**Group Title**

**Topic N: Deploying SQL Azure Databases**





[**http://msdn.microsoft.com/rampup**](http://msdn.microsoft.com/rampup)

Contents

[Introduction 2](#_Toc245107535)

[Creating a new Database 2](#_Toc245107536)

[Migrating an Existing SQL Database to SQL Azure 3](#_Toc245107537)

[Example: Deployment of the Northwind Sample Database to SQL Azure 3](#_Toc245107538)

[Using SQL Server Management Studio + BCP 3](#_Toc245107539)

[Using the SQL Azure Migration Wizard 4](#_Toc245107540)

[Additional Resources 5](#_Toc245107541)

## Introduction

Microsoft SQL Azure Database is a cloud-based relational database platform built on SQL Server technologies. By using SQL Azure, you can easily provision and deploy relational database solutions to the cloud, and take advantage of a distributed data center that provides enterprise-class availability, scalability, and security with the benefits of built-in data protection and self-healing.

SQL Azure is built on the same technologies used for Microsoft’s SQL Server 2008. Just as an instance of SQL Server installed in your own environment, SQL Azure can be accessed using the tabular data stream protocol (TDS), and exposes a definition and query/command API based upon SQL Server 2008’s Transact-SQL (T-SQL) language. By leveraging these existing protocols and APIs, your own applications can consume SQL Azure as a data layer in the much same way that they currently do with the installable version of SQL Server.

Although SQL Azure is based upon the SQL Server product, management of servers and deployment of databases is slightly different. Unlike a locally-installed SQL Server instance, SQL Azure abstracts away the process of managing server hardware and operations and simply exposes the ability to manage databases and database access. The SQL Azure hosting fabric manages the process of keeping the databases running and accessible and hides the hardware-level details such as hard drives, servers, and network topology.

Because of this abstraction, deploying and administration of databases on the SQL Azure platform is different from that of SQL Server. This article focuses on the details of deploying a database to the SQL Azure platform, but for a more detailed analysis of the differences between the installable version of SQL Server and the SQL Azure service including differences between APIs and features, please refer to the article entitled “SQL Azure Overview” as well as the SQL Azure Developer’s portal.

This article assumes that you have performed the steps necessary to create a SQL Azure account and that you have provisioned a server as well as configured the SQL Azure firewall to grant access to your SQL Azure database server to machines in your development environment. For instructions on how to perform this step as well as screen-shots detailing the user interface, see the section on SQL Azure in the related “Setting up Accounts” article.

## Creating a new Database

Just as with SQL Server, it is possible to use SQL Azure both as a development environment as well as a production environment. In such cases, a developer would create and implement their databases on SQL Azure and use the connection strings supplied by SQL Azure to have their applications run against the databases in the cloud.

To create a new database, you may either use the SQL Azure portal’s Web-based user interface, or you may use the T-SQL interface using a tool such as SQLCMD. To use the Web interface, simply navigate to the server management interface in the SQL Azure portal and click the “Create Database” button. If using a T-SQL interface, simply login using credentials in the sysadmin role and execute a CREATE DATABASE command specifying a MAXSIZE of 1GB for the Web Edition or 10GB for the Business edition.



**Figure 1: The SQL Azure Management Interface**

Once the database has been created, a developer may connect to the database using either the command-line tools or a graphical interface such as SQL Server Management Studio, and execute the T-SQL commands to create tables, views, stored procedures, and logins.

While using an approach such as this to create and develop a database while running on SQL Azure does guarantee that the database is completely compatible with SQL Azure, it must be noted that this is not a recommended approach due to the fact that SQL Azure is a metered service. During development, the process of running tests as well as modifying table structures, writing stored procedures, and tuning indices will consume the pool of query hours allocated to the database server. Therefore, it is recommended to perform as much development as possible in a local instance of SQL Server and periodically deploy the database under development to the cloud for testing and verification.

## Migrating an Existing SQL Database to SQL Azure

As mentioned above, it is possible to create and populate a database entirely in SQL Azure Database by executing T-SQL scripts under a management tool such as the command-line SQLCMD utility but it is expected that in most cases developers or administrators will design and populate the database locally in a development environment. Once the schema and supporting logic has been is completed and tested, the database is then ready to be deployed to the cloud for testing and production.

Another anticipated scenario is that an application developer will want to take an existing application and deploy it to the Azure hosting environment rather than hosting it on their own database servers. In many such scenarios, the developer can take their existing databases and effectively “copy” it to SQL Azure, thereby hosting the database in the cloud alongside their application. However, if the database uses T-SQL constructs or SQL Server features that are incompatible with SQL Azure, the database will need to be modified to meet the capabilities exposed by SQL Azure.

This article will now examine the process of deploying an existing database that has is currently deployed on SQL Server but does not leverage features incompatible with SQL Azure such as Service Broker Queues.

### Example: Deployment of the Northwind Sample Database to SQL Azure

The **Northwind Traders** sample database contains the sales data for a fictitious company called Northwind Traders, which imports and exports specialty foods from around the world. If you would like to perform the migration that we will discuss in this document using your own server and SQL Azure account, you may download this database from the link supplied in the “Additional Resources” section at the end of this document.

NOTE: In order for a database to be compatible with SQL Azure, you will need to ensure that all database tables include a clustered index. In SQL Server Management Studio, this may be accomplished by examining the properties for the tables and ensuring that each table contains a clustered index.

### Using SQL Server Management Studio + BCP

The process for deploying a database from SQL Server to SQL Azure using a management tool such as SQL Server Management Studio is a relatively straightforward two-step process. First, one needs to generate a database script to ensure that the database structure can be created in SQL Azure. Next, the contents of the database need to be copied into the newly-created database. Finally, logins need to be created and granted role membership.



**Figure 2: Generating a database script using SQL Server Management Studio**

To use SQL Server Management Studio to generate the database script, simply right-click the database to display the context menu, select the “Tasks” item and select the “Generate Scripts…” item. This will launch the Generate SQL Server Scripts Wizard, which walks you through the process of creating a script for your SQL Azure database. In order to simplify the process, it is recommended that you choose to generate a script compatible with SQL Server 2008 and to disable options that are incompatible with SQL Azure by setting them to the “FALSE” option, such as “Script USE DATABASE.”



**Figure 3: Setting options for script generation**

After the script has been created, a few changes will need to be made to the script in order to make it compatible with SQL Azure. Since SQL Azure does not support filegroups, any references to filegroups such as “ON PRIMARY” or “TEXTIMAGE\_ON PRIMARY” need to be removed. Similarly, any usages of features such as Full-Text indexing or SQL Service Broker Queues must be removed. For a complete list of features not supported by SQL Azure, see the MSDN topic entitled “Unsupported Transact-SQL Statements.” Once you have performed this step, the database script is ready to be executed against the SQL Azure database using SQL Query Analyzer or SQLCMD. To use SQL Query Analyzer, click the “New Query” button; by default, this will open a query window against your local SQL Server instance. You will need to change the server address by clicking the Query menu, selecting Connection, then clicking “Change Connection…” You may then select your SQL Azure database by setting the server name to the fully-qualified DNS name of your SQL Azure server in the form of “<<servername>>.database.windows.net” and entering a login of that consists of the login name and just the server name portion of the DNS address, such as “<<adminaccount>>@<<servername>>.” In addition, you should also click the “Options” button and ensure that the “Connect to database” option on the “Connection Properties” tab is set to the Northwind database.



**Figure 4: Connecting to SQL Azure**



**Figure 5: Specifying the database name**

Once you have performed the step of connecting to the SQL Azure database, you may then execute the script to construct the database and verify that the script executed successfully.

Copying data using BCP may be accomplished from the command line. To ensure that your data is properly migrated, a set of three steps should be performed: create a format file that specifies the structure of the data being transferred, export the data from the data source into data files, and import the data into the target system from the data files. Below is an example:

C:\Temp>bcp Northwind.dbo.Customers format Customers.dat -S (local) -T -k -w -f Customers.fmt

C:\Temp>bcp Northwind.dbo.Customers out Customers.dat -S (local) -T -k -w -f Customers.fmt

C:\Temp>bcp Northwind.dbo.Customers in Customers.dat -S <<servername>>.database.windows.net -U <<admin>>@<<servername>> -P <<password>> -E -k -w -f Customers.fmt

For more detailed information regarding the usage and syntax of BCP to transfer data between systems, please refer to the link in the “Additional Resources” section entitled “Importing and Exporting Bulk Data by Using the BCP Utility.”

### Using the SQL Azure Migration Wizard

The SQL Azure Migration Wizard is an open-source graphical tool created by Microsoft and hosted on the CodePlex.com Web site. The Migration Wizard simplifies the process described above by extracting the scripts for your database, ensuring that they only use SQL Azure-compatible constructs, and executing the script against your SQL Azure database. It can also optionally copy the contents of your local database to the newly-created SQL Azure database.

## Additional Resources

Additional resources for learning about deploying SQL Azure databases are listed below:

|  |  |
| --- | --- |
| Name | Location |
| SQL Azure Developers Portal | <http://msdn.microsoft.com/en-us/sqlserver/dataservices/default.aspx> |
| SQL Azure Migration Wizard | <http://sqlazuremw.codeplex.com> |
| Northwind Sample Database | <http://www.microsoft.com/downloads/details.aspx?FamilyId=06616212-0356-46A0-8DA2-EEBC53A68034&displaylang=en> |
| Importing and Exporting Bulk Data by Using the BCP Utility | <http://msdn.microsoft.com/en-us/library/aa337544.aspx> |