White Paper

SQL Azure Primer

Phaneendra Babu Subnivis, Manoj Nair

Abstract

Cloud computing is useful to utilize the infrastructure capabilities and the technologies on need basis. The model is similar to the utility based services; pay for what you use. Due to the large subscriber base the cost one incurs to implement the IT applications is far lesser than enterprises pay today for the on-premise infrastructure and technologies. SQL Azure is cloud-based relational database service provided as part of Windows Azure platform. Though built on SQL server platform there are certain differences / changes that one needs to adopt while targeting the SQL Azure platform for development and deployment.

Contents

Introduction

In Information Technology (IT) Industry, first major shift of application development has happened when client – server technology took over from Mainframes. Since past few years, similar level of shift is being observed where in the cloud computing is taking over application development from client – server. Few of the driving factors for cloud computing to become popular are as follows:

- Lower infrastructure investments
- Pay-per-Use model
- Elastic scale to address dynamic resource needs
- Lower Total Cost of Ownership (TCO)

Looking at the way cloud computing taking over the shift in the industry, most of the software giants have started offering their products on cloud. Amazon, Salesforce.com, XCalibre etc. are some of the top software vendors having offerings on cloud computing. One of the major software giant Microsoft has introduced Azure as a platform for application development on cloud. Windows Azure is a product which enables develop and host applications on cloud. From storage perspective, Azure platform offers three types of storage services i.e., Blob, Table and Queues. However, these services do not support relational database features like Transaction processing, Business Intelligence etc. Further, efforts to migrate and/or synchronize data from the on – premise databases to Azure storage services is huge because the underlying principle in which they work is totally different. Hence Microsoft has released SQL Azure as relational database offering on cloud. Key benefit of this is the ease with which customers will be able to work with data across cloud and on – premise SQL Server. This document elaborates on SQL Azure features, explains various tools available for performing activities like development, deployment, migration etc. with merits, de-merits of each of them and recommended tool respectively. It even provides guidelines on which applications to be migrated to SQL Azure helping customers in decision making.

What is SQL Azure?

SQL Azure is a cloud based relational database platform which enables users to host their data on cloud and use it as a service and pay as per the usage. SQL Azure is highly available, scalable and multi-tenant relational database service on the cloud. Since Microsoft takes care of installation of software, patching, and managing servers at platform level, this can be categorized as Platform as a Service (PAAS) form of cloud computing. The database can be hosted and used based on demand without worrying about things like number of licenses, resources like memory/CPU, availability, server maintenance etc. Further, SQL Azure also offers framework around database sync up across database on cloud with the database on-premise version of SQL Server. Because of these features, there is a very high possibility of customers utilizing the SQL Azure as a database platform on cloud and get the best benefits out of it.

Having looked at the features of SQL Azure above, it is also important to know the limitations of SQL Azure. One of the main limitations of SQL Azure is that a database can grow up to 50 giga bytes (GB) of size. Hence while designing the database, one should keep this in mind and take appropriate calls on how to get around this limitation. To get the consolidated list of SQL Azure limitations refer to the link given under point 4 of references section.

> **View Point:**
>
> When compared to setting up on-premise SQL Server environment, the benefits offered by SQL Azure from cost savings perspective and elasticity in database growth are huge. These would be the driving factors to host/migrate databases onto SQL Azure. Further, SQL Azure is evolving very fast as a platform and Microsoft is doing its best to improve on its offering based on customer feedback. For example, it took only few months' time to enhance database ceiling limit from 10 GB to 50 GB. It is even enabling Reporting Services engine to be hosted on SQL Azure.

Why SQL Azure (Benefits of SQL Azure)?

Before delving into more details of SQL Azure, it is a must to understand why one should opt for moving/hosting their applications on SQL Azure and what are the advantages they get by doing so. Following are some of the benefits of SQL Azure:

- Databases hosted on SQL Azure are self-managed

    a. Provisioning and deploying databases on SQL Azure is very easy

    b. No administration overhead. All admin activities are taken care by the service provider

    c. High availability is built-in feature of SQL Azure – three copies of each database are maintained at the data center. Automatic failover is configured accordingly.

    d. Procurement of hardware based on database growth is eliminated

    e. Databases hosted on SQL Azure can be accessed from any part of the world

    f. Database instances will be upgraded to the latest version of SQL Server when new SQL Azure service updates are released

- Scalability of databases hosted on SQL Azure

    a. This is a service based offering. So, payment of the service depends on the size of the database, growth/shrink

    b. Stringent SLAs on database availability

    c. Based on the design, SQL Azure has flexibility of load balancing as well

- Developer's agility

    a. SQL Azure is nothing but an on-premise database engine being enabled on cloud

    b. Leverage existing skills and eco system of developer and management tools

    c. Minimal learning curve for designing applications on SQL Azure

> **View Point:**
>
> Generally while adapting to new platform, the biggest challenge would be to upgrade the skillsets of developers. In case of SQL Azure, since it is the same on-premise SQL Server database engine has been moved on to cloud, the learning curve is very minimal. It is required to understand the feature set that is enabled on cloud and design the applications accordingly.

SQL Azure Architecture

From an architecture perspective, SQL Azure is divided into four layers which work in conjunction to provide relational database functionality to the end users.
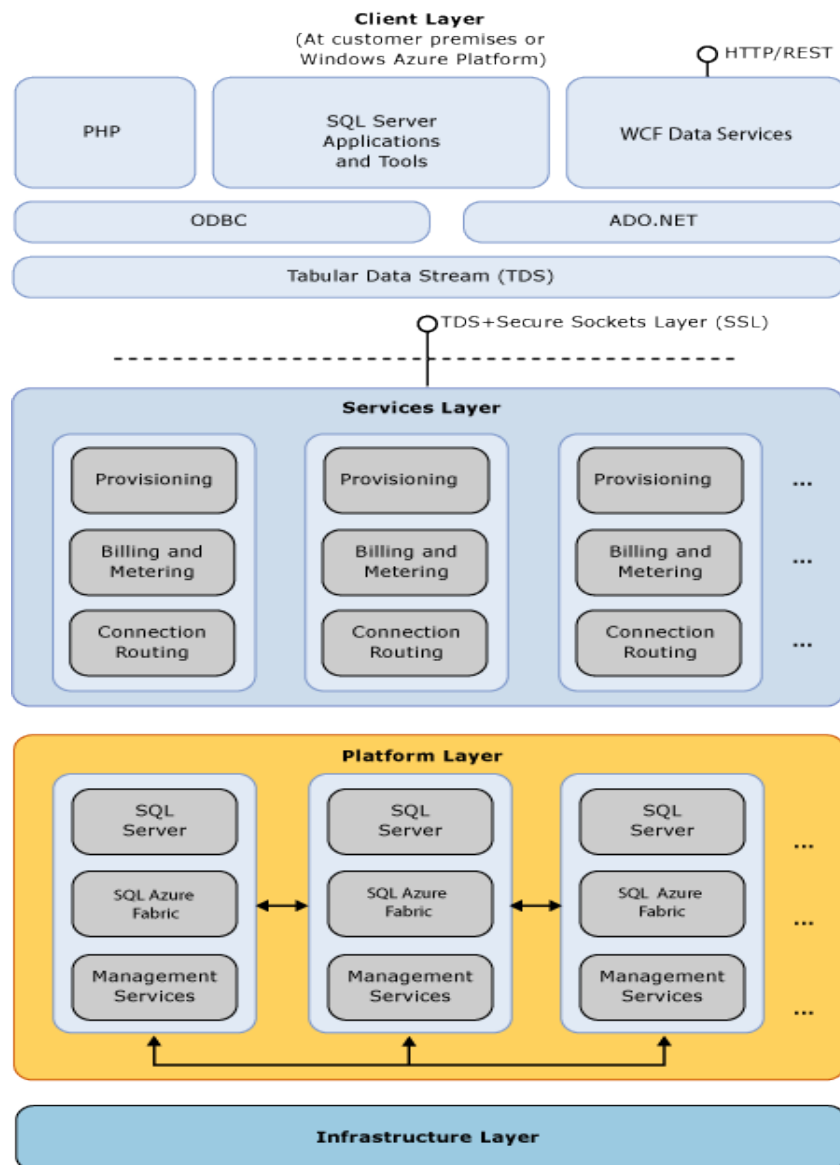
Figure – SQL Azure Architecture – Source: MSDN

The four architecture layers of SQL Azure are as follows

- **Client layer**: This layer acts as an interface for applications to access SQL Azure. This layer can reside on either on-premise or hosted on Windows Azure. Since SQL Azure provides the same Tabular data Stream (TDS) interface as SQL Azure, it can be accessed using ADO.NET and ODBC.

- **Service layer**: This layer acts a gateway between the client layer and the platform layer. It performs connection routing, provisioning and billing/metering. It consists of a group of machines which initially performs validation of the database request, authenticates the user and establishes the connection between the client and requested server on the platform. Once the session is created, the service layer further helps in routing the packets through the established connection.

- **Platform layer**: This layer consists of systems (referred as data nodes) hosting the actual SQL Server databases in the data center. Each SQL Azure database is stored in SQL Server instance on one of the nodes and replicated twice onto instances on other data nodes.

Below diagram shows how the databases are arranged in SQL Azure in the form of three nodes. The client will access the actual databases through the logical server. The SQL Azure Gateway Service is a set of systems which acts as a boundary for the SQL Server instances on the data nodes (Node 1, Node 2 and so on) and is responsible for login validation and routing request to the appropriate physical server. Each SQL Azure database has a primary replica on the SQL Server instance on one node and the two secondary replicas are stored on different servers (in this case Node 2 and Node 3 for the database highlighted in orange color). Also there is a component called the Fabric present in each of the physical servers which takes care of processes like failure detection of replicas, reconfiguration of replicas, location resolution, engine throttling etc.
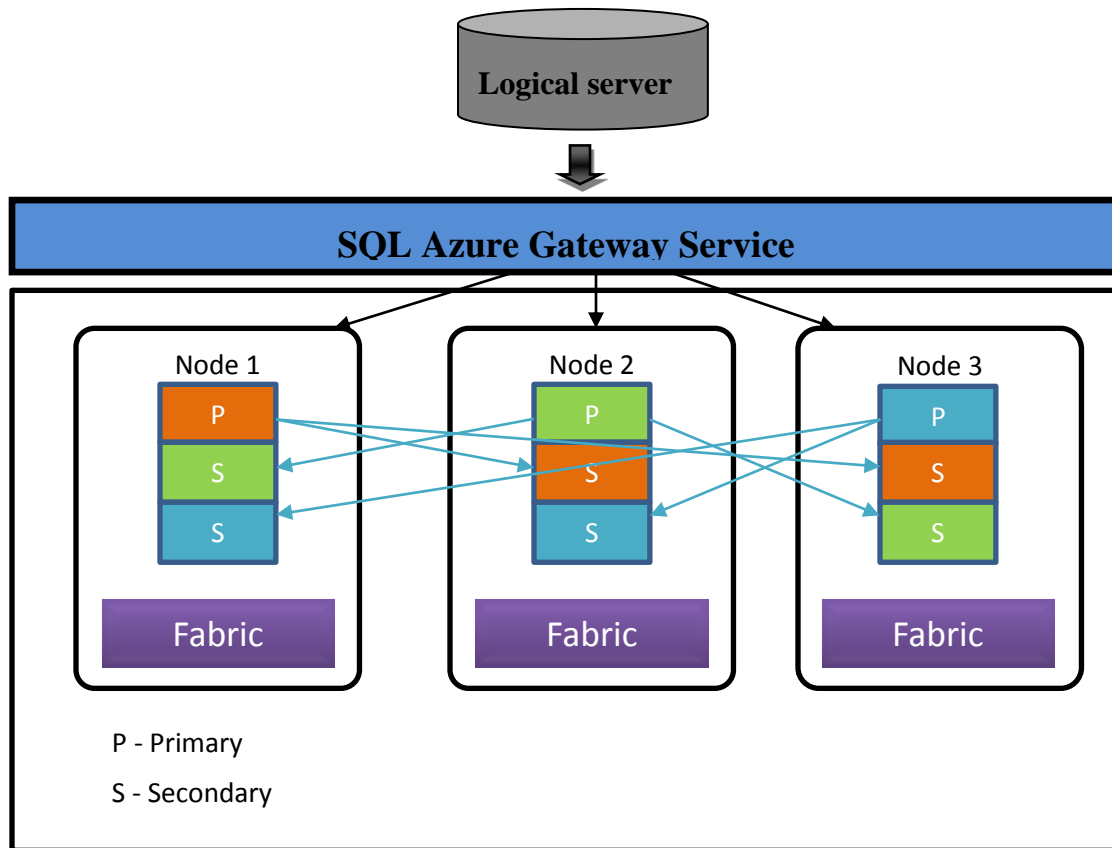
Infrastructure layer: This layer represents the IT administration of the physical hardware and operating systems that support the services layer.

SQL Azure connectivity

Following are the pre – requisites and steps to be performed to connect to SQL Azure:

➢ Register for Windows Azure Platform account using live id

➢ Login to SQL Azure portal from the link http://sql.azure.com using the above live id

➢ Select appropriate project and create the server by providing user name, password and the location. The userid mentioned here acts as an admin user on this server instance.

Screenshot 1 – Creating SQL Azure server/Admin user while logging in for first time

➢ On successful creation of Server on SQL Azure, enable SQL Azure Server firewall settings

Screenshot 2 – Enabling SQL Azure firewall settings

➢ Add rules to provide the client IP addresses from which users connect to SQL Azure server instance. Addition of IP address range is allowed in case of multiple clients.

Screenshot 3 – Adding client IP address/range as firewall rule

➢ In order to access SQL Azure across firewalls, a firewall client is to be installed on the client system which helps in passing through corporate firewalls and connect to SQL Azure.

➢ Configure IP address of the server (Client network administrator can provide this) in the firewall client. Add the same IP address (address of Server) in SQL Azure Server rule since the requests will be routed through server rather than directly from client.

Data modeling for SQL Azure

Designing data model that need to be hosted on SQL Azure is no different than how it is done for on – premise. However, there are certain limitations that need to be considered while designing the data model for SQL Azure. Following table consolidates the list specific to data modeling:

SQL Azure Limitation	Critical (Yes/No)	Comments	Possible work around
Maximum database size = 50 GB	Yes	Application breaks when it crosses 50 GB	During design phase itself, estimate the data load projections Have an appropriate plan upfront to handle data when it is nearing the 50 GB size Scale out options section provides more details
File groups are not supported	No	File groups are mainly meant for distribution of data evenly and improve the performance. SQL Azure by default provides best possible resources for database performance. Hence this seems to be addressed indirectly	
Every table should have a Primary Key	Yes	Heaps are not supported in SQL Azure	Identify best attribute in the entity to be part of the Primary Key Considering a candidate key with data type INT/BIGINT and have that as an IDENTITY column is another option
Following users are not supported for security requirements: 1. admin 2. administrator 3. guest 4. root 5. sa	No	Since these are special users in on-premise databases as well, the possibility of these users existence will be less	If the user names already exists on a database which is on-premise which needs to be moved on to cloud, then those user names need to be recreated with different user name on SQL Azure post migration

## Scaling out databases in SQL Azure

With the restrictions on the database size in SQL Azure, scale out options while hosting databases on SQL Azure becomes critical. Below section provides an overview on scale out methodologies:

Partitioning the database

Partitioning is a feature which helps in better managing tables/databases which grow to an uncontrollable size. It even contributes to improvement in performance depending on implementation. In context of SQL Azure, partitioning can be done at two ways as mentioned below:

Horizontal partitioning (application level)

In horizontal partitioning methodology, the data primarily gets divided into logical chunks. Further application should have capability of hitting right database based on the partitioning key. For example consider an order management application which is accessible from multiple geographies. It means users from different countries can place orders using this application. In this case, Country would be the partition key and the application will hit the SQL Azure database based on the country from where the order is being made.
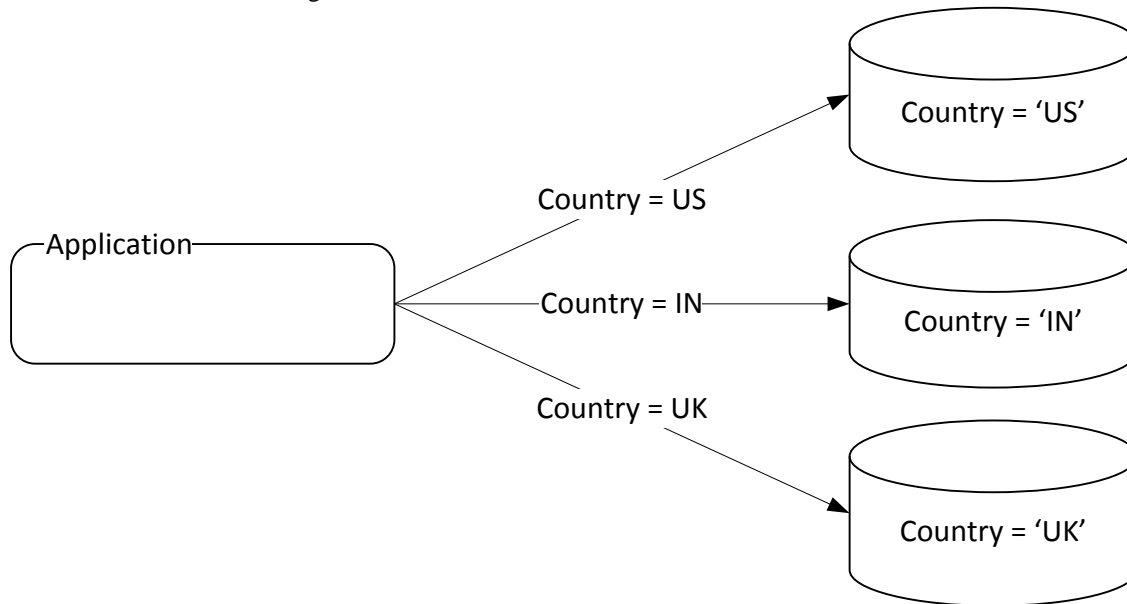
A typical horizontal partitioning refers to a large table being split into a set of partitions based on a partition key. In the above case, it goes further where in the table is partitioned in such a way that each partition is treated as an atomic unit of data and is spread across different server instances. This is called as Sharding.

Advantages:

➢ Workload on the database is spread across server instances where the database is hosted

➢ Since the data is spread out across server instances, it not only contributes to performance of data retrieval but also enables to maintain only the required data

➢ Scaling out by adding another database instance is more easier when new partition gets added i.e., as per this example, customer expanding his business in different country

➢ Hosting SQL Azure database at the data center which is nearby the country would reduce the network latency

➢ No administration overhead while scaling out the database

Disadvantages:

➢ Extra effort to include routing layer in the application

➢ Repartitioning/merging shared databases is very difficult

➢ In order to get best benefits out of it, application deployment may also be necessary along with the database deployment

➢ Replicating/modifying master data across the databases since querying across databases is not allowed.

Recommendation:

➢ Best suits for applications which have clear partitioning criteria as well as less dependency among data across partitions

Vertical partitioning (Object level)

Vertical partition can be defined as splitting a table with many columns into multiple tables with common primary key. While retrieving data, only those tables will be included in JOINs from which data is to be fetched. This needs to be done after completing the data modeling based on fundamental normalization concepts. For example, consider a customer table having Primary Address and Secondary Address Most of the times, only the primary address will be queried. Secondary address is queried only on need basis. But if both of them are stored together in a single table, every time whenever a customer data is queried, the entire data pages gets loaded into memory and thus increases IO. By implementing this vertical partitioning, this can be reduced.

However, this helps only in improving the performance within the database. Scaling out with vertical partitioning could be a rare scenario.

Advantages:

➢ Reduces the network IO by eliminating unnecessary data transfer

Disadvantages:

➢ May not be logical from a typical scale out perspective

SQL Azure Federation

Most of the customers had expressed concern over the upper cap on the databases that were hosted on SQL Azure. Although there are scale-out solutions existing around to address this, it requires changes to be made in the application by including a router layer with capability of hitting the appropriate server based on the key value for data access. In order to address these concerns, Microsoft has introduced a concept called Federation. It would not only provide the seamless scale-out mechanism but also takes care of routing requests to appropriate server based on the federation key. This section provides the basic concepts about SQL Azure federation and explains how the design aspects need to be handled while using SQL Federation with an example.

This design methodology is based on a federation key. A **federation key** is the key item of the table which enables distributing the data among various federations. A **federation** represents data that is partitioned based on the federation key defined on a particular table. So, data associated with a federation key can be called as an **atomic unit**. A range of such atomic units of data is stored in a physical database which is referred as **federation member** or a **Shard**. A separate database which contains the directory information about the distribution of data based on federation key is called as **Root** database. Root database is responsible to redirect the requests to appropriate physical database based on the federation key. The federation members can be distributed across various database server instances with in the data center.

Below diagram provides a logical view of how a federated database looks.
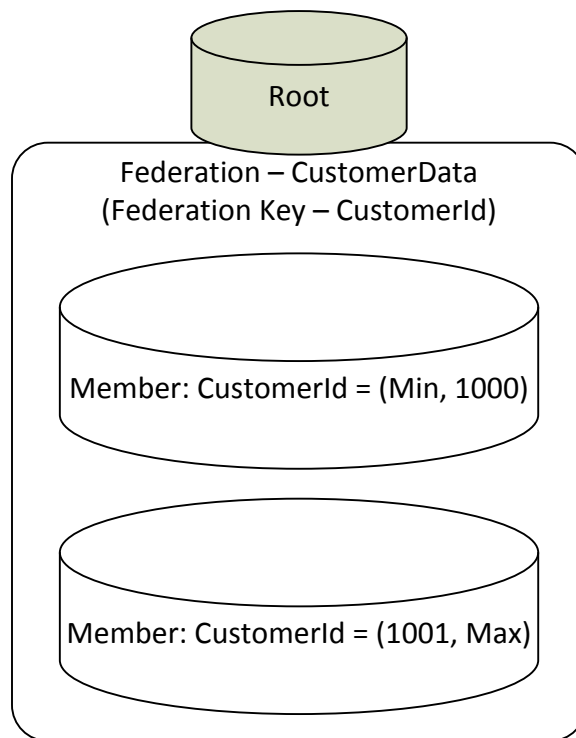
For example, consider an order processing system, where in customer places orders including the products that he needs as line items. In this case, the key entities are customer and product. However, since, the same product id can be ordered by multiple customers, CustomerId becomes the best choice for the federation key. It not only ensures the data distributed across federations but also forms atomic units of data. From the above figure, the first federation member will have data belonging to customer id till 1000 from the minimum value and the second one containing rest of the data. Adding new federation is also relatively an easier task since the CustomerId generation mostly happens in an incremental fashion. The root database contains required directory information which enables it to route the requests to appropriate directory.

Key to the success of SQL Azure Federation methodology is identifying appropriate federation key. A column which ensures the equal data distribution across federations would be the best choice for picking up as a federation key.

Advantages:

➢ Databases can be easily scaled out

➢ Request routing is taken care by federation feature itself.

Note:

The SQL Azure Federation feature is expected to be part of the next Service release.

Disadvantages:

➢ Master/lookup data need to be replicated across all the federation members

➢ Sometimes identifying right federation key could be a challenge

➢ Modifying a federation key for some genuine reason is a REAL pain.

> **Recommendation**:
>
> Looking at the flexibility the federation feature offers, it is recommended to make use of SQL Azure federation feature as a design option for scaling out databases in SQL Azure.

Development tools supporting SQL Azure

Following tools enable users to connect to SQL Azure to perform development activities viz.

- ➢ Management Studio (SSMS) of SQL Server 2008 R2
- ➢ SQL Azure Database Manager (Houston – CTP1)

SQL Server Management Studio

SSMS is a well-known SQL Server database management tool which helps developers/DBAs to develop and manage database objects like tables, stored procedures etc. SQL Server 2008 R2 release has enabled SQL Azure connectivity from SSMS. Users can connect to SQL Azure using SSMS same as they do it with on – premise version of SQL Server. They can connect to both Object Explorer as well as the Query Window from SSMS and perform development activities. Couple of points to notice while working on SQL Azure from SSMS is as follows:

- ➢ Windows authentication is not supported while connecting to SQL Azure
- ➢ While providing the server name in the login window, it should be given in full as shown on the SQL Azure portal i.e., <Server name>.database.windows.net.

Versions earlier to SQL Server 2008 R2 do not support connectivity to SQL Azure. It would be good if Microsoft enables SQL Azure connectivity even from SQL Server 2008 editions.

Advantages:

- ➢ Developers can start working without learning curve
- ➢ Features like Execution Plan can be leveraged to perform basic script/query performance analysis
- ➢ Enables working with database in disconnected mode and later move to SQL Azure

Disadvantage:

- ➢ A minimum of Express edition of SQL Server should be installed on client. Because of this, users may not be able to leverage the feature of connecting to SQL Azure database from anywhere in the world if the client machine do not have SQL Server Express edition.

Limitations:

- ➢ Features like debug/intelli-sense are not supported

SQL Azure Database Manager (Houston – Web based tool)

Before availability of Houston, the only way to connect to SQL Azure was through SQL Server Management Studio (SSMS). For this it was mandatory to install SQL Server, at least an Express edition to connect to SQL Azure which was a concern to some of the customers. In order to enable users to work with SQL Azure without any installation, Microsoft has developed Houston (code name). This is a Silverlight based client which runs on browser helping users to create/modify schema, objects like stored procedures, querying the tables, and manipulate with data. The key intention behind project Houston is to make it as a Database Manager for SQL Azure.

Recently, Microsoft has released a management portal for windows azure platform which enables users to manage Microsoft cloud offerings like SQL Azure, Reporting services, storage accounts etc. The Microsoft Windows Azure Platform management portal can be accessed from the link http://windows.azure.com

Below is the screenshot that appears when the user logs into the above mentioned portal:
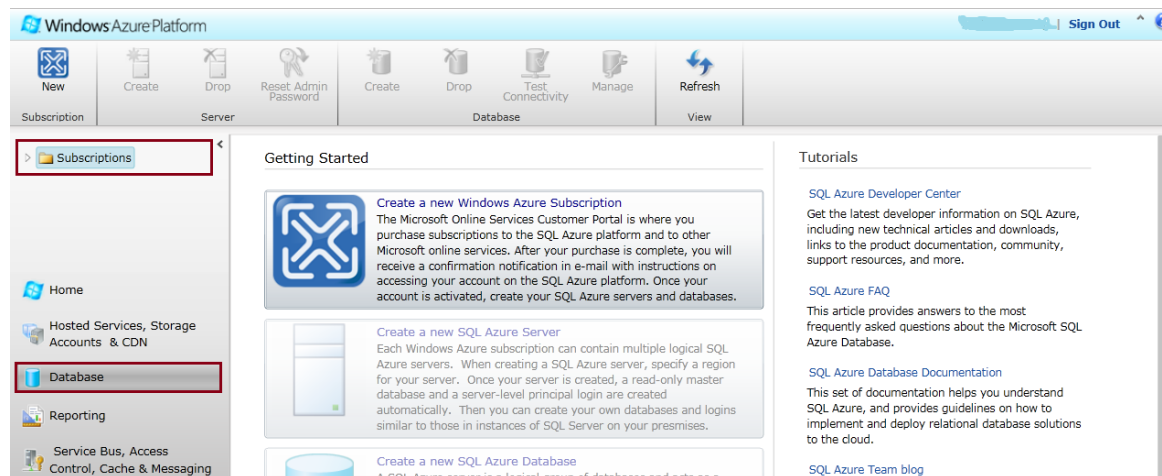
**Screenshot 4 – Windows Azure platform management portal – Home page**

Highlighted are the ones which are of our interest in current context. When the option database is selected, expand the subscriptions and select the server, below screenshot appears:
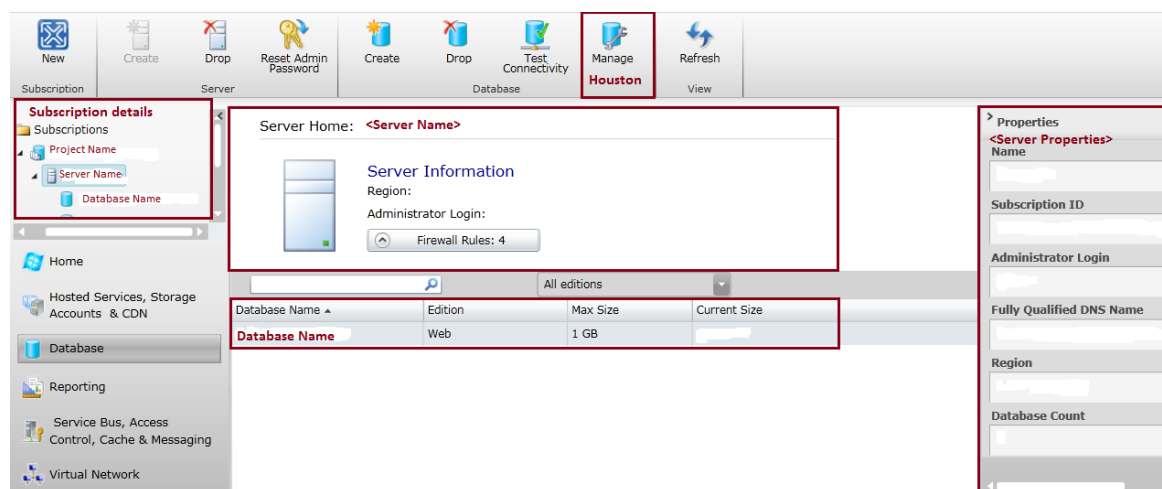
Screenshot 5 – Windows Azure platform management portal – SQL Azure server details

As mentioned by the labels (highlighted in rectangles bordered with maroon color), there are various sections which show the project details, server name and the database names that belong to the login account. It even shows the server information along with the set of databases hosted on the server under the login account. The user has option to filter the database list based on the edition (in the current screenshot "all editions" option is selected). It even has the properties section, which provides additional information like subscription id, administrator login etc., which helps user in understanding server and database relevant details.

In order to launch the Houston portal and start working with the database, click on the option "Manage" (Highlighted on the top menu options in the screenshot 5). On click of that and accepting the licensing terms and conditions and logon to the portal, below screen appears which provides a view similar to object explorer.
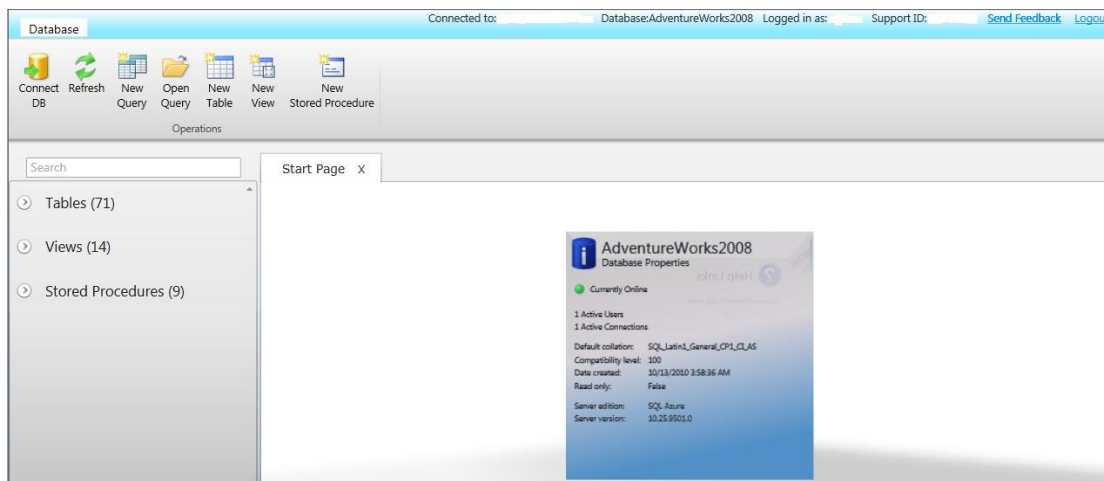
Screenshot 6 – Houston – Home page

The Start Page contains a box which shows the database name and various details like the active users, collation sequence, compatibility level, date of creation etc. This has option to rotate which shows various other options and help links and so on.

On the left hand side there is a view which shows the categorized view of objects available in the database with a grouping of Tables, Views, and stored procedures. On the top left corner there is another ribbon which has various options available for working with the database on SQL Azure which are self – explanatory (as captured in the screenshot below):
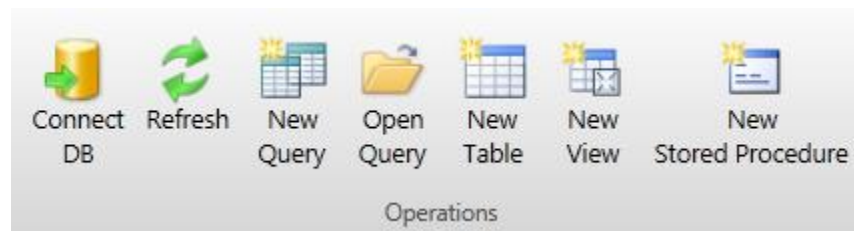
Screenshot 7 – Houston – Different options to perform database operations

Advantages:

➢ No need to install client application in order to connect to SQL Azure database

➢ Enables to connect to SQL Azure database from anywhere in the world

Disadvantages:

Execution plans are not supported in CTP1 of Houston

Current version of Houston is not as intuitive as SSMS

Recommendation:

Use SSMS as the development tool unless one wants to access the SQL Azure from client which doesn't have SQL Server client tools installed.

Points to remember during application development with SQL Azure:

✓ Idle connections to SQL Azure more than 30 minutes will be terminated

✓ When sessions use more than 5GB of TempDB space then the session gets terminated. Transactions executed in large batches, sort operations on large amount of data, rebuilding indexes etc. could lead to TempDB space usage.

✓ Transactions which run for more than 24 hours will be automatically killed.

Deployment tools on SQL Azure

Data Tier Applications (DAC)

Data Tier Application (DAC) Package (PAC) is a new feature introduced in SQL Server 2008 R2. A DAC PAC contains different objects present in the database, packaged and bundled together into a single unit of deployment which can be used for deploying the build across servers. It even manages the sequence in which the objects need to be deployed. Every DAC PAC has a version associated with it. By introducing this feature Microsoft has addressed, the deployment related hick ups DBAs faced while managing the deployment with scripts. The versioning capability of DAC PAC eliminates the manual efforts required to track the delta changes. This section explains DAC PAC features specific to SQL Azure i.e., creation, deployment of DAC PAC on SQL Azure.

The below section address how a DAC PAC can be created and deployed on SQL Server on-premise and SQL Azure. Though the DAC PAC creation is same in both cases, there is slight difference in the deployment processes which will be highlighted in the respective sections.

Creation:

1. **From existing database:** Creation of DAC PAC from existing databases is possible on servers whose SQL Server version is 2005 (Service Pack 4) and above including SQL Azure. Once a DAC PAC is created, it can be deployed on any SQL Server whose SQL Server version is 2008 SP2 and above.

2. **From Visual Studio 2010:** In Visual Studio 2010, new project type has been introduced i.e., Data Tier Project. Once a project is created with this project type, the developer can add scripts under respective folders based on object type. On building the project after completing the development, a DAC PAC gets created.

> **Note:** In the project properties, there is a provision to perform code analysis on the build. There are certain rules which need to be enabled as appropriate and they will be run against the build. The violation of rules will be reported as warnings or an error based on how the rule is configured.

Deployment:

The DAC PAC can be deployed on SQL Server on-premise or SQL Azure. It can be done either by using SSMS or even with Power shell scripts. In the management studio, Object Explorer, under the Management node, there is a node by name "Data-Tier Applications". This node contains all the DAC PACs that are deployed on the server. It has options to deploy new DAC PAC as well as upgrade the existing ones.

> **Note:** SQL Azure doesn't support upgrading a DAC PAC which is already in use. As a work around, either manual scripts need to be deployed or a fresh DAC PAC need to be deployed with a different name, move entire the new database and then rename it to old database name.

Advantages:

- ➢ Using DAC reduces lot of maintenance overhead both from developers as well as the DBA perspective
- ➢ Incremental changes are tracked by versioning DAC PAC.
- ➢ Helps in eliminating manual mistakes while creating scripts
- ➢ Deployment of DAC across servers is easy

Disadvantage:

- ➢ DAC cannot be created when certain features are used in the database. For example, using file groups.

Batch script

A database build generally contains scripts of various objects like tables, constraints, referential integrity, views. It could also contain procedures, functions, triggers which hold the business logic and validations as per the requirements. All these objects will be organized in a folder structure based on object type. Below screenshot shows only folder of one object type "Tables".

Name	Date modified	Type	Size
Deploy	11/24/2010 2:58 PM	File folder	
Tables	11/24/2010 1:33 PM	File folder	

Further, on navigating into the "tables" folder, various table scripts are being listed as shown in the below screenshot:

Name	Date modified	Type	Size
DimCustomer	11/24/2010 1:32 PM	Microsoft SQL Ser...	3 KB
DimProduct	11/24/2010 1:33 PM	Microsoft SQL Ser...	3 KB

Likewise, once scripts are organized based on their object types, a batch file is created which makes use of SQLCMD command to navigate through various folders/scripts available and executes them on the target server and database mentioned while executing the batch file.

Note: SQLCMD is a utility which can be used to connect to SQL Server database and execute batch scripts. As a practice, the batch file is placed with in the "Deploy" folder. The batch scripts written with a capability of redirecting the error/information messages while executing the batch file into a file called "Deploy.Log" which will also be placed under the "Deploy" folder. Batch script takes care of the order in which objects need to be created.

The batch script looks as mentioned in the below screen.

```
@ECHO Off

SET ServerName=%1
SET DatabaseName=%2
SET User=%3
SET Password=%4

SQLCMD -S %ServerName% -U %User% -P %Password% -d %DatabaseName% -i ..\Tables\DimCustomer.SQL >> Deploy.Log
SQLCMD -S %ServerName% -U %User% -P %Password% -d %DatabaseName% -i ..\Tables\DimProduct.SQL >> Deploy.Log
```

In order to execute the batch file, navigate to the appropriate folder and execute the command mentioned below:

DeployBuild.BAT <Server Name> <User Name> <Password> <Database Name>

On completion of execution, the objects get created on target server and errors are logged.

Advantages:

- ➢ This is the best method for maintaining database build after DAC
- ➢ Easy customization with meaningful failure log messaging
- ➢ Versioning can be maintained if required

Disadvantages:

- ➢ Tedious process to follow at the beginning

- ➢ Manual organizing of script dependencies

- ➢ Multiple rounds of unit testing is required for stabilizing the batch file when there are more database objects

Scripting using Management Studio

Using scripting option available out of box in SSMS 2008 onwards, Scripts (SQL Azure compatible) can be generated for both schema and data. Once generated, these scripts can be executed directly on a SQL Azure server. In order to navigate to Advanced Scripting Options, on the database server, highlight the database - > Tasks -> Generate Scripts. Select appropriate objects to be included in the script and then move to the Set Scripting Options.
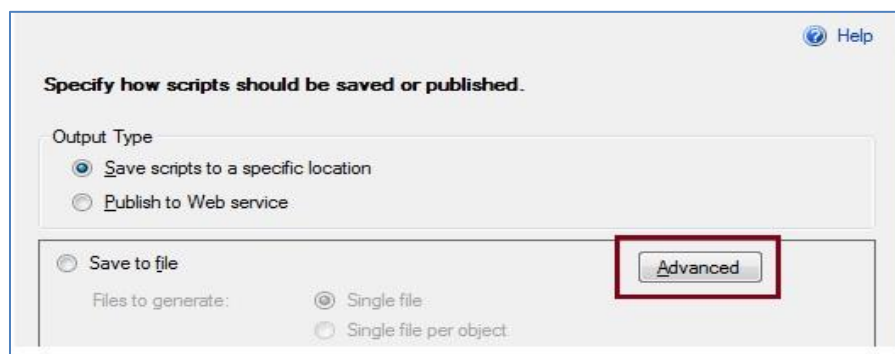
On clicking the "Advanced" button, below screenshot appears –
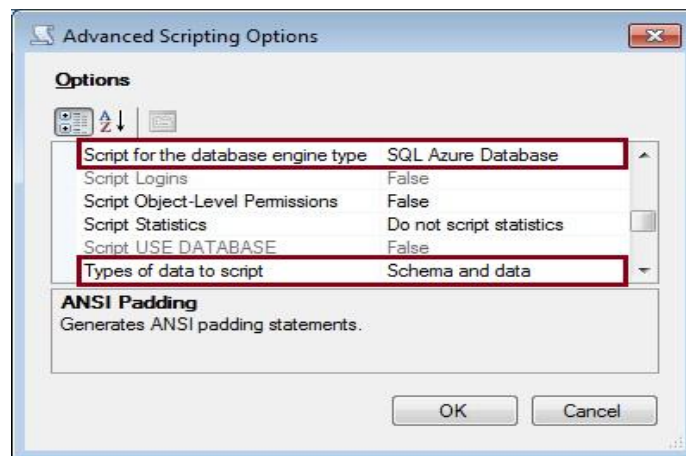
Highlighted properties enable users to provide the target database engine type and schema/data option for scripting.

Note: If any object of the database contains special data types like Geometry, are not supported by the scripting wizard (even though the destination database engine supports it). It throws error and stops the script generation.

Advantages:

- ➢ Script generation is possible based on target database engine type

➢ Enables creation of scripts for moving objects and data as well

➢ Easy to move lookup data in this methodology

Disadvantages:

➢ This method is usable only when a database exists on the SQL Server (or SQL Azure)instance Maintenance is tedious job

➢ Data movement is a series of insert statements. So, when look up data is huge, then this is not preferred method of data movement.

Tip: When the data to be moved into script format is bigger in number (in thousands), it is recommended to go with Batch script method described below, and use bcp utility for data movement. This is relatively faster and recommended while moving data across databases.

However, below table shows preferred option for database deployment among batch file and scripting option with SSMS:

	Preferred method	
Database Build Scenario	Batch file	Script generation using SSMS
New database creation	✓	✗
Existing database	✗	✓
Version logic required	✓	✗
Deployment log	✓	✗

Note: Script generation from SSMS would be useful only when a build need to be created from an existing database and deploy the same on new database/environment. Otherwise, both from creation/maintenance perspective, after DAC, Batch file methodology are preferred because of the customization that can be done for logging/execution of files.

Below table shows the preferred option including the DAC PAC choice as well:

		Preferred method	
Database Build Scenario	**DAC PAC**	**Batch file**	**Script generation using SSMS**
New database creation	✓	✓	✗
Existing database	✓	✗	✓
Version logic required	✓	✓	✗
Deployment log	✓	✓	✗

Recommendation:

Given these 3 choices available for us to create a deployment package, then DAC is preferred over the batch scripts and scripting option available with SSMS.

Migrating applications to SQL Azure

This section explains different methodologies to migrate the schema and data onto SQL Azure. There are two categories of databases to be migrated onto SQL Azure i.e.

1. Migrating on-premise SQL Server databases to SQL Azure
2. Migrating cross platform database like DB2 to SQL Azure

Migrating on-premise SQL Server database to SQL Azure

Prior to SQL Azure, the only option available for storing data on the cloud was using Windows Azure Table Storage which is a non-relational database. As a result, migrating data from on-premise to Table Storage was cumbersome. With the advent of SQL Azure which is a relational database, data migration is expected to be relatively easier. Following sections explain about various methodologies to migrate data in detail and ends the section with comparative analysis from performance perspective along scenario based recommendations.

BCP Utility

This is a one of the commonly used mechanism for moving data across databases. It moves data really fast and comes handy while moving tables with millions of records. Pre-requisite to this is to have same schema structure at both source and destination including the data types. Any variation in the structure would create an overhead of using format file while moving data using BCP utility. This methodology typically involves a two-step process i.e. loading source data into flat file structure which is called as "bcp out" operation. Use these flat files as input for moving data into destination table structures which is referred as "bcp in" operation.

In order to create table structures, it is recommended to adopt method of script generation offered out of box by SSMS as explained in section in "Deployment Methodologies".

Advantages:

- ➢ Very fast in moving data irrespective of the table size
- ➢ Best suited when table structures remain same and tables having millions of records

Disadvantages:

- ➢ Error handling is relatively tricky.
- ➢ Identifying exact failure records while loading data is a challenge
- ➢ Any changes to table structures would require extra effort to work with format files.

SQL Azure Migration Wizard (SAMW)

SQL Azure Migration Wizard is a Windows application contributed by community whose downloadable is available at codeplex website. This is used to "analyze" and "migrate" on-premise SQL Server database to SQL Azure. The "analyze" part takes care of the features that are not supported in SQL Azure like table without clustered indexes, replication, XML schema etc. and provides alternate scripts or comments so that the user can take suitable actions. The "migrate" part takes care of creating appropriate scripts for schema and flat files for data which subsequently used for moving data to SQL Azure. Both these steps are optional and the user can choose any of them at a particular point.
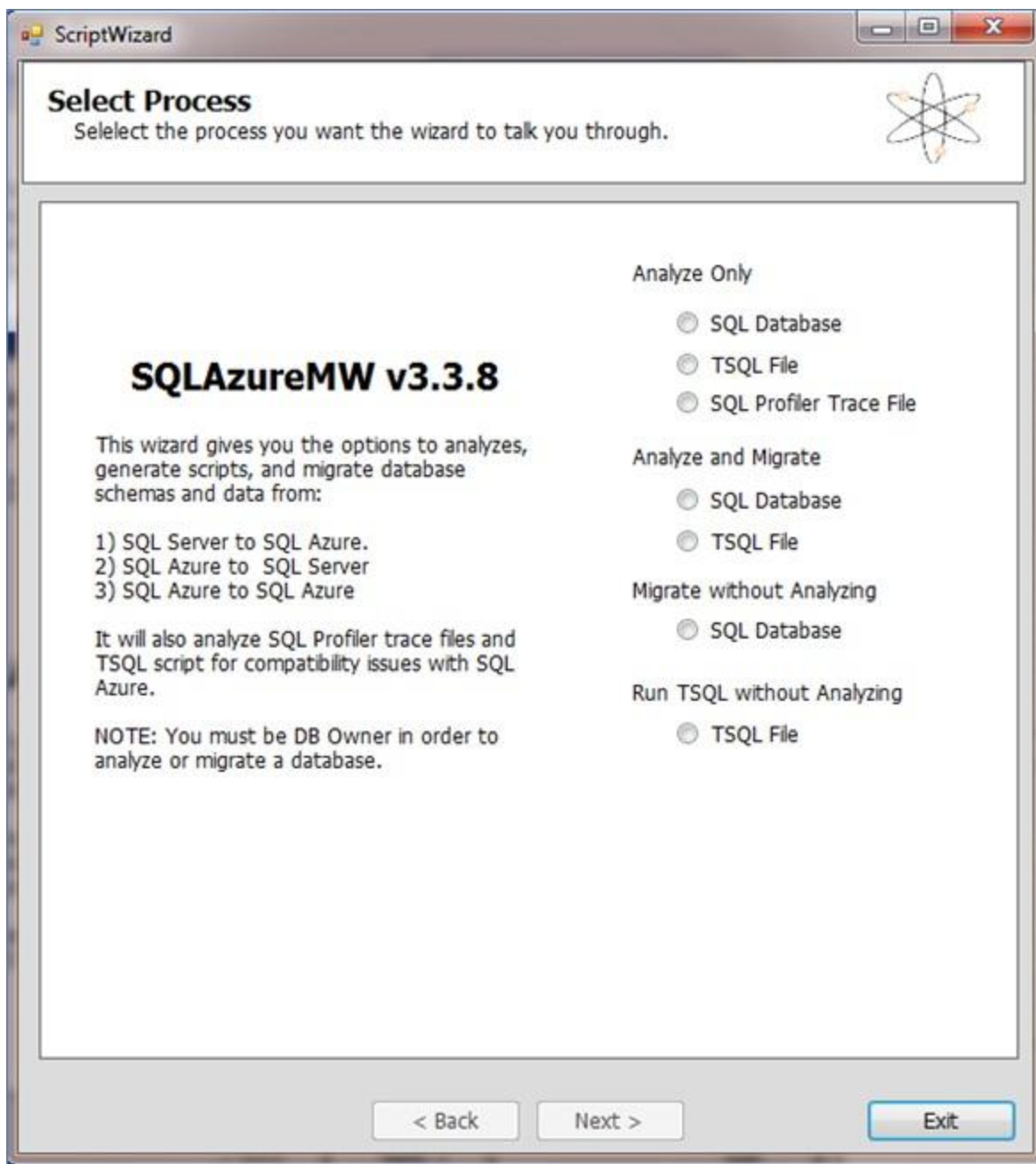
There are options to "analyze and/or migrate" data from a SQL Server database or a TSQL file. Additionally SAMW can analyze SQL Profiler Trace files as well to check for compatibility issues. Following screenshot shows migration of the table 'Department' from the Adventure Works database as an example. Select SQL Database option under Analyze and Migrate section and then select the database from which the table to be migrated in this case AdventureWorks2008 database.
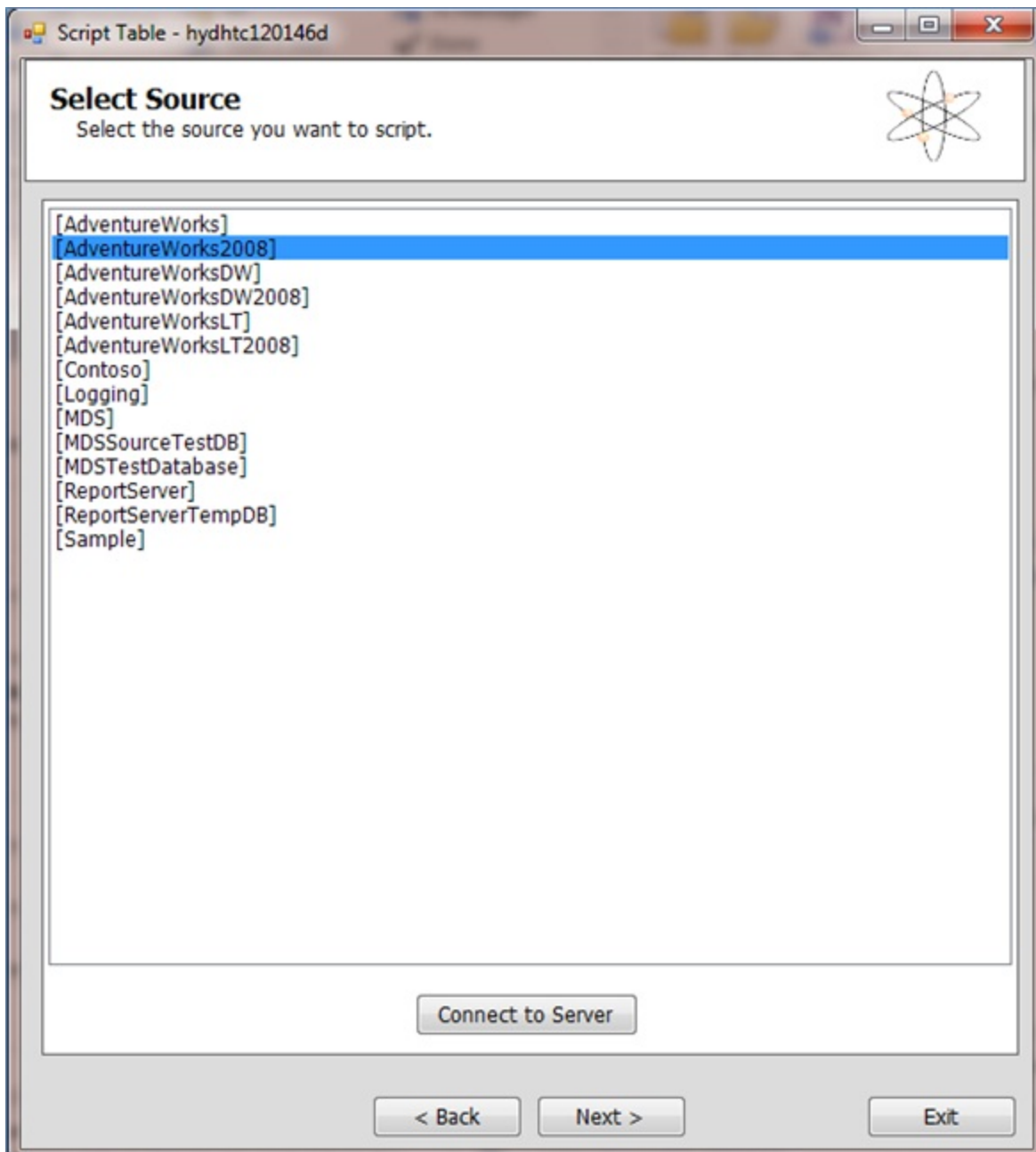
Select the Department table from the list of database objects
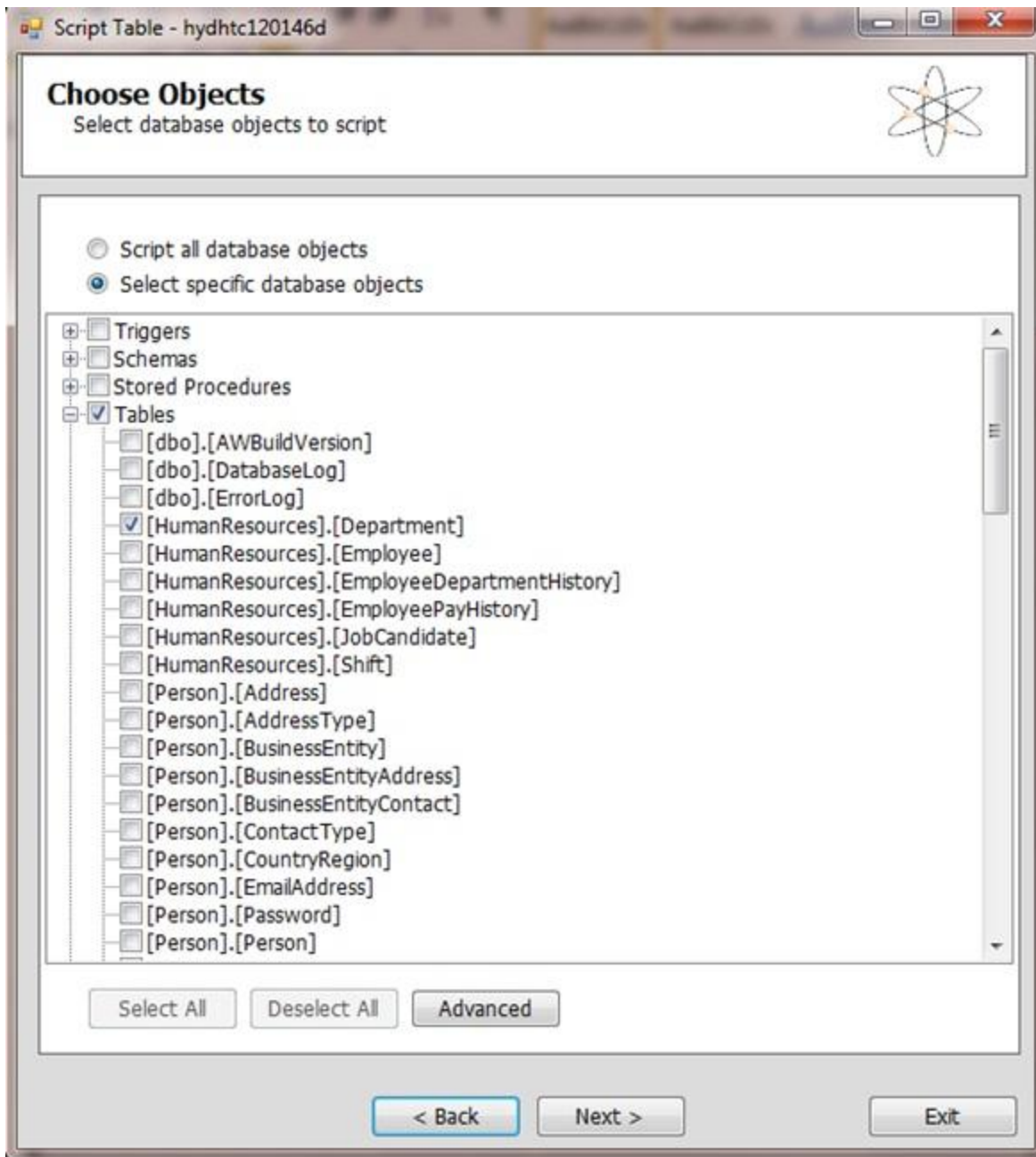
Once the required tables are selected, the scripts will be generated. The executions summary and the scripts will be displayed as shown below. The outputs can be saved for future references. As you can notice the values in the table is saved in the form of a .dat file which can be used by BCP utility to migrate data to SQL Azure.
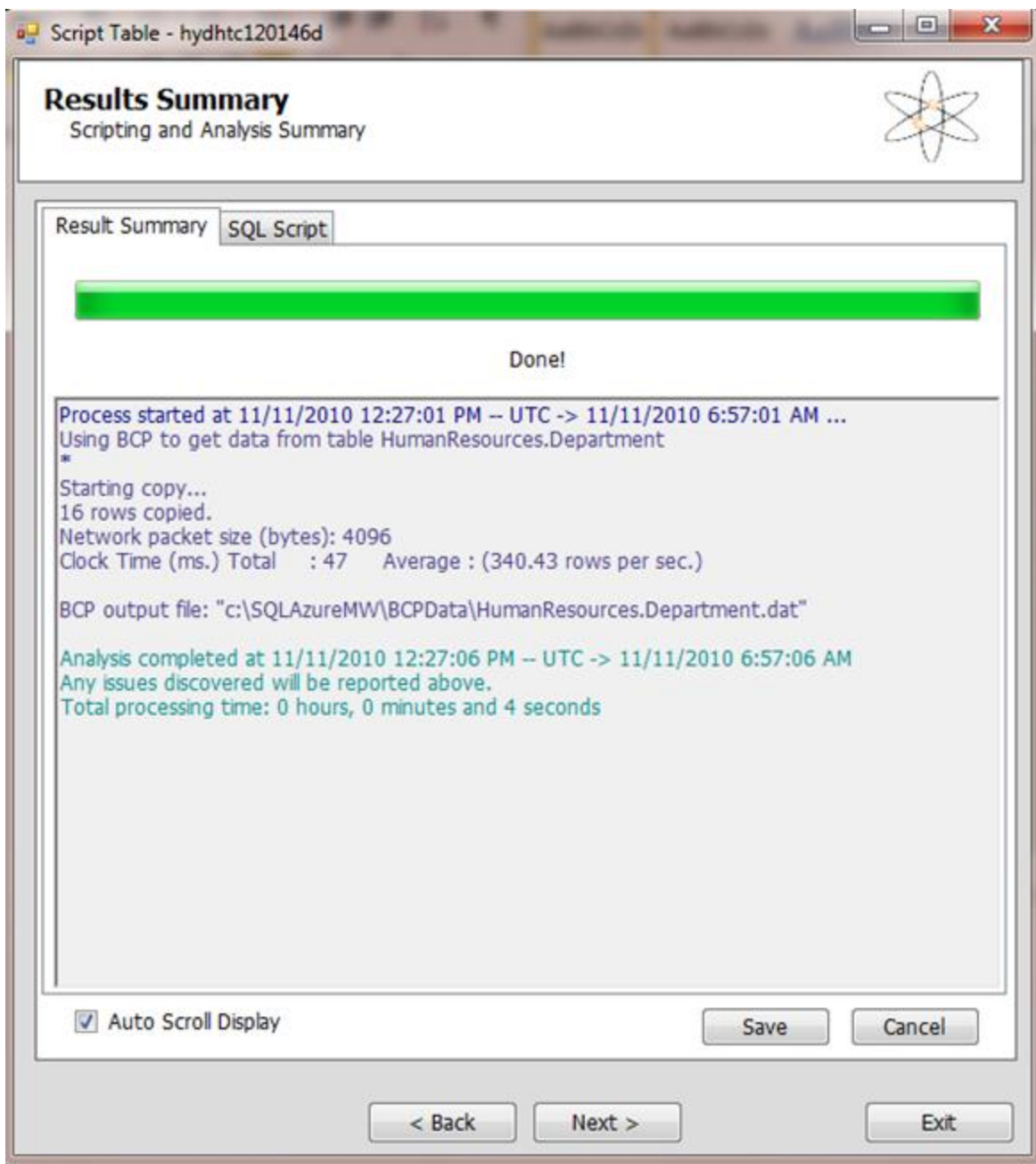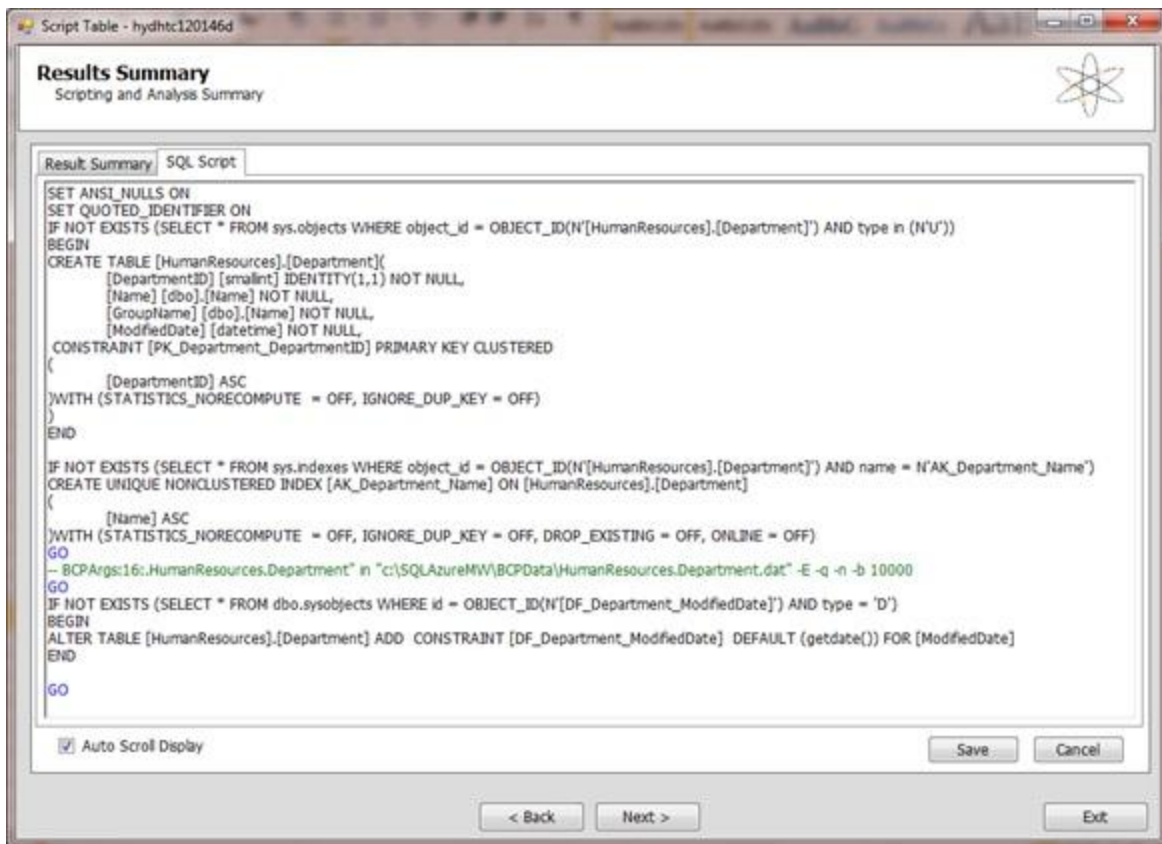
The above screenshot shows a snapshot of the schema scripts that get generated using the Migration Wizard. On proceeding further with the tool, you can connect to the target server (SQL Azure) and execute the scripts for schema generation as well as data migration on the required database. The user also has the 'Save' option to save the script files and the data files to be executed at a later point of time.

Advantages:

➢ The migration wizard takes care of data migration as well. It performs analysis on the code/schema and provides list of issues that come across during migration

Disadvantages:

➢ Structural changes to the target database cannot be handled with this tool

➢ Tracking issue report may be tedious when the migration activity is relatively bigger

Note: Migration Wizard tool is a contribution from user community and the downloadable is available at codeplex.com. This is not officially supported by Microsoft. However, the code can be downloaded and customized to meet our requirements.

Using SSIS packages

SQL Server Integration Services (SSIS) is a well-established ETL tool. Using SSIS packages, data can be migrated across on-premise and SQL Azure.

In order to achieve this using SSIS package, two connection managers and a data flow task are required. The connection managers would be used to connect to source and destination datbases and the data flow task would handle the transformational logic while moving the data. Below screenshot shows basic SSIS package for moving data from an on-premise SQL Server table to SQL Azure.
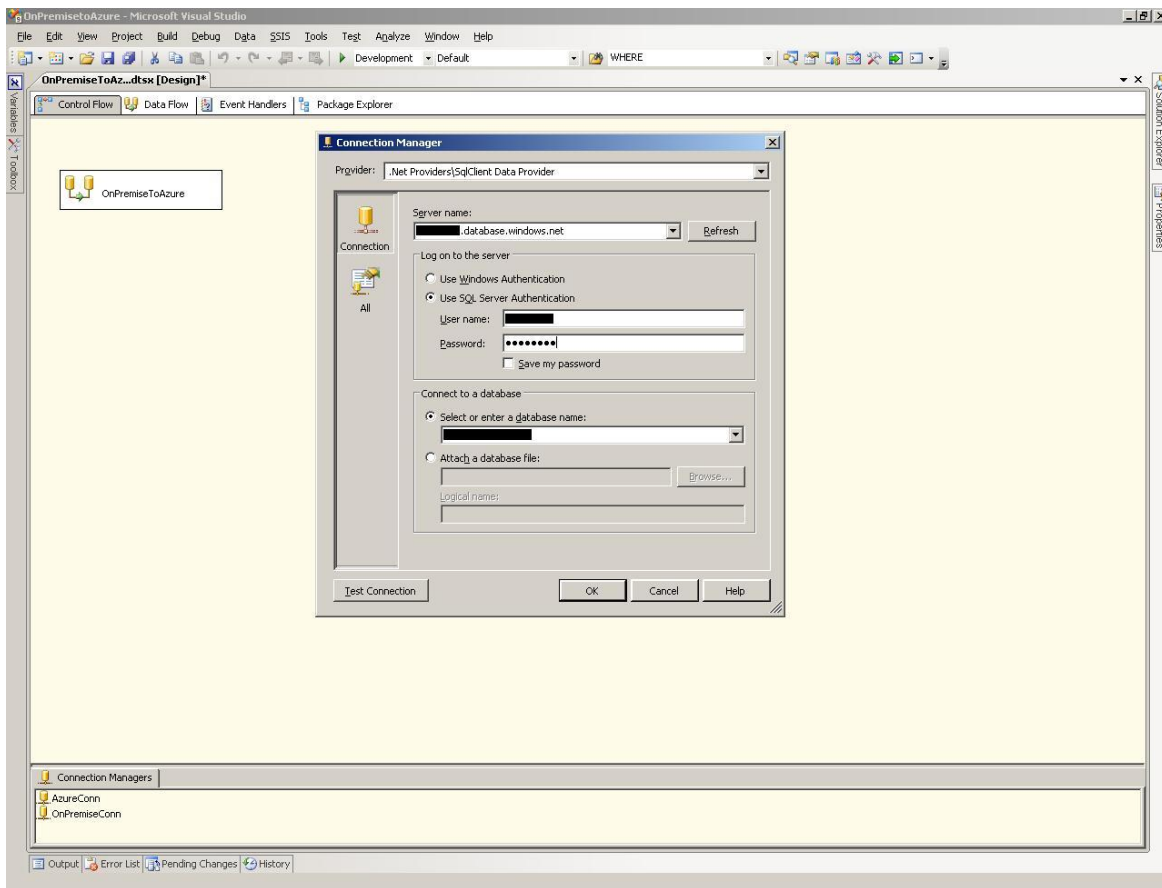
Create two connection managers which connect to on-premise SQL Server and SQL Azure. Note that SQL Azure connectivity is possible only by using .net provider (as shown in the above screenshot). Further, define a data flow task which includes the relevant transformational logic for transforming data from source to destination.

Below screenshot depicts the source and target table fields and mapping across them.
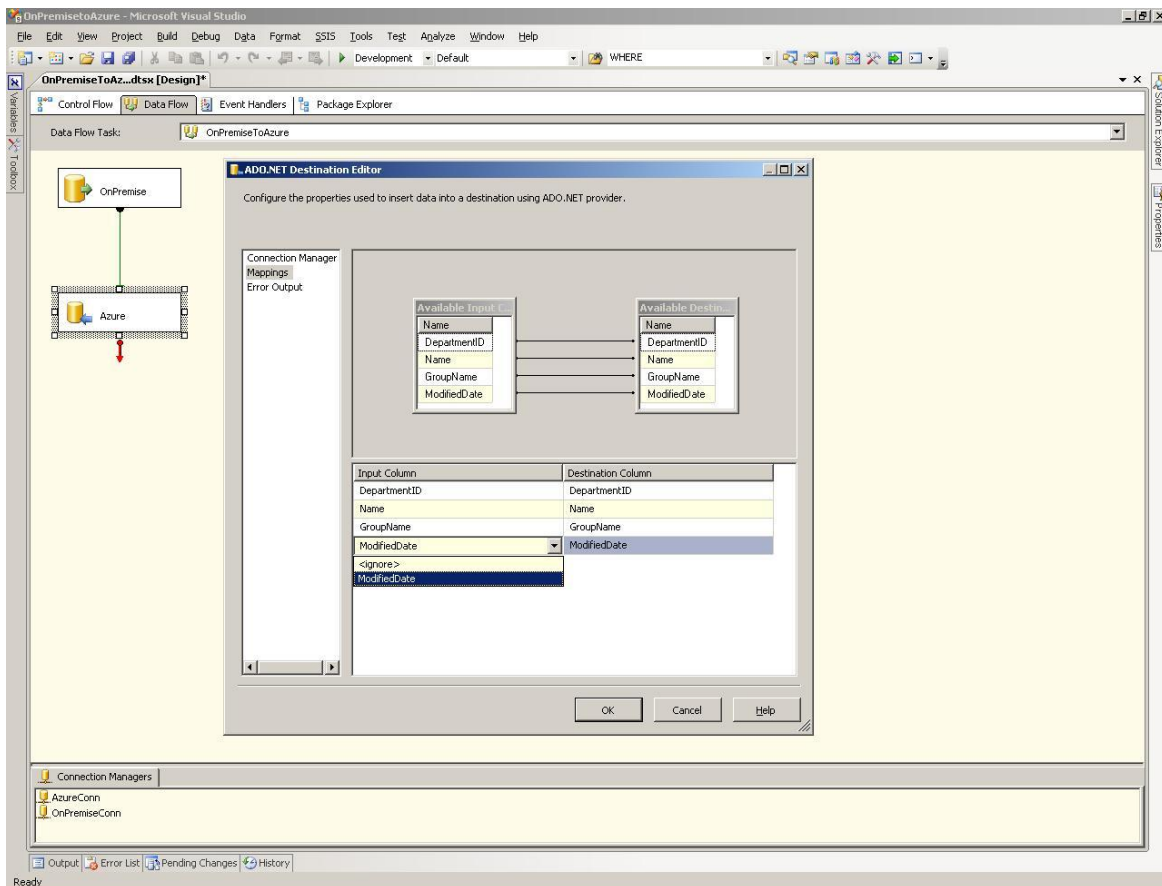
To simplify the sample package, transformations have not been included while moving data. However, while using SSIS package for data movement, entire feature set of SSIS can be used for building the data transformational logic.

Recommendation: This methodology is recommended when scenarios like on-premise OLTP database have to be converted to a warehouse/reporting database since it involves de-normalization of data as well as transformations.

Advantages:

➢ SSIS package is the best option while migrating data across on-premise and SQL Azure database if transformational logic is involved.

➢ Failure record handling will be relatively easy

➢ Packages can be tuned for moving even large amount of data

Note: Schema should already have been created while using SSIS package for data movement. Since SSIS is an ETL tool and is expected to be used only for data movement, even though it is possible to create schema with EXECUTESQL Task, it is advised to keep schema creation independent of data movement.

In order to perform comparative analysis of the above mentioned methodologies for moving data across on-premise SQL Server to SQL Azure, we have performed a sample test. For the testing purpose, we used FactSales table of Contoso database which has 3406089 records. We have used normal 32 bit desktop for carrying out the test. Following table shows results of the testing:

Methodology	Duration for data migration (In Mins)
BCP	39
Migration Wizard	38
SSIS Package	83

From the above results it is pretty evident that bcp is faster for moving data from SQL Server on-premise to SQL Azure. However, in general, speed alone will not be criteria. Especially with the usability factors of SQL Azure, there could be certain changes to schema/data transformations when on-premise data as a whole is not moved on to SQL Azure. In such cases, it is worth looking at SSIS packages for transferring data across SQL Server to SQL Azure.

Note: These results are only indicative. There could be lot of factors like CPU, memory, IO etc will play key role in performance. Since the test involves, data migration to cloud, internet bandwidth would be the most critical in speeding up/slowing down the data movement.

Recommendation: Based on our experience on migration, it is recommended to perform initial analysis, address the possible issues that could come during migration of schema/data and then proceed for migration. Hence it is recommended to make use of migration wizard to perform analysis and then deploy the scripts on SQL Azure. Subsequently depending on the amount of transformations involved, choose either bcp or SSIS for transferring data.

Migrating cross platform database to SQL Azure

Recent release i.e., version 4.2 of SQL Server Migration Assistant (SSMA) supports migrating schema and data of Access database to SQL Azure. Similarly SSMA 2008 for MySQL version 1.0 supports migrating MySQL schema and data to SQL Azure.

View point:

✓ Looking at the current set of offerings at SQL Azure and the legal requirements (wherever applicable like banking firms may have legal requirements to store data within organization premises), it would be better to have on-premise SQL Server holding most of the transactions part and build warehouse on cloud for reporting purposes. Enabling reporting services on cloud soon and Microsoft plans around enabling Analysis Services would make this job more easier

✓ Migration of database from on-premise SQL Server to SQL Azure is a two-step process i.e., migrating schema and then the data. Migrating schema is preferred to be done by executing scripts (as explained in section – Scripting using SSMS). However, for data migration, use BCP mechanism to migrate the data at a faster pace when there are not many transformations are available. In case if there are transformations, SSIS is the better option since it has capabilities of handling transformations as well as data exceptions. While deciding to go with this option, be aware of the fact that there would be certain extra amount of time spent for data migration.

Debugging and tuning techniques

SQL Azure is a multitenant service offering on cloud provided by Microsoft. Hence the control on server hosting the user database is very limited. This further limits usage of some of the common debugging tools used on – premise like SQL Server Profiler, Database Tuning Advisor (DTA). Hence, based on the availability of on – premise SQL Server, developers are advised to build and tune code initially on the SQL Server on – premise and then move on to SQL Azure. In SQL Azure, developers can make use of the following tools/methods for tuning the code:

- Execution plan – This is a cost based plan picked up by SQL database engine to execute the query. It enables developers understand cost for each operation performed while executing the query to retrieve the data. It even helps to understand the CPU, IO utilization during the query execution. By looking at the execution plan, the developers can easily identify the costliest operation from the execution plan and figure out ways to get the query tuned to reduce the cost to extent possible while tuning the query.

- Dynamic management views (DMV) – These views provide an insight of various aspects of resource utilization of SQL Server. There is an exhaustive list of DMVs available in on – premise SQL Server, but because of certain limitations in SQL Azure, only following list of DMVs are made available in SQL Azure which helps to get an understanding on performance of the scripts/objects/queries.

    - Database-related dynamic management views:

        o sys.dm_db_partition_stats

    - Execution-related dynamic management views and functions:

        o sys.dm_exec_connections

        o sys.dm_exec_query_plan

        o sys.dm_exec_query_stats

        o sys.dm_exec_requests

        o sys.dm_exec_sessions

        o sys.dm_exec_sql_text

        o sys.dm_exec_text_query_plan

    o Transaction-related dynamic management views:

        o sys.dm_tran_active_transactions

        o sys.dm_tran_database_transactions

        o sys.dm_tran_locks

        o sys.dm_tran_session_transactions

In many forums/ events Microsoft has emphasized on its investments towards getting the debugging/troubleshooting tools like SQL Server profiler etc., to help developers in writing quality code.

Backing up and Restoring SQL Azure databases

SQL Azure is built on the Windows Server and SQL Server technologies and hence is capable of handling any kind of failover problems due to hardware. SQL Azure usually replicates data (one primary replica and two secondary replicas) onto multiple physical servers and keeps them in sync in order to provide high availability. The service provides automatic failover to maintain business continuity. When a hardware problem occurs on the primary replica, SQL Azure detects it and falls over to the secondary replica.

Failures or data loss caused due to user errors are to be taken care by developers while building the code itself. At the database level, SQL Azure provides the option to create a full copy of an existing operational database on the same datacenter. This operation is asynchronous i.e. the statement will return a success message even though the actual copy process is still in progress. The statement for creating the copy will look like

```
CREATE DATABASE destination_database_name AS COPY OF
[source_server_name.]source_database_name
```

This statement has to be executed on the master database of the destination SQL Azure server. Below is an example which creates a copy of the SampleDACPAC database on the same SQL Azure server

```
CREATE DATABASE SAMPLEDACPAC_BK
AS COPY OF SampleDACPAC
```

The process can be monitored by querying the `sys.dm_database_copies` or the `sys.databases` views which will show state of the copy process.
```
select * from sys.dm_database_copies
```

	database_id	start_date	modify_date	percent_complete	error_code	error_desc	error_severity	error_state
1	5	2011-01-11 04:58:37.7400000 +00:00	2011-01-11 04:58:37.7400000 +00:00	NULL	NULL	NULL	NULL	NULL

> **Note:** These copies of a particular database can be created only on a SQL Azure server. For creating an offline backup, we need to make use of SQL Azure Data Sync.

For offline backups, users can make use of SQL Azure Data Sync tool. It is a cloud-based data synchronization service built on Microsoft Sync Framework technologies. Using this tool, the SQL Azure database can be synchronized with one or more on-premise databases. Initially after the complete data is moved, only subsequent changes are moved to the database (similar to a differential backup). In this way, users can maintain a local copy of their cloud databases and use them when any disruption happens on the cloud.

Though this seems to be a good solution there are some drawbacks involved

➤ Backing up databases on on-premise servers defeats the very purpose of using SQL Azure. Additional setup need to be done locally to take care of this scenario.

➤ If there is some corrupt data in the SQL Azure database, it will get replicated on the on-premise database during synchronization rendering it unusable.

> **View point:** Though this approach of creating offline backups seems feasible, there are some drawbacks involved
>
> ✓ Backing up databases on on-premise servers defeats the very purpose of using SQL Azure. Additional setup need to be done locally to take care of this scenario.

SQL Azure Security

Users can connect to SQL Azure only by using SQL authentication. Windows authentication is not supported. From security perspective, in order to connect to SQL Azure, the user has to pass through the SQL Azure firewall. Users cannot connect to SQL Azure bypassing the fire wall.
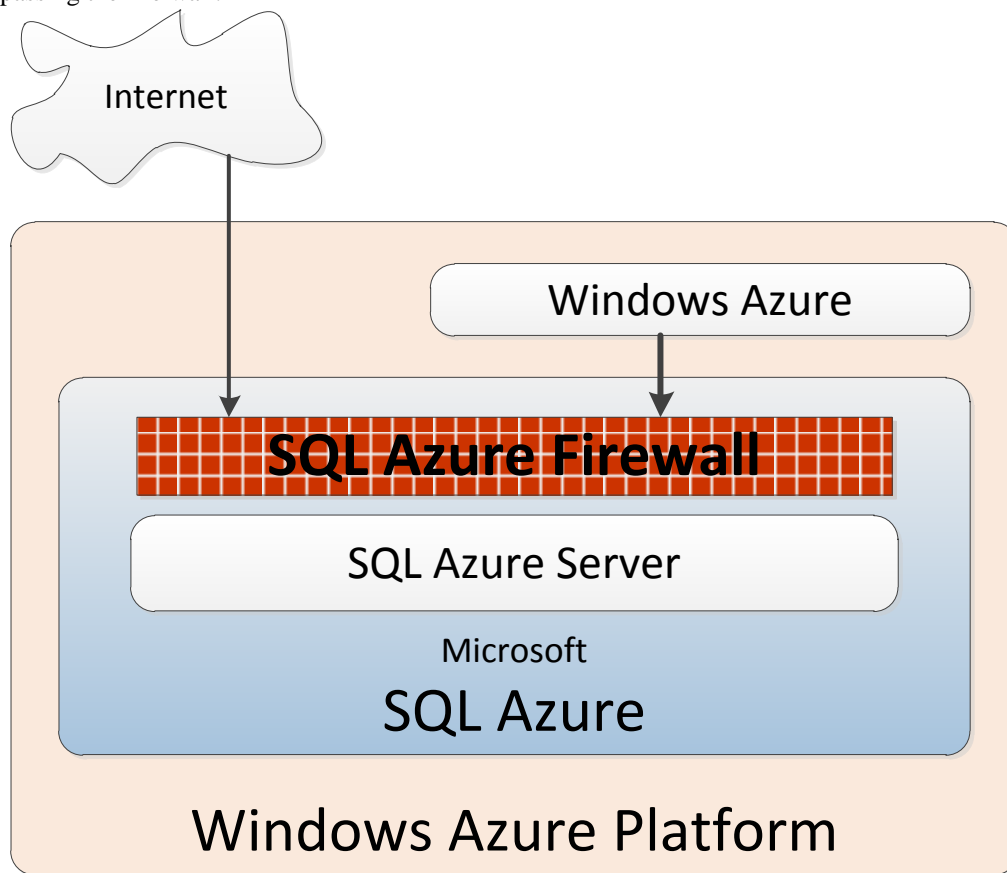
Figure – SQL Azure Security architecture – Source: MSDN
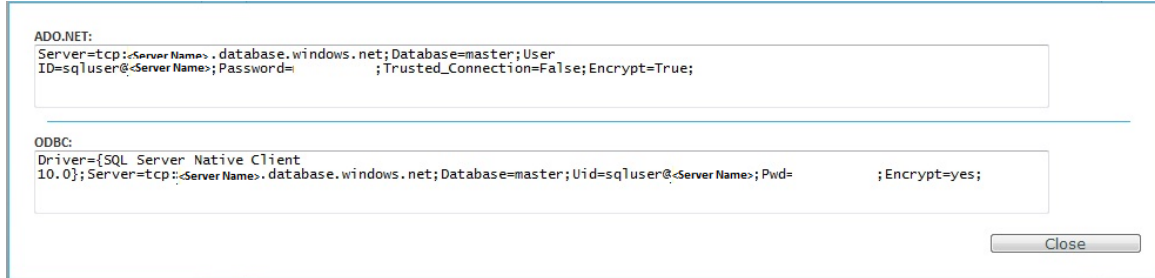
Configuring SQL Azure firewall:

As discussed in Section "How to connect to SQL Azure", one should configure the client IP address (range) to access SQL Azure resources placed behind the firewall. This mechanism assures customers that their data can be accessed only by those clients which are configured. The IP address of client machines can be given in a range. It is advised to provide specific ranges of the client systems rather than providing generic range while configuring the SQL Azure firewall. This enables the primary level of security of SQL Azure at the main data access level. The firewall setting can also be done programmatically using the system stored procedure sp_set_firewall_rule. List of firewall rules configured can be retrieved by querying the system view sys.firewall_rules. Deletion of a firewall rule can be done by using the system stored procedure sp_delete_firewall_rule.

Creation of logins/Users on SQL Azure:

In on – premise SQL Server, logins are used to provide access at server level and users are created to provide access at the database level. These definitions hold good even in SQL Azure. However, because of multi-tenant nature of SQL Azure, no customer of SQL Azure gets hold on server level access permissions. It means that the users will not have certain privileges like changing collation sequence; configuring SQL Jobs, schedule maintenance plans and so on. In SQL Azure, User level information gets stored in "master" database. In order to retrieve the user related details, one should refer to the view "sys.sql_logins". However, in SSMS, the user name gets listed under the specific database for which the access permissions were given.

Specifying connection string in applications connecting to SQL Azure:

On connecting to SQL Azure portal, after creating the database, there is an option to get the connection string that can be copied and used while developing applications as shown in the below screenshot.

```
ADO.NET:
Server=tcp:<Server Name>.database.windows.net;Database=master;User
ID=sqluser@<Server Name>;Password=          ;Trusted_Connection=False;Encrypt=True;


ODBC:
Driver={SQL Server Native Client
10.0};Server=tcp:<Server Name>.database.windows.net;Database=master;Uid=sqluser@<Server Name>;Pwd=          ;Encrypt=yes;

                                                                              Close
```

Note that while providing the username in the connection string, it is mandatory to add @<server name > followed by the user name. It is a best practice to encrypt the connection string in order to store them securely with in the application.

Refer to appendix for scripts which can be used to create logins/users.

Best practices from security perspective:

➢ Use parameterized queries to avoid SQL Injection

➢ Ensure that port 1433 is blocked for inbound calls since SQL Azure requires only outbound connectivity on the port.

➢ While specifying connection string, set encrypt option to TRUE. SQL Azure doesn't support connection strings which are not encrypted. This is to avoid Man-In-The-Middle attack while connecting and transforming the data.

Appendix

Scripts for Login/user creation:

There are different levels of access permissions which can be given to the users based on requirement. Below sequence of steps shows creation of read only user

Step 1 – Create login:

It is always advisable to have separate login and user combination on a database for specific access permissions. Remember, for each database, only one login and user combination with a level of access permissions defined.

```
CREATE LOGIN ReadOnlyLogin WITH PASSWORD = '<Password>'
```

Step 2 – Create a user mapping to the login created above

```
CREATE USER ReadOnlyUser FROM LOGIN ReadOnlyLogin
```

Note: While creating the user, one should connect to the specific database on which the user access permissions are being

Step3 – Grant access permissions

```
EXEC sp_addrolemember 'db_datareader', 'ReadOnlyUser'
```

Similarly, at database level, there are various roles which can be used to provide access permissions based on the requirement.

References

1. Microsoft commitment to cloud Computing
   o http://cloudcomputing.sys-con.com/node/1435637

2. SQL Azure pricing details
   o http://msdn.microsoft.com/en-us/subscriptions/ee461076.aspx

3. Inside SQL Azure
   o http://social.technet.microsoft.com/wiki/contents/articles/inside-sql-azure.aspx

4. SQL Azure Architecture
   o http://msdn.microsoft.com/en-us/library/ee336271.aspx

5. Guidelines and limitations on SQL Azure database
   o http://msdn.microsoft.com/en-us/library/ff394102.aspx

6. Connection management in SQL Azure
   o http://social.technet.microsoft.com/wiki/contents/articles/sql-azure-connection-management-in-sql-azure.aspx

7. SQL Azure storage VS Windows table storage
   o http://msdn.microsoft.com/en-us/magazine/gg309178.aspx

8. SQL Server VS SQL Azure
   o http://www.microsoft.com/downloads/en/details.aspx?FamilyID=86f12b41-1eba-4567-9ac8-02eaa7d12034&displaylang=en

9. Managing databases and Logins in SQL Azure
   o http://msdn.microsoft.com/en-us/library/ee336235.aspx

10. FAQ on SQL Azure
   o http://social.technet.microsoft.com/wiki/contents/articles/sql-azure-faq.aspx

11. Understanding Data Tier Applications
   o http://msdn.microsoft.com/en-us/library/ee240739.aspx

12. Whitepaper on Sharding SQL Azure database
   o http://social.technet.microsoft.com/wiki/contents/articles/sharding-with-sql-azure.aspx

13. Whitepaper on SQL Azure Internals
   o http://social.technet.microsoft.com/wiki/contents/articles/inside-sql-azure.aspx

Infosys among the world's top 50 most respected companies

Reputation Institute's Global Reputation Pulse 2009 ranked Infosys among the world's top 50 most respected companies.

Infosys®

POWERED BY INTELLECT
DRIVEN BY VALUES

About Infosys

Many of the world's most successful organizations rely on Infosys to deliver measurable business value. Infosys provides business consulting, technology, engineering and outsourcing services to help clients in over 30 countries build tomorrow's enterprise.

For more information about Infosys (NASDAQ:INFY), visit www.infosys.com.

Global presence

Americas	Asia Pacific	Europe	Middle East and Africa
Brazil	Australia	Belgium	Kingdom of Saudi Arabia
Nova Lima	Brisbane	Brussels	Riyadh
Canada	Melbourne	Czech Republic	Mauritius
Calgary	Perth	Brno	Reduit
Montreal	Sydney	Prague	UAE
Toronto	China	Denmark	Dubai
Mexico	Shanghai	Copenhagen	Sharjah
Monterrey	Hangzhou	Finland	
United States	Hong Kong	Helsinki	
Atlanta	Central	France	
Bellevue	India	Paris	
Bentonville	Bangalore	Germany	
Bridgewater	Bhubaneshwar	Frankfurt	
Charlotte	Chandigarh	Stuttgart	
Detroit	Chennai	Walldorf	
Fremont	Gurgaon	Ireland	
Hartford	Hyderabad	Dublin	
Houston	Jaipur	Italy	
Lake Forest	Mangalore	Milano	
Lisle	Mumbai	Norway	
New York	Mysore	Oslo	
Phoenix	New Delhi	Poland	
Plano	Pune	Lodz	
Quincy	Thiruvananthapuram	Spain	
Reston	Japan	Madrid	
	Tokyo	Sweden	
	New Zealand	Stockholm	
	Wellington	Switzerland	
	Philippines	Geneva	
	Metro Manila	Zurich	
	Singapore	The Netherlands	
	Singapore	Amsterdam	
		United Kingdom (UK)	
		London	

For more information, contact askus@infosys.com   www.infosys.com

Building Tomorrow's Enterprise