Chapter 8

Creating and Managing Databases with Scripts

Learning Objectives

- *Understand the rationale for using scripts to iteratively evolve a database design*
- *Use SQL Server Management Studio to convert a database into scripts*
- *Use SQL scripts to automatically drop and create a database*
- *Use SQL scripts to automatically create database tables and other database objects*
- *Execute a suite of SQL scripts via a DOS batch file*
- *Organize database scripts to ease configuration management*

Introduction

In this chapter I'm going to show you how to take the EmployeeTraining database designed in Chapter 7 and convert it into a set of scripts to facilitate iterative database development. Along the way I'll show you how to organize your database scripts to ease configuration management as well as how to execute a suite of SQL database scripts using a DOS batch file.

To complete the activities shown in this chapter you'll need to have created a database using SQL Server Management Studio. You'll also need a good text editor like Notepad++.

You should by now have set up a projects folder according to the instructions given in Chapter 3 and be using Subversion to commit project artifacts to the repository.

I know this may sound theatrical, but what you are about to learn in this chapter may very well be a life-changing developer skill.

Process Overview

The process I'm going to demonstrate to you in this chapter in a nutshell includes the following steps:

Step 1: Create release folder and a logs folder in Projects\EmployeeTrainingProject\trunk\database folder.

Step 2: Use SQL Server Management Studio to generate database scripts for a selected database.

Step 3: Segregate the scripts into logically related sections and save as individual SQL script files. You'll have one script to drop the database, another script to create the database, and still another script to create the tables, etc.

Step 4: Organize the database scripts according to release. Group related database scripts in their respective release folders.

Step 5: Create additional scripts to automatically insert test data into the database for development and testing.

Step 6: Create a DOS batch file to execute the scripts automatically. Name these batch files according to release. As the database design evolves over the life of the project you'll create a chain of these batch files so that you need only execute the latest release batch file, which will call the previous release batch file, and so on.

Step 7: Run a release script each time you want to recreate the database. This is done frequently during application development and testing.

Once you have your scripts created, organized, and working, dropping and completely recreating the database takes only a matter of seconds. You can return to SQL Server Management Studio any time to modify the database and incorporate the changes into your script files. Alternatively, once you gain proficiency in writing database scripts, you can edit the scripts directly to modify the database design.

Generating Database Scripts

In this section I will step through the process I described in the previous section. As I said in the introduction, I'm assuming you have created a projects folder that contains a local working copy of a Subversion project. The name of the project I'm working on is EmployeeTrainingProject.

Step 1: Create Release and Logs Folders

The first step is to get organized. In the Projects\EmployeeTrainingProjects\trunk\database folder create two folders, one named "logs" and another named "Release_1.0_DB_Scripts". The logs folder will be used to hold the output of the SQL database scripts while the Release_1.0_DB_Scripts folder will be used to hold all the scripts related to release 1.0 of the database. When you create your new folders your database folder will look like figure 8-1.

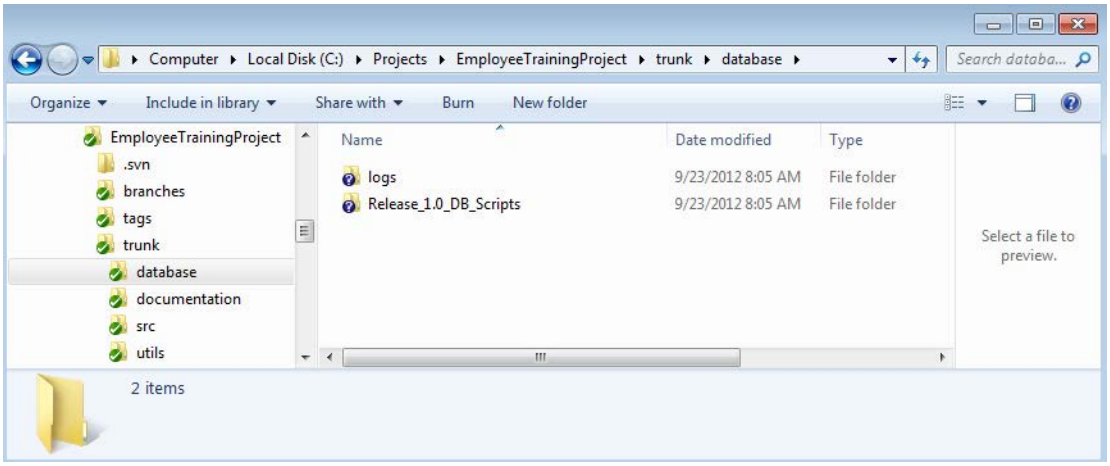

Figure 8-1: Contents of database Folder After Creating New Folders

Referring to figure 8-1 — The question mark on each folder means they have not yet been added to the repository. You can wait until you've created all the scripts and tested them to ensure they work properly before committing them to the repository.

Step 2: Generate Scripts

You'll actually generate several scripts in this step. First, you'll script the database so that it can be dropped and created. Next, you'll script the objects within the database. These include tables, views, users, triggers, etc. At this point the EmployeeTraining database has only a few tables, their foreign key relationships, and one user so there aren't too many objects to script.

Script The Database

Launch SQL Server Management Studio and right-click on the EmployeeTraining database and from the pop-up menu select **Script Database as -> DROP And CREATE To -> Clipboard** as is shown in figure 8-2.

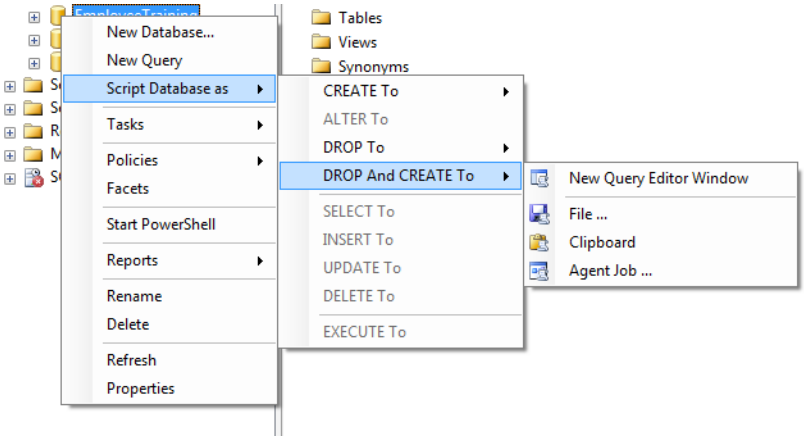

Figure 8-2: Script the Database to the Clipboard

Referring to figure 8-2 — Select **Clipboard** and when the pop-up menu goes away open Notepad++ and paste the contents of the clipboard into a new file called DropAndCreateDatabase.sql. The contents of DropAndCreateDatabase.sql should look similar to example 8.1.

8.1 DropAndCreateDatabase.sql

```
1       USE [master]
2       GO
3
4       /****** Object:  Database [EmployeeTraining]    Script Date: 09/23/2012 10:34:43 ******/
5       IF  EXISTS (SELECT name FROM sys.databases WHERE name = N'EmployeeTraining')
6       DROP DATABASE [EmployeeTraining]
7       GO
8
9       USE [master]
10      GO
11
12      /****** Object:  Database [EmployeeTraining]    Script Date: 09/23/2012 10:34:43 ******/
13      CREATE DATABASE [EmployeeTraining] ON  PRIMARY
14      ( NAME = N'EmployeeTraining', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\EmployeeTraining.mdf' , SIZE = 2048KB , MAXSIZE = UNLIMITED, FILEGROWTH
= 1024KB )
15       LOG ON
16      ( NAME = N'EmployeeTraining_log', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\EmployeeTraining_log.ldf' , SIZE = 1024KB , MAXSIZE = 2048GB ,
FILEGROWTH = 10%)
17      GO
18
19      ALTER DATABASE [EmployeeTraining] SET COMPATIBILITY_LEVEL = 100
20      GO
21
22      IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
23      begin
24      EXEC [EmployeeTraining].[dbo].[sp_fulltext_database] @action = 'enable'
25      end
26      GO
27
28      ALTER DATABASE [EmployeeTraining] SET ANSI_NULL_DEFAULT OFF
29      GO
30
31      ALTER DATABASE [EmployeeTraining] SET ANSI_NULLS OFF
32      GO
33
34      ALTER DATABASE [EmployeeTraining] SET ANSI_PADDING OFF
35      GO
36
37      ALTER DATABASE [EmployeeTraining] SET ANSI_WARNINGS OFF
38      GO
39
40      ALTER DATABASE [EmployeeTraining] SET ARITHABORT OFF
41      GO
42
43      ALTER DATABASE [EmployeeTraining] SET AUTO_CLOSE OFF
44      GO
45
46      ALTER DATABASE [EmployeeTraining] SET AUTO_CREATE_STATISTICS ON
47      GO
48
49      ALTER DATABASE [EmployeeTraining] SET AUTO_SHRINK OFF
50      GO
51
52      ALTER DATABASE [EmployeeTraining] SET AUTO_UPDATE_STATISTICS ON
53      GO
54
55      ALTER DATABASE [EmployeeTraining] SET CURSOR_CLOSE_ON_COMMIT OFF
56      GO
57
58      ALTER DATABASE [EmployeeTraining] SET CURSOR_DEFAULT  GLOBAL
59      GO
60
61      ALTER DATABASE [EmployeeTraining] SET CONCAT_NULL_YIELDS_NULL OFF
62      GO
63
64      ALTER DATABASE [EmployeeTraining] SET NUMERIC_ROUNDABORT OFF
65      GO
66
67      ALTER DATABASE [EmployeeTraining] SET QUOTED_IDENTIFIER OFF
68      GO
69
70      ALTER DATABASE [EmployeeTraining] SET RECURSIVE_TRIGGERS OFF
71      GO
72
73      ALTER DATABASE [EmployeeTraining] SET  DISABLE_BROKER
74      GO
75
```



```
76      ALTER DATABASE [EmployeeTraining] SET AUTO_UPDATE_STATISTICS_ASYNC OFF
77      GO
78
79      ALTER DATABASE [EmployeeTraining] SET DATE_CORRELATION_OPTIMIZATION OFF
80      GO
81
82      ALTER DATABASE [EmployeeTraining] SET TRUSTWORTHY OFF
83      GO
84
85      ALTER DATABASE [EmployeeTraining] SET ALLOW_SNAPSHOT_ISOLATION OFF
86      GO
87
88      ALTER DATABASE [EmployeeTraining] SET PARAMETERIZATION SIMPLE
89      GO
90
91      ALTER DATABASE [EmployeeTraining] SET READ_COMMITTED_SNAPSHOT OFF
92      GO
93
94      ALTER DATABASE [EmployeeTraining] SET HONOR_BROKER_PRIORITY OFF
95      GO
96
97      ALTER DATABASE [EmployeeTraining] SET  READ_WRITE
98      GO
99
100     ALTER DATABASE [EmployeeTraining] SET RECOVERY FULL
101     GO
102
103     ALTER DATABASE [EmployeeTraining] SET  MULTI_USER
104     GO
105
106     ALTER DATABASE [EmployeeTraining] SET PAGE_VERIFY CHECKSUM
107     GO
108
109     ALTER DATABASE [EmployeeTraining] SET DB_CHAINING OFF
110     GO
```

Referring to example 8.1 — Note that I've let lines 14 and 16 wrap around to the next line. Also note that on line 1 the `USE [master]` command specifies that this script must be run at the master database level.

In Step 3 I will segregate this script into two distinct scripts: one named DropDatabase.sql and the other named CreateDatabase.sql, but for now, we'll leave this script alone and proceed to scripting the remaining database objects.

Script The User

In SQL Server Management Studio expand the EmployeeTraining -> Security -> Users folder, right-click the EmployeeTraining user, and from the pop-up menu select **Script user as -> Create To -> Clipboard** as is shown in figure 8-3.

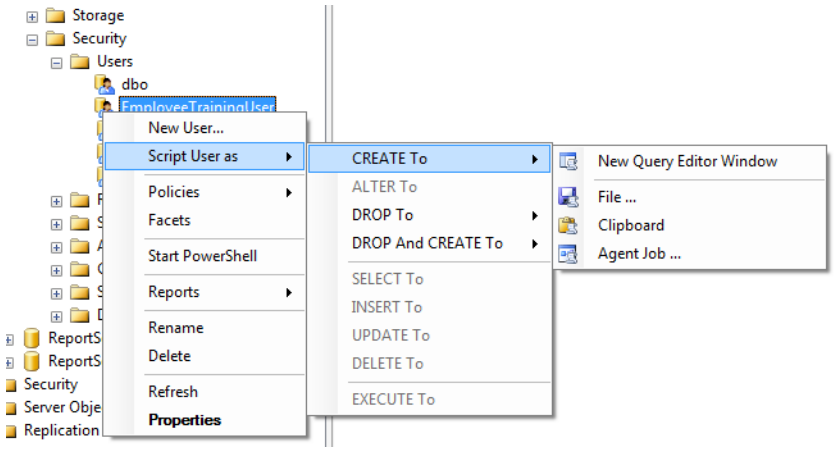

Figure 8-3: Script the User to the Clipboard

Referring to figure 8-3 — You only need to script the user as Create because when you drop the database you will drop all objects within the database, including the user. Thus, you only need a script to create the user. The resulting script is listed in example 8.2.

8.2 CreateUser.sql

```
1       USE [EmployeeTraining]
2       GO
3
4       /****** Object:  User [EmployeeTrainingUser]    Script Date: 10/13/2012 08:30:07 ******/
5       GO
6
7       CREATE USER [EmployeeTrainingUser] FOR LOGIN [EmployeeTraining] WITH DEFAULT_SCHEMA=[dbo]
8       GO
```

Referring to example 8.2 — Note that on line 1 the `USE [EmployeeTraining]` command specifies that this script must be run at the EmployeeTraining database level.

Script The Login

In SQL Server Management Studio expand the outermost Security folder and the Logins subfolder. Right-click the EmployeeTraining login and select **Script Login as -> DROP And CREATE To -> Clipboard** as is shown in figure 8-4.

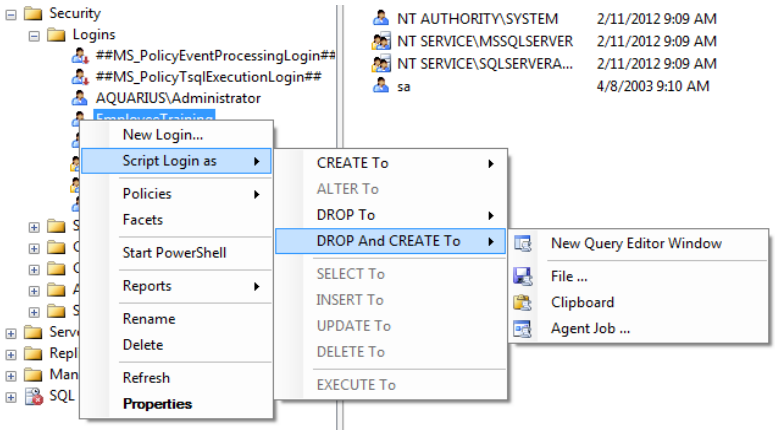

Figure 8-4: Scripting The Login

Referring to figure 8-4 — The reason you want to Drop and Create the Login is because it exists outside the EmployeeTraining database and will not be dropped when the database is dropped, thus the need to drop the login before creating it. Example 8.3 gives the resulting script.

8.3 DropAndCreateLogin.sql

```
1       /****** Object:  Login [EmployeeTraining]    Script Date: 10/13/2012 08:52:28 ******/
2       IF  EXISTS (SELECT * FROM sys.server_principals WHERE name = N'EmployeeTraining')
3       DROP LOGIN [EmployeeTraining]
4       GO
5
6       /* For security reasons the login is created disabled and with a random password. */
7       /****** Object:  Login [EmployeeTraining]    Script Date: 10/13/2012 08:52:28 ******/
8       CREATE LOGIN [EmployeeTraining] WITH PASSWORD=N'ë?ÈÎ??ýik]??r? 4~-¤?§Â·}¥FÇ ?',
DEFAULT_DATABASE=[EmployeeTraining], DEFAULT_LANGUAGE=[us_english], CHECK_EXPIRATION=OFF, CHECK_POLICY=OFF
9       GO
10
11      EXEC sys.sp_addsrvrolemember @loginame = N'EmployeeTraining', @rolename = N'dbcreator'
12      GO
13
14      ALTER LOGIN [EmployeeTraining] DISABLE
15      GO
```

Referring to example 8.3 — I've let line 8 wrap around due to its length. Also note that when you generate this script the login password will show up as unreadable characters as you can see above. Also, line 14 disables the login. You'll need to edit this script to give it a usable password and remove the ALTER LOGIN statement on line 14. The edited script is given in example 8.4.

8.4 DropAndCreateLogin.sql (edited)

```
1       /****** Object:  Login [EmployeeTraining]    Script Date: 10/13/2012 08:52:28 ******/
2       IF  EXISTS (SELECT * FROM sys.server_principals WHERE name = N'EmployeeTraining')
3       DROP LOGIN [EmployeeTraining]
4       GO
5
6       /* For security reasons the login is created disabled and with a random password. */
7       /****** Object:  Login [EmployeeTraining]    Script Date: 10/13/2012 08:52:28 ******/
```



```
8       CREATE LOGIN [EmployeeTraining] WITH PASSWORD=N'password', DEFAULT_DATABASE=[EmployeeTraining],
DEFAULT_LANGUAGE=[us_english], CHECK_EXPIRATION=OFF, CHECK_POLICY=OFF
9       GO
10
11      EXEC sys.sp_addsrvrolemember @loginame = N'EmployeeTraining', @rolename = N'dbcreator'
12      GO
```

Script The Tables

It's now time to script the remaining database objects. At this point the EmployeeTraining database contains only three tables and their foreign key relationships. To script these, right-click the EmployeeTraining database and select **Tasks -> Generate Scripts...** as is shown in figure 8-5.

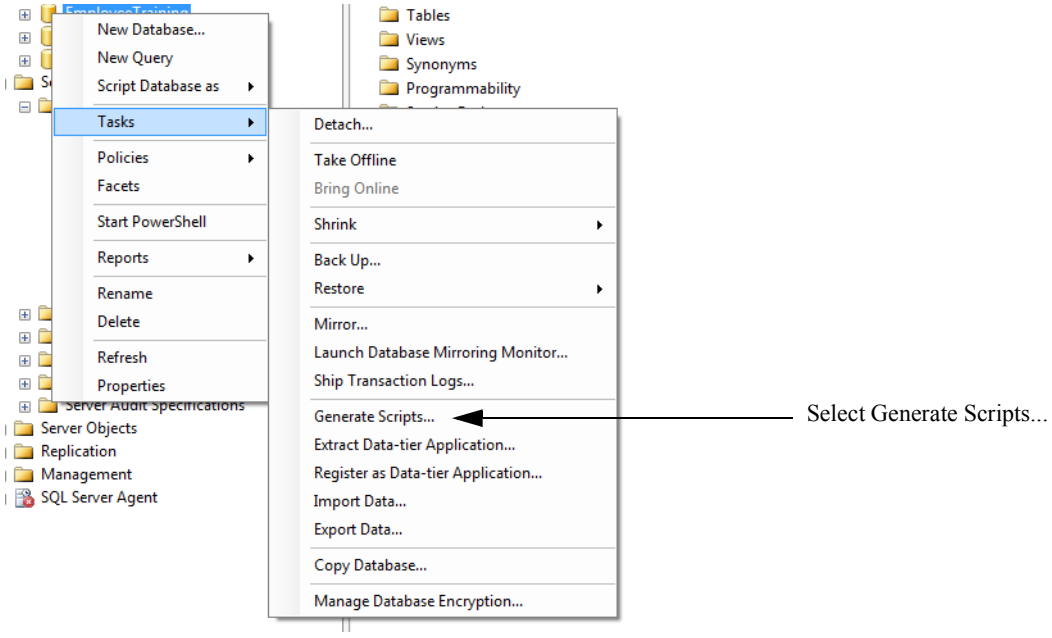

Figure 8-5: Select Tasks -> Generate Scripts...

Referring to figure 8-5 — Selecting **Tasks -> Generate Scripts...** will open the Generate and Publish Scripts wizard as is shown in figure 8-6.

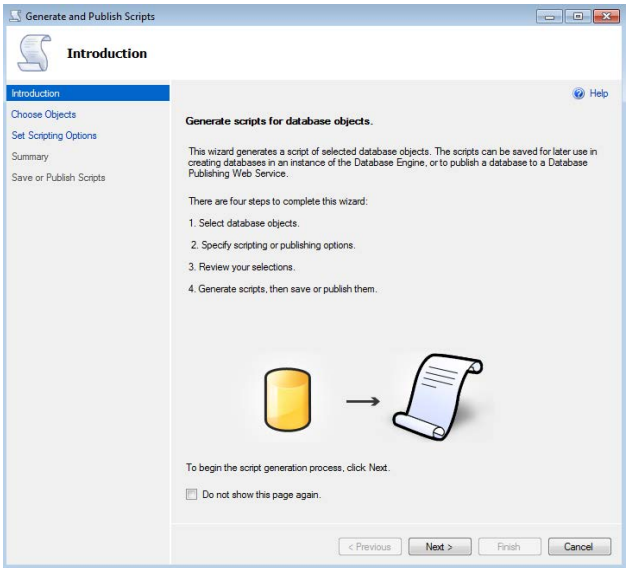

Figure 8-6: Generate and Publish Scripts Wizard Introduction Page

Referring to figure 8-6 — Click the **Next >** button to proceed to the Choose Objects page as is shown in figure 8-7.

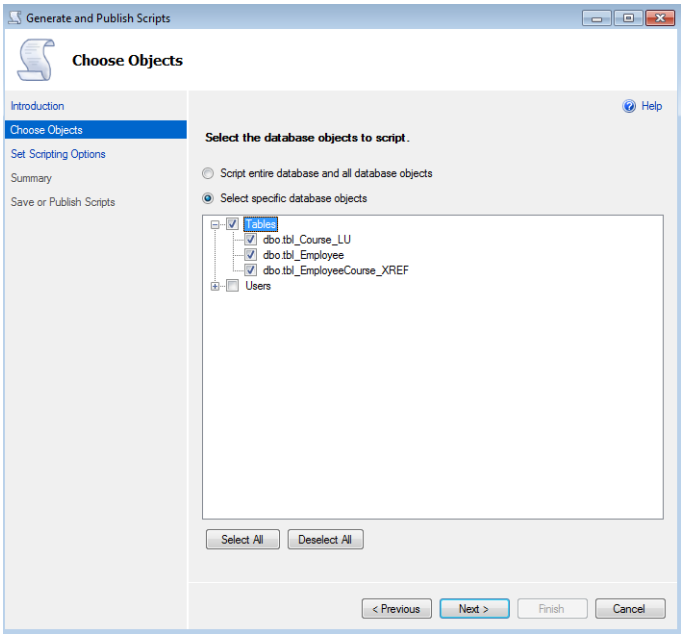

Figure 8-7: Choose Objects Page

Referring to figure 8-7 — Click the **Tables** checkbox and click the **Next >** button to proceed to the Set Scripting Options page shown in figure 8-8.

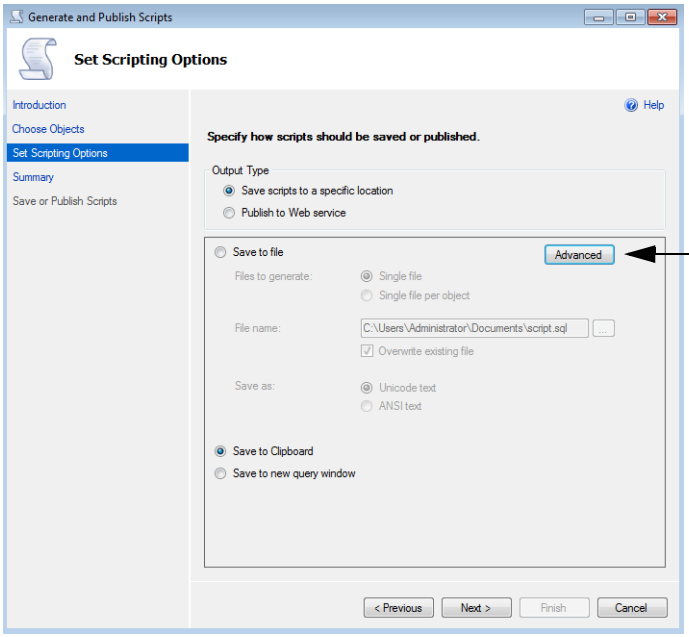

Figure 8-8: Set Scripting Options Page

Referring to figure 8-8 — Under Output Type click the **Save Scripts to a specific location** radio button. Next, click the **Save to Clipboard** radio button. To set advanced scripting options click the **Advanced** button. This brings up the Advanced Scripting Options dialog as shown in figure 8-9.

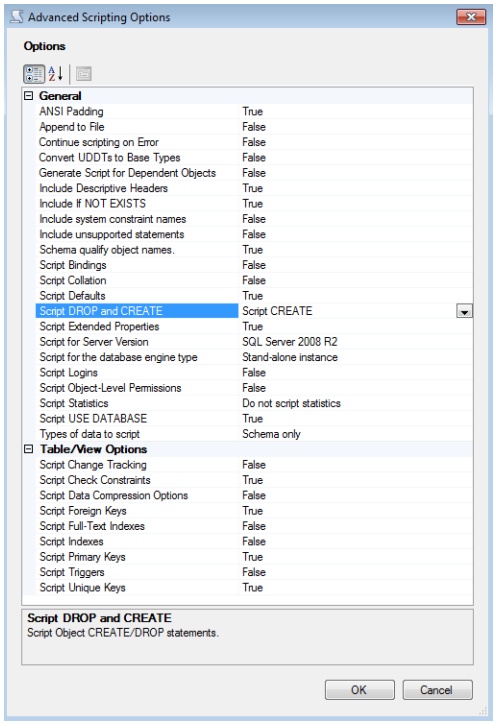

Figure 8-9: Advanced Scripting Options Page

Referring to figure 8-9 — I've set "Include if NOT EXISTS" to True and "Script DROP and CREATE" to Script CREATE. When you've finished setting these options click the **OK** button to dismiss the dialog. Click the **Next >** button on the Set Scripting Options page to proceed to the Summary page as is shown in figure 8-10.

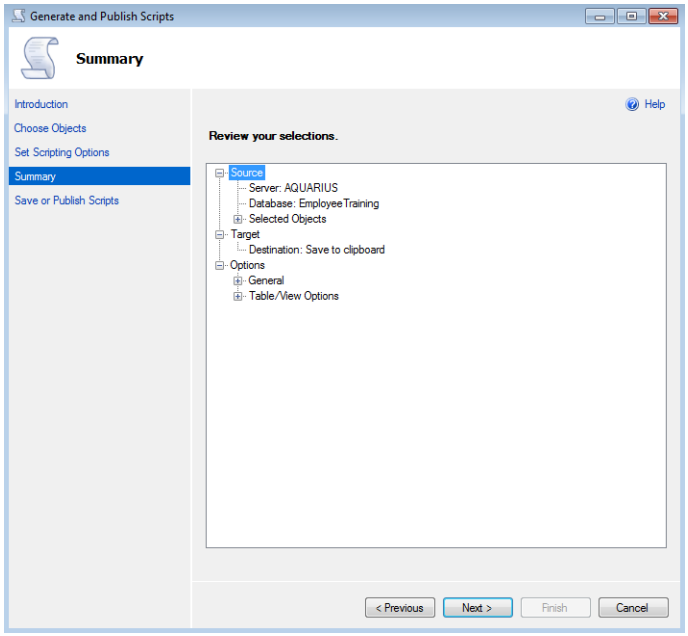

Figure 8-10: Summary Page

Referring to figure 8-10 — Review your selections on the Summary page by expanding each section. When you're ready to generate the scripts click the **Next >** button. This will bring you to the Save or Publish Scripts page and as each phase of script generation completes a green circle with a white check mark will appear next to each listed action as is shown in figure 8-11.

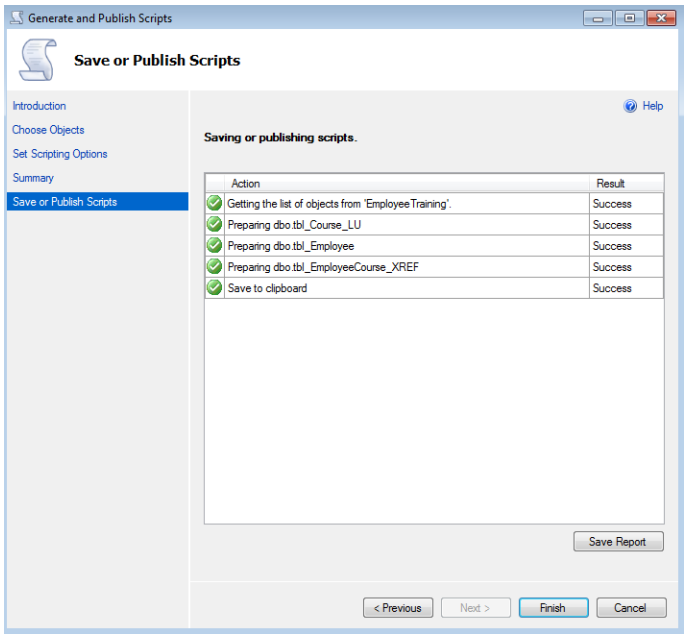

Figure 8-11: Save or Publish Scripts Page

Referring to figure 8-11 — When script generation completes click the **Finish** button. Your scripts will now be in the clipboard and ready to be pasted into a Notepad++ file. Example 8.5 gives the resulting script. I've saved it in a file named CreateTables.sql.

8.5 CreateTables.sql

```
1       USE [EmployeeTraining]
2       GO
3       /****** Object:  Table [dbo].[tbl_Employee]    Script Date: 10/13/2012 09:55:02 ******/
4       SET ANSI_NULLS ON
5       GO
6       SET QUOTED_IDENTIFIER ON
7       GO
8       SET ANSI_PADDING ON
9       GO
10      IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[tbl_Employee]') AND type
in (N'U'))
11      BEGIN
12      CREATE TABLE [dbo].[tbl_Employee](
13          [EmployeeID] [int] IDENTITY(1,1) NOT NULL,
14          [FirstName] [varchar](50) NOT NULL,
15          [MiddleName] [varchar](50) NOT NULL,
16          [LastName] [varchar](50) NOT NULL,
17          [Birthday] [datetime] NOT NULL,
18          [Picture] [varbinary](max) NULL,
19          [HireDate] [datetime] NOT NULL,
20       CONSTRAINT [PK_tbl_Employee] PRIMARY KEY CLUSTERED
21      (
22          [EmployeeID] ASC
23      )WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON,
ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
24      ) ON [PRIMARY]
25      END
26      GO
27      SET ANSI_PADDING OFF
28      GO
29      /****** Object:  Table [dbo].[tbl_Course_LU]    Script Date: 10/13/2012 09:55:02 ******/
30      SET ANSI_NULLS ON
31      GO
32      SET QUOTED_IDENTIFIER ON
33      GO
34      SET ANSI_PADDING ON
35      GO
36      IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[tbl_Course_LU]') AND type
in (N'U'))
37      BEGIN
38      CREATE TABLE [dbo].[tbl_Course_LU](
```



```
39          [CourseID] [int] IDENTITY(1,1) NOT NULL,
40          [Code] [varchar](6) NOT NULL,
41          [Title] [varchar](200) NOT NULL,
42          [Description] [varchar](1000) NOT NULL,
43       CONSTRAINT [PK_tbl_Course_LU] PRIMARY KEY CLUSTERED
44      (
45          [CourseID] ASC
46      )WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON,
ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
47      ) ON [PRIMARY]
48      END
49      GO
50      SET ANSI_PADDING OFF
51      GO
52      /****** Object:  Table [dbo].[tbl_EmployeeCourse_XREF]    Script Date: 10/13/2012 09:55:02 ******/
53      SET ANSI_NULLS ON
54      GO
55      SET QUOTED_IDENTIFIER ON
56      GO
57      IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[tbl_EmployeeCourse_XREF]')
AND type in (N'U'))
58      BEGIN
59      CREATE TABLE [dbo].[tbl_EmployeeCourse_XREF](
60          [FK_EmployeeID] [int] NOT NULL,
61          [FK_CourseID] [int] NOT NULL,
62          [DateCompleted] [datetime] NOT NULL,
63          [Grade] [float] NOT NULL
64      ) ON [PRIMARY]
65      END
66      GO
67      /****** Object:  ForeignKey [FK_tbl_EmployeeCourse_XREF_tbl_Course_LU]    Script Date: 10/13/2012
09:55:02 ******/
68      IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_tbl_EmployeeCourse_XREF_tbl_Course_LU]') AND parent_object_id =
OBJECT_ID(N'[dbo].[tbl_EmployeeCourse_XREF]'))
69      ALTER TABLE [dbo].[tbl_EmployeeCourse_XREF]  WITH CHECK ADD  CONSTRAINT
[FK_tbl_EmployeeCourse_XREF_tbl_Course_LU] FOREIGN KEY([FK_CourseID])
70      REFERENCES [dbo].[tbl_Course_LU] ([CourseID])
71      ON UPDATE CASCADE
72      ON DELETE CASCADE
73      GO
74      IF  EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_tbl_EmployeeCourse_XREF_tbl_Course_LU]') AND parent_object_id =
OBJECT_ID(N'[dbo].[tbl_EmployeeCourse_XREF]'))
75      ALTER TABLE [dbo].[tbl_EmployeeCourse_XREF] CHECK CONSTRAINT [FK_tbl_EmployeeCourse_XREF_tbl_Course_LU]
76      GO
77      /****** Object:  ForeignKey [FK_tbl_EmployeeCourse_XREF_tbl_Employee]    Script Date: 10/13/2012
09:55:02 ******/
78      IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_tbl_EmployeeCourse_XREF_tbl_Employee]') AND parent_object_id =
OBJECT_ID(N'[dbo].[tbl_EmployeeCourse_XREF]'))
79      ALTER TABLE [dbo].[tbl_EmployeeCourse_XREF]  WITH CHECK ADD  CONSTRAINT
[FK_tbl_EmployeeCourse_XREF_tbl_Employee] FOREIGN KEY([FK_EmployeeID])
80      REFERENCES [dbo].[tbl_Employee] ([EmployeeID])
81      ON UPDATE CASCADE
82      ON DELETE CASCADE
83      GO
84      IF  EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_tbl_EmployeeCourse_XREF_tbl_Employee]') AND parent_object_id =
OBJECT_ID(N'[dbo].[tbl_EmployeeCourse_XREF]'))
85      ALTER TABLE [dbo].[tbl_EmployeeCourse_XREF] CHECK CONSTRAINT [FK_tbl_EmployeeCourse_XREF_tbl_Employee]
86      GO
```

Referring to example 8.5 — I've let the long lines wrap around to the next line. Note that line 1 contains the `USE [EmployeeTraining]` command specifying that this script must be run at the EmployeeTraining database level.

Step 3: Segregate Scripts Into Related Sections

At this point, I consider the scripts generated in Step 2 above to be raw scripts. They need to be further segregated to clarify their purpose and to ensure they will execute properly. For example, I will segregate the scripts such that they can be called according to their function, beginning with the DropDatabase.sql script. A rough outline of the order I want to call my scripts goes something like this:

```
DropDatabase.sql
DropLogin.sql
CreateDatabase.sql
```



```
CreateLogin.sql
CreateUser.sql
CreateTables.sql
```

So your job in Step 3 is to edit the scripts and create new scripts as necessary to achieve this ordering. Note that the User has a dependency on the Login so the CreateLogin.sql script must execute before the CreateUser.sql script.

I will start by creating separate scripts that Drop and Create the database respectively. These will be named, naturally enough, DropDatabase.sql and CreateDatabase.sql and are listed in examples 8.6 and 8.7.

8.6 DropDatabase.sql

```
1       USE [master]
2       GO
3
4       /****** Object:  Database [EmployeeTraining]    Script Date: 09/23/2012 10:34:43 ******/
5       IF  EXISTS (SELECT name FROM sys.databases WHERE name = N'EmployeeTraining')
6       DROP DATABASE [EmployeeTraining]
7       GO
```

8.7 CreateDatabase.sql

```
1       USE [master]
2       GO
3
4       /****** Object:  Database [EmployeeTraining]    Script Date: 09/23/2012 10:34:43 ******/
5       CREATE DATABASE [EmployeeTraining] ON  PRIMARY
6       ( NAME = N'EmployeeTraining', FILENAME = N'C:\database\mssql\EmployeeTraining\EmployeeTraining.mdf' ,
SIZE = 2048KB , MAXSIZE = UNLIMITED, FILEGROWTH = 1024KB )
7        LOG ON
8       ( NAME = N'EmployeeTraining_log', FILENAME =
N'C:\database\mssql\EmployeeTraining\EmployeeTraining_log.ldf' , SIZE = 1024KB , MAXSIZE = 2048GB , FILEGROWTH =
10%)
9       GO
10
11      ALTER DATABASE [EmployeeTraining] SET COMPATIBILITY_LEVEL = 100
12      GO
13
14      IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
15      begin
16      EXEC [EmployeeTraining].[dbo].[sp_fulltext_database] @action = 'enable'
17      end
18      GO
19
20      ALTER DATABASE [EmployeeTraining] SET ANSI_NULL_DEFAULT OFF
21      GO
22
23      ALTER DATABASE [EmployeeTraining] SET ANSI_NULLS OFF
24      GO
25
26      ALTER DATABASE [EmployeeTraining] SET ANSI_PADDING OFF
27      GO
28
29      ALTER DATABASE [EmployeeTraining] SET ANSI_WARNINGS OFF
30      GO
31
32      ALTER DATABASE [EmployeeTraining] SET ARITHABORT OFF
33      GO
34
35      ALTER DATABASE [EmployeeTraining] SET AUTO_CLOSE OFF
36      GO
37
38      ALTER DATABASE [EmployeeTraining] SET AUTO_CREATE_STATISTICS ON
39      GO
40
41      ALTER DATABASE [EmployeeTraining] SET AUTO_SHRINK OFF
42      GO
43
44      ALTER DATABASE [EmployeeTraining] SET AUTO_UPDATE_STATISTICS ON
45      GO
46
47      ALTER DATABASE [EmployeeTraining] SET CURSOR_CLOSE_ON_COMMIT OFF
48      GO
49
50      ALTER DATABASE [EmployeeTraining] SET CURSOR_DEFAULT  GLOBAL
51      GO
52
53      ALTER DATABASE [EmployeeTraining] SET CONCAT_NULL_YIELDS_NULL OFF
54      GO
55
56      ALTER DATABASE [EmployeeTraining] SET NUMERIC_ROUNDABORT OFF
57      GO
58
```



```
59      ALTER DATABASE [EmployeeTraining] SET QUOTED_IDENTIFIER OFF
60      GO
61
62      ALTER DATABASE [EmployeeTraining] SET RECURSIVE_TRIGGERS OFF
63      GO
64
65      ALTER DATABASE [EmployeeTraining] SET  DISABLE_BROKER
66      GO
67
68      ALTER DATABASE [EmployeeTraining] SET AUTO_UPDATE_STATISTICS_ASYNC OFF
69      GO
70
71      ALTER DATABASE [EmployeeTraining] SET DATE_CORRELATION_OPTIMIZATION OFF
72      GO
73
74      ALTER DATABASE [EmployeeTraining] SET TRUSTWORTHY OFF
75      GO
76
77      ALTER DATABASE [EmployeeTraining] SET ALLOW_SNAPSHOT_ISOLATION OFF
78      GO
79
80      ALTER DATABASE [EmployeeTraining] SET PARAMETERIZATION SIMPLE
81      GO
82
83      ALTER DATABASE [EmployeeTraining] SET READ_COMMITTED_SNAPSHOT OFF
84      GO
85
86      ALTER DATABASE [EmployeeTraining] SET HONOR_BROKER_PRIORITY OFF
87      GO
88
89      ALTER DATABASE [EmployeeTraining] SET  READ_WRITE
90      GO
91
92      ALTER DATABASE [EmployeeTraining] SET RECOVERY FULL
93      GO
94
95      ALTER DATABASE [EmployeeTraining] SET  MULTI_USER
96      GO
97
98      ALTER DATABASE [EmployeeTraining] SET PAGE_VERIFY CHECKSUM
99      GO
100
101     ALTER DATABASE [EmployeeTraining] SET DB_CHAINING OFF
102     GO
```

Referring to example 8.7 — Note that on lines 6 and 8 I've edited the path to the database and log files. I'm going to create a separate folder on my C: drive named *database*. In that folder I am creating a subfolder named *mssql*, and in that folder I'm creating a subfolder named *EmployeeTraining*. That is where I will create the EmployeeTraining.mdf and EmployeeTraining_log.ldf files.

The next set of scripts to create is the DropLogin.sql and CreateLogin.sql scripts, which are listed in examples 8.8 and 8.9 respectively.

8.8 DropLogin.sql

```
1       /****** Object:  Login [EmployeeTraining]    Script Date: 10/13/2012 08:52:28 ******/
2       IF  EXISTS (SELECT * FROM sys.server_principals WHERE name = N'EmployeeTraining')
3       DROP LOGIN [EmployeeTraining]
4       GO
```

8.9 CreateLogin.sql

```
1       /****** Object:  Login [EmployeeTraining]    Script Date: 10/13/2012 08:52:28 ******/
2       CREATE LOGIN [EmployeeTraining] WITH PASSWORD=N'password', DEFAULT_DATABASE=[EmployeeTraining],
DEFAULT_LANGUAGE=[us_english], CHECK_EXPIRATION=OFF, CHECK_POLICY=OFF
3       GO
4
5       EXEC sys.sp_addsrvrolemember @loginame = N'EmployeeTraining', @rolename = N'dbcreator'
6       GO
```

The next script will be the CreateUser.sql script, which is given in example 8.10.

8.10 CreateUser.sql

```
1       USE [EmployeeTraining]
2       GO
3
4       /****** Object:  User [EmployeeTrainingUser]    Script Date: 10/13/2012 08:30:07 ******/
5       GO
6
7       CREATE USER [EmployeeTrainingUser] FOR LOGIN [EmployeeTraining] WITH DEFAULT_SCHEMA=[dbo]
8       GO
```

Finally, the CreateTables.sql script is given in example 8.11.

8.11 CreateTables.sql

```
1       USE [EmployeeTraining]
2       GO
3       /****** Object:  Table [dbo].[tbl_Employee]    Script Date: 10/13/2012 09:55:02 ******/
4       SET ANSI_NULLS ON
5       GO
6       SET QUOTED_IDENTIFIER ON
7       GO
8       SET ANSI_PADDING ON
9       GO
10      IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[tbl_Employee]') AND type
in (N'U'))
11      BEGIN
12      CREATE TABLE [dbo].[tbl_Employee](
13          [EmployeeID] [int] IDENTITY(1,1) NOT NULL,
14          [FirstName] [varchar](50) NOT NULL,
15          [MiddleName] [varchar](50) NOT NULL,
16          [LastName] [varchar](50) NOT NULL,
17          [Birthday] [datetime] NOT NULL,
18          [Picture] [varbinary](max) NULL,
19          [HireDate] [datetime] NOT NULL,
20       CONSTRAINT [PK_tbl_Employee] PRIMARY KEY CLUSTERED
21      (
22          [EmployeeID] ASC
23      )WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON,
ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
24      ) ON [PRIMARY]
25      END
26      GO
27      SET ANSI_PADDING OFF
28      GO
29      /****** Object:  Table [dbo].[tbl_Course_LU]    Script Date: 10/13/2012 09:55:02 ******/
30      SET ANSI_NULLS ON
31      GO
32      SET QUOTED_IDENTIFIER ON
33      GO
34      SET ANSI_PADDING ON
35      GO
36      IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[tbl_Course_LU]') AND type
in (N'U'))
37      BEGIN
38      CREATE TABLE [dbo].[tbl_Course_LU](
39          [CourseID] [int] IDENTITY(1,1) NOT NULL,
40          [Code] [varchar](6) NOT NULL,
41          [Title] [varchar](200) NOT NULL,
42          [Description] [varchar](1000) NOT NULL,
43       CONSTRAINT [PK_tbl_Course_LU] PRIMARY KEY CLUSTERED
44      (
45          [CourseID] ASC
46      )WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON,
ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
47      ) ON [PRIMARY]
48      END
49      GO
50      SET ANSI_PADDING OFF
51      GO
52      /****** Object:  Table [dbo].[tbl_EmployeeCourse_XREF]    Script Date: 10/13/2012 09:55:02 ******/
53      SET ANSI_NULLS ON
54      GO
55      SET QUOTED_IDENTIFIER ON
56      GO
57      IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[tbl_EmployeeCourse_XREF]')
AND type in (N'U'))
58      BEGIN
59      CREATE TABLE [dbo].[tbl_EmployeeCourse_XREF](
60          [FK_EmployeeID] [int] NOT NULL,
61          [FK_CourseID] [int] NOT NULL,
62          [DateCompleted] [datetime] NOT NULL,
63          [Grade] [float] NOT NULL
64      ) ON [PRIMARY]
65      END
66      GO
67      /****** Object:  ForeignKey [FK_tbl_EmployeeCourse_XREF_tbl_Course_LU]    Script Date: 10/13/2012
09:55:02 ******/
68      IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_tbl_EmployeeCourse_XREF_tbl_Course_LU]') AND parent_object_id =
OBJECT_ID(N'[dbo].[tbl_EmployeeCourse_XREF]'))
69      ALTER TABLE [dbo].[tbl_EmployeeCourse_XREF]  WITH CHECK ADD  CONSTRAINT
[FK_tbl_EmployeeCourse_XREF_tbl_Course_LU] FOREIGN KEY([FK_CourseID])
70      REFERENCES [dbo].[tbl_Course_LU] ([CourseID])
```



```
71       ON UPDATE CASCADE
72       ON DELETE CASCADE
73       GO
74       IF  EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_tbl_EmployeeCourse_XREF_tbl_Course_LU]') AND parent_object_id =
OBJECT_ID(N'[dbo].[tbl_EmployeeCourse_XREF]'))
75       ALTER TABLE [dbo].[tbl_EmployeeCourse_XREF] CHECK CONSTRAINT [FK_tbl_EmployeeCourse_XREF_tbl_Course_LU]
76       GO
77       /****** Object:  ForeignKey [FK_tbl_EmployeeCourse_XREF_tbl_Employee]    Script Date: 10/13/2012
09:55:02 ******/
78       IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_tbl_EmployeeCourse_XREF_tbl_Employee]') AND parent_object_id =
OBJECT_ID(N'[dbo].[tbl_EmployeeCourse_XREF]'))
79       ALTER TABLE [dbo].[tbl_EmployeeCourse_XREF]  WITH CHECK ADD  CONSTRAINT
[FK_tbl_EmployeeCourse_XREF_tbl_Employee] FOREIGN KEY([FK_EmployeeID])
80       REFERENCES [dbo].[tbl_Employee] ([EmployeeID])
81       ON UPDATE CASCADE
82       ON DELETE CASCADE
83       GO
84       IF  EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_tbl_EmployeeCourse_XREF_tbl_Employee]') AND parent_object_id =
OBJECT_ID(N'[dbo].[tbl_EmployeeCourse_XREF]'))
85       ALTER TABLE [dbo].[tbl_EmployeeCourse_XREF] CHECK CONSTRAINT [FK_tbl_EmployeeCourse_XREF_tbl_Employee]
86       GO
```

Referring to example 8.11 — This script is the same as the raw script since no editing or segregation was necessary.

Step 4: Organize Scripts According To Release

In this step, you need to gather up the scripts you created in Step 3 above and put them in the release folder you created in Step 1. The release folder should have been created in the EmployeeTrainingProject\trunk\database folder as is shown in figure 8.12.

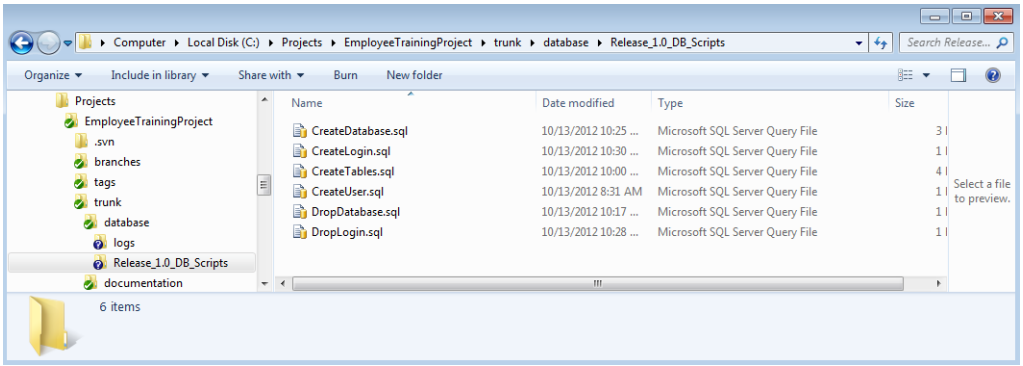

Figure 8-12: Scripts Placed in Release 1.0 DB Scripts Folder

Step 5: Create Test Data Generation Scripts

What good is a database without data? You can create a script that automatically inserts data into tables to support testing. I will create such a script and name it InsertTestData.sql, which is given in example 8.12.

8.12 InsertTestData.sql

```
1        USE [EmployeeTraining]
2        GO
3        /* Insert test data into tbl_Employee */
4        INSERT INTO dbo.tbl_Employee (FirstName, MiddleName, LastName, Birthday, HireDate)
5               VALUES ('Rick', 'Warren', 'Miller', '1/1/1970 12:00:00 AM', '12/1/1998 12:00:00 AM');
6        INSERT INTO dbo.tbl_Employee (FirstName, MiddleName, LastName, Birthday, HireDate)
7               VALUES ('Coralie', 'Sylvia', 'Miller', '1/1/1975 12:00:00 AM', '08/21/2001 12:00:00 AM');
8        INSERT INTO dbo.tbl_Employee (FirstName, MiddleName, LastName, Birthday, HireDate)
9               VALUES ('Steve', 'Jacob', 'Hester', '09/12/1986 12:00:00 AM', '07/07/2003 12:00:00 AM');
10       INSERT INTO dbo.tbl_Employee (FirstName, MiddleName, LastName, Birthday, HireDate)
11              VALUES ('Nancy', 'Jo', 'Coats', '1/1/1961 12:00:00 AM', '12/1/1998 12:00:00 AM');
12       INSERT INTO dbo.tbl_Employee (FirstName, MiddleName, LastName, Birthday, HireDate)
13              VALUES ('William', 'Brent', 'Darby', '11/12/1969 12:00:00 AM', '04/04/1986 12:00:00 AM');
14
```

```
15      /* Insert test data into tbl_tbl_Course_LU */
16      INSERT INTO dbo.tbl_Course_LU (Code, Title, Description)
17             VALUES ('IST101', 'Introduction to Programming', 'Description test here...');
18      INSERT INTO dbo.tbl_Course_LU (Code, Title, Description)
19             VALUES ('IST102', 'Intermediate Programming', 'Description test here...');
20      INSERT INTO dbo.tbl_Course_LU (Code, Title, Description)
21             VALUES ('PMP101', 'Introduction to Program Management', 'Description test here...');
22      INSERT INTO dbo.tbl_Course_LU (Code, Title, Description)
23             VALUES ('PMP102', 'Intermediate Program Management', 'Description test here...');
24      INSERT INTO dbo.tbl_Course_LU (Code, Title, Description)
25             VALUES ('PMP103', 'Earned Value Management', 'Description test here...');
26
27      /* Insert test data into tbl_EmployeeCourse_XREF */
28      INSERT INTO dbo.tbl_EmployeeCourse_XREF (FK_EmployeeID, FK_CourseID, DateCompleted, Grade)
29             VALUES (1, 1, '1/1/2012 12:00:00 AM', '4.0');
30      INSERT INTO dbo.tbl_EmployeeCourse_XREF (FK_EmployeeID, FK_CourseID, DateCompleted, Grade)
31             VALUES (1, 2, '1/1/2012 12:00:00 AM', '4.0');
32      INSERT INTO dbo.tbl_EmployeeCourse_XREF (FK_EmployeeID, FK_CourseID, DateCompleted, Grade)
33             VALUES (2, 3, '1/1/2012 12:00:00 AM', '4.0');
34      INSERT INTO dbo.tbl_EmployeeCourse_XREF (FK_EmployeeID, FK_CourseID, DateCompleted, Grade)
35             VALUES (2, 4, '1/1/2012 12:00:00 AM', '4.0');
36      INSERT INTO dbo.tbl_EmployeeCourse_XREF (FK_EmployeeID, FK_CourseID, DateCompleted, Grade)
37             VALUES (3, 1, '1/1/2012 12:00:00 AM', '4.0');
```

Referring to example 8.12 — Note that before you can insert data into the tbl_EmployeeCourse_XREF table you must have data in both the tbl_Employee and tbl_Course_LU tables. Also, you can safely omit data from fields where null values are acceptable. In the tbl_Employee table I have not inserted a picture because that field allows null values.

Step 6: Create DOS Batch File To Execute Scripts Automatically

The next thing to do is to create a DOS batch file that will execute all the SQL scripts automatically. I'll create two batch files: the first batch file will be named Set_Environment.bat and will be used to setup the logs directory environment variable and to create the logs and database directory structures. The second batch file will be used to execute the database scripts. Examples 8.13 and 8.14 lists the Set_Environment.bat and CreateRelease_1.0_Database.bat files respectively.

8.13 Set_Environment.bat

```
1       rem set DB_LOGS environment variable for use in batch files
2       set DB_LOGS=logs
3       rem create logs directory
4       if not exist %DB_LOGS% mkdir %DB_LOGS%
5
6       rem create database directories for deployment
7       if not exist c:\database mkdir c:\database
8       if not exist c:\database\mssql mkdir c:\database\mssql
9       if not exist c:\database\mssql\EmployeeTraining mkdir c:\database\mssql\EmployeeTraining
```

Referring to example 8.13 — Note that the DB_LOGS variable will be used in subsequent batch files to provide a path to the logs directory. Also, the path to the database directory should match that in the CreateDatabase.sql script.

8.14 CreateRelease_1.0_Database.bat

```
1       @echo off
2
3       echo.
4
5       echo Setting up environment parameters...
6       call Set_Environment.bat
7
8       echo Dropping EmployeeTraining Database...
9       sqlcmd -i "Release_1.0_DB_Scripts\DropDatabase.sql" -b -o %DB_LOGS%\DropDatabaseOutput.txt
10      echo Dropping EmployeeTraining Login...
11      sqlcmd -i "Release_1.0_DB_Scripts\DropLogin.sql" -b -o %DB_LOGS%\DropLoginOutput.txt
12      echo Creating EmployeeTraining Database...
13      sqlcmd -i "Release_1.0_DB_Scripts\CreateDatabase.sql" -b -o %DB_LOGS%\CreateDatabaseOutput.txt
14      echo Creating EmployeeTraining Login...
15      sqlcmd -i "Release_1.0_DB_Scripts\CreateLogin.sql" -b -o %DB_LOGS%\CreateLoginOutput.txt
16      echo Creating EmployeeTraining User
17      sqlcmd -i "Release_1.0_DB_Scripts\CreateUser.sql" -b -o %DB_LOGS%\CreateUserOutput.txt
18      echo Creating EmployeeTraining Database Tables...
19      sqlcmd -i "Release_1.0_DB_Scripts\CreateTables.sql" -b -o %DB_LOGS%\CreateTablesOutput.txt
20      echo Inserting test data...
21      sqlcmd -i "Release_1.0_DB_Scripts\InsertTestData.sql" -b -o %DB_LOGS%\InsertTestDataOutput.txt
```

Referring to examples 8.13 and 8.14 — These two batch files are placed in the trunk\database directory. Your trunk\database directory should now look similar to figure 8.13.

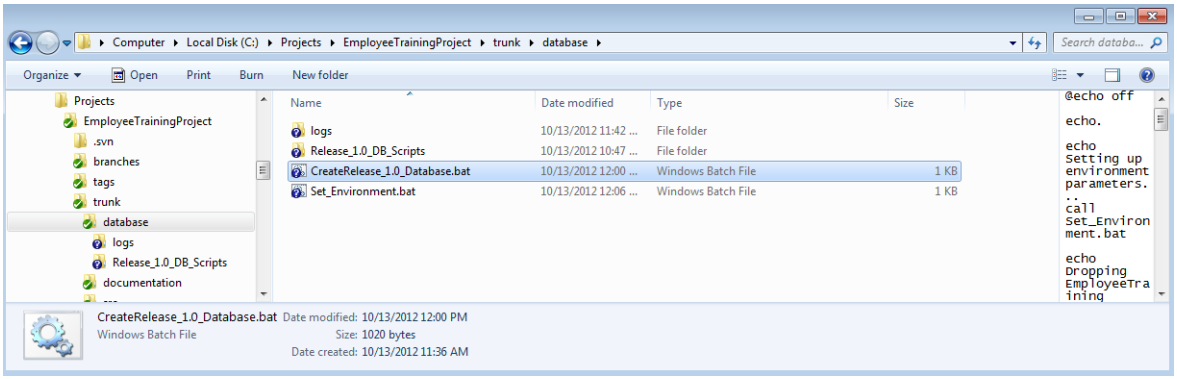

Figure 8-13: Contents of trunk\database Folder

Step 7: Run Release Script

All that's left now is to test the scripts. You can double-click the CreateRelease_1.0_Database.bat file or open a command console window and execute it by typing its name. As the script executes you should see the comments echoed to the console as is shown in figure 8.14.

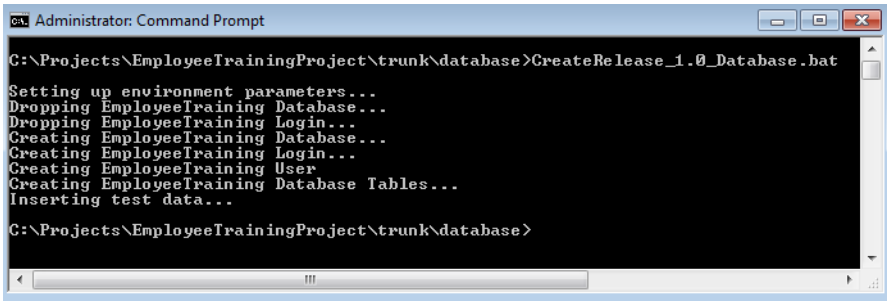

Figure 8-14: Results of Executing CreateRelease_1.0_Database.bat File

Referring to figure 8-14 — When the script finishes executing you can check the contents of each log file located in the logs folder. The output to each log file should be minimal. If an sql file contained a USE statement on line one then you'll see a line in the log file that says something like: "Changed database context to 'master'" or "Changed database context to 'EmployeeTraining'". Otherwise, if there's an error, you will see more output indicating on what line of the script the error occurred.

If all goes well your database will be dropped and recreated in a matter of seconds. You can verify everything is as it should be by opening SQL Server Management Studio and inspecting the database. Note that if you open the Database Diagrams folder you'll have to create a new diagram and add the existing tables to it as is shown in figure 8.15.

Referring to figure 8-15 — Simply select all the tables you wish to add to the diagram and click the **Add** button. Click the **Close** button to dismiss the Add Table dialog and inspect the database diagram for completeness.

Commit Tested Scripts To Subversion

Once you've tested the scripts and verified that everything works like it should, add everything but the logs folder to the subversion repository. If you look at the contents of the trunk\database folder, you'll notice that Tortoise has placed a purple circle with a white question mark next to each artifact that has not yet been placed in the repository. To add your new database scripts, first select the Release_1.0_DB_Scripts folder and the two batch files and right-click and select **TortoiseSVN -> Add** as is shown in figure 8.16.

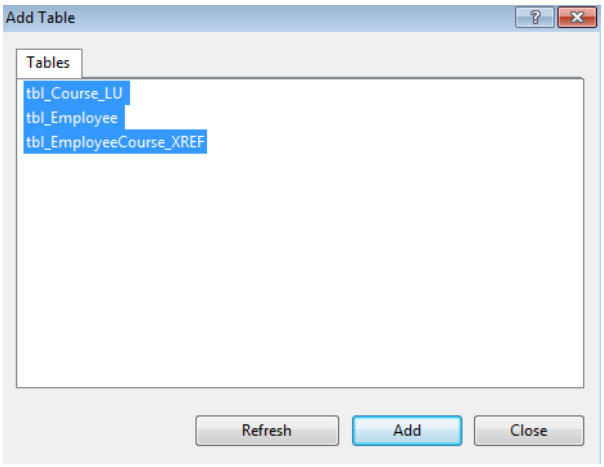

Figure 8-15: Adding Tables to a Database Diagram

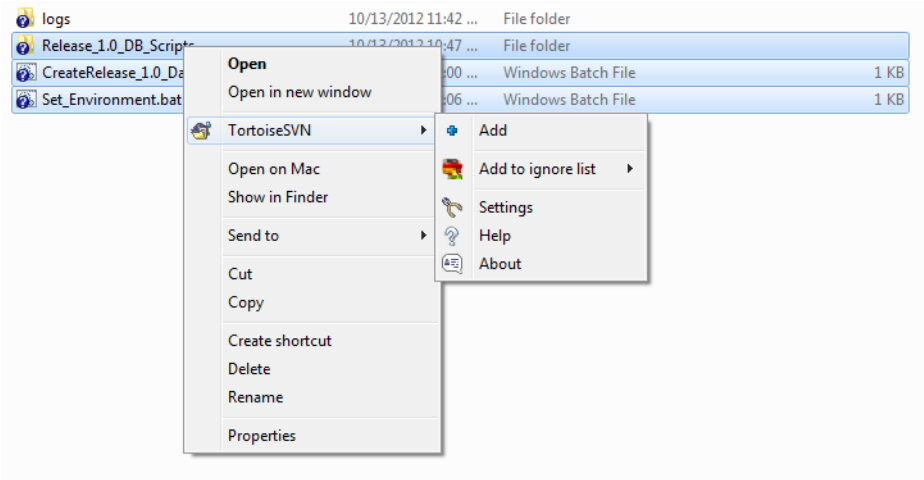

Figure 8-16: Right-Click Selected Artifacts and Select TortoiseSVN -> Add

Referring to figure 8-16 — Select **TortoiseSVN -> Add** to bring up the Tortoise Add dialog as is shown in figure 8.17.

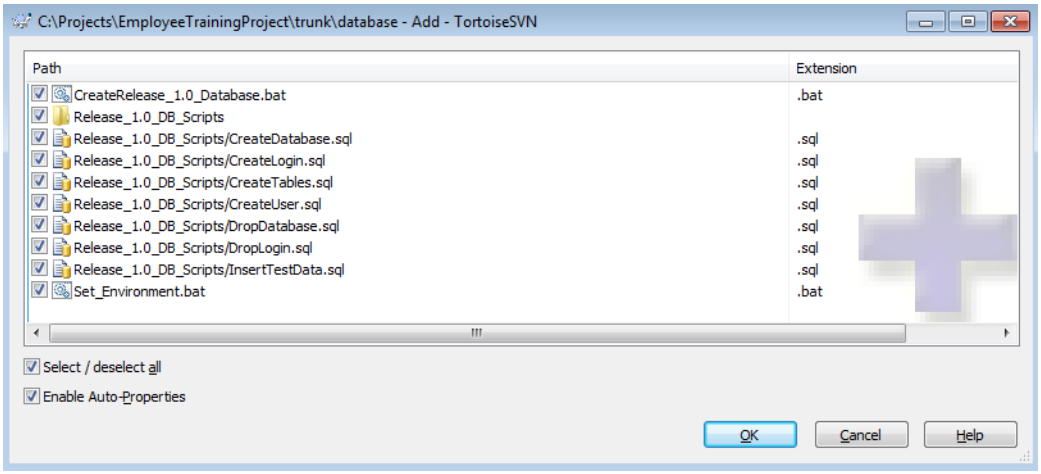

Figure 8-17: Tortoise Add Dialog

Referring to figure 8-17 — Inspect the Add dialog to ensure you're adding the right scripts and not the log files. When you're ready click the **OK** button. If everything goes well the Add dialog will look similar to figure 8-18.

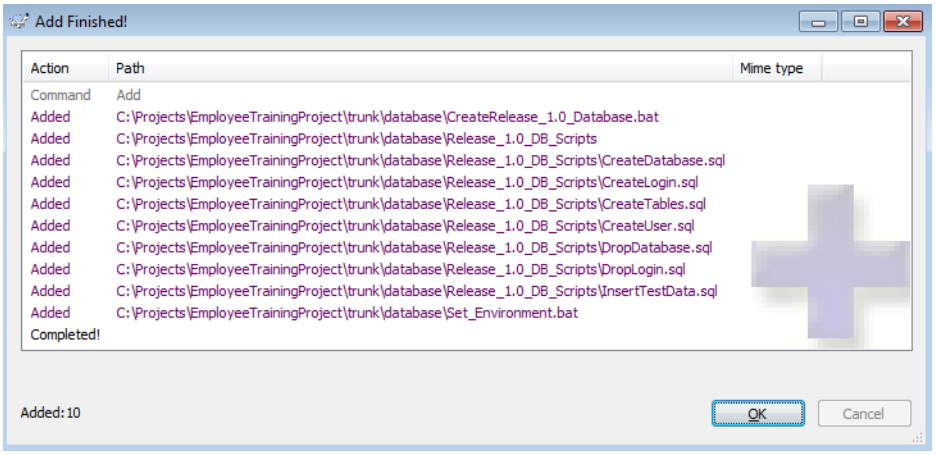

Figure 8-18: Add Completed Successfully - Now It's Time to Commit

Referring to figure 8-18 — When the add completes successfully, click the **OK** button. The artifacts you just added will now be adorned with a blue plus sign. Next, you need to *commit* the scripts you just added to the repository. To do this, select the artifacts you wish to commit then right-click and select **SVN Commit...** from the pop-up menu as is shown in figure 8-19. This will bring up the Commit dialog as is shown in figure 8-20.

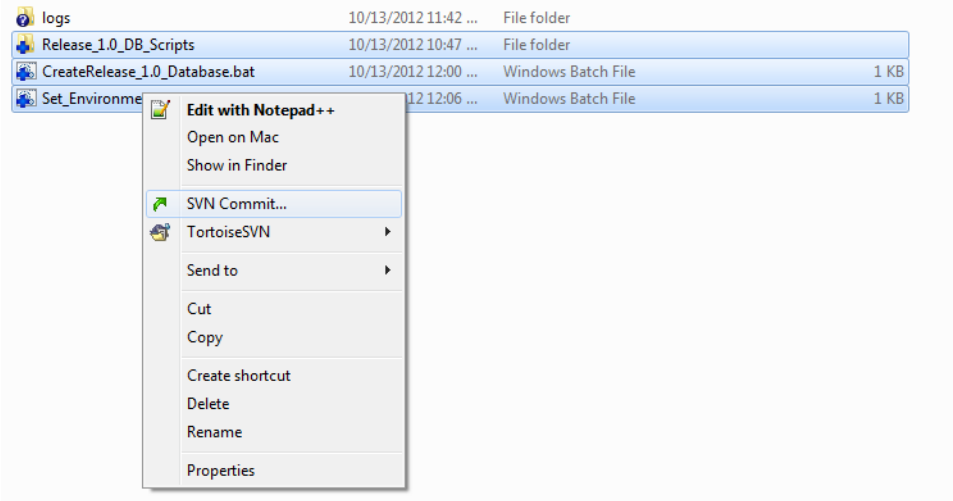

Figure 8-19: Committing Scripts to Repository

Referring to figure 8-20 — Add a comment in the Message box and when ready click the **OK** button. The results of the commit operation will look similar to figure 8-21. Click the **OK** button to close the Commit dialog. Committed artifacts will now be adorned with a green circle containing a white check mark.

Where To Go From Here

The design of the EmployeeTraining database will most certainly evolve during the course of this book as new requirements are discovered. The same holds true for real-world projects. Database scripts such as the ones created in this chapter allow developers to better manage the database design and evolution process.

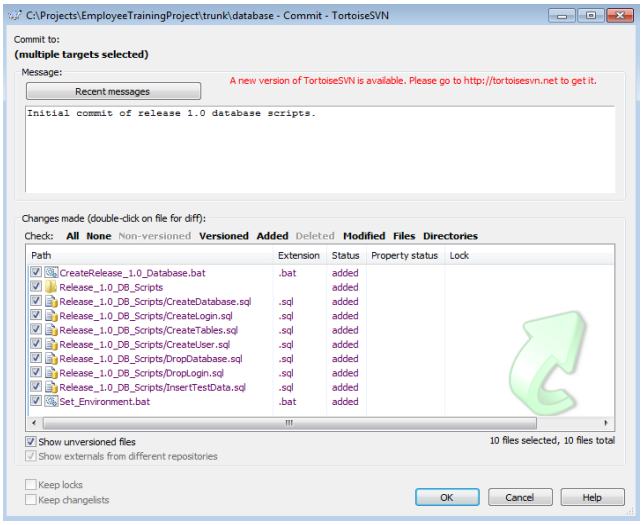

Figure 8-20: TortoiseSVN Commit Dialog - Add Comments Each Time You Commit

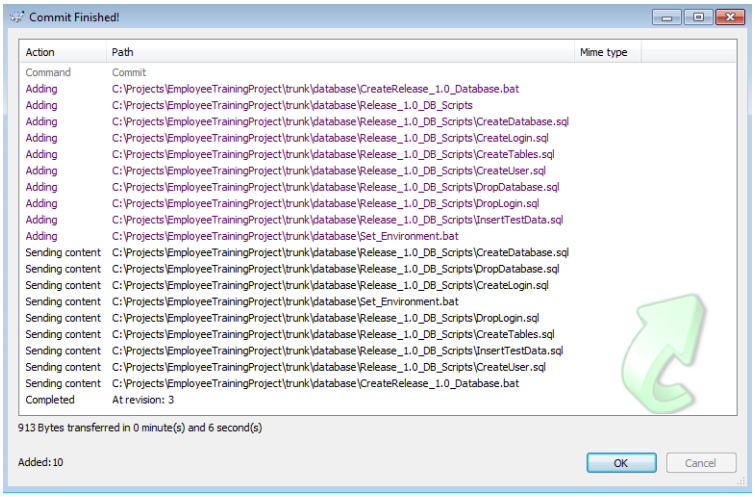

Figure 8-21: Commit Results

When the time comes to evolve the database I will show you how to incorporate those changes into a new set of release scripts. I'll show you how to call one release batch file from another on a technique I call *batch file chaining*. It sounds complicated but when you see an example you'll slap your forehead and say "Ah ha!"

Summary

The iterative evolution of a database design can be better managed with scripts. Database scripts let you drop and recreate the database in a matter of seconds. Database scripts can also be committed to a Subversion repository and kept under configuration management.

When converting a database to scripts follow the seven-step procedure outlined in this chapter:

Step 1: Create a release folder and a logs folder in your Subversion project's \trunk\database folder.

Step 2: Use SQL Server Management Studio to generate database scripts for a selected database. This step has multiple sub-steps.

Step 3: Segregate the scripts into logically related sections and save as individual SQL script files. You'll have one script to drop the database, another script to create the database, and still another script to create the tables, etc.

Step 4: Organize the database scripts according to release. Group related database scripts in their respective release folders.

Step 5: Create additional scripts to automatically insert test data into the database for development and testing.

Step 6: Create a DOS batch file to execute the scripts automatically. Name these batch files according to release. As the database design evolves over the life of the project you'll create a chain of these batch files so that you need only execute the latest release batch file, which will call the previous release batch file, and so on.

Step 7: Run a release script each time you want to recreate the database. This is done frequently during development and testing.

When you drop and recreate the database with your scripts you'll need to recreate any database diagrams you had created before dropping the database. All you need to do is create a new diagram, add the required tables from a list of tables in the database, and arrange them for clarity.

Once you have created your first set of scripts you'll only need to add scripts here and there that modify the database design.

References

John Paul Mueller, Windows Command Line Administration Instant Reference, Sybex, 2010, ISBN: 9780470650462.

Microsoft sqlcmd Utility online reference: http://msdn.microsoft.com/en-us/library/ms162773.aspx

Microsoft SQL Server Management Studio online reference: http://msdn.microsoft.com/en-us/library/ms162773.aspx

Notes

