Microsoft

Data Storage Offerings in Windows Azure

December 2011

Contents

Data Storage Offerings in Windows Azure

This article describes the Windows Azure data storage offerings: Blobs, Tables, and SQL Azure. These offerings are hosted in Windows Azure data centers and are available to your applications whether they are running on-premises, hosted within a Windows Azure data center, or hosted within a competing cloud service. The data storage offerings offer many benefits including, high availability, scalability, fault tolerance, geo-replication, easy manageability, limitless storage, and security. In addition, the data is accessible via the Internet to any operating system (OS) and programming language.

If you are running your application code in a Windows Azure data center, then the virtual machine (VM) that is hosting your application exposes two additional storage options called local storage and Windows Azure Drives. These storage options will be discussed at the end of this article.

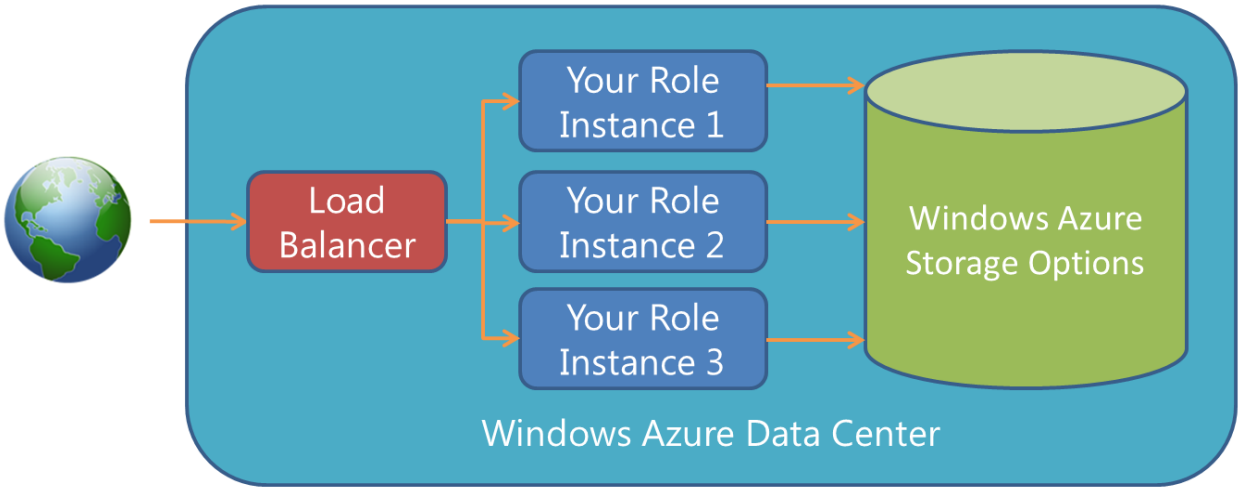
Storing Data in Windows Azure

In this section, we'll discuss how storing your data centrally, outside of machines running your code, lets you produce highly-available and scalable applications. We'll also describe how you can leverage the Windows Azure data storage offerings to assist you when creating highly available and scalable applications.

Imagine your application is running on a machine in which a hardware failure occurs. The hardware failure would make your application not accessible to client requests. If your application is hosted in Windows Azure, Windows Azure detects the hardware failure and automatically moves your application code to a new machine so that your application remains available to clients. However, Windows Azure does not move any data from one machine to another. In fact, it would be impossible to move the data if the hardware failure was due to a hard disk crash. Storing your data outside the actual machine allows the new machine to access it.

In addition, you may want to increase or decrease the number of machines running your application code in order to handle the actual load being placed on your application by client requests. If you have multiple machines running the code then multiple client requests from a single client could be directed to different machines. In order to keep data for a single client accessible to multiple machines, your application has to keep the client's data in a central repository accessible to all the machines running your application code.

The figure below shows how client requests could be distributed to different machines all of which have access the same data storage.



For example, let's say that your application allows a client to place items into a shopping cart and later, the client purchases the items they previously put into their cart. In this example, the client's request to add an item to the shopping cart could occur on one machine and the request to checkout could be directed to a different machine. The second machine would need to know the items in the client's cart.

In order to achieve highly available and scalable applications, Windows Azure Platform Services offers multitenant storage machines within the various Windows Azure data centers. These machines replicate your data ensuring that if one replica fails, others are still viable. These storage offerings can be accessed by applications running in a Windows Azure data center and can also be accessed directly by applications running on-premises or hosted in another cloud service.

The figure above shows Blobs, Tables, and SQL Azure as the central data repository. To store data in any of these offerings, you must first select which of the Windows Azure data centers you want to house your data. The table below shows the locations of 6 Windows Azure data centers throughout the globe:

Country/Region	Sub-regions	
United States	South Central	North Central
Europe	North	West
Asia	Southeast	East

Normally, you select a sub-region close to the clients that access your data in order to reduce latency thereby improving performance. In addition, you would select a sub-region that meets any jurisdiction issues relevant to your data. The data you store is replicated three times within this sub-region in order to you give you a high degree of fault tolerance. In addition, since the data is replicated, it can be served from multiple machines simultaneously thus increasing the performance of accessing your data. In fact, the data storage service constantly monitors the load on its machines and automatically load balances your data across its servers to maintain a high degree of throughput.

Furthermore, Blobs and Tables are, by default, replicated to the other sub-region providing you with a high degree of disaster recovery. For example, if you select the United States North Central sub-region to house your data, then your data will be replicated in this sub-region and will also be replicated in the United States South Central sub-region as well. Should the North Central data center experience a disaster, then your data will automatically start being served

from the South Central data center. If you choose, you can opt out of geo-replication by telephoning Windows Azure support.

When using Blobs, Tables, or SQL Azure, the data can be encrypted as it goes over the wire. However, the data is stored unencrypted inside the Windows Azure data centers. In many cases, you can encrypt the data yourself before sending it over the wire; Azure will simply store whatever you send it and now the data will be stored encrypted. The Windows Azure SDK includes a storage emulator which you can use on your local machine to test your code's use of Blobs and Tables. The emulator is free of charge to use and requires the use of SQL Server or SQL Express. You can test your code's use of SQL Azure by having it work against a locally installed instance of SQL Server or SQL Express.

The pricing for Blobs and Tables is related to the storage you consume, the number of bytes you transfer out of the data center (transfer into the data center is free) and the number of I/O operations you perform. The pricing for SQL Azure if based on the size of each database and the number of bytes you transfer out of the data center. To understand the latest pricing of these various storage mechanisms, please see Windows Azure Pricing.

Windows Azure Blobs and Tables

Blobs and Tables reside within a Windows Azure Storage account. A single Windows Azure Storage account can hold up to 100 TB of data. If you need to store more data, then you can create additional storage accounts. Within a storage account, you can store data in Blobs and Tables. A storage account can also contain Queues (for more information see How to Use the Queue Storage Service). Here is a brief description for Blobs and Tables (they are described in more detail in the following sections):

- **Blobs (Binary Large Objects).** Blobs are for storing individual data items (like files) which can be large in size. A single Blob typically contains a document (xml, docx, pptx, xlxs, pdf, etc.), a picture, a song or a video. Blobs can be publicly exposed for read-only access and retrieved via HTTP(S) URLs.
- **Tables.** Tables contain large collections of property-bag state (called entities) such as customer information, order data, news feed items. Tables sort their entities and can return a filtered subset of the entities.

These abstractions are accessible via HTTP(S) REST APIs (overs ports 80 and 443) making them available to all operating systems and all programming languages. To simplify making the web requests for software developers, you can use several *client libraries* targeting specific operating systems and programming languages. Libraries exist for .NET, Node.js, Java, and PHP, and are available at the Windows Azure website. The term "client libraries" is used since your application is a client to the data storage servers. The figure below shows how your application code makes a call into a client library which, in turn, makes an HTTP(S) REST request to Blobs or Tables.

When you create a storage account, your account is assigned two 256-bit account keys. One of these two keys must be specified in a header that is part of the HTTP(S) request. Having two keys allows for key rotation in order to maintain good security on your data. Typically, your applications would use one of the keys to access your data. Then,

after a period of time (determined by you), you have your applications switch over to using the second key. Once you know your applications are using the second key, you retire the first key and then generate a new key. Using the two keys this way allows your applications access to the data without incurring any downtime.

Blobs

Blobs provide a way to store large amounts of unstructured, binary data, such as documents, pictures, audio, video, etc. In fact, one of the features of Blobs is streaming content such as audio or video. A storage account has blob containers and a container contains blobs. Containers are similar to directories and contain blobs. There are two kinds of Blobs:

- **Block Blobs** are ideal for most scenarios and allow you to easily insert, delete, and reorder blocks within a blob. Each block is identified with an ID and contains up to 4MB of data. The maximum size of a Block Blob is 200GB.
- **Page Blobs** are typically used for virtual hard drives (VHDs) and are much less commonly used than Block Blobs. Page Blobs allow you to save money when blobs contain long runs of zeros and have better performance for random write operations. A Page Blob is composed of 512-byte pages. Any page that you do not write to logically contains zeros and is not physically backed by storage and therefore you are not charged for these pages. The maximum size of a Page Blob is 1TB.

There are many features available to you when using Blobs. The list below enumerates some of them:

- Blobs can be accessed via HTTP(S) REST APIs or via a client library
- Blobs can be exposed for read-only public access which makes them great for storing public pictures, videos, etc.
- You can create a Shared Access Signature (SAS) for a blob allowing others that do not have one of the 256-bit storage account keys to read, write, or delete the blob during a specified window of time.
- You can create snapshots of a blob effectively making read-only versions. This is useful for maintaining different versions of a document, photo, or video.
- An application can acquire a lease on a blob allowing it to make multiple exclusive updates to the blob.
- You can use the Content Delivery Network (CDN) to cache and deliver blobs from data centers that are closer to your customers thereby improving performance. The CDN also offers smooth streaming support for blob videos.

To ease the migration of existing applications that use standard File I/O APIs to access files, Windows Azure offers a feature called Windows Azure Drives. This feature effectively mounts a VHD stored as a Page Blob as a disk drive. This allows your application code to use standard FILE I/O APIs against the mounted drive letter and redirect the I/O operations to the backing blob. This feature is only available to applications running in Windows Azure because the VMs must have a special NTFS device driver installed on them. See Windows Azure Drives for more details.

Tables

Tables provide a non-relational key/property bag collection useful for storing tabular data such as customer information, orders, news feeds, and game scores. Key benefits of tables are that they are pay-for-consumption, can handle small and large (100 TB) amounts of data efficiently, and are accessible via HTTP(S). For data that requires

server-side computation such as joins, sorts, views, and stored procedures, you should consider using SQL Azure as Tables do not support these features. See SQL Azure for more details.

A table contains entities. Unlike database tables, the entities within a single table can have differing sets of properties. For example, you can store customer entities and order entities within the same table. Tables are similar to rows within a spreadsheet application (such as Excel) in that each row can contain a different number of columns and the cells can vary their data type.

An entity can be up to 1MB in size and consist of a set of up to 255 tuples. Each tuple represents a property with its name, data type, and value. The set of valid data types are shown below:

Date Type	Description
String	A UTF-16-encoded string of characters up to 64 KB in size.
Byte array	An array of bytes up to 64 KB in size.
Guid	A 128-bit globally unique identifier (can be null).
DateTime	A 64-bit value expressed as Coordinated Universal Time (UTC) ranging from 12:00 midnight, January 1, 1601 A.D. (C.E.) to December 31, 9999 (can be null).
Int32	A 32-bit signed integer (can be null)
Int64	A 64-bit signed integer (can be null).
Double	A 64-bit floating point value (can be null)
Boolean	A Boolean value (can be null)

Here is how to visualize entities within a table:

PartitionKey	RowKey	Timestamp	Property Collection		
			Name	Type	Value
Jeff	C_Jeff	🕑	Kind Name City	String String String	"Customer" "Jeffrey" "Seattle"
Jeff	O_Cereal	🕑	Kind Item Quantity	String String Int32	"Order" "Cereal" 1
Jeff	O_Milk	🕑	Kind Item Quantity	String String Int32	"Order" "Milk" 2
Paul	C_Paul	🕑	Kind Name City	String String String	"Customer" "Paul" "New York"

As you can see, all entities must have the following three tuples (the remaining 252 are determined by you):

- Timestamp: a DateTime
- PartitionKey: a 1KB String
- RowKey: a 1KB String

The timestamp value is always assigned by the Windows Azure; you do not have control over this value. Whenever an entity is added or updated, Windows Azure assigns the current time to the property. This property is used for versioning and optimistic concurrency scenarios.

The PartitonKey and RowKey together must uniquely identify a single entity within a table. That is, you cannot have two entities with identical PartitionKey and RowKey values inside a single table. What you choose to use for a PartitionKey is critically important when you're designing a table because the PartitionKey value controls several things which are often in conflict with each other. Specifically, the PartitionKey controls:

- **Entity Sort Order**. Within a table, entities are sorted in order of their PartitionKey value. Entities that share the same PartitionKey value are sorted by the RowKey value.
- **Entity Partitioning**. The Windows Azure Table Service can serve different partitions from different machines. So, the value you chose for the PartitionKey determines the scalability of accessing entities within your table. If all entities have the same PartitionKey value, then the service cannot serve some entity requests from 1 machine and some entity request from another machine. Queries for entities within a single partition are faster than queries that span partitions.
- **Entity Group Transactions**. Tables allow multiple entities to be updated as a single atomic transaction; this is called an *Entity Group Transaction (EGT)*. However, an EGT is allowed only for entities that share the same PartitionKey value because all these entities are guaranteed to be served from a single machine. EGTs are the main reason to allow entities with different schemas to exist in a single table. For example, using an EGT, you can update a customer and all their orders as a single atomic transaction.

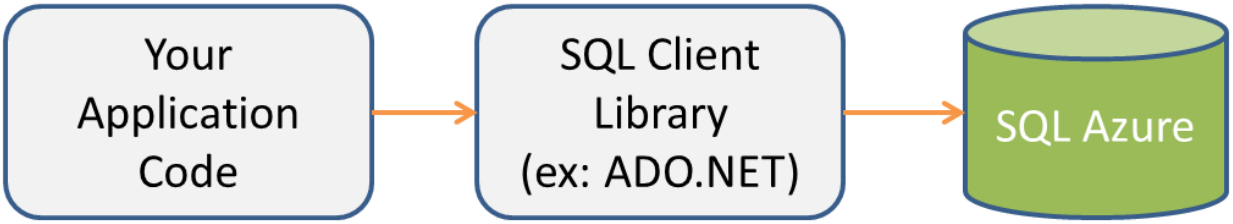
Table storage supports basic create, read (with filtering), update, & delete (CRUD) operations and these operations are performed using OData over HTTP(S) REST APIs. But, unlike database tables, it does not support joins, foreign keys, stored procedures, triggers, or any processing (other than filtering) on the storage engine side. Features like these are supported by SQL Azure, described in the next section. Queries returning a large number of results, or queries that take more than 5 seconds return partial results along with a *continuation token*. You can process the results returned and then pass the *continuation token* back to the service to get the next set of results.

SQL Azure

SQL Azure provides a highly-available and scalable relational database management system for Windows Azure. Key benefits of SQL Azure are SQL Server compatibility which allows you to use familiar T-SQL, SQL library APIs, tools, etc. For example, ADO.NET and ODBC continue to work with SQL Azure with minimal code changes. For data that requires server-side computation such as joins, sorts, views, and stored procedures, SQL Azure is an ideal choice. Alternatively, if you do not need these features, then Windows Azure Tables may be a better option.

SQL Azure is a multitenant service which maintains scalability by automatically moving databases from heavily accessed machines to other machines. You access the database using the Tabular Data Stream (TDS) protocol on port 1433; In addition, access is granted from specific IP ranges and a password. Like Blobs & Tables, SQL Azure can also be accessed via applications running on-premises, inside a Windows Azure data center or via some other cloud service.

The figure below shows how your application code makes a call into a SQL client library (such as ADO.NET) which, in turn, makes a TDS request over TCP to SQL Azure.



Since Windows Azure does not provide direct access to the underlying hardware, administration tasks that involve hardware access, such as defining where the database file is located, are inaccessible in SQL Azure. Physical administration tasks are handled automatically by the platform, though you must still perform logical administration tasks such as creating logins, users, roles, etc. Because you cannot directly access the hardware, there are some differences between SQL Server and SQL Azure in terms of administration, provisioning, T-SQL support, programming model and features.

Unlike the other storage offerings discussed so-far, SQL Azure provides more than simple data storage; it also provides server side processing, which allows you to perform complex processing on stored data without having to retrieve and process the entire data set within your application. For example, a query to find all salesmen with sales greater than $1,000.00 in the past year, within a specific region of the country, and provide a sum total of all their sales, can be executed completely by SQL Azure and the results returned to your application.

A SQL Azure database can be up to 150GB in size and can contain multiple tables with complex relationships between data in the tables. Rows can be up to 8MB in size, and can contain 1024 columns. A table within SQL Azure can have one clustered index on any column, and up to 999 secondary indexes.

Additional Storage Options for Applications Hosted in Windows Azure

When your application is running in Windows Azure, the VMs running your application code have access to local storage and can also leverage Windows Azure Drives. The next two sections explain these storage mechanisms.

Local Storage

When you run your application in Windows Azure, it is hosted in VMs. Each VM has a virtual hard drive connected to it and you can access subdirectories contained on this drive. The subdirectories provide fast, temporary storage for your application instance. In your code, you use standard file I/O operations (such as .NET's FileStream class) to store any data you choose in these subdirectories.

The subdirectories are only accessible by the code running on the VM and can be configured to persist or be erased whenever the VM reboots. However, if Windows Azure moves your code to another machine due to hardware failure or maintenance, be aware that the data in the subdirectories will not be moved with your code; the data will be lost. For this reason, local storage is best thought of as a cache allowing fast access to frequently-used data. Local storage is also useful for building up data (such as logs) that is then frequently transferred to a more durable storage mechanism such as Blobs, Tables, or SQL Azure. Consuming and accessing local storage is completely free.

Each local storage subdirectory can hold between 1MB and the maximum amount allowed by the VM size that you have configured for your service. The table below shows the available VM sizes and the maximum amount of disk space available across all the local storage subdirectories on the VM:

VM Size	Max disk space for Local Storage subdirectories
Extra Small	20GB
Small	225GB
Medium	490GB
Large	1TB
Extra Large	2TB

Windows Azure Drives

Windows Azure Drives provides enables existing applications to run in Windows Azure with minimal code changes by providing a hard disk that is backed by a blob.

A lot of existing applications use normal file system APIs to store data on the local hard drive. However, as pointed out at the beginning of this article, local storage is not accessible to other machines and therefore prohibits the creating of highly-available applications. A Windows Azure Drive allows you to create a virtual hard drive (VHD) backed by a Page Blob and mount that drive into the VM running your application code. You can create a VHD on your computer and then upload it to blob storage or you can use the REST API to have Windows Azure create the blob.

Once in Blob storage, your application code can mount the VHD into your application's VM. The VM has a special file system device driver on it that accepts normal file system I/O requests and translates them into operations on the blob-backed VHD. If the machine fails, then another machine can run the same code and mount the VHD accessing the data that was saved previously. Note that only one machine at a time can mount a Windows Azure Drive for writing which is why this feature is not for applications that are scaling out with multiple instances. However, it is possible to mount a read-only (snapshot) version of the blob into multiple VMs simultaneously. This can make Windows Azure Drives useful if you have a lot of read-only static content (like static HTML files).

References

- Windows Azure Pricing
- Windows Azure Storage
- Windows Azure Storage Services REST API Reference
- Windows Azure Content Delivery Network (CDN)
- Storing and Accessing Data in Windows Azure
- General Guidelines and Limitations (SQL Azure Database)
- Comparing SQL Server with SQL Azure
- SQL Azure MSDN library
- SQL Azure Survival Guide
- Overview of Options for Migrating Data and Schema to SQL Azure
- Handling Transactions in SQL Azure
- Configuring Local Storage Resources