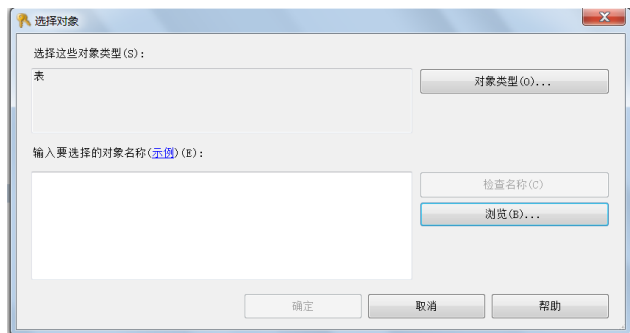
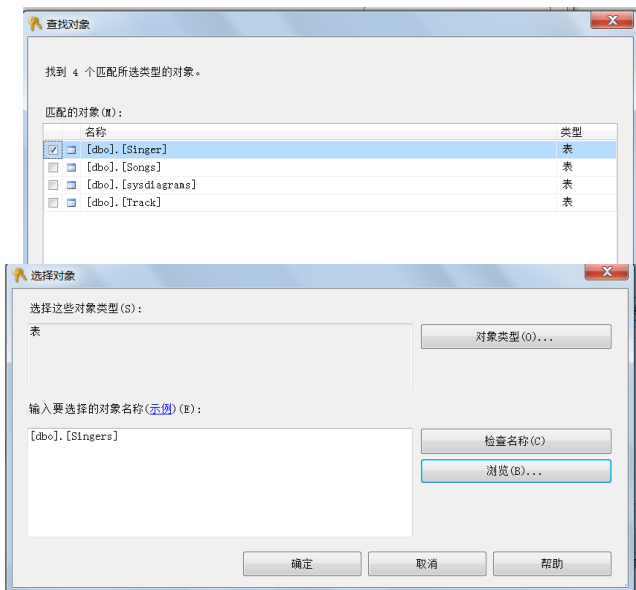


图 14- 22 已选择“表”对象的对话框



成功为用户设置权限后，可以进行验证。本例中，以 Music 登录服务器，在 Music 数据库的表中，可以看到选择多了一个 Singer 表，在查询编辑器中，输入“SELECT * FROM Singer”语句，可以查看该表的信息，如图 14-26 所示。同样，可以验证该用户也具有 Insert 的权限。

2) SQL 语句授权

上面的授予语句权限工作也可以用 GRANT 语句来完成，代码如下：

```
--将 Singer 表的 Select 和 Insert 权限赋给 Music 用户
grant Select,Insert on Singer to Music
```

图 14-24 添加 Singer 对象的对话框

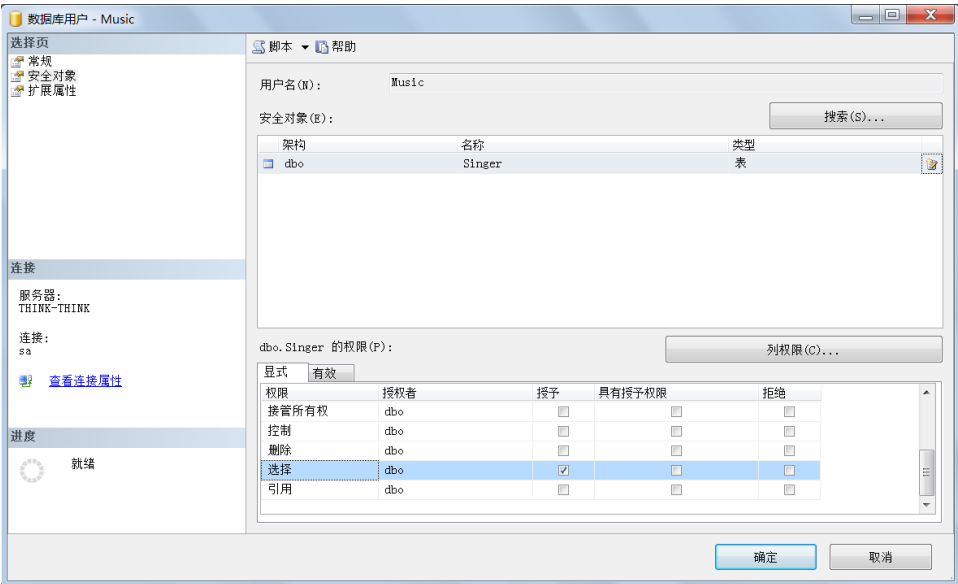


图 14-25 添加数据库对象权限

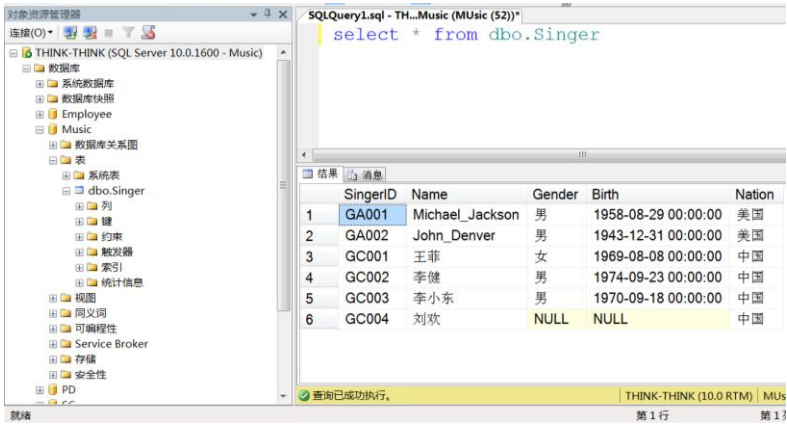


图 14- 26 Music 用户的查询 Singer 表的权限验证

SQL 的授权语句详见第七章 7.1.5 的介绍。

14.2 SQL Server 的备份和恢复

14.2.1 SQL Server 的恢复模式

SQL Server 的备份和还原操作是在“恢复模式”下进行的。恢复模式是一个数据库属性，它用于控制数据库备份和还原操作的基本行为。比如，恢复模式控制了将事务日志记录在日志中的方式，事务日志是否需要备份以及可用的还原操作。新的数据库可继承 model 数据库的恢复模式。

恢复模式的优点：

- ◆ 简化了恢复计划；
- ◆ 简化了备份和恢复过程；
- ◆ 明确了系统操作要求之间的权衡；
- ◆ 明确了可用性和恢复要求之间的权衡。

1. 三种恢复模式

SQL Server 包括 3 种恢复模式，其中每种恢复模式都能够在数据库发生故障的时候恢复相关的数据。不同的恢复模式在 SQL Server 备份、恢复的方式和性能方面存在差异，而且，采用不同的恢复模式对于避免数据损失的程度也不同。每个数据库必须选择三种恢复模式中的一种以确定备份数据库的备份方式。

1) 简单恢复模式

对于小型、不经常更新数据且对数据安全要求不太高的数据库，一般使用简单恢复模式。在该模式下，数据库会自动把不活动的日志删除，因此没有事务日志备份，这简化了备份和还原，但是代价是增加了在灾难事件中丢失数据的可能。没有日志备份，数据库只能恢复到最近的数据备份时间，而不能恢复到失败的时间点。

简单还原模式的优点在于日志的存储空间较小，能够提高磁盘的可用空间，而且也是最容易实现的模式。但是，使用简单恢复模式无法将数据库还原到故障点或特定的即时点。在该模式下，数据库只能做完整备份和差异备份。

2) 完全恢复模式

在完全恢复模式中，所有的事务都被记录下来，并保留所有的日志记录，直至将它们备份。如果日志文件本身没有损坏，则除了发生故障时正在进行的事务，SQL Server 可以还原所有的数据，可以将数据库还原到任意时间点。通常来说，对数据可靠性要求比较高的数据库需要使用该恢复模式。该模式也是 SQL Server 的默认恢复模式。

完整恢复模式支持所有的还原方案，可在最大范围内防止故障丢失数据，它包括数据库备份和事务日志备份，并提供全面保护，使数据库免受媒体故障影响。当然，它的时空和管理开销也最大。在该模式下，应该定期做事务日志备份，否则事务日志文件会变得很大。

3) 大容量日志记录恢复模式

大容量日志记录恢复模式是对完整恢复模式的补充。在该恢复模式下，只对某些大规模或者大容量数据操作（比如 **SELECT INTO**、**CREATE INDEX**、大批量装载数据、处理大批量数据）进行最小记录，保护大容量操作不受媒体故障的危害，提供最佳性能和最少的日志使用空间。例如，一次在数据库中插入数万条记录时，在完整恢复模式下，每一个插入记录的动作都会被记录在日志中，那么数万条记录将会使日志文件变得非常大。在大容量日志记录恢复模式下，只记录必要的操作，不记录所有日志，这样可以大大提高数据的性能。但是由于日志不完整，一旦出现问题，数据将有可能无法恢复。因此，一般只有在需要进行大量数据操作时，才将恢复模式改为大容量日志恢复模式，数据处理完成后，马上恢复到完整恢复模式。表 14- 4 列出了三种恢复模式的性能比较。

表 14- 4 三种恢复模式的性能比较

恢复模式	说明	数据丢失的风险	能否恢复到时间点
简单	无日志备份。 自动回收日志空间以减少空间需求，实际上不再需要管理事务日志空间。	最新备份之后的更改不受保护。在发生灾难时，这些更改必须重做。	只能恢复到备份的结尾。
完整	需要日志备份。 数据文件丢失或损坏不会导致丢失工作。 可以恢复到任意时间点。	正常情况下没有。 如果日志尾部损坏，则必须重做自最新日志备份之后所做的更改。	如果备份在接近特定的时点完成，则可以恢复到该时点。

大容量日志	<p>需要日志备份。</p> <p>是完整恢复模式的附加模式，允许执行高性能的大容量复制操作。</p> <p>通过使用最小方式记录大多数大容量操作，减少日志空间使用量。</p>	<p>如果在最新日志备份后发生日志损坏或执行大容量日志记录操作，则必须重做自该上次备份之后所做的更改，否则不丢失任何工作。</p>	<p>可以恢复到任何备份的结尾。</p> <p>不支持时点恢复。</p>
-------	--	---	--------------------------------------

2. 查看和修改数据库的恢复模式

可使用 SSMS 查看或更改数据库的恢复模式，方法如下：在“对象资源管理器”中，选择要设置恢复模式的数据库，右击数据库名，在弹出的快捷菜单中选择“属性”选项，在弹出的“数据库属性”对话框中选择“选项”标签在“恢复模式”下拉框中可以更改数据库的恢复模式，如图 14-27 所示。选择完成后，单击“确定”按钮完成操作。

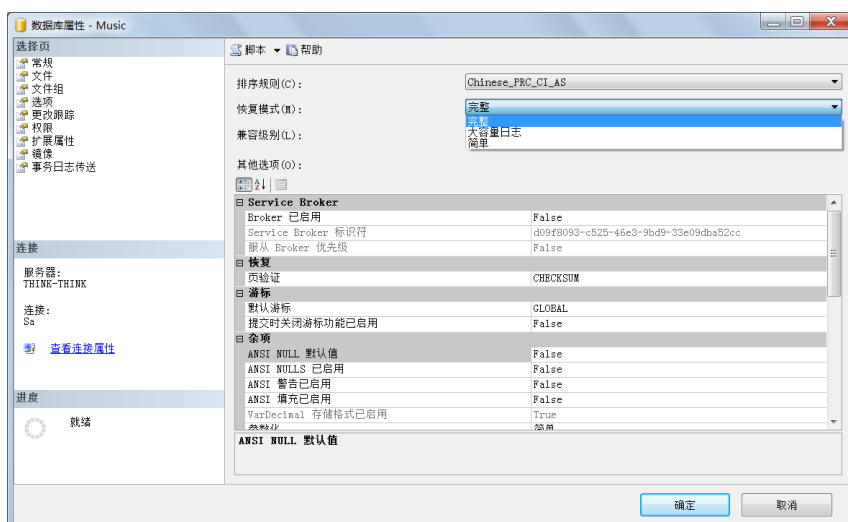


图 14- 27 数据库恢复模式

14.2.2 SQL Server 的备份

数据库备份就是创建完整数据库的副本，并将所有的数据项都复制到备份集，以便在数据库遭到破坏时能够恢复数据库。

1. 数据库的备份操作和对象

在备份数据库的时候，SQL Server 会执行如下操作：

- (1) 将数据库所有的数据页写到备份介质上。
- (2) 记录最早的事务日志记录的序列号。
- (3) 把所有的错误日志记录写到备份介质上。

在 SQL Server 系统中，只有获得许可的角色才可以备份数据，分别是固定的服务器角色 sysadmin、db_owner 和 db_backupoperator。当然，管理员也可以授权某些用户来执行备份工作。

2. 备份类型

SQL Server 提供了高性能的备份和恢复功能，用户可以根据需求设计自己的备份策略，以保护存储在 SQL Server 数据库中的关键数据。

SQL Server 支持多种备份类型，以下介绍其中的 4 种。

1) 完整数据库备份

完整数据库备份就是备份整个数据库，包括事务日志部分（以便可以恢复整个备份）。通过完整备份中的事务日志，可以使数据库恢复到备份完成时的状态。

完整数据库备份是任何备份策略中都要求完成的第一种备份类型，因为其他所有备份类型都依赖于完整备份。比如，如果没有执行完整备份，就无法执行差异备份和事务日志备份。

完整数据库备份与差异备份和日志备份相比，在备份的过程中需要花费更多的空间和时间，所以完整备份不需要频繁的进行，如果只使用完整数据库备份，那么进行数据恢复时只能恢复到最后一次完整备份时的状态，该状态之后的所有改变都将丢失。

还原数据库时，只需要一个完整备份文件，即可重新创建整个数据库。如果还原目标中已经存在数据库，还原操作将覆盖现有的数据库；如果目标中不存在数据库，还原操作将创建数据库。还原数据库将与备份完成时的数据库状态相符，但不包含未提交的事务。恢复数据库后，将回滚未提交的事务。

2) 差异数据库备份

差异备份仅记录自最近一次完整数据库备份以后发生改变的数据，比如，如果在完整备份后将某个文件添加至数据库，则下一个差异备份会包括该新文件。

差异备份基于以前的完整备份，是完整备份的补充，它比完整备份更小、更快，因此可以简化频繁的备份操作，减少数据丢失的风险。基于此，差异备份通常作为经常使用的备份。

在还原数据时，先要还原前一次做的完整备份后再还原最近一次所做的差异备份，这样才能让数据库中的数据恢复到与最后一次差异备份时的内容相同。

差异备份可以把数据库还原到完成差异备份的时刻，为了恢复到故障点，必须使用事务日志备份。

3) 事务日志备份

事务日志并不备份数据库本身，它只记录事务日志内容。事务日志记录了上一次完整备份或事务日志备份后数据库的所有变动过程，即它记录的是数据库某一段时间内的数据库变动情况，因此，事务日志备份依赖于完整备份，在做事务日志备份前，也必须要做完整备份。

事务日志备份比完整数据库节省时间和空间，而且利用事务日志进行恢复时，可以指定恢复到某一个事务，比如可以将其恢复到某个破坏性操作执行的前一个事务，完整备份和差异备份则不能做到。但是与完整数据库备份和差异备份相比，用日志备份恢复数据库要花费较长的时间，这是因为在还原数据时，除了先要还原完整备份，还要依次还原每个事务日志文件，而不是只还原最后一个事务日志文件。所以，通常情况下，事务日志备份经常与完整备份和差异备份结合使用，比如，每周进行一次完整备份，每天进行一次差异备份，每小时进行一次日志备份。这样，最多只会丢失一个小时的数据。

事务日志备份仅用于完整恢复模式和大容量日志恢复模式。

4) 文件和文件组备份

如果在创建数据库时，为数据库创建了多个数据库文件或文件组，可以使用该备份方式，选择对数据库中的部分文件或文件组进行备份。

利用文件和文件组备份，每次可以备份这些文件其中的一个或多个文件，而不是同时备份整个数据库，避免大型数据库备份的时间过长。另外，当数据库中的某个或某些文件损坏时，可以只还原损坏的文件或文件组备份。

第一次文件和文件组备份时必须备份完整的文件或文件组，即相当于一次完整备份。

说明：为了使恢复的文件与数据库的其余部分保持一致，执行文件和文件组备份之后，必须执行事务日志说明：

志备份。

3. 备份策略

当为特定数据库选择了满足业务要求的恢复模式后，需要计划并实现相应的备份策略。最佳备份策略取决于各种因素，以下因素尤其重要：

(1) 一天中应用程序访问数据库的时间长度：如果存在一个可预测的非高峰时段，则建议用户将完整数据库备份安排在此时段。

(2) 更改和更新可能发生的频率。如果更改经常发生，请考虑下列事项：

① 在简单恢复模式下，请考虑将差异备份安排在完整数据库备份之间。差异备份只能捕获自上次完整数据库备份之后的更改。

② 在完整恢复模式下，应安排经常的日志备份。在完整备份之间安排差异备份可减少数据还原后需要还原的日志备份数，从而缩短还原时间。

(3) 更改数据库所涉及的内容是小部分还是大部分？对于更改集中于部分文件或文件组的大型数据库，文件和文件组备份非常实用。

备份策略还要考虑的一个重要问题是如何提高备份和还原操作的速度。SQL Server 提供了以下两种加速备份和还原操作的方式：

(1) 使用多个备份设备：使得可将备份并行写入所有设备。备份设备的速度是备份吞吐量的一个潜在瓶颈。使用多个设备可按使用的设备数成比例提高吞吐量。同样，可将设备并行从多个设备还原。对于具有大型数据库的企业，使用多个备份设备可减少执行备份和还原操作的时间。SQL Server 最多支持 64 个备份设备同时执行一个备份操作。使用多个备份设备执行备份操作时，所有的备份媒体只能用于 SQL Server 备份操作。

(2) 结合完整备份、差异备份以及事务日志备份，可以最大限度地缩短恢复时间。创建差异数据库备份通常比完整数据库备份快，并减少了恢复数据库所需的事务日志量。

通常的备份策略是组合几种备份类型以形成适度的备份方案，以弥补单独使用一种类型的缺陷。常见的备份类型组合有：

(1) 完整备份

每次都对备份目标执行完整备份；备份和恢复操作简单，时空开销最大；适合于数据量不很大且更改不很频繁的情况。

(2) 完整备份+事务日志备份

定期进行数据库完整备份，并在两次完整备份之间按一定的时间间隔创建日志备份，增加事务日志备份的次数，以减少备份时间。此策略适合于不希望经常创建完整备份，但又不允许丢失太多的情况。

(3) 完整备份+事务日志备份+差异日志备份

创建定期的数据库完整备份，并在两次数据库完整备份之间按一定时间间隔创建差异备份，在完整备份之间安排差异备份可减少数据还原后需要还原的日志备份数，从而缩短还原时间；再在两次差异备份之间创建一些日志备份。此策略的优点是备份和还原的速度比较快，并且当系统出现故障时，丢失的数据也比较少。

14.2.3 备份数据库

服务器上所有备份和还原操作的完整历史记录都存储在 msdb 数据库中。SSMS 使用 msdb 中的备份历史记录来识别所指定的备份媒体上的数据库备份和所有事务日志备份，并创建还原计划。

1. 备份设备

备份或还原操作时使用的磁带机或磁盘驱动器称为“备份设备”。SQL Server 可以将数据库、事务日志或文件和文件组备份到磁盘或磁带设备上。

创建备份时，必须选择要将数据写入的备份设备。常见的备份设备可以分为 3 种类型：磁盘备份设备、磁带备份设备、物理和逻辑设备。

1) 磁盘备份设备

磁盘备份设备就是存储在硬盘或其他磁盘媒体上的文件，与常规操作系统文件一样，引用磁盘备份设备与引用任何其他操作系统文件一样。可以在服务器的本地磁盘上或共享网络资源的远程磁盘上定义磁盘备份设备，磁盘备份设备根据需要可大可小。最大的文件大小相当于磁盘上可用的闲置空间。

说明： 建议不要将数据库事务日志备份到数据库所在的同一物理磁盘上的文件中。如果包含数据库的磁盘设备发生故障，由于备份位于同一发生故障的磁盘上，因此无法恢复数据库。

2) 磁带备份设备

磁带备份设备的用法与磁盘设备相同，不过磁带设备必须物理连接到运行 SQL Server 实例的计算机上。若要将 SQL Server 数据备份到磁带，那么需要使用磁带备份设备或者 Windows 平台支持的磁带驱动器。

3) 物理和逻辑备份设备

SQL Server 使用物理设备名称或逻辑设备名称来标识备份设备。物理备份设备名称主要用来供操作系统对备份设备进行引用和管理，如：C:\Backups\Acco-unting\Full.bak。逻辑备份设备是物理备份设备的别名，通常比物理备份设备更能简单、有效地描述备份设备的特征。逻辑备份设备名称被永久保存在 SQL Server 的系统表中。备份或还原数据库时，物理备份设备名称和逻辑备份设备名称可以互换使用。

2. 创建备份设备

使用 SSMS 创建备份设备的操作步骤如下：

- 1) 在“对象资源管理器”中，单击服务器名称以展开服务器树。展开“服务器对象”节点，然后用鼠标右键单击“备份设备”选项。
- 2) 从弹出的菜单中选择“新建备份设备”命令，打开“备份设备”窗口。
- 3) 在“备份设备”窗口，在“设备名称”文本框中输入逻辑备份文件名，在“目标”→“文件”中，输入相应的物理备份设备名。这里创建一个名称为“bk_music”的备份设备。如图 14- 28 所示。
- 4) 单击“确定”按钮，完成备份设备的创建。展开“备份设备”节点，就可以看到刚创建的备份设备。

3. 备份数据库

SQL Server 备份数据库是动态的，即在数据库联机或者正在进行时可以执行备份操作。但在数据库备份操作时，不允许进行下列操作：创建或删除数据库文件；在数据库或数据库文件上执行收缩操作时截断文件。

使用 SSMS 图形化工具对数据库进行备份的操作步骤如下（以 Music 数据库创建完整备份为例）：

1) 在“对象资源管理器”中，展开“数据库”节点，要备份的数据库，在弹出的命令菜单中选择“任务”→“备份”选项。

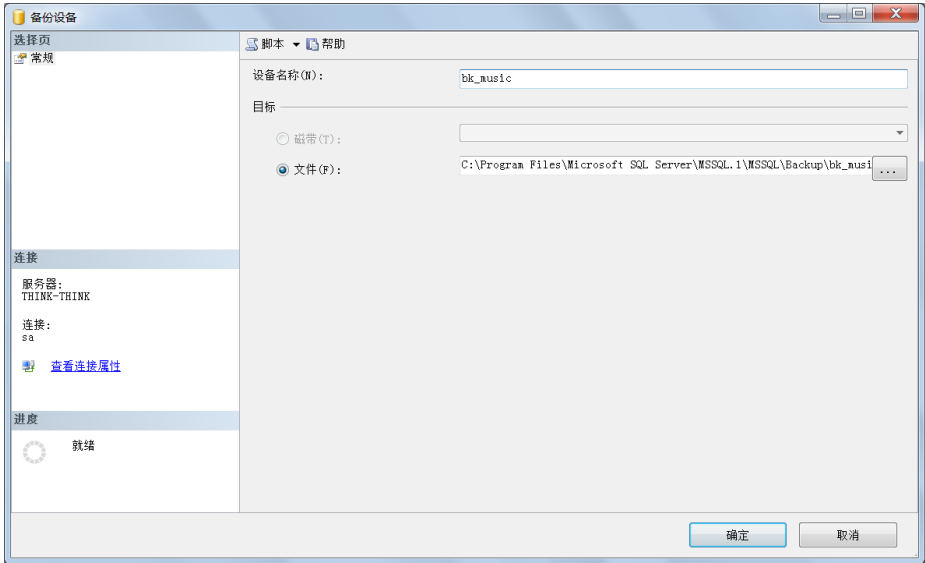


图 14- 28 创建备份设备

2) 出现如图 14- 29 所示的“备份数据库”窗口的“常规”选项卡。在此，输入备份的数据库，备份类型（本例选择完整备份）、备份组件、备份集名称以及目标（默认情况下为硬盘上以.bak 为扩展名的文件）。

另外，不同的备份类型，备份组件的选择也会不同。它们之间关系如表 14- 5 所示。

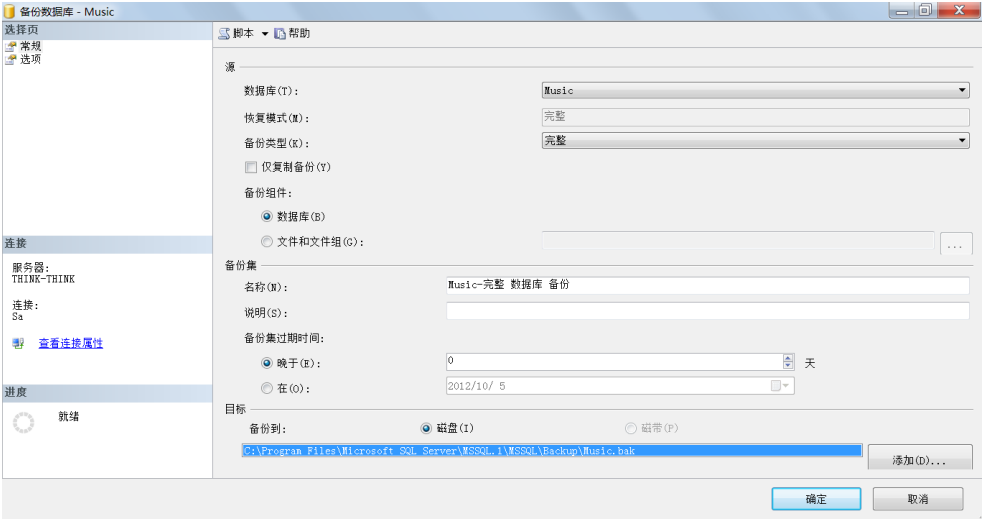


图 14- 29 备份数据库对话框

表 14- 5 数据库备份类型

备份类型	适用的备份组件	限制
完整备份	数据库、文件和文件组	对于 Master 数据库，只能执行完整备份。 在简单恢复模式下，文件和文件组只适用于只读文件组。
差异备份	数据库、文件和文件组	在简单恢复模式下，文件和文件组只适用于只读文件组。
事务日志	事务日志	事务日志备份不适合简单恢复模式。

3) 单击“确定”按钮，SQL Server 将自动完整备份过程。

根据以上的步骤，可以类似创建 Music 数据库的差异备份文件（如图 14- 30）和事务日志备份文件（如图 14- 31）。

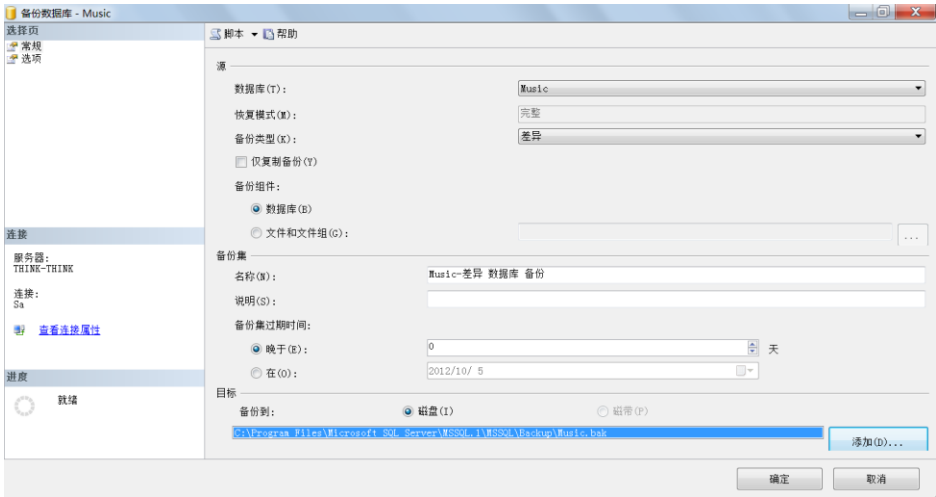


图 14- 30 差异备份

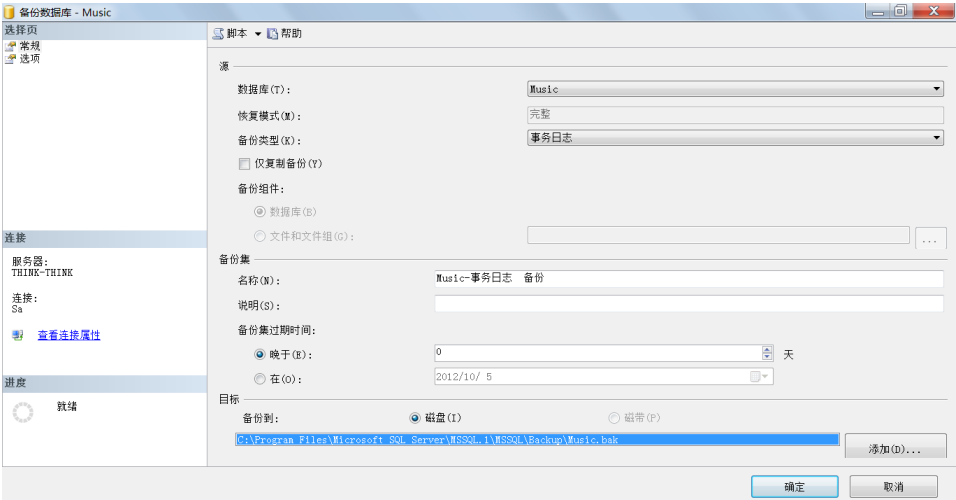


图 14- 31 事务日志备份

14.2.4 恢复数据库

1. 常规恢复

恢复数据库，就是让数据库根据备份的数据回到备份时的状态。当恢复数据库时，SQL Server 会自动将备份文件中的数据全部拷贝到数据库，并回滚任何未完成的事务，以保证数据库中的数据完整性。

恢复数据前，管理员应当断开准备恢复的数据库和客户端应用程序之间的一切连接，此时，所有用户都不允许访问该数据库，并且执行恢复操作的管理员也必须更改数据库连接到 master 或其他数据库，否则不能启动恢复进程。

在执行任何恢复操作前，用户要对事务日志进行备份，这样有助于保证数据的完整性。如果用户在恢复之前不备份事务日志，那么用户将丢失从最近一次数据库备份到数据库脱机

之间的数据更新。

使用 SSMS 工具恢复数据库的操作步骤如下：

1) 在对象资源管理器中，展开“数据库”节点，右击 Music 数据库，在弹出的命令菜单中选择“任务”→“还原”→“数据库”命令，打开“还原数据库”窗口，如图 14- 32 所示。

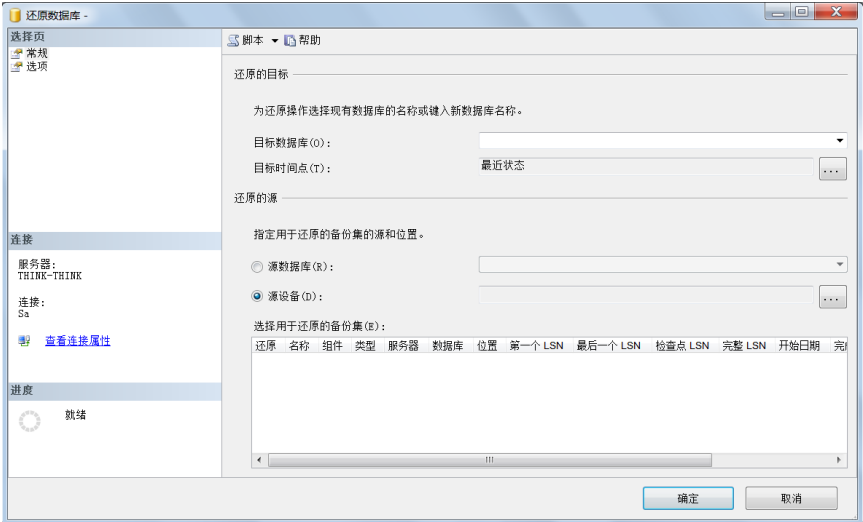



图 14- 32 还原数据库对话框

2) 在“还原数据库”窗口中输入目标数据库名，选中“源设备”单选按钮，然后单击  弹出一个“指定备份”对话框，在“备份媒体”选项中选择“文件”选项，然后单击“添加”按钮，从选择文件对话框中选择之前备份文件，如图 14- 33 所示。

3) 选择完成后，单击“确定”按钮返回。在“还原数据库”窗口，就可以看到该备份设备中的所有的数据库备份内容。

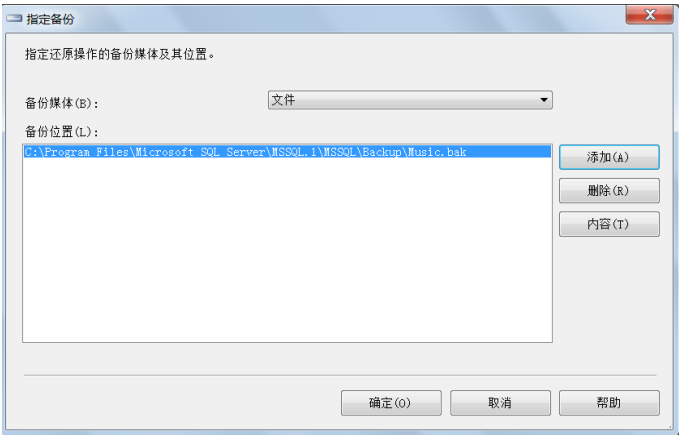


图 14- 33 指定备份

本例中有三种备份文件：完整备份、差异备份和日志备份。因此可以有多种的恢复方案：

- (1) 利用完整备份
- (2) 利用完整备份+差异备份
- (3) 利用完整备份+日志备份
- (4) 利用完整备份+差异备份+日志备份

根据不同的恢复方案来选择需要的备份文件。本例中复选“完整”、“差异”和“日志”

三种备份，可这使数据库恢复到最近一次备份的正确状态，如图 14- 34 所示。

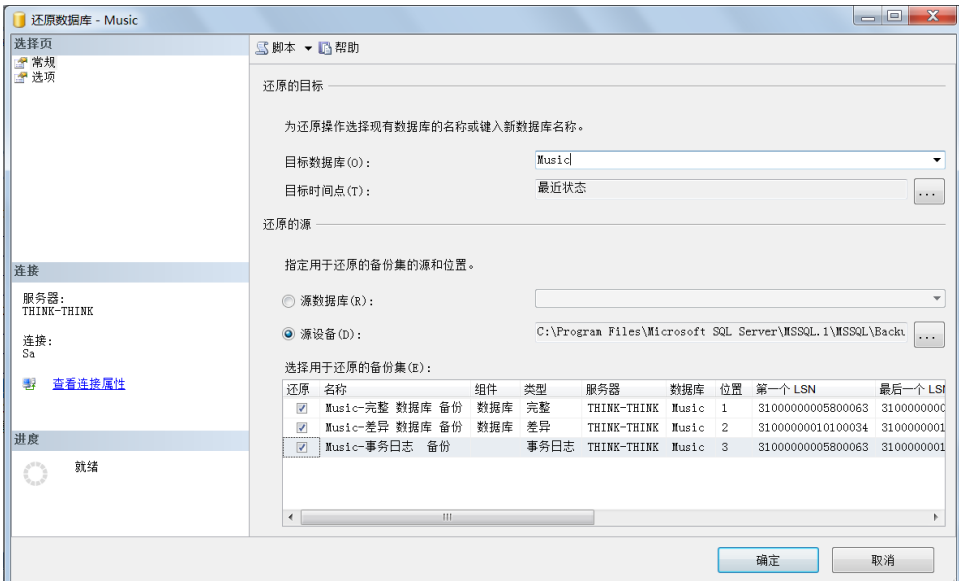


图 14- 34 选择备份集

4) 单击“选项页”中的“选项”，在该页面可以设置“还原选项”和“恢复状态”（图 14- 35），如果还需要恢复别的备份文件，需要选择 **RESTORE WITH NORECOVERY** 选项，恢复完成后，数据库会显示处于正在还原状态，无法进行操作，必须到最后一个备份还原为止。



图 14- 35 选项窗口

5) 单击“确定”按钮，完成对数据库的还原操作。还原完成弹出还原成功消息对话框。

说明： 当执行还原最后一个备份时候，必须选择 **RESTORE WITH RECOVERY** 选项，否则数据库将一直处于还原状态。

2. 时间点恢复

SQL Server 在进行事务日志备份时，不仅给事务日志的每个事务标上了日志号，还给它们都标上了时间。如果某事务错误更改了一些数据，则可能需要将该数据库恢复到紧邻在不正确数据项之前的那个恢复点。为数据库指定恢复点的任何恢复都称为“时间点恢复”。

时间点恢复仅适用于使用完整恢复模式或大容量日志恢复模式的 SQL Server 数据库，时间点恢复的恢复点通常位于事务日志备份中。

图 14- 36 显示了在时间 t9 处执行的到事务日志中间某个恢复点的还原。在时间 t10 处执行的此备份剩余部分以及随后日志备份中的更改将被丢弃。

假设数据库在 21:17 分做了完整备份，在 21:24 分做了事务日志备份。如果在 21:20 的时间做了误操作，清除了很多数据，那么可以通过日志备份的时间点恢复，将时间点设置在 21:20，既可以保存在 21:20 之前的数据修改，又可以忽略 21:20 之后的错误操作。该时间点恢复的步骤如下：

- 1) 在“还原数据库”窗口中，选择完整备份和事务日志备份（图 14- 37）。
- 2) 单击“目标时间点”按钮，打开“时点还原”窗口，启用“具体时间和日期”单选按钮，输入具体时间，如图 14- 38 所示。
- 3) 设置完成后，单击“确定”按钮，然后还原备份。

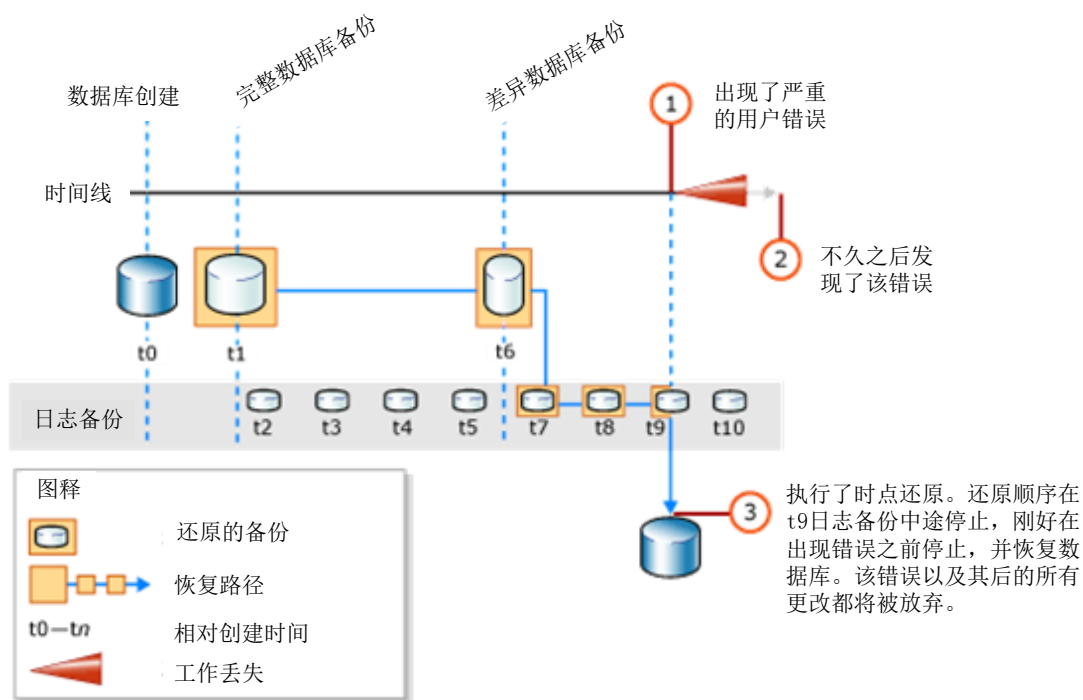


图 14- 36 时间点恢复示例

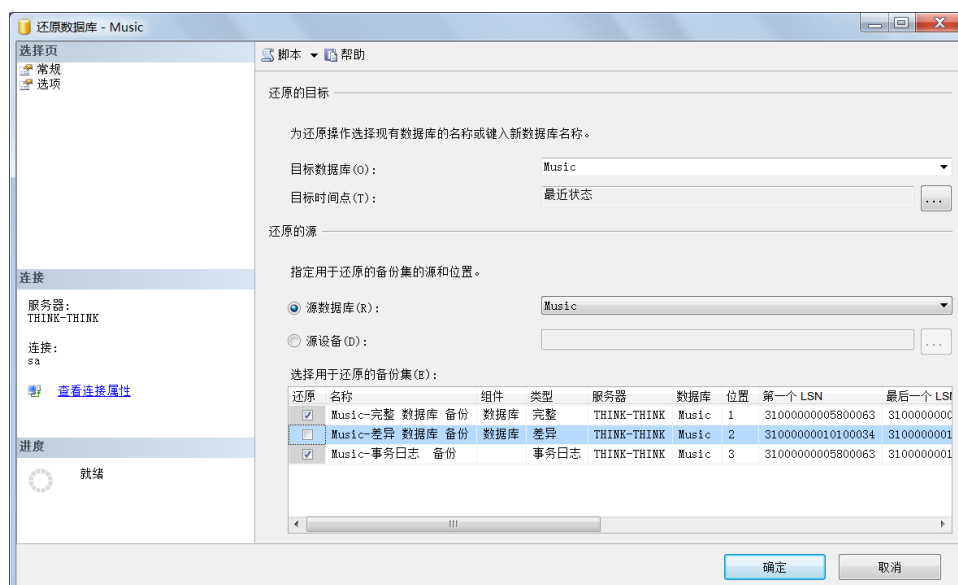


图 14- 37 选择备份文件

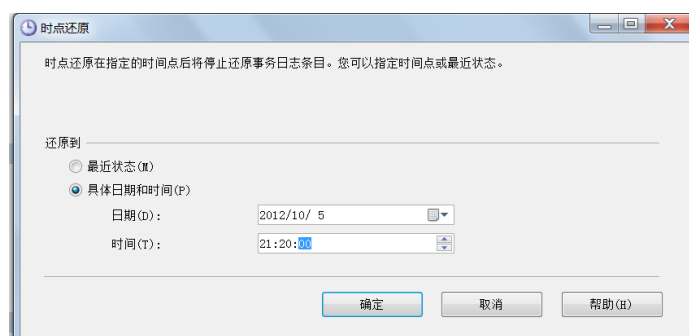


图 14- 38 设置时间点

14.3 SQL Server 的并发机制

事务和锁是并发控制的主要机制，SQL Server 通过支持事务机制来管理多个事务，保证数据的一致性，并使用事务日志保证修改的完整性和可恢复性。

SQL Server 遵从三级封锁协议，从而有效的控制并发操作可能产生的丢失更新、读“脏”数据、不可重复读等错误。

SQL Server 具有多种不同粒度的锁，允许事务锁定不同的资源，并能自动使用与任务相对应的等级锁来锁定资源对象，以使锁的成本最小化。

14.3.1 SQL Server 的隔离级别

1. 隔离级别

由 7.3.2 节可知，并发操作会带来一些一致性问题：丢失更新、脏读、不可重复读和幻读。在 SQL Server 中，可以为事务设置隔离级别，来决定允许以上的哪些问题。

隔离（Isolation）是计算机安全学中的一种概念，其本质是一种封锁机制。在 SQL Server 中，隔离级别（Isolation Level）定义一个事务必须与由其他事务进行的资源或数据更改相隔离的程度。由隔离级别可以指定数据库如何保护（锁定）那些当前正在被其他事务请求使用的数据。隔离级别从允许的并发副作用（例如，脏读或幻读）的角度进行描述。应用程序要

求的隔离等级确定了 SQL Server 使用的锁定行为。

事务隔离级别控制：

- 1) 读取数据时是否占用锁以及所请求的锁类型。
- 2) 占用读取锁的时间。
- 3) 引用其他事务修改的行的读取操作是否：
 - (1) 在该行上的排他锁被释放之前阻塞其他事务。
 - (2) 检索在启动语句或事务时存在的行的已提交版本。
 - (3) 读取未提交的数据修改。

较低的隔离级别可以增加并发，但代价是降低数据的正确性。相反，较高的隔离级别可以确保数据的正确性，但可能对并发产生负面影响。

SQL Server 支持以下的隔离级别：

1) READ UNCOMMITTED (未提交读)

READ UNCOMMITTED 级别是限制最低的隔离级别，当 READ UNCOMMITTED 被应用到事务时，事务中的查询将不会在数据库对象上应用任何锁（包括共享锁），相当于使用了 No Lock 提示的 Select 语句。所以当使用这个隔离级别的时候，将不会阻止其他的事务读或者写当前事务请求的数据。由于这种特性，导致应用 READ UNCOMMITTED 的事务存在读取其他事务修改但并未提交的数据（脏读）。

2) READ COMMITTED (已提交读)

READ COMMITTED 级别是 SQL Server 的默认事务隔离级别。指定事务不能读取已由其他事务修改但尚未提交的数据，因此，可以避免脏读。

READ COMMITTED 有两种实现方式，这取决于 READ_COMMITTED_SNAPSHOT (已提交读快照) 数据库选项的设置情况。当 READ_COMMITTED_SNAPSHOT 为 ON 时，数据库引擎将会通过行版本控制为事务中的每一个查询建立一个事务级的数据快照。数据快照包含在执行查询前的所有符合查询条件的数据，使用这种方法将不会对数据库对象应用任何数据库锁。

如果 READ_COMMITTED_SNAPSHOT 为 OFF 时，数据库引擎将在执行查询时，对数据库对象应用共享锁来避免其他事务修改当前事务中正在执行的查询语句所访问的对象。

已提交读在读数据的时候使用共享锁，但在读操作完成后会立即释放这个锁。因此，其他事务可以更改刚被读过的数据，所以使用 READ COMMITTED 级别时有可能出现不可重复读或幻读的情况。

说明： 激活已提交读快照级别的语句：`ALTER DATABASE 数据库名 SET READ_COMMITTED_SNAPSHOT ON`

3) REPEATABLE READ (可重复读)

REPEATABLE READ 在 READ COMMITTED 的基础上，延长了执行查询时对访问的数据应用的共享锁的周期。在每次执行查询时，都要对事务中的每个语句所读取的全部数据都设置了共享锁，并且该共享锁一直保持到事务完成为止。这样可以防止其他事务修改当前事务读取的任何行（避免不可重复读）。但是其他事务可以插入与当前事务所发出语句的搜索条件相匹配的新行。如果当前事务随后重试执行该语句，它会检索新行，从而产生幻读。

4) SNAPSHOT (快照)

SNAPSHOT 隔离级别会为事务中的所有查询语句建立事务级的数据快照，数据快照包含开始事务前的所有已经提交的数据。这种模式同样利用了行版本控制机制。

每个事务对自己复制的数据进行修改，当事务准备更新的时候，检查数据从开始使用以来是否被修改并且决定是否更新数据。

5) SERIALIZABLE (可序列化)

SERIALIZABLE 是最严格的隔离级别，并发最少，事务之间完全隔离，等同于串行。数据被读取或更新的时候，没有事务可以读取、修改或插入数据。

与 REPEATABLE 相比，应用这个隔离级别的事务将会在访问数据上加键范围锁，键范围锁处于与事务中执行的每个语句的搜索条件相匹配的键值范围之内，其他事务将不能在数据库对象上修改或者插入任何数据。键范围锁也会被一直占用直到整个事务结束（相当于 Select 语句的 HOLDLOCK 查询提示）。因此，使用这个隔离级别可以避免幻象数据。

键范围锁放置在索引上，指定开始键值和结束键值。此锁将阻止任何要插入、更新或删除任何带有该范围内的键值的行的尝试，因为这些操作会首先获取索引上的锁。例如，可序列化事务可能说明：发出了一个 SELECT 语句，以读取其键值介于 'AAA' 与 'CZZ' 之间的所有行。从 'AAA' 到 'CZZ' 范围内的键值上的键范围锁可阻止其他事务插入带有该范围内的键值（例如 'ADG'、'BBD' 或 'CAL'）的行。

表 14- 6 总结了每种隔离级别允许的行为。

表 14- 6 四种隔离级别允许的行为

隔离级别	更新丢失	脏读	重复读取	幻读
未提交读取	N	Y	Y	Y
提交读取	N	N	Y	Y
可重复读	N	N	N	Y
快照	N	N	N	N
可序列化	N	N	N	N

2. 事务隔离级别的系统开销

由于数据库锁机制和行版本控制都要占用系统资源，并且数据库锁的占用和释放将会影响到并发事务处理的响应速度和数据库死锁，所以在使用事务隔离级别时要考虑到不同的应用和各种不同隔离级别的系统开销情况。

1) READ COMMITTED (已提交读)

如果实现 READ COMMITTED 隔离，将会对事务中查询的目标数据库对象加共享锁。如果 READ_COMMITTED_SNAPSHOT 为 ON 时，对系统的主要影响在于数据库引擎需要提供额外的资源来管理数据快照，所有的数据快照将被保存在数据库的 TempDB 中，所以当访问大量数据时，需要考虑 I/O 子系统的吞吐量以及 TempDB 是否有充分的空间来维护数据库实例中所有数据库的行版本请求，以及其他涉及 TempDB 的操作。此外，数据库服务器的处理器系统和快速存储系统也将承受更大的负载。

2) REPEATABLE (可重复读)

在事务执行过程中一直占用数据库对象的锁资源将会导致其他事务在当前事务执行过程中被永久性阻塞。如果当前事务中存在长时间的操作且系统没有超时处理机制，将会严重影响事务处理的响应速度，甚至出现数据库死锁。由于在 REPEATABLE 事务隔离级别下，其他事务可以对当前事务访问的数据库对象执行插入操作，将不会影响并发的插入操作（但需要在幻象数据与性能之间进行权衡）。

3) SERIALIZABLE (可序列化)

尽量避免使用的隔离级别，应用该隔离级别的事务将完全阻塞其他需要访问当前事务正在访问的数据库对象的事务。在并发操作数量庞大时，即使事务处理响应速度较快，也会大大降低系统整体响应速度。

3. 设置隔离级别

使用 `SET TRANSACTION ISOLATION LEVEL` 语句设置隔离级别。语法为：

```
SET TRANSACTION ISOLATION LEVEL
{READ UNCOMMITTED → READ COMMITTED → REPEATABLE READ
→ SNAPSHOT → SERIALIZABLE } [ ;]

BEGIN TRANSACTION
.....
.....
COMMIT TRANSACTION
```

例如以下的事务设置隔离级别为 `READ COMMITTED`。

```
--事务设置隔离级别为 READ COMMITTED
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
BEGIN TRANSACTION TR
BEGIN TRY
UPDATE Contact SET EmailAddress='jolyn@yahoo.com'
WHERE ContactID = 1070
UPDATE EmployeeAddress SET AddressID = 32533
WHERE EmployeeID = 1
COMMIT TRANSACTION TR
SELECT 'Transaction Executed'
END TRY
BEGIN CATCH
ROLLBACK TRANSACTION TR
SELECT 'Transaction Rollbacked'
END CATCH
```

14.3.2 SQL Server 的锁模式

1. SQL Server 的锁控制

锁是 SQL Server 数据库引擎用来同步多个用户同时对同一个数据块的访问的一种机制。在事务获取数据块读取或修改某一数据块之前，它必须保护自己不受其他事务对同一数据进行修改的影响。事务通过请求锁定数据块来达到此目的。

对于 SQL 的锁，说明如下：

1) 锁有多种模式，如共享或独占。锁模式定义了事务对数据所拥有的依赖关系级别。如果某个事务已获得特定数据的锁，则其他事务不能获得会与该锁模式发生冲突的锁。如果事务请求的锁模式与已授予同一数据的锁发生冲突，则数据库引擎实例将暂停事务请求直到第一个锁释放。

2) 当事务修改某个数据块时，它将持有保护所做修改的锁直到事务结束。事务持有（所获取的用来保护读取操作的）锁的时间长度，取决于事务隔离级别设置。一个事务持有的所有锁都在事务完成（无论是提交还是回滚）时释放。

3) 锁由数据库引擎的一个部件（称为“锁管理器”）在内部管理。当数据库引擎实例处

理 T-SQL 语句时，数据库引擎查询处理器会决定将要访问哪些资源。查询处理器根据访问类型和事务隔离级别设置来确定保护每一资源所需的锁的类型，然后，查询处理器将向锁管理器请求适当的锁。如果与其他事务所持有的锁不会发生冲突，锁管理器将授予该锁。

4) 一般情况下，SQL Server 能自动提供加锁功能，不需要用户专门设置，这些功能表现在：

(1) 当用 SELECT 语句访问数据库时，系统能自动用共享锁访问数据；在使用 INSERT、UPDATE 和 DELETE 语句增加、修改和删除数据时，系统会自动给使用数据加排它锁。

(2) 系统用意向锁使锁之间的冲突最小化。意向锁建立一个锁机制的分层结构，其结构按行级锁层、页级锁层和表级锁层设置。

(3) 当系统修改一个页时，会自动加更新锁。更新锁与共享锁兼容，而当修改了某页后，修改锁会上升为排它锁。

(4) 当操作涉及到参照表或索引时，SQL Server 会自动提供模式锁和修改锁。

因此，SQL Server 能自动使用与任务相对应的等级锁来锁定资源对象，以使锁的成本最小化。所以，用户只需要了解封锁机制的基本原理，使用中不涉及锁的操作。也可以说，SQL Server 的封锁机制对用户是透明的。

2. SQL Server 的锁模式

SQL Server 数据库引擎使用不同的锁模式锁定资源，这些锁模式确定了并发事务访问资源的方式。

1) 共享锁 (Shared locks)：共享锁 (S 锁) 允许并发事务读取 (select) 一个资源。资源上存在共享锁 (S 锁) 时，任何其他事务都不能修改数据。一旦读取操作一完成，就立即释放资源上的共享锁 (S 锁)，除非将事务隔离级别设置为可重复读或更高级别，或者在事务生存周期内用锁定提示 (holdlock) 保留共享锁。

2) 排他锁 (Exclusice locks)：排他锁 (X 锁) 可以防止并发事务对资源进行访问。使用排他锁 (X 锁) 时，任何其他事务都无法修改数据。仅在使用 NOLOCK 提示或未提交读隔离级别时才会进行读取操作。

数据修改语句 (如 INSERT、UPDATE 和 DELETE) 合并了修改和读取操作，语句在执行所需的修改操作之前首先执行读取操作以获取数据。因此，数据修改语句通常请求共享锁和排他锁。例如，UPDATE 语句可能根据与一个表的联接修改另一个表中的行，在此情况下，除了请求更新行上的排他锁之外，UPDATE 语句还将请求在联接表中读取的行上的共享锁。

3) 更新锁 (Update locks)：用在可更新的资源中，可以防止常见的死锁。因为更新是一个事务，该事务先读取记录，获得资源的 S 锁，完后修改，此操作要求把锁转换为 X 锁，如果两个事务获得了资源上的 S 锁，然后试图同时更新数据，会发生死锁。

一次只有一个事务可以获得资源的更新锁 (U 锁)。如果事务修改资源，则更新锁 (U 锁) 转换为排他锁 (X 锁)。当 SQL Server 准备更新数据时，它首先对数据对象作更新锁锁定，这样数据将不能被修改，但可以读取。等到 SQL Server 确定要进行更新数据操作时，它会自动将更新锁换为独占锁。但当对象上有其它锁存在时，无法对其作更新锁锁定。

4) 意向锁 (Intent locks)：意向锁的含义是如果对一个结点加意向锁，则说明该结点的下层结点正在被加锁；对任一结点加锁时，必须先对它的上层结点加意向锁。

例如，对任一元组加锁时，必须先对它所在的关系加意向锁。数据库引擎使用意向锁来保护共享锁 (S 锁) 或排他锁 (X 锁) 放置在锁层次结构的底层资源上。

意向锁包括意向共享 (IS)、意向排他 (IX) 以及意向排他共享 (SIX)。

5) 模式锁 (Schema locks)：执行表的数据定义语言 (DDL) 操作 (例如添加列或删除表) 时使用模式修改锁 (Sch-M 锁)。在模式修改锁 (Sch-M 锁) 起作用的期间，会防止

对表的并发访问。这意味着在释放模式修改锁（Sch-M 锁）之前，该锁之外的所有操作都将被阻止。

6) 批量更新锁（Bulk update locks）：大容量更新锁（BU 锁）允许多个线程将数据并发地大容量加载到同一表，同时防止其他不进行大容量加载数据的进程访问该表。

7) 键范围锁：在使用可序列化事务隔离级别时，对于 T-SQL 语句读取的记录集，键范围锁可以隐式保护该记录集中包含的行范围，键范围锁可防止幻读。

表 14- 7 中给出了 SQL Server 的锁模式及其说明。

表 14- 7 SQL Server 的锁模式及说明

锁模式	说明
共享 (S)	用于不更改或不更新数据的读取操作，如 SELECT 语句。
更新 (U)	用于可更新的资源中。防止当多个会话在读取、锁定以及随后可能进行的资源更新时发生常见形式的死锁。
排他 (X)	用于数据修改操作，例如 INSERT、UPDATE 或 DELETE。确保不会同时对同一资源进行多重更新。
意向	用于建立锁的层次结构。意向锁的类型有：意向共享 (IS)、意向排他 (IX) 以及意向排他共享 (SIX)。
架构	在执行依赖于表架构的操作时使用。架构锁的类型有：架构修改 (Sch-M) 和架构稳定性 (Sch-S)。
批量更新 (BU)	在向表进行大容量数据复制且指定了 TABLOCK 提示时使用。
键范围	当使用可序列化事务隔离级别时保护查询读取的行的范围，确保再次运行查询时其他事务无法插入符合可序列化事务的查询的行。

14.3.3 SQL Server 中死锁的处理

在两个或多个任务中，如果每个任务锁定了其他任务试图锁定的资源，此时会造成这些任务永久阻塞，从而出现死锁。如图 14- 39 所示，

任务 T1 具有资源 R1 的锁（通过从 R1 指向 T1 的箭头指示），并请求资源 R2 的锁（通过从 T1 指向 R2 的箭头指示）。

任务 T2 具有资源 R2 的锁（通过从 R2 指向 T2 的箭头指示），并请求资源 R1 的锁（通过从 T2 指向 R1 的箭头指示）。

因为这两个任务都需要有资源可用才能继续，而这两个资源又必须等到其中一个任务继续才会释放出来，所以陷入了死锁状态。

本节中主要介绍 SQL Server 如何处理死锁问题。首先我们从一个死锁的例子入手，比如做以下操作：

1) 在第一个查询窗口中执行如下代码：

```
SET NOCOUNT ON
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
WHILE 1=1 --循环语句
BEGIN
    BEGIN TRAN
```

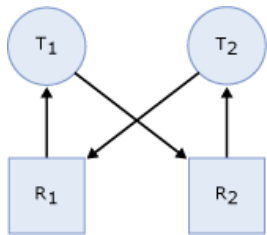


图 14- 39 死锁状态


```

UPDATE Track SET Circulation= 10 WHERE SongID ='S0002'
UPDATE Track SET Circulation= 9 WHERE SongID = 'S0001'
COMMIT TRAN
END

```

2) 在第二个查询窗口中执行如下代码:

```

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
WHILE 1=1
BEGIN
BEGIN TRAN
UPDATE Track SET Circulation= 9 WHERE SongID = 'S0001'
UPDATE Track SET Circulation= 10 WHERE SongID = 'S0002'
COMMIT TRAN
END

```

会发现, 在其中一个窗口中, 会出现“消息1205, 级别13, 状态51, 第10 行。事务(进程 ID 53) 与另一个进程被死锁在锁资源上, 并且已被选作死锁牺牲品。请重新运行该事务。”如图14- 40所示。

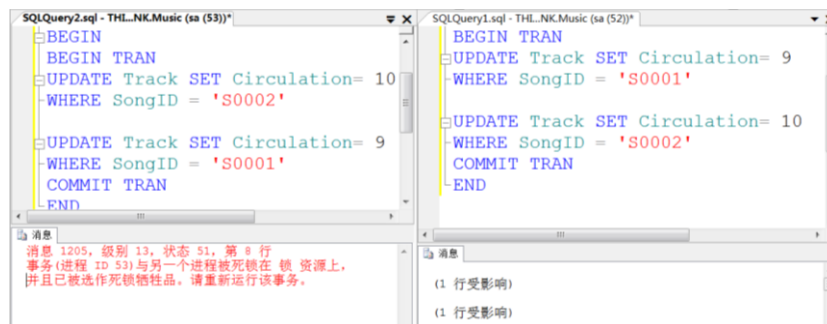


图 14- 40 死锁示例

由以上例子, 可以看出, 当出现死锁时, SQL Server 会选择一个事务作为死锁牺牲品。具体来说, SQL Server 对死锁的处理如下:

- 1) SQL Server 内部有一个锁监视器 (Lock Monitor) 线程执行死锁检查, 该线程定期搜索数据库引擎实例的所有任务, 默认时间间隔为 5 秒。当死锁发生时, 死锁检测间隔就会缩短, 最短为 100ms。

- 2) 锁监视器对特定线程启动死锁搜索时, 会标识线程正在等待的资源; 然后查找特定资源的所有者, 并递归地继续执行对那些线程的死锁搜索, 直到找到一个构成死锁条件的循环。

- 3) 检测到死锁后, 默认情况下, 数据库引擎选择运行回滚开销最小的事务的会话作为死锁牺牲品。此外, 用户也可以使用 SET DEADLOCK_PRIORITY 语句指定死锁情况下会话的优先级, 可以将其设置为范围 (-10 到 10) 间的任一整数值, 也可以将优先级设置为 LOW(等于-5)、NORMAL(等于0) 或 HIGH(等于5)。死锁优先级的默认设置为 NORMAL。如果两个会话的死锁优先级不同, 则会选择优先级较低的会话作为死锁牺牲品; 如果两个会话的死锁优先级相同, 则会选择回滚开销最低的事务的会话作为死锁牺牲品。如果死锁循环中会话的死锁优先级和开销都相同, 则会随机选择死锁牺牲品。

- 4) 死锁牺牲品选定后, 数据库引擎返回 1205 错误, 回滚死锁牺牲品的事务并释放该事务持有的所有锁, 使其他线程的事务可以请求资源并继续运行。

如果 Microsoft SQL Server 数据库引擎实例由于其他事务已拥有资源的冲突锁而无法将锁授予给某个事务, 则该事务被阻塞, 等待现有锁被释放。默认情况下, 没有强制的超时

期限。用户还可以为语句定制死锁超时。命令为：

```
SET LOCK_TIMEOUT timeout_period
```

其中，参数 `timeout_period` 表示等待死锁资源的最大时间,单位为毫秒。默认值为 -1，表示没有超时期限（即无限期等待）。值为 0 时表示根本不等待，一遇到锁就返回消息。

如果某个语句等待的时间超过设置时间，则被阻塞的语句自动取消，并会有错误消息“1222 (Lock request time-out period exceeded)”返回给应用程序。但是，SQL Server 不会回滚或取消任何包含语句的事务。因此，应用程序必须具有可以捕获错误消息 1222 的错误处理程序。如果应用程序不能捕获错误，则会在不知道事务中已有个别语句被取消的情况下继续运行，由于事务中后面的语句可能依赖于从未执行过的语句，因此会出现错误。实现捕获错误消息 1222 的错误处理程序后，应用程序可以处理超时情况，并采取补救措施，例如：自动重新提交被阻塞的语句或回滚整个事务。

14.3.4 SQL Server Profiler 查看死锁

使用 SQL Server Profiler，可以创建记录、重播和显示死锁事件的跟踪以进行分析。Profiler 查看死锁的设置步骤如下：

1) 在 SQL Server Profiler 的“文件”菜单上，单击“新建跟踪”，再连接到 SQL Server 实例。将出现“跟踪属性”对话框，如图 14-41 所示。

3) 在“常规”选项页中，可以在“跟踪属性”对话框的“跟踪名称”框中，键入跟踪的名称；选中“保存到文件”复选框以将跟踪捕获到文件中；也指定“设置最大文件大小”的值；可以选择“启用文件滚动更新”和“服务器处理跟踪数据”。选中“保存到表”复选框以将跟踪捕获到数据库表中。根据需要，可以单击“设置最大行数”，并指定值。可以选中“启用跟踪停止时间”复选框，再指定停止日期和时间。

4) 单击“事件选择”选项卡。在“事件”数据列中，展开“Locks”事件类别，然后选中“Deadlock Graph”复选框（图 14-42）。如果没有显示“Locks”事件类别，请选中“显示所有事件”以显示该类别。

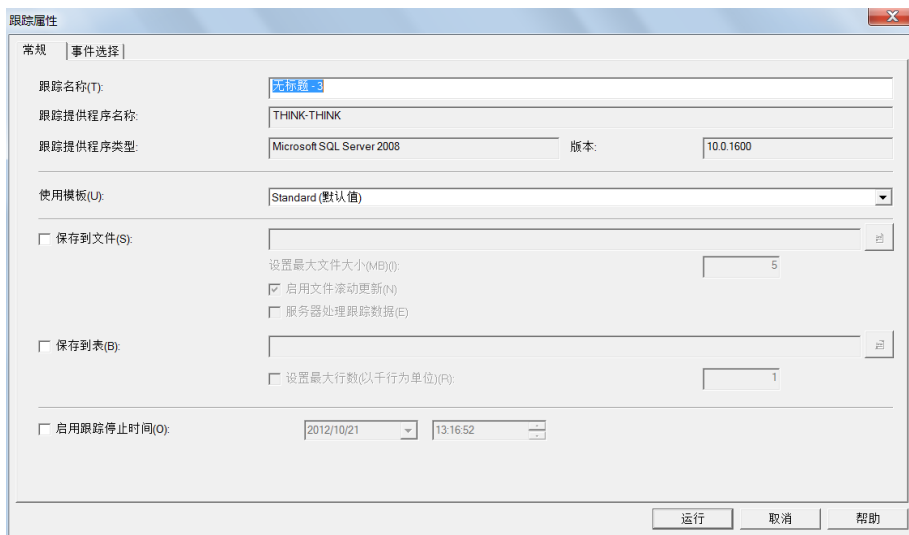


图 14-41 “常规”选项页

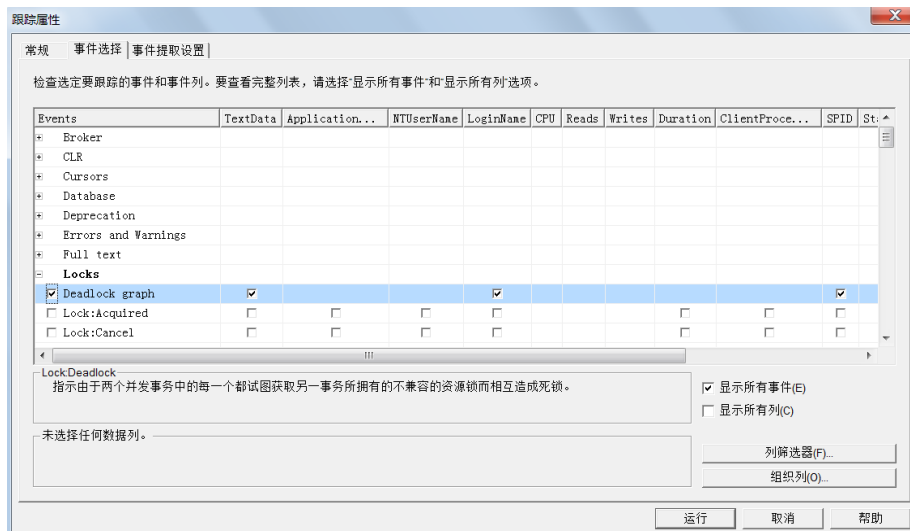


图 14-42 选择“Deadlock Graph”

SQL Server Profiler 中所跟踪到的死锁图形如图 14-43 所示。

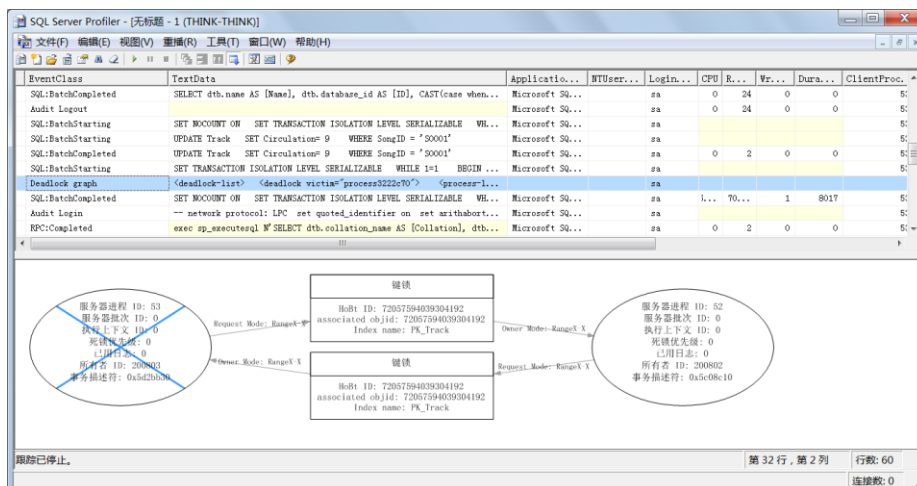


图 14-43 死锁图形

通过这个死锁图，可以更直观的看到死锁产生的主体和资源，死锁的牺牲品进程会被打叉号。并且鼠标移到主体上时，还可以显示造成死锁的语句（图 14-44）。

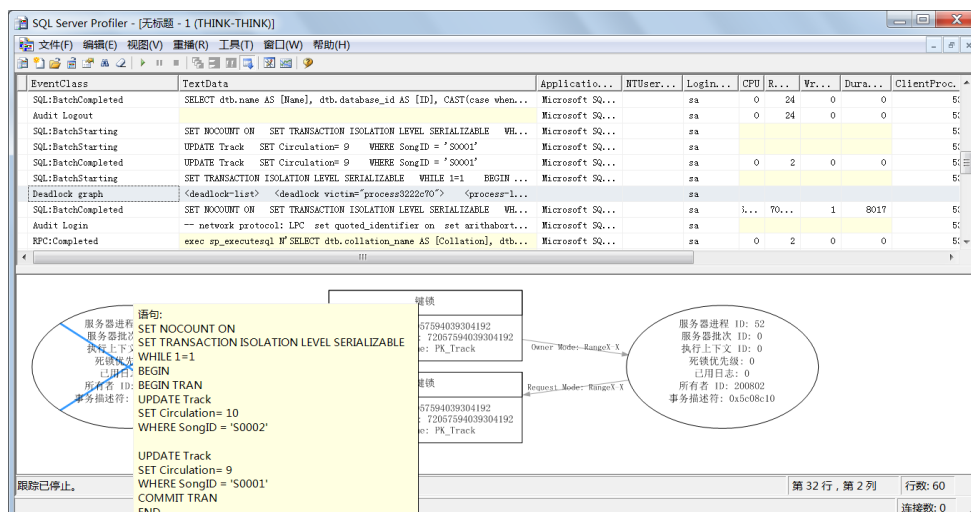


图 14-44 显示死锁语句

14.4 *扩展知识

14.4.1 SQL Server 事务日志与恢复模式

SQL Server 使用了预写式日志(Write-Ahead Logging (WAL)),即“先写日志,再写数据”的机制来确保了事务的原子性和持久性。实际上,不光是 SQL Server,基本上主流的关系数据库包括 oracle,mysql,db2 都使用了 WAL 技术。在事务日志中,数据变化被记录在一个连续的日志记录中,且每一个记录都有一个编号,叫做日志序列编号(Log Sequence Number, LSN)。

比如修改数据的操作,SQL Server 的处理如下:

- 1) 在 SQL Server 的缓冲区的日志中写入“Begin Tran”记录
- 2) 在 SQL Server 的缓冲区的日志页写入要修改的信息
- 3) 在 SQL Server 的缓冲区将要修改的数据写入数据页
- 4) 在 SQL Server 的缓冲区的日志中写入“Commit”记录
- 5) 将缓冲区的日志写入日志文件
- 6) 发送确认信息(ACK)到客户端(SMSS, ODBC 等)

可以看到,事务日志并不是一步步写入磁盘.而是首先写入缓冲区后,一次性写入日志到磁盘,这样既能在日志写入磁盘这块减少 IO,还能保证日志 LSN 的顺序。

在事务日志中,每一个日志记录都被存储在一个虚拟日志文件中。事务日志可以有任意多个虚拟日志文件,数量的多少取决于数据库引擎,而且每个虚拟日志文件的大小也不是固定的(图 14-45)。活动区间(active portion)的日志就是包含用户事务的区域,这区间就是完整恢复数据库所需要的。当更多的事务被创建时,活动区间的日志也会随着增长。

当 CheckPoint 被执行时,所有有变化的数据写到数据文件中,然后创建一个检查点记录(CheckPoint record)。以图 14-45 中的事务为例,当 CheckPoint 被执行时,由事务 1, 2, 3 所导致的变化将会被写到数据文件中。因为事务 3 没有被提交,所以活动区间日志的范围变成了从 LSN50 到 LSN52 之间(图 14-46)。

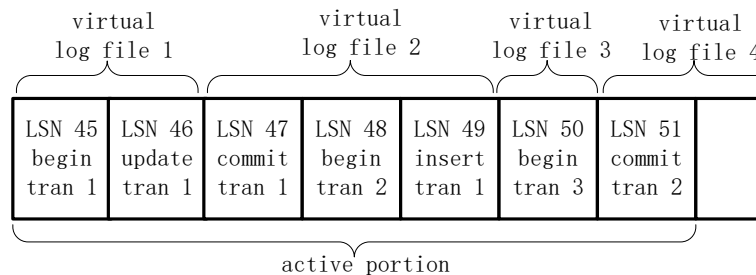


图 14- 45 虚拟日志文件和活动区间

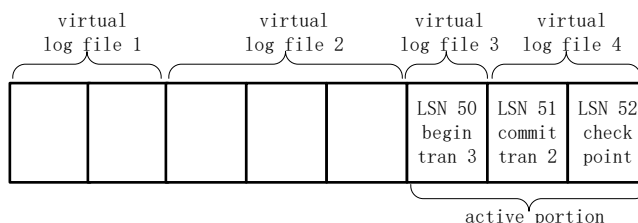


图 14- 46 CheckPoint 后的活动区间

如果使用简单恢复模型的话，那么 LSN45 到 LSN49 之间区域可以被重用，因为那些记录已经不再需要了。如果数据库运行在完整或是批量日志恢复模型下，那么从 LSN45 到 49 之间的区域将被删除(delete)，而且直到事务日志被备份后，这段区域的空间才会被重用。

如果当有新的事务被创建，在简单模式下，日志的起始空间将会被重用。如图 14- 47 所示，虚拟文件 1 和 2 作为可重用区域记录新的事务日志 LSN53、54 和 55。

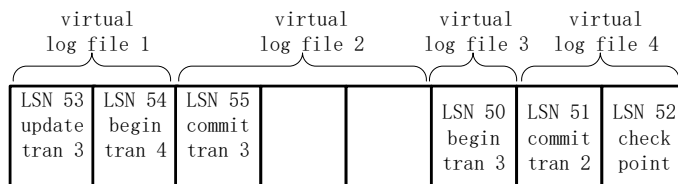


图 14- 47 简单模式的日志文件

如果是在完整或是大批量日志恢复模型下，事务日志的空间则会被扩展。图 14- 48 所示，新的日志 LSN53 到 55 将记录在扩展的虚拟文件 5 中。

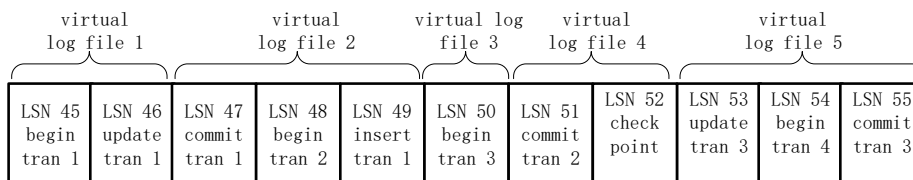


图 14- 48 完整模式或大批量日志恢复模型下的日志文件

由以上的分析，可以看出，日志文件的作用是用于数据库的恢复，它需要记录用户对用户的在各种变更操作。因此，对于变化较多的数据库，日志文件的增长很快，即日志文件的

空间需求较大。因此，用户在权衡空间需求和数据库的恢复要求以后，才能确定数据库的恢复模型。

完全恢复和大容量日志记录的恢复模型下，日志文件的空间只要允许会一直增长下去，直到 DBA 执行备份操作。备份操作将自动截断日志文件中不活动的部分，即已经完成的事务部分。

简单模型下，每发生一次检查点，都会自动截断日志文件中不活动的部分。这样简单模型下的数据库的日志文件就不会暴涨。但由于自动截断日志，对于用户数据库来说是危险的，因为丢失了日志。

习题 14

1. SQL Server 的用户或角色分为二级：一级为服务器级用户或角色；另一级为_____。
2. SQL Server 有两种安全认证模式，即 Windows 安全认证模式和_____。
3. 简述 SQL Server 的安全机制。
4. 什么是 Windows 身份验证？什么是混合身份验证？
5. 一个数据库用户的权限可以通过几种途径获得？
6. 简述常用的备份类型。
7. 简述 SQL Server 的三种恢复模式。
8. 在 SQL Server 中，表级的操作权限有哪些？
9. 登录账号和用户账号的联系、区别是什么？
10. 什么是角色？角色和用户有什么关系？当一个用户被添加到某一角色中后，其权限发生怎样的变化？
11. SQL Server 的权限有哪几种类型？
12. 备份设备有哪些？
13. 完全备份、差异备份、日志备份各有什么特点，以你所知的一台服务器为例，设计一种备份方案。
14. 某企业的数据库每周日晚 12 点进行一次全库备份，每晚 12 点进行一次差异备份，每小时进行一次日志备份，数据库在 2012-8-23 3:30 崩溃，应如何将其恢复使数据库损失最小。
15. SQL Server 如何处理死锁？
16. SQL Server 的隔离级别有几种？
17. SQL Server 有哪几种锁？