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.

anize • Include in library •	Share with Burn New folder		8=	- •
EmployeeTrainingProject	^ Name	Date modified	Туре	
"svn	Ø logs	9/23/2012 8:05 AM	File folder	
M branches	Release_1.0_DB_Scripts	9/23/2012 8:05 AM	File folder	
💋 tags	E			Select a fi
S trunk				preview
atabase				
🛃 utile				
ouis 🖉	·			

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 /***** Object: Database [EmployeeTraining] Script Date: 09/23/2012 10:34:43 *****/ 4 IF EXISTS (SELECT name FROM sys.databases WHERE name = N'EmployeeTraining') 5 DROP DATABASE [EmployeeTraining] 6 7 GO 8 USE [master] 9 10 GO 11 12 /***** Object: Database [EmployeeTraining] Script Date: 09/23/2012 10:34:43 *****/ CREATE DATABASE [EmployeeTraining] ON PRIMARY 13 14 (NAME = N'EmployeeTraining', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\EmployeeTraining.mdf', SIZE = 2048KB, MAXSIZE = UNLIMITED, FILEGROWTH = 1024 KB) 15 LOG ON 16 (NAME = N'EmployeeTraining log', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\EmployeeTraining_log.Idf' , SIZE = 1024KB , MAXSIZE = 2048GB , FILEGROWTH = 10%17 GO 18 ALTER DATABASE [EmployeeTraining] SET COMPATIBILITY LEVEL = 100 19 20 GO 21 IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled')) 22 23 begin 24 EXEC [EmployeeTraining].[dbo].[sp_fulltext_database] @action = 'enable' 25 end 26 GO 27 ALTER DATABASE [EmployeeTraining] SET ANSI_NULL_DEFAULT OFF 28 29 GO 30 31 ALTER DATABASE [EmployeeTraining] SET ANSI_NULLS OFF 32 GO 33 ALTER DATABASE [EmployeeTraining] SET ANSI_PADDING OFF 34 35 GO 36 ALTER DATABASE [EmployeeTraining] SET ANSI_WARNINGS OFF 37 38 GO 39 ALTER DATABASE [EmployeeTraining] SET ARITHABORT OFF 40 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 ALTER DATABASE [EmployeeTraining] SET AUTO_SHRINK OFF 49 50 GO 51 ALTER DATABASE [EmployeeTraining] SET AUTO_UPDATE_STATISTICS ON 52 53 GO 54 55 ALTER DATABASE [EmployeeTraining] SET CURSOR_CLOSE_ON_COMMIT OFF 56 GO 57 ALTER DATABASE [EmployeeTraining] SET CURSOR_DEFAULT GLOBAL 58 59 GO 60 61 ALTER DATABASE [EmployeeTraining] SET CONCAT NULL YIELDS NULL OFF 62 GO 63 ALTER DATABASE [EmployeeTraining] SET NUMERIC_ROUNDABORT OFF 64 65 GO 66 ALTER DATABASE [EmployeeTraining] SET QUOTED_IDENTIFIER OFF 67 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 ALTER DATABASE [EmployeeTraining] SET ALLOW SNAPSHOT ISOLATION OFF 85 86 GO 87 ALTER DATABASE [EmployeeTraining] SET PARAMETERIZATION SIMPLE 88 89 GO 90 ALTER DATABASE [EmployeeTraining] SET READ_COMMITTED_SNAPSHOT OFF 91 92 GO 93 94 ALTER DATABASE [EmployeeTraining] SET HONOR_BROKER_PRIORITY OFF 95 GO 96 ALTER DATABASE [EmployeeTraining] SET READ WRITE 97 98 GO 99 ALTER DATABASE [EmployeeTraining] SET RECOVERY FULL 100 101 GO 102 103 ALTER DATABASE [EmployeeTraining] SET MULTI USER 104 GO 105 ALTER DATABASE [EmployeeTraining] SET PAGE_VERIFY CHECKSUM 106 107 GO 108 ALTER DATABASE [EmployeeTraining] SET DB CHAINING OFF 109 110 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.

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

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.

```
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 G0
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 *****/
```

8.4 DropAndCreateLogin.sql (edited)

8.3 DropAndCreateLogin.sql

```
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.

	New Database New Query Script Database as	Tables Views Synonyms Programmability	
	Tasks 🕨	Detach	
* • • • • • • • • • • • • • • • • • • •	Policies Facets Start PowerShell Reports Rename Delete Refresh Properties ver August Specifications Objects	Take Offline Bring Online Shrink Back Up Restore Mirror Launch Database Mirroring Monitor Ship Transaction Logs Generate Scripts	Select Generate Scripts
🚞 Replica 🚞 Manag 🎇 SQL Se	ation gement erver Agent	Extract Data-tier Application Register as Data-tier Application Import Data Export Data Copy Database Manage Database Encryption	

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.

S Generate and Publish Scripts	
Introduction	
Introduction	Help
Choese Objects Set Scripting Options Summary Save or Publish Scripts	Generate scripts for database objects. This vitrard generates a script of selected database objects. The scripts can be saved for later use in publishing Vets Service. There are four steps to complete this vizard: 3. Select database objects. 3. Review your selections. 4. Generate scripts, then save or publish them.
	To begin the script generation process, click Next. Do not show this page again. Process

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

7.

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

S Generate and Publish Scripts	
Choose Objects	
Introduction	Help
Choose Objects	Select the database objects to script.
Set Scripting Options	
Summary	Script entire database and all database objects
Save or Publish Scripts	Select specific database objects
	Image: Second state of the second s
	Select All Deselect All
	< Previous Next > Finish Cancel

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.

S Generate and Publish Scripts				
Set Scripting O	ptions			
Introduction			 Help 	
Choose Objects	Specify how scripts she	ould be saved or published.		
Set Scripting Options	Output Type			
Save or Publish Scripts	Save scripts to a sp	ecific location		
	Publish to Web served	ice		
	Save to file		Advanced	Click the Advanced button to
	Files to generate:	Single file		set advanced scripting options
		 Single file per object 		r o'r
	File name:	C:\Users\Administrator\Documents\s	cript.sql	
		✓ Overwrite existing file		
	Save as:	 Unicode text 		
		 ANSI text 		
	Save to Clipboard Save to pew guepr with Save to pew guepr with	ndaw		
	Save to new query wi	NOW Y		
		< Previous Next >	Enish Cancel	
		IVEAL 2		

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.

Options	
<mark></mark>	
∃ General	
ANSI Padding	True
Append to File	False
Continue scripting on Error	False
Convert UDDTs to Base Types	False
Generate Script for Dependent Objects	False
Include Descriptive Headers	True
Include If NOT EXISTS	True
Include system constraint names	False
Include unsupported statements	False
Schema qualify object names.	True
Script Bindings	False
Script Collation	False
Script Defaults	True
Script DROP and CREATE	Script CREATE
Script Extended Properties	True
Script for Server Version	SQL Server 2008 R2
Script for the database engine type	Stand-alone instance
Script Logins	False
Script Object-Level Permissions	False
Script Statistics	Do not script statistics
Script USE DATABASE	True
Types of data to script	Schema only
Table/View Options	
Script Change Tracking	False
Script Check Constraints	True
Script Data Compression Options	False
Script Foreign Keys	True
Script Full-Text Indexes	False
Script Indexes	False
Script Primary Keys	Irue
Script Triggers	False
Script Unique Keys	Irue
Script DROP and CREATE Script Object CREATE/DROP statements.	

Figure 8-9: Advanced Scripting Options Page

Referring to figure 8-9 - 1'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.

S Generate and Publish Scripts		
Summary		
Introduction		🕢 Help
Choose Objects	Review your selections.	
Set Scripting Options		
Ser Surgering Option's Surgering	Sever: AQUARIUS Sever: AQUARIUS Database: Employee Training Selected Objects Target Destination: Save to clipboard Options Grenal Table/New Options	
	< Previous Next > Finish	Cancel

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.

🖾 Generate and Publish Scripts		- • ×
Save or Publish	Scripts	
Introduction		🕢 Help
Choose Objects	Saving or publishing scripts.	
Set Scripting Options		
Summary	Action	Result
Save or Publish Scripts	Getting the list of objects from 'Employee Training'.	Success
	Preparing dbo.tbl_Course_LU	Success
	Preparing dbo.tbl_Employee	Success
	Preparing dbo.tbl_EmployeeCourse_XREF	Success
	Save to clipboard	Success
		Save Report
	< Previous Next > Finish	Cancel

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
         /****** Object: Table [dbo].[tbl_Employee] Script Date: 10/13/2012 09:55:02 *****/
3
        SET ANSI_NULLS ON
4
5
        GO
        SET QUOTED_IDENTIFIER ON
6
7
        GO
8
         SET ANSI_PADDING ON
9
        GO
        IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[tbl_Employee]') AND type
10
in (N'U'))
        BEGIN
11
12
        CREATE TABLE [dbo].[tbl Employee](
            [EmployeeID] [int] IDENTITY(1,1) NOT NULL,
13
             [FirstName] [varchar] (50) NOT NULL,
14
15
             [MiddleName] [varchar] (50) NOT NULL,
16
             [LastName] [varchar](50) NOT NULL,
17
             [Birthday] [datetime] NOT NULL,
             [Picture] [varbinary] (max) NULL,
18
             [HireDate] [datetime] NOT NULL,
19
         CONSTRAINT [PK tbl Employee] PRIMARY KEY CLUSTERED
20
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]
        ) ON [PRIMARY]
24
25
        END
26
        GO
27
        SET ANSI_PADDING OFF
28
        GO
         /****** Object: Table [dbo].[tbl_Course_LU] Script Date: 10/13/2012 09:55:02 ******/
29
        SET ANSI NULLS ON
30
31
        GO
        SET QUOTED_IDENTIFIER ON
32
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, [Code] [varchar](6) NOT NULL, 40 41 [Title] [varchar] (200) NOT NULL. [Description] [varchar] (1000) NOT NULL, 42 43 CONSTRAINT [PK_tbl_Course_LU] PRIMARY KEY CLUSTERED 44 (45 [CourseID] ASC)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, 46 ALLOW PAGE LOCKS = ON) ON [PRIMARY] 47) ON [PRIMARY] 48 END GO 49 SET ANSI_PADDING OFF 50 GO 51 /***** Object: Table [dbo].[tbl_EmployeeCourse_XREF] Script Date: 10/13/2012 09:55:02 ******/ 52 SET ANSI NULLS ON 53 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, [DateCompleted] [datetime] NOT NULL, 62 [Grade] [float] NOT NULL 63) ON [PRIMARY] 64 END 65 66 GO /***** Object: ForeignKey [FK_tbl_EmployeeCourse_XREF_tbl_Course_LU] 67 Script Date: 10/13/2012 09:55:02 *****/ IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id = 68 OBJECT_ID(N'[dbo].[FK_tbl_EmployeeCourse_XREF_tbl_Course_LU]') AND parent_object_id = OBJECT_ID(N'[dbo].[tbl_EmployeeCourse_XREF]')) 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 IF EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id = 74 OBJECT_ID(N'[dbo].[FK_tbl_EmployeeCourse_XREF_tbl_Course_LU]') AND parent_object_id = OBJECT_ID(N'[dbo].[tbl_EmployeeCourse_XREF]')) ALTER TABLE [dbo].[tbl_EmployeeCourse_XREF] CHECK CONSTRAINT [FK_tbl_EmployeeCourse_XREF_tbl Course LU] 75 76 GO /****** Object: ForeignKey [FK_tbl_EmployeeCourse_XREF_tbl_Employee] Script Date: 10/13/2012 77 09:55:02 *****/ IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id = 78 OBJECT_ID(N'[dbo].[FK_tbl_EmployeeCourse_XREF_tbl_Employee]') AND parent_object_id = OBJECT_ID(N'[dbo].[tbl_EmployeeCourse_XREF]')) 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 GO 83 IF EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id = 84 OBJECT_ID[N'[dbo].[FK_tbl_EmployeeCourse_XREF_tbl_Employee]') AND parent_object_id = OBJECT_ID[N'[dbo].[tbl EmployeeCourse_XREF]')) ALTER TABLE [dbo].[tbl_EmployeeCourse_XREF] CHECK CONSTRAINT [FK_tbl_EmployeeCourse_XREF_tbl_Employee] 85 86

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
```

USE [master]

1

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

2 GO 3 /***** Object: Database [EmployeeTraining] Script Date: 09/23/2012 10:34:43 *****/ 4 5 IF EXISTS (SELECT name FROM sys.databases WHERE name = N'EmployeeTraining') DROP DATABASE [EmployeeTraining] 6 7 GO 8.7 CreateDatabase.sql 1 USE [master] 2 GO 3 Script Date: 09/23/2012 10:34:43 *****/ 4 /***** Object: Database [EmployeeTraining] CREATE DATABASE [EmployeeTraining] ON PRIMARY 5 (NAME = N'EmployeeTraining', FILENAME = N'C:\database\mssql\EmployeeTraining\EmployeeTraining.mdf', SIZE = 2048KB , MAXSIZE = UNLIMITED, FILEGROWTH = 1024KB) LOG ON 7 (NAME = N'EmployeeTraining log', FILENAME = 8 N'C:\database\mssql\EmployeeTraining\EmployeeTraining_log.ldf', SIZE = 1024KB, MAXSIZE = 2048GB, FILEGROWTH = 10응) GO 9 10 ALTER DATABASE [EmployeeTraining] SET COMPATIBILITY LEVEL = 100 11 12 GO 13 IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled')) 14 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 ALTER DATABASE [EmployeeTraining] SET ANSI_PADDING OFF 26 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 ALTER DATABASE [EmployeeTraining] SET AUTO CREATE STATISTICS ON 38 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 ALTER DATABASE [EmployeeTraining] SET CURSOR DEFAULT GLOBAL 50 51 GO 52 ALTER DATABASE [EmployeeTraining] SET CONCAT_NULL_YIELDS_NULL OFF 53 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 ALTER DATABASE [EmployeeTraining] SET AUTO_UPDATE_STATISTICS_ASYNC OFF 68 69 GO 70 71 ALTER DATABASE [EmployeeTraining] SET DATE_CORRELATION_OPTIMIZATION OFF 72 GO 73 ALTER DATABASE [EmployeeTraining] SET TRUSTWORTHY OFF 74 75 GO 76 77 ALTER DATABASE [EmployeeTraining] SET ALLOW_SNAPSHOT_ISOLATION OFF 78 GO 79 ALTER DATABASE [EmployeeTraining] SET PARAMETERIZATION SIMPLE 80 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 ALTER DATABASE [EmployeeTraining] SET READ_WRITE 89 90 GO 91 ALTER DATABASE [EmployeeTraining] SET RECOVERY FULL 92 93 GO 94 ALTER DATABASE [EmployeeTraining] SET MULTI USER 95 96 GO 97 98 ALTER DATABASE [EmployeeTraining] SET PAGE VERIFY CHECKSUM 99 GO 100 ALTER DATABASE [EmployeeTraining] SET DB CHAINING OFF 101 102 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 Employ-eeTraining.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.

Script Date: 10/13/2012 08:52:28 *****/ /***** Object: Login [EmployeeTraining] 1 2 IF EXISTS (SELECT * FROM sys.server principals WHERE name = N'EmployeeTraining') 3 DROP LOGIN [EmployeeTraining] 4 GO 8.9 CreateLogin.sql /***** Object: Login [EmployeeTraining] Script Date: 10/13/2012 08:52:28 *****/ 1 CREATE LOGIN [EmployeeTraining] WITH PASSWORD=N'password', DEFAULT_DATABASE=[EmployeeTraining], 2 DEFAULT_LANGUAGE=[us_english], CHECK_EXPIRATION=OFF, CHECK_POLICY=OFF 3 GO 4 EXEC sys.sp_addsrvrolemember @loginame = N'EmployeeTraining', @rolename = N'dbcreator' 5 6 The next script will be the CreateUser.sql script, which is given in example 8.10. 8.10 CreateUser.sql USE [EmployeeTraining] 1 2 GO 3 /***** Object: User [EmployeeTrainingUser] Script Date: 10/13/2012 08:30:07 *****/ 4 5 GO 6 CREATE USER [EmployeeTrainingUser] FOR LOGIN [EmployeeTraining] WITH DEFAULT SCHEMA=[dbo] 7 8

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

8.8 DropLogin.sql

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

sary.

```
71
        ON UPDATE CASCADE
        ON DELETE CASCADE
72
73
        GO
        IF EXISTS (SELECT * FROM sys.foreign keys WHERE object id =
74
OBJECT_ID(N'[dbo].[FK_tbl_EmployeeCourse_XREF_tbl_Course_LU]') AND parent_object_id =
OBJECT_ID(N'[dbo].[tbl_EmployeeCourse_XREF]'))
        ALTER TABLE [dbo].[tbl_EmployeeCourse_XREF] CHECK CONSTRAINT [FK_tbl_EmployeeCourse_XREF_tbl_Course_LU]
75
76
        GO
        /***** Object: ForeignKey [FK_tbl_EmployeeCourse_XREF_tbl_Employee]
                                                                                  Script Date: 10/13/2012
77
09:55:02 *****/
        IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
78
OBJECT_ID(N'[dbo].[FK_tbl_EmployeeCourse_XREF_tbl_Employee]') AND parent_object_id =
OBJECT_ID(N'[dbo].[tbl_EmployeeCourse_XREF]'))
        ALTER TABLE [dbo].[tbl_EmployeeCourse_XREF] WITH CHECK ADD CONSTRAINT
79
[FK_tbl_EmployeeCourse_XREF_tbl_Employee] FOREIGN KEY([FK_EmployeeID])
        REFERENCES [dbo]. [tbl_Employee] ([EmployeeID])
80
        ON UPDATE CASCADE
81
82
        ON DELETE CASCADE
83
        GO
        IF EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
84
OBJECT_ID(N'[dbo].[FK_tbl_EmployeeCourse_XREF_tbl_Employee]') AND parent_object_id =
OBJECT_ID(N'[dbo].[tbl_EmployeeCourse_XREF]'))
        ALTER TABLE [dbo].[tbl_EmployeeCourse_XREF] CHECK CONSTRAINT [FK_tbl_EmployeeCourse_XREF_tbl_Employee]
85
86
        GO
    Referring to example 8.11 — This script is the same as the raw script since no editing or segregation was neces-
```

```
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.

Organize 🔻 Include in library 👻 S	Share w	ith 🔻 Burn New folder			• ==	
Projects	^	Name	Date modified	Туре	Size	
EmployeeTrainingProject		📄 CreateDatabase.sql	10/13/2012 10:25	Microsoft SQL Server Query File	3	l.
J. svn		📄 CreateLogin.sql	10/13/2012 10:30	Microsoft SQL Server Query File	1	I
S branches	_	📄 CreateTables.sql	10/13/2012 10:00	Microsoft SQL Server Query File	4	
💋 tags	Ξ	📄 CreateUser.sql	10/13/2012 8:31 AM	Microsoft SQL Server Query File	1	Select a file
S trunk		📄 DropDatabase.sql	10/13/2012 10:17	Microsoft SQL Server Query File	1	I
🔗 database		📄 DropLogin.sql	10/13/2012 10:28	Microsoft SQL Server Query File	1	I
Release_1.0_DB_Scripts						
i documentation	-	•			•	
6 items						

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

```
USE [EmployeeTraining]
1
2
        GO
         /* Insert test data into tbl Employee */
3
        INSERT INTO dbo.tbl Employee (FirstName, MiddleName, LastName, Birthday, HireDate)
4
               VALUES ('Rick', 'Warren', 'Miller', '1/1/1970 12:00:00 AM', '12/1/1998 12:00:00 AM');
5
        INSERT INTO dbo.tbl_Employee (FirstName, MiddleName, LastName, Birthday, HireDate)
6
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)
                VALUES ('Nancy', 'Jo', 'Coats', '1/1/1961 12:00:00 AM', '12/1/1998 12:00:00 AM');
11
        INSERT INTO dbo.tbl_Employee (FirstName, MiddleName, LastName, Birthday, HireDate)
12
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 */
        INSERT INTO dbo.tbl Course LU (Code, Title, Description)
16
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...');
        INSERT INTO dbo.tbl_Course_LU (Code, Title, Description)
20
               VALUES ('PMP101', 'Introduction to Program Management', 'Description test here...');
21
22
        INSERT INTO dbo.tbl_Course_LU (Code, Title, Description)
               VALUES ('PMP102', 'Intermediate Program Management', 'Description test here...');
23
        INSERT INTO dbo.tbl Course LU (Code, Title, Description)
24
25
               VALUES ('PMP103', 'Earned Value Management', 'Description test here...');
26
        /* Insert test data into tbl_EmployeeCourse_XREF */
27
        INSERT INTO dbo.tbl_EmployeeCourse_XREF (FK_EmployeeID, FK_CourseID, DateCompleted, Grade)
28
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)
               VALUES (1, 2, '1/1/2012 12:00:00 AM', '4.0');
31
        INSERT INTO dbo.tbl_EmployeeCourse_XREF (FK_EmployeeID, FK_CourseID, DateCompleted, Grade)
32
                                                      '4.0');
33
                VALUES (2, 3, '1/1/2012 12:00:00 AM',
        INSERT INTO dbo.tbl_EmployeeCourse_XREF (FK_EmployeeID, FK_CourseID, DateCompleted, Grade)
34
                VALUES (2, 4, '1/1/2012 12:00:00 AM', '4.0');
35
        INSERT INTO dbo.tbl_EmployeeCourse_XREF (FK_EmployeeID, FK_CourseID, DateCompleted, Grade)
36
                VALUES (3, 1, '1/1/2012 12:00:00 AM', '4.0');
37
```

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

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

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

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

21 sqlcmd -i "Release_1.0_DB_Scripts\InsertTestData.sql" -b -0 %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.

Computer > Local Disk (C:) 🕨	Projects EmployeeTrainingProject trunk	▶ database ▶		•	Search databa 🔎
Organize	furn	New folder Name logs Release 1.0_DB_Scripts	Date modified 10/13/2012 11:42 10/13/2012 10:47	Type File folder File folder	Size	echo setting up
⊘ banches ⊘ tags ⊘ trunk ⊘ database	Ш	(%) CreateRelease_1.0_Database.bat (%) Set_Environment.bat	10/13/2012 12:00 10/13/2012 12:06	Windows Batch File Windows Batch File	1 KB	environment parameters. call Set_Environ ment.bat
 Release_1.0_DB_Scripts documentation 						echo Dropping EmployeeTra ining -
CreateRelease_1.0_Databa Windows Batch File	se.bat	: Date modified: 10/13/2012 12:00 PM Size: 1020 bytes Date created: 10/13/2012 11:36 AM				

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.

🔤 Administrator: Command Prompt	×
C:\Projects\EmployeeTrainingProject\trunk\database>CreateRelease_1.0_Database.ba	t 着
Setting up environment parameters	
Dropping EmployeeTraining Database Dropping EmployeeTraining Login	
Creating EmployeeTraining Database Creating EmployeeTraining Login	
Creating EmployeeTraining User	
Inserting test data	
C:\Projects\EmployeeTrainingProject\trunk\database>	
	-
<	►

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.

Add	Table				? X
T	ables				
t	bl_Course_LU				
t	bl_Employee				
t	bl_EmployeeCourse_Xf	REF			
			Refresh	Add	Close

Figure 8-15: Adding Tables to a Database Diagram

Release_1.0_DB_Script Open	
Open of the second seco	
CreateRelease_1.0_Da 07 00 Windows Batch File	1 KB
Set_Environment.bat Open in new window 06 Windows Batch File	1 KB
🕤 TortoiseSVN 🔹 Add	
Open on Mac 📑 Add to ignore list 🕨	
Show in Finder 🍸 Settings	
Send to	
Cut	
Сору	
Create shortcut	
Delete	
Rename	
Properties	

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.

양 C\Projects\EmployeeTrainingProject\trunk\database - Add - TortoiseSVN	
Path	Extension
CreateRelease_1.0_Database.bat	.bat
Release_1.0_DB_Scripts	
🗹 📄 Release_1.0_DB_Scripts/CreateDatabase.sql	.sql
Release_1.0_DB_Scripts/CreateLogin.sql	.sql
Release_1.0_DB_Scripts/CreateTables.sql	.sql
Release_1.0_DB_Scripts/CreateUser.sql	.sql
Release_1.0_DB_Scripts/DropDatabase.sql	.sql
🛛 🛐 Release_1.0_DB_Scripts/DropLogin.sql	.sql
Release_1.0_DB_Scripts/InsertTestData.sql	.sql
Set_Environment.bat	.bat
۲ III III III III III III III III III I	۲. F
Enable Auto-Properties	
	K <u>C</u> ancel <u>H</u> elp

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.

🦨 Add Finish	ed!	
Action	Path	Mime type
Command	Add	
Added	C: \Projects \EmployeeTrainingProject \trunk \database \CreateRelease_1.0_Database.bat	
Added	C: \Projects \EmployeeTrainingProject \trunk \database \Release_1.0_DB_Scripts	
Added	C:\Projects\EmployeeTrainingProject\trunk\database\Release_1.0_DB_Scripts\CreateDatabase.sql	
Added	C: \Projects \EmployeeTrainingProject \trunk \database \Release_1.0_DB_Scripts \CreateLogin.sql	
Added	C: \Projects \EmployeeTrainingProject \trunk \database \Release _1.0_DB_Scripts \CreateTables.sql	
Added	C: \Projects \EmployeeTrainingProject \trunk \database \Release _1.0_DB_Scripts \CreateUser.sql	
Added	C: \Projects\EmployeeTrainingProject\trunk\database\Release_1.0_DB_Scripts\DropDatabase.sql	the second se
Added	C: \Projects \EmployeeTrainingProject \trunk \database \Release_1.0_DB_Scripts \DropLogin.sql	
Added	C: \Projects \EmployeeTrainingProject \trunk \database \Release _1.0_DB_Scripts \InsertTestData.sql	The second se
Added	C: \Projects\EmployeeTrainingProject\trunk\database\Set_Environment.bat	
Completed!		
Added: 10		OK Cancel

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.

👌 logs			10/13/20)12 11:42	File folder
👪 Release_1.0_DB	_Scrip	ots	10/13/20)12 10:47	File folder
🚯 CreateRelease_	🖏 CreateRelease_1.0_Database.bat)12 12:00	Windows Batch File 1 KB
Set_Environme	1.0_D	atabase.bat Edit with Notepad++ Open on Mac Show in Finder SVN Commit TortoiseSVN Send to Cut Copy	10/13/20	112 12:00 112 12:06	Windows Batch File 1 KB Windows Batch File 1 KB
		Create shortcut Delete Rename Properties			

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.

C:\Projects\EmployeeTrainingProject\trunk\databa:	se - Commit	- Tortois	eSVN	
nmit to:				
ultiple targets selected)				
lessage:				
Recent messages	ersion of Torto	iseSVN is	available. Please ç	go to http://tortoisesvn.net to get it.
Initial commit of release 1.0 database	scripts.			
hanges made (double-click on file for diff):				
check: All None Non-versioned Versioned Ad	ided Delete	ed Mod	ified Files Dire	ectories
Path	Extension	Status	Property status	Lock
CreateRelease_1.0_Database.bat	.bat	added		
Release_1.0_DB_Scripts		added		
Release_1.0_DB_Scripts/CreateDatabase.sql	.sql	added		
Release_1.0_DB_Scripts/CreateLogin.sql	.sql	added		
Release_1.0_DB_Scripts/CreateTables.sql	.sql	added		
Release 1.0 DB Scripts/CreateUser.sql	.sql	added		
Release 1.0 DB Scripts/DropDatabase.sql	.sql	added		
Release 1.0 DB Scripts/DropLogin.sgl	.sql	added		
Release 1.0 DB Scripts/InsertTestData.sol	.sal	added		
Set Environment.bat	.bat	added		
<	III	_		Þ
Show unversioned files				10 files selected, 10 files total
Show unversioned files				10 files selected, 10 files total
Show unversioned files Show externals from different repositories				10 files selected, 10 files total
Show unversioned files Show externals from different repositories Keep locks				10 files selected, 10 files total

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

🖗 Commit Finish	ed!		
Action	Path	Mime type	
Command	Commit		
Adding	C:\Projects\EmployeeTrainingProject\trunk\database\CreateRelease_1.0_Database.bat		
Adding	C:\Projects\EmployeeTrainingProject\trunk\database\Release_1.0_DB_Scripts		
Adding	$C: \label{eq:c:projects} Employee Training Project \trunk \database \Release _ 1.0 _ DB _ Scripts \Create Database .sql \label{eq:c:projects} Create \Release _ 1.0 _ DB _ Scripts \Create \Release \Release _ 1.0 _ DB _ Scripts \Create \Release \Release _ 1.0 _ DB _ Scripts \Release \Rele$		
Adding	C:\Projects\EmployeeTrainingProject\trunk\database\Release_1.0_DB_Scripts\CreateLogin.sql		
Adding	C:\Projects\EmployeeTrainingProject\trunk\database\Release_1.0_DB_Scripts\CreateTables.sql		
Adding	C:\Projects\EmployeeTrainingProject\trunk\database\Release_1.0_DB_Scripts\CreateUser.sql		
Adding	C:\Projects\EmployeeTrainingProject\trunk\database\Release_1.0_DB_Scripts\DropDatabase.sql		
Adding	C:\Projects\EmployeeTrainingProject\trunk\database\Release_1.0_DB_Scripts\DropLogin.sql		
Adding	C:\Projects\EmployeeTrainingProject\trunk\database\Release_1.0_DB_Scripts\InsertTestData.sql		
Adding	C:\Projects\EmployeeTrainingProject\trunk\database\Set_Environment.bat		
Sending content	C:\Projects\EmployeeTrainingProject\trunk\database\Release_1.0_DB_Scripts\CreateDatabase.sql		
Sending content	C:\Projects\EmployeeTrainingProject\trunk\database\Release_1.0_DB_Scripts\DropDatabase.sql		
Sending content	C:\Projects\EmployeeTrainingProject\trunk\database\Release_1.0_DB_Scripts\CreateLogin.sql		
Sending content	C:\Projects\EmployeeTrainingProject\trunk\database\Set_Environment.bat		
Sending content	C:\Projects\EmployeeTrainingProject\trunk\database\Release_1.0_DB_Scripts\DropLogin.sql	<pre></pre>	
Sending content	C:\Projects\EmployeeTrainingProject\trunk\database\Release_1.0_DB_Scripts\CreateTables.sql	/	
Sending content	C:\Projects\EmployeeTrainingProject\trunk\database\Release_1.0_DB_Scripts\InsertTestData.sql	/	
Sending content	C:\Projects\EmployeeTrainingProject\trunk\database\Release_1.0_DB_Scripts\CreateUser.sql		
Sending content	C:\Projects\EmployeeTrainingProject\trunk\database\CreateRelease_1.0_Database.bat		En la
Completed	At revision: 3		9
913 Bytes transfer	red in 0 minute(s) and 6 second(s)		
Added: 10		ОК	Cancel

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