

11.2 表

11.2.1 表的概述

数据库中包含许多对象,其中最重要的就是表,数据库中的数据或信息都是存储在表中。表中数据的组织形式是行和列的组合,其中每一行代表一条记录,例如,公司运送的每个部件在部件表中均占一行;每一列代表由一个属性,例如,一个部件表有 ID 列、颜色列和重量列。

说明: 在表中不必对行进行排序。如果需要对结果集中的数据排序,需要在 select 查询语句中显式指定排序。

11.2.2 表的分类

SQL Server 中,可以从两种角度来对表进行分类。

1. 按照数据存储的时间分类:

1) 永久表

表建立后,除非人工删除,否则一直保存。在 master、model、msdb 和用户数据库中建立的表都是永久表。

2) 临时表

临时表的数据只在数据库运行期间临时保存数据,临时表存储在 tempdb 中。

2. 按照表的用途来分类

1) 用户表

用户创建的,用于开发各种数据库应用系统的表。

2) 系统表

维护 SQL Server 服务器和数据库正常工作的数据表。每个数据库都会建立很多系统表,这些表不允许用户进行更改,只能由 DBMS 自行维护。

3) 临时表

SQL Server 的临时表有两种类型:本地临时表和全局临时表。

本地临时表只对于创建者是可见的。当用户与 SQL Server 实例断开连接后,将删除本地临时表。

全局临时表在创建后对任何用户和任何连接都是可见的,当引用该表的所有用户都与 SQL Server 实例断开连接后,将删除全局临时表。

如果 SQL Server 服务器关闭,则所有的本地和全局临时表都被清空、关闭。

从表名称上来看,本地临时表的名称前面有一个“#”符号;而全局临时表的名称前面有两个“##”符号。

说明: 临时表的作用——当对数据库执行大数据量的排序等操作时,要产生大量的中间运算结果,因此需要消耗大量的内存资源。如果内存资源不够用,那么 SQL Server 将在临时数据库 tempdb 中创建临时表用于存放这些中间结果。

4) 分区表

当一个表中的数据量过于庞大时,可以使用分区表。分区表是将数据水平划分为多个单

元的表，这些单元可以分布到数据库中的多个文件组中。在维护整个集合的完整性时，使用分区可以快速而有效地访问或管理数据子集，从而使大型表或索引更易于管理。

11.2.3 创建表

创建表的实质就是定义表的结构以及约束等属性，因此创建表需要使用不同的数据库对象，包括数据类型、约束、默认值和索引等。SQL Server 中，表必须建在某一数据库中，不能单独存在，也不能以操作系统文件形式存在。表名和列名要有意义，不能使用系统关键字且必须遵守标识符的规则；列名在一个表中必须是唯一的。

1. 图形界面下创建表

在 SSMS 中，用户可以通过图形界面快捷地创建表。下面以 Music 中的 Singers 表为例说明如何在图形化界面下创建表。

表 11- 1 Singer 表

字段名称	字段类型	字段宽度	是否 NULL	说明
SingerID	Char	10	NOT NULL	歌手编号
Name	Varchar	8	NOT NULL	歌手名称
Gender	Varchar	2	NULL	性别
Birth	Datetime		NULL	出生日期
Nation	Varchar	20	NULL	籍贯

创建步骤如下：

1) 在“对象资源管理器”中，展开 Music 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，出现“表设计器”界面，如图 11- 1 所示。

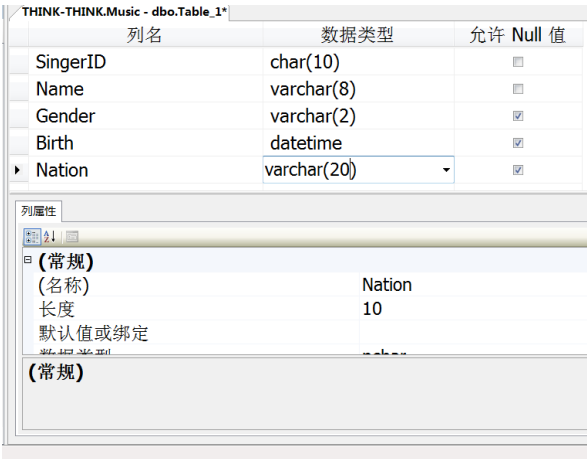


图 11- 1 表设计器界面

2) 在“表设计器”页面中，可以定义列名、数据类型和设置是否允许为空。

- (1) 在“列名”单元格中，输入表的列名，如 SingerID。
- (2) 在“数据类型”下拉框中，选择对应的数据类型（常见的数据类型见第九章数据类型），并设置宽度，如 char(10)。
- (3) 在“允许 NULL 值”列下设置各列是否允许为空。打“√”的表示允许为空，否则表示不允许为空值。Singer 表中各列的具体设置见图 11- 1。

3) 各列的设置完成后，单击工具栏的“保存”按钮，将出现“选择名称”对话框，输入表名后就可以保存该表了。

表创建完成后，展开“Music”数据库下的“表”节点，就可以看到刚创建的 Singer 表，如图 11-2 所示。

说明：

创建字符数据库类型的列时，如果用户没有定义字符宽度，那么系统默认 char（10），varchar（50）。

以上创建表的操作对应的 SQL 语句为：

```
CREATE TABLE Singer(  
    SingerID char(10) NOT NULL,  
    Name varchar(50) NOT NULL,  
    Gender varchar(2) NULL,  
    Birth datetime NULL,  
    Nation varchar(20) NULL  
)
```

用户也可在查询编辑器中输入以上 CREATE TABLE 建表语句来创建 Singer 表。CREATE TABLE 语句的具体语法和说明见第四章 4.3 节。

例 11- 1：在 SSMS 中建立一个 Student 表，结构如下：

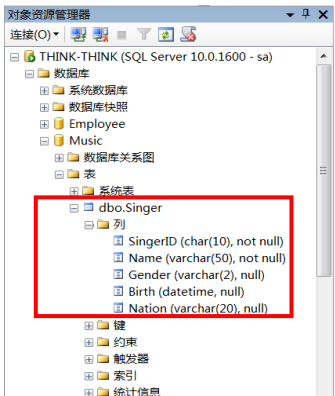


图 11- 2 Singer 表

表 11- 2 Student 表的表结构

字段名称	字段类型	字段宽度	是否 NULL	说明
sno	int		NOT NULL	学号
sname	Varchar	10	NOT NULL	姓名
sex	Varchar	2	NULL	性别
hometown	Datetime		NULL	籍贯
introduction	Varchar	400	NULL	简介
birthdate	Datetime		NULL	出生日期

结果如图 11- 3 所示。

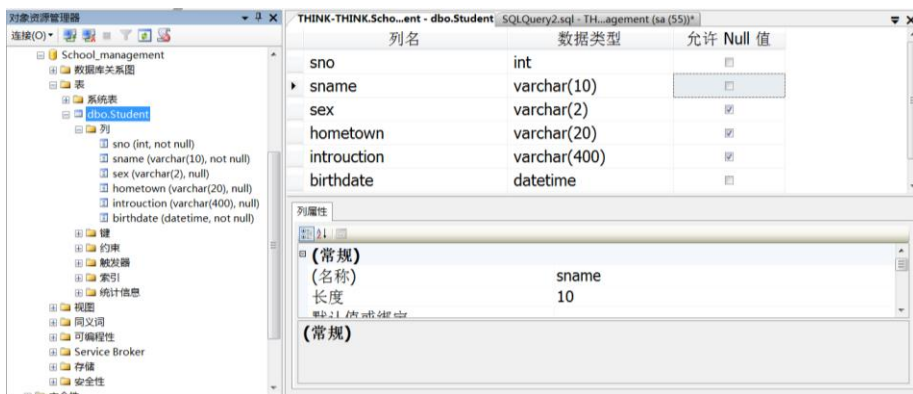


图 11- 3 Student 表

2. 列属性说明与设置

以上创建的表中，我们只定义了列名、数据类型和是否 NULL 值。要设计好一张表，还有更多的问题需要考虑，比如哪些字段的内容不能重复，哪些内容是必须的等等，这就需要掌握列的更多属性的设置。

1) NULL 的含义

NULL（空值）表示数值未知或将在以后添加的数据。比如，“Singer”表中“Birth”字段，在不知道的情况下不填写，这就是空值。在“查询编辑器”中查看表数据时，空值在结果中显示为 NULL。用户也可显式的输入 NULL，不管这一列是何种数据类型。

注意： 输入 NULL 时，NULL 不能放在引号内，否则会被解释为字符串而不是空值。

NULL 不同于空白、0 或长度为零的字符串。没有两个相等的空值。两个 NULL 值相比或 NULL 与其他数值相比，返回是未知，因为每个空值均为未知。

NOT NULL（不允许空值）表示数据列不允许空值。如果一列设为 NOT NULL，那么在向表中写数据时必须输入一个值，否则该行不被接收，有助于维护数据完整性。

注意： 定义了 PRIMARY KEY 约束或 IDENTITY 属性的列不允许空值。

2) 指定列的默认值

默认值是指当向表中插入数据时，如果用户没有明确给出某列的值，SQL Server 自动为该列添加的值。当某一列被设置为默认值约束，该列的取值可以输入，也可以不输入，不输入时，取值为默认值。

例如，如果“Student”表中大部分学生来自江苏，那么，可以为“Hometown”一列设置默认值为江苏。指定默认值有两种方法：可以在表设计器中指定；也可以在使用 CREATE TABLE 语句及 ALTER TABLE 语句时使用 DEFAULT 关键字定义。

（1）使用表设计器指定列的默认值

具体设置步骤如下：

① 在“表设计器”中选中要设置默认值的列。

② 展开在“表设计器”页面下方“列属性”中的“常规”节点，在“默认值或绑定”后面的文本框中输入其默认值即可。设置默认值时，需要注意该列的数据类型，比如 Hometown 是字符型，因此在默认值为一个字符型常量（加单引号）‘江苏’。如图 11- 4 所

示。

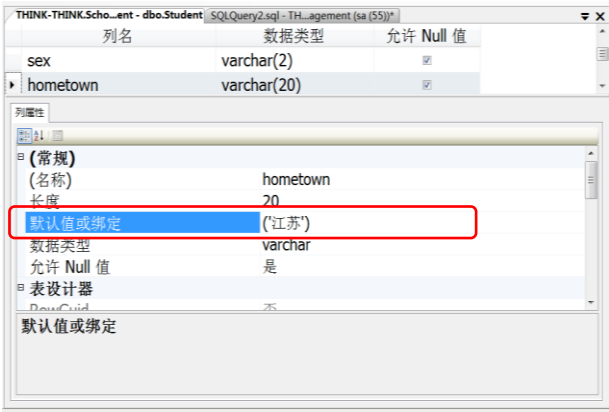


图 11-4 设置默认值

③ 以上设置的默认值，实际上是为表加了一个 **DEFAULT**（默认值）约束。在“对象资源管理器”中“表”→“约束”下，会添加以 **DF** 为前缀的一个约束，代表是 **DEFAULT** 约束，这个名字是系统自动给定的，如图 11-5 所示。

（2）使用 **DEFAULT** 关键字

可以在 **CREATE TABLE** 语句及 **ALTER TABLE** 语句中使用 **DEFAULT** 关键字指定列的默认值。语法格式为：

```
DEFAULT <default value>
```

在关键字 **DEFAULT** 的后面，必须指定其默认值。

以上对 **Hometown** 设置默认值，可以使用以下语句：

```
create table Student
(sno int not null,
sname varchar(10) not null,
sex varchar(2),
hometown varchar(20) null default('江苏'), /*设置默认值*/
introuction varchar(50) null,
birthdate datetime not null )
```

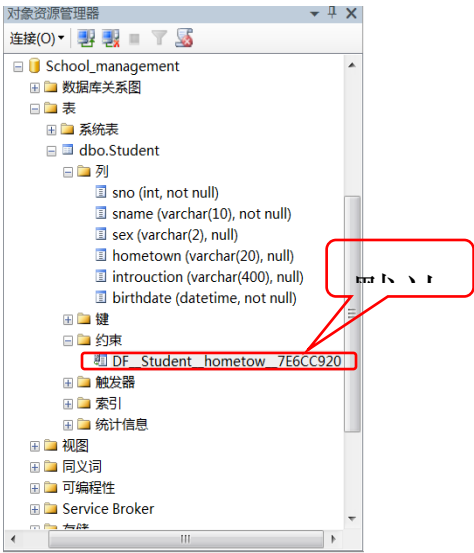


图 11-5 默认值约束

3) **IDENTITY** 属性

使用 **IDENTITY** 关键字定义的字段又称标识字段。开发人员可以为标识字段指定标识种子（**Identity Seed** 属性）和标识增量（**Identity Increment** 属性），系统按照给定的种子和增量自动生成标识号，该标识号是唯一的。

使用 **IDENTITY** 属性时要注意：

- 说明：
- 一个表只能有一个使用 **IDENTITY** 属性定义的列；
 - **IDENTITY** 只适用于 decimal（小数部分为 0）、int、numeric（小数部分为 0）、smallint、

bigint 或 tinyint 数据类型;

- 标识种子和增量的默认值均为 1;
- 标识符列不能允许为空值, 也不能包含 DEFAULT 定义或对象;
- 标识符列不能更新;
- 如果在经常进行删除操作的表中存在标识符列, 那么标识值之间可能会出现断缺。

在 SQL Server 中, 定义标识字段有两种方法: 可以使用表设计器定义标识字段; 也可以在 CREATE TABLE 语句及 ALTER TABLE 语句中使用 IDENTITY 关键字定义。

例如, Student 表中的 sno (学号) 是 int 型, 因为学生的学号不允许相同且一般都是连续的, 所以可将 sno 设置为标识字段。以下以此为例进行说明。

(1) 使用表设计器定义标识字段

在“表设计器”中选择 sno 字段, 展开表设计器下面“列属性”中的“标识规范”节点, 在“是标识”后的下拉菜单中选择“是”选项。标识增量和种子按需要更改即可。这里将学号设置为从 201102001 开始, 每次增加 1。如图 11-6 所示。

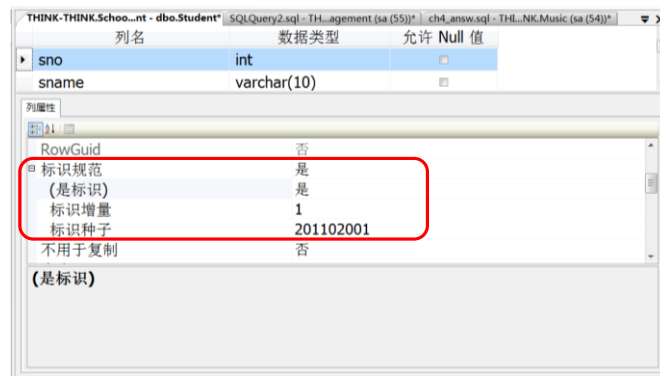


图 11-6 设置标识字段

(2) 使用 IDENTITY 关键字

在 CREATE TABLE 语句及 ALTER TABLE 语句中使用 IDENTITY 关键字可以指定标识字段。语法格式如下:

```
IDENTITY (seed, increment)
```

说明: seed 表示标识种子, increment 表示标识增量。
必须两者同时指定或同时不指定。如果两者都未指定, 则取默认值 (1,1)。

对“sno”设置 IDENTITY, 可以使用以下语句:

```
create table Student
(
    sno int not null identity(201102001,1),    /*设置 IDENTITY*/
    sname varchar(50) not null,
    sex varchar(2),
    hometown varchar(50) null default('江苏'), /*设置默认值*/
    introuction varchar(50) null,
```

```
birthdate datetime not null )
```

例 11- 2：在 “Music” 数据库 中建立 Songs（歌曲）表和 Track（曲目）表。结构如表 11- 3、表 11- 4 所示。

表 11- 3 Songs 表结构

字段名称	字段类型	字段宽度	说明	要求
SongID	Char	10	歌曲编号	非空
Name	Varchar	50	歌曲名字	
Lyricist	Varchar	20	词作者	
Composer	Varchar	20	曲作者	
Lang	Varchar	16	语言类别	默认 “中文”

表 11- 4 Track 表结构

字段名称	字段类型	字段宽度	说明	要求
SongID	Char	10	歌曲编号	非空
SingerID	Char	10	歌手编号	非空
Album	Varchar	50	专辑	
Style	Varchar	20	曲风类别	默认 “流行”
Circulation	INT		发行量	
PubYear	INT		发行时间	

Songs 的创建如图 11- 8、图 11- 10 所示。

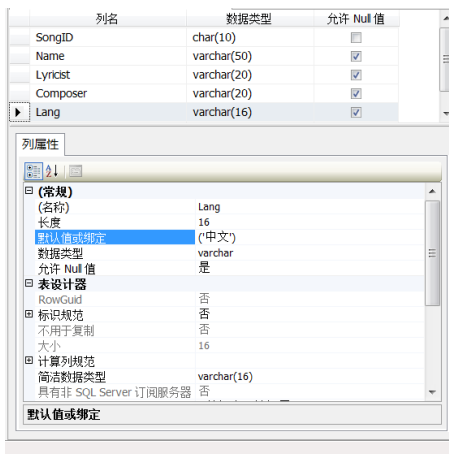


图 11-7 创建 Songs 表

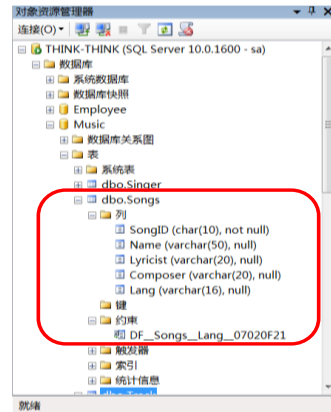


图 11-8 Songs 表

Songs 的建表语句为:

```
CREATE TABLE Songs
( SongID CHAR(10) not null, Name VARCHAR(50),
  Lyricist VARCHAR(20), Composer VARCHAR(20),
  Lang VARCHAR(16) default('中文') )
```

Track 的创建如图 11-9 和图 11-10 所示。

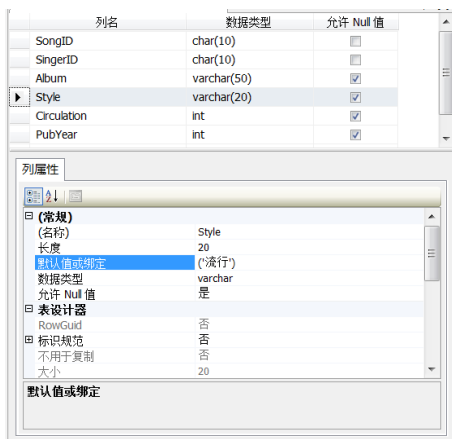


图 11-9 创建 Track 表

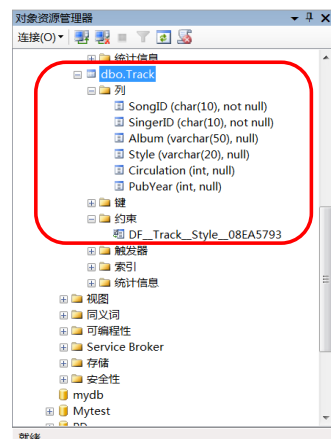


图 11-10 Track 表

Track 的建表语句为:

```
CREATE TABLE Track
(SongID CHAR(10) not null,
 SingerID CHAR(10) not null,
 Album VARCHAR(50),
 Style VARCHAR(20) default('流行'),
 Circulation INTEGER,
 PubYear INTEGER)
```


11.2.4 数据完整性的实现

数据库完整性（Database Integrity）是指数据库中数据的正确性和相容性。在第三章 3.4 节中，我们介绍过关系有三类完整性：实体完整性、参照完整性和用户定义完整性。在 SQL Server 中，这三类完整性是由各种各样的完整性约束来保证的，因此可以说数据库完整性设计就是数据库完整性约束的设计。

SQL Server 为了保证数据完整性共提供了以下 6 种约束。

- 1. 非空（NOT NULL）约束
- 2. 主键（PRIMARY KEY）约束
- 3. 外键（FOREIGN KEY）约束
- 4. 唯一性（UNIQUE）约束
- 5. 检查（CHECK）约束
- 6. 默认（DEFAULT）约束

这些约束通过限制列中的数据、行中的数据和表之间数据来保证数据完整性。表 11- 5 列出 SQL Server 的约束所能实现的完整性功能。

表 11- 5 约束类型和完整性功能

完整性类型	约束类型	完整性功能描述
实体完整性	PRIMARY KEY(主键)	指定主键，确保主键不重复，不允许主键为空值
	UNIQUE(惟一值)	指出数据应具有惟一值，防止出现冗余
用户定义完整性	CHECK(检查)	指定某个列可以接受值的范围，或指定数据应满足的条件
参照完整性	FOREIGN KEY(外键)	定义外键、被参照表和其主键

在 11.2.3 节创建表的过程中，已经讲了非空约束和默认值约束的含义及设置方法。以下主要介绍其余 4 种约束。


1. 实体完整性及其实现

1) PRIMARY KEY 约束

表通常具有包含唯一标识表中每一行值的一列或一组列，这样的一列或多列称为表的主键（PRIMARY KEY）。换句话说，表主键就是用来约束数据表中不能存在相同的两行数据。而且，PRIMARY KEY 约束的列应具有确定的数据，不能接受空值。在管理数据时，应确保每一个数据表都拥有一个唯一的主键，从而实现实体的完整性。

	<ul style="list-style-type: none">● 一个表只能有一个 PRIMARY KEY 约束，并且 PRIMARY KEY 约束中的列不能接受空值。
说明：	<ul style="list-style-type: none">● 如果对多列定义了 PRIMARY KEY 约束，则一列中的值可能会重复，但来自 PRIMARY KEY 约束定义中所有列的任何值组合必须唯一。● 当定义主键约束时，SQL Server 在主键列上建立唯一聚簇索引，以加快查询速度。

（1）单一主键约束的实现

以 Songs 表设置主键为例进行说明。步骤如下：在“表设计器”中在主键的列（如 SongID 列）上右击，在弹出的快捷菜单中选择“设置主键”命令，如图 11- 11 所示，则该列被设置为主键。“保存”后，列名左边出现主键图标，并且在“表”→“键”下出现以“PK_”开头的一个主键约束，如图 11- 12 所示。

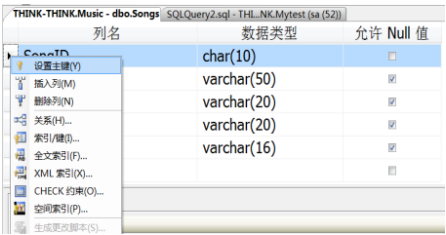


图 11- 11 “设置主键”

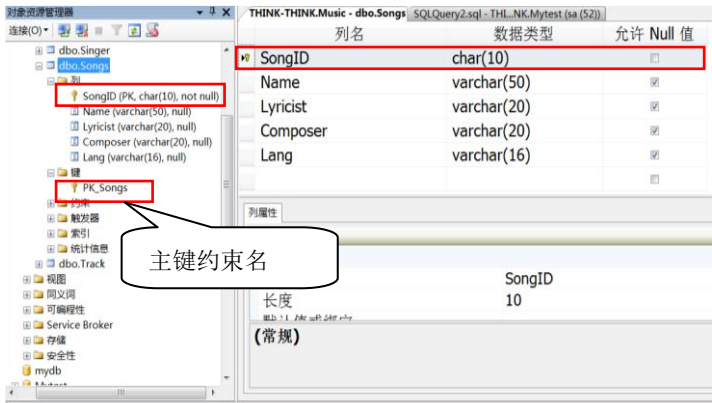



图 11- 12 设置主键后的表

设置主键也可在 CREATE TABLE 或 ALTER TABLE 中使用 PRIMARY KEY 关键字。对应的 SQL 语句：

```
CREATE TABLE Songs
(
    SongID CHAR(10) not null PRIMARY KEY, /*主键约束*/,
    Name VARCHAR(50),
    Lyricist VARCHAR(20),
    Composer VARCHAR(20),
    Lang VARCHAR(16) default('中文')) /*DEFAULT 约束 */
```

(2) 组合主键的设置

有的表的主键有多列组成，比如 Track 表，该表中任何一个单独的列都不能作为主键。因为一个歌手可能演唱多首歌，而同一首歌又可能被多名歌手演唱。因此，只能 SingerID 和 SongID 组合在一起作为主键。

Track 中组合主键的设置操作为：选中作为主键的两列（按住 Shift 键，用鼠标左键分别单击这两列左边的列标志块，使列所在行高亮显示）→单击右键→选择快捷菜单中的“设置主键”（如图 11- 13），则主键列的左边均出现主键图标。保存并刷新表，在“表”→“列”中可看到主键标志；在“键”中看到主键约束名（如图 11-15）。

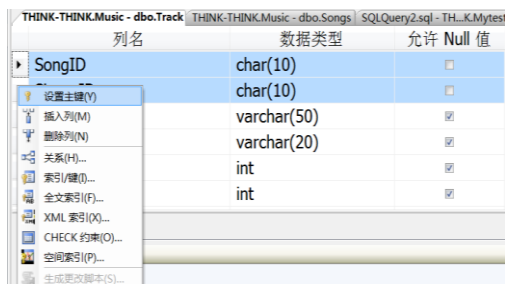


图 11-13 设置组合主键

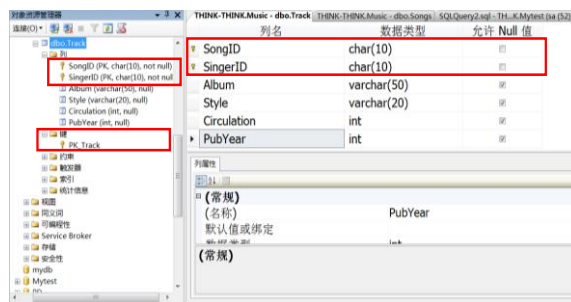


图 11-14 设置主键后的 Track 表

对应的 SQL 语句为：

```
CREATE TABLE Track
(
    SongID CHAR(10) not null,
    SingerID CHAR(10) not null,
    Album VARCHAR(50),
    Style VARCHAR(20) default('流行'), /*DEFAULT 约束*/
    Circulation INTEGER, PubYear INTEGER,
    PRIMARY KEY(SongID, SingerID) /*主键约束，该约束是表级完整性约束*/
)
```

2) UNIQUE 约束

唯一约束被用来增强非主键列的唯一性。设置了唯一约束的列不能有重复值，可以但最多允许一个 NULL 值。每个 UNIQUE 约束都生成一个唯一索引,所以唯一约束和唯一索引的创建方法相同。


- 主键约束与唯一约束的异同：

两者都要求约束的列不能有重复值；

说明：主键约束要求主键列不能为空；唯一约束允许有空值，但只允许一个 NULL 值。

- 在一个表上可以定义多个 UNIQUE 约束；
- 可以在多个列上定义一个 UNIQUE 约束，表示这些列组合起来不能有重复值。

假设要为 Singer 中的 Name 设置唯一约束，可以如下操作：在“表设计器”中右键打开快捷菜单，选择“索引/键”（如图 11-15），即可打开“索引/键”设置窗口。该窗口列出了该表中已经建好的索引/键（如主键 PK_Singer），点击“添加”按钮，在“常规”列表下进行设置：

(1) 单击“列”最右侧的  按钮，如图 11-16 所示，在打开的“索引/键”对话框中，

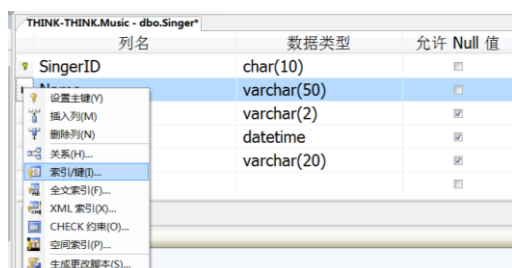


图 11-15 选择“索引/键”

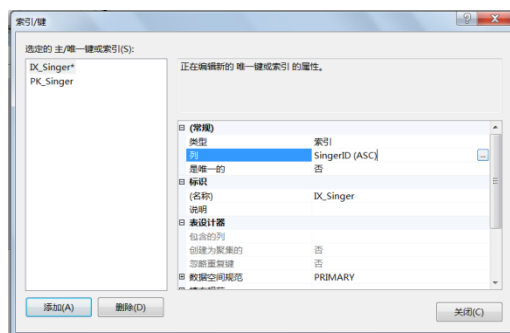


图 11-16 “索引/键”对话框

设置唯一约束的列，在“列名”下拉列表中选择 Name 项，单击“确定”按钮，如图 11-17 所示。

(2) 在“索引/键”对话框中，选择“是唯一的”项，单击其右边的组合框下三角按钮，选择“是”，如图 11-18 所示。

(3) 可以在“名称”行中修改约束的名称，本例采用系统默认名称 IX_Singer。

“保存”并刷新 Singer 表，在“索引”节点下出现唯一约束标志，如图 11-19 所示。

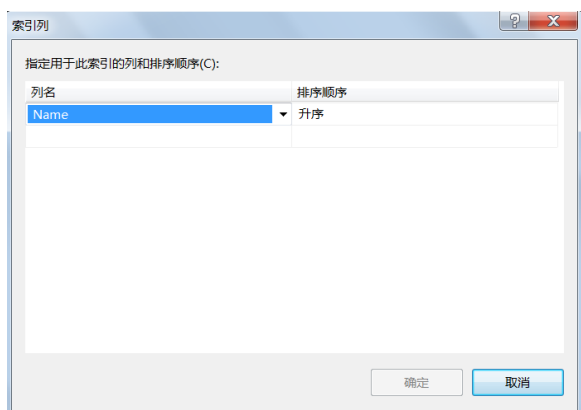


图 11-17 选择“索引/键”

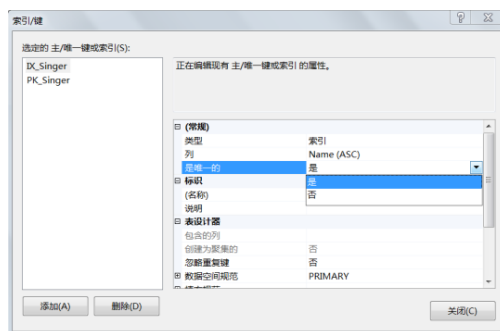


图 11-18 唯一性设置

UNIQUE 约束也可以在 CREATE TABLE 或 ALTER TABLE 中使用 UNIQUE 关键字进行设置，代码如下：

```
CREATE TABLE Singer(  
-- 主键约束  
SingerID char(10) NOT NULL PRIMARY KEY,  
-- 唯一约束  
Name varchar(50) NOT NULL UNIQUE,  
Gender varchar(2) NULL, Birth datetime NULL,  
Nation varchar(20) NULL)
```

2. 用户定义完整性及实现

用户定义的完整性是通过检查约束来实现的。检查约束用于限制列的取值在指定范围内，这样使得数据库中存放的值都是有意义的。当一列被设置了 CHECK 约束，该列的取值必须符合检查约束所设置的条件。约束条件应是逻辑表达式，多个条件可以用 AND 或 OR 组合。

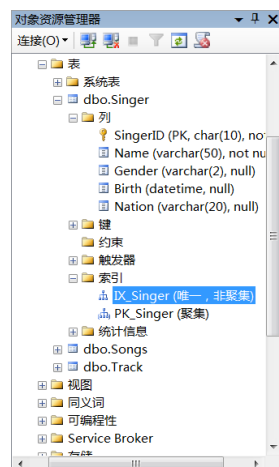


图 11-19 唯一约束

- 1) 列级的约束只能引用被约束的列上的值，一个列上可以有任意多个 CHECK 约束，多个 CHECK 约束按创建顺序进行验证。
- 2) 可以在表上建立一个可以在多个列上使用的 CHECK 约束，但是表级约束只能引用同一表中的列。
- 3) 不能在 text、ntext 或 image 列上定义 CHECK 约束。
- 4) CHECK 可以使用 IN、LIKE、BETWEEN 关键字。

假设 Singer 表中 Gender（性别）列取值为“男”或“女”，则可为其设置 CHECK 约束。方法如下：右键点击“表设计器”→选择快捷菜单中“CHECK 约束”，在弹出的 CHECK 约束设置窗口中单击“添加”，选择“表达式”行，点击右侧的 [...] 按钮（图 11-21），在弹出的“约束表达式”输入框中输入表达式（本例输入 Gender in(‘男’;‘女’))，点击“确定”按钮（图 11-21）；关闭 CHECK 约束设置窗口后，按“保存”按钮，并刷新表，在“表”→“约束”节点下出现 CHECK 约束名（本例为 CK_Singer）。

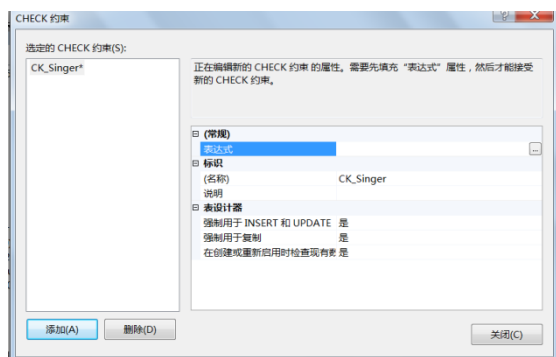


图 11- 20 CHECK 约束设置窗口

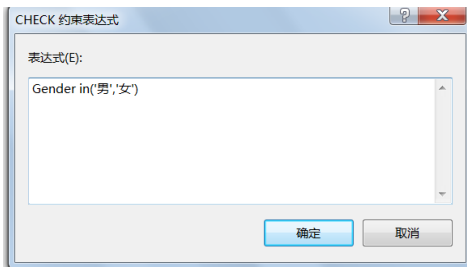


图 11-21 CHECK 约束表达式

CHECK 约束也可以在 CREATE TABLE 或 ALTER TABLE 中使用 CHECK 关键字进行设置。代码如下所示：

```
CREATE TABLE Singer (
    SingerID char(10) NOT NULL PRIMARY KEY,          /* 主键约束 */
    Name varchar(50) NOT NULL
    UNIQUE,                                           /* 唯一约束 */
    Gender varchar(2) NULL CHECK(Gender in('男','女')), /* 检查约束 */
    Birth datetime NULL, Nation varchar(20) NULL)
```

例 11- 3：表级的 CHECK 约束示例代码如下：

```
CREATE TABLE 工作表
(工作编号 char(8) primary key,
最低工资 int, 最高工资 int,
check (最低工资<最高工资) )    --限制最低工资小于最高工资
```

3. 参照完整性及实现

参照完整性主要通过主键与外键的联系来实现的。主键所在的表称为主表，外键所在的表称为子表。外键的取值参照主键的取值，即外键列的值有两种可能：一是等于主键的某个值；二是为空值，否则将返回违反外键约束的错误信息。

通过设置参照完整性，SQL Server 将禁止用户在外键列中引用主键中不存在的值；在默

认级联规则下，禁止修改主键中的值而不修改外键中的值；禁止删除被外键引用的主键值。

SQL Server 通过 FOREIGN KEY（外键）约束来实现参照完整性。一张表可含有多个 FOREIGN KEY 约束。

- FOREIGN KEY 约束只能引用同一个服务器上的同一数据库中的表。跨数据库的参照完整性必须通过触发器实现。
 - FOREIGN KEY 可引用同一个表中的其他列，这称为自引用。
- 说明：
- FOREIGN KEY 约束并不仅仅可以与另一表的 PRIMARY KEY 约束相链接，它还可以定义为引用另一表的 UNIQUE 约束。
 - 不能更改定义了 FOREIGN KEY 约束的列的长度，因为外键列和主键列的数据类型和长度需一致。

例如，“Singer”表中的“SingerID”是主键，“Track”表中的“SingerID”要参照“Singer”表中的“SingerID”，因此需设置“Track”表中的“SingerID”为外键。“Singer”表是主表，“Track”表是子表。设置外键有三种方法：

1) 利用表设计器设置参照完整性

进入子表（本例为“Track”表）表设计器，右键选中“关系”选项，进入“外键关系”对话框，如图 11-22 所示。点击“添加”按钮，选择“表和列规范”行，点击右侧的 [...] 按钮，进入“表和列”选择界面，在此设置主键表、主键列以及外键列，单击“确定”，完成设置。保存并刷新“Track”表，在“表”→“键”下出现外键标识（本例为 FK_Track_Singer）。

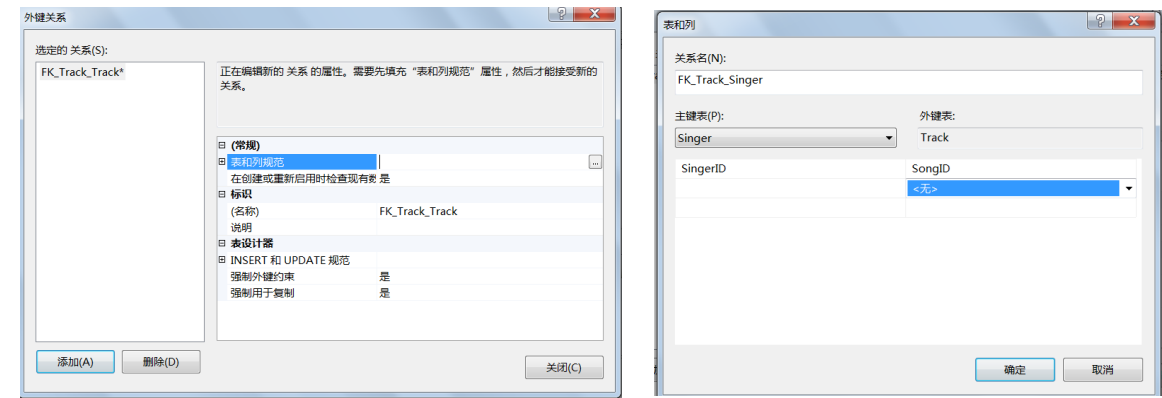


图 11-22 设置参照完整性

2) 利用数据库关系图设计器设置参照完整性

“Music”数据库“Track”表中的“SongID”需参照“Songs”表中“SongID”字段，以此为例来介绍在数据库关系图设计器中设置此联系。步骤如下：

（1）在“对象资源管理器”中，依次展开“数据库”→“Music”节点，右击“数据库关系图”，选择“新建数据库关系图”选项。

(2) 在打开的“添加表”对话框中选择要添加到关系图中的表(如图 11-23 所示, 本例选择“Songs”和“Track”表), 单击“添加”并“关闭”按钮。

(3) 在关系图设计器中, 选中主键的主键列(本例为“Songs”表的“SongID”列), 拖动鼠标直至外键表的外键列(本例为“Track”表的“SongID”列)。松开鼠标, 此时将弹出“外键关系”对话框(图 11-24)和“表和列”对话框(图 11-25)。

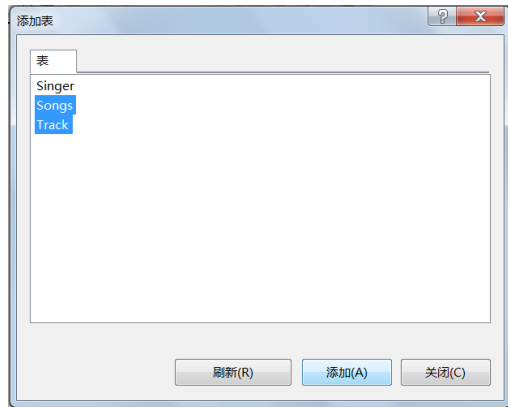


图 11-23 “添加表”对话框

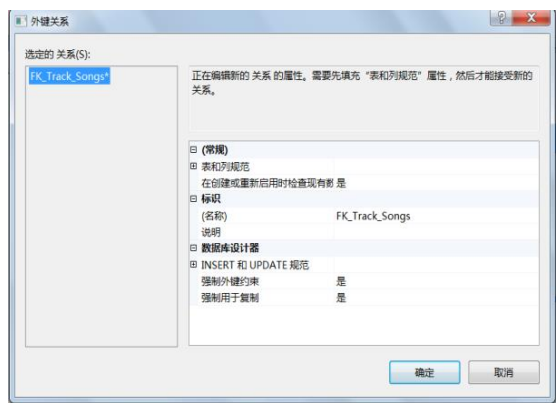


图 11-24 “外键关系”对话框

(4) 在“表和列”对话框中, 设置主键表、主键列和外键列, 如图 11-26 所示。单击“确定”按钮返回到“外键关系”对话框, 单击“确定”按钮即可完成参照完整性设置。

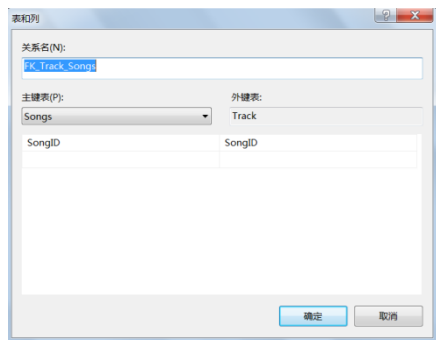


图 11-25 “表和列”对话框

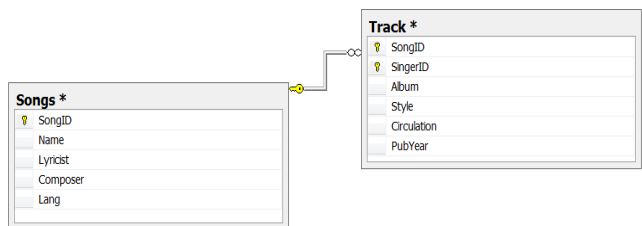


图 11-26 Track 与 Songs 的参照完整性

设置完毕后, 在“数据关系图”设计器中有参照完整性设置的表之间会有一条连线相连。连线有方向, 钥匙端指向的是主表, 圆环端指向的是子表, 如图 11-26 所示。

3) 利用 FOREIGN KEY.....REFERENCES 关键字设置参照完整性

参照完整性也可以在 CREATE TABLE 或 ALTER TABLE 中使用 FOREIGN KEY.....REFERENCES 关键字进行设置。比如以下代码是在 CREATE TABLE 语句中设置了 SongID 和 SingerID 为外键。

```
CREATE TABLE Track
(/ *外键约束*/
SongID CHAR(10) not null FOREIGN KEY REFERENCES Songs (SongID),
/*外键约束*/
SingerID CHAR(10) not null
FOREIGN KEY REFERENCES Singers (SingerID),
```

```
Album VARCHAR(50),
Style VARCHAR(20) default('流行'),           /*DEFAULT 约束*/
Circulation INTEGER, PubYear INTEGER,
PRIMARY KEY(SongID, SingerID) )              /*主键约束*/
```

4) 定义级联规则—— ON UPDATE 和 ON DELETE

SQL Server 2008 中，可以在 CREATE TABLE 语句和 ALTER TABLE 语句的 REFERENCES 子句中使用 ON DELETE 子句和 ON UPDATE 子句来定义当用户试图删除或更新现有外键指向的主键值时，SQL Server 执行的操作。

语法为：

```
REFERENCES referenced_table_name [ ( ref_column ) ]
[ ON DELETE { NO ACTION → CASCADE → SET NULL → SET DEFAULT } ]
[ ON UPDATE { NO ACTION → CASCADE → SET NULL → SET DEFAULT } ]
```

如果没有指定 ON DELETE 或 ON UPDATE，则默认为 NO ACTION。

对应的级联操作参数说明见表 11- 6：

表 11- 6 级联操作参数说明

参数	说明
NO ACTION	指定如果试图删除(或更新)某一行，而该行的键被其他表的现有行中的外键所引用，则产生错误并回滚 DELETE（或 UPDATE）语句。
CASCADE（级联）	指定如果试图删除(或更新)某一行，而该行的键被其他表的现有行中的外键所引用，则也将删除(或更新)所有包含那些外键的行。
SET NULL	指定如果试图删除(或更新)某一行，而该行的键被其他表的现有行中的外键所引用，则组成被引用行中的外键的所有值将被设置为 NULL。为了执行此约束，目标表的所有外键列必须可为空值。
SET DEFAULT	指定如果试图删除(或更新)某一行，而该行的键被其他表的现有行中的外键所引用，则组成被引用行中的外键的所有值将被设置为它们的默认值。为了执行此约束，目标表的所有外键列必须具有默认定义。如果某个列可为空值，并且未设置显式的默认值，则将使用 NULL 作为该列的隐式默认值。因 ON DELETE（或 UPDATE）SET DEFAULT 而设置的任何非空值在主表中必须有对应的值，才能维护外键约束的有效性。

以上的级别操作也可以使用“外键关系”对话框来设置，如图 11- 27。

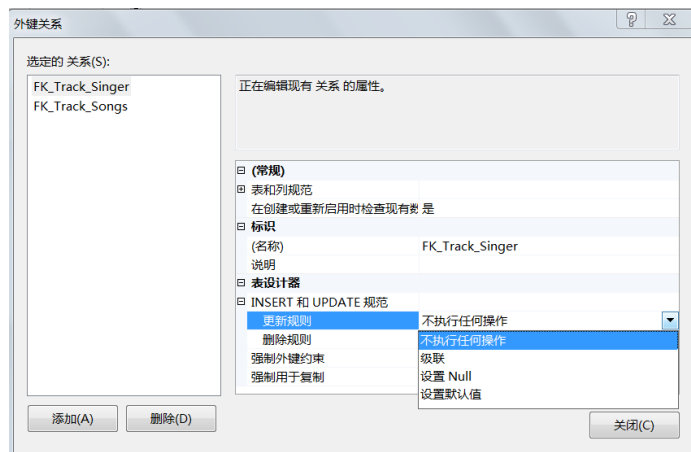


图 11-27 设置级联规则

- 说明：
- 不能为带有 INSTEAD OF DELETE 触发器的表指定 ON DELETE CASCADE。
 - 对于带有 INSTEAD OF UPDATE 触发器的表，不能指定下列各项：ON DELETE SET NULL、ON DELETE SET DEFAULT、ON UPDATE CASCADE、ON UPDATE SET NULL 以及 ON UPDATE SET DEFAULT。

例如，Track 表中的外键 FK_Track_Songs 默认的级联规则是“不执行任何操作”，因此，当用户修改主键表 Songs 的 SongID 值时，会产生错误并拒绝该操作，如图 11-28 所示。

下面我们来修改该外键的级联规则，假设将级联规则设为 CASCADE（级联），即主键值改变，对应的外键值也随之改变。首先，先删除原来的外键约束，用以下的修改表语句为 Track 表添加约束，且指定当主表进行 update 操作时，子表级联更新。

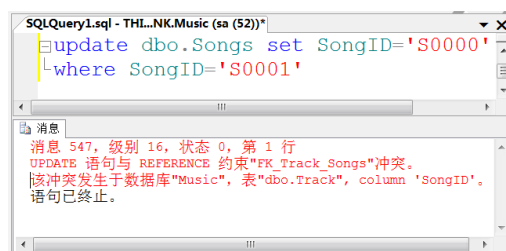


图 11-28 更新主表数据出错

--为 Track 添加外键，指定级联规则为级别更新

```
alter table Track add foreign key(SongID) references Songs(Songid) on
update cascade
```

设置外键后，执行以下修改语句，并查看两表记录，可以看到 Track 表中的 SongID 值也随之改变了，如图 11-29 所示。

--修改 Songs 表的 SongID

```
update dbo.Songs set SongID='S0000' where SongID='S0001'
```

--查看 Songs 记录

```
select top 3 * from Songs
```

--查看 Track 记录

```
select top 3 * from dbo.Track
```

结果	消息	Songs表中更新后的记录				
	SongID	Name	Lyricist	Composer	Lang	
1	S0000	传奇	左右	李健	中文	
2	S0002	后来	施人诚	玉城千春	中文	
3	S0101	Take Me Home, Country Road	John Denver	John Denver	英文	

	SongID	SingerID	Album	Style	Circulation	PubYear	Track表中更新后的记录
1	S0000	GC002	NULL	流行	8	NULL	
2	S0002	GA001	GOD BLE...	爵士	NULL	1975	
3	S0101	GA002	MEMORY	乡村	10	1971	

查询已成功执行。 THINK-THINK (10.0 RTM) | Sa (52) | Music | 00:00:00

图 11-29 更新数据后的表数据

4. 完整性约束案例分析

以某公司员工信息系统后台数据库设计为例，通过“员工信息”的定义代码实现 SQL Server 数据完整性控制。“员工管理”数据库中的部分表的结构如下：

员工信息（员工编号，姓名，出生日期，性别，地址，邮政编码，电话号码，部门编号）

部门信息（部门编号，部门名，备注）

利用 SQL Server 数据库 DDL 语言实现“员工信息表”6 种约束的实例代码：

```

CREATE TABLE 员工信息
(员工编号 char(6) PRIMARY KEY,          --主键约束
员工姓名 char(10) Not Null,             --NOT NULL 约束
出生日期 SmallDateTime,
性别 char(2) DEFAULT '男',              --默认约束
地址 varchar(40), 邮政编码 char(6) ,
电话号码 char(11), 部门编号 char(3),
UNIQUE(员工姓名),                       --唯一性约束
CHECK(性别 IN('男', '女')),             --检查约束
CHECK(邮政编码 LIKE '[0-9][0-9][0-9][0-9][0-9][0-9]'), --检查约束
CHECK(电话号码 LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'), --检查约束
FOREIGN KEY (部门编号) REFERENCE 部门信息(部门编号) ) --外键约束

```

“员工信息”表的以上定义实现了下列的数据完整性：

- 1) “员工编号”设置为主键（PRIMARY KEY），从而实现了实体完整性。
- 2) “员工姓名”设置了 NOT NULL 约束，防止有编号无员工姓名的状况。同时，为保证同一员工（假设无同名的员工）只有一个编号，为“员工姓名”建立了唯一（UNIQUE）约束。
- 3) 因该公司大多数为男性员工，“性别”设置默认值（DEFAULT）约束“男”，减少用户性别值的输入；同时，为防止用户误输入数据，还定义了该字段的检查（CHECK）约束，规定性别的值域为“男”和“女”。
- 4) “邮政编码”和“电话号码”的数据域为字符型数字信息，为防止录入信息的错误，也设置了检查（CHECK）约束，规定它们的取值由‘0’到‘9’字符型数据组成。规定取值范围的约束都是用户根据数据取值特征自行设计的，其主要目的是减少用户输入数据的误操作，从而实现用户自定义完整性。
- 5) “员工信息”表中的“部门编号”是外键，其取值类型和取值范围应参照“部门信息”表中的“部门编号”主键，为此，建立了外键约束，实现了参照完整性。

11.2.5 修改表和删除表

1. 修改表

在 SQL Server 中提供了两种修改表的方式：一是在 SSMS 中修改表；二是通过执行 T-SQL 语句修改表。

1) 在 SSMS 中修改表

在 SSMS 中，用户可以在图形界面下快捷地修改表。用鼠标选中要修改的表，右键单击表，选择“设计”选项，进入“表设计器”修改表。在“表设计器”中可以直接修改列名、列属性、默认值、标识选项等。也可以在表设计器中单击右键，通过选择快捷菜单选项，对表进行插入列、删除列、设置约束等。这些操作与建表时类似，限于篇幅，不再详细介绍。

2) 利用 T-SQL 语句修改表

T-SQL 提供了 ALTER TABLE 语句来修改表。不同的数据库厂商的数据库产品的 ALTER TABLE 语句格式略有不同，这里给出 SQL Server 支持的 ALTER TABLE 语句格式。

```
ALTER TABLE table_name
{ ALTER COLUMN column_name           --修改列定义
→ [ WITH { CHECK → NOCHECK } ]       --是否检查现有数据选项
→ADD COLUMN <column_definition>     --添加新列
→ADD [CONSTRAINT constraint_name] constraint_type --添加约束
→DROP [CONSTRAINT] constraint_name } --删除约束
```

其中添加列、删除列、修改列的语句同第四章中的修改表语句，具体见例 4-3 到 4-5。

例 11- 4：为 Student 添加约束，要求：主键是 sno；sex 只能“男”或“女”；sname 值不重复。实现代码如下：

```
alter table dbo.Student
add constraint pk_sno primary key(sno),           --主键约束
constraint ck_sex check(sex in('男','女')),      --检查约束
constraint unique_sname unique(sname)            --唯一约束
```

其中：添加唯一约束还可写成：

```
alter table dbo.Student add unique(sname)
```

例 11- 5：删除 unique_sname 约束。

```
alter table dbo.Student
drop unique_sname
```

- 在删除某列前要先解除与该列相关联的约束等关系。
- 删除列使用 ALTER TABLE 的 DROP COLUMN 子句，其中 COLUMN 不能省略，因为
注意：如果省略，则成了 ALTER TABLE DROP，这是删除约束的语句。
- 删除 PRIMARY KEY 约束时，如果另一个表中的 FOREIGN KEY 约束引用了该 PRIMARY KEY 约束，则必须先删除 FOREIGN KEY 约束。

3) WITH NOCHECK 的说明

表创建以后添加约束的时候，默认是 with check 选项，这种情况下，会检查已有数据，如果现有数据违反了拟添加的约束，则数据库引擎将返回错误信息，并且不添加约束。如果不想根据现有数据验证新的约束，可以使用 WITH NOCHECK 选项。比如：

```
alter table dbo.student
with nocheck
```

```
add constraint ck_dept check(dept in('computer','math'))
```

2. 删除表

删除表的操作将删除关于该表的所有定义（包括表的结构、索引、触发器、约束等）和表的数据。如果要删除主表，必须先删除子表或先在子表中删除对应的 FOREIGN KEY 约束。

在某些 DBMS 中，删除表会同时删除该表上建立的视图等对象。SQL Server 中的处理方法不同。SQL Server 中，删除表后，建立在该表上的视图定义仍然保留在数据字典中，但是当用户引用时出错。因此，任何引用已删除表的视图或存储过程都必须使用 DROP VIEW 或 DROP PROCEDURE 显式删除。

可在 SSMS 中使用图形界面删除表，步骤为：在 SSMS 中，右键点击拟删除的表，选择快捷菜单中的“删除”选项，进入“删除对象”窗口，单击“确定”，即可删除表。

也可用 T-SQL 的 DROP TABLE 语句删除表。如：

```
DROP TABLE mytest
```

说明：如果要删除表中的所有数据而不删除表本身，即清空表。可使用 TRUNCATE TABLE 语句。
语法为：TRUNCATE TABLE 表名

11.2.6 表的数据操作

1. 添加、修改和删除数据

表的数据操作包括添加记录、修改记录和删除记录。可以通过 SSMS 图形界面对表中的数据操作，也可以通过执行 SQL 语句对表中的数据进行操作。

1) 使用 SSMS 进行添加、修改和删除表数据

步骤如下：

（1）在“对象资源管理器”中，右键单击拟操作的表，选择“编辑前 200 行”，进入表数据窗口。

（2）在表数据窗口中，可以添加、删除、修改数据，结合右键快捷菜单，还可以进行剪切、复制和粘贴和删除记录等操作。如图 11-31 所示。

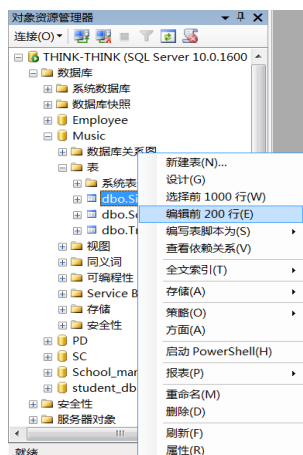


图 11- 30 选择编辑表

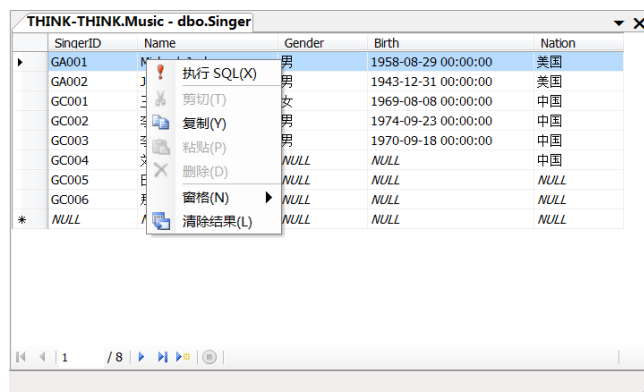


图 11- 31 表数据操作

在对表中数据进行操作的时候，需注意：

- 不得对设置为 IDENTITY 的列输入、修改和删除数据；
 - 对于设置 NOT NULL 但未设置 DEFAULT 值的列，必须赋值；
- 说明：
- 对于设置 PRIMARY KEY 或 UNIQUE 约束的列，不能赋空值或重复值（UNIQUE 约束列可赋一个空值）；
 - 如果对表设置了参照完整性，则应该先向主表插入数据，再向子表插入数据；删除数据时，先删除子表中的数据，再删除主表中相应的数据，否则会违反参照完整性规则。

2) 使用 SQL 语句对数据操作

对表中数据插入、修改和删除操作，SQL Server 支持标准 SQL 语句中的 INSERT、UPDATE 和 DELETE 语句来完成，这些语句的具体使用语法和示例见第四章 4.5.1 节。

说明：DELETE 语句只从表中删除数据，它不删除表定义本身。如果要删除表定义和表数据，应用 DROP TABLE 语句。

除此以外，SQL Server 还提供了针对数据操作的以下扩展功能。

(1) 使用行构造器插入多行数据

SQL Server 在 2008 版本中新增了行构造器（Row Constructors）用来在 insert 语句中一次性插入多行给定值的数据。语法为：

```
INSERT [ INTO ] table_name
Values ( { DEFAULT → NULL → expression } [ ,...n ] ) [ ,...n ]
```

可以在单个 INSERT 语句中插入的最大行数为 1000。若要插入超过 1000 行的数据，可以创建多个 INSERT 语句，或者通过使用 bcp 实用工具或 BULK INSERT 语句大容量导入数据。例如如下代码可以在 Singer 中插入两行记录,分别为('GC005','田震')和('GC006','那英')。

```
insert into dbo.Singer(SingerID,Name)
values('GC005','田震'),('GC006','那英')
```

(2) 在 INSERT、UPDATE 或 DELETE 中使用 OUTPUT 子句

在 SQL Server 2005 之前，用户如果希望在 INSERT、UPDATE、DELETE 执行时，查看这些命令所影响的行的信息，只能通过触发器来实现（在 DML 触发器中，返回 Inserted 表或 Deleted 表的记录，具体见 13.2.3 的介绍）。

SQL Server 2005 版本开始，为 INSERT、UPDATE、DELETE 语句增加了一个可选项 OUTPUT 子句，该子句可以返回受 INSERT、UPDATE、DELETE 影响的各行中的信息，或返回基于受这些语句影响的各行的表达式。这些结果可以返回到处理应用程序，以供在确认消息、存档以及其他类似的应用程序要求中使用；也可以将这些结果插入表或表变量。另外，还可以捕获嵌入的 INSERT、UPDATE、DELETE 或 MERGE 语句中 OUTPUT 子句的结果，然后将这些结果插入目标表或视图。

OUTPUT 子句在 INSERT、UPDATE、DELETE 语句中使用的语法结构如下：

```
--在 INSERT 语句中使用
INSERT [ INTO ] table_name[ ( column_list ) ]
    [ <OUTPUT Clause> ]                /*OUTPUT 子句*/
    VALUES ( { DEFAULT → NULL → expression } [ ,...n ] ) [ ,...n ]

--在 UPDATE 语句中使用
UPDATE table_name SET column_name = { expression → DEFAULT → NULL }
    [ <OUTPUT Clause> ]                /*OUTPUT 子句*/
    [ WHERE { <search_condition>

--在 DELETE 语句中使用
DELETE [ FROM ] table_name
    [ <OUTPUT Clause> ]                /*OUTPUT 子句*/
    [ WHERE { <search_condition>
```

其中，OUTPUT Clause 的语法如下：

```
<OUTPUT_CLAUSE> ::=
    { [OUTPUT                                dml_select_list                INTO
    { @table_variable → output_table } ( (column_list) ) ]
    [OUTPUT dml_select_list] }
```

其中，

① @table_variable:指定一个 table 变量，返回的行将插入此变量，而不是返回给调用方。

② output_table:指定一个表，返回的行将被插入该表中而不是返回到调用方。output_table 可为临时表。

③ column_list: INTO 子句目标表上列名的可选列表。

④ dml_select_list:定义希望返回的列。其格式如下：

```
< dml_select_list > ::= { DELETED → INSERTED } . { * → column_name }
```

其中，列名的前缀是 DELETED 表或 INSERTED 表。

说明： DELETED 表和 INSERTED 表是由系统创建和维护的两张逻辑表，这两张表中包含了在 DML 操作中插入或删除的所有记录，因此这两个表的结构总是与 DML 作用的表的结构相同。

其中，

① DELETED: 列前缀，指定由更新或删除操作删除的值。以 DELETED 为前缀的列反映了 UPDATE、DELETE 语句完成之前的值。

② INSERTED: 列前缀，指定由插入操作或更新操作添加的值。以 INSERTED 为前缀的列反映了在 UPDATE、INSERT 语句完成之后但在触发器执行之前的值。

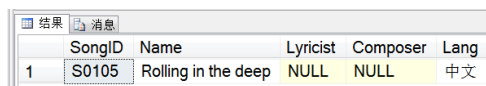
由 OUTPUT 子句的语法可以看出，OUTPUT 有两种用法：

- ① 可以像 SELECT 命令那样将结果返回给用户；
- ② 可以与 INTO 子句一起将结果插入表或表变量中。

下面我们举例来说明 OUTPUT 子句的用法。

例 11-6: 在 Songs 表中插入一条记录，并在 INSERT 语句中使用 OUTPUT 返回该记录。
代码如下：

```
insert into Songs (SongID,name)
--在界面上返回插入的记录
output inserted.*
values('S0105','Rolling in the deep')
```



	SongID	Name	Lyricist	Composer	Lang
1	S0105	Rolling in the deep	NULL	NULL	中文

图 11-32 在 insert 中使用 OUTPUT 子句

结果如图 11-33 所示。

例 11-7: 修改 Songs 表，把 'S0105' 歌曲的语言改为英文，并在 UPDATE 语句中使用 OUTPUT 返回修改前后的值。

代码如下：

```
update dbo.Songs set Lang='英文'
OUTPUT inserted.Lang as 修改后的值,deleted.lang as 修
改前的值 where SongID='S0105'
```



	修改后的值	修改前的值
1	英文	中文

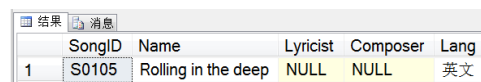
图 11-33 OUTPUT 子句

结果如图 11-33 所示。

例 11-8: 在 Songs 表删除 ID 为 'S0105' 的歌曲，并把删除的记录存储在临时表 #temp_tb 中。

为此，我们要先创建 #temp_tb，然后在 DELETE 语句中使用 OUTPUT INTO 子句把删除的记录插入到 #temp_tb 中。代码如下：

```
--创建 #temp_tb
create table #temp_tb (
SongID char(10) NOT NULL,
Name varchar(50) NULL,
Lyricist varchar(20) NULL,
Composer varchar(20) NULL,
Lang varchar(16) NULL)
--删除数据，并把删除的数据插入到 temp_tb
delete from dbo.Songs
OUTPUT deleted.* into #temp_tb
where SongID='S0105'
```



	SongID	Name	Lyricist	Composer	Lang
1	S0105	Rolling in the deep	NULL	NULL	英文

图 11-34 在 DELETE 中使用 OUTPUT 子句

可以用 “select * from #temp_tb” 语句查看删除的记录，结果如图 11-34 所示。

2. TRUNCATE TABLE 语句

TRUNCATE TABLE 与不带 WHERE 子句的 DELETE 语句功能相同，均可以删除表中的所有行。但是，TRUNCATE TABLE 速度更快，并且使用更少的系统资源和事务日志资源。具体来说，与 DELETE 相比，TRUNCATE 具有以下优点：

- 1) 所用的事务日志空间较少

DELETE 语句每次删除一行，并在事务日志中为所删除的每行做记录，以防止删除失败时可以使用事务日志来恢复数据。TRUNCATE TABLE 通过释放用于存储表数据的数据页来删除数据，并且在事务日志中只记录页的释放操作，所以大大减少了日志的数量，从而可以加快删除数据的速度。

说明： 由于 TRUNCATE TABLE 命令不会在事务日志中记录数据删除操作，因此不能激活触发器。

2) 使用的锁通常较少

当使用行锁执行 DELETE 语句时，将锁定表中各行以便删除。TRUNCATE TABLE 始终锁定表和页，而不是锁定各行。

如要删除 student 中的所有数据，可使用如下语句：

```
TRUNCATE TABLE student
```

3. 数据查询

1) 查询设计器中设计查询

SQL Server 中对于表数据的查询可以直接在“查询编辑器”中输入 SELECT 语句（SELECT 语句的写法可参见第四章 4.4 节数据查询）；SSMS 还提供了“查询设计器”，便于不熟悉 SQL 编程的用户迅速构建查询。以下介绍用“查询设计器”来构建查询，步骤如下：

（1）在 SSMS 中，点击工具栏中的“新建查询”，选择主菜单“查询”→“在编辑器中设计查询”选项，如图 11-35 所示，即可启动查询设计器，用它可快速构建查询。

（2）“查询设计器”启动后，先弹出“添加表”对话框（图 11-36），其中列出了当前数据库中可以使用的源表或视图。本例中选择三张表（可按住“SHIFT”多选），点击“添加”并“关闭”该对话框。

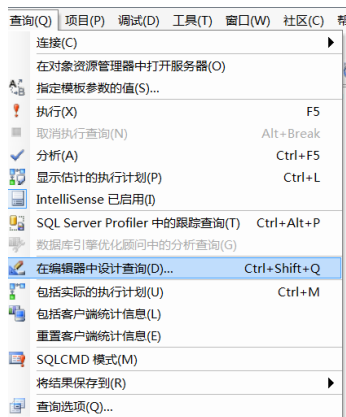


图 11-35 启动“查询设计器”

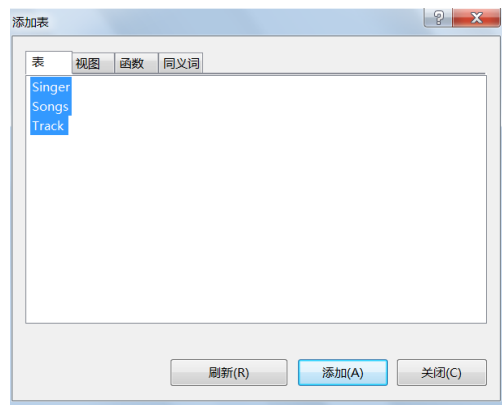


图 11-36 “添加表”对话框

(3) 查询是在如图 11-37 所示的“查询设计器”中进行设计的。该界面主要区域的功能如下。

① “表选择区”：添加或删除用于定义视图的表。在该区域内，在右键菜单中选择“添加表”选项，可重新激活“添加表”对话框。

② “用户输入区”：用户可以在此区域添加或删除用于定义视图中的列，打√标识选中该列输出；定义 WHERE 条件；定义排序等。其中“筛选器”和“或...”均表示条件，各栏下的行与行之间为 AND 关系，栏与栏之间是 OR 关系。

③ “SQL 语句区”：系统根据用户在“用户输入区”中指定的内容自动构建 SQL 查询语句并显示在该区域。

在图 11-37 中的查询是显示歌唱风格为“摇滚”的歌手名、歌曲名、风格及发行量，结果按发行量降序排序。

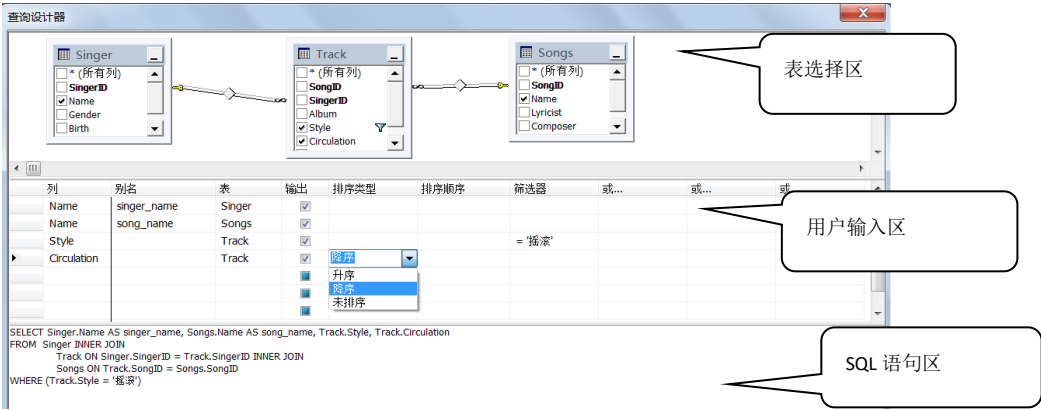


图 11-37 “查询设计器”

2) 查看语句的执行计划

(1) 在菜单栏单击“包括实际的执行计划”按钮，如图 11-38 所示。



图 11-38 “包括实际执行计划”按钮

(2) 重新执行 SQL 语句，然后切换到“执行计划”选项卡，可以查看该语句的执行计划，如图 11-39 所示。

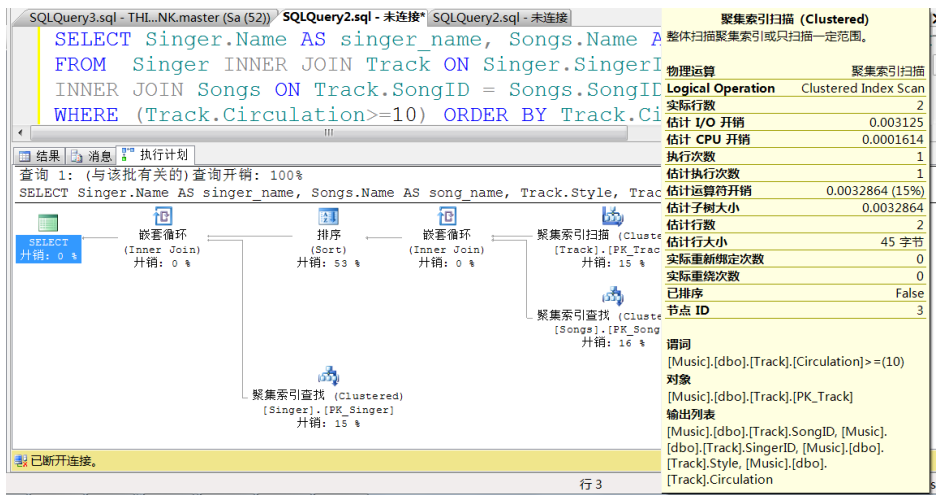


图 11-39 SQL 语句的执行计划

可以看出, 执行计划以树状结构显示。如果 SQL Server 执行的语句是 SELECT、INSERT、DELETE 或 UPDATE 语句, 则该语句是树根, 父节点相同的子节点被绘制在同一列中。当鼠标移动到某一节点时, 会显示出该节点的详细说明, 如 I/O 开销、CPU 开销、执行次数等。

3) 查看客户端统计信息

(1) 在菜单栏单击“包括客户端统计信息”按钮, 如图 11-40 所示。



图 11-40 “包含客户端统计信息”按钮

(2) 重新执行 SQL 语句, 然后切换到“客户端统计信息”选项卡, 出现如图 11-41 所示的客户端统计信息。

(3) 显示的客户端统计信息包含 3 部分。

- ①查询配置文件统计信息可以判断该 SQL 语句消耗的服务器资源情况;
 - ②网络统计信息可以判断该 SQL 语句消耗的网络资源情况;
 - ③时间统计信息可以判断服务器与客户机之间的通信情况。
- 可以通过以上信息评估 SQL 语句的效率。

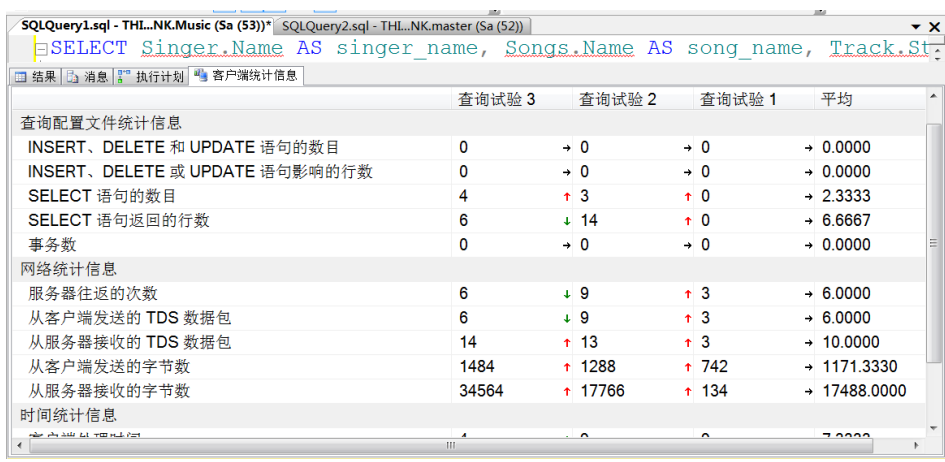


图 11-41 客户端统计信息

4) 查询结果的存储

(1) SELECT INTO 语句

SELECT INTO 语句用于创建一个新表，并用 SELECT 语句的结果集填充该表；还可将几个表或视图中的数据组合成一个表。新表的结构由选择列表中表达式的属性定义。SELECT INTO 语句常用于创建表的备份复件或者用于对记录进行归档。

SELECT INTO 语句的基本语法如下：

```
SELECT <select_list> INTO new_table
FROM {<table_source>} [,...n] ...
```

- 说明：
- INTO 子句应放在 FROM 子句之前；
 - 如果当前数据库中已有与拟创建表同名的表，则创建不能成功完成。

以下的示例语句是从多个雇员和与地址相关的表中选择 7 列来创建表 dbo.EmployeeAddresses。

```
SELECT c.FirstName, c.LastName, e.Title, a.AddressLine1, a.City, sp.Name
AS [State/Province], a.PostalCode INTO dbo.EmployeeAddresses
FROM Contact AS c JOIN Employee AS e ON e.ContactID = c.ContactID JOIN
EmployeeAddress AS ea ON ea.EmployeeID = e.EmployeeID
JOIN Address AS a on a.AddressID= ea.AddressID
JOIN StateProvince as sp ON sp.StateProvinceID = a.StateProvinceID;
```

(2) 将查询结果保存到文件

设置方法：进入查询编辑器窗口，选择主菜单“查询”→“将结果保存到”→“将结果保存到文件”，如**错误!未找到引用源。**所示。这样，此后所进行的查询，其结果将以文本形式输出到磁盘文件中。

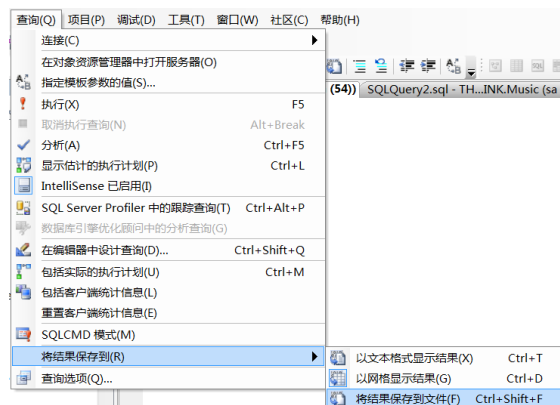


图 11-42 设置查询结果保存到文件