SQL Source Control 1.0

June 16th, 2010

Note: this documentation applies to an old version of this product.

For the latest documentation, see documentation.red-gate.com

Contents

Worked example: setting up SQL Source Control with SVN for development teams 4
Worked example: setting up SQL Source Control with TFS for development teams.....16
Linking a database to source control 26
Committing changes 28
Getting the latest version.....31
Viewing source control logs / history 33
Getting a specific version.....34
Undoing changes 35
Conflicts 37
Deploying a database from source control.....39
Setting up a local Subversion repository 41
Continuous integration 43
Source controlling data.....44
Evaluation center.....46

Worked example: setting up SQL Source Control with SVN for development teams

This example demonstrates how to set up database source control so that teams of developers can work on a database update.

This example uses:

- The Subversion (http://subversion.apache.org/) source control system
- The Tortoise SVN (http://tortoisesvn.tigris.org/) client for Subversion

To follow this example, you should download and install the latest version of TortoiseSVN (http://tortoisesvn.net/downloads)

In the example, the Magic Widget Company has a SQL Server database running on a live web server. This database contains a number of tables, views, stored procedures, and other database objects. The Magic Widget Company's development team wants to begin working on an update to this database. They have already created a copy of the production database, as a baseline to develop against.

They now need to source control the development database, so that each developer can get their own dedicated copy to work on.

This example has 6 steps:

1. Set up the database
2. Link the database to source control
3. Commit the database objects to source control
4. Get the latest version
5. Make development changes
6. Commit the development changes to source control

1. Set up the database

This worked example uses the *WidgetDevelopment* database.

To create the database on your SQL Server:

1. If it already exists, delete *WidgetDevelopment*
2. Click here to view the SQL creation script (/support/SQL_Source_Control/help/2.0/WidgetDevScript.sql) for the database.
3. Copy the script, paste it into a query window in SQL Server Management Studio, and run it.

The database and its schema are created.

2. Link the database to source control

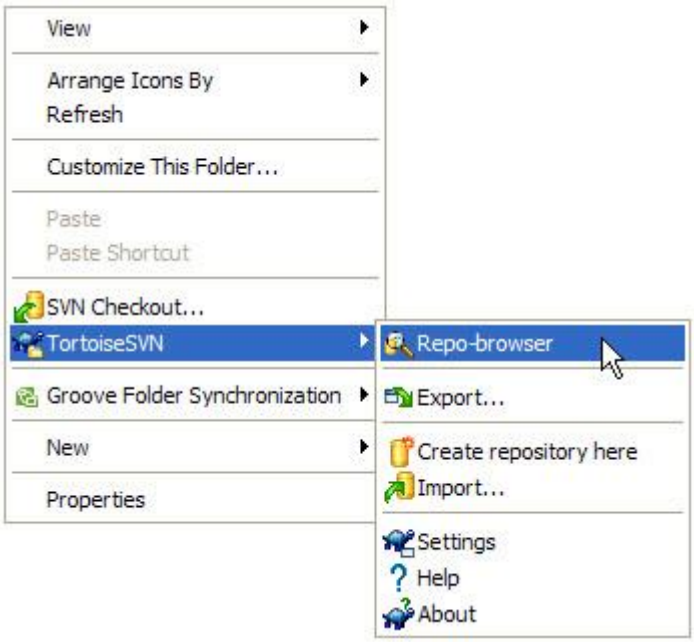
Linking associates the database with a location in source control.

That location must be an existing, empty folder.

We will create a folder and link the database to that location.

Create a directory in the repository

1. In a Windows Explorer window, right-click, and from TortoiseSVN select **Repo-browser**:

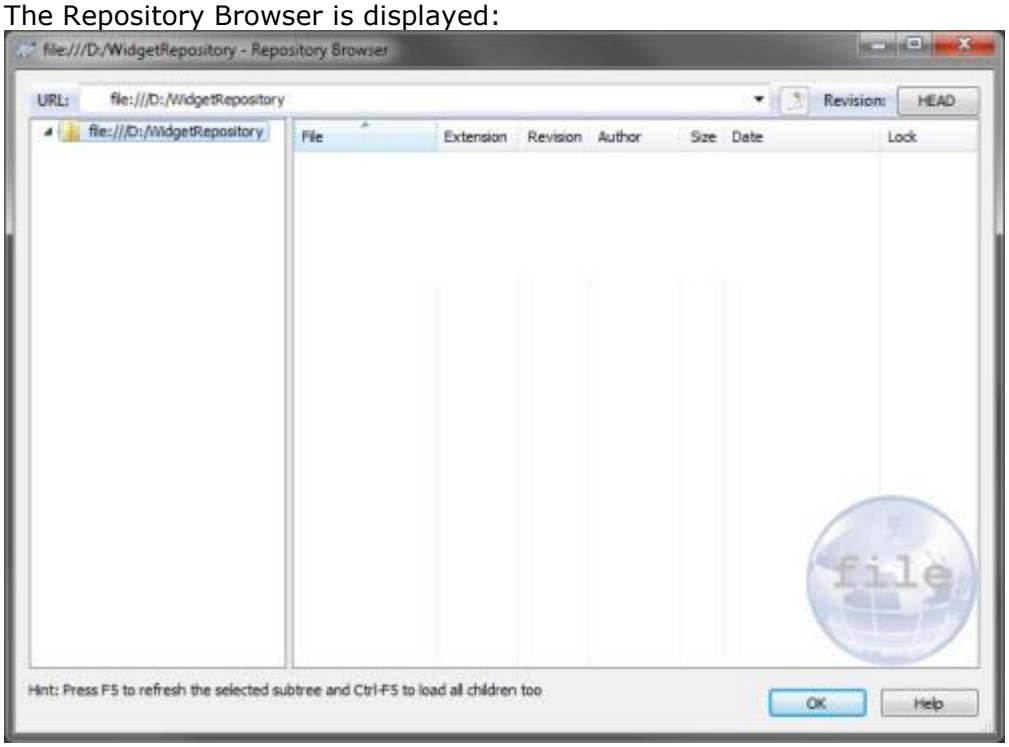


Note that the TortoiseSVN menu does not appear if the folder you right-click is a Windows 7 library.

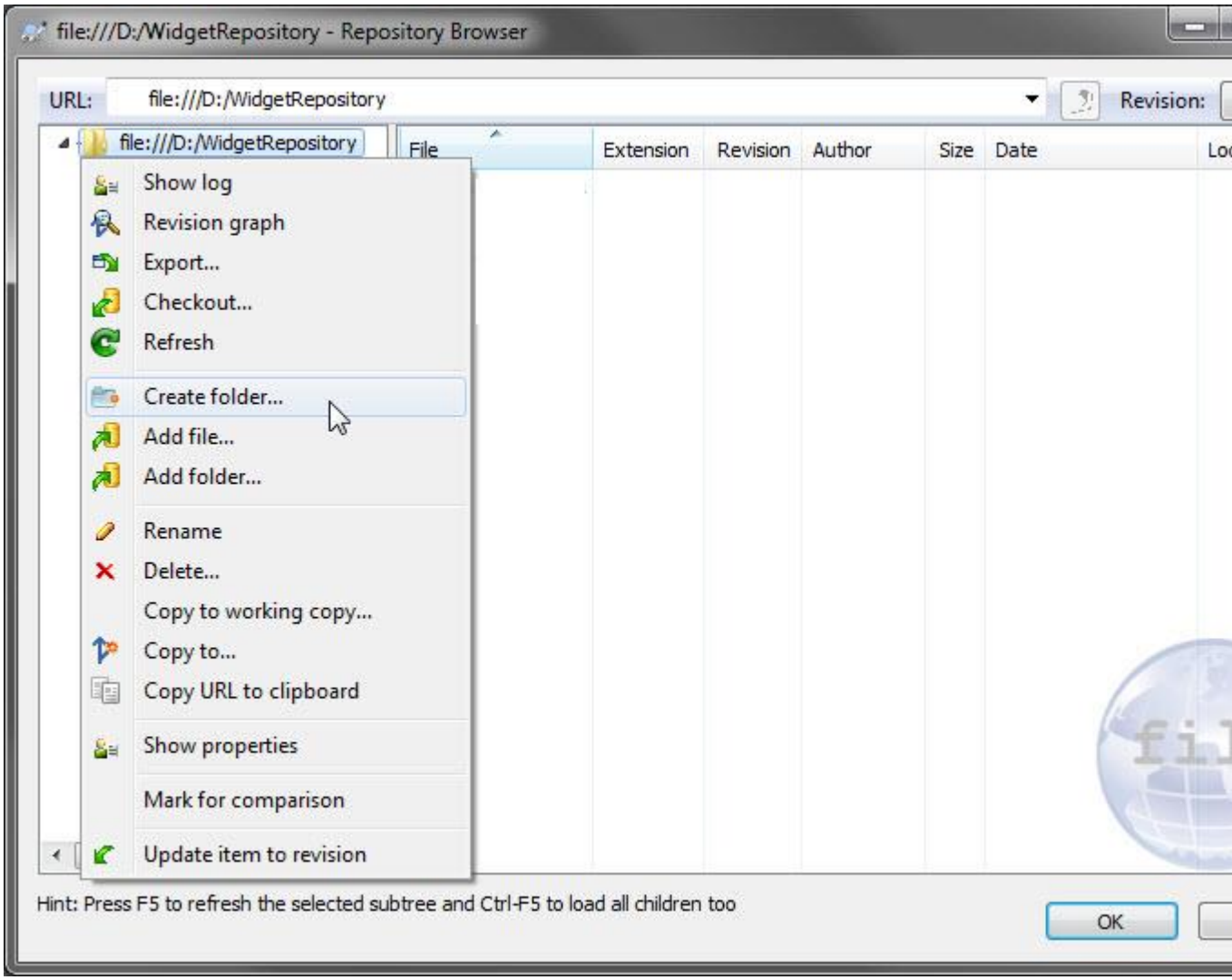
The URL dialog box is displayed.

2. In **URL**, type or paste the URL for your repository, and click **OK**

   The Repository Browser is displayed:



3. In the Repository Browser, in the left pane, select the folder in the repository where you want to source control the database. Then right-click the folder, and select **Create folder**:



On the Create Folder dialog box, name the folder *WidgetDev* and click **OK**

You may be prompted to supply a comment.

The folder is created.

4. Right-click the new folder, and select **Copy URL to clipboard**

   You can now use the URL of this folder on the Setup tab in SQL Source Control to create a link between your database and the repository.

Create a link to source control

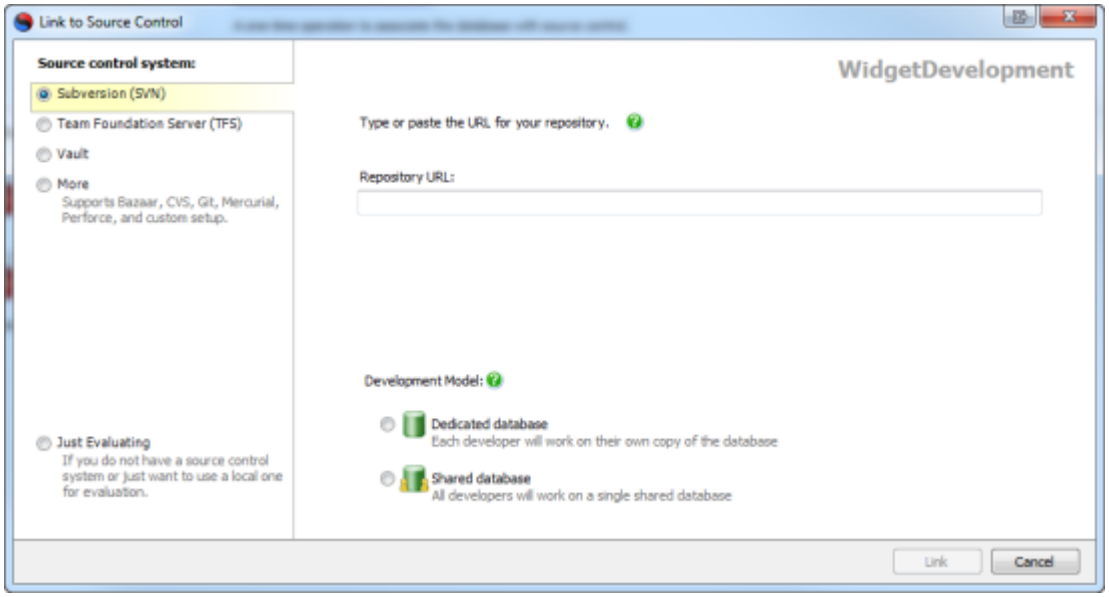
1. Open SQL Server Management Studio if it is not already open.

2. In the **Object Explorer**, select the *WidgetDevelopment* database, right-click, and click **Link database to source control**

   The SQL Source Control Setup tab is displayed.

3. Click **Create new link to source control**

   The Create Link to Source Control dialog box is displayed:



4. Under **Source control system**, on the left hand side, ensure *Subversion* is selected.
5. In **Repository URL,** type or paste the URL for the folder you created.

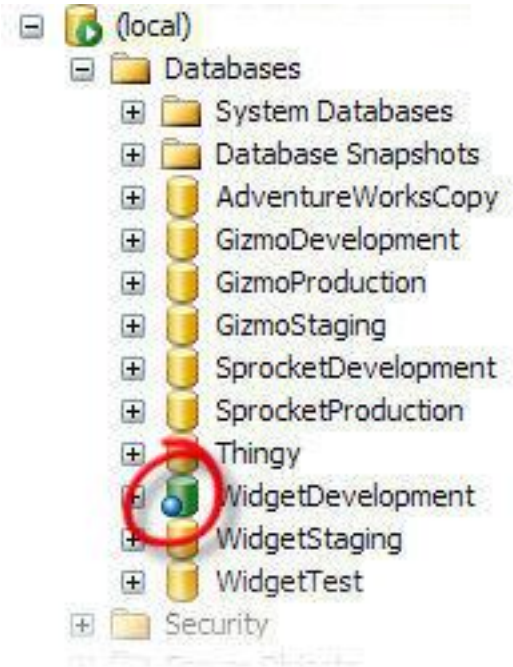
   Note that the repository URL is case sensitive.

6. Click **Create Link**

   You may be prompted for login credentials for your source control repository.

   A link to the repository is created, and SQL Source Control is now able to determine differences between the database and the repository.

The database icon in the Object Explorer changes, indicating that the database is linked to source control, and that there are changes to commit:



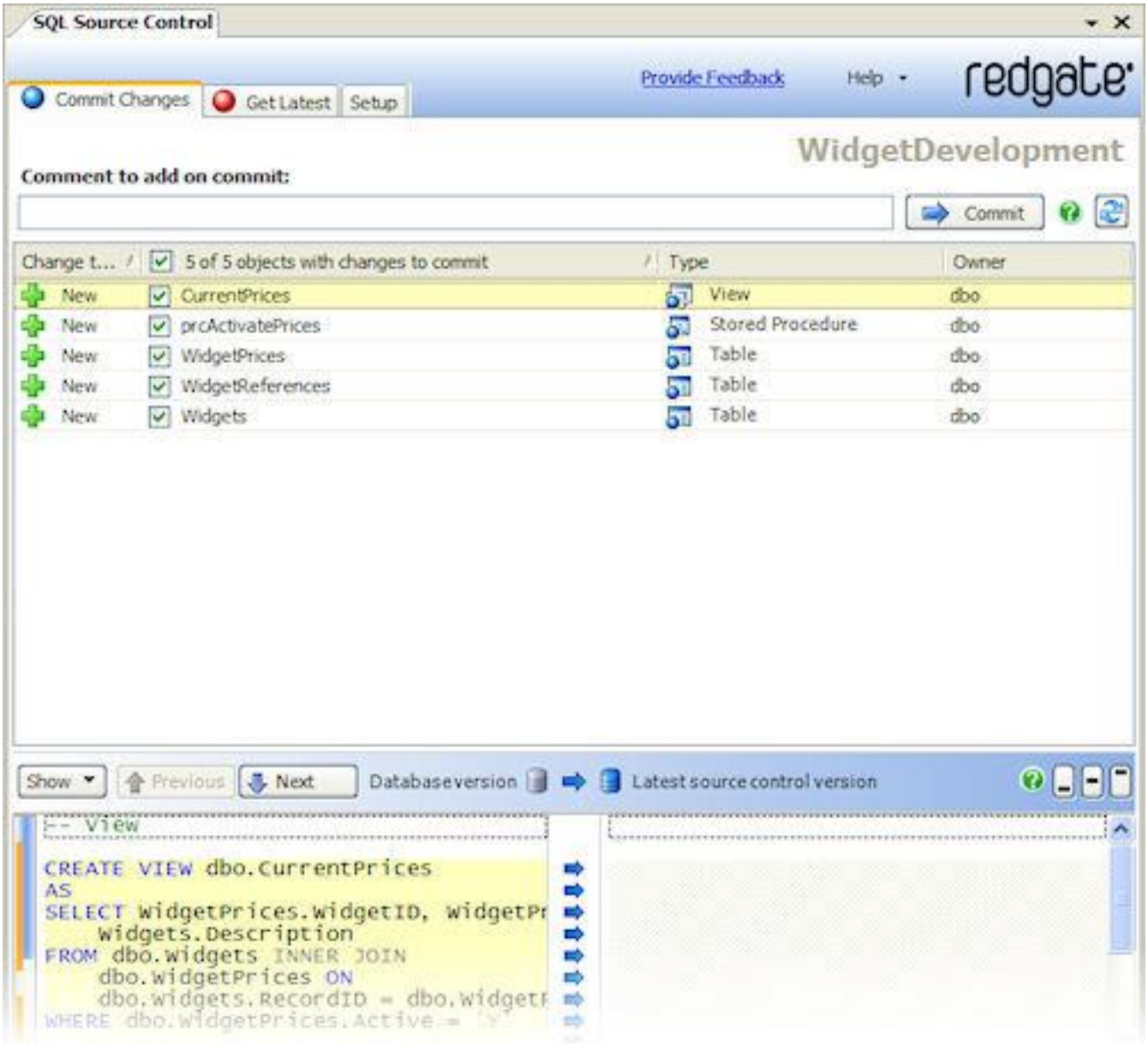
Note that the database objects have not yet been committed to source control.

3. Commit the database objects to source control

To complete the process of source controlling the database, commit the objects:

1. In SQL Source Control, click the **Commit Changes** tab.

The Commit Changes tab displays a list of all the objects with database versions that do not match the latest source control version:



Because none of the objects yet have versions in source control, they are all listed.

You can view the creation script for an object by clicking it. The script is displayed in the Object Differences pane, below the list of objects to commit.

2. In **Comment to add on commit**, type or paste a comment.

   Comments are recommended as they help provide a detailed change history.

   In this example, type *Initial commit of all objects*.

3. Click **Commit**

   SQL Source Control displays a message dialog box showing the progress of the commit.

When the commit is complete, click **OK** to close the message box.

The objects are committed to source control and other users can now get the latest version of the database.

The Object Explorer is updated to show that there are now no outstanding changes to commit.

4. Get the latest version

Now the database is in source control, another user can get the latest version, and make development changes.

They create a new database, link it to source control, and update it with the latest version of all the objects.

Linking the database

1. In SQL Server Management Studio, create a new database, with the name *WidgetDevelopment*
2. In the **Object Explorer**, ensure *WidgetDevelopment* is selected. On the **Setup** tab, click **Link to a database already in source control**.

   The Link Database to Source Control dialog box is displayed.
3. Under **Source control system**, on the left hand side, ensure *Subversion* is selected.
4. In **Repository URL**, type or paste the URL of the database in source control.

   Note that the repository URL is case sensitive.
5. Click **Link**

   You may be prompted for login credentials for your source control repository.

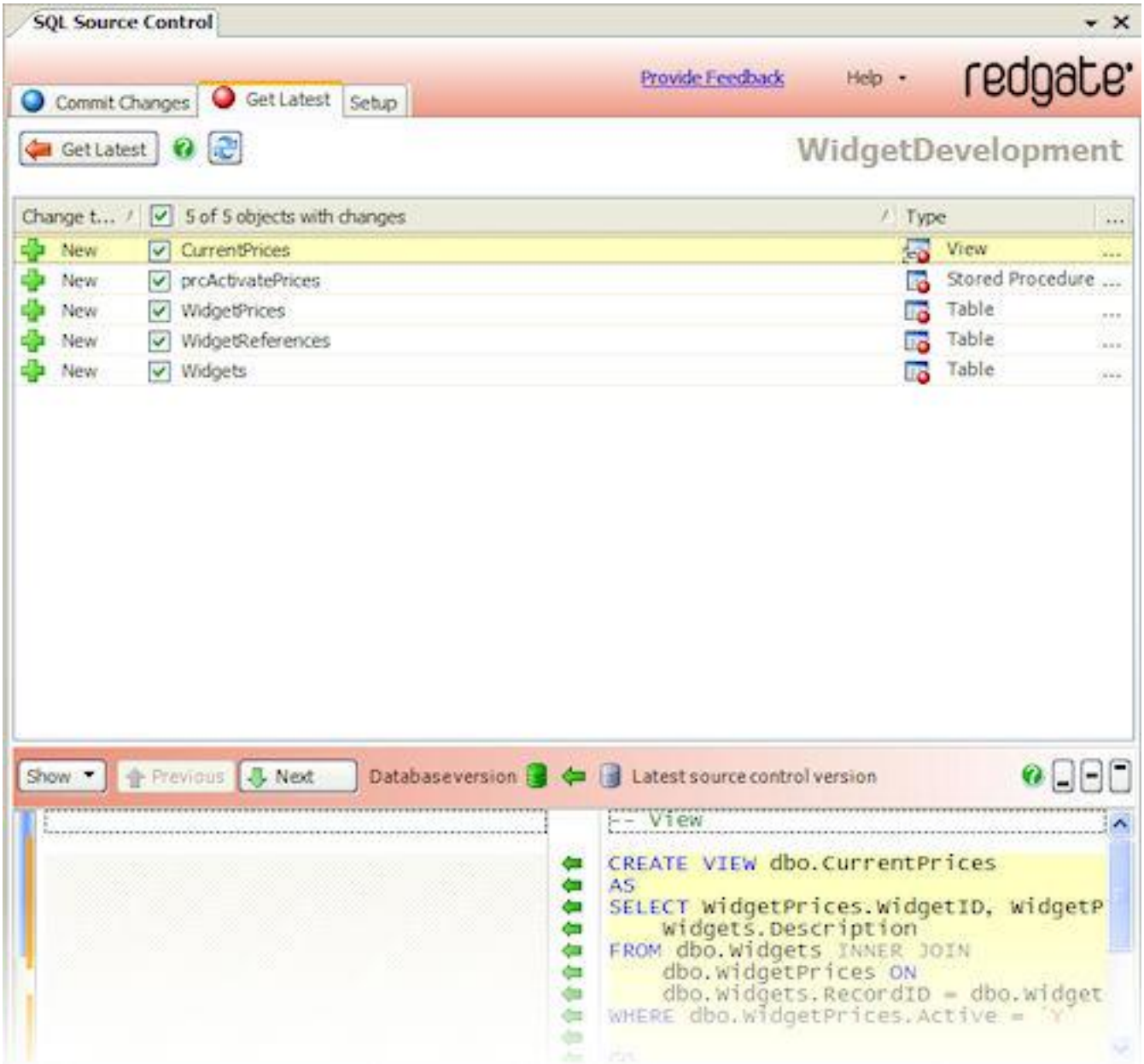
   A link to the repository is created, and SQL Source Control is able to determine differences between the database and the repository.

   However, the database has not yet been updated with the objects from source control.

Getting the latest version

1. Click the **Get Latest** tab.

Because you do not yet have any of the objects in your database, all the objects in *WidgetDevelopment* are listed here.



2. Ensure all of the objects are selected.
3. Click **Get Latest**

   A progress dialog box is displayed while SQL Source Control updates the database.

   The database is updated to the latest version.

5. Make development changes

Development proceeds normally, and the database is modified.

A column is changed in the table *Widgets*, to allow longer widget descriptions. This change is committed to source control.

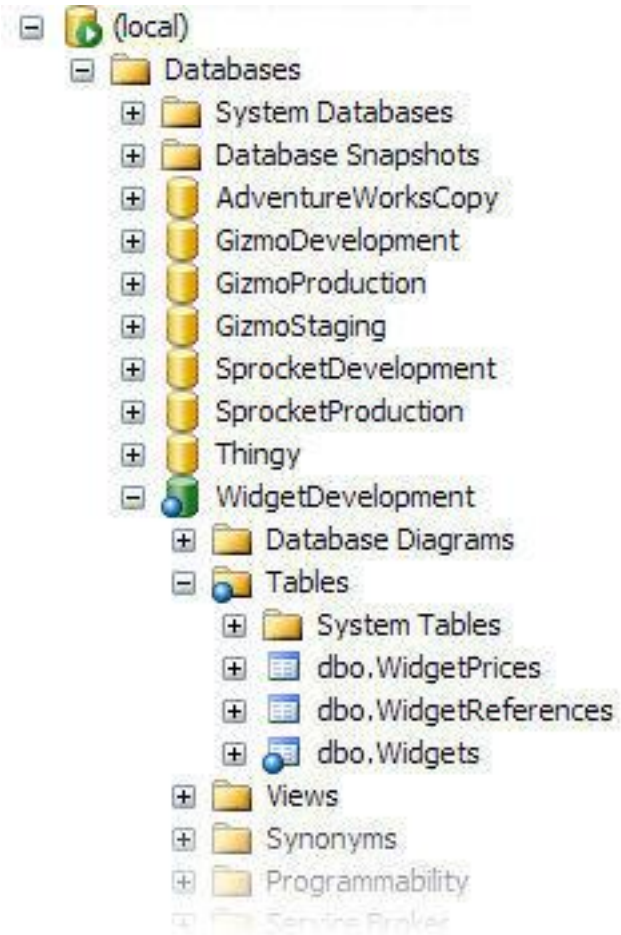
1. In SQL Server Management Studio, open a new query window, and type or paste the following SQL statement:

```
USE WidgetDevelopment
GO
ALTER TABLE [dbo].[Widgets] ALTER COLUMN [Description] [nvarchar] (100)
```

2. Click **Execute** or press **F5**

   The script runs; the database is updated.

3. SQL Source Control detects the change to the database, and highlights the affected object in the Object Explorer:

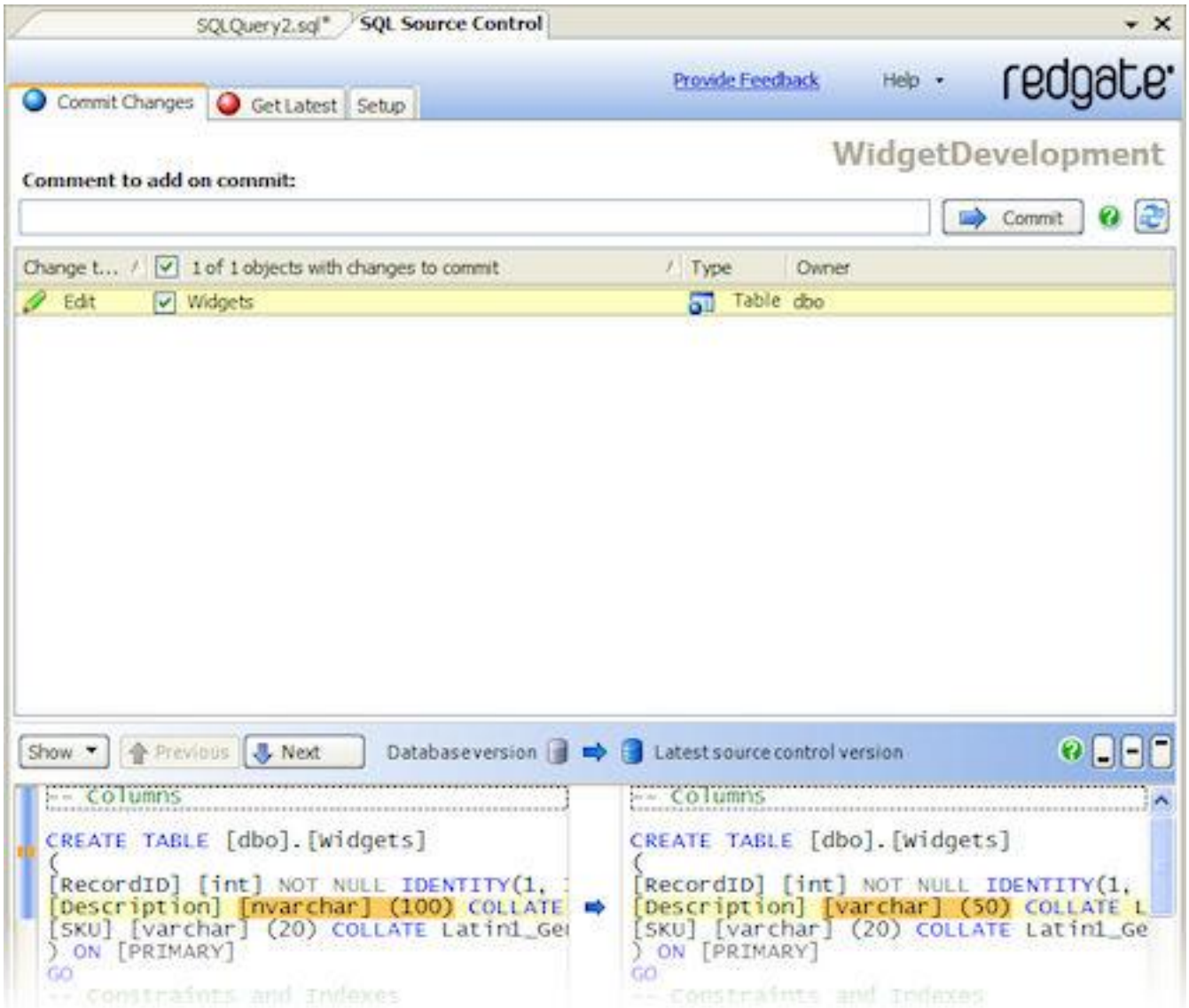


You can now commit the change.

6. Commit the development changes to source control

1. In the Object Explorer, right click the table *Widgets,* and click **Commit changes to source control**

The Commit changes tab is displayed:



The change to the table *Widgets* is listed as an edit you can commit.

The Object Differences pane shows the difference between your database version, and the latest version in source control.

2. In **Comment to add on commit**, type or paste a comment.

   In this example, type *Modified Description column*

3. Click **Commit**

   SQL Source Control displays a message dialog box that shows the progress of the commit.

   When the commit is complete, click **OK** to close the message box.

Your change is committed to source control. The Commit Changes tab lists no objects with changes to commit, and no objects are highlighted in the Object Explorer.

Other users can now get the latest version of the table.

Worked example: setting up SQL Source Control with TFS for development teams

This example demonstrates how to set up database source control so that teams of developers can work on a database update.

This example requires:

- The Team Foundation Server (TFS) source control system
- Microsoft Team Explorer client for Visual Studio

In the example, the Magic Widget Company has a SQL Server database. This database contains a number of tables, views, stored procedures, and other database objects. The Magic Widget Company's development team wants to begin working on an update to this database. They have already created a copy of the production database, as a baseline to develop against.

They now need to source control the development database, so that each developer can work on their own dedicated copy.

This example has 6 steps:

1. Set up the database
2. Link the database to source control
3. Commit the database objects to source control
4. Get the latest version
5. Make development changes
6. Commit the development changes to source control

1. Set up the database

This worked example uses the *WidgetDevelopment* database.

To create the database on your SQL Server:

1. If it already exists, delete *WidgetDevelopment*
2. Click here to view the SQL creation script (/support/SQL_Source_Control/help/2.0/WidgetDevScript.sql) for the database.
3. Copy the script, paste it into a query window in SQL Server Management Studio, and run it.

The database and its schema are created.

2. Link the database to source control

Linking associates the database with a location in source control.

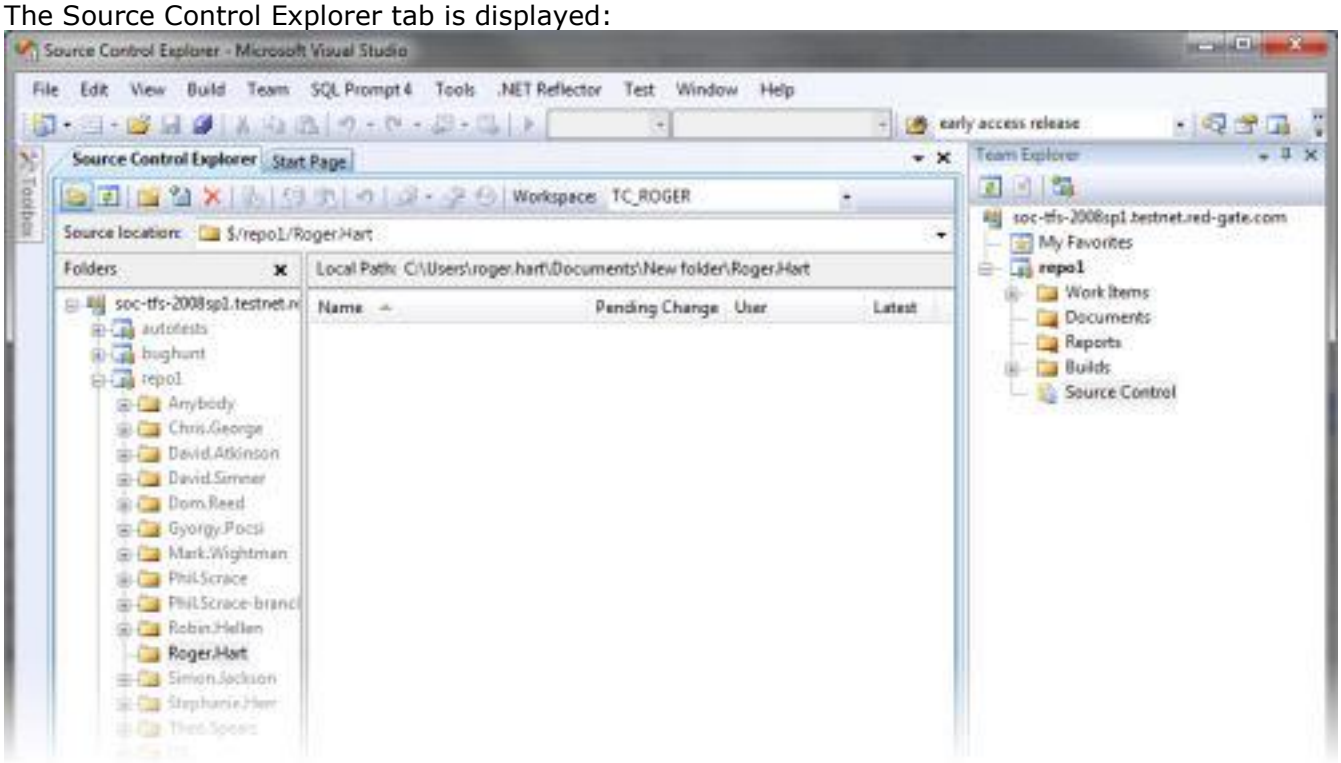
That location must be an existing, empty folder.

We will create a folder and link the database to that location.

Create a folder in source control

1. Open Visual Studio if it is not already running, and in the **Team Explorer** pane, under the server you are using, double-click **Source Control**

   The Source Control Explorer tab is displayed:



2. In the Folders pane, browse to your source control location.
3. In the file list, right-click, and click **New Folder**

   A folder is created. Name it *WidgetDev*

   Note that in some circumstances, the option to create a new folder is not available.

   If this occurs, in the Folders pane, right click your source control location, and click **Get Latest**.

   Once the local copy has updated, the option to create a folder becomes available.
4. Right-click *WidgetDev*, and click **Check In Pending Changes**

   The Check In dialog box is displayed.
5. Type a comment, and click **Check in**

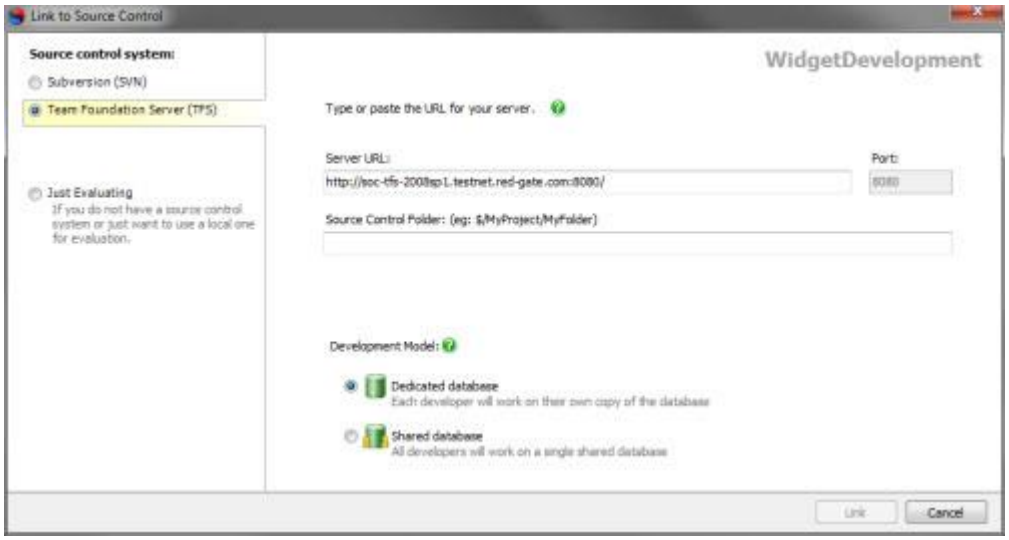
   The folder is committed to source control.

Create a link to source control

1. Open SQL Server Management Studio if it is not already open.
2. In the **Object Explorer**, select the *WidgetDevelopmen*t database, right-click, and click **Link database to source control**

   The SQL Source Control Setup tab is displayed.
3. Click **Create new link to source control**

   The Create Link to Source Control dialog box is displayed:



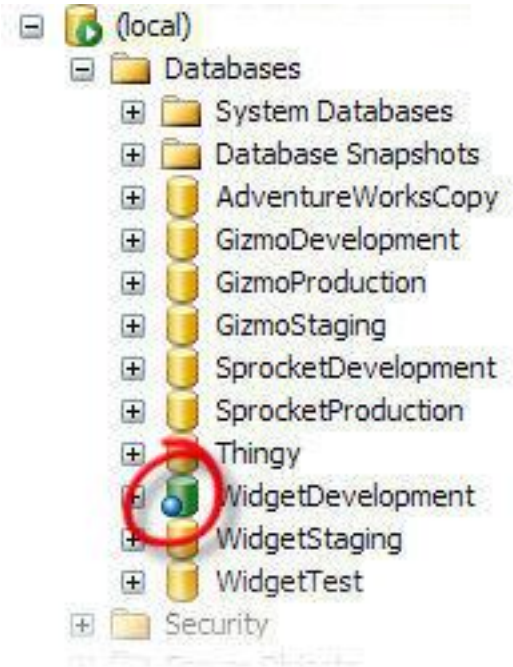
4. Under **Source control system**, on the left hand side, ensure *Team Foundation Server* is selected.
5. In **Server URL,** type or paste the URL for the server you are using if it is not already filled in.
6. In **Source Control Folder**, type or paste the location of the WidgetDev folder you created.

   Alternatively, click **Browse** and browse to the location of the folder.
7. Click **Create Link**

   You may be prompted for login credentials for your source control repository.

   A link to source control is created, and SQL Source Control is now able to determine differences between the database and source control.

The database icon in the Object Explorer changes, indicating that the database is linked to source control, and that there are changes to commit:



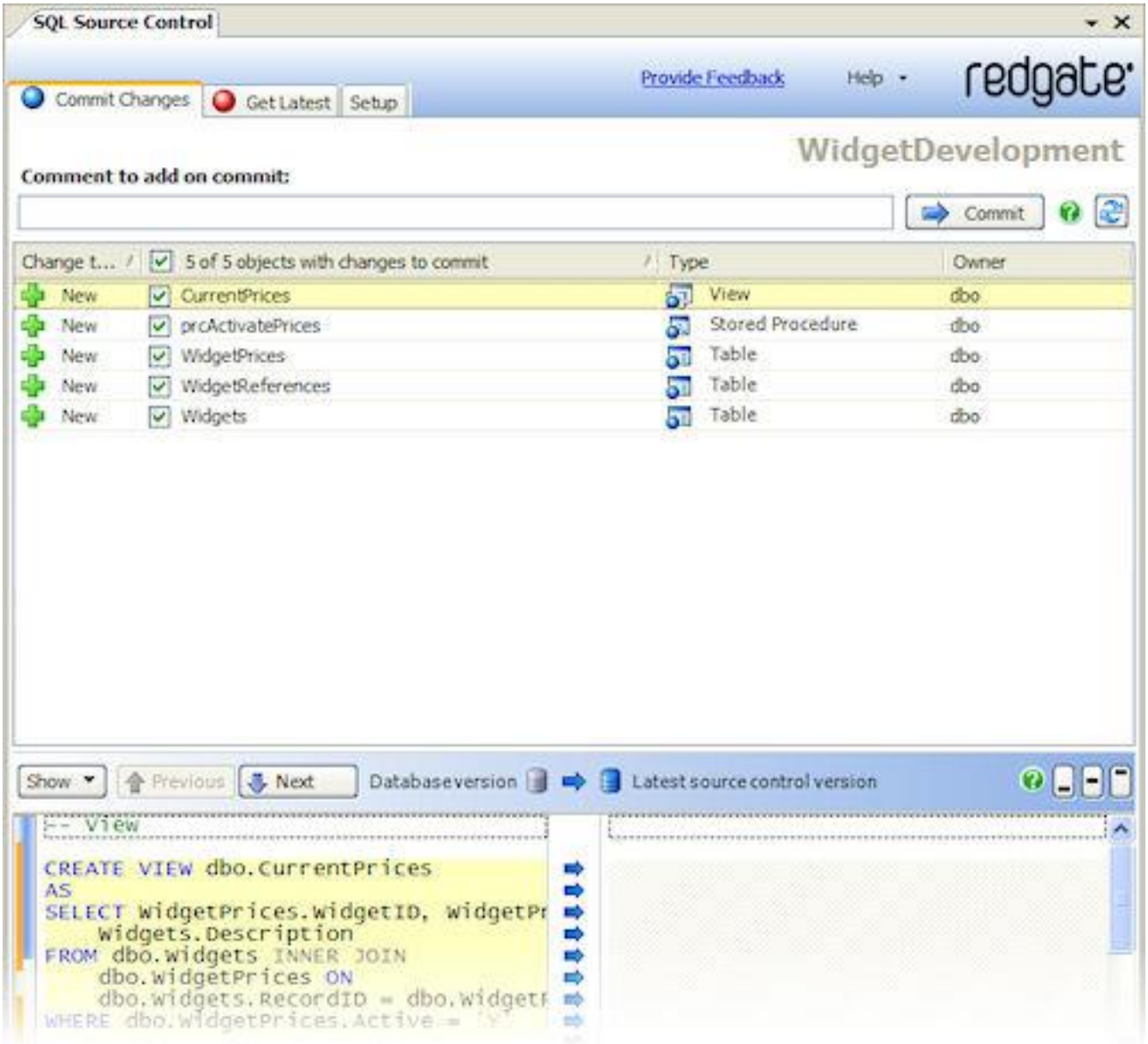
Note that the database objects have not yet been committed to source control.

3. Commit the database objects to source control

To complete the process of source controlling the database, commit the objects:

1. In SQL Source Control, click the **Commit Changes** tab.

The Commit Changes tab displays a list of all the objects with database versions that do not match the latest source control version:



Because none of the objects yet have versions in source control, they are all listed.

You can view the creation script for an object by clicking it. The script is displayed in the Object Differences pane, below the list of objects to commit.

2. In **Comment to add on commit**, type or paste a comment.

   Comments are recommended as they help provide a detailed change history.

   In this example, type *Initial commit of all objects*.

3. Click **Commit**

   SQL Source Control displays a message dialog box showing the progress of the commit.

When the commit is complete, click **OK** to close the message box.

The objects are committed to source control and other users can now get the latest version of the database.

The Object Explorer is updated to show that there are now no outstanding changes to commit.

4. Get the latest version

Now the database is in source control, another user can get the latest version, and begin making development changes.

They create a new database, link it to source control, and update it with the latest version of all the objects.

Linking the database

1. In SQL Server Management Studio, create a new database, and call it *WidgetDevelopment*
2. In the **Object Explorer**, ensure *WidgetDevelopment* is selected. On the **Setup** tab, click **Link to a database already in source control**.

   The Link Database to Source Control dialog box is displayed.
3. Under **Source control system**, on the left hand side, ensure *Team Foundation Server* is selected.
4. In **Server URL**, type or paste the URL for the server you are using if it is not already filled in.
5. In **Source Control Folder**, type or paste the location of the WidgetDev folder.

   Alternatively, click **Browse** and browse to the location of the folder.
6. Click **Link**

   You may be prompted for login credentials for your source control repository.

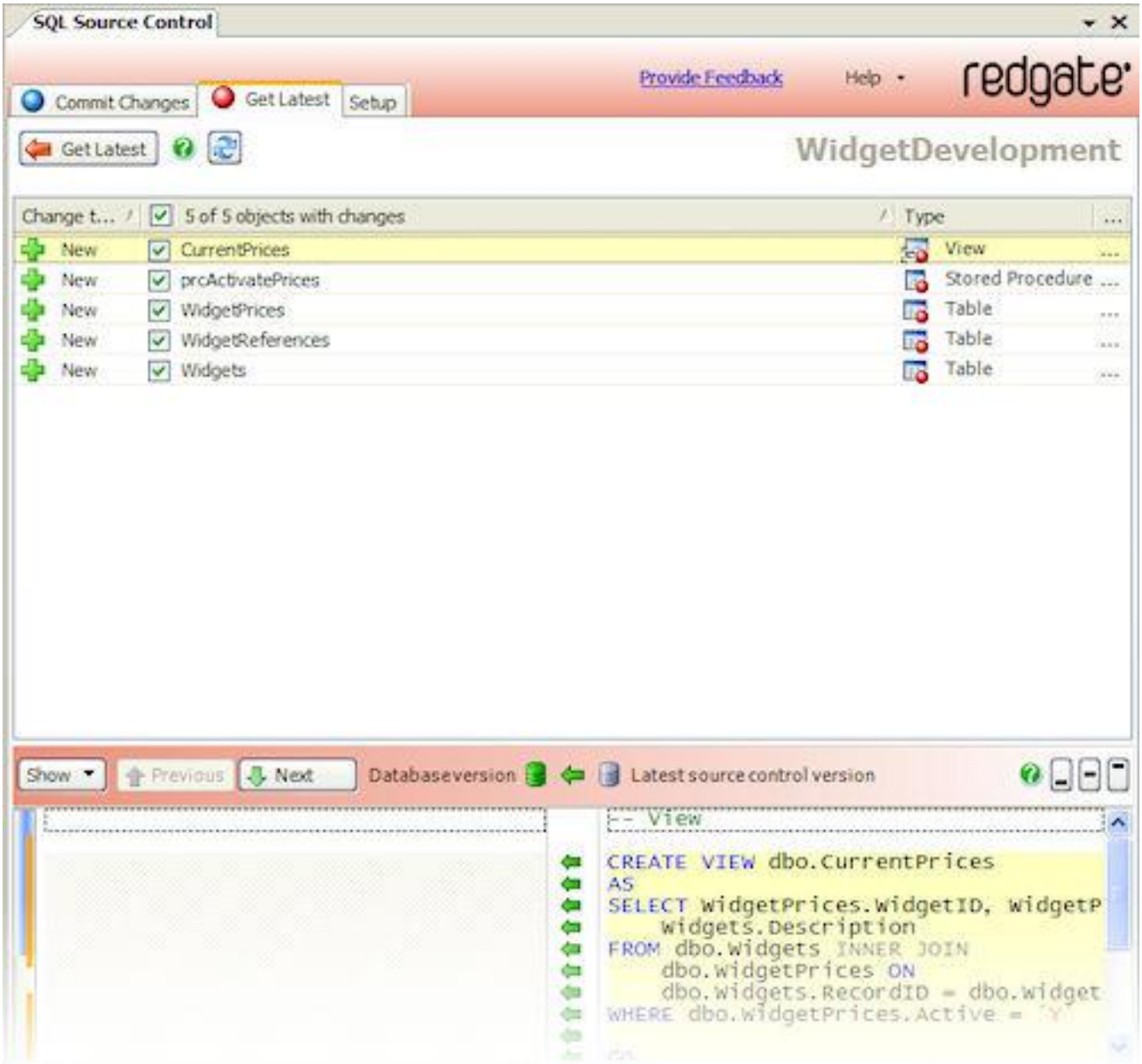
   A link to the repository is created, and SQL Source Control is able to determine differences between the database and the repository.

   However, the database has not yet been updated with the objects from source control.

Getting the latest version

1. Click the **Get Latest** tab.

Because you do not yet have any of the objects in your database, all the objects in *WidgetDevelopment* are listed here.



2. Ensure all of the objects are selected.
3. Click **Get Latest**

   A progress dialog box is displayed while SQL Source Control updates the database.

   The database is updated to the latest version.

5. Make development changes

Development proceeds normally, and the database is modified.

A column is changed in the table *Widgets*, to allow longer widget descriptions. This change is committed to source control.

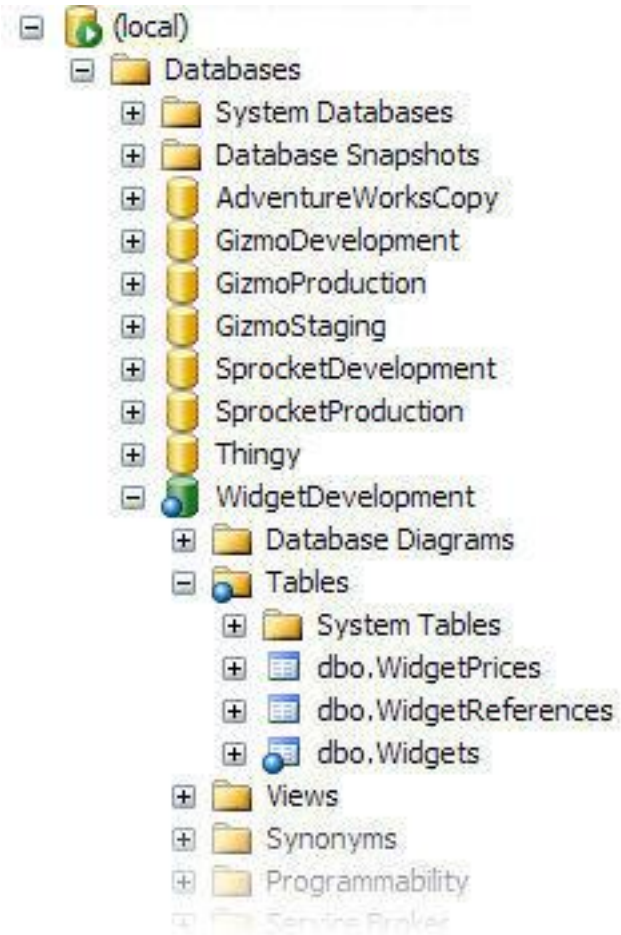
1. In SQL Server Management Studio, open a new query window, and type or paste the following SQL statement:

```
USE WidgetDevelopment
GO
ALTER TABLE [dbo].[Widgets] ALTER COLUMN [Description] [nvarchar] (100)
```

2. Click **Execute** or press **F5**

   The script runs; the database is updated.

3. SQL Source Control detects the change to the database, and highlights the affected object in the Object Explorer:

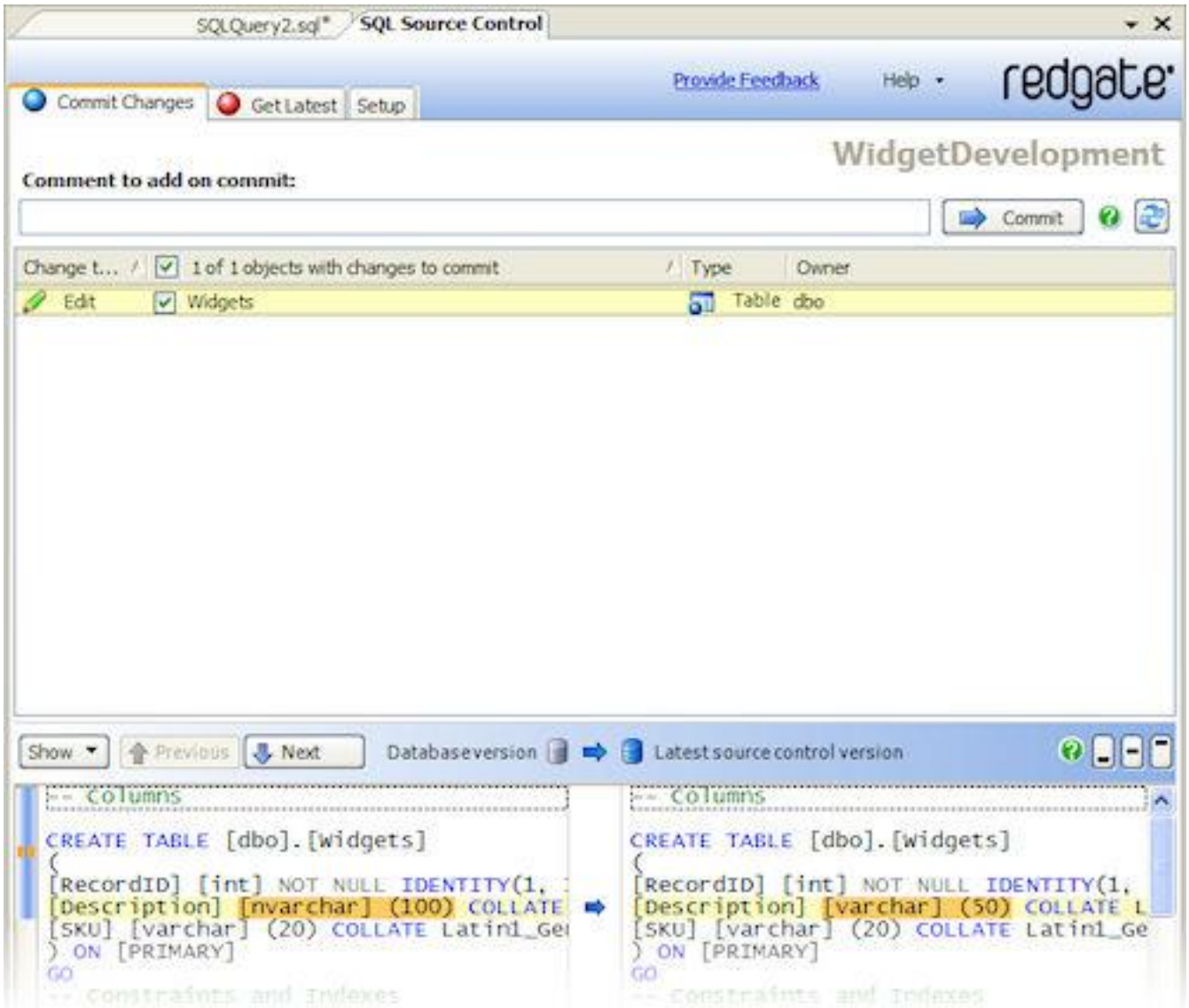


You can now commit the change.

6. Commit the development changes to source control

1. In the Object Explorer, right click the table *Widgets,* and click **Commit changes to source control**

The Commit changes tab is displayed:



The change to the table *Widgets* is listed as an edit you can commit.

The Object Differences pane shows the difference between your database version, and the latest version in source control.

2. In **Comment to add on commit**, type or paste a comment.

   In this example, type *Modified Description column*

3. Click **Commit**

   SQL Source Control displays a message dialog box that shows the progress of the commit.

   When the commit is complete, click **OK** to close the message box.

Your change is committed to source control. The Commit Changes tab lists no objects with changes to commit, and no objects are highlighted in the Object Explorer.

Other users can now get the latest version of the table.

Linking a database to source control

Linking associates a database with a location in source control, allowing SQL Source Control to monitor changes.

Linking is the first step to source controlling you database, and must occur before you can commit the database objects.

When linking a database, you have two options:

- **Create a new link to source control**

  This is used to source control a database for the first time.

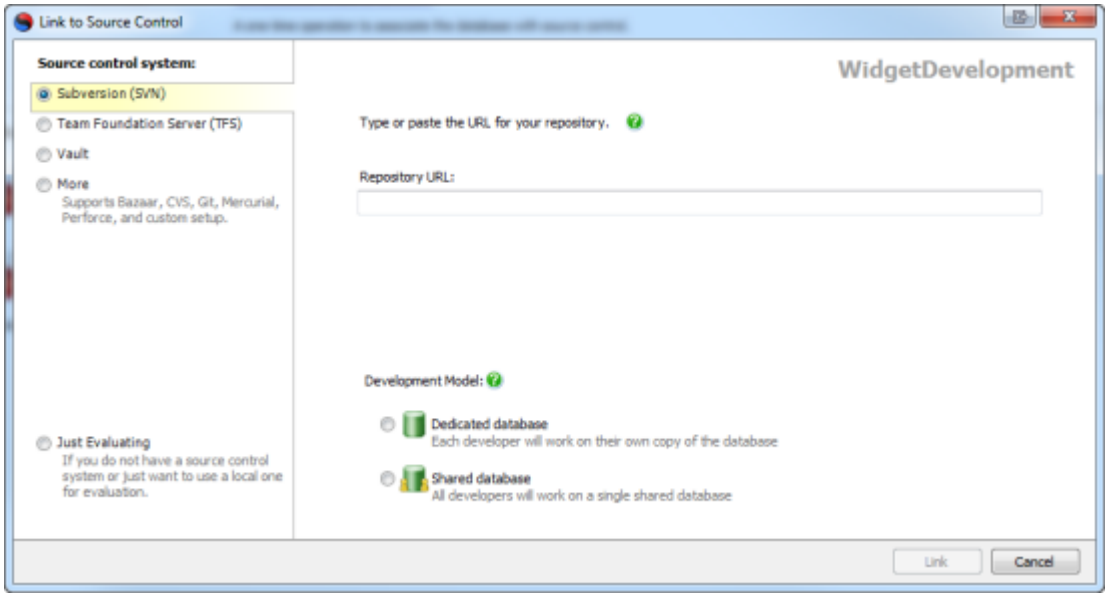
- **Link to a database that is already in source control**

  This is used to associate your version of a database with one that is already source controlled, or to get a new database from source control.

To link a database to source control:

1. In SQL Source Control, on the **Setup** tab, ensure a database is selected, and click either **Create new link to source control**, or **Link to a database already in source control**.

   The Link Database to Source Control dialog box is displayed:



2. On the left hand side, select your source control system; either Subversion (SVN) or Team Foundation Server (TFS).

3. Provide details for the source control location you want to link the database to.

   For Subversion:

   Type or paste the URL for your Subversion repository.

   If you are creating a new link, you must specify an existing, empty folder in your Subversion repository. SQL Source Control does not create the folder for you.

For Team Foundation Server:

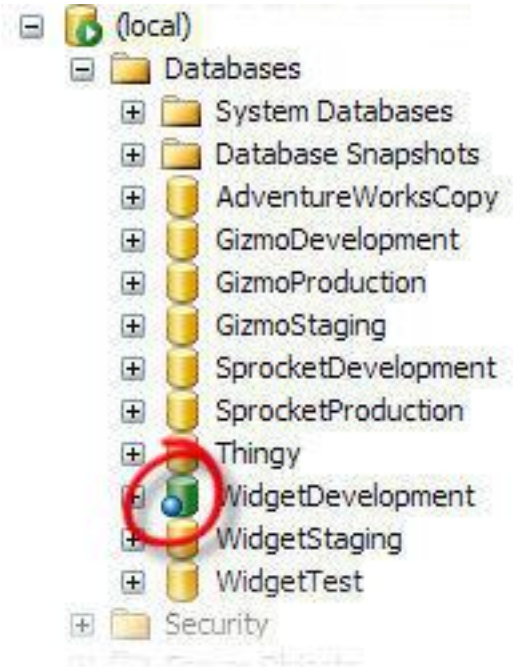
Type or paste the URL of your Team Foundation Server and the port number to connect to that server; type or browse to the location of the Team Project and solution you want to associate with the database.

If you are creating a new link you must specify the location of an existing, empty source control folder. You are recommended to create a new folder. SQL Source Control does not create it for you.

4. Click **Link**

   The database is linked to source control.

   The database icon in the Object Explorer changes to show that the database is linked:



Note that linking only associates the database with a location in source control.

If you are creating a new link, the database objects are not yet source controlled.

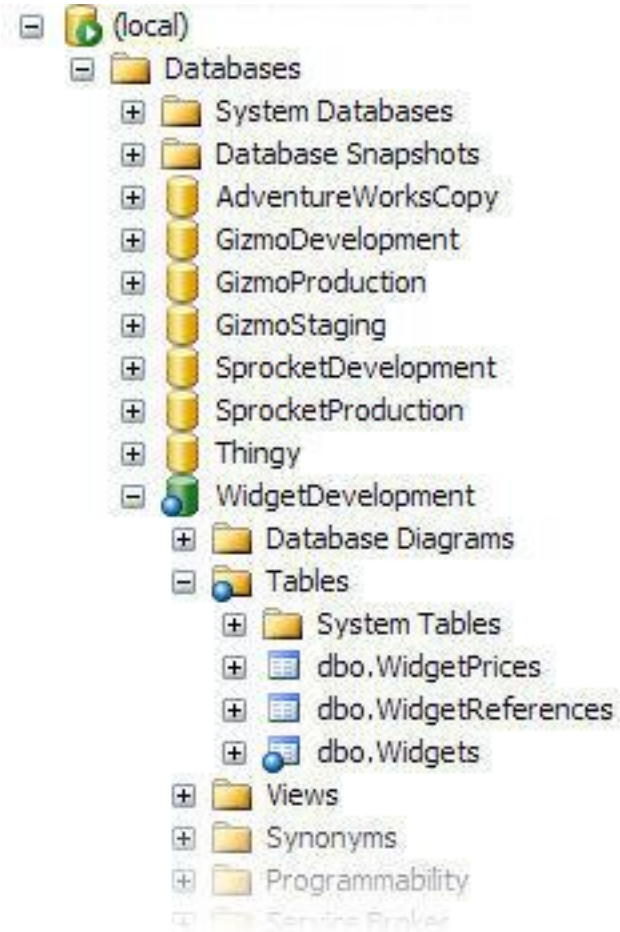
If you are linking to an existing version, your database has not yet been updated to that version.

To commit the objects or get the latest version, next go to either the **Commit Changes** tab or the **Get Latest** tab.

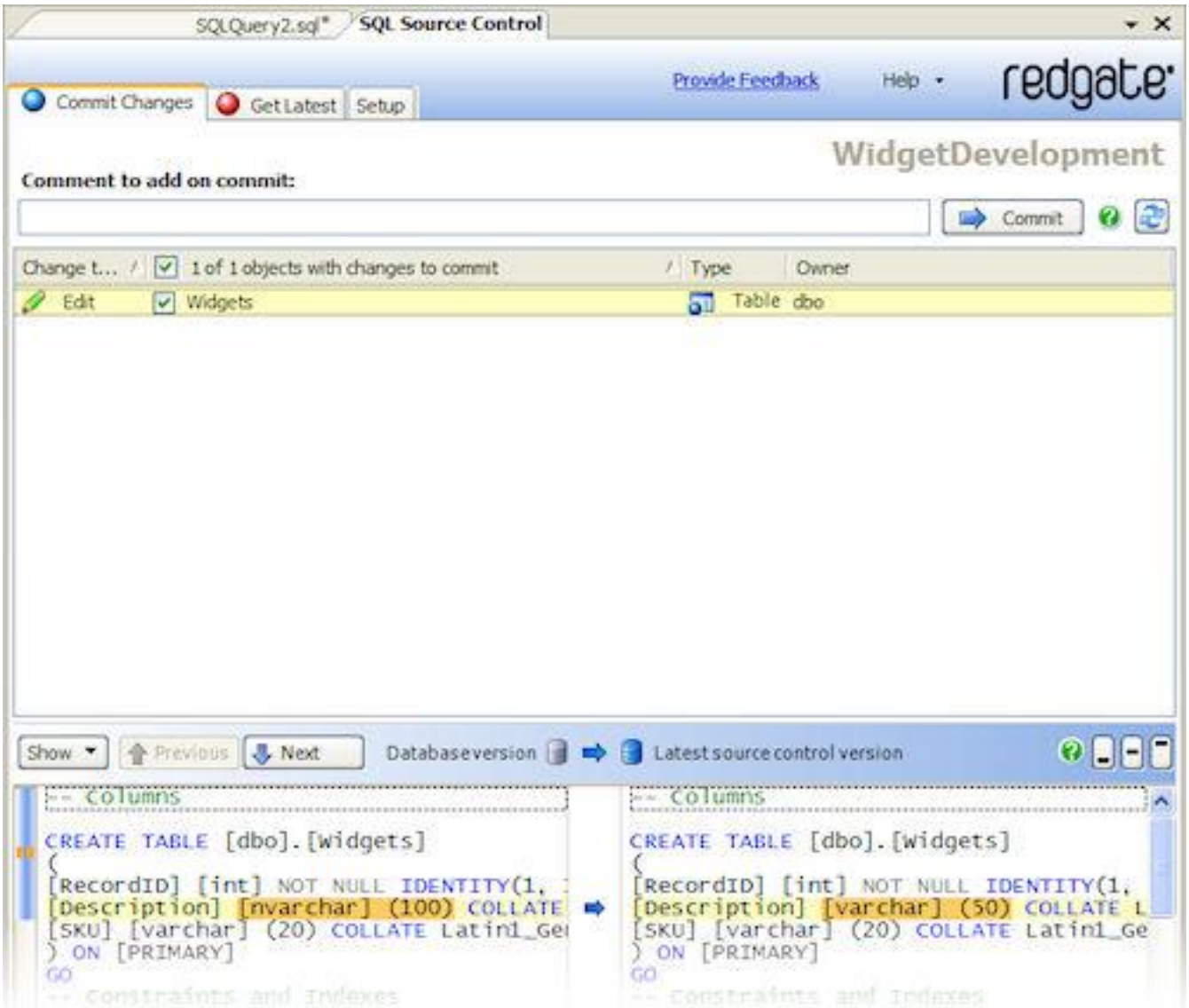
Committing changes

Committing a change updates the source control version of the database with that change.

When you make changes to a database that is linked to source control, SQL Source Control highlights affected objects in the Object Explorer:



To update source control with your changes, click the **Commit Changes** tab:



The Commit Changes tab lists all the objects in your database that anyone has made changes to.

To commit a changes:

1. Select the objects you want to commit.
2. Optionally type or paste a comment.
3. Click **Commit**

   A progress dialog box is displayed while SQL Source Control commits the changes to your source control system.

   Click **OK** to close the dialog box.

Source control is updated with your changes.

You can also commit changes by right-clicking an object, folder, or database in the Object Explorer, and clicking **Commit changes to source control**. The **Commit Changes** tab is displayed. The objects you clicked selected to commit.

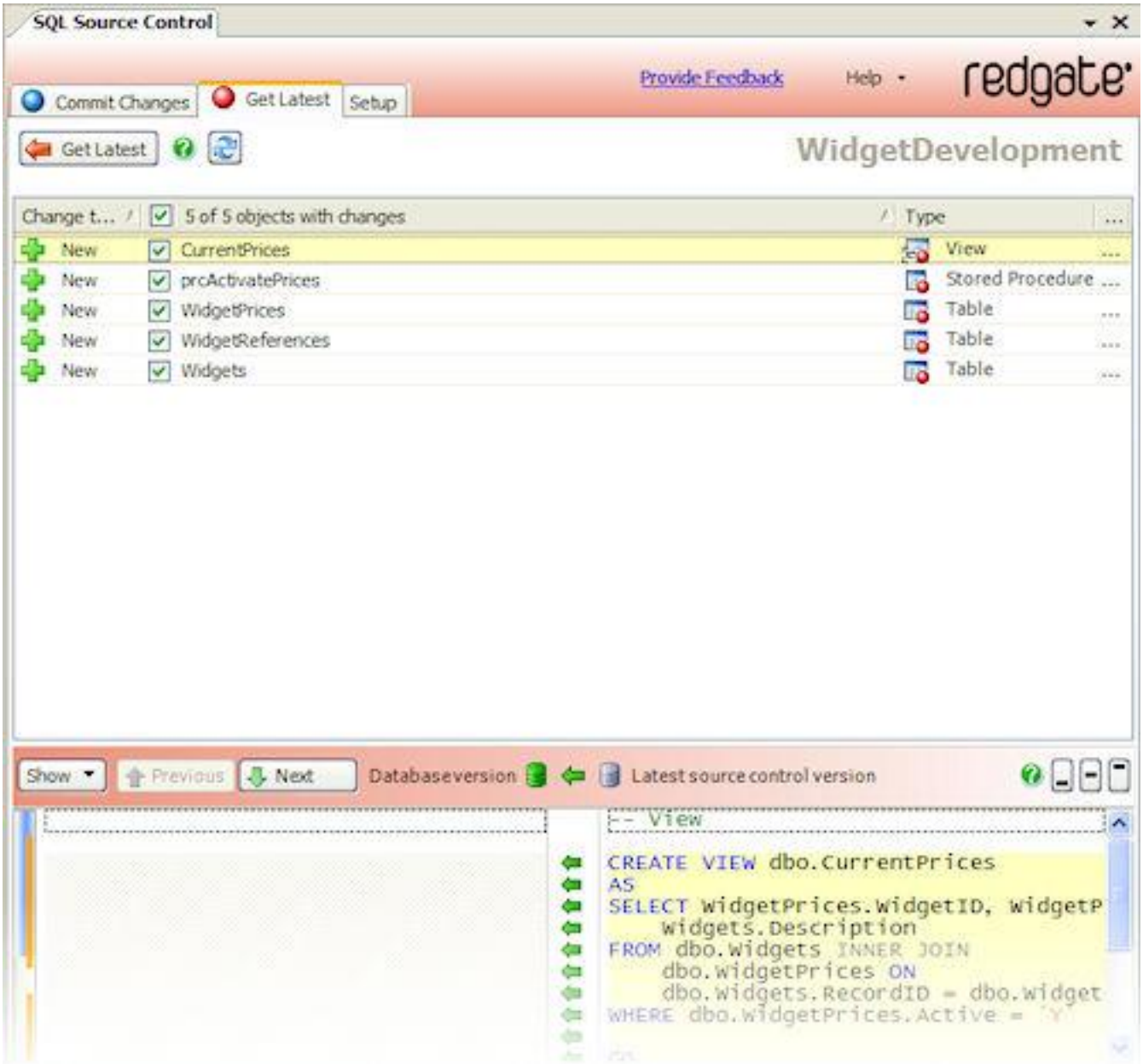
Integrating with bug tracking systems

SQL Source Control enables you to integrate your database changes with a bug or issue tracking system (for example Jira, or FogBugz), or the project management features of Team Foundation Server. It does this by letting you associate a commit with an SVN bug ID or TFS work item.

For more information, see: Using SVN Bug IDs & TFS Work Items

Getting the latest version

If an object has a more recent source control version than the version currently in the database, you can get that version on the **Get Latest** tab:



To get the latest version of an object:

1. On the **Get Latest** tab, select the objects you want to update to the latest version
2. Click **Get Latest**

   A progress dialog box is displayed while SQL Source Control updates the database.

The database is updated to the latest version.

You can also get the latest version by right-clicking an object, folder, or database in the Object Explorer, and clicking **Get latest changes from source control**. You are taken to the **Get Latest** tab, where the objects you clicked selected.

Viewing source control logs / history

SQL Source Control does not currently support viewing logs or change history information within SQL Server Management Studio.

To see a history of what changes were made, by whom, and why, use your source control system.

Viewing logs with Subversion

You can view the Subversion logs for a folder or object using the Subversion command line interface or a client such as Tortoise SVN (http://tortoisesvn.net/downloads)

To view logs with Tortoise SVN:

1. In a Windows Explorer window, right-click, and from **TortoiseSVN** select **Show log**

   The Log Messages dialog box is displayed, showing the revision number, type of revision, user, date, and associated comment for each modification.

Viewing the history with Team Foundation Server

You can view the TFS history using the Team Explorer client for Visual Studio.

To view the history:

1. In the Visual Studio Source Control Explorer, on the **Folders** pane, right click the object or folder you are interested in.
2. Click **View History**.

   The History pane is displayed, showing the changeset, user, date, and associated comment for each change.

   To see more detailed information, in the **History** pane, right click a change, and click **Details**.

Getting a specific version

SQL Source Control currently only supports getting the latest version of a database within SQL Server Management Studio. To update your database to a specific version or revision, create a local copy and synchronize it with the database using SQL Compare (http://www.red-gate.com/products/sql-development/sql-compare/)

Example: getting a specific revision with TortoiseSVN

This example creates a local copy of the revision, and synchronizes the database using SQL Compare:

1. Check out the latest version to a new folder.
2. Right-click the folder, and from the **TortoiseSVN** menu, select **Show log**

   The Log Messages dialog box is displayed.
3. Select the revision you want to get, right-click, and click **Revert to this revision**
4. A confirmation dialog box is displayed. Click **Yes**

   The folder of scripts is updated to the revision you selected.
5. Using SQL Compare, set the scripts folder as the source for a comparison, and the database as the target, then compare and synchronize.

   The database is updated to the revision you selected.

Example: getting a specific changeset with TFS

This example updates the TFS local copy to a specific changeset, and synchronizes the database using SQL Compare:

1. In Visual Studio, in the **Source Control Explorer** tab, select the database, right-click, and click **Get Specific Version**

   The **Get** dialog box is displayed.
2. Under **Version**, in **Type** select *Changeset*
3. Type the changeset number, or click the browse button to display the Find Changesets dialog box, and select the changeset you want.
4. Click **Get**

   The local scripts folder has been updated to the changeset you selected.
5. Using SQL Compare, set the local scripts folder as the source for a comparison, and the database as the target, then compare and synchronize.

   The database is updated to the revision you selected.

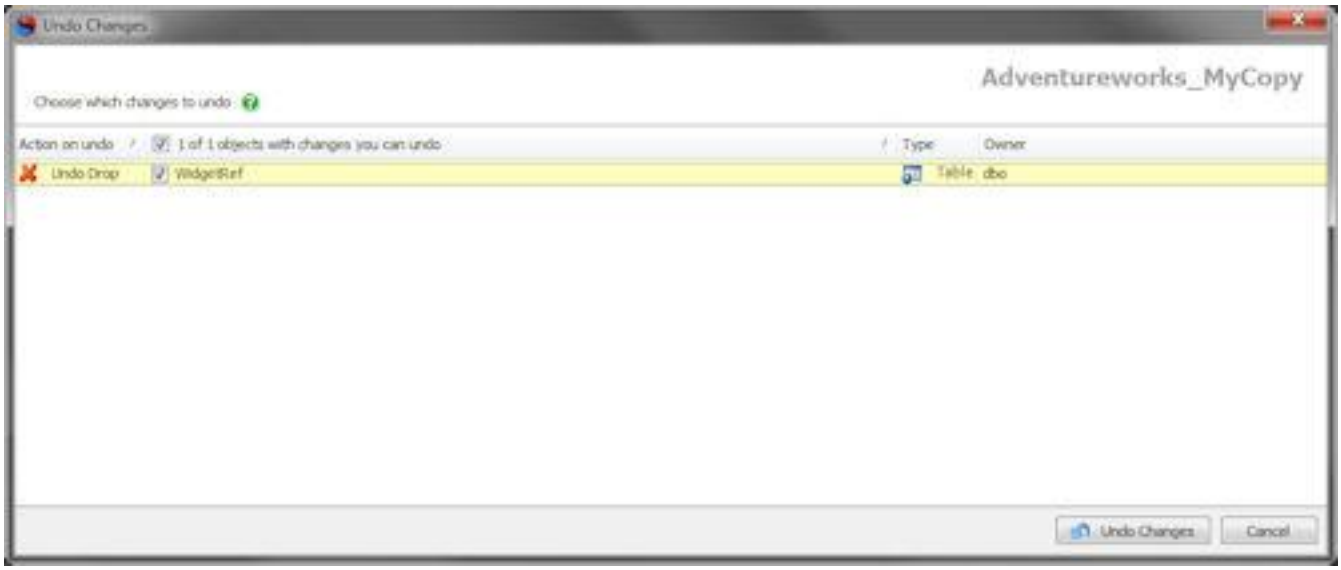
Undoing changes

SQL Source Control enables you to undo schema changes you have made to a database, but not yet committed to source control.

To undo a change:

1. In the Object Explorer, select the object, folder, or database with changes you want to undo.
2. Right-click, select **Other SQL Source Control Tasks** and click **Undo changes**

   The Undo Changes dialog box is displayed:



3. Select the objects with changes you want to undo, and click **Undo Changes**

   A progress dialog box is displayed while SQL Source Control runs the script to undo the changes.
4. Close the dialog box when the operation is complete.

   The changes are undone.

Changes you cannot undo

The most common types of change SQL Source Control cannot undo are listed below:

- **Committed changes**

  You can only use SQL Source Control to undo changes you have not committed. To undo changes that have been committed, use your source control system.

  For example, to undo using TortoiseSVN:

1. Check out the latest version to a new folder.

2. Right-click the folder, and from the **TortoiseSVN** menu, select **Show log**

   The Log Messages dialog box is displayed.

3. Select the revision that does not contain the changes you want to undo, right-click, and click **Revert to this revision**

4. A confirmation dialog box is displayed. Click **Yes**

   The folder of scripts is updated to the revision you selected.

5. Right-click the folder and click **SVN Commit**

   The Commit dialog box is displayed.

6. Click **OK**

   The latest source control version is updated to the revision you selected, so the unwanted changes are undone in source control.

7. In SQL Source Control get the latest version.

   This updates the database, so the unwanted changes are undone in the database version.

- **Dropped data**

  If you drop a table or column that contained data, the data is not restored when you undo the drop.

- **NOT NULL columns**

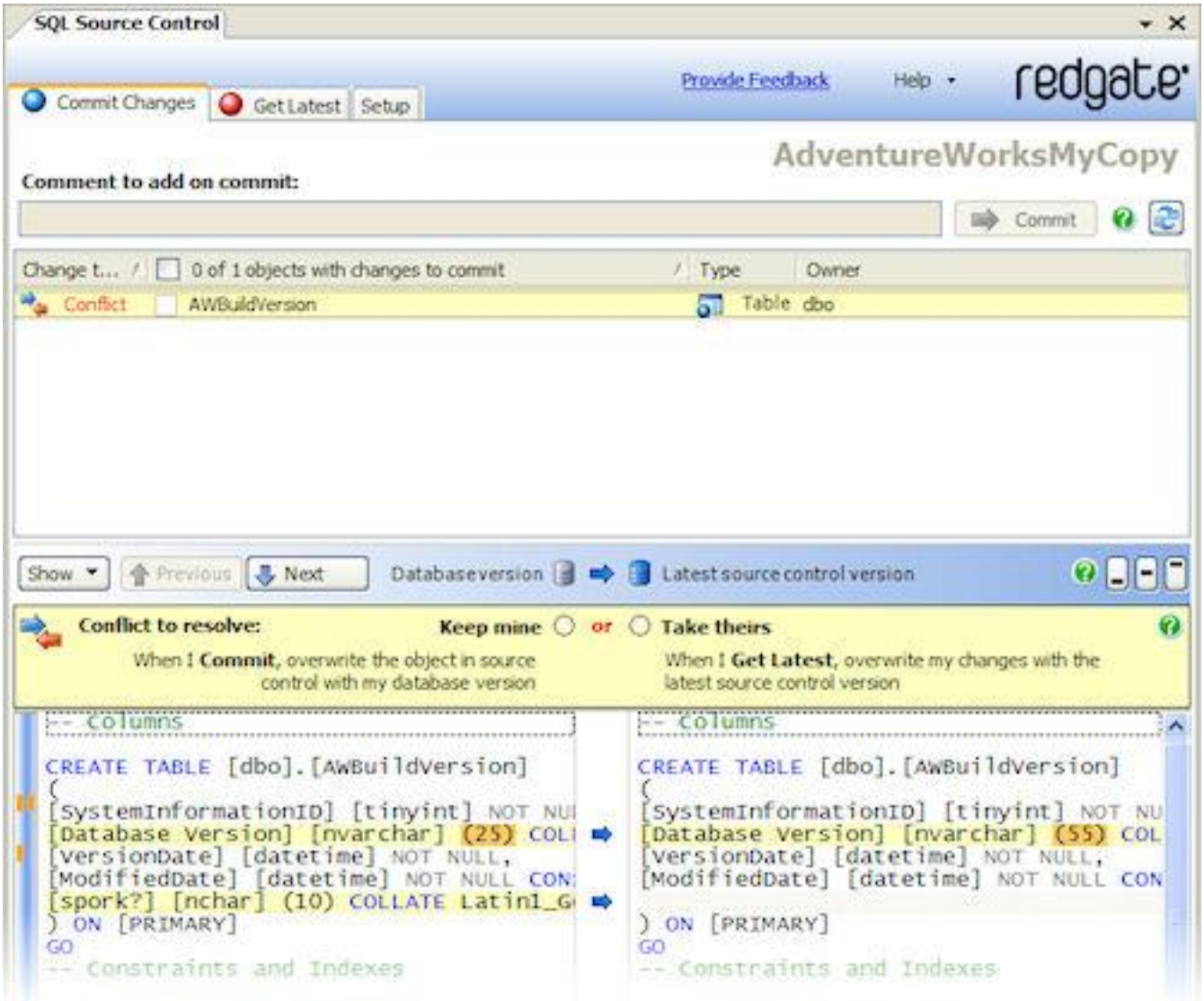
  If you drop a NOT NULL column from a table that contained data, undoing the drop will fail if the column does not have a default value.

Conflicts

Conflicts occur when the latest version of an object in source control and the latest version in the database are incompatible. Conflicted objects cannot be committed or retrieved until the conflict has been resolved.

This can happen, for example, if two people modify the same stored procedure.

When a conflict occurs, the **Conflict to resolve** bar is displayed in the Object Differences pane:



To resolve a conflict, choose either:

- **Keep mine**

  Your changes are committed to source control.

This replaces any changes to the latest source control version that are not present in your database version.

Or

- **Take theirs**

  Your changes are discarded, and the database is updated to the latest source control version.

Note that merging is not currently supported.

To merge changes from two conflicting versions of an object, resolve the conflict, and then manually edit the object to include the changes from the other version.

To make this easier, you can copy either version of an object's creation script from the Object Differences pane, and paste it into a new query window.

To avoid conflicts, you are recommended always to get the latest version of an object before modifying it.

Deploying a database from source control

You can use SQL Source Control and SQL Compare (http://www.red-gate.com/products/SQL_Compare/index.htm) to deploy a database from source control to a server.

The SQL Source Control scripts folders share the format used by SQL Compare and SQL Data Compare, so deployment is relatively simple.

To deploy a database:

1. Create a local copy of the scripts folder
2. Migrate the local copy to the target server using SQL Compare

Optionally, you can also deploy any relevant static data using SQL Data Compare.

Example: deploying a database

In this example the database *WidgetDev* is already in source control.

This example uses the SQL Compare and Subversion command line interfaces.

It is also possible to deploy the database using the SQL Compare graphical user interface, and a source control client such as TortoiseSVN (http://tortoisesvn.net/downloads)

1) Create a local copy of the database

At a command prompt, type:

```
cd C:\program files\subversion\bin

svn update http://svn-eval.red-gate.com/svn/<project>/trunk/WidgetDev
"C:\WidgetDevScripts"
```

Where:

- *http://svn-eval.red-gate.com/svn/<project>/trunk/WidgetDev* is the URL for the database in your Subversion repository
- *"C:\WidgetDevScripts"* is the file path for the directory where the local copy will be created

A local copy of the scripts folder is created. This is a Subversion working copy, and is associated with the Subversion repository.

2) Migrate the local copy to the target server

At a command prompt, type:

```
cd C:\program files\red gate\SQL Compare 8
     /sqlcompare /scr1:"C:\WidgetDevScripts"
     /S2:WidgetServer /U2:<username> /P2:<password>
```

```
/db2:"WidgetTest"
/sync
```

Where:

- */scr1:"C:\WidgetDevScripts"* specifies the local copy, WidgetDevScripts, as the source for a SQL Compare synchronization
- */S2:WidgetServer /U2:<username> /P2:<password>* specify the server, user name, and password you are using
- */db2:WidgetTest* specifies WidgetTest as the target of a SQL Compare synchronization
- */sync* performs the SQL Compare synchronization, making schema of WidgetTest the same as the schema in WidgetDevScripts

The database is updated. Its schema is now the same as the version you checked out of source control.

For more information, see Simple examples using the SQL Compare command line (http://www.red-gate.com/supportcenter/Content.aspx?p=SQL%20Compare&c=SQL_Compare/help/9.0/sc_cl_examplesusingthecl.htm&toc=SQL_Compare/help/9.0/toc846154.htm)

Setting up a local Subversion repository

This topic describes how to set up a local Subversion repository using TortoiseSVN, a free Subversion client for Windows.

To download the latest version of TortoiseSVN, see the TortoiseSVN download page (http://tortoisesvn.net/downloads)

Alternatively, you can use the Subversion command line interface.

For more information, see the Subversion documentation (http://svnbook.red-bean.com/)

Creating a repository

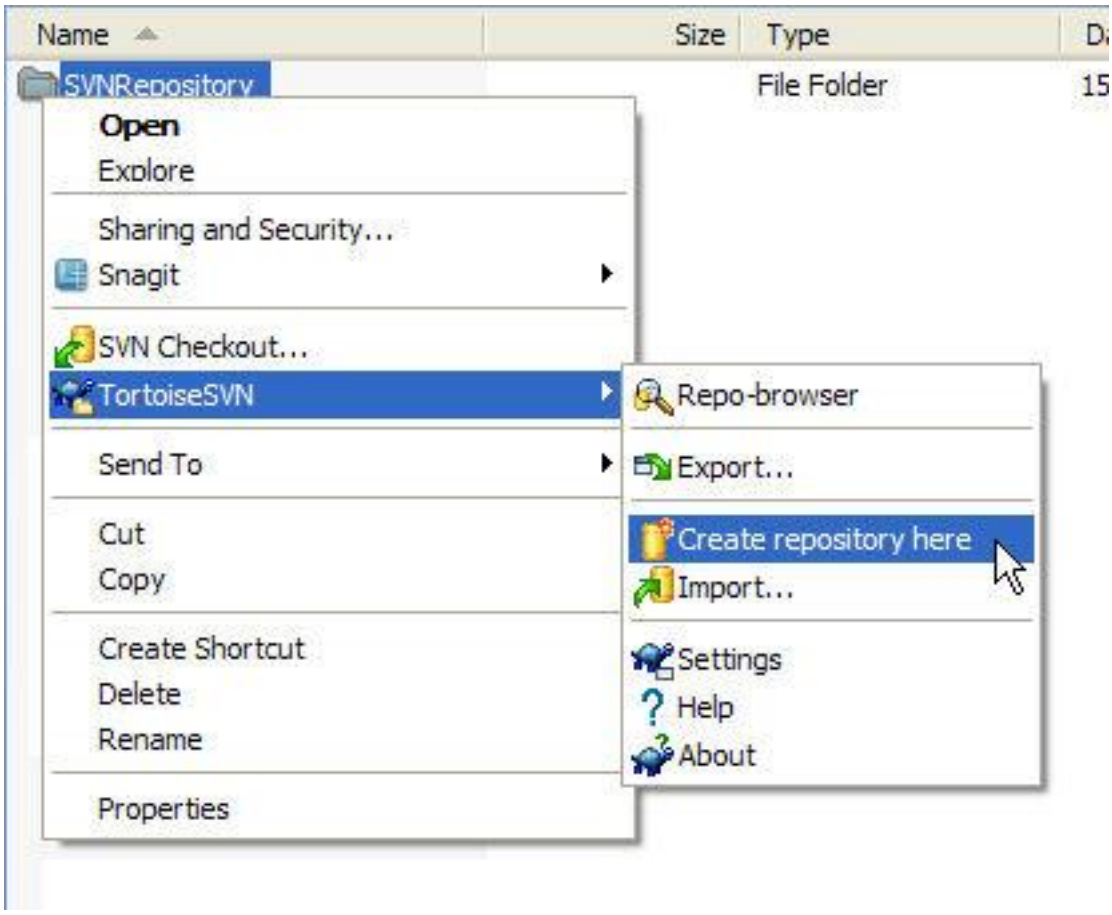
To create a local repository:

1. Download and install Tortoise SVN.

   You may need to restart your computer after installation.

2. In Windows Explorer, browse to or create an empty folder where you want to create the repository, for example *C:\SVNRepository*

3. Right-click the folder, and in the TortoiseSVN menu, select **Create repository here**:



The repository is created.

Using the repository

You can now use the repository with SQL Source Control.

The URL for a local repository takes the form:

file:\\\C:\<RepositoryFilePath>

Use this URL to link your database to source control

Note that the URL is case sensitive.

Continuous integration

SQL Source Control can be used with SQL Compare Pro and, optionally, SQL Data Compare Pro to implement continuous integration for databases.

This is discussed in detail in the article: Continuous integration for databases using Red Gate SQL tools (pdf) http://downloads.red-gate.com/HelpPDF/ContinuousIntegrationForDatabasesUsingRedGateSQLTools.pdf

Source controlling data

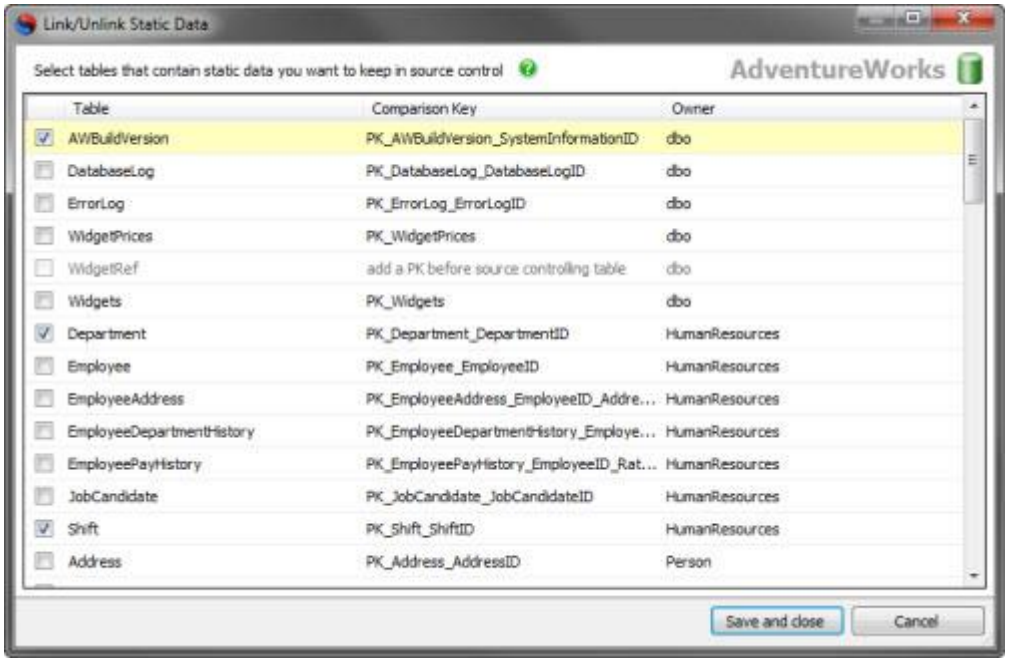
SQL Source Control lets you source control your static (lookup/reference) data. Static data is the non-transactional data an application may depend on. It typically does not change frequently; an example would be a table of US states.

To source control data, first link it to source control, then commit it.

Linking associates a table's data with source control, and means that subsequent data changes are detected, and static data can be shared across the team.

To source control a table's data:

1. In the Object Explorer, right-click the database or table with data you want to source control.
2. Select **Other SQL Source Control Tasks**, then click **Link/Unlink Static Data**
3. The Link/Unlink Static Data dialog box is displayed:



   You can see a list of the tables in the database and select the ones you want to link to source control.

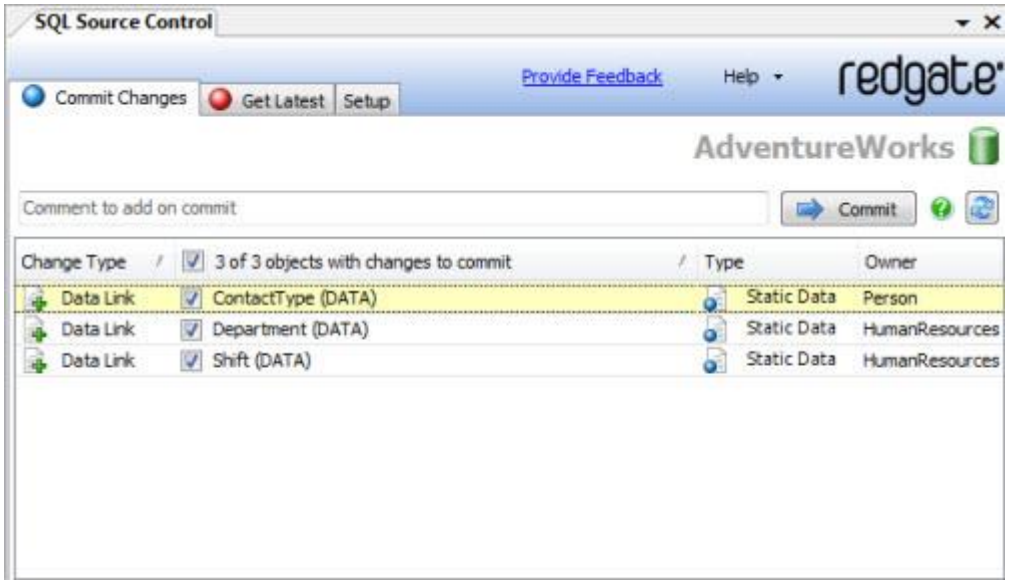
   Note that you can only source control data in tables with a valid primary key.

   The primary key is used as the comparison key (/supportcenter/Content?p=SQL Data Compare&c=SQL_Data_Compare/help/9.0/9067.htm&toc=SQL_Data_Compare/help/9.0/toc1419819.htm) to identify corresponding rows.

4. Select the tables you want to link and click **Save and close**

A file for the table's data has now been created, but the data itself has not yet been committed. To commit the data, commit the resulting *Data Link* change on the Commit Changes tab.

5. On the Commit Changes tab, ensure all Data Link changes are selected, optionally type a comment, and click **Commit**



The data is committed to source control.

Note that committing and getting the latest version of data can be slow. It is therefore recommended only to source control static (lookup) data.

Evaluation center

To learn more about evaluating SQL Source Control, see the Evaluation center (http://www.red-gate.com/products/sql_source_control/evaluation_center/index.htm)