

# Migrating Databases to SQL Azure

## Overview

SQL Azure Database is based on Microsoft SQL Server, which makes it relatively easy to move existing SQL Server databases to SQL Azure. In this hands-on lab, you will learn how to move a schema and data to SQL Azure by using brute force Scripts with BCP command-line tool to copy data into and out of SQL Azure.

### Objectives

In this hands-on lab, you will learn how to:

* Use SQL Azure as a cloud hosted database platform for your applications.
* You will learn how to
  + Migrate an existing database (schema & data)

### Prerequisites

The following is required to complete this hands-on lab:

* [SQL Server Management Studio 2008 R2 Express Edition](http://www.microsoft.com/express/Database/InstallOptions.aspx)

### Setup

For convenience, much of the code used in this hands-on lab is available as Visual Studio code snippets. To check the prerequisites of the lab and install the code snippets:

1. Open a Windows Explorer window and browse to the lab’s **Source\Setup** folder.
2. Double-click the **Dependencies.dep** file in this folder to launch the Dependency Checker tool and install any missing prerequisites and the Visual Studio code snippets.
3. If the User Account Control dialog is shown, confirm the action to proceed.

**Note:** This process may require elevation. The *.dep* extension is associated with the Dependency Checker tool during its installation. For additional information about the setup procedure and how to install the Dependency Checker tool, refer to the **Setup.docx** document in the **Assets** folder of the training kit.

### Exercises

This hands-on lab includes the following exercises:

1. Migrating an on-premise database to the cloud

Estimated time to complete this lab: **30 minutes**.

## Exercise 1: Moving an Existing Database to the Cloud

In this exercise, you will move an existing on-premise database from SQL Server to the cloud. You will do this by using the brute force approach, using scripts.

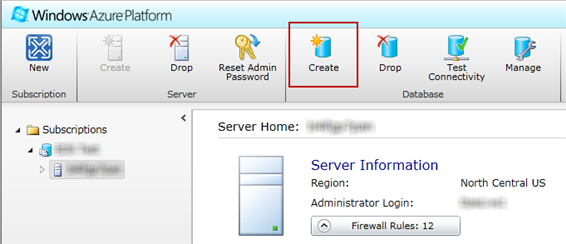
### Task 1 – Provision the destination SQL Azure database

In this step you will provision an empty SQL Azure database that you will use to migrate all of the *AdventureWorksDW2008R2* database objects contained in your local SQL Server instance.

1. Sign into the [Windows Azure Management Portal](http://windows.azure.com/), click the **SQL Azure** tab and select the **Project Name** associated with your SQL Azure subscription.

2. Make a note of the information in the **Server Information** section, including: **Server Name** (e.g. *myserver.database.windows.net*) **Administrative Username** (e.g. *mysqladminuser*)

3. In the **Ribbon** menu, click **Create**.

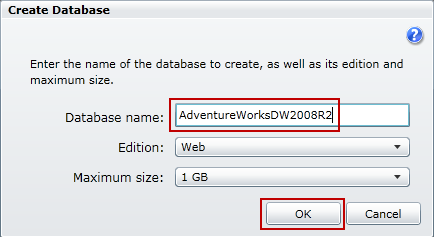


**Figure 1 - Create Database**

4. Enter the following information:

* + **Name of your database**: AdventureWorksDW2008R2
  + **Specify an edition**: Web
  + **Specify the max size**: 1 GB

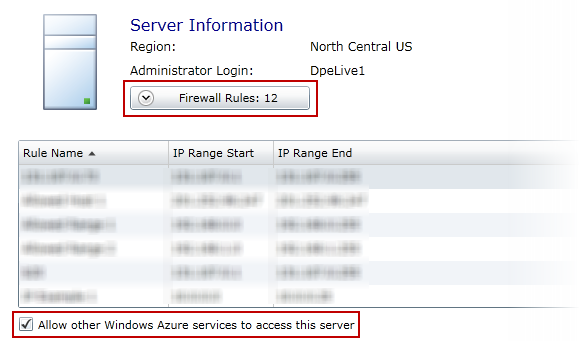
**Note**: Right now, the main difference between **Web** and **Business** database editions is [storage and price](file:///C:\WindowsAzurePlatformTrainingKit\Labs\MigratingDatabasesToSQLAzure\Lab.html\html\docSet_#sql). Web edition databases can be provisioned for up to 5 GB of relational storage. Business edition databases can be provisioned for up to 50 GB of relational storage. Web edition is significantly less expensive for paid subscriptions. As SQL Azure evolves, “enterprise” features will most likely be monetized by requiring Business edition.



**Figure 2 - Create Database**

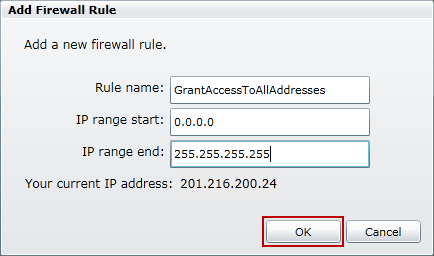
5. Click **OK**.

6. Click **Firewall Rules** button, and check the “**Allow other Windows Azure services to access this server”** option. This will allow other services running in the Azure Services environment to interoperate with your SQL Azure server.



**Figure 3 - Firewall Rules**

7. Now, click the **Add** button, and enter the correct IP address range to grant access to the IP addresses you will use to connect to SQL Azure, for example: **Rule Name**: *GrantAccessToAllAddresses* **IP Range start**: 0.0.0.0 **IP Range end**: 255.255.255.255



**Figure 4 - Firewall Rules – Add Firewall Rule**

**Note**: The example shown here grants access to all IP addresses. Grant access only to the IP addresses you actually need to avoid unauthorized usage.

### Task 2 – Generate a DDL script

In this step you will generate a DDL script capable of recreating all of the objects in your source database. DDL stands for “Data Definition Language” which is shorthand for the various Transact-SQL commands used to create database objects such as tables, views and stored procedures.

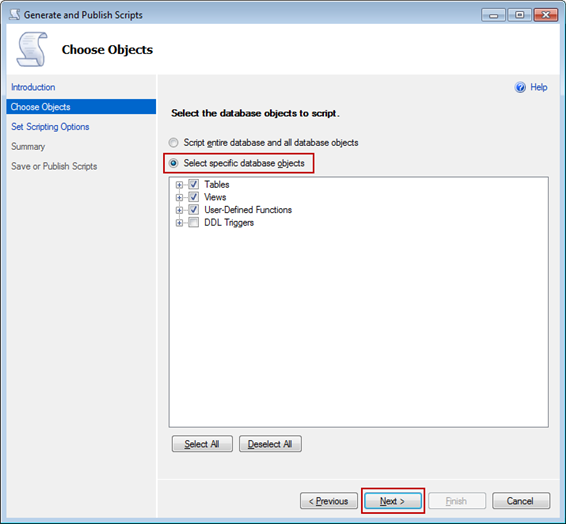
1. Launch **SQL Server Management Studio**.

2. Connect to the SQL Server instance that you installed the **AdventureWorksDW2008R2** database on.

3. Expand the **Databases** node in Object Explorer, right-click **AdventureWorksDW2008R2** and select **Tasks** > **Generate Scripts**.

4. If there appears an **Introduction** dialog, click **Next**.

5. In the **Choose Objects** dialog, choose the “**Select specific database objects**” option, then check the **Tables**, **Views** and **User-Defined Functions** options.

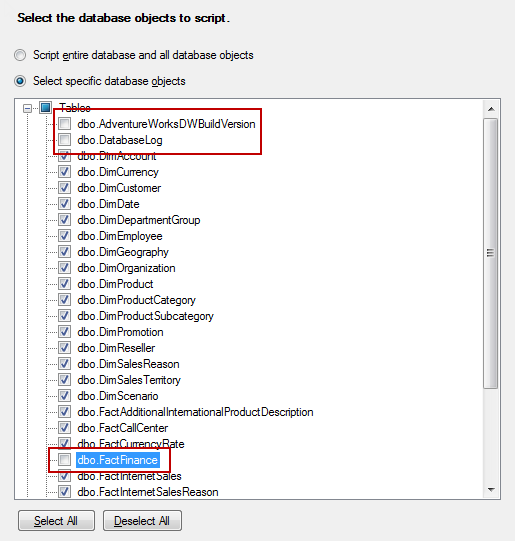


**Figure 5 - Generate Scripts – Choose Objects**

**Note**: **AdventureWorksDW2008R2** contains a DDL Trigger. SQL Azure does not currently support DDL Triggers so do not select this option.

6. Expand the **Tables** node and uncheck the following tables:

* **AdventureWorksDWBuildVersion**
* **DatabaseLog**
* **FactFinance**

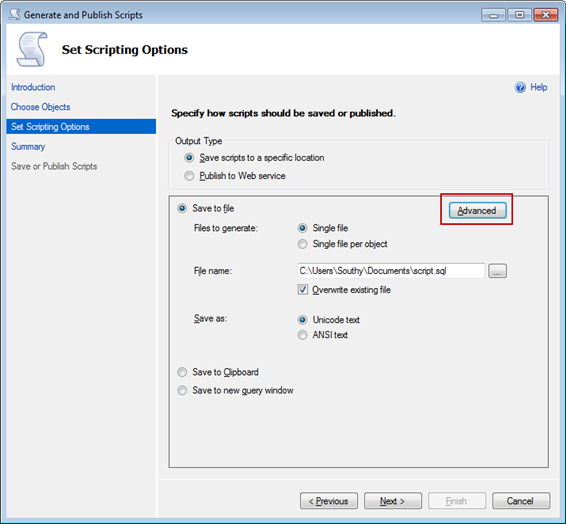


**Figure 6 - Generate Scripts – Choose Objects**

**Note**: SQL Azure requires all tables to have a clustered index. These tables do not have clustered indexes, and they are not essential for this particular sample database.

7. In the **Choose Objects** dialog, click **Next**.

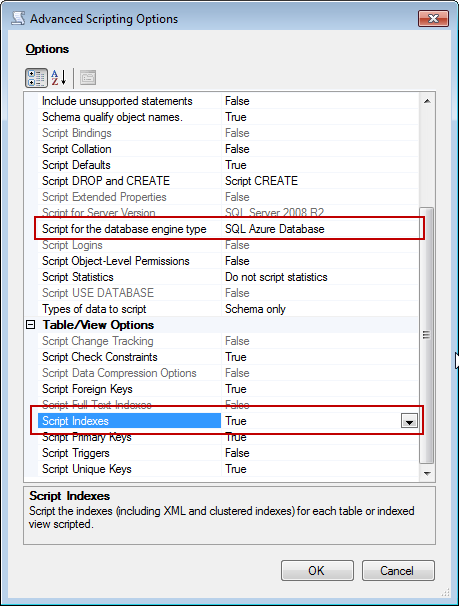
8. In the **Set Scripting Options** dialog, click the **Advanced** button and make the following changes, then click **OK**:



**Figure 7 - Generate Scripts – Choose Objects**

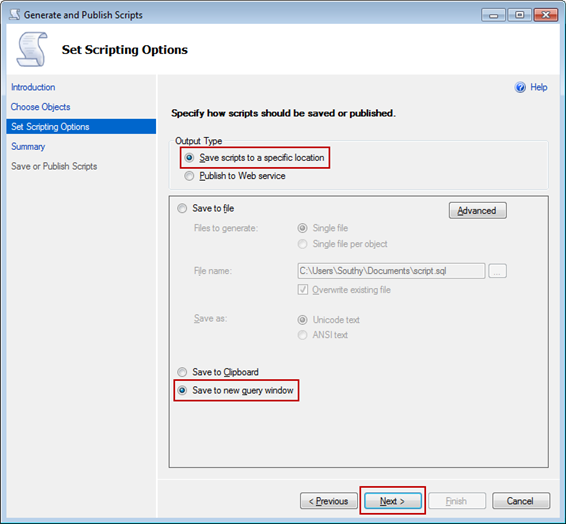
9. Set the **General** > **“Script for the database engine type”** option to “**SQL Azure Database**”

10. Set the **Table/View Options** > **Script Indexes** option to **True**



**Figure 8 - Generate Scripts – Choose Objects**

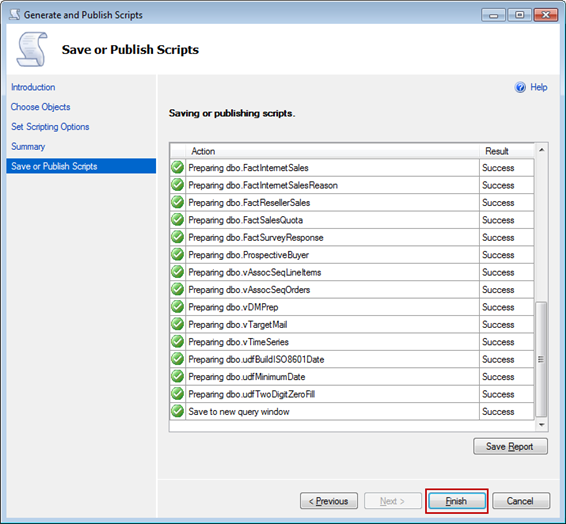
11. In the **Set Scripting Options** dialog, set the output type to “**Save scripts to a specific location**”, and choose the “**Save to new query window**” option, then click **Next**.



**Figure 9 - Generate Scripts – Set Scripting Options**

12. Review the information in the **Summary** dialog, then click **Next**. At this point the wizard will interrogate the schema of the source database and generate a script to rebuild all of the objects in the database into a new query window.

13. Review the information in the **Save or Publish Scripts** dialog then click **Finish**.



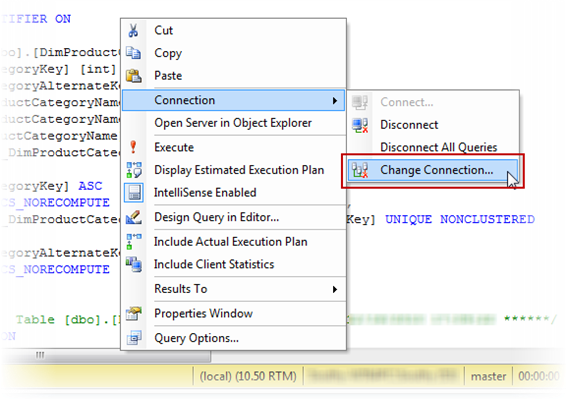
**Figure 10 - Generate Scripts – Save or Publish Scripts**

14. Return to **SQL Server Management Studio** and review the script that was generated.

### Task 3 – Execute the DDL script against SQL Azure

In this step you will execute the DDL Script you have generated against the AdventureWorksDW2008R2 database you provisioned in SQL Azure. Once completed, all of the tables, views, stored procedures and other objects from your source database will exist in your newly provisioned SQL Azure database. At this point there will be no data in the tables.

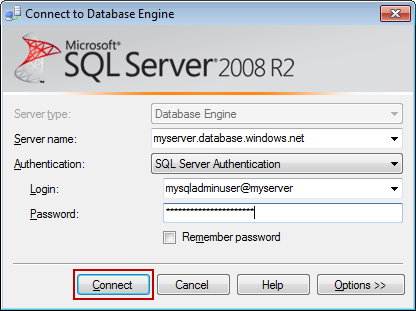
1. Right-click the query window and select Connection > Change Connection.



**Figure 11 - Generate Scripts – Choose Objects**

2. This will bring up the Connect to Database Engine dialog so you can connect to your newly created AdventureWorksDW2008R2 database in SQL Azure. Enter the connection information you noted earlier, for example:

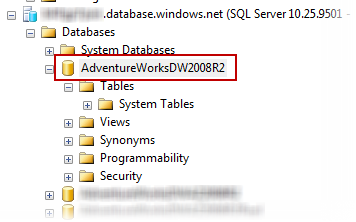
* Server name: [*[*myserver.database.windows.net]*]*
* Authentication: SQL Server Authentication
* Login: [*[*mysqladminuser@myserver]
* Password: [*[*mysqladminuserpassword]



**Figure 12 - Change Connection**

3. Click Connect.

4. Change the active database to AdventureWorksDW2008R2 using the Available Databases drop-down list box in the tool bar.

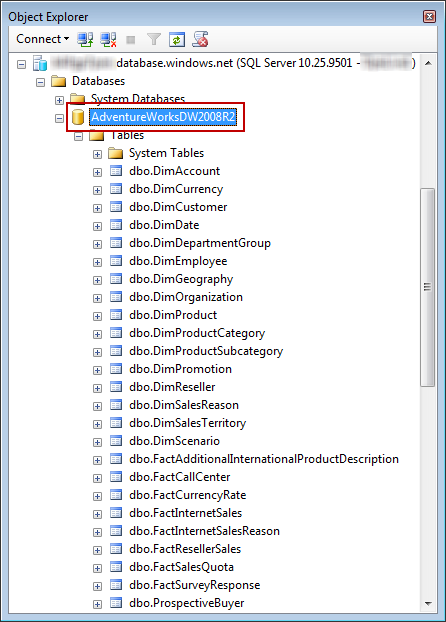
1. 

**Figure 13 - Change Connection**

5. Execute the query by pressing F5 or by selecting the Query > Execute menu option.

6. Review the status information at the bottom of the query window and verify that the query executed successfully.

7. Using Object Explorer, connect to your SQL Azure server with the appropriate credentials. Expand the Databases node, then expand the Tables node inside **AdventureWorksDW2008R2** database. Review the tables that were created and compare them to those in your local SQL Server instance.



**Figure 14 - Review generated tables**

### Task 4 – Create migration assistant stored procedures

In this step you will create two stored procedures in the SQL Azure *AdventureWorksDW2008R2* database that will simplify the process of migrating your data.

The **SetForeignKeyEnabledStatus** stored procedure will be used to disable foreign key constraints during data migration. This will permit you to load your tables in whatever order is convenient without regard to primary key / foreign key relationships. You will run this stored procedure again when data migration is complete to re-enable all of the foreign key constraints.

The **SetIndexEnabledStatus** stored procedure will be used to disable all non-clustered indexes during data migration. This speeds up the data loading process. You will run this stored procedure again when data migration is complete to rebuild all of the non-clustered indexes.

Use the following procedure to create the stored procedures. The source code for the **CreateMigrationAssistantProcedures.sql** Transact-SQL script file referenced in the instructions is included at the end of this section. To execute the script, you will use the sqlcmd utility, which is a useful command-prompt utility for performing batch operations against SQL Server and SQL Azure.

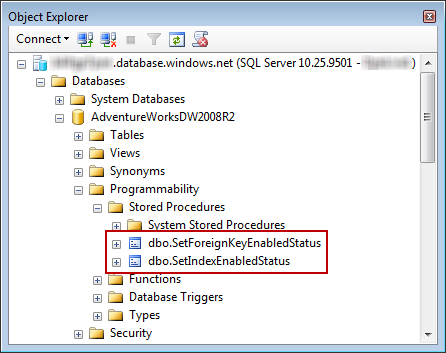
1. Open a command prompt and change the current directory to the *[labPath]\Assets* folder.

2. Execute the following command using the appropriate connection details for your SQL Azure server:

**sqlcmd -S myserver.database.windows.net -U mysqladminuser@myserver -P mysqladminuserpassword -d AdventureWorksDW2008R2 -i CreateMigrationAssistantProcedures.sql –b**

**Note**: The –b parameter ensures that sqlcmd will return a non-zero error code if an error occurs when running a script. This can be used to terminate a batch file if an error occurs.

3. Launch SQL Server Management Studio, connect to your SQL Azure server, and expand the **Databases**> **AdventureWorksDW2008R2** > **Programmability** > **Stored Procedures** node. Verify that the SetForeignKeyEnabledStatus and SetIndexEnabledStatus stored procedures are in the list.



**Figure 15 - Review generated stored procedures**

These two stored procedures iterate through objects (foreign keys and indexes) in system catalog views. For each iteration, these objects dynamically construct a Transact-SQL command to enable or disable the object, and then they execute the dynamically constructed command.

### Task 5 – Disable foreign key constraints

In this step you will use the **SetForeignKeyEnabledStatus** stored procedure created in the previous step to disable all of the foreign key constraints in the **AdventureWorksDW2008R2** SQL Azure database. This will make it easier to load data by removing the need to do it in a specific order to avoid violating foreign key constraints. You will re-enable the foreign key constraints later after you are finished loading your data.

Note that by default, bcp will bypass constraint checking, however it is still useful to do this if you are using other mechanisms to load your data, such as script files or some other application.

1. Open a command prompt.

2. Execute the following command using the appropriate connection details for your SQL Azure server:

**sqlcmd -S myserver.database.windows.net -U mysqladminuser@myserver -P mysqladminuserpassword -d AdventureWorksDW2008R2 -Q "EXECUTE [dbo].[SetForeignKeyEnabledStatus] 0" -b**

**Note**: Use the –Q parameter to send a single command instead of executing a script file.

### Task 6 – Disable non-clustered indexes

In this step you will use the **SetIndexEnabledStatus** stored procedure created previously to disable all of the non-clustered indexes in the AdventureWorksDW2008R2 SQL Azure database. This will improve the performance of data loading by avoiding incremental index rebuilds during the loading process. You will re-build the non-clustered indexes later after you are finished loading your data. Note that you will not disable clustered indexes since the data is loaded in clustered index order, and SQL Azure requires clustered indexes on all tables.

1. Open a command prompt.

2. Execute the following command using the appropriate connection details for your SQL Azure server:

**sqlcmd -S myserver.database.windows.net -U mysqladminuser@myserver -P mysqladminuserpassword -d AdventureWorksDW2008R2 -Q "EXECUTE [dbo].[SetIndexEnabledStatus] 0" -b**

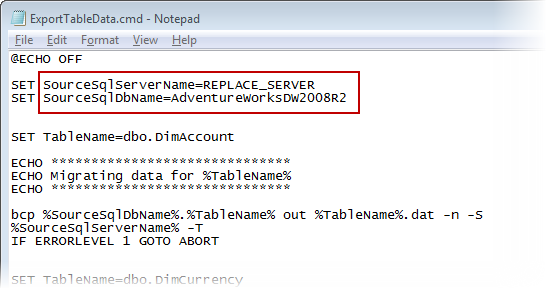
### Task 7 – Use bcp to export table data from your source SQL Server database

In this step you will use the bcp (bulk copy program) command-line utility to export data from your source SQL Server database. When you complete this step, you should have one “native” file for each of the tables.

1. Open the **ExportTableData.cmd** file from the *[labPath]\Assets* folder.

2. Replace the following variables at the beginning of the document with the appropriate values:

* **SourceSqlServerName**: Local SQL Server name
* **SourceSqlDbName**: Local SQL Server database name



**Figure 16 - Export Table Data - Replace script’s variables**

3. Close the **ExportTableData.cmd** file.

4. Open a command prompt and change the current directory to the *[labPath]\Assets* folder.

5. Execute the **ExportTableData.cmd** script.

Internally the script will execute the following command once for each table:

|  |  |
| --- | --- |
| **Command Prompt** | **C:\WindowsAzurePlatformTrainingKit\Labs\MigratingDatabasesToSQLAzure\Lab.html\local\copycode.gifCopy Code** |
| **bcp dbo.MyTableName out MyTableName.dat -n -S MySqlServerName -T** | |

|  |
| --- |
| **C:\WindowsAzurePlatformTrainingKit\Labs\MigratingDatabasesToSQLAzure\Lab.html\local\note.gifNote:** |
| Note: Use the –n parameter to use the “native” file format for export. The native format improves performance on import by avoiding unnecessary conversions. Use the –T parameter to connect to SQL Server using your Windows credentials. |

1. Examine the output and verify that the files were created.

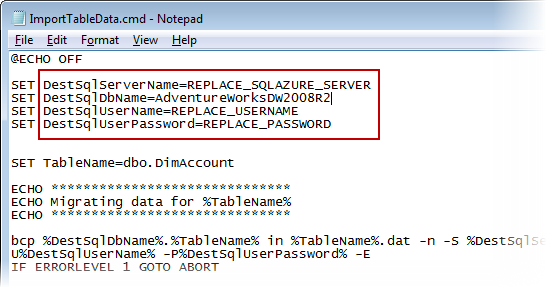
### Task 8 – Use bcp to import table data into your destination SQL Azure database

In this step you will use the bcp command-line utility to import data from a “native” file into your destination SQL Azure data base. When you complete this step, all of your data should be loaded.

1. Open the **ImportTableData.cmd** file from the *[labPath]\Assets* folder.

2. Replace the following variables at the beginning of the document with the appropriate values:

* **DestSqlServerName**: [myserver.database.windows.net]
* **DestSqlDbName**: SQL Azure Server database name
* **DestSqlUserName**: [mysqladminuser@myserver]
* **DestSqlUserPassword**: SQL Azure Account Login password



**Figure 17 - Import Table Data - Replace script’s variables**

3. Close the **ImportTableData.cmd** file.

4. Open a command prompt and change the current directory to the *[labPath]\Assets* folder.

5. Execute the **ImportTableData.cmd** script.

Internally the script will execute the following command once for each table::

**bcp dbo.MyTableName in MyTableName.dat -n -S myserver.database.windows.net -Umysqladminuser@myserver -Pmysqladminuserpassword -E**

|  |
| --- |
| **C:\WindowsAzurePlatformTrainingKit\Labs\MigratingDatabasesToSQLAzure\Lab.html\local\note.gifNote:** |
| Note: Bcp is very picky about parameter formats. Do not use a space after the –U and –P parameters. |

|  |
| --- |
| **C:\WindowsAzurePlatformTrainingKit\Labs\MigratingDatabasesToSQLAzure\Lab.html\local\note.gifNote:** |
| Note: Be careful when importing data into columns that use the [IDENTITY property](http://msdn.microsoft.com/en-us/library/ms191131.aspx). Use the –E parameter to use values in the data file for IDENTITY columns instead of generating new identity values during import. If you do not do this you risk breaking foreign key relationships. |

1. Examine the output to ensure that all data was successfully copied into the destination table.

### Task 9 – Rebuild non-clustered indexes

In this step you will use the **SetIndexEnabledStatus** stored procedure to rebuild the non-clustered indexes you disabled in step 6. It’s best to do this in one shot after loading all of your data.

1. Open a command prompt.

2. Execute the following command using the appropriate connection details for your SQL Azure server:

**sqlcmd -S myserver.database.windows.net -U mysqladminuser@myserver -P mysqladminuserpassword -d AdventureWorksDW2008R2 -Q "EXECUTE [dbo].[SetIndexEnabledStatus] 1" –b**

### Task 10 – Enable foreign key constraints

In this step you will use the **SetForeignKeyEnabledStatus** stored procedure to enable the foreign key constraints you disabled in step 5.

1. Open a command prompt.

2. Execute the following command using the appropriate connection details for your SQL Azure server:

**sqlcmd -S myserver.database.windows.net -U mysqladminuser@myserver -P mysqladminuserpassword -d AdventureWorksDW2008R2 -Q "EXECUTE [dbo].[SetForeignKeyEnabledStatus] 1" -b**