



#02

súbete a la nube de Microsoft

Almacenamiento en Windows Azure

Ibón Landa Martín
Unai Zorrilla Castro

campus
MVP.com



Plain Concepts
The Microsoft Technologies Company

Súbete a la nube de Microsoft

Parte 2: Almacenamiento en Windows Azure



Ibón Landa Martín
Unai Zorrilla Castro



SÚBETE A LA NUBE DE MICROSOFT PARTE 2: ALMACENAMIENTO EN WINDOWS AZURE

Noviembre de 2011



Esta obra está editada por **Krasis Consulting, S.L.** (www.Krasis.com) y **Plain Concepts S.L.** (<http://www.PlainConcepts.com>) bajo los términos de la licencia “**Creative Commons Reconocimiento-NoComercial-SinObraDerivada Unported (CC BY-NC-ND 3.0)**”, que permite su copia y distribución por cualquier medio siempre que mantenga el reconocimiento a sus autores, no haga uso comercial de la obra y no realice ninguna modificación de ella.

Contenido

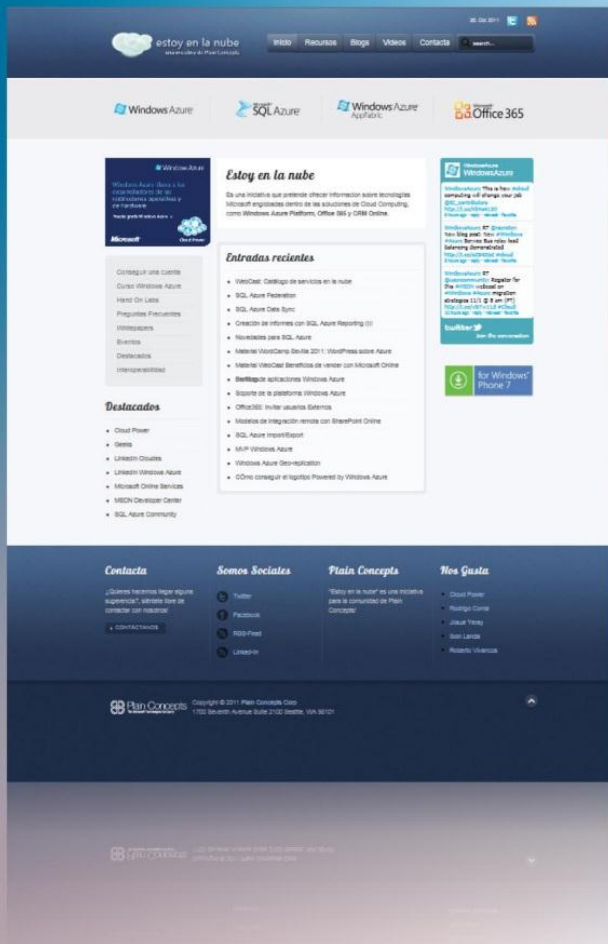
CONTENIDO	3
SQL AZURE	1
1.- Arquitectura.....	2
2.- Acceso a datos en SQL azure	3
3.- Modelo de aprovisionamiento.....	4
3.1.- Cuentas de SQL Azure.....	4
3.2.- Servidores	4
3.3.- Base de datos	4
3.4.- Modelo de seguridad.....	5
4.- Trabajo con SQL azure.....	5
5.- Database manager	9
6.- Particularidades de SQL azure respecto a SQL server	10
6.1.- Soporte T-SQL.....	11
6.2.- Índices cluster	11
6.3.- Modelo de seguridad.....	11
6.4.- Tamaño de SQL Azure.....	12
6.5.- El collation de SQL Azure	13
6.6.- DMVs	14
7.- Acceso programático a SQL azure	14
7.1.- Conectarse desde ASP.NET	15
7.2.- Conectarse desde Entity Framework.....	15
7.3.- Conectarse desde PHP.....	15
8.- Administración de SQL azure	17
8.1.- Migración de SQL Server a SQL Azure	17
8.2.- Conectarse desde SQL Server Management Studio.....	20
8.3.- Conectarse a SQL Azure usando SQLcmd.....	21
8.4.- SQL Azure Migration Wizard.....	22
9.- Sobre el tamaño de SQL azure.....	23
10.- DMV.....	24
11.- Exponer por odata el contenido de SQL azure	25
12.- Herramientas para trabajar con SQL azure.....	26
12.1.- SQL Azure Migration Wizard.....	26
12.2.- Microsoft SQL Server Migration Wizard	31
13.- SQL Import/Export.....	35
14.- SQL Azure Data Sync.....	39
14.1.- Sincronización entre base de datos SQL Azure	40
14.2.- Sincronización con un servidor on-premise	45
15.- SQL Azure Reporting Services	48
16.- SQL Azure Federation	53
WINDOWS AZURE STORAGE	55
1.- Almacenamiento en Windows Azure	55
2.- Windows azure tables.....	56
2.1.- Entidades y tablas.....	56
2.2.- Contexto de acceso a datos	57
2.3.- Orígenes de datos	58
2.4.- La cadena de conexión	58
3.- Windows azure blobs.....	59
3.1.- Trabajo con Blobs.....	60

4.-	Windows azure queue	62
4.1.-	Trabajo con colas.....	62
5.-	Windows azure drive	64
6.-	Del compute emuletor a la nube	66
7.-	Windows Azure mmc	68
8.-	Cloud Storage Studio	69
9.-	Windows azure CDN	73
10.-	Windows Azure Geo-Replication	74

estoy en la nube

INICIATIVA DE PLAIN CONCEPTS

www.estoyenlanube.com



 Windows Azure

www.plainconcepts.com

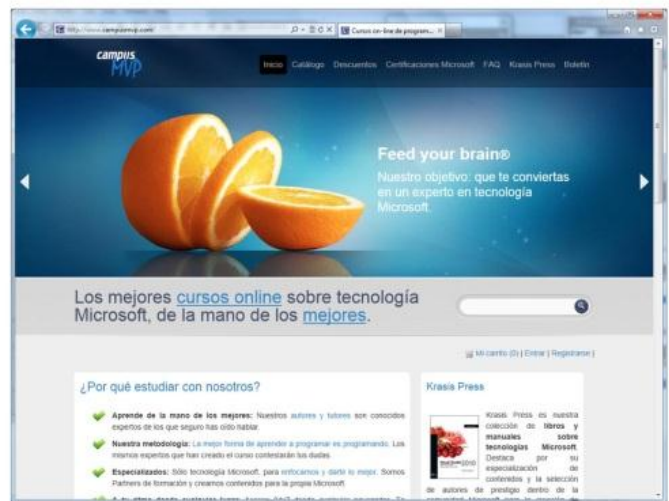
Plain Concepts is a company specialized in Microsoft technologies, agile methodologies, Application Lifecycle Management, performance tuning, advanced debugging, software architecture and User Experience.

Plain Concepts focuses on delivering high quality consulting, mentoring and training as well as in being an effective and reliable team resolving all type of software development issues.

¿Aún quieres más?

**campus
MVP**

Formación online especializada
en tecnologías Microsoft.



**krasis
PRESS**

Los libros que lo saben todo sobre
tecnologías Microsoft.

Síguenos y descubrirás los mejores trucos y recursos:

[facebook.com/campusmvp](https://www.facebook.com/campusmvp) twitter.com/campusmvp

 **feed your brain®**

- ☑ Sin tener que desplazarse
- ☑ Sin romper el ritmo de trabajo
- ☑ Preguntándole a los que más saben

infórmate ya:

902 876 475
www.campusmvp.com

<http://www.krasis.com>



krasis

Microsoft Partner
Silver Learning
Silver Software Development



SQL Azure

SQL Azure es una base de datos relacional en la nube construida sobre la tecnología de SQL Server. Proporciona servicios de bases de datos altamente escalables y con altísima disponibilidad alojados por Microsoft en la nube. Estos servicios facilitan enormemente el despliegue de bases de datos.

La gran ventaja de utilizar SQL Azure frente a otros sistemas de almacenamiento en la nube es que todos los conocimientos sobre bases de datos relacionales y el lenguaje de consulta SQL siguen siendo válidos. No es necesario adaptar los conocimientos a nuevos paradigmas de almacenamiento, como pasa con otros sistemas de almacenamiento en la nube no basados en bases de datos relacionales ni SQL. Si sabes utilizar SQL Server, todos tus conocimientos te valen para SQL Azure.

Este punto es justamente uno de los aspectos clave desde el punto de vista de desarrollo. La mayoría de las aplicaciones que desarrollamos usan en mayor o menor medida un almacenamiento relacional. El hecho de disponer de este tipo de almacenamiento en la nube puede posibilitar y simplificar la migración de aplicaciones a esta plataforma.

Conocer las limitaciones de SQL Azure es clave para saber si podríamos usarla o no.

SQL Azure permite incluso migrar los backend de datos a la nube si tener que tocar ni una sola línea de código de las aplicaciones en un gran número de escenarios. Es cierto que hay ciertas características de SQL Server que SQL Azure no soporta, pero si soporta todas las más usadas:

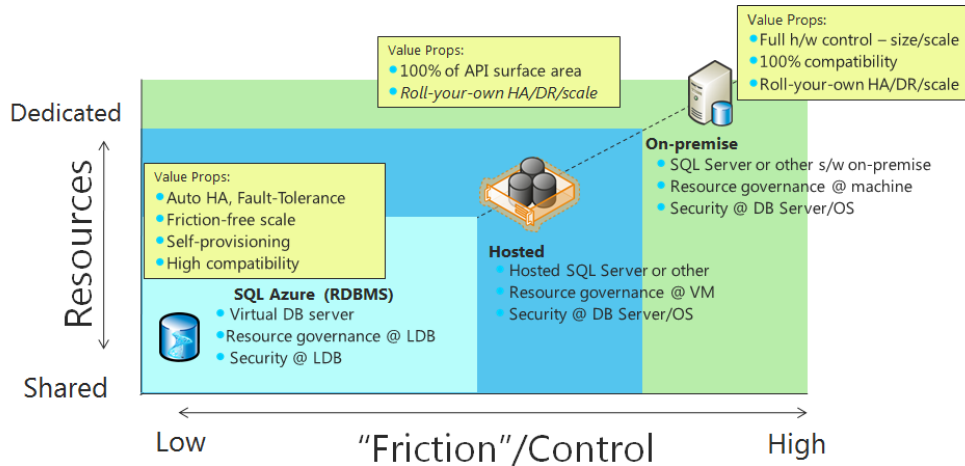
- Tablas, tablas temporales, vistas, índices, roles, procedimientos almacenados y funciones.
- Consultas complejas y 'joins' entre múltiples tablas.
- Insert, update y delete.
- Restricciones
- Transacciones

Entre las características no soportadas cabe destacar:

- Transacciones distribuidas
- El broker de mensajes de SQL Server
- Consultas a servidores remotos
- Acceso desde tecnologías antiguas, ya obsoletas, en concreto OleDb.

A la hora de conectar desde nuestras aplicaciones clientes, podemos elegir varios tipos de conexión:

- ADO.NET, incluido Entity Framework.
- Acceso ODBC nativo.
- Soporte para PHP.



I.-ARQUITECTURA

SQL Azure expone a través del protocolo TDS (Tabular Data Stream) las bases de datos existentes en la nube. El protocolo TDS es el mismo protocolo que emplea SQL Server, por lo que una aplicación cliente puede conectarse a SQL Azure de la misma manera en que se conecta a un SQL Server.

Este hecho provoca que la gran mayoría de las aplicaciones existentes puede llegar a conectar a SQL Azure sin que este hecho suponga cambiar una sólo línea de código. A la hora de conectar desde las aplicaciones clientes, SQL Azure permite elegir varios tipos de conexión:

- ADO.NET, incluido Entity Framework.
- Acceso ODBC nativo.
- Soporte para PHP.

De hecho, la gran ventaja de utilizar SQL Azure frente a otros sistemas de almacenamiento en la nube es que todos los conocimientos sobre bases de datos relacionales y el lenguaje de consulta SQL siguen siendo válidos. No es necesario adaptar los conocimientos a nuevos paradigmas de almacenamiento, como pasa con otros sistemas de almacenamiento en la nube no basados en bases de datos relacionales ni SQL.

Aunque la forma de conectarse sea idéntica a la forma de conectarse a un SQL Sever, no debe olvidarse la latencia de la red. Desde el punto de vista de rendimiento siempre hay que considerar que la base de datos de SQL Azure se encuentra en un Data Center remoto y que el nivel de eficiencia no puede ser el mismo que podría llegar a conseguirse empleando un SQL Server dentro de la misma red.

Por este motivo, es recomendable elegir correctamente la ubicación del servidor SQL Azure, eligiendo la ubicación más cercana posible a la aplicación para evitar los costes adicionales del consumo de ancho de banda y lograr un mejor rendimiento.

SQL Azure, dentro del mismo Data Center, está diseñado como un sistema de réplicas a través de múltiples servidor físicos. Esta arquitectura proporciona un sistema automático de balanceo de carga y recuperación ante errores.

Existen tres instancias de SQL Azure por cada servidor. El sistema de réplicas en un sistema completamente transparente a la aplicación, cuya única finalidad es ofrecer un sistema de escalabilidad y disponibilidad.

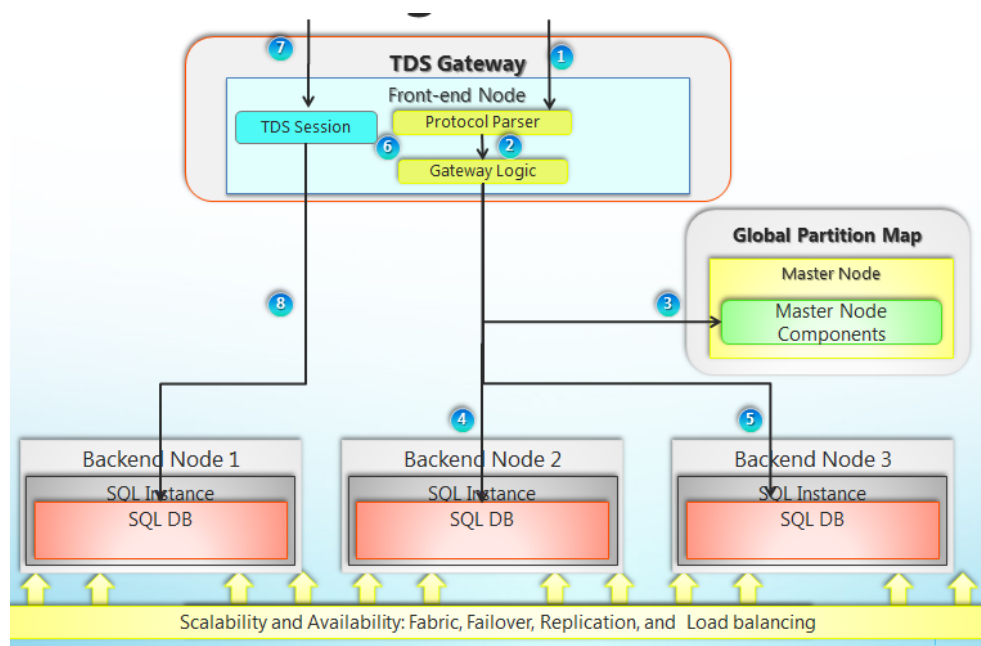


Figura I.1.- Arquitectura SQL Azure

2.-ACCESO A DATOS EN SQL AZURE

La gran ventaja de utilizar SQL Azure frente a otros sistemas de almacenamiento en la nube es que todos los conocimientos sobre bases de datos relacionales y el lenguaje de consulta SQL siguen siendo válidos. No es necesario adaptarse a nuevos paradigmas de almacenamiento, como pasa con otros sistemas de almacenamiento en la nube no basados en bases de datos relacionales ni SQL.

SQL Azure, construido sobre SQL Server, permite incluso migrar backends de datos a la nube si tener que tocar ni una sola línea de código de las aplicaciones en un gran número de escenarios.

SQL Azure expone a través del protocolo TDS (Tabular Data Stream) las bases de datos existentes en la nube. El protocolo TDS es el mismo protocolo que emplea SQL Server, por lo que una aplicación cliente puede conectarse a SQL Azure de la misma manera en que se conecta a un SQL Server.

A la hora de conectar desde nuestras aplicaciones clientes, podemos elegir varios tipos de conexión:

- ADO.NET, incluido Entity Framework.
- Acceso ODBC nativo.
- Soporte para PHP.

En cuando a la ubicación de la aplicación cliente, no existen restricciones a la hora de conectarse a SQL Azure. Cualquier aplicación puede conectarse a una base de datos de SQL Azure siempre y cuando tenga los permisos adecuados y las reglas del firewall de SQL Azure se encuentren debidamente configuradas.

Aunque la forma de conectarse sea idéntica a la forma de conectarse a un SQL Sever, no debe olvidarse la latencia de la red. Desde el punto de vista de rendimiento debemos siempre considerar que la base de datos de SQL Azure se encuentra en un Data Center remoto y que el nivel de eficiencia no puede ser el mismo que podría llegar a conseguirse empleando un SQL Server dentro de la misma red.

Por este motivo, desde el punto de vista de diseño de la arquitectura de una aplicación, debe valorarse cuál podría ser la mejor ubicación de la aplicación cliente para poder ofrecer el rendimiento esperado. En muchas ocasiones, el escenario que mejor rendimiento permite es ubicar la aplicación cliente dentro de Windows Azure, en el Data Center más cercano posible.

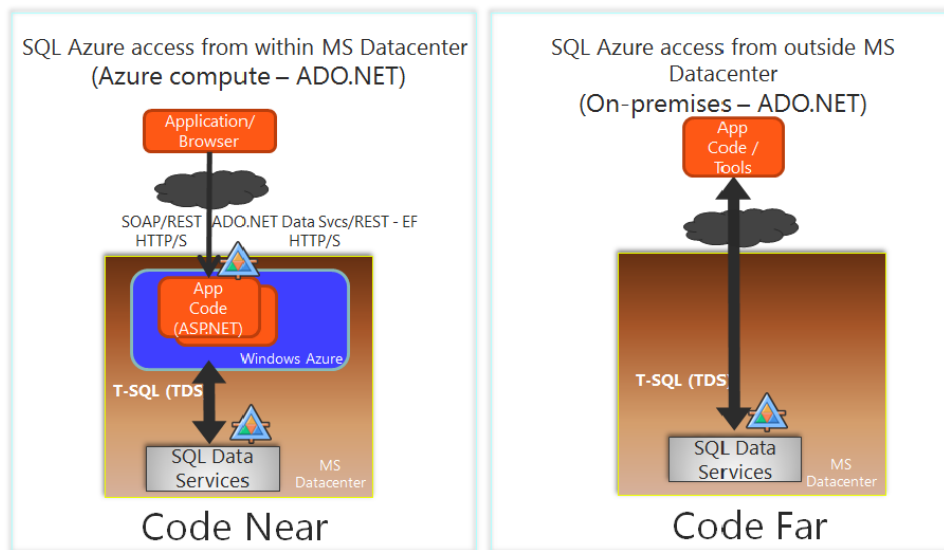


Figura 1.2.- Acceso a datos

3.-MODELO DE APROVISIONAMIENTO

SQL Azure es una base de datos relacional en la nube construida sobre la tecnología de SQL Server. Proporciona servicios de bases de datos altamente escalables y con altísima disponibilidad alojados por Microsoft en la nube. Estos servicios facilitan enormemente el despliegue de bases de datos.

Una ventaja añadida es que los desarrolladores y el personal de IT no necesitan instalar, actualizar y gestionar la infraestructura de bases de datos. La alta disponibilidad, aspecto siempre complejo, es gestionado para nosotros de manera transparente.

SQL Azure dispone de un modelo jerárquico de aprovisionamiento que se describe a continuación.

3.1.- Cuentas de SQL Azure

Para poder comenzar a utilizar SQL Azure el primer paso que debe realizar es crear una cuenta de la plataforma Windows Azure. Una vez creada la cuenta será posible acceder a todas las capacidades que la cuenta ofrece, entre ellas, la posibilidad de usar SQL Azure.

Importante: Una sola cuenta puede contener cero o más suscripciones. Una cuenta representa la forma en que se establece una relación de facturación con Microsoft. Una sola suscripción de Windows Azure puede contener múltiples servicios, como Windows Azure, Windows Azure AppFabric, y Azure SQL.

3.2.- Servidores

Cada una cuenta de Windows Azure puede contener varios servidores de SQL Azure. Estos servidores no deben entenderse como instancias de SQL Server, en cambio, sí se puede entender como un concepto lógico que se utiliza para proporcionar un punto de administración central para múltiples servidores SQL Azure.

Cada servidor dispone de la lógica necesaria para incluir los inicios de sesión, del mismo modo que éstos se realizan en SQL Server, pudiendo indicar en cada servidor la región geográfica dónde debe ubicarse el mismo.

Se utiliza el portal de SQL Azure para crear y gestionar el servidor de base de datos.

3.3.- Base de datos

Cada servidor de base de datos SQL Azure puede contener múltiples bases de datos.

Un servidor de base de datos tiene una base de datos master, similar a la que se puede encontrar en un SQL Server.

En cada base de datos se podrán realizar las labores típicas que pueden realizarse sobre SQL Server. Es cierto que hay ciertas características de SQL Server que SQL Azure no soporta, pero si soporta todas las más usadas:

- Tablas, tablas temporales, vistas, índices, roles, procedimientos almacenados y funciones.
- Consultas complejas y 'joins' entre múltiples tablas.
- Insert, update y delete.
- Restricciones
- Transacciones

SQL Azure, dentro del mismo Data Center, está desarrollado como un sistema de réplicas a través de múltiples servidor físicos. Esta arquitectura proporciona un sistema automático de balanceo de carga y recuperación ante errores.

3.4.- Modelo de seguridad

Las bases de datos de SQL Azure pueden contener información confidencial por lo que es esencial para controlar cuidadosamente el acceso. Este aspecto se hace especialmente importante debido a que una base de datos de una cuenta puede compartir el mismo servidor con otras bases de datos de otras cuentas. SQL Azure debe poder ofrecer un nivel de aislamiento adecuado para asegurar la confidencialidad de los datos.

SQL Azure se basa en los mismos mecanismos de seguridad existentes en SQL Server.

- Logins de SQL Server: Acceso autenticado a SQL Azure a nivel de servidor.
- Usuarios de base de datos: Permite dar permisos a nivel de base de datos.
- Roles de base de datos: Permite dar permisos a nivel de base de datos a grupos.

4.- TRABAJO CON SQL AZURE

A continuación se muestra algunas de las labores que podemos realizar trabajando con SQL Azure.

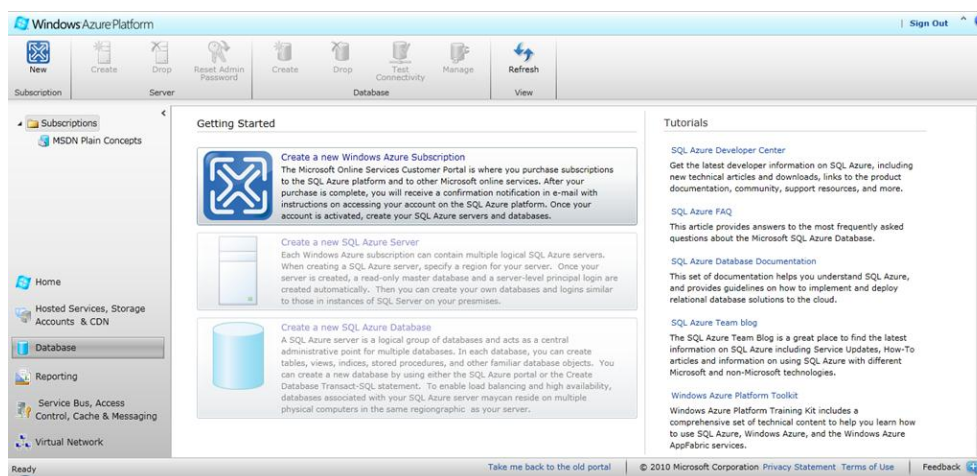


Figura 1.3.- Portal de Windows Azure

Lo primero que debe hacerse es crear un nuevo servidor de base de datos sobre la cuenta de Azure que se seleccione. Una cuenta puede tener múltiples servidores asociados. Un mismo servidor puede tener múltiples base de datos asociadas.

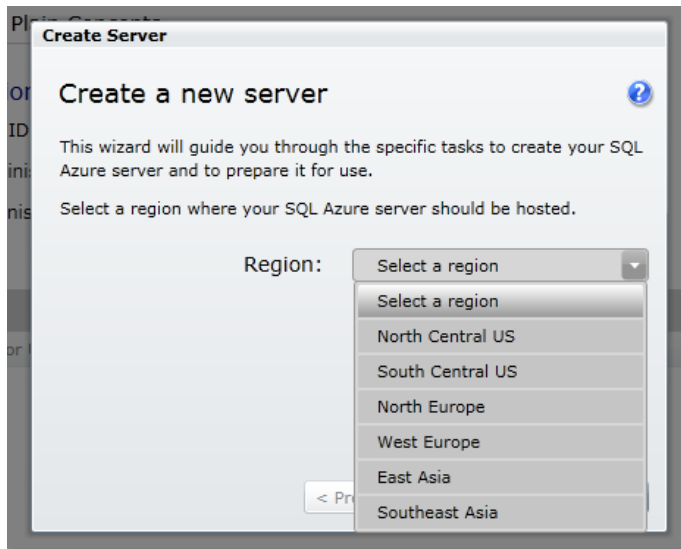


Figura 1.4.- Seleccionar la región

El siguiente paso es indicar la cuenta de administrador del servidor.

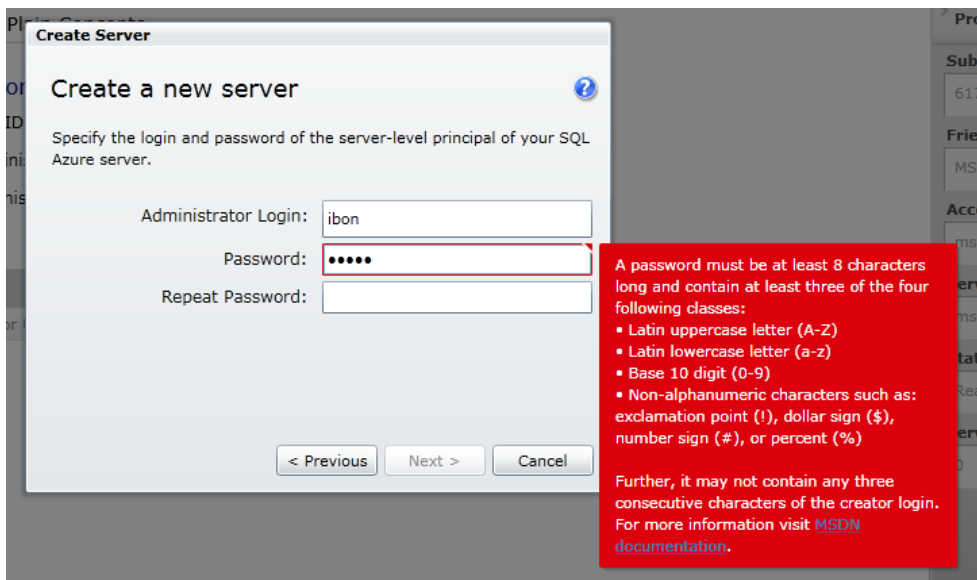


Figura 1.5.- Credenciales de administrador

Se ha incluido como parte del proceso de creación del servidor la configuración del firewall.

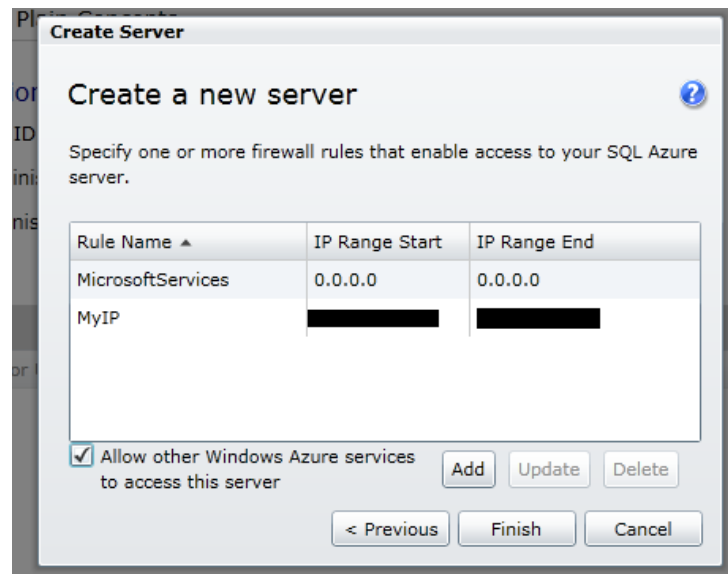


Figura I.6.- Configuración del firewall

Una vez creado el servidor, se puede acceder a toda la información del mismo; base de datos asociadas, datos de conexión etc...

Como siempre, en la barra de herramientas superior estarán disponibles todas las opciones habituales en función de lo que se está viendo en la pantalla.

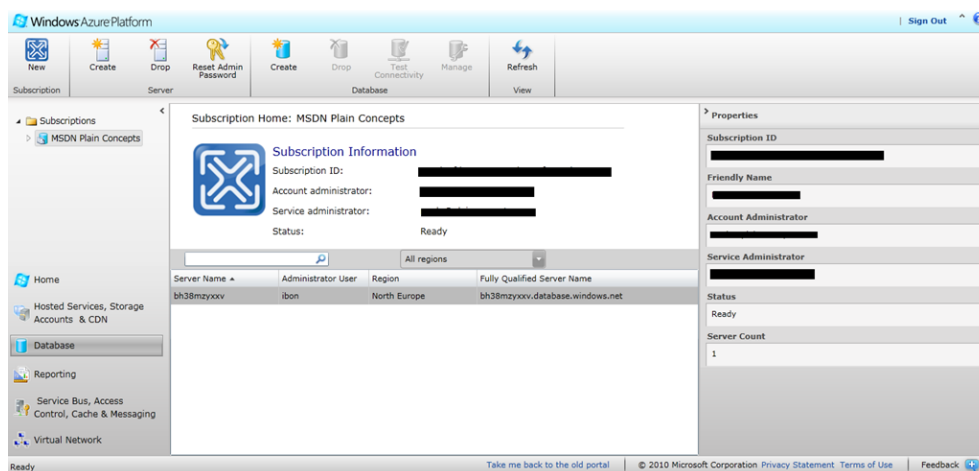


Figura I.7.- Administración SQL Azure

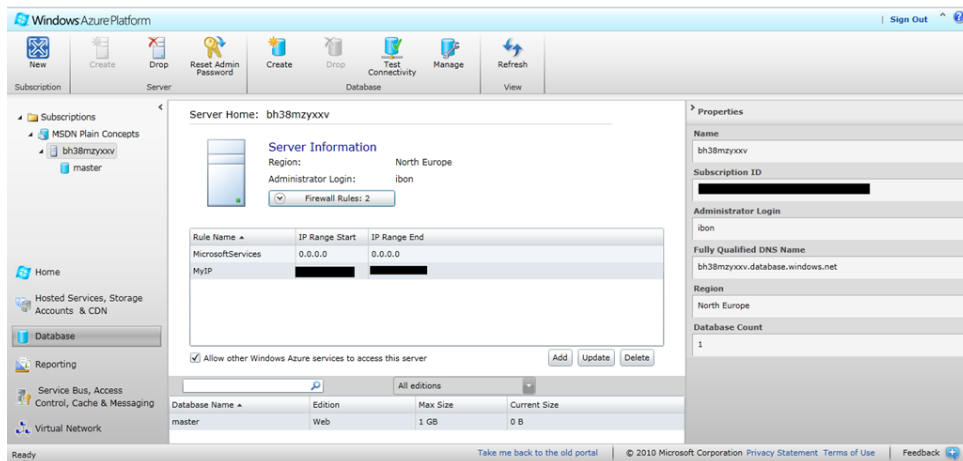


Figura I.8.- Administración SQL Azure

Se pueden crear tantas bases de datos como se quieran asociadas al servidor de base de datos, indicando la edición y el tamaño de la misma.

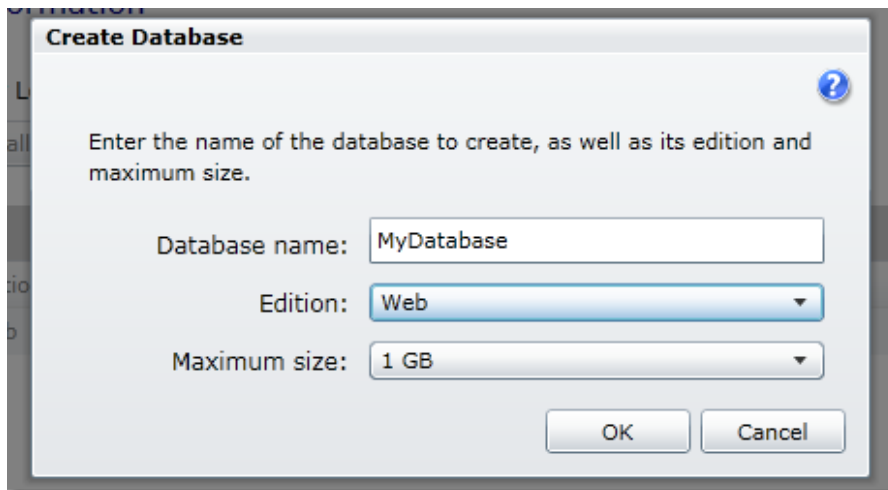


Figura I.9.- Seleccionar tipo de base de datos

Y ya por último, otro tema interesante, es que desde el mismo portal de administración se tiene disponible información interesante sobre cómo trabajar con SQL Azure.

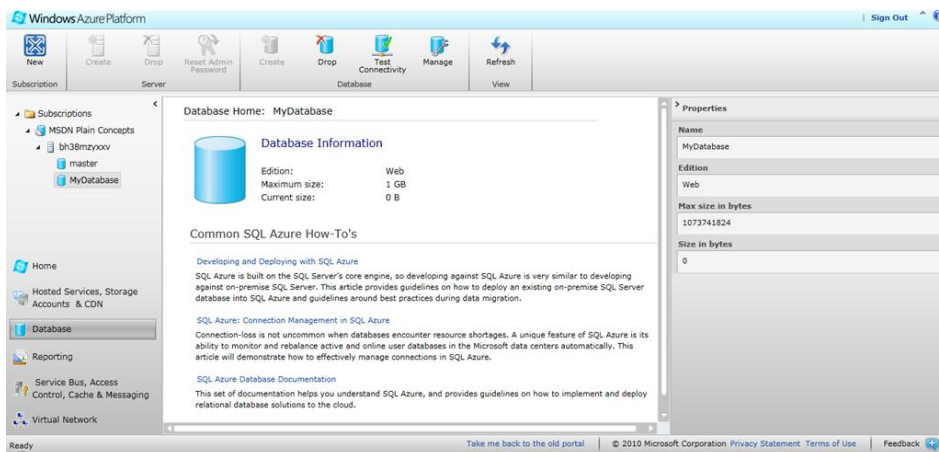


Figura I.10.- Información de SQL Azure

5.-DATABASE MANAGER

Database Manager es una herramienta web hecha en Silverlight, que permite conectarse a una base de datos SQL Azure y realizar las labores típicas que se puede realizar; manejar tablas, vistas, procedimientos almacenados, lanzar queries, abrir y lanzar ficheros SQL.

Database Manager se encuentra completamente integrado dentro del portal de administración de Windows Azure.

Si se selecciona una base de datos se habilitará la opción “Manage” dentro de la barra de herramientas de la parte superior.

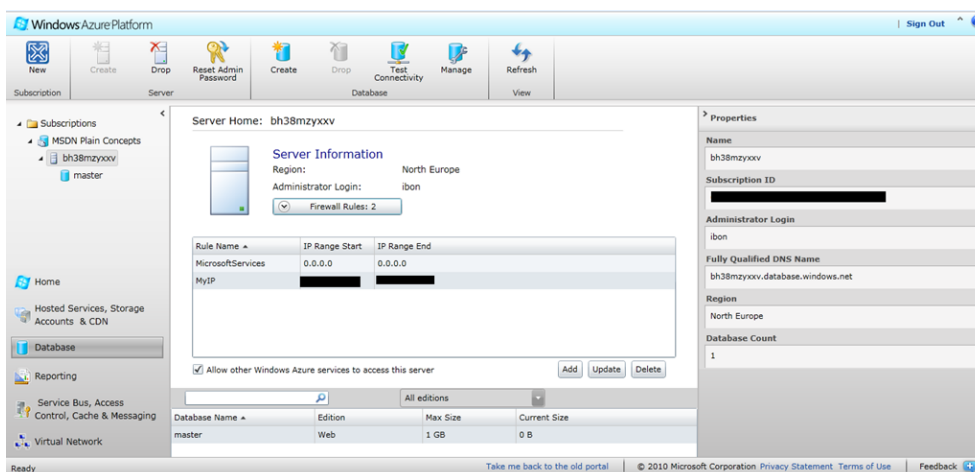


Figura 1.11.- Portal de SQL Azure

Una vez se selecciona la opción “Manage” se abrirá una nueva pestaña del navegador que abrirá una pestaña con Database Manager.

Si es la primera vez que se accederá será necesario aceptar los términos de licencia e indicar las credenciales de acceso a la base de datos.

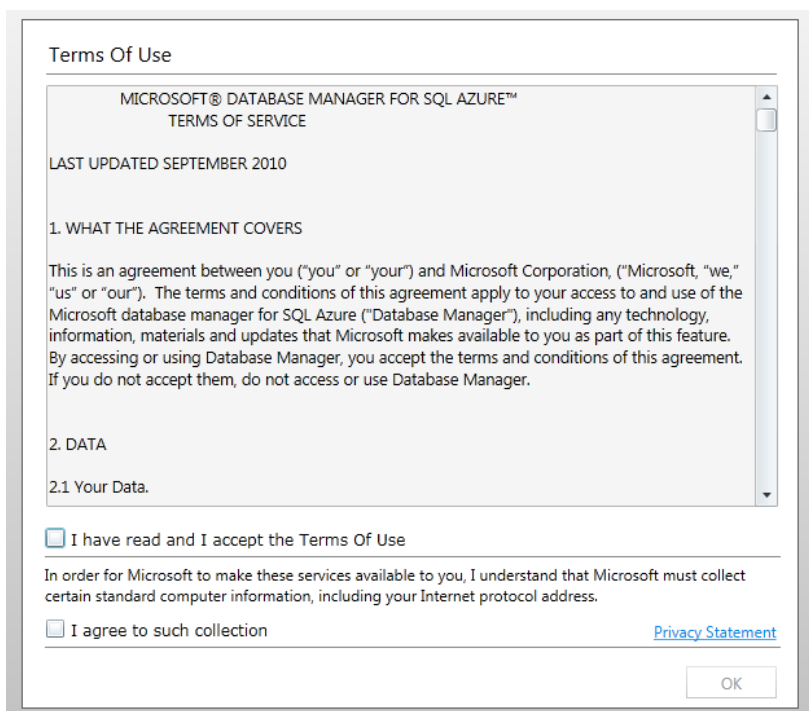


Figura 1.12.- Acuerdo de licencia

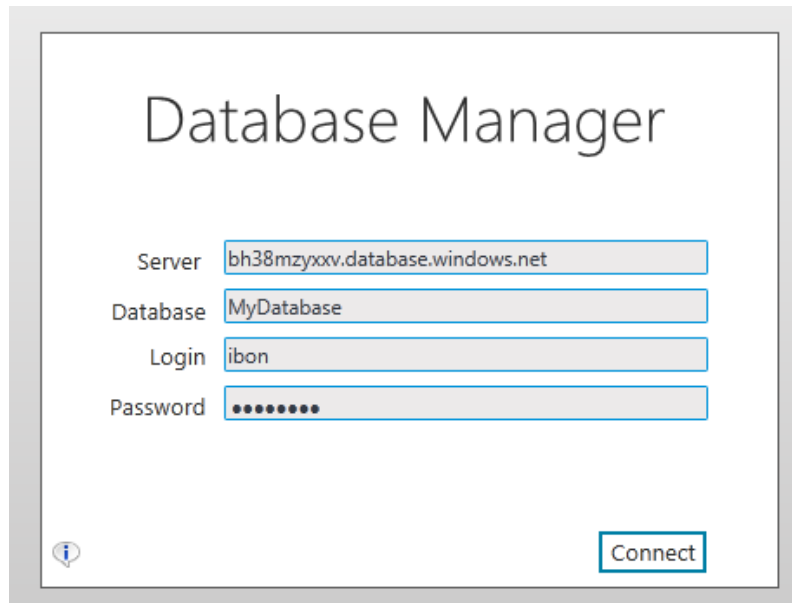


Figura I.13.- Credenciales de acceso

Una vez conectados la herramienta nos permite hacer las labores típicas; manejar tablas, vistas, procedimientos almacenados, lanzar queries, abrir y lanzar ficheros SQL...

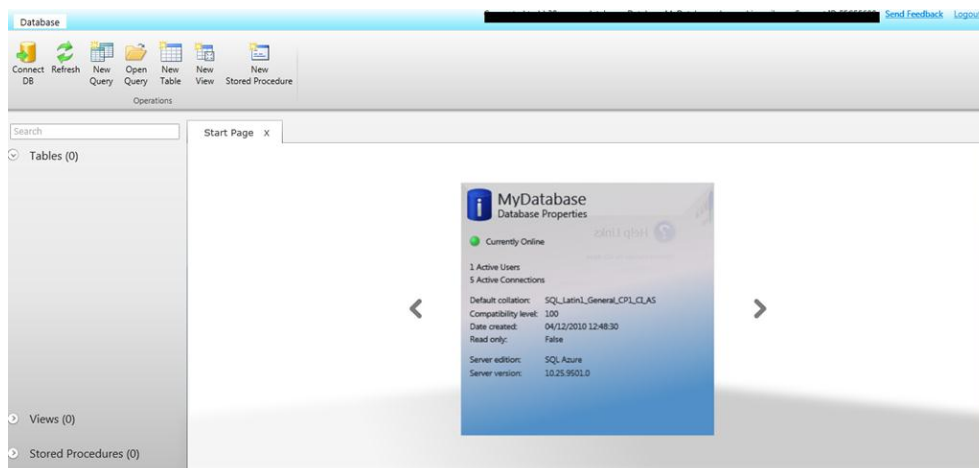


Figura I.14.- Pantalla principal de Database Manager

6.-PARTICULARIDADES DE SQL AZURE RESPECTO A SQL SERVER

SQL Azure, construido sobre SQL Server, permite incluso migrar backend de datos a la nube si tener que tocar ni una sola línea de código de nuestras aplicaciones en un gran número de escenarios. Es cierto que hay ciertas características de SQL Server que SQL Azure no soporta, pero si soporta todas las más usadas:

- Tablas, tablas temporales, vistas, índices, roles, procedimientos almacenados y funciones.
- Consultas complejas y 'joins' entre múltiples tablas.
- Insert, update y delete.

- Restricciones
- Transacciones
- Entre las características no soportadas cabe destacar:
- Transacciones distribuidas
- El broker de mensajes de SQL Server
- Consultas a servidores remotos
- Acceso desde tecnologías antiguas, ya obsoletas, en concreto OleDb.

A la hora de conectar desde nuestras aplicaciones clientes, podemos elegir varios tipos de conexión:

- ADO.NET, incluido Entity Framework.
- Acceso ODBC nativo.
- Soporte para PHP.

6.1.- Soporte T-SQL

Aunque SQL Azure está basado en SQL Server no dispone del soporte de un soporte completo a las sentencias T-SQL de SQL Server. Antes de migrar una aplicación a SQL Azure es conveniente revisar en profundidad las limitaciones para poder determinar la viabilidad de la migración.

Es importante indicar que la evolución de SQL Azure dentro de la plataforma está siendo bastante importante, ya que algunas limitaciones de la misma pueden ir superándose en versiones futuras.

6.2.- Índices cluster

Una peculiaridad interesante respecto a SQL Azure es que todas las tablas necesitar tener obligatoriamente un índice clúster. Los índices clúster se encargan de guardar y ordenar los datos por el valor del índice. Sólo puede haber un índice clúster por tabla porque los datos sólo pueden estar almacenados en un orden.

```
CREATE TABLE SampleTable (Id int NOT NULL IDENTITY, [Name] nvarchar(max),
CONSTRAINT [PK SampleTable] PRIMARY KEY CLUSTERED
(
    [Id] ASC
))
```

Si en el momento de la creación no se especifica un índice clúster SQL Azure no avisará del error pero sí se producirá un error si se intentan insertar datos en una tabla sin índice cluster .

Esta característica afecta únicamente a las tablas "permanentes" de la base de datos, no afectando por tanto a las tablas temporales.

6.3.- Modelo de seguridad

Las bases de datos de SQL Azure pueden contener información confidencial por lo que es esencial para controlar cuidadosamente el acceso. Este aspecto se hace especialmente importante debido a que una base de datos de una cuenta puede compartir el mismo servidor con otras bases de datos de otras cuentas. SQL Azure debe poder ofrecer un nivel de aislamiento adecuado para asegurar la confidencialidad de los datos.

SQL Azure se basa en los mismos mecanismos de seguridad existentes en SQL Server.

- Logins de SQL Server: Acceso autenticado a SQL Azure a nivel de servidor.
- usuarios de base de datos: Permite dar permisos a nivel de base de datos.
- Roles de base de datos: Permite dar permisos a nivel de base de datos a grupos.

6.4.- Tamaño de SQL Azure

Una de las restricciones de SQL Azure es el tamaño. A día de hoy existen dos versiones de SQL Azure; Web Edition y Business Edition. La versión Web Edition tiene un máximo de 5Gb, mientras que la versión Business Edition tiene un máximo de 50 Gb.

En la creación de una base de datos SQL Azure es posible elegir el tamaño de la misma, siempre teniendo en cuenta el máximo permitido. Los rangos que pueden elegirse son 1, 5, 10, 20, 30, 40 y 50 GB.

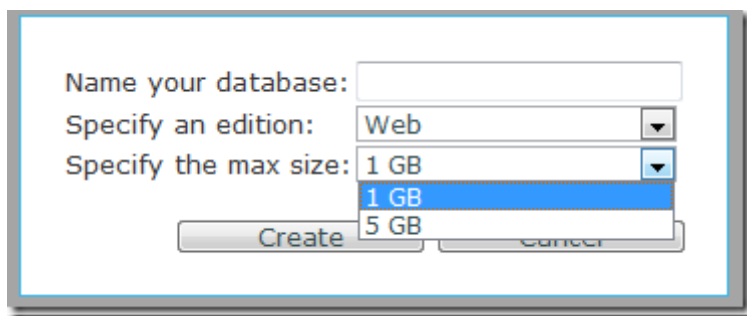


Figura I.15.- Crear base de datos Web Edition

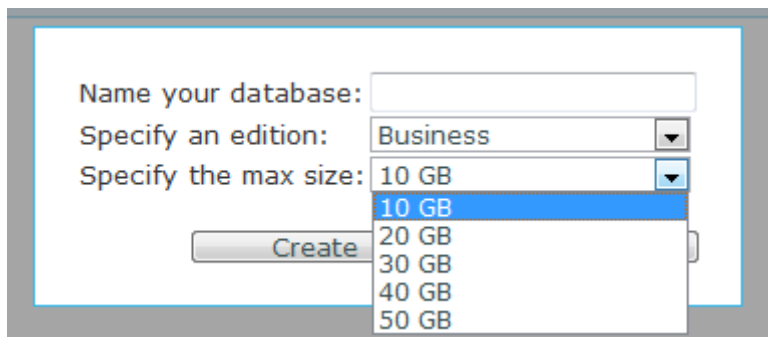


Figura I.16.- Crear una base de datos Business Edition

Este comportamiento puede suscitar múltiples preguntas sobre cómo SQL Azure se comporta al llegar a los límites establecidos. Por ejemplo ¿Qué pasaría si se elige una base de datos de 1 Gb y alcanza 1 Gb de espacio usado? ¿Qué pasaría si se elige una base de datos de 50 Gb y se llega al máximo permitido? ¿Si se crea una base de datos de 50 Gb y sólo se hace uso de 1 Gb? ¿Cómo se realiza la facturación?

El primero punto a tener en cuenta es que sea cuál sea el tamaño elegido, el tamaño de la base de datos se puede modificar, ya sea aumentando tamaño o disminuyéndolo, lógicamente, teniendo en cuenta que el máximo es 50 Gb. Más allá de este tamaño sería necesario crear una segunda base de datos y particionar los datos de la misma.

El segundo aspecto tener en cuenta, es que la facturación no se realiza por el tamaño de la base de datos elegida al crearla. Si crea una base de datos de 50 Gb y se usan 12 Gb, no se factura por una de 50 Gb, sino por una de 20 Gb.

En la facturación se tienen en cuenta los rangos posibles; si la base de datos ocupa 200mb, se pagará por el precio de la de 1 Gb, si la base de datos ocupa 19 Gb se pagará por el precio de la de 20 Gb.

Para saber el rango en el que se encuentra la base de datos y por tanto lo que se factura por el uso, puede lanzarse la siguiente sentencia:

```
SELECT DATABASEPROPERTYEX ('MyDDBB' , 'MaxSizeInBytes' )
```

Tamaño real de la base de datos, que lógicamente siempre será inferior al tamaño devuelto por la sentencia anterior.

```
SELECT SUM(reserved page count) * 8192 FROM sys.dm db partition stats
```

Si se llega al tamaño máximo de la base de datos y se intenta insertar más datos, se mostrará el siguiente error: Msg 40544, Level 20, State 5, Line 1 The database has reached its size quota. Partition or delete data, drop indexes, or consult the documentation for possible resolutions. Code: 524289

Anteriormente comentábamos que una base de datos Azure puede aumentar de tamaño pero no siempre lo puede hacer de manera automática.

Si la base de datos es una versión Web Edition de 1Gb y se necesita aumentar el tamaño, SQL Azure, de manera automática, aumenta la base de datos y la coloca en el siguiente rango, hasta un máximo de 5 Gb.

Si la base de datos es una versión Business Edition, podrá ir aumentando de manera automática en los rangos soportados dentro de esta versión; 1, 5, 10, 20, 30, 40 y 50.

Lo que no se hace de manera automática es el hecho de pasar de una versión Web Edition a una Business Edition. Si es una versión Web Edition y se necesitan más de 5 Gb, la base de datos no aumentará de forma automática a 10 Gb.

SQL Azure no lo hace de manera automática pero sí se puede hacer de forma manual.

```
ALTER DATABASE MyDDBB MODIFY (EDITION='BUSINESS', MAXSIZE=10GB)
```

6.5.- El collation de SQL Azure

Otra de las peculiaridades de SQL Azure está en el collation que emplea. SQL Azure dispone de una configuración a nivel de servidor y base de datos que no puede ser modificada.

El collation del servidor es siempre SQL_Latin1_General_CP1_CI_AS.

El collation de todas las base de datos es siempre SQL_Latin1_General_CP1_CI_AS.

Si se desea comprobar que realmente esa configuración puede lanzarse este comando:

```
SELECT SERVERPROPERTY('Collation') SELECT DATABASEPROPERTYEX('GeeksDDBB', 'Collation')
```

A nivel de servidor o de base de datos no puede cambiarse la configuración, pero sí se podría cambiar a nivel de columna. Al definir una columna es posible establecer un collation diferente al de la base de datos.

```
Campo nvarchar(20) COLLATE SQL_Latin1_General_CP1_CI_AS,
```

Las expresiones usando un collation diferente también están permitidas, como por ejemplo:

```
SELECT LastName FROM Person.Person ORDER BY LastName COLLATE Traditional_Spanish_ci_ai ASC;
```

6.6.- DMVs

Desde su aparición en SQL Server 2005 las DMVs han sido una herramienta muy necesaria para poder detectar y solucionar problemas de rendimiento en SQL Server.

SQL Azure está basado en SQL Server, pero muchas de las DMVs no están disponibles en esta primera versión de Azure.

El principal problema por el cuál no están soportadas todas las DMVs es porque en SQL Server éstas funcionan a nivel de instancia mientras que en SQL Azure tendrían que funcionar a nivel de base de datos.

Un ejemplo clarificador puede ser el permiso que hay que tener para trabajar con DMVs. En SQL Azure es VIEW DATABASE STATE, mientras que en SQL Server es VIEW SERVER STATE.

A continuación se muestra algún ejemplo de algunas acciones disponibles en SQL Azure.

Identificar los planes de ejecución que provocan excesivas recompilaciones:

```
select top 25 SQL text.text, SQL handle, plan generation num,
      execution count, dbid, objectid      from sys.dm_exec_query_stats a cross apply
      sys.dm_exec_sql_text(SQL handle) as SQL text where plan generation num >1
order by plan_generation_num desc}
```

Identificar planes de ejecución ineficientes:

```
select highest cpu queries.plan handle,
      highest cpu queries.total worker time, q.dbid, q.objectid, q.number,
      q.encrypted, q.[text]      from (select top 50 qs.plan_handle,
      qs.total_worker_time from sys.dm_exec_query_stats qs order by
      qs.total_worker_time desc)      as highest_cpu_queries cross apply
      sys.dm_exec_sql_text(plan handle) as q order by
      highest cpu queries.total worker time desc
```

7.-ACCESO PROGRAMÁTICO A SQL AZURE

Conectarse desde una aplicación a SQL Azure puede llegar a realizarse sin tener que modificar una sólo línea de código, ya en muchas ocasiones el único cambio que será necesario realizar será el de cambiar la cadena de conexión.

```
<connectionStrings>
  <add name="SQLAzureConnection" connectionString="Data Source="
  <ProvideServerName>
    .database.windows.net;Initial Catalog=TestDb;User ID=<ProvideUserName>
    ;Password=<ProvidePassword>;Encrypt=true;Trusted_Connection=false;"
    providerName="System.Data.SqlClient" />
</connectionStrings>
```

Las aplicaciones clientes pueden conectarse a través de los siguientes protocolos:

- ADO.NET, incluido Entity Framework.
- Acceso ODBC nativo.
- SQL Azure incluye soporte para PHP.

7.1.- Conectarse desde ASP.NET

Trabajando con SQL Azure se pueden realizar los bindings entre la base de datos de Azure y los controles de ASP.NET del mismo modo en que se trabaja con SQL Server.

Es importante tener en cuenta que en fase de desarrollo no es conveniente trabajar directamente contra una base de datos de SQL Azure, ya que aparte de la latencia de la red, habría que tener en cuenta el coste que supondría.

La clase `SQLDataSource` es perfectamente compatible con SQL Azure por lo que pueden convertirse aplicaciones con sólo cambiar la cadena de conexión.

```
<asp:SQLDataSource ID="SQLAzureDataSource" runat="server"
  ConnectionString="<%$ ConnectionStrings:SQLAzureConnection %>"
  InsertCommand="INSERT INTO [Table1] ([Col1], [Col2]) VALUES (@Col1, @Col2)"
  SelectCommand="SELECT * FROM [Table1]"
  UpdateCommand="UPDATE [Table1] SET [Col2] = @Col2 WHERE [Col1] = @Col1"
  DeleteCommand="DELETE FROM [Table1] WHERE [Col1] = @Col1">
  <UpdateParameters>
    <asp:Parameter Name="Col2" Type="String" />
    <asp:Parameter Name="Col1" Type="Int32" />
  </UpdateParameters>
  <InsertParameters>
    <asp:formParameter Name="Col1" FormField="TextBox1" />
    <asp:formParameter Name="Col2" FormField="TextBox2" />
  </InsertParameters>
  <DeleteParameters>
    <asp:Parameter Name="Col1" Type="Int32" />
  </DeleteParameters>
</asp:SQLDataSource>
```

7.2.- Conectarse desde Entity Framework

Como ya se ha comentado anteriormente Entity Framework es completamente compatible con SQL Azure. Por lo tanto, la migración de una aplicación que emplee esta tecnología, a nivel de acceso a datos, podría ser tan simple como cambiar la cadena de conexión en el fichero de configuración:

```
<connectionStrings>
  <add name="SchoolEntities" connectionString="metadata=res://*/SchoolDataModel.csdl|
res://*/SchoolDataModel.ssdl|res://*/SchoolDataModel.msl;provider=System.Data.SqlClient;
provider connection string="Data Source=<provideServerName>.database.windows.net;
Initial Catalog=School;Integrated Security=False;UserID=<provideUserID>;
Password=<providePassword>;MultipleActiveResultSets=False;Encrypt=True;
TrustServerCertificate=False";" providerName="System.Data.EntityClient"/>
</connectionStrings>
```

Por lo tanto, tecnologías en las que Entity Framework suele ser un elemento fundamental, como WCF Data Services, WCF RIA Services o ASP.NET Dynamic Data será perfectamente compatibles en cuanto al acceso a datos se refiere.

7.3.- Conectarse desde PHP

SQL Azure permite que aplicaciones realizadas en PHP puedan conectarse empleando el driver de SQL Server para PHP. Un aplicación PHP se incluye en Windows Azure como un Web CGI Role.

Después de crear la aplicación con Web CGI Role es necesario configurar el handle para FastCGI. Dentro del fichero de configuración dentro de la sección `<handlers>` de `<system.webServer>` habrá que añadir lo siguiente:

```
<add name="PHP_FastCGI"
      verb="*"
      path="*.php"
      scriptProcessor="%RoleRoot%\approot\php-cgi.exe"
      modules="FastCgiModule"
      resourceType="Unspecified" />
```

En el fichero web.roleconfig también será necesario añadir la siguiente sección:

```
<application fullPath="%RoleRoot%\approot\php-cgi.exe"/
```

Después, se podría añadir un nuevo fichero con extensión php y en sus propiedades establecer la propiedad Content a Copy Always.

Y dentro del fichero php se podría escribir un código como el siguiente, completamente compatible con SQL Azure.

```
<?php

$serverName = "tcp:ProvideServerName.database.windows.net,1433";
$username = 'ProvideUserName@ProvideServerName';
$password = 'ProvidePassword';
$dbName = "TestDB";
$table = "tablePHP";

$connectionInfo = array("Database"=>$dbName, "UID"=>$username, "PWD"=>$password,
"MultipleActiveResultSets"=>true);

SQLsrv_configure('WarningsReturnAsErrors', 0);
$conn = SQLsrv_connect( $serverName, $connectionInfo);
if($conn === false)
{
    FatalError("Failed to connect...");
}

CreateTable($conn, $table, "Col1 int primary key, Col2 varchar(20)");

$stmtSQL = "INSERT INTO [$table] (Col1, Col2) VALUES (1, 'string1'), (2, 'string2')";
$stmt = SQLsrv_query($conn, $stmtSQL);
if ($stmt === false)
{
    FatalError("Failed to insert data into test table: ".$stmtSQL);
}
SQLsrv_free_stmt($stmt);

$stmtSQL = "SELECT Col1, Col2 FROM [$table]";
$stmt = SQLsrv_query($conn, $stmtSQL);
if ($stmt === false)
{
    FatalError("Failed to query test table: ".$stmtSQL);
}
else
{
    while($row = SQLsrv_fetch_array($stmt, SQLSRV_FETCH_NUMERIC))
    {
        echo "Col1: ".$row[0]."\n";
        echo "Col2: ".$row[1]."\n";
    }

    SQLsrv_free_stmt($stmt);
}

SQLsrv_close($conn);

function CreateTable($conn, $tableName, $dataType)
```

```

{
  $SQL = "CREATE TABLE [$tableName] ($dataType)";
  DropTable($conn,$tableName);
  $stmt = SQLSRV query($conn, $SQL);
  if ($stmt === false)
  {
    FatalError("Failed to create test table: ".$SQL);
  }
  SQLSRV free stmt($stmt);
}

function DropTable($conn, $tableName)
{
  $stmt = SQLSRV query($conn, "DROP TABLE [$tableName]");
  if ($stmt === false)
  {
  }
  else
  {
    SQLSRV free stmt($stmt);
  }
}

function FatalError($errorMsg)
{
  Handle Errors();
  die($errorMsg."\n");
}

function Handle Errors()
{
  $errors = SQLSRV errors(SQLSRV_ERR_ERRORS);
  $count = count($errors);
  if($count == 0)
  {
    $errors = SQLSRV errors(SQLSRV_ERR_ALL);
    $count = count($errors);
  }
  if($count > 0)
  {
    for($i = 0; $i < $count; $i++)
    {
      echo $errors[$i]['message']."\n";
    }
  }
}

?>

```

8.-ADMINISTRACIÓN DE SQL AZURE

Las herramientas cliente de SQL Server 2008 R2 incluyen la compatibilidad con SQL Azure. Será esta herramienta la que habitualmente se deba emplear para realizar las labores típicas de administración.

SQL Server Management Studio 2008 R2 cuenta con soporte completo para SQL Azure en términos de conectividad: Ver objetos en el explorador de objetos, scripting, SMO etc...

8.1.- Migración de SQL Server a SQL Azure

Una de las acciones más habituales si se desea trabajar con SQL Azure es ser capaz de migrar una base de datos SQL Server a SQL Azure.

Existen diversas maneras de llevar a cabo esta tarea. Una de las opciones puede ser la de generar un script de datos en la base de datos origen que contenga todo lo necesario para crear la base de datos en Azure.

Para poder crear un script de datos de una base de datos SQL Server es necesario tener SQL Server Management Studio. Desde la versión 2008 R2 de la herramienta se dispone de la compatibilidad con SQL Azure, con lo que posibilidad poder crear scripts de datos que sean compatibles con la plataforma.

A continuación se muestra de forma gráfica el proceso:

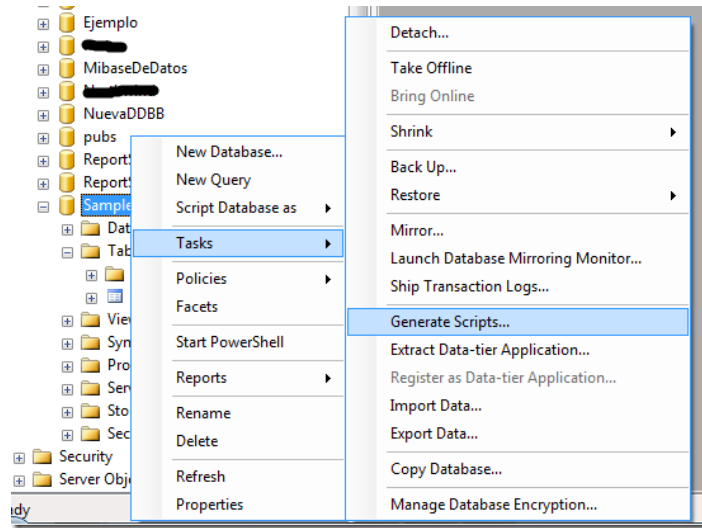


Figura 1.17.- Opción generar scripts

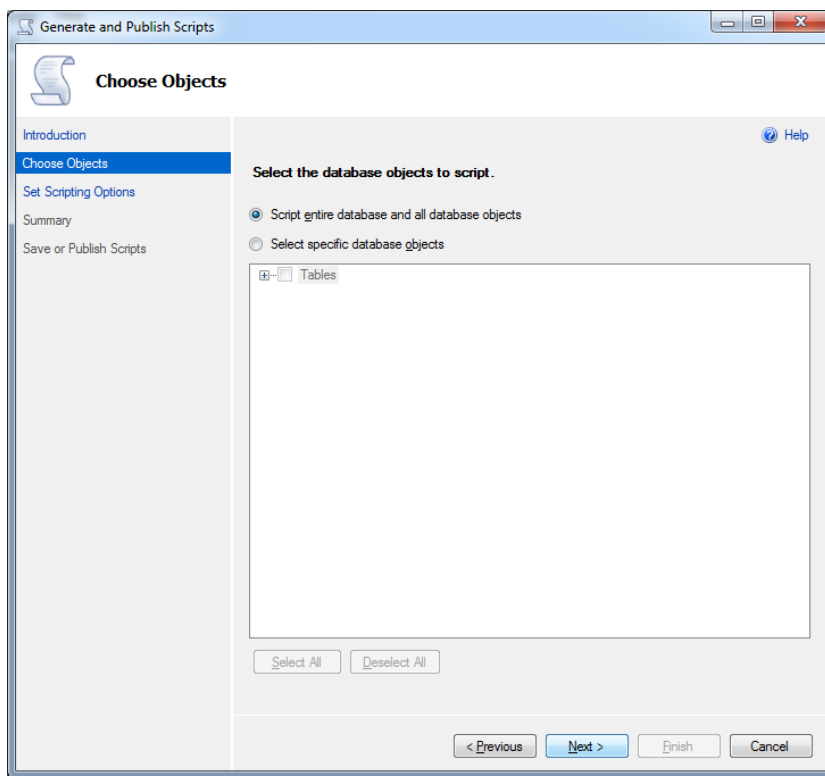


Figura 1.18.- Seleccionar los objetos a exportar

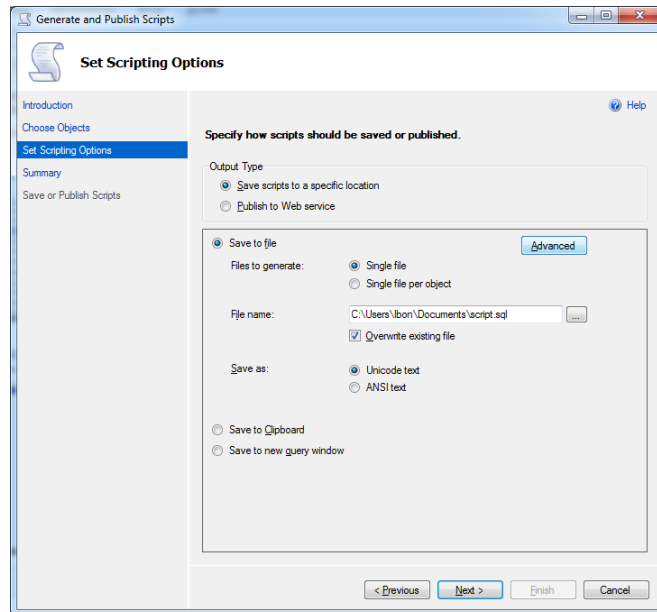


Figura 1.19.- Destino del fichero generado

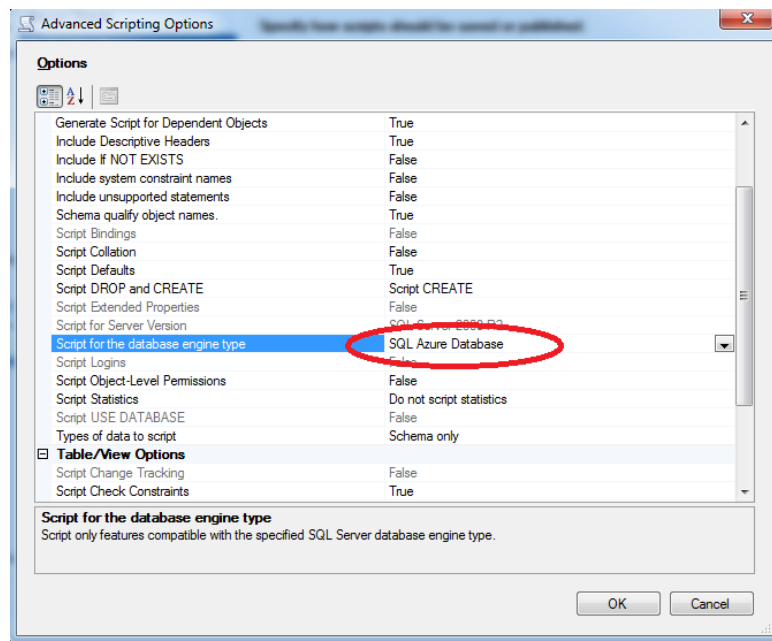


Figura 1.20.- Indicar que el script debe ser compatible con SQL Azure

El proceso generará un script con la información que se haya configurado. Una vez creado el script, también desde la herramienta SQL Server Management Studio será necesario conectarse a la base de datos SQL Azure y lanzar el script para terminar el proceso de migración.

```

/***** Object: Table [dbo].[SampleTable] Script Date: 06/20/2010
20:10:51 *****/
SET ANSI NULLS ON GO SET QUOTED IDENTIFIER ON GO CREATE
TABLE [dbo].[SampleTable]( [MyRowID] [int] NOT NULL, PRIMARY KEY CLUSTERED (
[MyRowID] ASC )WITH (STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF) )
GO

```

8.2.- Conectarse desde SQL Server Management Studio

A continuación se muestra el proceso que a seguir para poder conectarse a una base de datos de SQL Azure empleando SQL Server Management Studio 2008 R2. Hay que tener en cuenta que el proceso puede variar ligeramente en función de la versión de SQL Server Management Studio que se utilice.

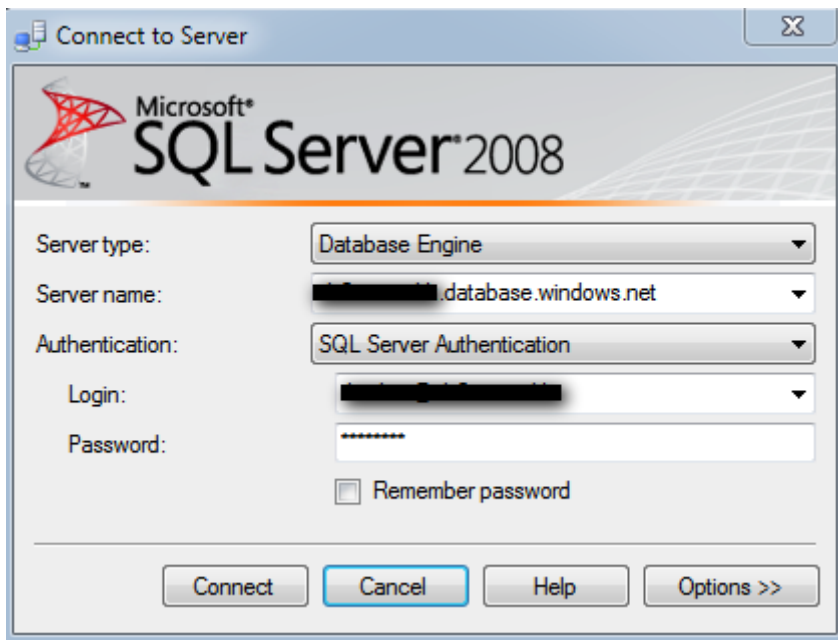


Figura 1.21.- Pantalla de conexión a SQL Azure

La entrada debe estar en el formato: usuario@nombre de servidor

Una aspecto clave a tener en cuenta trabajando con SQL Azure es conocer que el comando USE no está permitido. Cuando se realiza una conexión a SQL Azure, la conexión se debe hacer contra una base de datos concreta. Una vez hecha la conexión, no es posible cambiar la base de datos sobre la que se está trabajando, salvo creando una nueva conexión.

Si necesita conectarse a una base de datos específica, hay que introducir el nombre de base de datos en la pestaña "Conectar a base de datos".

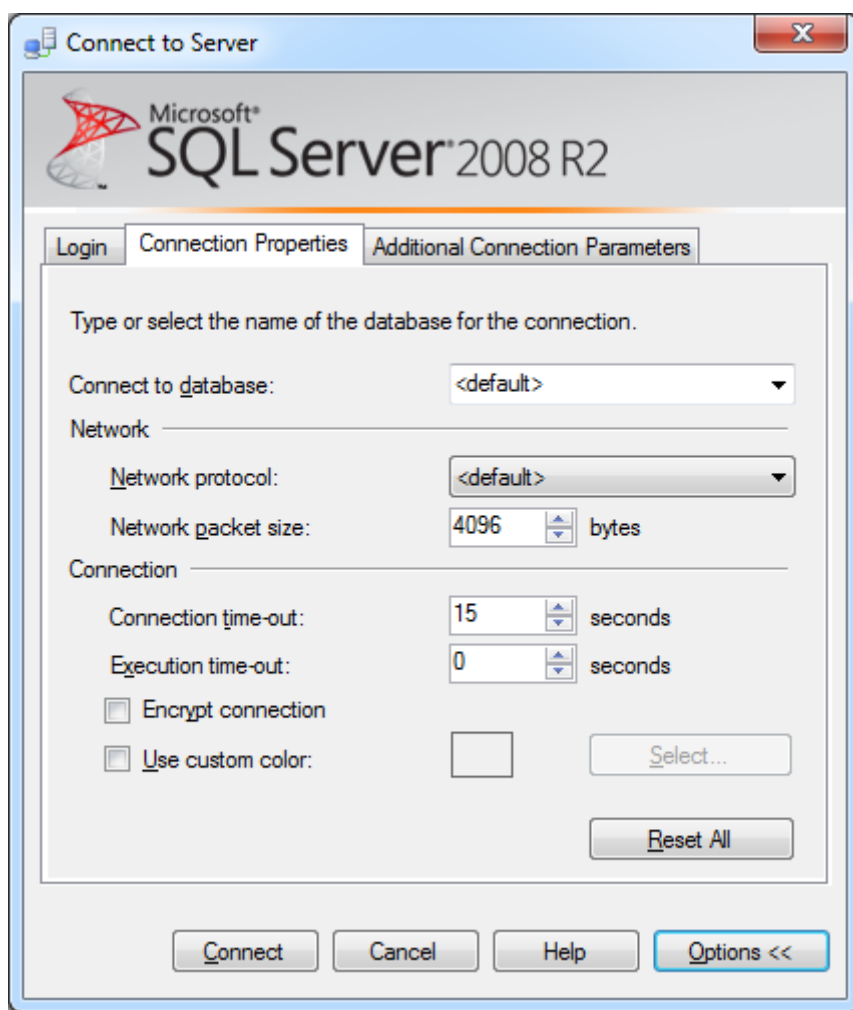


Figura 1.22.- Seleccionar la base de datos de SQL Azure

Debe tenerse en cuenta que antes de realizar la conexión ha habido que configurar correctamente las reglas del firewall para permitir la conexión desde la IP desde la cual realiza la conexión. Si se encuentra dentro de una red empresarial, hay que asegurarse de que el firewall corporativo permita las conexiones TCP por el puerto 1433.

Una vez realizada esta acción y si las credenciales de acceso son correctas, ya se dispondrá de una conexión a la base de datos SQL Azure seleccionada, pudiendo hacer las operaciones que se necesiten.

8.3.- Conectarse a SQL Azure usando SQLcmd

Otra alternativa para poder conectarse a SQL Azure es utilizar la herramienta de línea de comandos SQLCMD disponible con SQL Server.

SQLCMD permite conectarse a una base de datos SQL Azure, lanzar sentencias SQL, procedimientos almacenados e incluso ficheros de script con sentencias T-SQL.

SQLCMD no se incluye dentro de la instalación base de SQL Server o de las herramientas clientes. Se instala junto SQL Server 2008 R2 Feature Pack.

A continuación se muestra un ejemplo de cómo crear una tabla empleando SQLCMD.

```
C:\>SQLcmd -U <login@Server> -P <Password> -S <ProvideServerName> -d <ProvideServerName>
1> CREATE TABLE table1 (Col1 int primary key, Col2 varchar(20));
2> GO
3> QUIT
```

8.4.- SQL Azure Migration Wizard

En el primer punto de este contenido se describe el proceso de migración de SQL Server a SQL Azure generando un script de datos.

Otra alternativa disponible es la herramienta SQL Azure Migration Wizard. Se trata de una herramienta que permite migrar una base de datos a SQL Azure de una manera cómoda y sencilla.

La herramienta ofrece diferentes opciones que puede simplificar las tareas de aprovisionamiento de SQL Azure:

- Analizar una base de datos para ver si es compatible con SQL Azure,
- Generar scripts de datos, haciendo uso del comando BCP.
- Permite la conexión entre SQL Server-SQL Azure, SQL Azure-SQL Server y entre SQL Azure-SQL Azure.
- Es capaz de analizar trazas del SQL Profiler y scripts con T-SQL para detectar posibles incompatibilidades a la hora de migrar a SQL Azure.

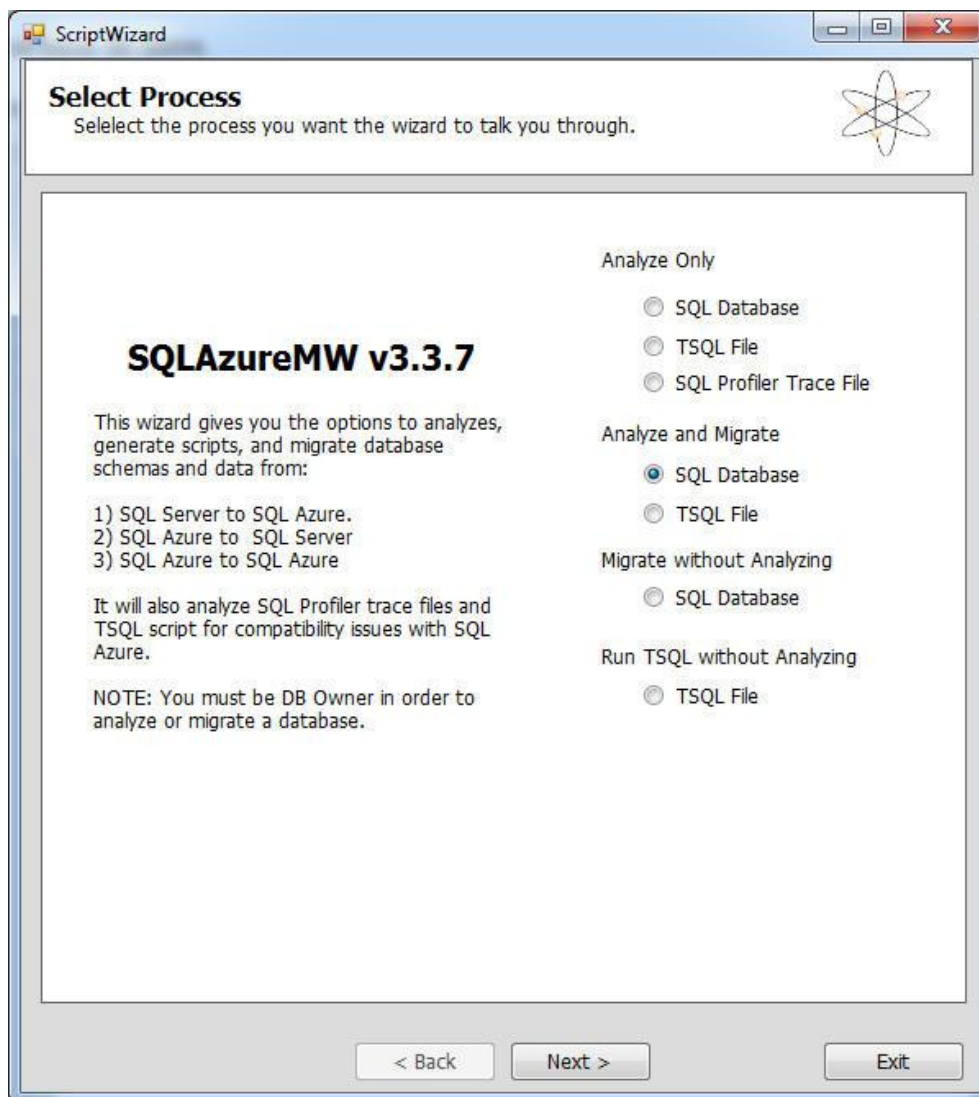


Figura 1.23.- Opciones disponibles en SQL Azure Migration Wizard

9.-SOBRE EL TAMAÑO DE SQL AZURE

Anteriormente se ha comentado en varias ocasiones las diferentes versiones y tamaños que puede tener SQL Azure.

A día de hoy existen dos versiones de SQL Azure; Web Edition y Business Edition. La versión Web Edition tiene un máximo de 5Gb, mientras que la versión Business tiene un máximo de 50 Gb.

Los rangos que se pueden elegir para el tamaño de la base de datos son 1, 5, 10, 20, 30, 40 y 50 GB.

¿Qué pasa si se elige una base de datos de 1 Gb y se llega a 1 Gb de espacio usado? ¿Qué pasa si se elige una base de datos de 50 Gb y se necesita más? ¿Si se tiene una base de datos de 50 Gb y sólo se usa 1 Gb? ¿Cuánto se paga?

Nota: Durante la realización de este contenido Microsoft ha anunciado oficialmente una nueva versión de SQL Azure antes de finales de 2011, versión donde el tamaño máximo se implicará a 150 GB.

Create Database

Enter the name of the database to create, as well as its edition and maximum size.

Database name:

Edition:

Maximum Size:

- 10 GB
- 20 GB
- 30 GB
- 40 GB
- 50 GB

Figura 1.24.- Crear base de datos

Lo primero, es que sea como cuál sea el tamaño elegido, el tamaño de la base de datos se puede modificar, ya se aumentando tamaño o disminuyéndolo, teniendo en cuenta que el máximo es 50 Gb. Más allá de este tamaño sería necesario crear una segunda base de datos y particionar los datos de la misma.

Otro tema a tener en cuenta, es que la facturación no se hace por el tamaño de la base de datos elegida al crear la base de datos. Si se elige una base de datos de 50 Gb y se ocupan 12 Gb, no se paga por una de 50, sino por una de 20.

A la hora de facturar se tienen en cuenta los rangos posibles; si la base de datos ocupa 200mb, se pagará por el precio de la de 1 Gb, si la base de datos ocupa 19 Gb se pagará por el precio de la de 20 Gb.

Si se quiere saber el rango en el que se encuentra la base de datos y por tanto lo que se pagará, se puede lanzar la siguiente sentencia:

```
SELECT DATABASEPROPERTYEX ('MyDDBB' , 'MaxSizeInBytes' )
```

Para conocer el tamaño real de la base de datos, que lógicamente siempre será inferior al tamaño devuelto por la sentencia anterior.

```
SELECT SUM(reserved_page_count) * 8192 FROM sys.dm_db_partition_stats
```

Si se llega al tamaño máximo de la base de datos y se intenta insertar más datos, se mostrará el siguiente error:

Msg 40544, Level 20, State 5, Line 1

The database has reached its size quota. Partition or delete data, drop indexes, or consult the documentation for possible resolutions. Code: 524289

Si se necesita controlar el código de error en nuestra aplicación C#, el código devuelto es el 40544.

Si la base de datos es una versión Web Edition de 1Gb y se necesita más, de manera automática SQL Azure aumenta la base de datos y la coloca en el siguiente rango, hasta un máximo de 5 Gb.

Si la base de datos es una versión Business Edition, podrá ir aumentando de manera automática en los rangos soportados dentro de esta versión; 1, 5, 10, 20, 30, 40 y 50.

Lo que no se hace de manera automática es el hecho de pasar de una versión Web Edition a una Business Edition. Si es una versión Web Edition y se necesita más de 5 Gb, la base de datos no aumentará de forma automática a 10 Gb.

No se hace de manera automática, pero SÍ se puede hacer de forma manual.

```
ALTER DATABASE MyDDBB MODIFY (EDITION='BUSINESS', MAXSIZE=10GB)
```

10.-DMV

Desde su aparición en SQL Server 2005 las DMVs han sido una herramienta muy necesaria para poder detectar y solucionar problemas de rendimiento en SQL Server.

SQL Azure está basado en SQL Server, pero desgraciadamente muchas de las DMVs no está disponibles en esta primera versión de Azure...aunque sí que quieren ir añadiéndolas en sucesivas versiones.

El principal problema por el cuál no están soportadas todas las DMVs es porque en SQL Server éstas funcionan a nivel de instancia, cosa que en SQL Azure no es posible, tendrían que funcionar a nivel de base de datos. Este trabajo es el que deben hacer para que todas las DMVs estén accesibles también en SQL Azure.

Por ejemplo, el permiso que hay que tener para trabajar con DMVs en Azure es VIEW DATABASE STATE, mientras que en SQL Server es VIEW SERVER STATE).

Microsoft ha publicado un documento bastante interesante dónde se comentan las diferentes DMVs disponibles en esta primera versión de SQL

Identificar los planes de ejecución que provocan excesivas recompilaciones.

```
select top 25
    SQL_text.text,
    SQL_handle,
    plan_generation_num,
    execution count,
    dbid,
    objectid
from
    sys.dm_exec_query_stats a
    cross apply sys.dm_exec_sql_text(SQL_handle) as SQL_text
where
    plan_generation_num >1
order by plan_generation_num desc
```

Identificar planes de ejecución ineficientes:

```

select
    highest cpu queries.plan handle,
    highest cpu queries.total worker time,
    q.dbid,
    q.objectid,
    q.number,
    q.encrypted,
    q.[text]
from
    (select top 50
        qs.plan_handle,
        qs.total_worker_time
    from
        sys.dm_exec_query_stats qs
    order by qs.total_worker_time desc) as highest cpu queries
cross apply sys.dm_exec_sql_text(plan_handle) as q
order by highest_cpu_queries.total_worker_time desc

```

11.-EXPONER POR ODATA EL CONTENIDO DE SQL AZURE

Dentro de los SQL Azure Labs se pueden encontrar, a modo de preview, algunas de las funcionalidades que llegarán pronto a SQL Azure.

Entre una de esas funcionalidades está la opción de poder exponer por OData el contenido de una base de datos SQL Azure.

OData es un protocolo basado en REST cuyo objetivo principal es buscar la interoperabilidad entre las aplicaciones para el intercambio de datos, independientemente de la tecnología.

Para el ejemplo se parte de un servidor de SQL Azure creado desde una cuenta de Azure, desde el portal de producción de Windows Azure. Después de crear el servidor se han dado permisos en el firewall para que puedan conectarse aplicaciones que residan en la nube de Microsoft.

Dentro del servidor también se ha creado una base de datos con diferentes tablas y datos.

Server Home: p2gnm147tb

Server Information
 Region: West Europe
 Administrator Login: ibon
 Firewall Rules: 1

Rule Name	IP Range Start	IP Range End
MicrosoftServices	0.0.0.0	0.0.0.0

Allow other Windows Azure services to access this server

Database Name	Edition	Max Size	State
master	Web	1 GB	Online

Figura 1.25.- Configuración firewall

Una vez que se tiene la base de datos creada en un servidor de Azure, ya se puede ir al SQL Azure Labs y exponer la información por OData. Para ello el primer paso que se tiene que realizar es conectarse al servidor y seleccionar la base de datos sobre la que se desea realizar la operación, como se muestra a continuación:

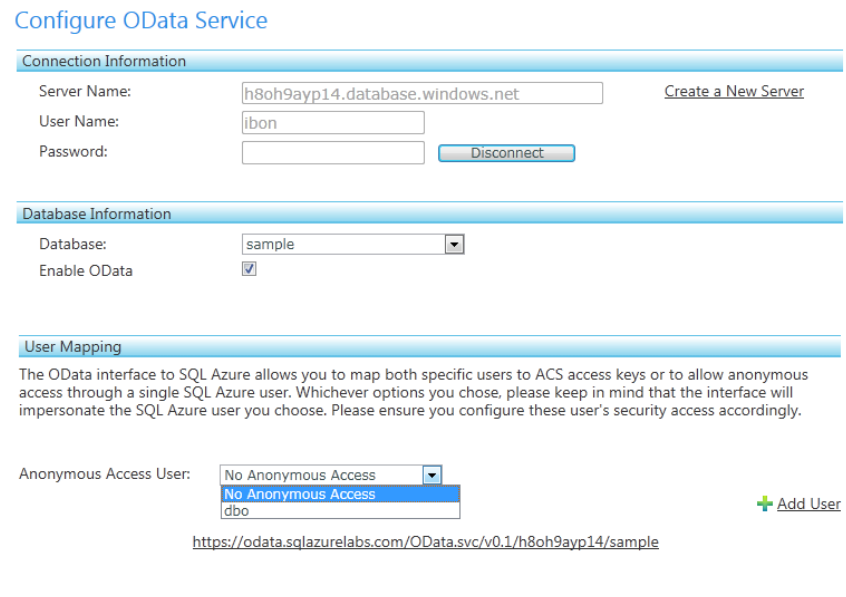


Figura I.26.- Configuración servicio de OData

A través de la URI que te proporciona se tendría expuestas por OData las tablas de la base de datos. El acceso a esta información se puede securizar haciendo uso de Access Control.

Si se pone un ejemplo de una petición REST en el navegador, se vería algo como esto.

Lógicamente, también se puede consultar la información desde una aplicación, esté hecha con .NET o con otra tecnología.

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
- <feed xml:base="https://odata.sqlazurelabs.com/OData.svc/v0.1/h8oh9ayp14/sample/" xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata" xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata" xmlns="http://www.w3.org/2003/01/XMLSchema-instance">
  <title type="text">SampleTables</title>
  <id>https://odata.sqlazurelabs.com/OData.svc/v0.1/h8oh9ayp14/sample/SampleTables/</id>
  <updated>2010-10-04T18:54:31Z</updated>
  <link rel="self" title="SampleTables" href="SampleTables" />
- <entry>
  <id>https://odata.sqlazurelabs.com/OData.svc/v0.1/h8oh9ayp14/sample/SampleTables(1)</id>
  <title type="text" />
  <updated>2010-10-04T18:54:31Z</updated>
  <author>
    <name />
  </author>
  <link rel="edit" title="SampleTable" href="SampleTables(1)" />
  <category term="sample.SampleTable" scheme="http://schemas.microsoft.com/ado/2007/08/dataservices" />
  <content type="application/xml">
    <m:properties>
      <d:MyRowID m:type="Edm.Int32">1</d:MyRowID>
    </m:properties>
  </content>
  </entry>
- <entry>
  <id>https://odata.sqlazurelabs.com/OData.svc/v0.1/h8oh9ayp14/sample/SampleTables(2)</id>
  <title type="text" />
  <updated>2010-10-04T18:54:31Z</updated>
  <author>
```

Figura I.27.- Ejemplo de información

12.- HERRAMIENTAS PARA TRABAJAR CON SQL AZURE

12.1.- SQL Azure Migration Wizard

SQL Azure Migration Wizard es una herramienta esencial, disponible en CodePlex, para trabajar con SQL Azure, ya que permite migrar una base de datos a SQL Azure de una manera cómodo y sencilla.

La herramienta ofrece diferentes opciones que puede simplificar las tareas de aprovisionamiento de SQL Azure:

- Analizar una base de datos para ver si es compatible con SQL Azure,
- Generar scripts de datos, haciendo uso del comando BCP.
- Permite la conexión entre SQL Server-SQL Azure, SQL Azure-SQL Server y entre SQL Azure-SQL Azure.
- Es capaz de analizar trazas del SQL Profiler y scripts con T-SQL para detectar posibles incompatibilidades a la hora de migrar a SQL Azure.

A continuación se muestra un ejemplo del proceso de migración de una base de datos SQL Server a SQL Azure. El primer paso es seleccionar la operación que se desea realizar.

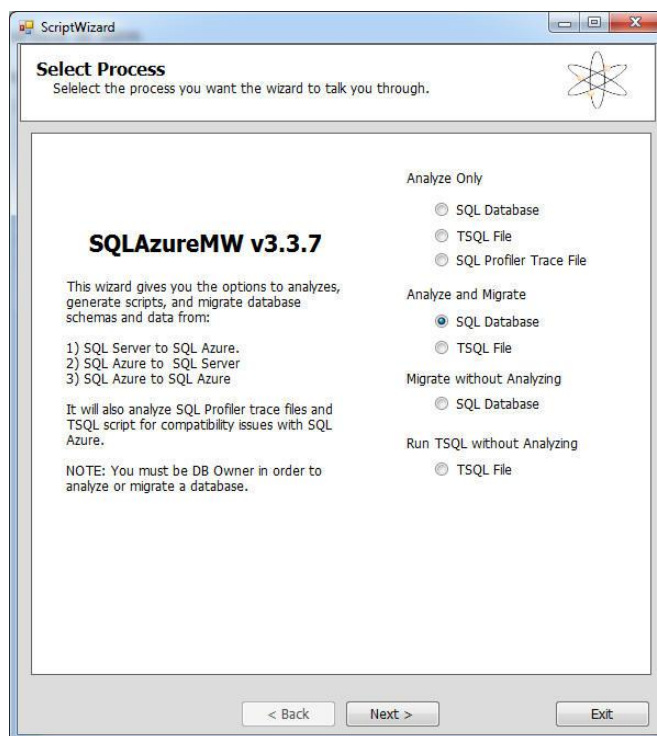


Figura 1.28.- Opciones disponibles en SQL Azure Migration Wizard

Una vez seleccionada la operación de analizar y migrar una base de datos SQL Server, se deberá indicar la conexión al servidor de base de datos dónde se encuentra la base de datos a migrar.

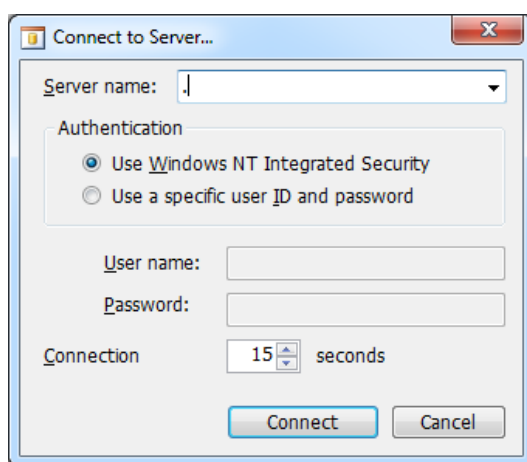


Figura 1.29.- Conectarse a SQL Server

Seleccionar la base a datos a migrar.

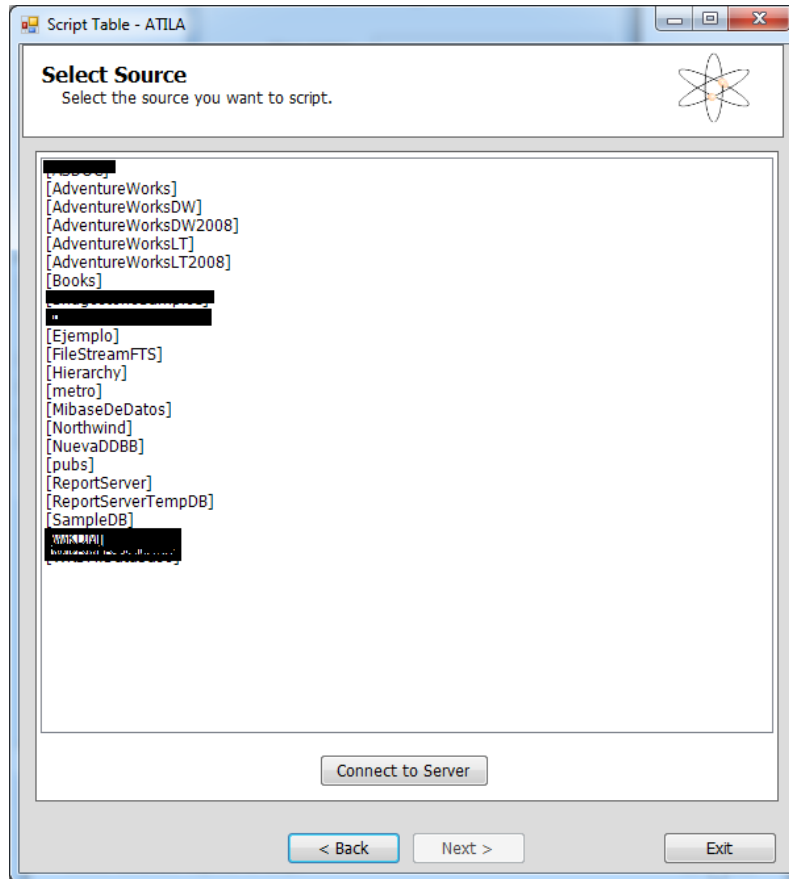


Figura 1.30.- Seleccionar la base de datos a migrar

Seleccionar los objetos que se desean migrar.

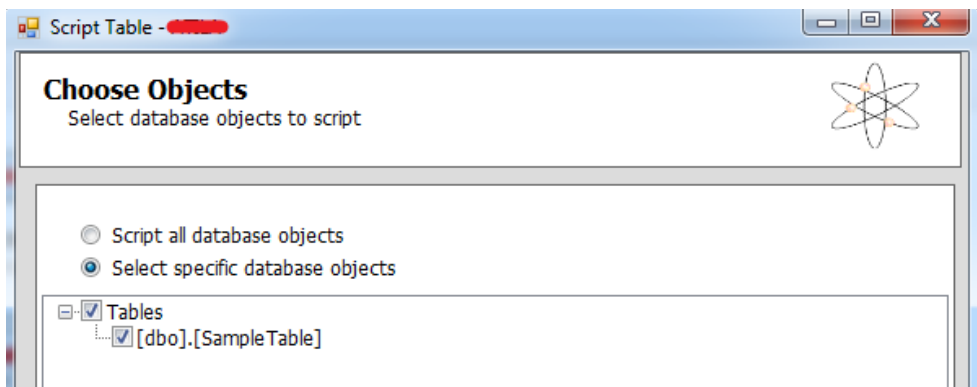


Figura 1.31.- Selección de los objetos de SQL Server a migrar

Una vez terminado el proceso, la herramienta se encargará de analizar la base de datos y detectar posibles incompatibilidades.

Si todo es correcto, generará lo necesario para migrar la base de datos a SQL Azure, tanto el schema como los datos que contenga, si es que así se ha configurado.

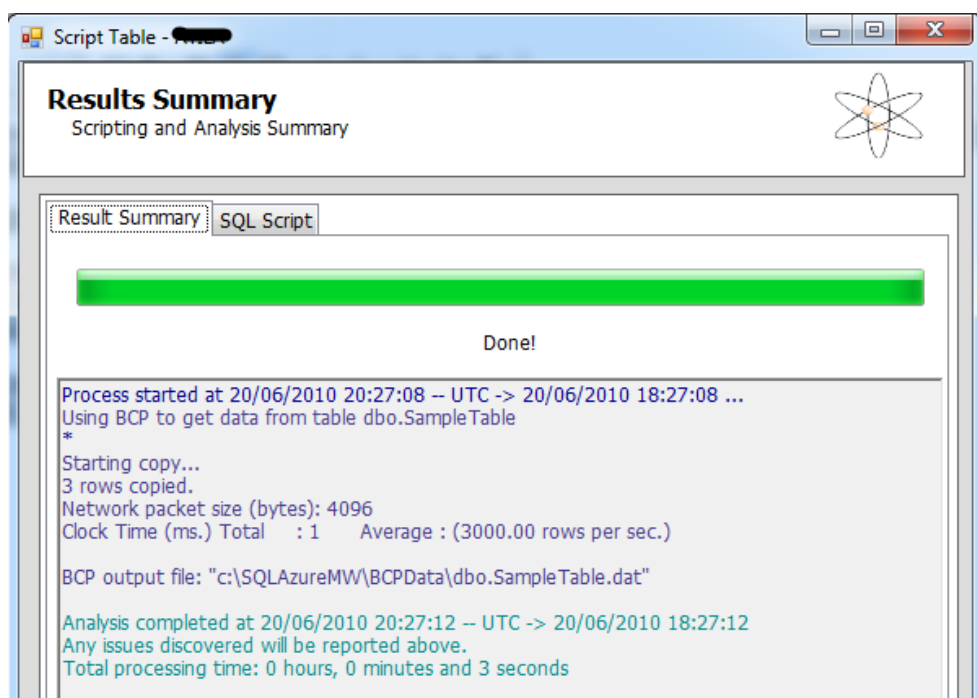


Figura I.32.- Pantalla de progreso

Una vez generados los scripts, da la posibilidad de conectarse a SQL Azure desde la herramienta para lanzar los scripts, creando por tanto la base de datos.

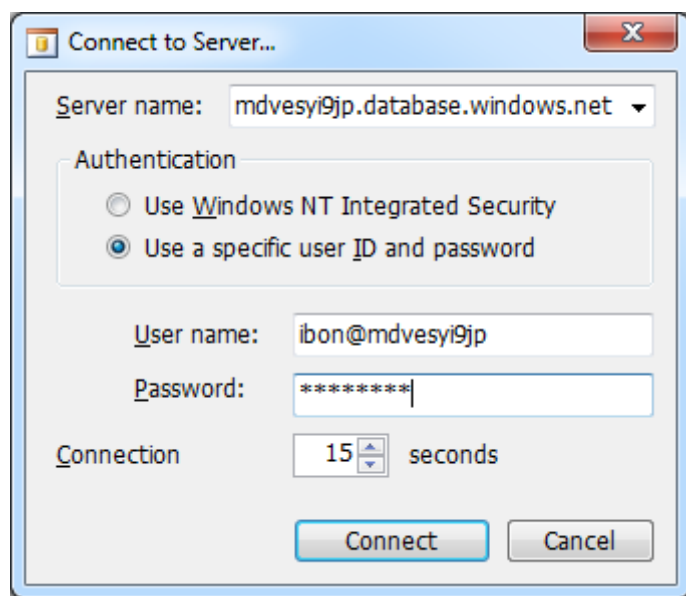


Figura I.33.- Conectarse a SQL Azure

Se debe indicar la opción de crear una base de datos nueva.

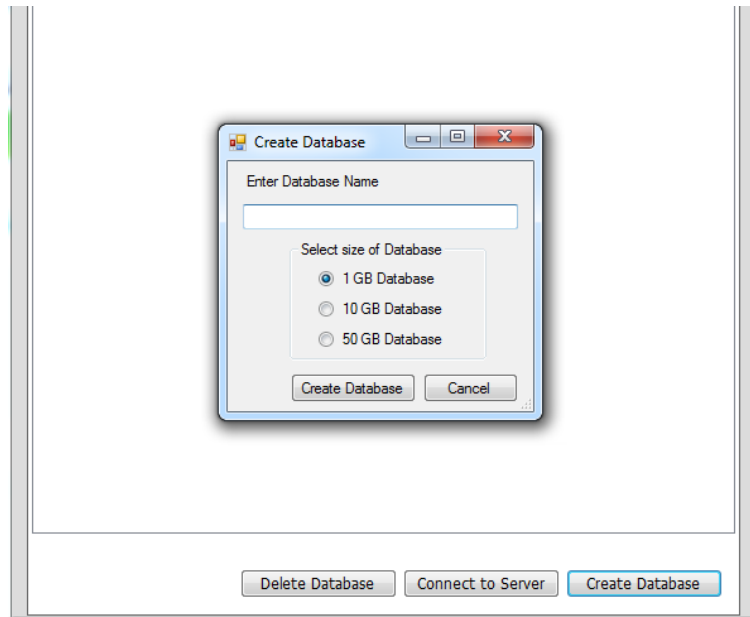


Figura I.34.- Crear una base de datos SQL Azure

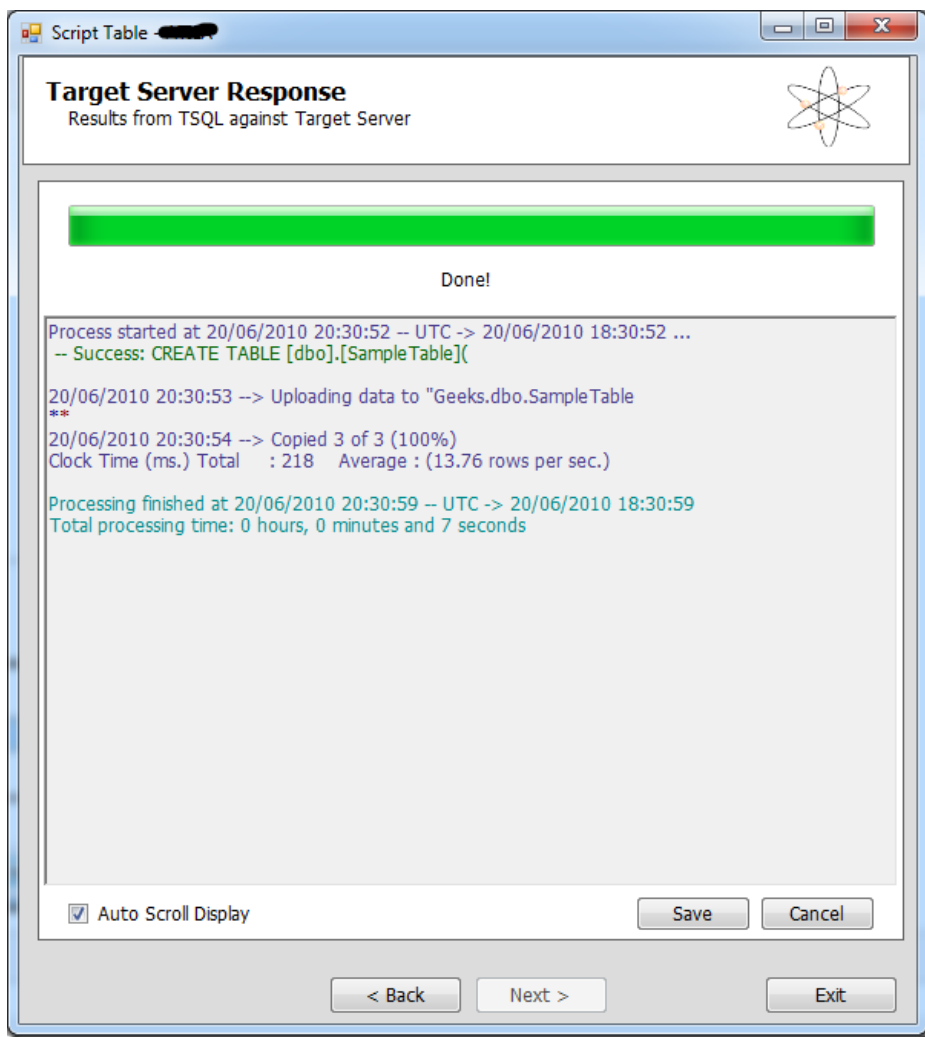


Figura I.35.- Pantalla final del wizard de migración

12.2.- Microsoft SQL Server Migration Wizard

Microsoft dispone de una herramienta que será de gran utilidad para aquellos que trabajen con MySQL 4.1 o superior, ya que permite migrar este tipo de base de datos a SQL Azure. Realmente permite migrar una base de datos MySQL a SQL Server 2005, SQL Server 2008, SQL Server 2008 R2 o SQL Azure.

La herramienta se llama Microsoft SQL Server Migration Assistant for MySQL v1.0

Del mismo modo también existe un Microsoft SQL Server Migration Assistant for Access v4.2, para poder migrar de Access a SQL Server o SQL Azure.

En este apartado se mostrará paso a paso el proceso de migración de una base de datos Access 2010 a SQL Azure.

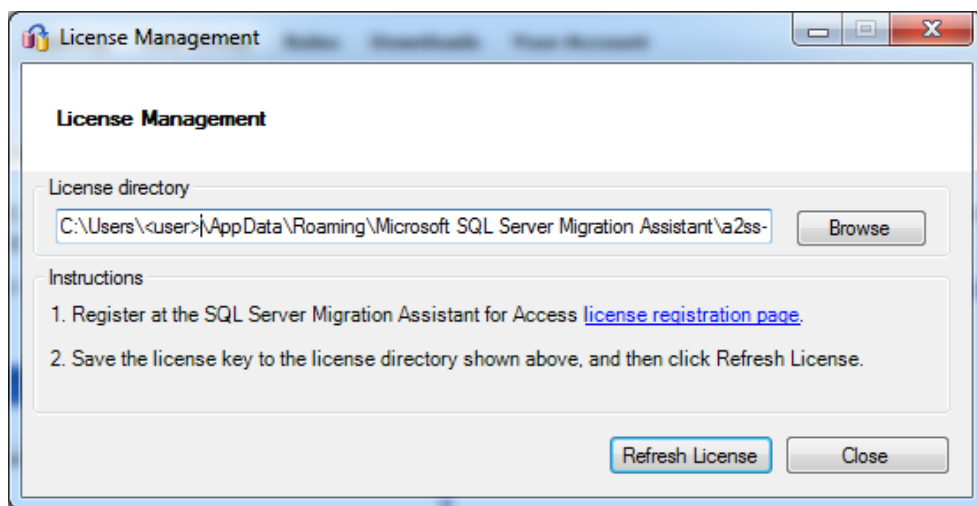


Figura 1.36.- Aceptar la licencia

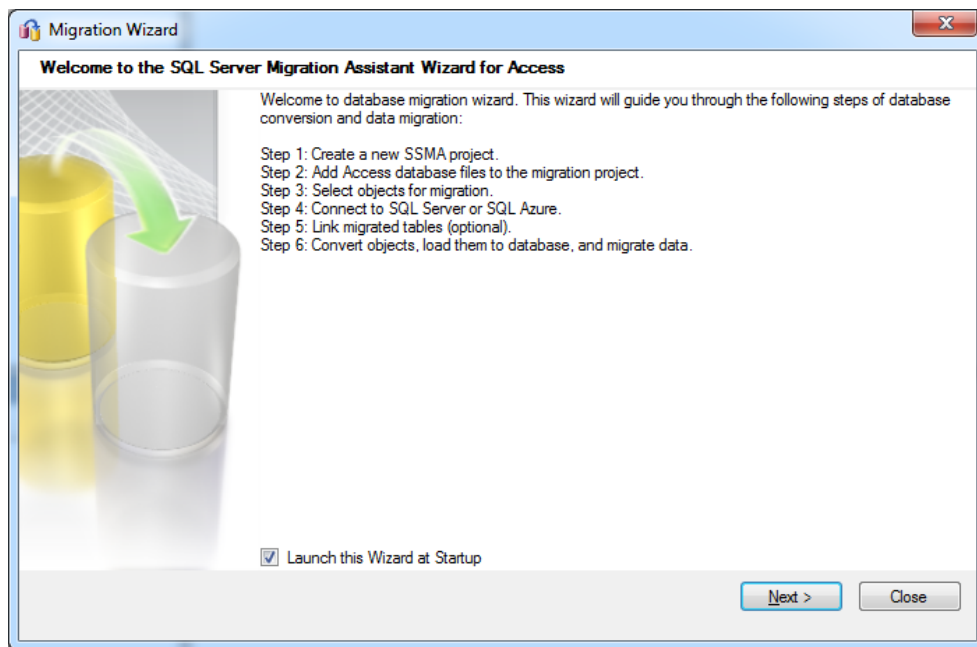


Figura 1.37.- Pantalla de bienvenida

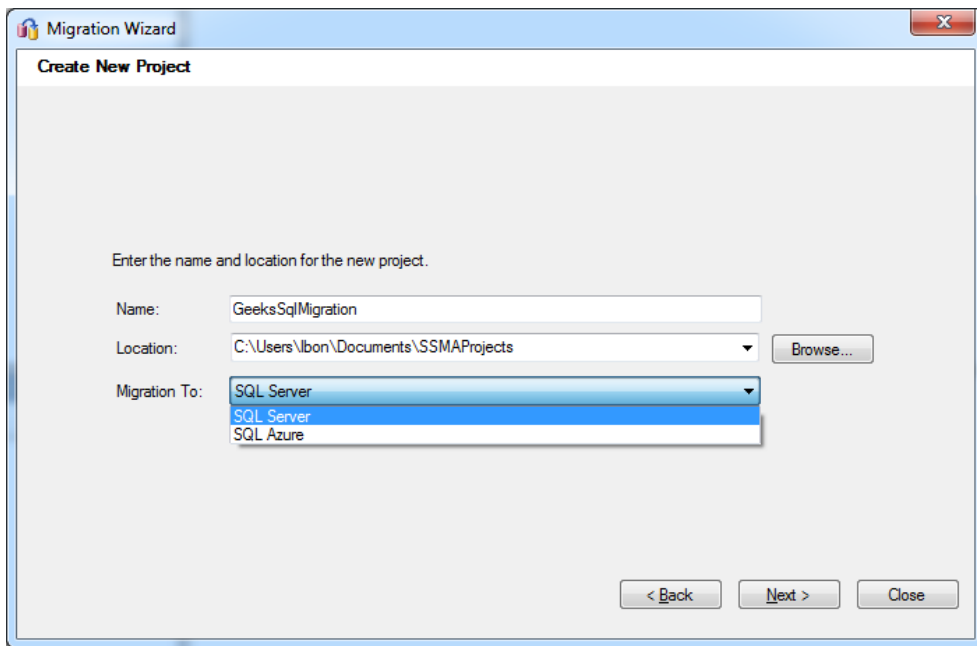


Figura I.38.- Seleccionar el tipo de migración

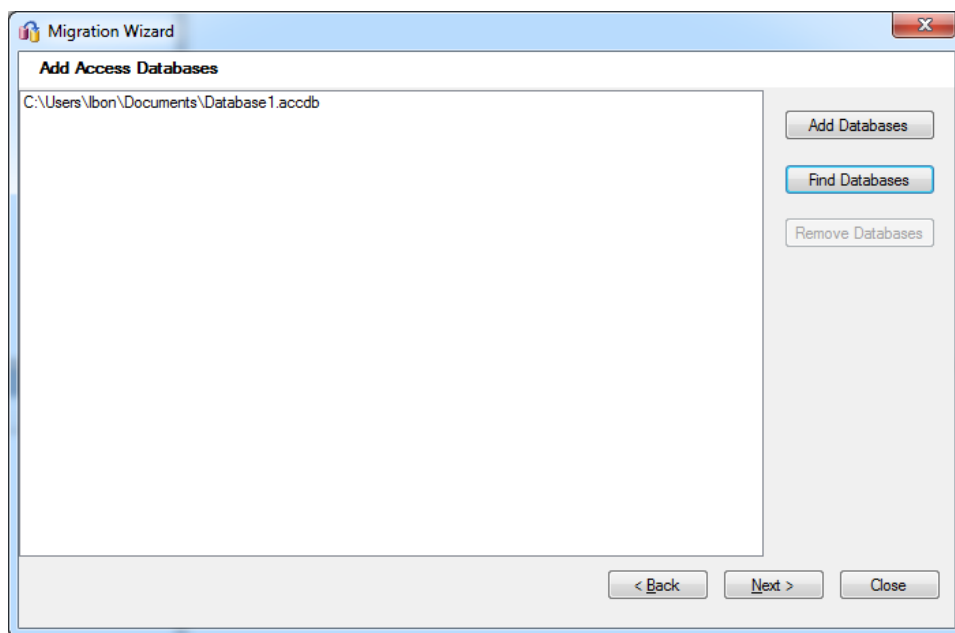


Figura I.39.- Base de datos a migrar

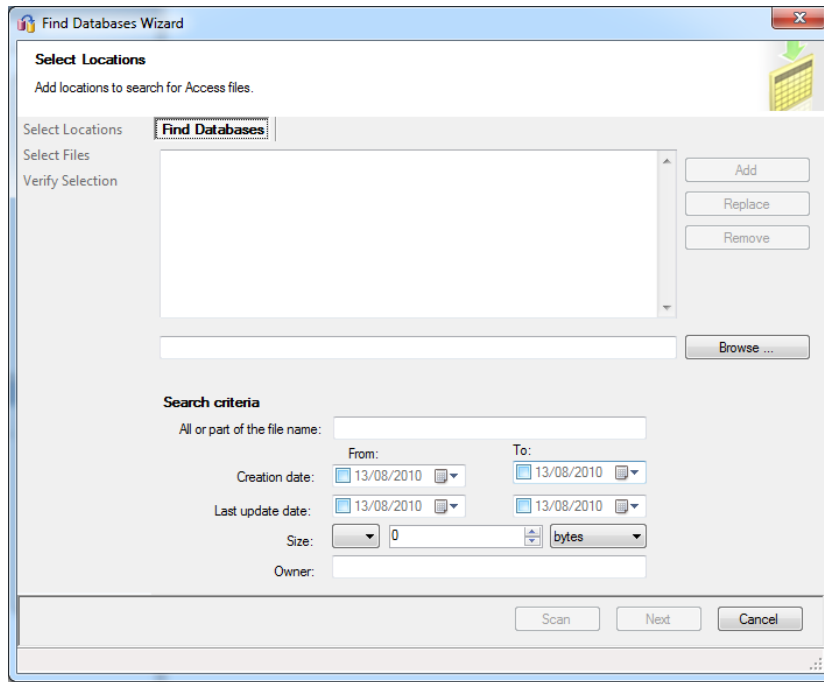


Figura I.40.- Añadir localizaciones

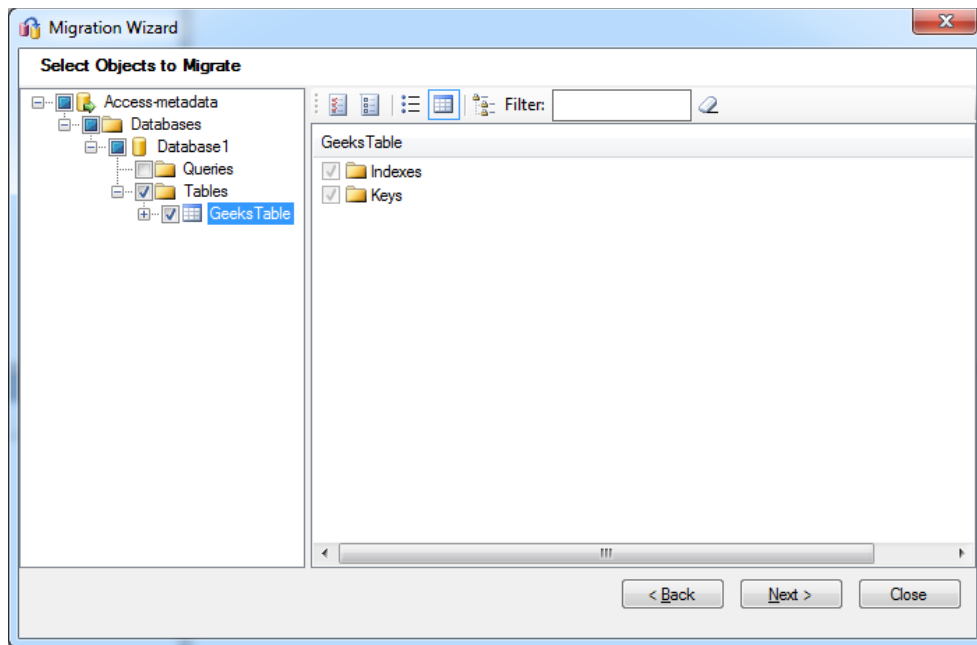


Figura I.41.- Elementos a migrar

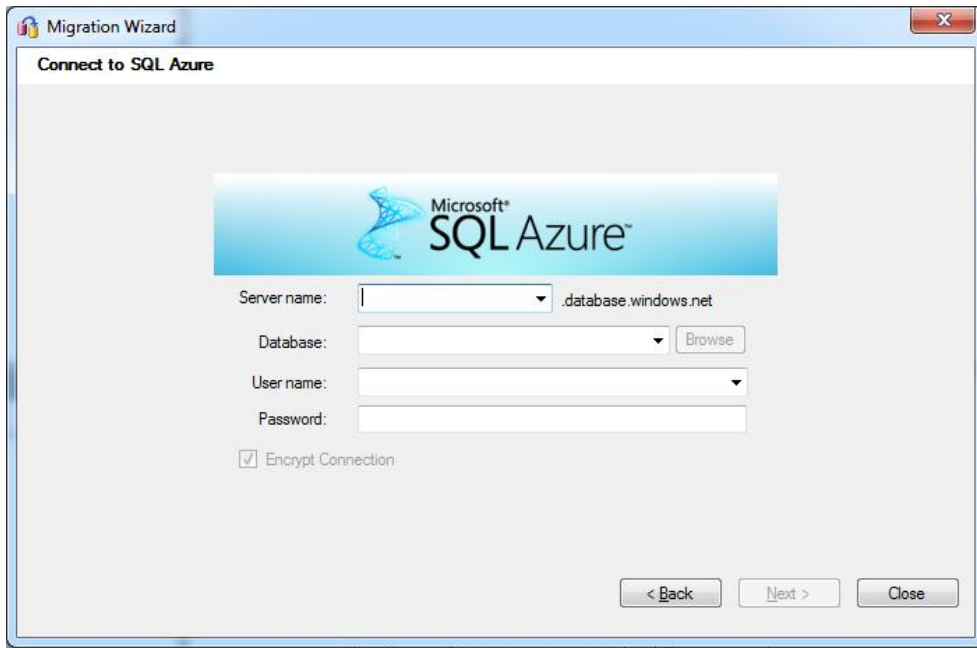


Figura I.42.- Conexión a SQL Azure

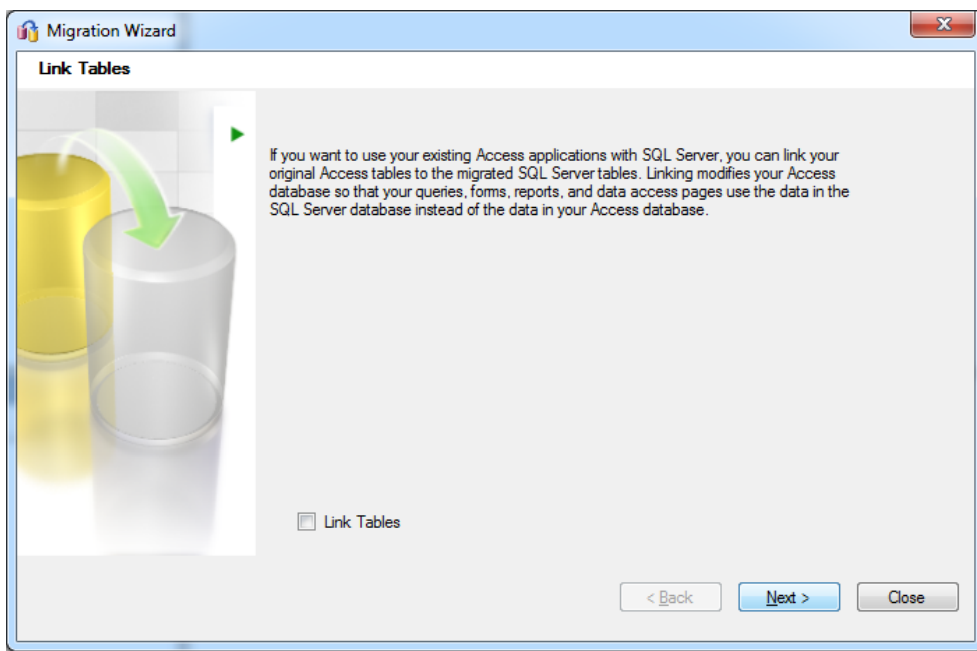


Figura I.43.- Indicar si se desea añadir "links" a SQL Azure dentro de Access

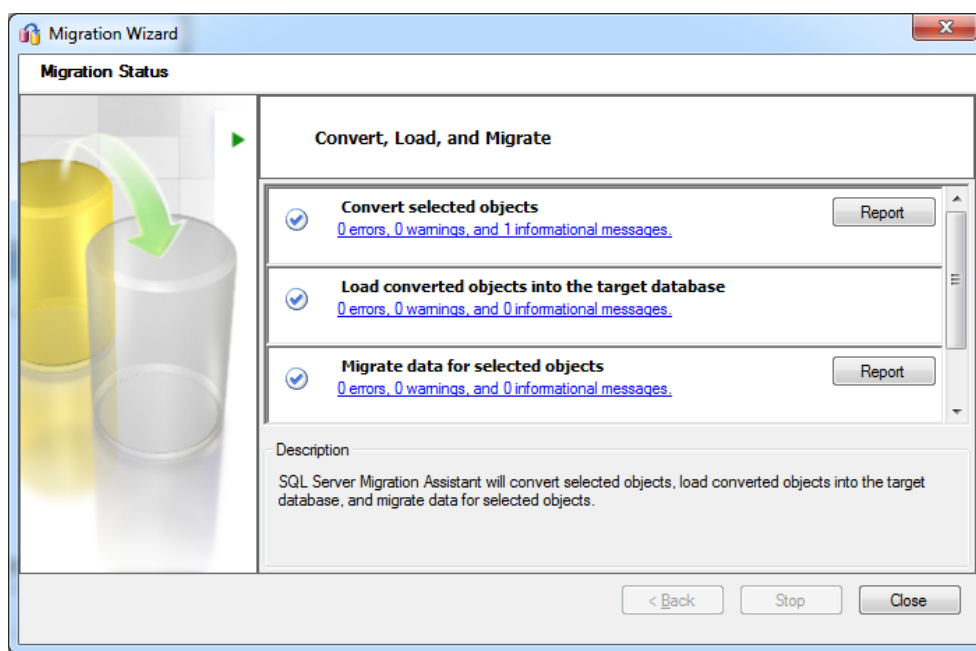


Figura I.44.- Proceso de migración

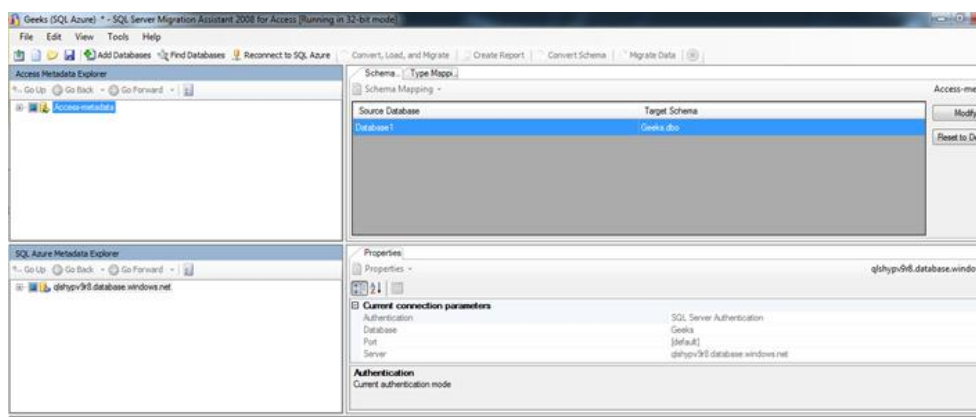


Figura I.45.- Ventana principal

13.-SQL IMPORT/EXPORT

Desde hace ya tiempo es posible disfrutar de una nueva funcionalidad de SQL Azure, todavía en CTP, que permite exportar e importar base de datos al Windows Azure Storage.

Desde el portal de administración es muy fácil exportar una base de datos completa al Windows Azure Storage y volver a importarla.

Como se puede ver, si se accede al portal de administración está disponibles dichas opciones en la barra superior.

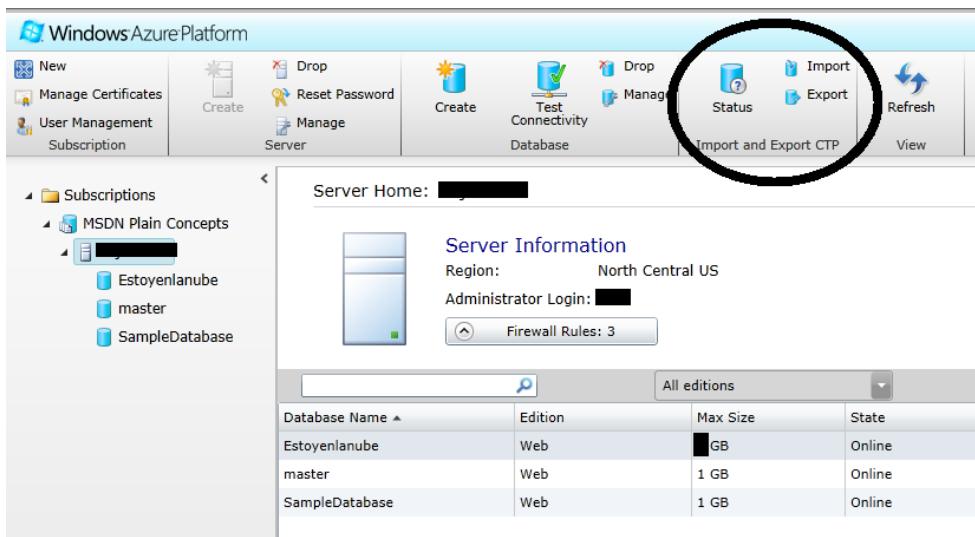


Figura I.46.- Administración de SQL Azure

Seleccionando la base de datos que interese es posible seleccionar la opción exportar, la cual solicitará la información necesaria para la operación.

En primer lugar nos solicitará las credenciales de la base de datos y en segundo lugar habrá que indicar la ubicación dónde debe dejar el resultado de la operación.

La URL debe ser la URL de un blob que no exista. El contenedor sí debe existir.

Para acceder al contenedor es posible usar la clave del storage (Access Key) o una clave compartida (Shared Key).

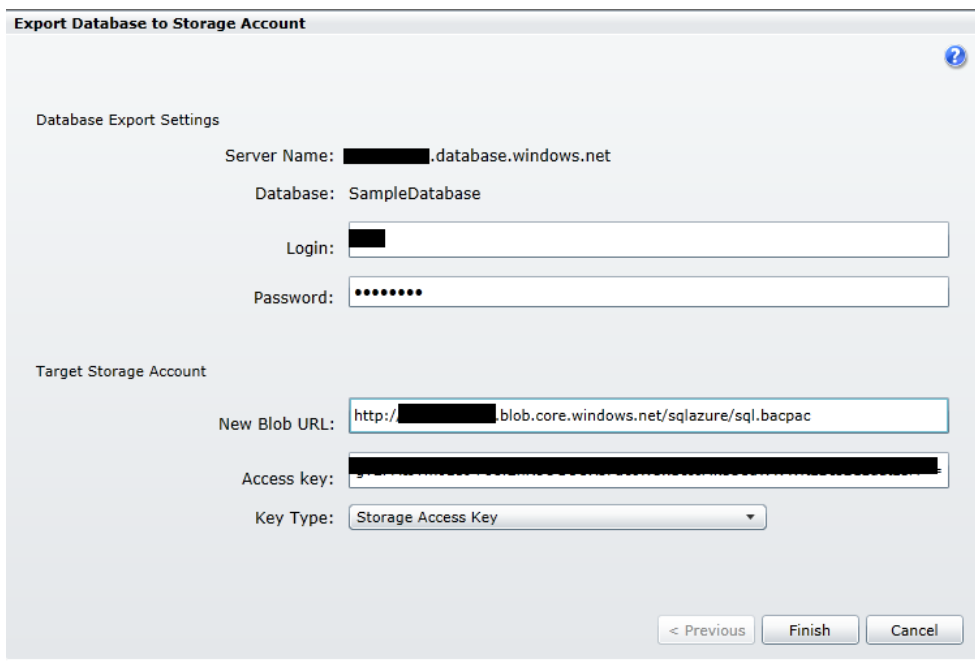


Figura I.47.- Opciones de exportación

Una vez realizada la operación el proceso de exportación se iniciará. El proceso no es inmediato, depende del tamaño de la base de datos.

Por este motivo el portal también ofrece una ventana de estado dónde se puede ver el estado de todas las operaciones de exportación e importación, ya que ambas operaciones se hacen de forma asíncrona.

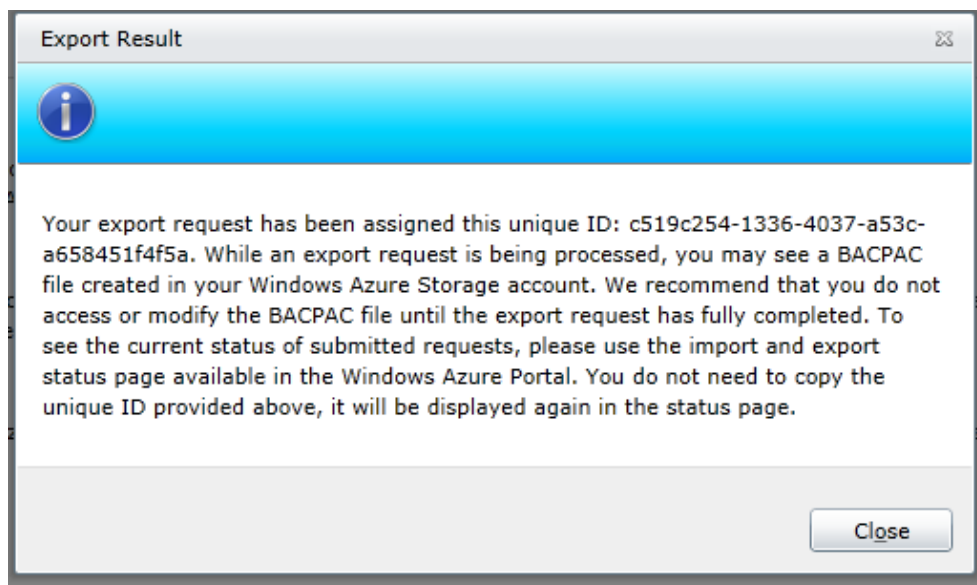


Figura I.48.- Aviso de finalización

Una vez hecha la operación se puede ver que se ha creado un nuevo blob en el contenedor especificado.

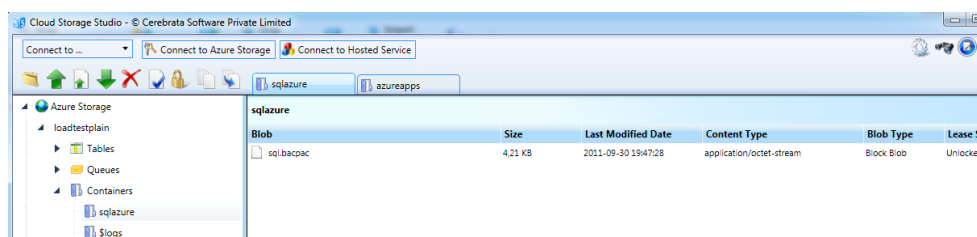


Figura I.49.- Vista del Storage

Y si se quiere importarla, tan fácil como seleccionar la opción importar, indicar las credenciales, el nombre de la nueva base de datos dónde se importarán los datos y el blob dónde está la base de datos.

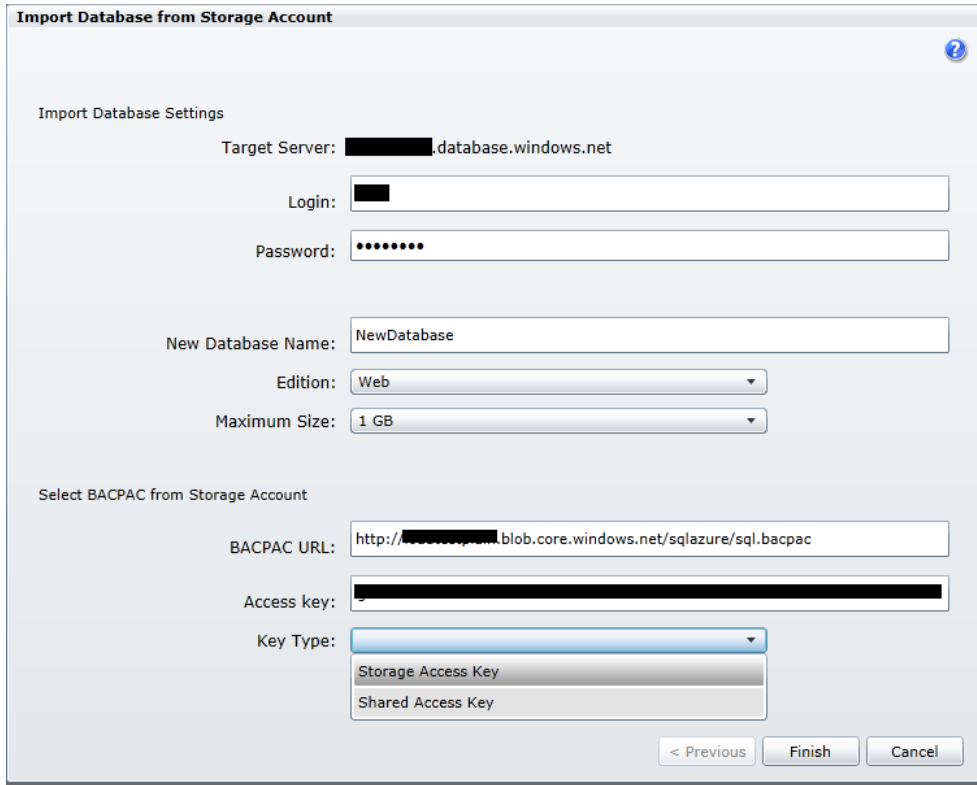


Figura 1.50.- Opciones de importación

La ventana de estado mostrará el estado de todas las operaciones de importación y exportación.

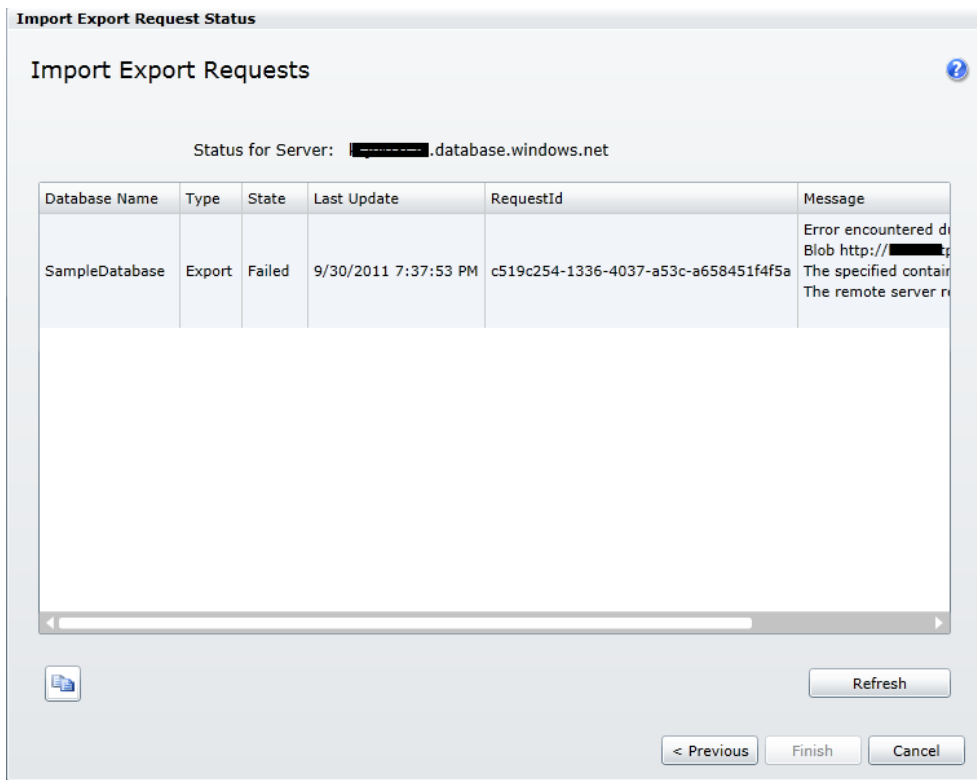


Figura 1.51.- Ventana de estado

Y ya por último, comentar que esta funcionalidad también está disponible a través del API que ofrece Windows Azure, lo que hace que se podría llegar a automatizar los procesos de exportación e importación.

14.-SQL AZURE DATA SYNC

SQL Azure Data Sync es una de las funcionalidades de SQL Azure accesible desde el portal de administración de Windows Azure, aunque esté todavía en CTP.

Básicamente, como se puede entender de su nombre, es un servicio de sincronización de datos, que permite sincronizar base de datos de SQL Azure que estén en el mismo o diferente datacenter o based de datos que se encuentren on-premise con base de datos que se encuentren hospedadas en Windows Azure.

La sincronización entre las base de datos se podrá hacer de forma bidireccional.

Como paso adicional que se verá posteriormente, la sincronización con una base de datos on-premise requiere la instalación de un agente de sincronización en el servidor on-premise. Este paso no es necesario en el caso de realizar el proceso entre base de datos que estén en Windows Azure.

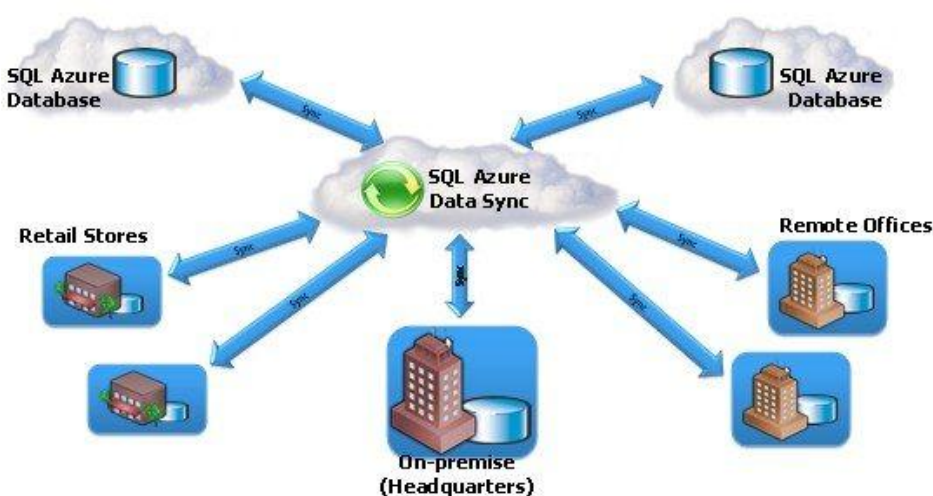


Figura 1.52.- SQL Azure Data Sync

Como se ha comentado anteriormente toda la funcionalidad está accesible desde el portal de administración de Windows Azure.

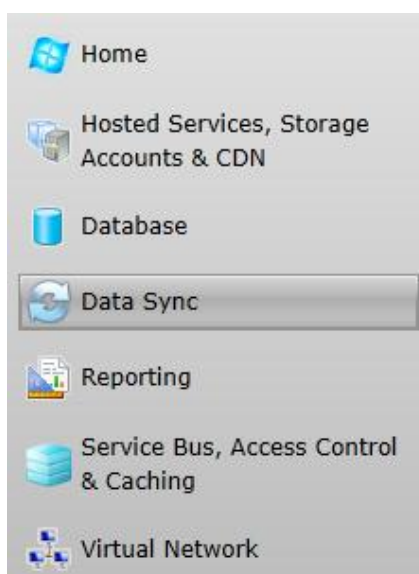


Figura 1.53.- Acceso a Data Sync

Una vez se accede a la característica de sincronización, el primer paso será realizar un aprovisionamiento del servicio. El proceso de aprovisionamiento pedirá aceptar los términos de licencia e indicar en qué datacenter se quiere realizar dicho aprovisionamiento.

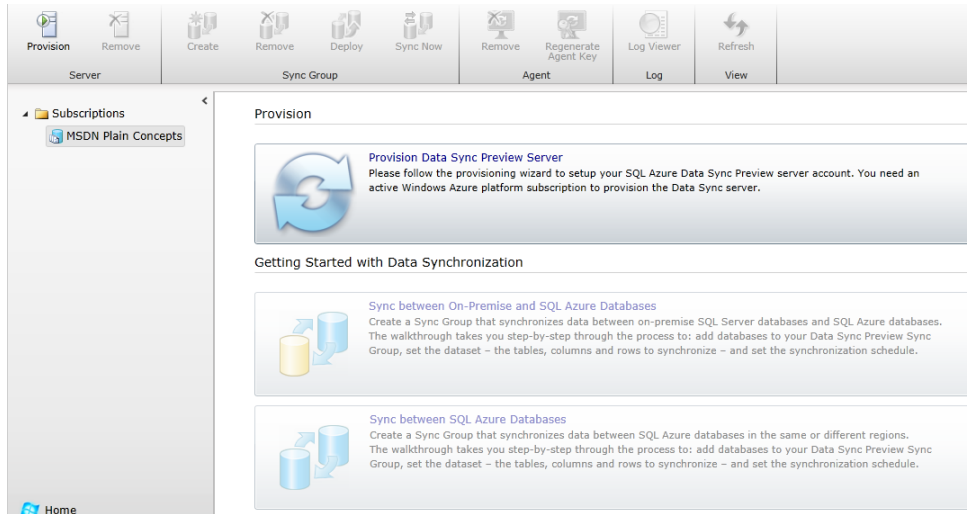


Figura I.54.- Aprovisionar el servicio

Una vez realizado el proceso se habilitarán las opciones de sincronización. En proceso de sincronización es un asistente en el cuál se guiará paso a paso sobre todas las acciones necesarias para configurar un proceso de sincronización.

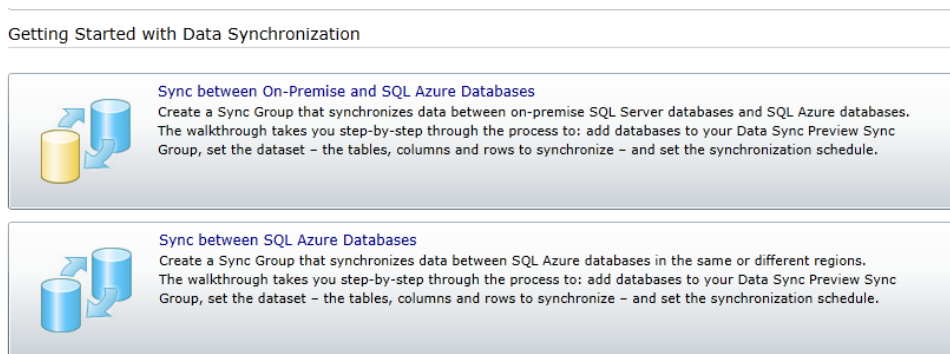


Figura I.55.- Opciones de sincronización

14.1.- Sincronización entre base de datos SQL Azure

Si se selecciona la opción de sincronización entre base de datos SQL Azure, el primer paso será dar un nombre a dicho proceso.

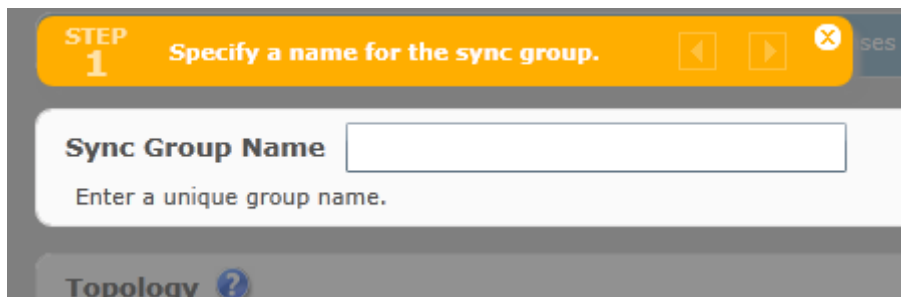
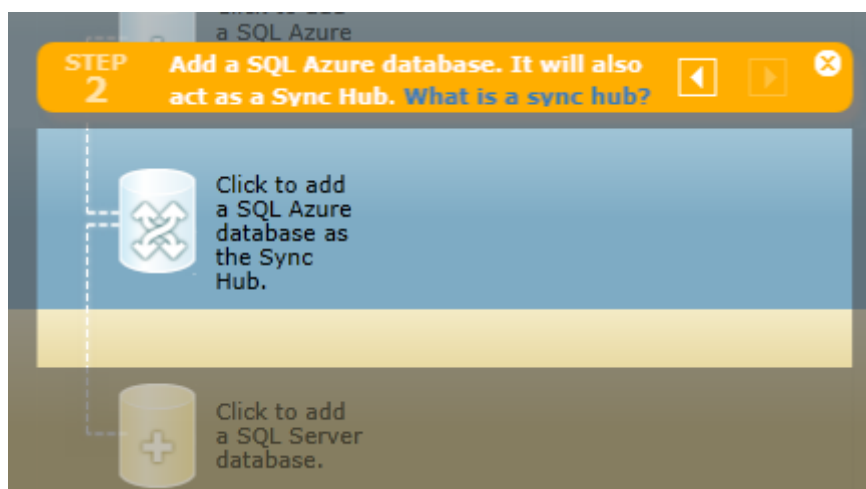


Figura I.56.- Seleccionar el nombre del grupo

El siguiente paso será añadir una de las base de datos a sincronizar, indicando la ubicación de la misma y las credenciales de acceso.

**Figura I.57.- Añadir una base de datos SQL Azure****Figura I.58.- Base de datos a sincronizar**

El siguiente paso será elegir la base de datos con la cual se realizará la sincronización. En este paso, como se puede apreciar en la figura, se debe elegir la dirección del proceso de sincronización.

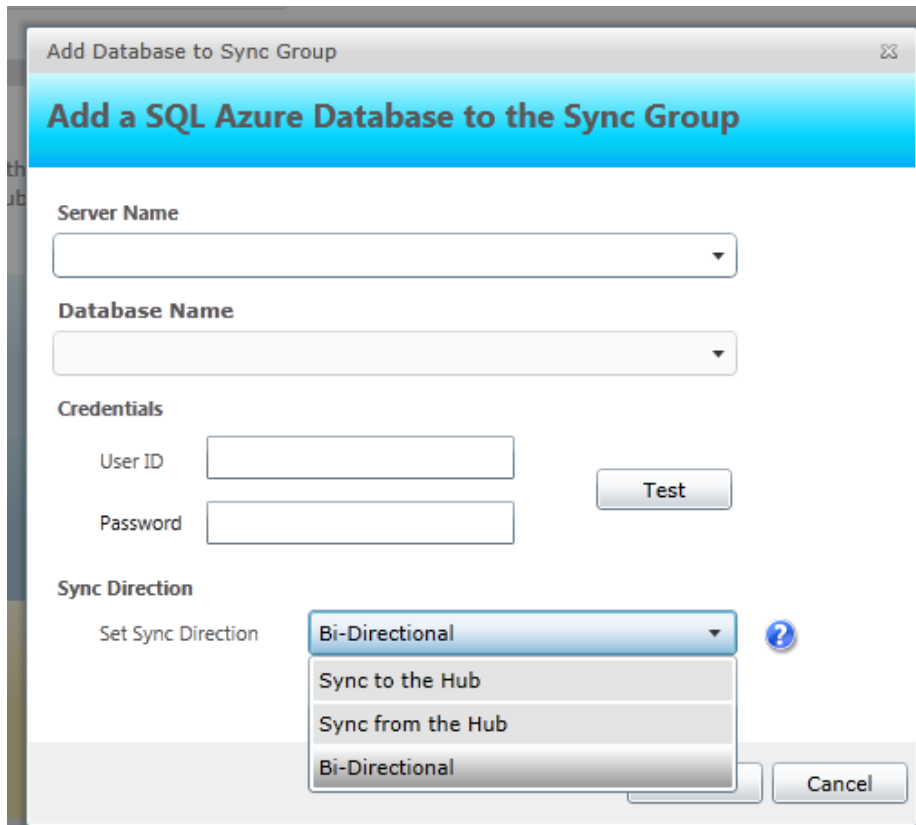


Figura 1.59.- Base de datos a sincronizar

Una vez indicados los dos puntos de la sincronización el siguiente paso será indicar la periodicidad del proceso y el método de resolución de conflictos que quiere aplicarse.

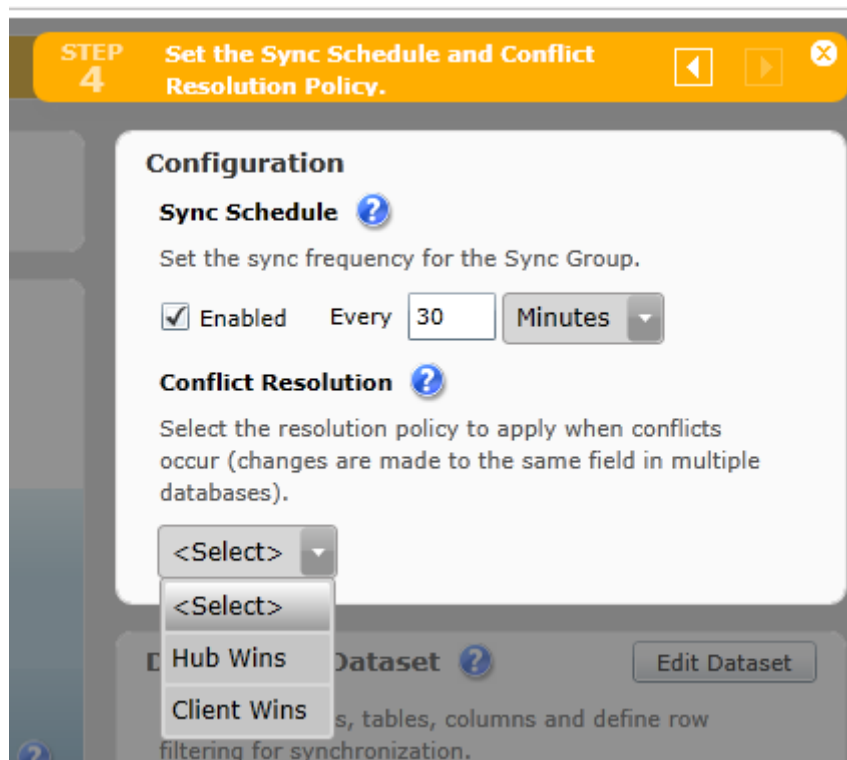


Figura 1.60.- Programar la sincronización

Y en el último paso puede definirse con más detalle qué información es la que se desea sincronizar; qué tablas, qué columnas etc...

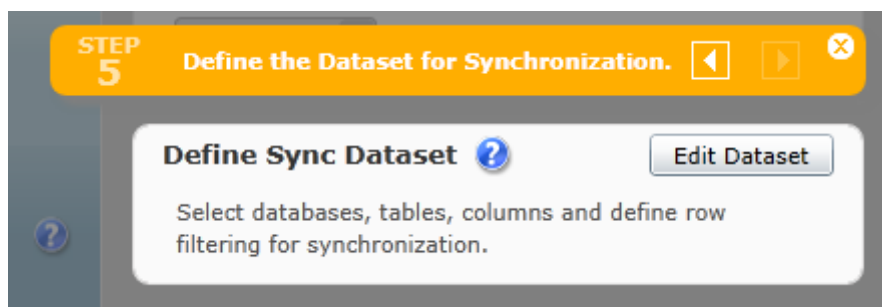


Figura I.61.- Configurar los elementos a sincronizar

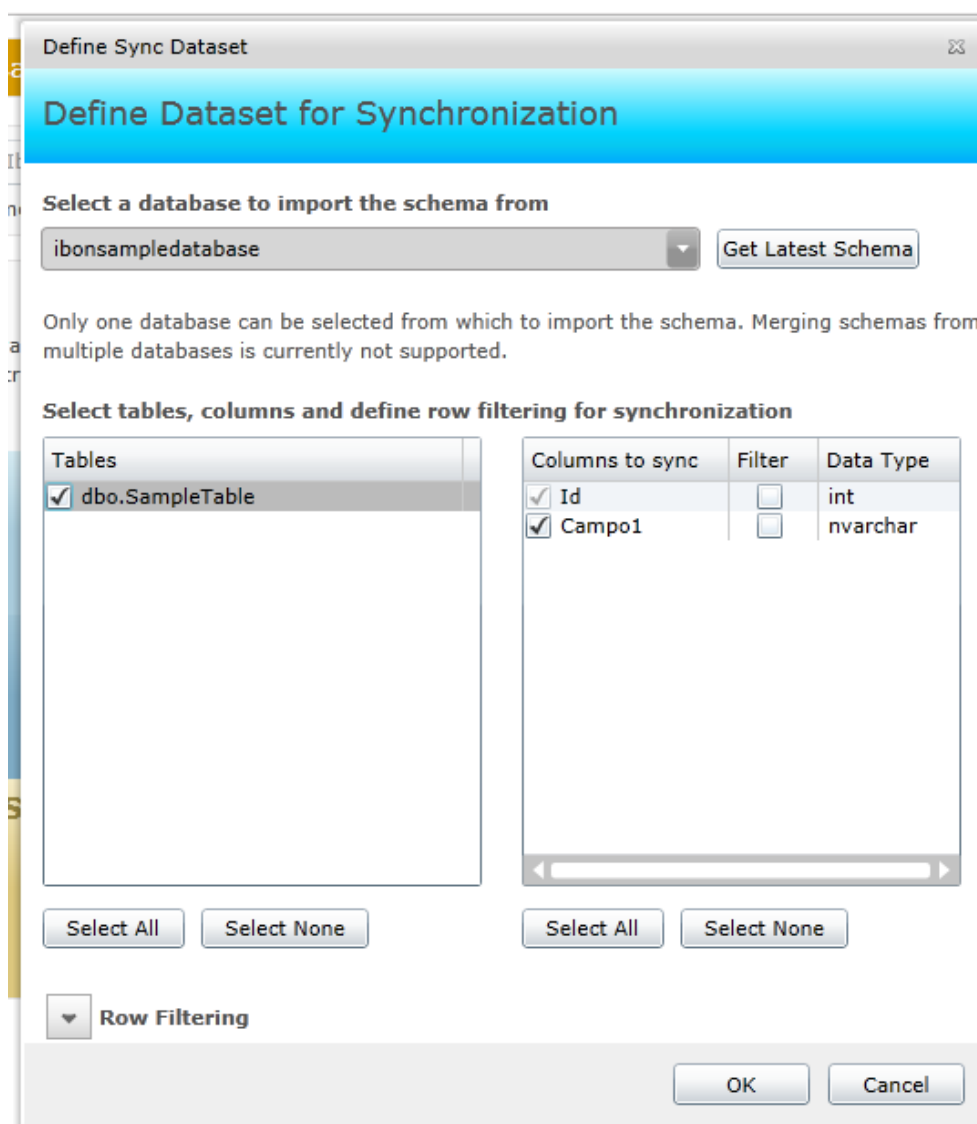


Figura I.62.- Configurar los elementos a sincronizar

Y por último, el asistente pedirá guardar los cambios seleccionando la opción de “deploy”.

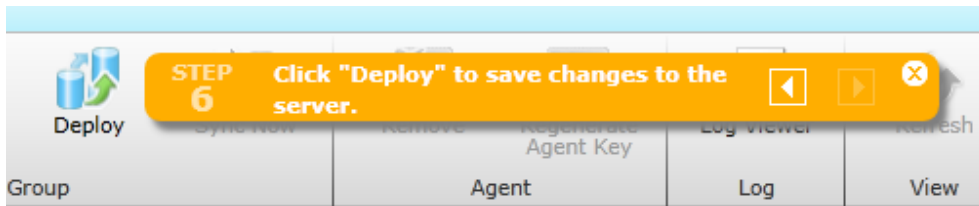


Figura I.63.- Deploy

Una vez configurado todo el proceso desde el portal de administración podrá verse toda la información del proceso y claro está, modificar cualquiera de los elementos configurados en el asistente.

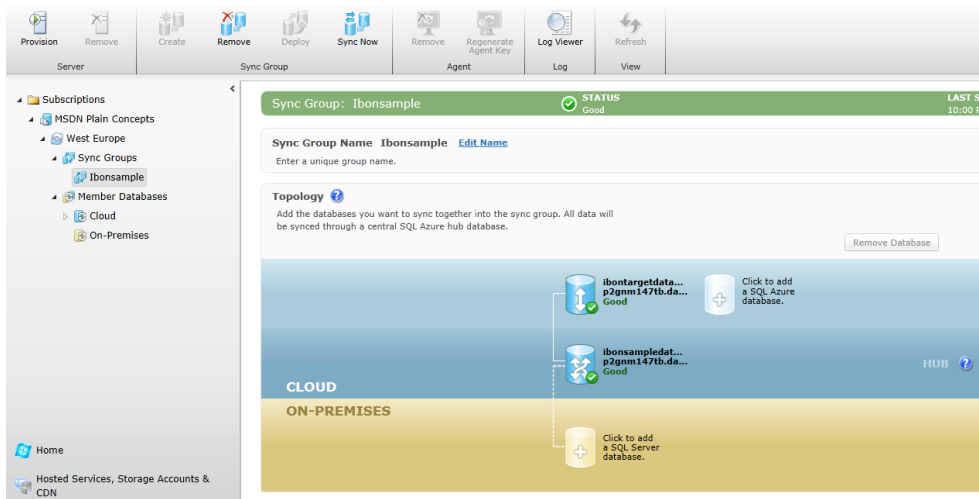


Figura I.64.- Información del proceso de sincronización

Así mismo también el portal de administración nos ofrece la posibilidad de consultar un log con todas las operaciones que se realizan en los procesos de sincronización.

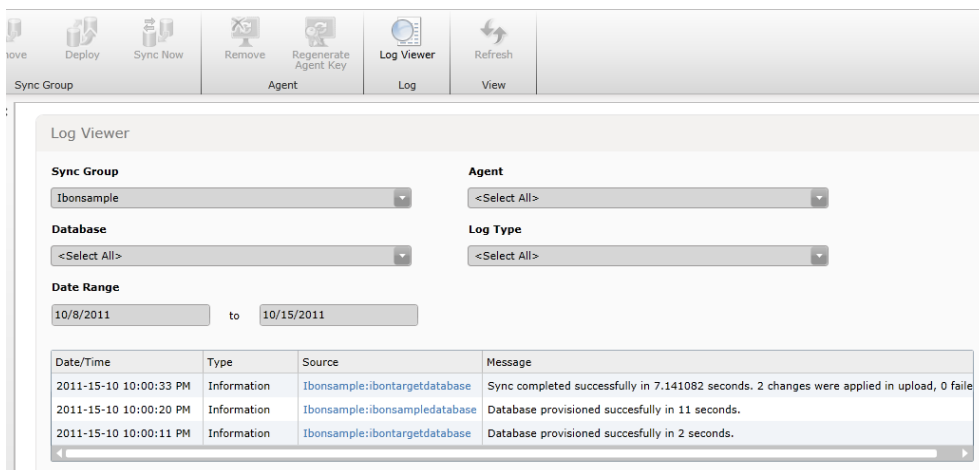


Figura I.65.- LogViewer

Comentar por último que aunque en este apartado el proceso de creación del grupo de sincronización se ha hecho a través de un asistente, existe también la posibilidad de no usarlo y de crear el proceso de sincronización de forma “manual” consiguiendo, claro está, el mismo resultado.

14.2.- Sincronización con un servidor on-premise

El proceso de sincronización con un servidor on-premise es prácticamente igual al realizado entre servidores SQL Azure, salvo por el hecho que implica la instalación de un agente de sincronización en el servidor on-premise.

Aunque para realizar un proceso de este tipo se podría utilizar un asistente similar al visto anteriormente, en este ejemplo lo que se va a realizar es añadir un servidor on-premise al grupo creado anteriormente.

Para ello, sobre el diagrama de la Figura 14.14 puede seleccionarse la opción de añadir un servidor on-premise.

Dicha opción permite añadir un servidor que ya disponga del agente de sincronización o un nuevo servidor que no lo tenga. En este caso se optará por la segunda opción.

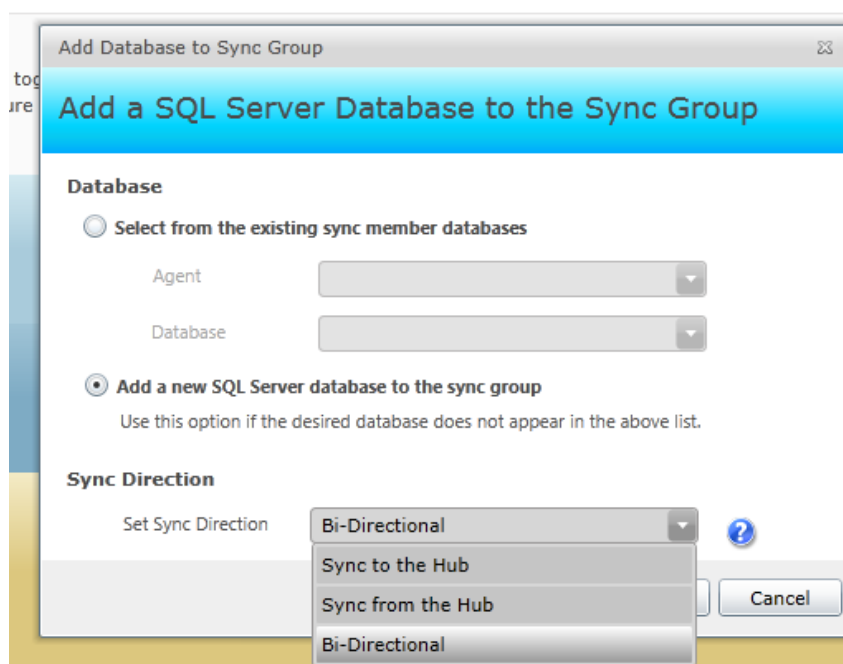


Figura I.66.- Instalar agente de sincronización

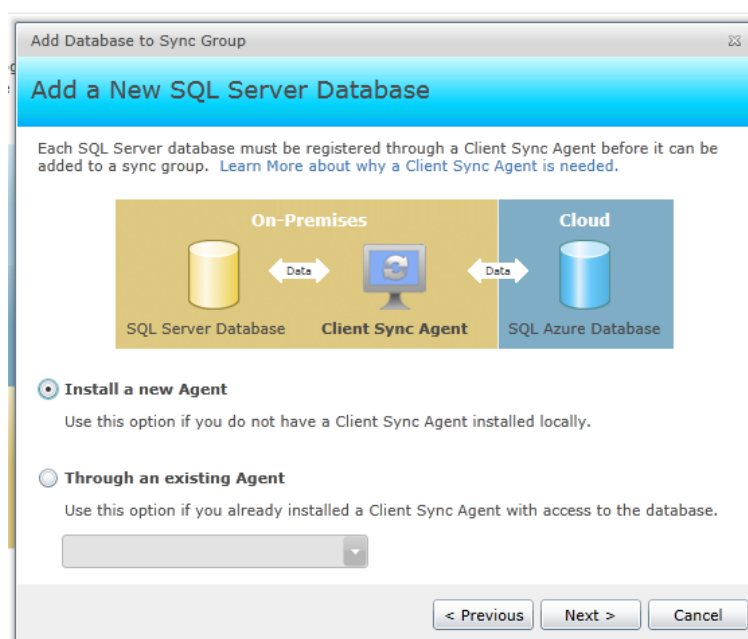


Figura I.67.- Instalar agente de sincronización

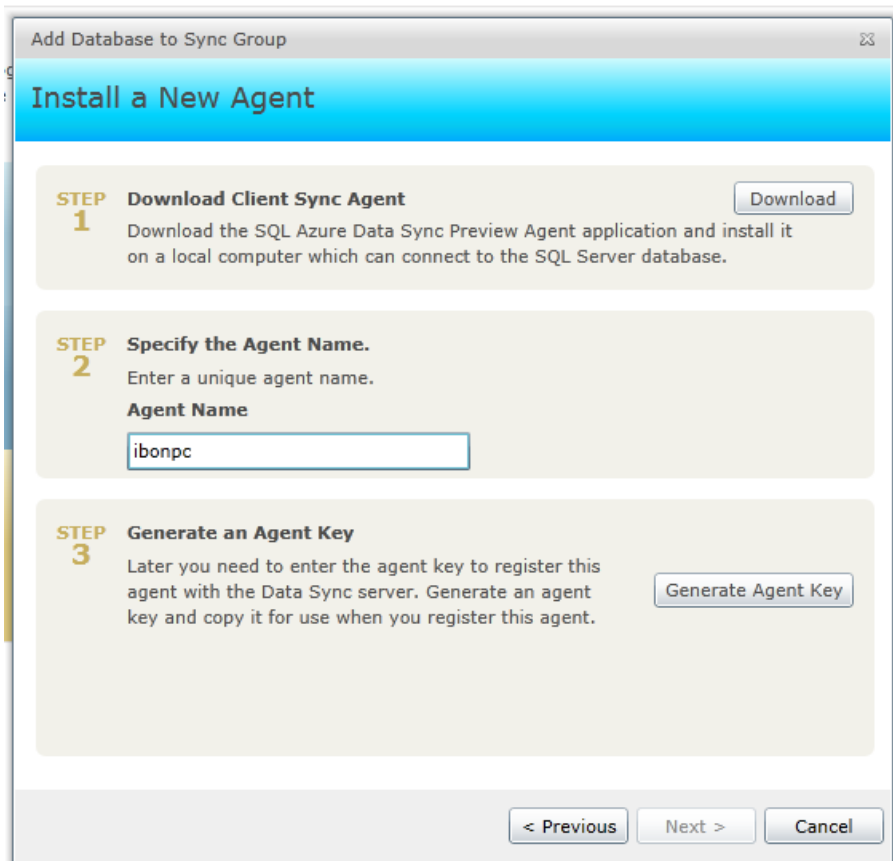


Figura I.68.- Instalar agente de sincronización

Después de este paso será necesario descargarse el agente de sincronización (msi) e instalarlo en el servidor on-premise. Una vez instalado hay que arrancar el agente (desde el menú inicio) e indicarle la clave de acceso generada en el paso 3 de la figura 14.17.

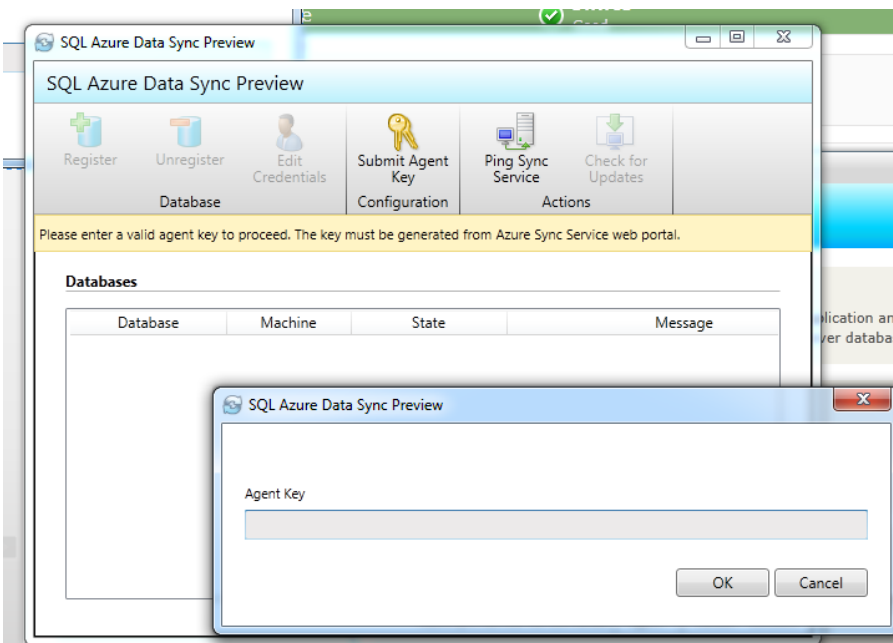


Figura I.69.- Configurar agente de sincronización

Una vez hayamos configurado la clave, el siguiente paso será registrar la base de datos local que queremos sincronizar.

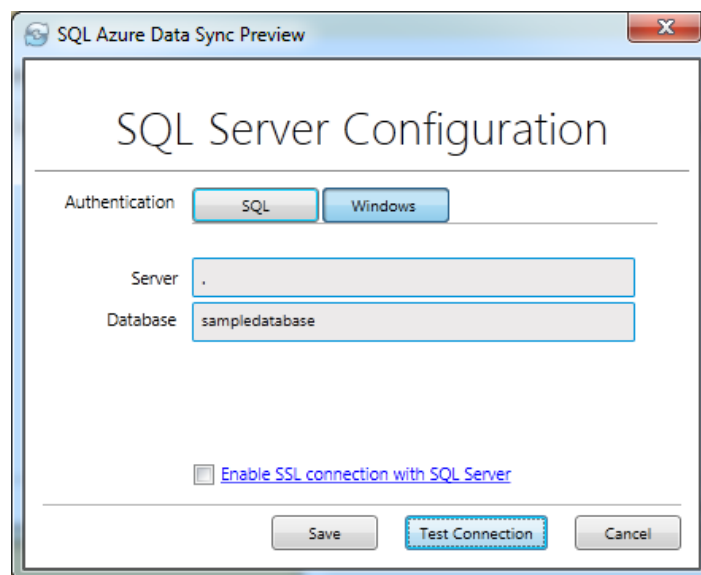


Figura I.70.- Registrar una base de datos

Una vez hecho este paso, se podrá ver cómo la base de datos registrada se muestra en el servidor y podemos seleccionarla para ser usada en el proceso de sincronización.



Figura I.71.- Seleccionar la base de datos a sincronizar

Una vez añadido el servidor, para que la modificación tenga efecto será necesario volver a realizar un despliegue, seleccionando la acción “deploy”.



Figura I.72.- Topología del grupo de sincronización

15.-SQL AZURE REPORTING SERVICES

Una de las características disponibles en CTP es SQL Azure Reporting Services, la cual es una versión reducida y simplificada de la versión de Reporting Services disponible en servidores on-premise.

La versión actual dispone de algunas limitaciones y características que no existen en la versión on-premise, como:

- Únicamente soporta una base de datos SQL Azure como origen de datos.
- No dispone de ninguna característica relacionada con la programación y envío de informes.
- No dispone de la funcionalidad de caché.
- No se puede lanzar el generador de informes desde el portal de Windows Azure.

Con todas estas restricciones SQL Azure Reporting Services se convierte en un sistema básico que permite el despliegue de informes (rpt) y que sólo pueden conectarse a SQL Azure.

El primer paso si se quiere trabajar con SQL Azure Reporting Services será crear un servidor desde el portal de administración de Windows Azure.

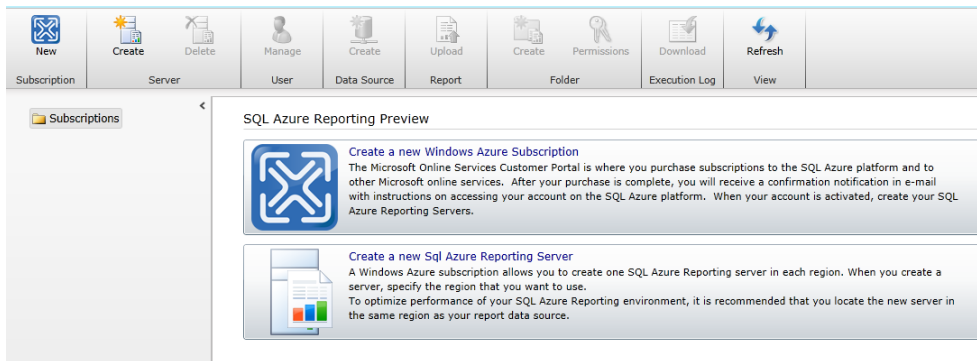


Figura I.73.- Crear servidor de reporting services

Al seleccionar la opción de crear un servidor y después de haber aceptado el acuerdo de licencia, se pedirá la ubicación que se desea para el servidor, que claro está, es conveniente tener el mismo datacenter dónde se encontrarán las base de datos SQL Azure que usarán los informes.

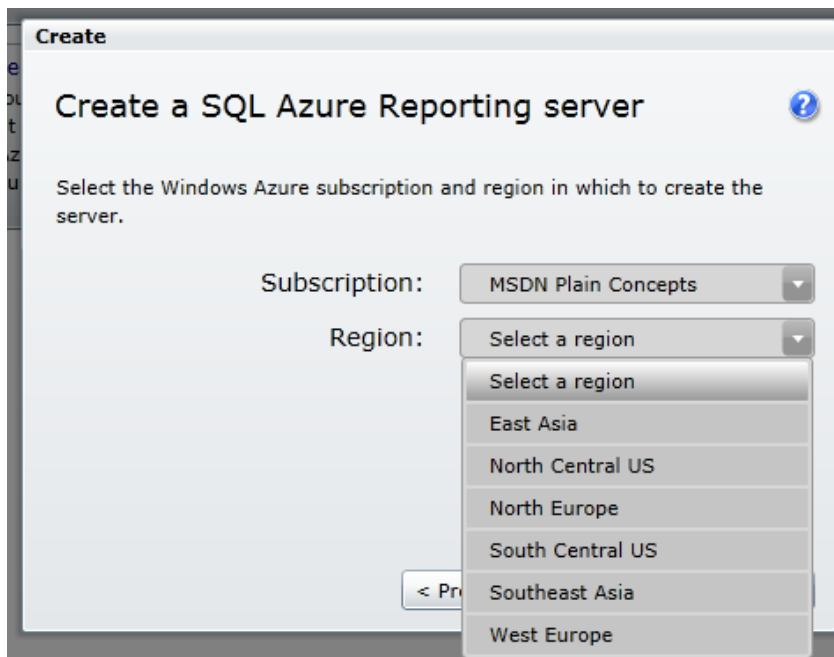


Figura I.74.- Seleccionar la región

Una vez seleccionado el datacenter, el siguiente paso es indicar las credenciales de administrador del servidor.



Figura I.75.- Indicar las credenciales de administración

Con estos simples pasos ya se ha creado un servidor en el cuál se puedan desplegar los informes de reporting services, los cuáles se pueden crear del mismo modo en que se pueden crear los informes que se desplegarían en un servidor on-premise.

Seleccionado el servidor de reporting recién creado se pueden ver todas las operaciones disponibles sobre el mismo viendo la barra superior.

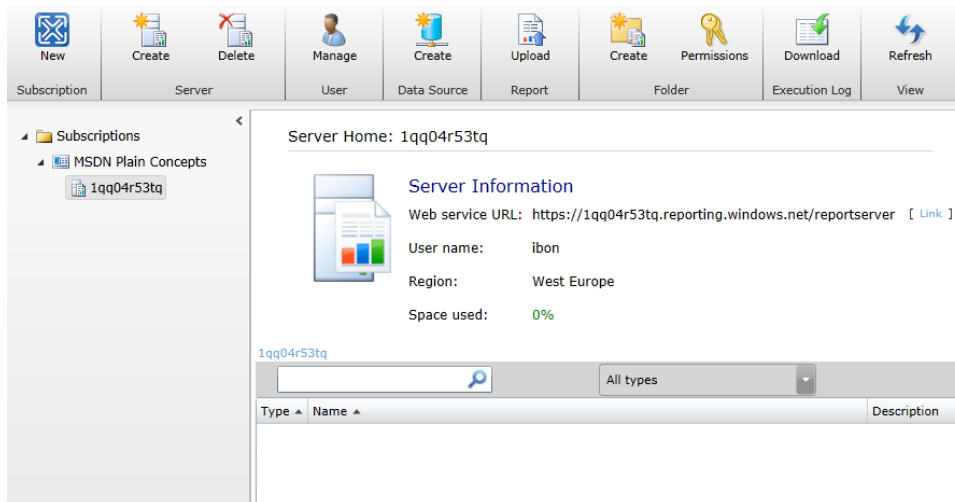


Figura I.76.- Acciones del servidor

Las acciones posibles son:

- Crear un nuevo servidor.
- Eliminar un servidor.
- Administrar la seguridad, permitiendo añadir usuarios y los roles de cada usuario.
- Crear un origen de datos para que se use en los informes.
- Subir un informe desde el disco duro local.
- Crear carpetas para organizar los informes.
- Gestionar la seguridad a nivel de carpeta.
- Descargar el log.

El significado de estas acciones es el mismo que en un servidor on-premise y se debe gestionar la seguridad del mismo modo en que se harían en un servidor on-premise.

Por último, en la parte derecha se ofrece información sobre el servidor, sobre los informes desplegados en éste y sobre la URL de acceso al report server.

Esta es la URL que se deberán usar a la hora de crear los informes desde las herramientas de desarrollo de informes, ya sea Visual Studio o Report Builder.

A continuación se muestran una serie de pantallazos dónde se ven algunas de las acciones comentadas anteriormente.

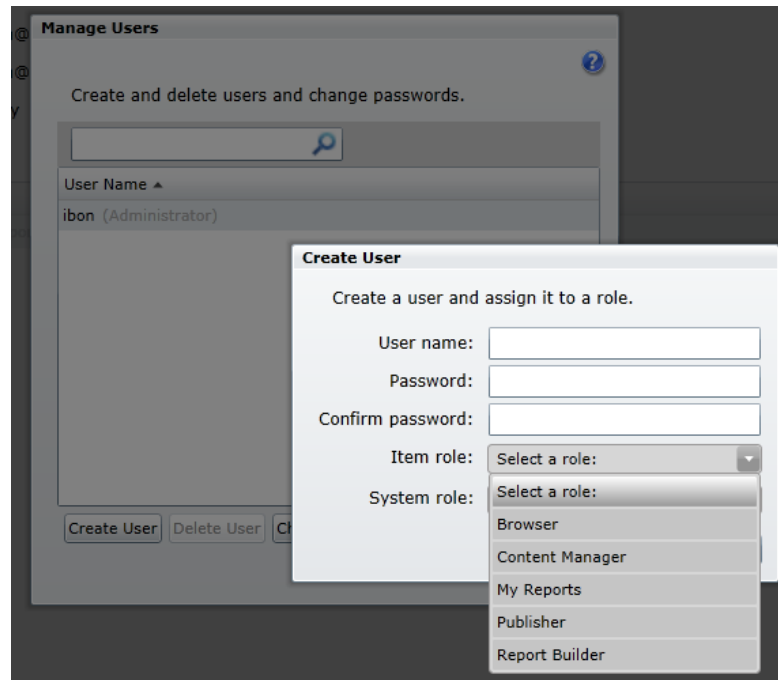


Figura I.77.- Añadir un usuario

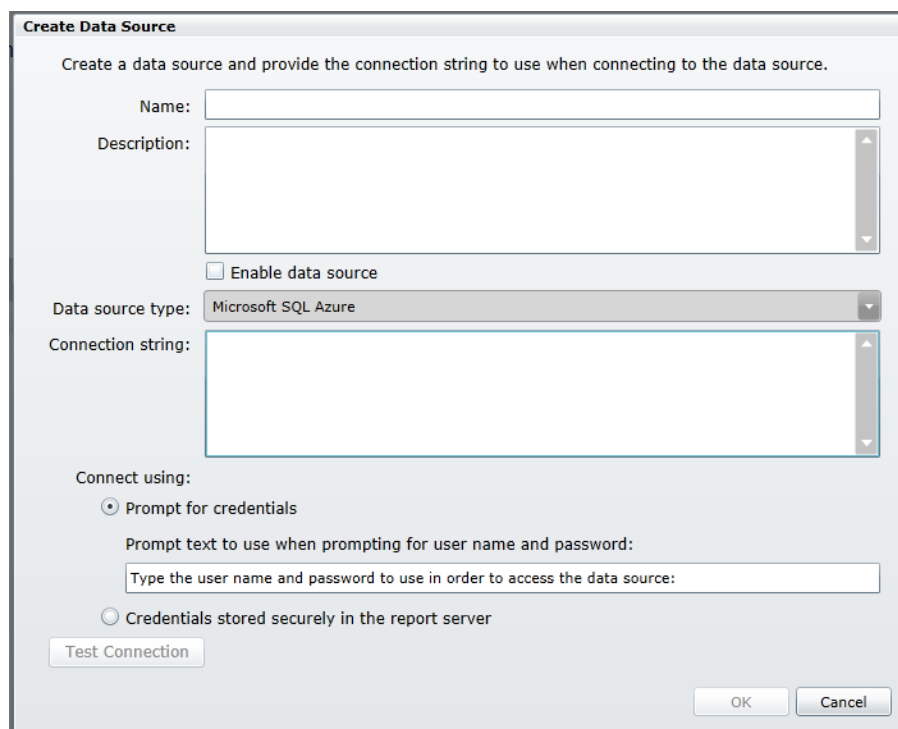


Figura I.78.- Añadir un datasource

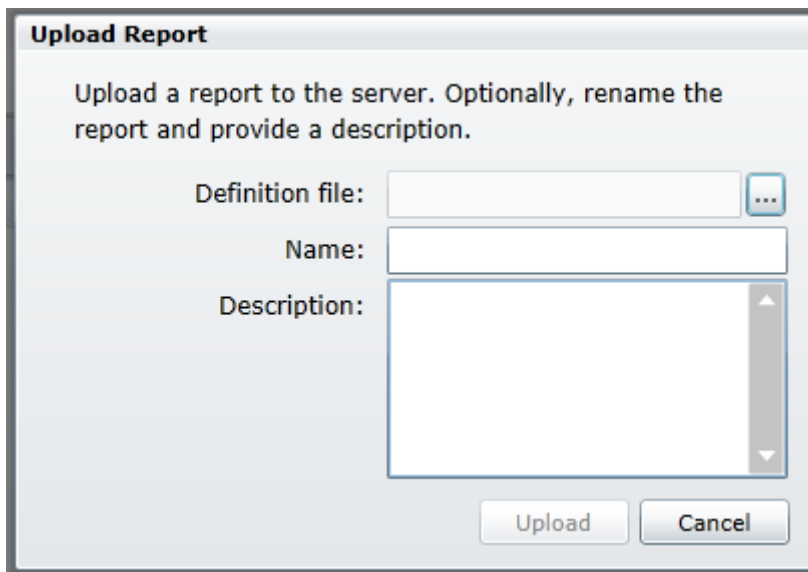


Figura 1.79.- Subir un informe

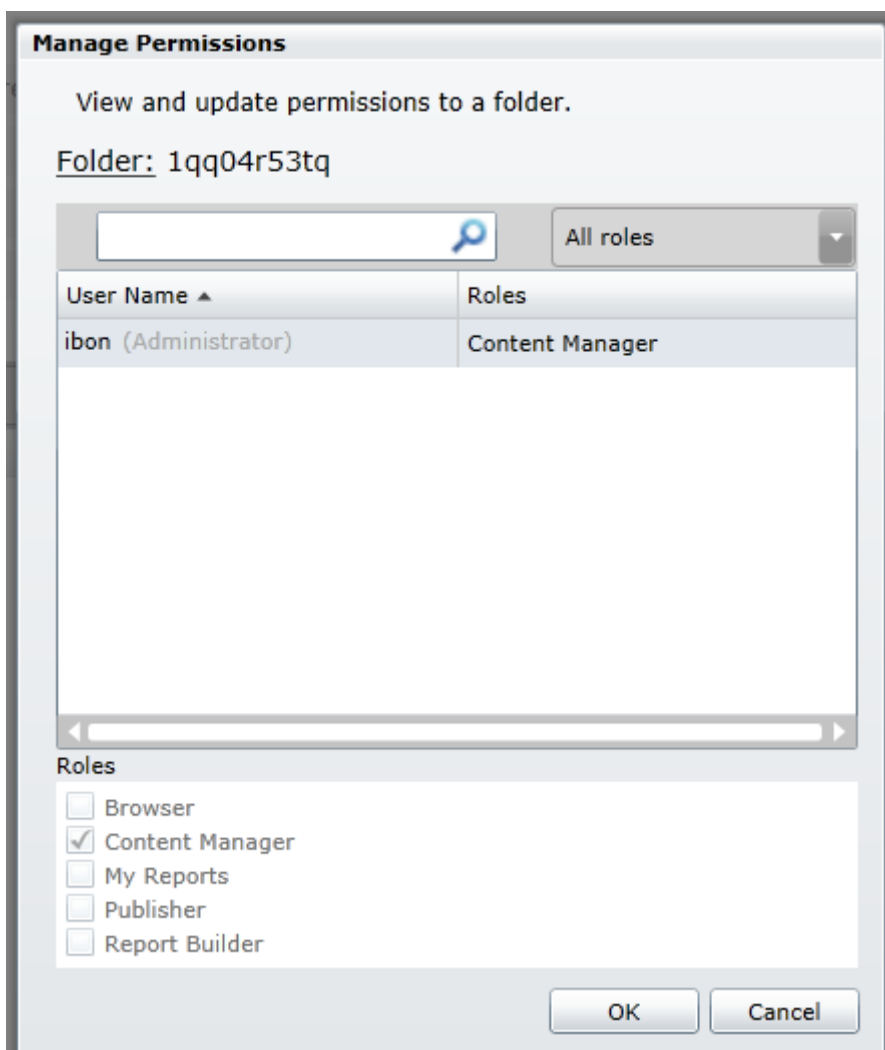


Figura 1.80.- Gestionar permisos sobre las carpetas



Figura I.81.- Acceso al Report Server

16.- SQL AZURE FEDERATION

Una de las nuevas funcionalidades que estarán disponibles en SQL Azure para finales de 2011 es la característica de federación.

Uno de los grandes beneficios de la plataforma Windows Azure es la capacidad que ofrece a la hora de escalar aplicaciones. De manera muy sencilla y dinámica un usuario puede modificar el número de instancias de una aplicación.

Hablando de instancias de Windows Azure este punto es cierto, pero la base de datos podría convertirse en un cuello de botella, ya que ésta no tiene la capacidad de escalado que tienen las instancias de Azure. El usuario no dispone de la posibilidad de realizar un escalado horizontal y aumentar o disminuir las bases de datos SQL Azure en función de las necesidades de la aplicación.

Por ejemplo, este es un escenario típico en aplicaciones multi-tenant, dónde con un mismo despliegue se quiere ofrece el servicio a diferentes clientes.

Al implementar soluciones de este tipo la primera aproximación es usar una única base de datos capaz de contener la información de todos los clientes.

Esta solución es perfectamente viable, pero en algunos escenarios surgen problemas de escalabilidad en caso de que la aplicación crezca en el número de clientes.

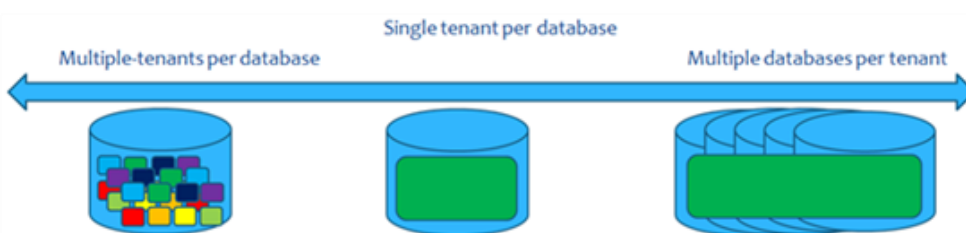


Figura I.82.- Opciones de escalado

Otra de las soluciones aportadas, buscando un alto grado de escalabilidad ha sido utilizar múltiples base de datos, en lugar de una única que contenga todos los datos de la aplicación.

La aplicación es capaz de gestionar múltiples clientes diferentes, cada uno con su información, pero a nivel de base de datos cada uno tiene su base de datos propia.

Este escenario es perfectamente viable, pero implica que si existen 1000 clientes existirán 1000 base de datos, lo que aumenta la complejidad de administración de la aplicación y claro está, los costes de despliegue.

SQL Azure Federation viene a cubrir este tipo de escenario, ya que esta característica simplificará enormemente el escalado horizontal, posibilitando que el usuario pueda aumentar o disminuir las bases de datos de la aplicación de forma dinámica y sin que este hecho provoque una parada del servicio.

Es el usuario quién decide los miembros de la federación, es quién decide cuándo añadir o disminuir miembros y es quién decide en base a qué criterio debe realizarse la partición de la información (en la primera versión sólo soportará la partición por rangos, por ejemplo, por cierto valor de un determinado campo).

La sintaxis de la sentencia para crear una federación es la siguiente:

```
CREATE FEDERATION federation_name { <federation_distribution_scheme>
}

<federation distribution scheme> ::= <federation distribution>

<federation distribution> ::=
(distribution_name <data_type> <distribution_type>)

<data type> ::=
[ system type name . ] type name
```

Dentro de la federación existe un elemento que se conoce como “Federation Root”. Este elemento representa el nombre lógico de la base de datos y es quién conoce todos los miembros de la federación y qué información tiene cada uno de ellos.

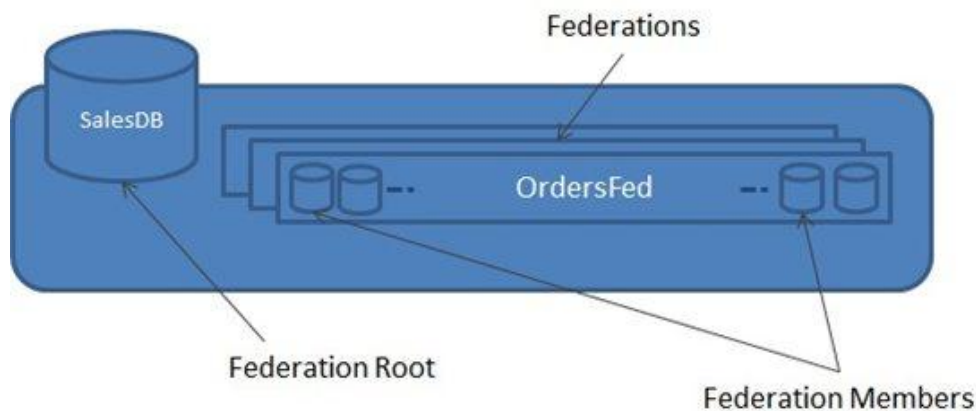


Figura 1.83.- SQL Azure Federation

Las aplicaciones siempre debe conectarse al “root”, y éste quién enruta las peticiones a la base de datos correspondiente.

Windows Azure Storage

Windows Azure además de los servicios de ejecución proporciona servicios de almacenamiento no relacional y colas con acceso autenticado, alta redundancia (triple) y accesible mediante una interfaz REST que se puede utilizar desde cualquier lenguaje que tenga la capacidad de realizar peticiones HTTP, que es tanto como decir cualquier lenguaje moderno.

En este capítulo se entrará a describir las diferentes funcionalidades que ofrece Windows Azure desde el punto de vista de almacenamiento.

I.-ALMACENAMIENTO EN WINDOWS AZURE

La plataforma Windows Azure, además de los servicios para ejecución de aplicaciones en la nube también proporciona servicios de almacenamiento. Este almacenamiento es conocido como Windows Azure Storage.

Estos servicios de almacenamiento están diseñados para ser consumido mediante REST de manera que estén totalmente accesibles desde cualquier lenguaje de programación con capacidad para hacer peticiones usando HTTP como protocolo, lo que es tanto como decir cualquier lenguaje.

En Windows Azure Storage existen los siguientes servicios:

- El servicio de Blobs (Blob Service)
- El servicios de Tablas (Table Service)
- El servicio de colas (Queue Service)
- Windows Azure Drive

Cada uno sirve a propósitos y necesidades diferentes, pero salvo el servicio de Drives, todos tienen algunas características en común:

- Son servicios diseñados para la alta disponibilidad y escalabilidad que las aplicaciones en la nube exigen.
- Son servicios diseñados para ser accesibles mediante peticiones HTTP o HTTPS y un API basada en REST.
- Nuestros servicios de almacenamiento siempre colgaran de una cuenta de almacenamiento de Windows Azure que define el namespace de nuestros servicios mediante una raíz URL única y común a los servicios que comparten cuenta.

2.-WINDOWS AZURE TABLES

El servicio de tablas de Windows Azure proporciona almacenamiento estructurado no relacional basado en tablas. Como todos los servicios de almacenamiento de Windows Azure proporciona un API REST.

En este caso el API REST de las tablas de Windows Azure sigue el contrato definido por ADO.NET Data Services, lo que proporciona una comodidad extra: es posible usar la librería cliente para ADO.NET Data Service para acceder a las tablas e incluso realizar consultas LINQ (con ciertas limitaciones) sobre las tablas de Windows Azure sin tener que trabajar directamente con REST. Es la librería cliente para ADO.NET Data Service quien se encarga de convertir las peticiones que haga la aplicación en las peticiones HTTP correspondientes.

2.1.- Entidades y tablas

Dentro de una cuenta de almacenamiento de Windows Azure es posible crear múltiples tablas. Las tablas se diferencian por su nombre.

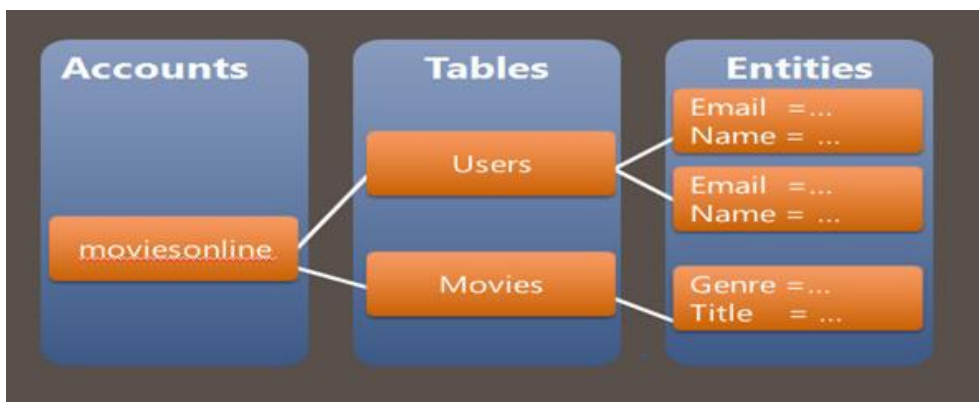


Figura 2.1.- Entidades y tablas

Dentro de las tablas se almacenarán entidades. Las entidades se representan en C# o en VB.Net como clases que derivan de la clase TableServiceEntity y que tiene una serie de propiedades públicas que serán almacenadas en la tabla cuando los objetos sean persistidos.

A continuación puede verse como sería una entidad Visita que tiene las propiedades Nombre, Lugar y Descripción.

```
public class VisitaEntity : TableServiceEntity
{
    public VisitaEntity()
    : base(Guid.NewGuid().ToString(), String.Empty)
    {
    }

    public VisitaEntity(string nombre, string lugar, string
descripcion)
    : base(lugar, nombre)
    {
        Nombre = nombre;
        Lugar = lugar;
        Descripcion = descripcion;
    }

    public string Nombre { get; set; }
    public string Lugar { get; set; }
    public string Descripcion { get; set; }
}
```

Las entidades son, desde el punto de vista de Azure Storage son colecciones de pares propiedad - valor, similares a las filas de una tabla en una base de datos relacional. Todas las tablas y por tanto todas las entidades tiene dos propiedades que siempre deben aparecer y que identifican de manera univoca al objeto y el lugar físico donde se almacena (partición).

La primera propiedad es la clave de la partición, `PartitionKey`, que identifica a que partición pertenece una entidad. La única garantía que existe sobre las particiones, es que, acceder a entidades almacenadas en la misma partición va a tener, típicamente, un menor coste que acceder a entidades en particiones diferentes.

Es importante poner cuidado a la hora de elegir la clave de partición de las entidades para asegurar que entidades que se acceden típicamente al mismo tiempo comparten la misma clave de partición. También es importante que las particiones sean homogéneas en tamaño. Por ejemplo, si se almacenan clientes, una posible clave de partición podría ser el código postal, si típicamente la aplicación trabaja con los clientes de un determinado código postal, por ejemplo para realizar estadísticas.

La segunda propiedad relevante es la clave de la entidad, `EntityKey`, que identifica de manera univoca una entidad dentro de una partición.

La combinación de `PartitionKey` más `EntityKey` identifica de manera única una entidad de manera global. Sería el equivalente a una clave única compuesta en el mundo relacional.

A diferencia de las tablas de las bases de datos relacionales una tabla puede contener entidades que tengan diferente número de propiedades. Además no existen conceptos relacionados con la integridad referencial, será la aplicación la encargada de mantener esta integridad de los datos.

Lógicamente antes de poder almacenar datos en una tabla se necesario crearla si es que no existe, algo muy fácil gracias al API de Azure.

```
var storageAccount =
    CloudStorageAccount.FromConfigurationSetting("DataConnectionString");
...

storageAccount.CreateCloudTableClient().CreateTableIfNotExist(DondeHasEstadoDataContext.VisitasTableName);
```

2.2.- Contexto de acceso a datos

Otro concepto importante a la hora de trabajar con las tablas de Windows Azure es el contexto.

El contexto es una clase que deriva de `TableServiceContext` y es el objeto que permite, usando LINQ, acceder a las tablas almacenadas en la cuenta de almacenamiento de Azure. Al contexto será necesario indicarle la URL de acceso al sistema de almacenamiento, URL que proporciona la plataforma Azure cuando se crea una cuenta de almacenamiento.

Esta es una típica clase de contexto de tabla:

```
public class DondeHasEstadoDataContext
    : TableServiceContext
{
    public const string VisitasTableName = "Visitas";

    public DondeHasEstadoDataContext(string baseAddress,
        StorageCredentials credentials) :
        base(baseAddress, credentials)
    {
    }

    public IQueryable<VisitaEntity> VisitasTable
    {
        get
        {
            return
                this.CreateQuery<VisitaEntity>(VisitasTableName);
        }
    }
}
```


Las clases de contexto proporcionan acceso a las tablas mediante LINQ devolviendo una implementación de IQueryable para la clase concreta de nuestra entidad. Devolver un IQueryable va a permitir realizar consultas LINQ sobre una tabla.

2.3.- Orígenes de datos

Por último, como siempre que se accede a datos, será necesario algún tipo de objeto que permita conectarse a la fuente de datos.

Estos objetos en Windows Azure Storage se llaman por convención 'data sources' y se utilizan para implementar en un solo objeto todas las operaciones de acceso a datos que se realizarán sobre las entidades almacenadas en una tabla. Al contrario que las clases anteriores, esta clase no tiene por qué derivar de ninguna otra, tal y como se puede ver en la siguiente porción de código:

```
public class VisitasDataSource
{
    private DondeHasEstadoDataContext dataContext = null;

    public VisitasDataSource()
    {
        var storageAccount =
        CloudStorageAccount.FromConfigurationSetting("DataConnectionString");
        dataContext = new
        DondeHasEstadoDataContext(storageAccount.TableEndpoint.ToString()
        storageAccount.Credentials);

        storageAccount.CreateCloudTableClient()
        .CreateTableIfNotExist(DondeHasEstadoDataContext.VisitasTableName);
    }

    public IEnumerable<VisitaEntity> Select()
    {
        var results = from v in dataContext.VisitasTable
        select v;

        return results.AsTableServiceQuery<VisitaEntity>().Execute();
    }

    public void Delete(VisitaEntity itemToDelete)
    {
        _dataContext.AttachTo(DondeHasEstadoDataContext.VisitasTableName,
        itemToDelete, "*");
        dataContext.DeleteObject(itemToDelete);
        dataContext.SaveChanges();
    }

    public void Insert(VisitaEntity newItem)
    {
        dataContext.AddObject(DondeHasEstadoDataContext.VisitasTableName, newItem);
        _dataContext.SaveChanges();
    }
}
```

2.4.- La cadena de conexión

Otro aspecto fundamental siempre que se habla de acceso a datos es establecer la cadena de conexión. Lo mismo ocurre en el almacenamiento de Windows Azure.

El SDK de Windows Azure proporciona un entorno simulado del almacenamiento de Azure, el Windows Azure Development Storage que corre en la máquina de desarrollo sobre SQL Server Express.

Para conectarse al Azure Development Storage se puede establecer la siguiente entrada en la sección de cadenas de conexión del archivo de configuración de la aplicación:

```
<connectionStrings>
  <add name="DataConnectionString"
  connectionString="UseDevelopmentStorage=true"/>
</connectionStrings>
```

Para obtener la cadena de conexión desde el código será tan simple como hacer:

```
var storageAccount =
CloudStorageAccount.FromConfigurationSetting("DataConnectionString");
_dataContext = new
DondeHasEstadoDataContext(storageAccount.TableEndpoint.ToString(), storageAccount.Credentials)
```

3.-WINDOWS AZURE BLOBS

El servicio de Blob de Windows Azure Storage proporciona almacenamiento para entidades binarias o archivos. Sería el almacenamiento adecuado cuando es necesario almacenar archivos de imágenes, sonidos, videos, archivos del tipo que sea, etc... El API de Blob es también un API basada en REST.

El servicio de Blob expone dos entidades fundamentales, contenedores y blobs. Todo blob debe alojarse dentro de un contenedor.

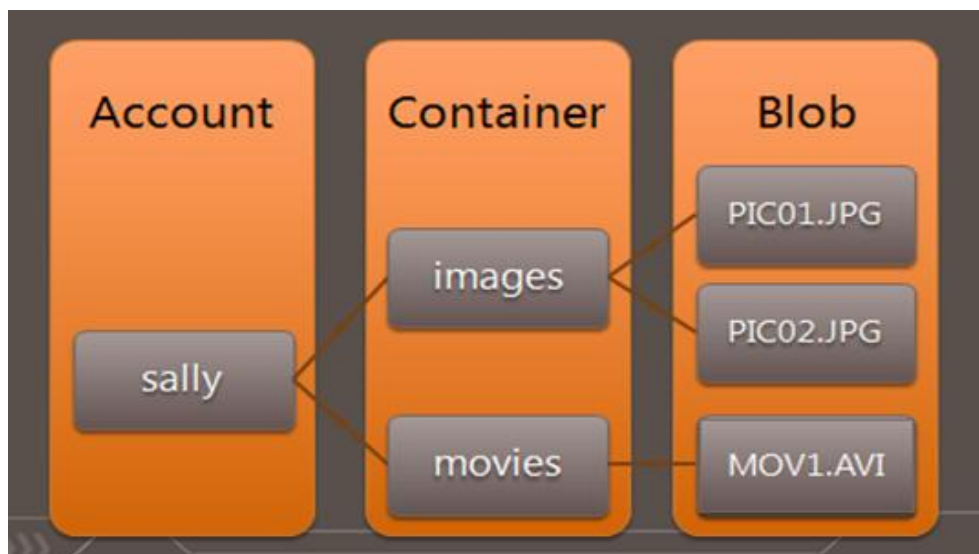


Figura 2.2.- Contenedores y Blobs

Existen dos tipos de blobs:

- Blobs de bloque (Block Blobs), que están optimizados para streaming.
- Blobs de página (Page Blobs), que están optimizados para el acceso aleatorio de escritura y lectura en lugares arbitrarios del blob.

Evidentemente en muchas ocasiones es necesario ser capaces de distinguir unos blobs de otros una vez almacenados, por ello los blobs soportan que se asocie metadatos con ellos.

Usando el API del servicio de Blob, los desarrolladores pueden crear una jerarquía de namespaces similar a un sistema de archivos. Los nombre de los Blobs pueden seguir una codificación que use un separador de 'directorios' configurable (típicamente /) de manera que llamar a un Blob Imagenes/MiImagen1 y a otro Sonidos/MiSonido1 implique una organización a nivel lógico de los Blobs almacenados.

El API de Azure Storage va a permitir operaciones de enumeración de tal manera que puedan enumerarse todos los Blobs que se encuentran dentro de la jerarquía Imágenes.

3.1.- Trabajo con Blobs

Para trabajar con Blobs el primer paso es establecer una conexión con el servicio de almacenamiento. Para realizar esta acción basta con obtener la cadena de conexión del fichero de configuración:

```
<connectionStrings>
  <add name="DataConnectionString"
  connectionString="UseDevelopmentStorage=true"/>
</connectionStrings>
```

Y abrir una conexión con el siguiente código que usa la clase `CloudStorageAccount` para desde una cadena de conexión obtener un `BlobStorageClient` que permitirá luego manipular los Blobs disponibles en Windows Azure Storage:

```
// Establecer la conexión con Windows Azure

var storageAccount =
CloudStorageAccount.FromConfigurationSetting("DataConnectionString");

BlobClient = storageAccount.CreateCloudBlobClient();
```

Una vez se dispone de un `BlobStorageClient`, lógicamente será necesario añadir un nuevo contenedor de Blobs. Tarea simple gracias a la clase `CloudBlobContainer` y su método `CreateIfNotExists`, tal y como podéis ver en el siguiente código:

```
// Obtener el contenedor y clearlo si no existe

BlobContainer = BlobClient.GetContainerReference("fotos");

BlobContainer.CreateIfNotExists();
```

Los contenedores de Blobs pueden ser públicos o privados. En este caso se ha creado un contenedor público. Si el contenedor fuese privado, solo conociendo la clave de nuestro almacenamiento y la cuenta de Azure Storage se podría acceder. Es importante recordar que cualquiera puede acceder a un contenedor público.

Subir el contenido de un Blob al almacenamiento es una tarea sumamente simple. Basta con obtener una referencia al Blob, invocar al método `UploadFromStream` y establecer las propiedades y metadatos del blob, tal y como puede verse en el siguiente código:

```
// Creamos un nombre único para el blob y creamos una referencia a
ese blob

var blob = BlobContainer.GetBlobReference(Guid.NewGuid().ToString()
+ Path.GetExtension(FUPhoto.FileName));

//Subimos el archivo al almacenamiento
```

```

blob.UploadFromStream(FUPhoto.FileContent);

// Establecemos los metadatos del blob
blob.Metadata["Name"] = TBPhotoName.Text;
blob.Metadata["Author"] = TBAuthor.Text;
blob.SetMetadata();

// Establecemos el tipo de contenido del blob
blob.Properties.ContentType = FUPhoto.PostedFile.ContentType;
blob.SetProperties();

```

Para enumerar el contenido de un contenedor de Blobs se puede hacer tras obtener una referencia al mismo, y llamar al método `ListBlobs`. Tras enumerar los Blobs es posible ir accediendo a las propiedades y metadatos de cada uno de ellos, pero para ello primero es necesario llamar a `FetchAttributes` para forzar la recogida de los metadatos asociados al Blob.

A continuación puede verse estas operaciones en la siguiente porción de código:

```

// Obtenemos un lista de todos los blobs
var blobs = BlobContainer.ListBlobs();
var photos = new List<PhotoMetadata>();

//Para cada Blob que almacena una foto, obtenemos los metadatos de la
foto
//para mostrarlos en un grid.
foreach (var blobItem in blobs)
{
    var cloudBlob =
    _BlobContainer.GetBlobReference(blobItem.Uri.ToString());
    cloudBlob.FetchAttributes();

    photos.Add(new PhotoMetadata(blobItem.Uri,
    cloudBlob.Metadata["Name"], cloudBlob.Metadata["Author"]));
}

```

Por último eliminar un Blob es tan simple como llamar al método `DeleteIfExists` sobre la referencia a un Blob, tal y como puede verse a continuación:

```

// Obtener la referencia al blob y eliminarlo
var blob = BlobContainer.GetBlobReference(blobUri);
blob.DeleteIfExists();

```

4.-WINDOWS AZURE QUEUE

El servicio de colas de Windows Azure proporciona un mecanismo fiable y persistente para la comunicación entre aplicaciones de Windows Azure. El API de colas de Windows Azure, construido también sobre REST, está basado en dos tipos de abstracciones: colas y mensajes.

Las colas soportan atributos que se especifican como claves pares valor y que nos permiten asociar metadatos a las colas de manera que puedan identificarse o mantenerse datos asociados a las mismas.

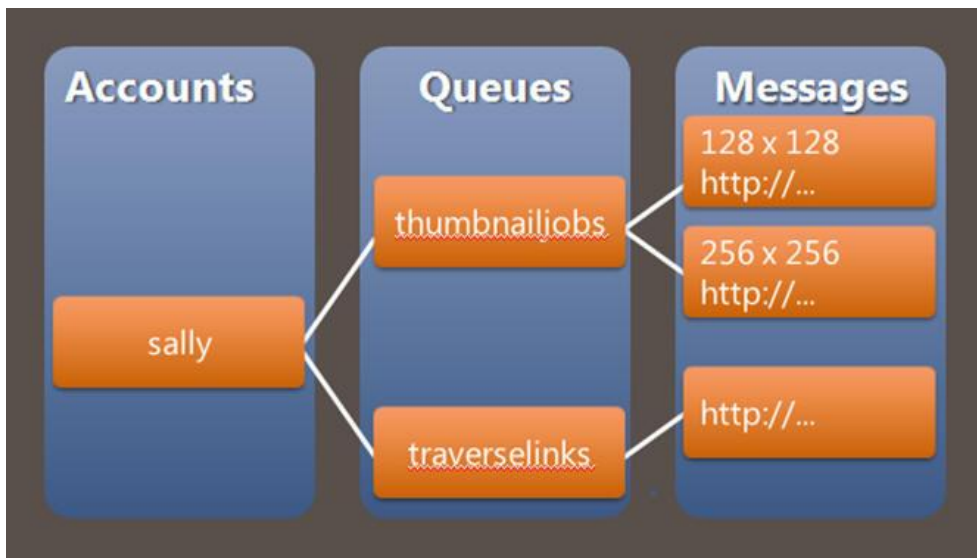


Figura 2.3.- Colas y mensajes

Cada cuenta de almacenamiento puede contener un número ilimitado de colas de mensajes y cada cola puede contener un número de mensajes ilimitado. La principal limitación en las colas de Windows Azure viene marcada por el hecho de que el tamaño máximo de mensaje es 8Kb.

Cuando un mensaje se lee desde una cola, el consumidor del mensaje es responsable de, tras procesarlo, eliminarlo. Una vez el mensaje es leído, durante un periodo de tiempo no estará disponible para otros consumidores. Si el mensaje no es borrado en ese intervalo de tiempo, su visibilidad es restablecida de manera que otros consumidores pueden procesarlo.

4.1.- Trabajo con colas

Para utilizar el servicio de colas de Windows Azure el primer paso que debe darse es conectar a una cola. Lógicamente se deberá crear la cola si no existe. El proceso es similar al que se realiza para otros tipos de almacenamiento.

El método `FromConfigurationSetting` de la clase `CloudStorageAccount` permite configurar la conexión al almacenamiento de Azure.

Luego es posible crear un cliente para el almacenamiento de colas llamando a `CreateCloudQueueClient`. El método `GetQueueReference` del `CloudQueueClient` permite obtener una referencia a la cola a través de la que llamaremos al método `CreateIfNotExists`, que creará la cola si es necesario.

Puede verse el proceso en el siguiente ejemplo de código:

```
// Establecer la conexión con Windows Azure Storage
var storageAccount =
CloudStorageAccount.FromConfigurationSetting("DataConnectionString");

//Obtener un cliente para el servicio de colas
_queueClient = storageAccount.CreateCloudQueueClient();
```

```
//Obtener una referencia a la cola y crearla si no existe  
  
Queue = QueueClient.GetQueueReference("cola");  
  
Queue.CreateIfNotExist();
```

El siguiente paso es enviar un mensaje a la cola. Para enviar un mensaje a la cola solo será necesario llamar al método `AddMessage` a través de la referencia a la cola.

```
//Añadimos el mensaje a la cola  
  
_Queue.AddMessage(new CloudQueueMessage(GetNumberOfPhotos()));
```

El método `AddMessage` permite enviar un mensaje como texto o como un array de bytes.

Nota: No debe olvidarse nunca que el código que consume y procesa el mensaje es el encargado de eliminar el mensaje de la cola.

El proceso de consumir un mensaje es muy simple. Basta con llamar al método `GetMessage` de la referencia a la cola, si hay algún mensaje el método devolverá el mensaje, sino `null`. Para acceder al contenido del mensaje se puede usar la propiedad `AsString` (si el mensaje se envió como un string) o `AsBytes` (si el mensaje se envió como un array de bytes).

```
//Vemos si hay un mensaje en la cola  
//Si lo hay leemos el número de fotos  
CloudQueueMessage message = Queue.GetMessage();  
  
if (message != null)  
{  
    int numberOfPhotos = int.Parse(message.AsString);  
  
    //Procesamos el mensaje  
  
    ...  
  
    //Borramos el mensaje  
  
    _Queue.DeleteMessage(message);  
}
```

5.-WINDOWS AZURE DRIVE

Windows Azure Drive permite a las aplicaciones desplegadas en Windows Azure montar unidades de disco NTFS, facilitando enormemente la migración de aplicaciones a Azure que tengan dependencias de una unidad de disco.

Las aplicaciones que se ejecutan en la nube pueden hacer uso de las API existentes NTFS para acceder a un almacenamiento persistente. Es importante recordar que un rol de Windows Azure dispone de un sistema de almacenamiento, pero que éste no es persistente ni es posible que sea compartido entre diferentes roles.

Con Windows Azure Drives es posible disponible de un sistema persistente y que puede ser compartido, accesible a través de las APIs de NTFS, es decir, como si se accediera a una unidad de disco.

Esta "unidad de red" realmente es un Page Blob de Windows Azure Storage que contiene un fichero en formato VHD.

Esta característica puede ayudar en la migración de aplicaciones a la nube, permitiendo a los clientes una experiencia de migración más uniforme al mismo tiempo reducir la cantidad de tiempo que se necesita para mover sus aplicaciones desde el propio entorno de Windows en un entorno Windows Azure.

El SDK de Windows Azure Drive ofrece diversos métodos para realizar tareas como crear una unidad, montarla, copiarla e incluso hacer un snapshot.

A continuación se mostrará un ejemplo muy sencillo de cómo hacer uso del API de Windows Azure Drive.

El primer paso es obtener una cuenta de acceso al sistema de almacenamiento de Windows Azure. En el entorno de desarrollo puede emplearse el Development Storage. Una vez creada la cuenta será necesario inicializar una caché en la instancia dónde resida el rol.

```
CloudStorageAccount storageAccount = CloudStorageAccount.DevelopmentStorageAccount;

LocalResource localCache = RoleEnvironment.GetLocalResource("InstanceDriveCache");
CloudDrive.InitializeCache(localCache.RootPath, localCache.MaximumSizeInMegabytes);
```

Una vez se dispone la cuenta es necesario obtener un objeto CloudBlobClient que permite tener acceso al contenedor del Page Block dónde se almacenará el fichero VHD.

Nota: Es importante tener en cuenta, que aunque una unidad de Drives se almacene como un Page Blob en Windows Azure Storage, en el Development Storage no funciona de la misma manera. Cuando se emplea el Development Storage una unidad de Drive se simula empleando una unidad de disco.

```
CloudBlobClient blobClient = storageAccount.CreateCloudBlobClient();
var container = blobClient.GetContainerReference("vhd");
var vhdBlob = container.GetBlobReference("MyDrive.vhd");
```

Para crear la unidad será necesario llamar al método CreateCloudDrive, indicándole la URI del blob que se utilizará como unidad.

```
CloudDrive myCloudDrive = storageAccount.CreateCloudDrive(
    blobClient
    .GetContainerReference("vhd")
    .GetPageBlobReference("MyDrive.vhd")
    .Uri.ToString()
);
```

Y llamar al método Create, que creará la unidad con un tamaño de 64 Mb.

```

try
{
    myCloudDrive.Create(64);
}
catch (CloudDriveException ex)
{
    // handle exception here
    // exception is also thrown if all is well but the drive already exists
}

```

Una vez creada la unidad será necesario "montarla", para disponer de una unidad, por ejemplo "X:\", a través de la cual se puede acceder al contenido del fichero VHD.

```

string driveLetter = string.Empty;

try
{
    driveLetter = myCloudDrive.Mount(localCache.MaximumSizeInMegabytes,
    DriveMountOptions.None);
}
catch (Exception)
{
}

```

Una vez montada la unidad, se puede acceder a la unidad empleando los sistemas habituales de acceso a disco. Si se escribe algo dentro de la unidad se está escribiendo dentro del fichero VHD, siendo este almacenamiento persistente entre varias ejecución de la aplicación.

```

string[] files = System.IO.Directory.GetFiles(driveLetter, "*.");
if (files.Length > 0)
    Label1.Text = System.IO.File.ReadAllText(files[0]);
else
{
    Label1.Text = "No files found";
    System.IO.File.WriteAllText(driveLetter + "\\fichero.txt", "hola!!");
}

```

Y una vez se ha terminado el proceso, puede "desmontarse" la unidad.

```

myCloudDrive.Unmount();

```

Aunque en el ejemplo que se acaba de mostrar se crea el fichero VHD durante la ejecución de la aplicación, un escenario más real sería partir de un fichero VHD previamente creado y con el contenido que necesite la aplicación.

En este caso la aplicación sólo debería montar la unidad y hacer uso de ella.

6.-DEL COMPUTE EMULETOR A LA NUBE

El SDK de Windows Azure proporciona un entorno de simulación local para el sistema de almacenamiento. Durante la fase de desarrollo y pruebas es conveniente trabajar con este almacenamiento de Azure local proporcionado por el Development Storage y construido sobre SQL Server.

El siguiente paso lógico es subir el almacenamiento a la nube. Este proceso es muy sencillo y consiste en dos pasos bien simples:

El primero crear una nueva cuenta de almacenamiento en la nube a través del portal de Azure. Una vez creado el servicio de almacenamiento a través del portal, éste proporcionará la URI de acceso al mismo, junto con la clave necesaria para poder conectarse.

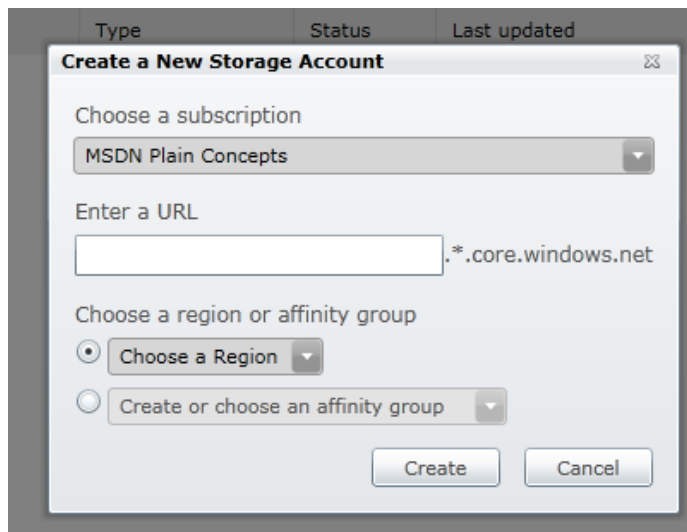


Figura 2.4.- Ventana principal del portal

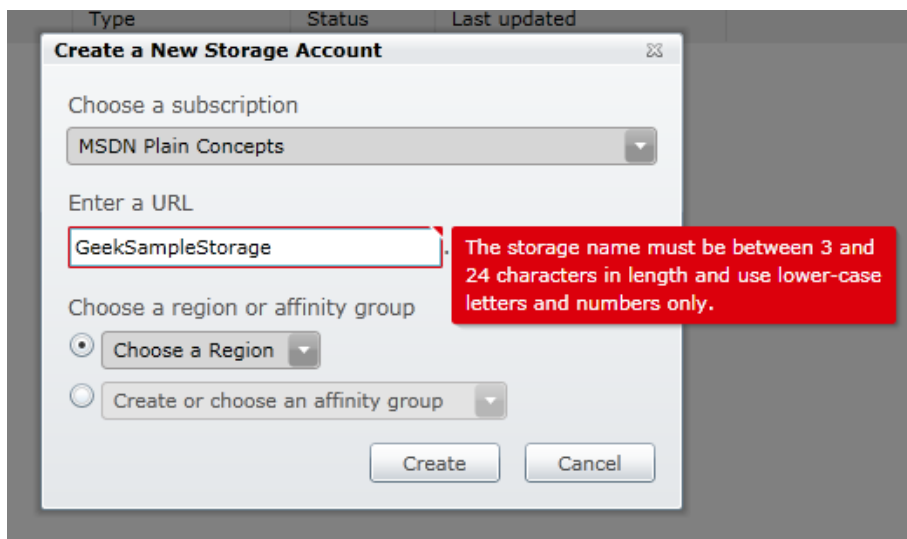


Figura 2.5.- Ventana principal del portal

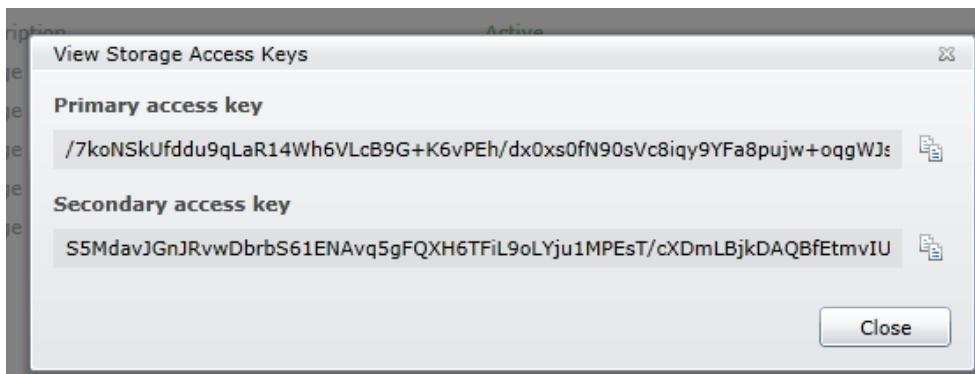


Figura 2.6.- Información de conexión a Windows Azure Storage

El segundo paso es cambiar la cadena de conexión de la aplicación al almacenamiento de Azure para que en lugar de usar el almacenamiento local utilice nuestra cuenta en la nube.

Cuando se emplea el entorno de simulación la cadena de conexión es como la que puede verse a continuación:

```
<Setting name="DataConnectionString" value="UseDevelopmentStorage=true" />
```

Será necesario cambiar la cadena de conexión para que apunte al almacenamiento real:

```
<Setting name="DiagnosticsConnectionString"
value="DefaultEndpointsProtocol=https;AccountName=ElNombreDeTuCuenta;AccountKey=" />
```

DefaultEndpointsProtocol puede tomar https o http como valor y permite especificar que protocolo queremos usar para el acceso a la cuenta de almacenamiento. Típicamente será Https por motivos de seguridad.

AccountName, será el nombre con el que se ha creado la cuenta de almacenamiento en el portal de Azure.

AccountKey, será la clave que el portal de Windows Azure proporciona al crear la cuenta de almacenamiento.

Desde Visual Studio, desde las opciones de configuración de rol, puede cambiar la cadena de conexión de una forma muy sencilla:

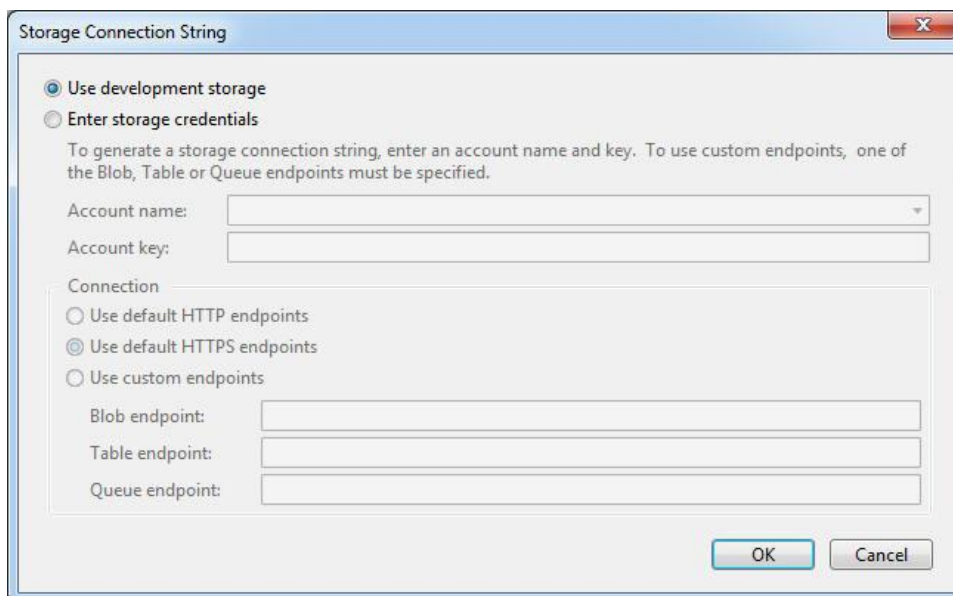


Figura 2.7.- Configurar la cadena de conexión

7.-WINDOWS AZURE MMC

Windows Azure Management Tool es una herramienta gratuita diseñada y creada para poder administrar los servicios y el sistema de almacenamiento de una cuenta de Windows Azure.

Esta herramienta provee de interesantes características que pueden simplificar enormemente el trabajo con Windows Azure:

- Permite realizar despliegues de servicios, actualizarlos y administrarlos.
- Permite configurar el sistema de diagnóstico de servicios hospedados en Windows Azure.
- Permite administrar los certificados disponibles en los servicios.
- Permite configurar las cuentas de almacenamiento.
- Permite la gestión de Blobs de Windows Azure; crear y eliminar contenedores, subir, eliminar o descargar blobs.
- Permite la gestión de Tablas de Windows Azure; consultar y borrar colas.
- Permite la gestión de las colas de Windows Azure; crear colas, purgarlas y borrar mensajes.

A continuación se muestran algunas imágenes de cómo es la herramienta.

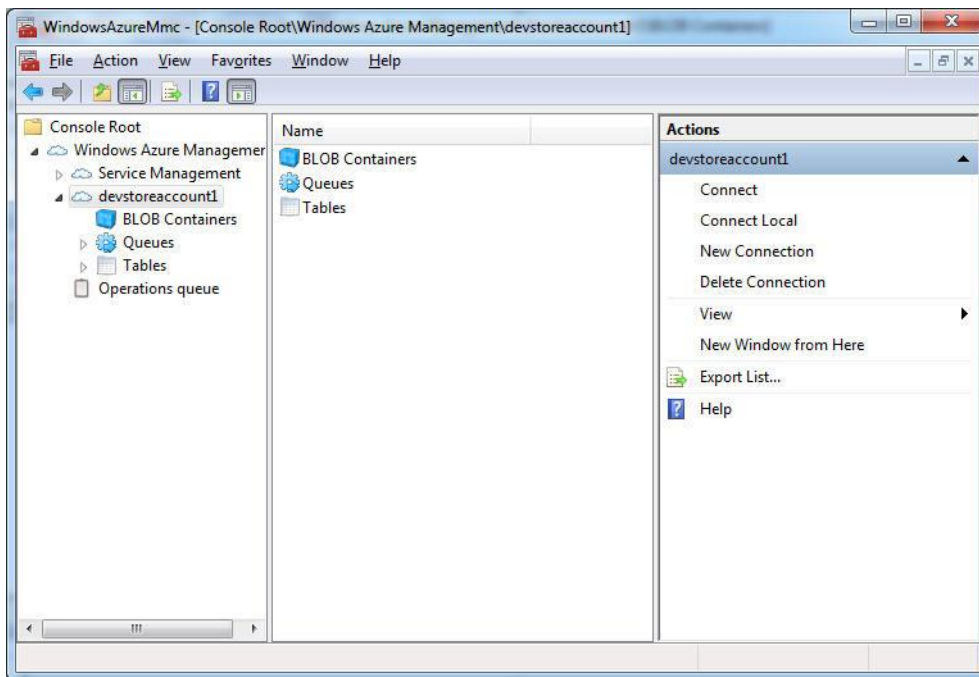


Figura 2.8.- AppFabric Service Bus

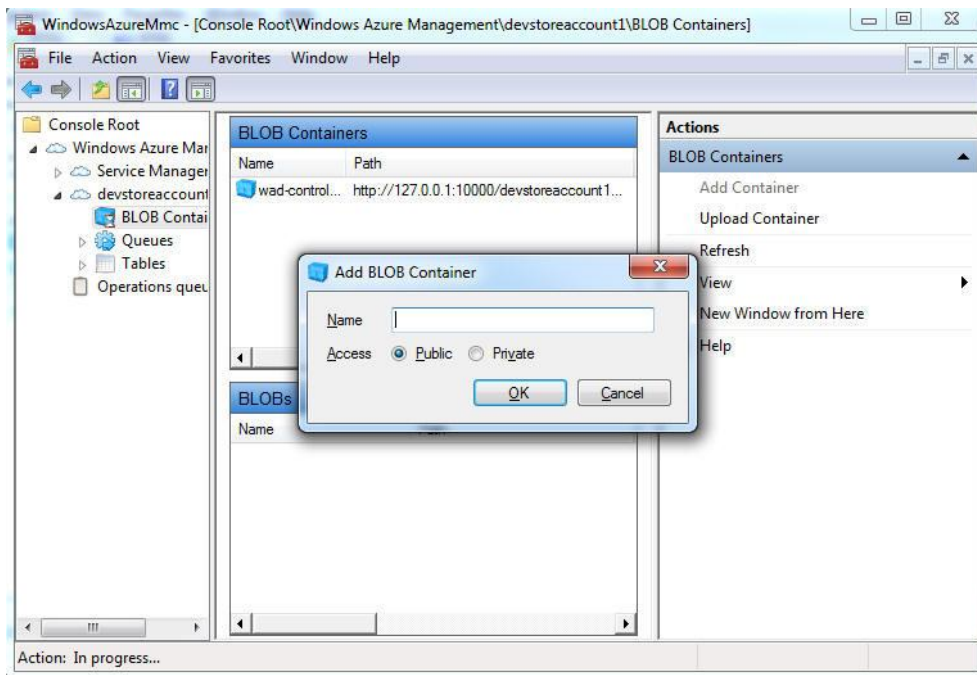


Figura 2.9.- AppFabric Service Bus

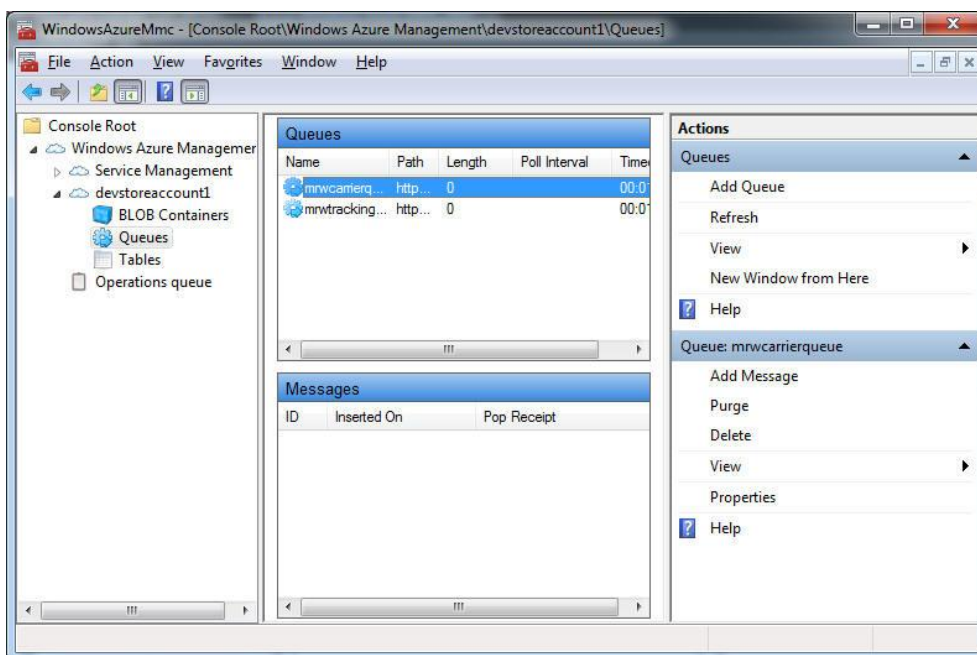


Figura 2.10.- AppFabric Service Bus

8.-CLOUD STORAGE STUDIO

Disponer de buenas herramientas en un punto clave para ser productivo en nuestro día a día y como no podía ser de otra manera, para trabajar con Windows Azure ocurre exactamente lo mismo.

Windows Azure es una plataforma abierta y dispone de APIs que permiten a partners y desarrolladores realizar nuevas herramientas que complementen a las ya existentes.

Cloud Storage Studio es una herramienta desarrollada por la empresa Cerebrata que permite trabajar de forma cómodo tanto con Windows Azure Storage como con las aplicaciones hospedadas en Windows Azure.

Cloud Storage Studio es una herramienta que requiere disponer de una licencia y que ofrece una variedad de funcionalidad relacionada con Windows Azure:

- Permite conectarse a una cuenta de Windows Azure Storage y administrar las tablas, blobs y colas.
- Permite conectarse y administrar el Development Storage.
- Permite crear tablas, eliminarlas, crear entidades, consultarlas...
- Permite gestionar contenedores de blobs, crear blobs; crear, eliminar, modificar, copiar, renombrar etc...
- Administrar colas de Azure; crear, actualizar, borrar etc...
- Administrar aplicaciones hospedadas;
- Hacer despliegues, actualizarlos, borrarlos, ver el estado de un servicio etc...
- Desplegar en producción, preproducción, hacer paso de preproducción a producción.
- Hacer actualizaciones de los servicios; swaping, upgrades manuales, upgrades automáticos...

A continuación se muestran algunas imágenes de cómo es la herramienta.

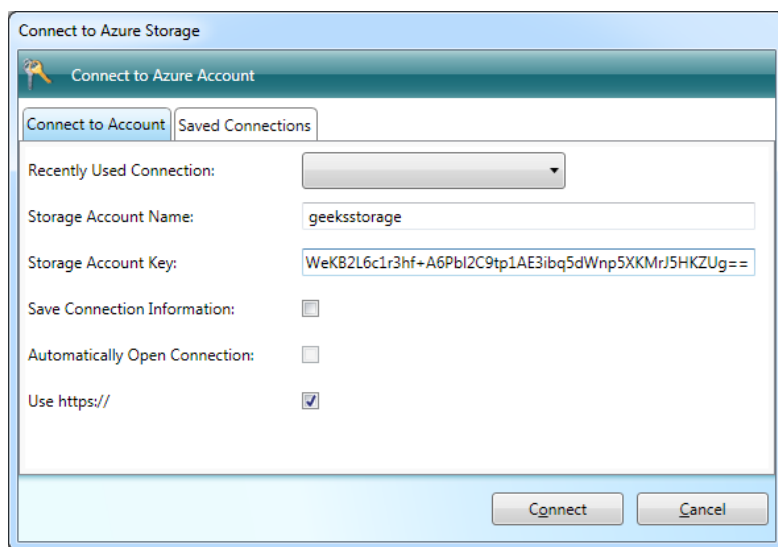


Figura 2.11.- Pantalla de conexión al servicio de almacenamiento Azure

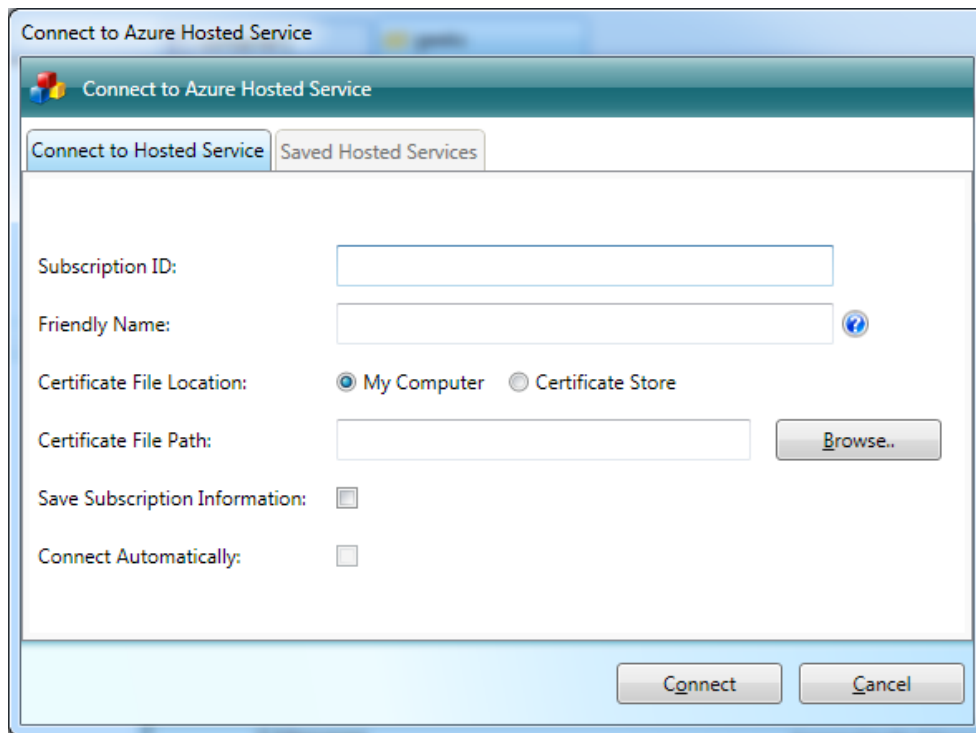


Figura 2.12.- Pantalla de conexión al sistema de ejecución de Windows Azure

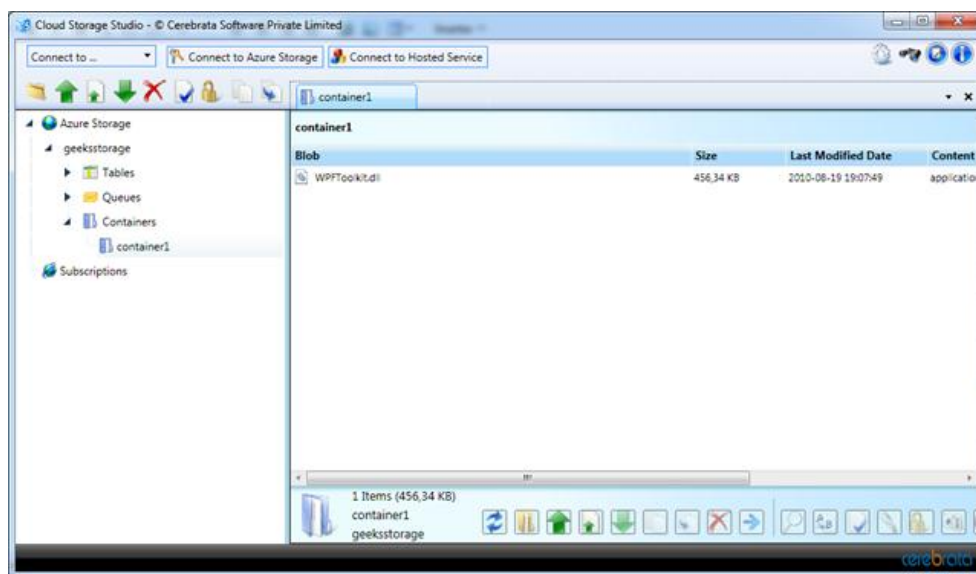


Figura 2.13.- Pantalla de visualización de los componentes de Windows Azure Storage.

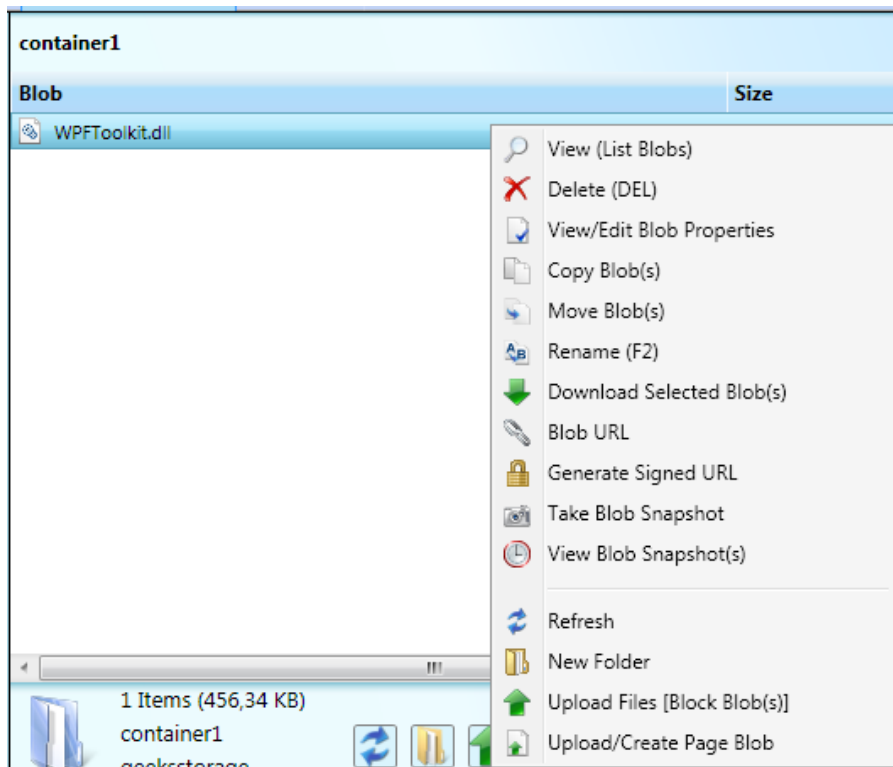


Figura 2.14.- Acciones posibles a realizar sobre un Blob.

Por ejemplo, también se podrían crear tablas y subir entidades a esas tablas.

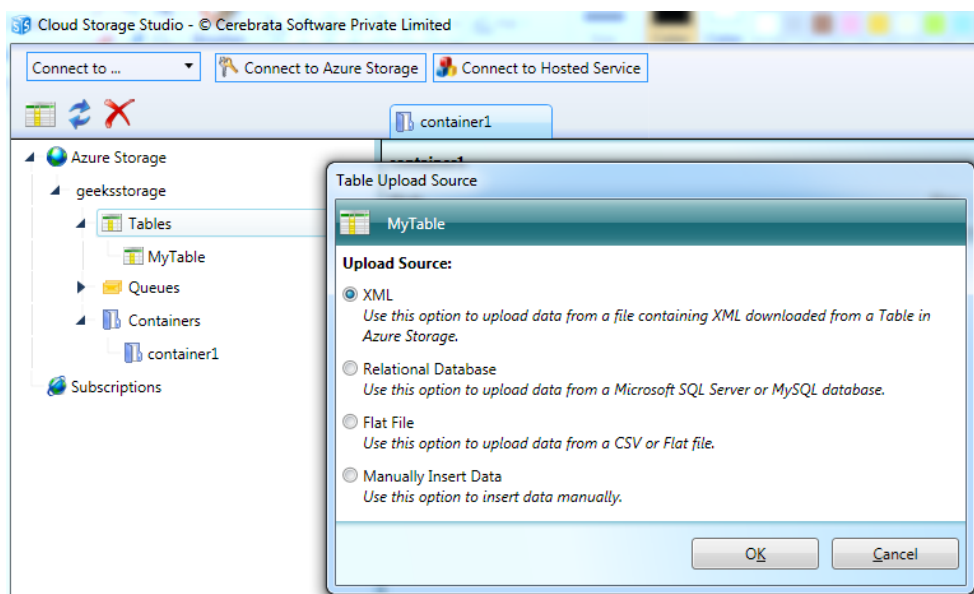


Figura 2.15.- Creación de tablas y subida de entidades

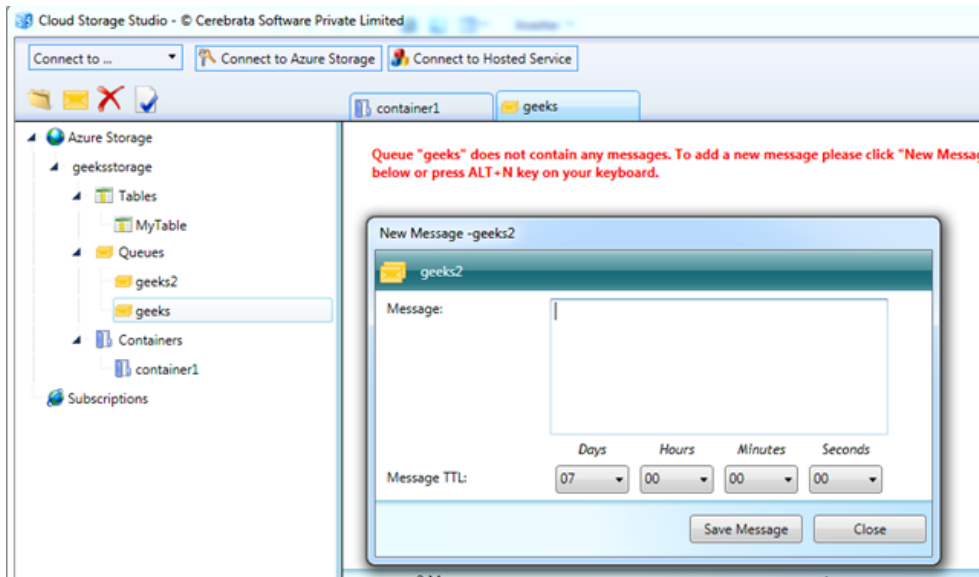


Figura 2.16.- Pantalla de administración de colas

9.-WINDOWS AZURE CDN

Un Content Delivery Network es una red de servidores en ubicaciones geográficas diferentes para la distribución de contenido.

La ventaja de este tipo de red es que al estar compuesta por multitud de servidores en diferentes ubicaciones geográficas cuando se realiza una petición al CDN, éste busca el servidor que está más cerca del usuario que hace la petición y utiliza ese servidor para responder al cliente.

A través de Windows Azure Storage se puede disponer de esta característica dentro de la plataforma Windows Azure. Desde el portal de Windows Azure, desde la configuración de un servicio de almacenamiento puede activarse esta característica.

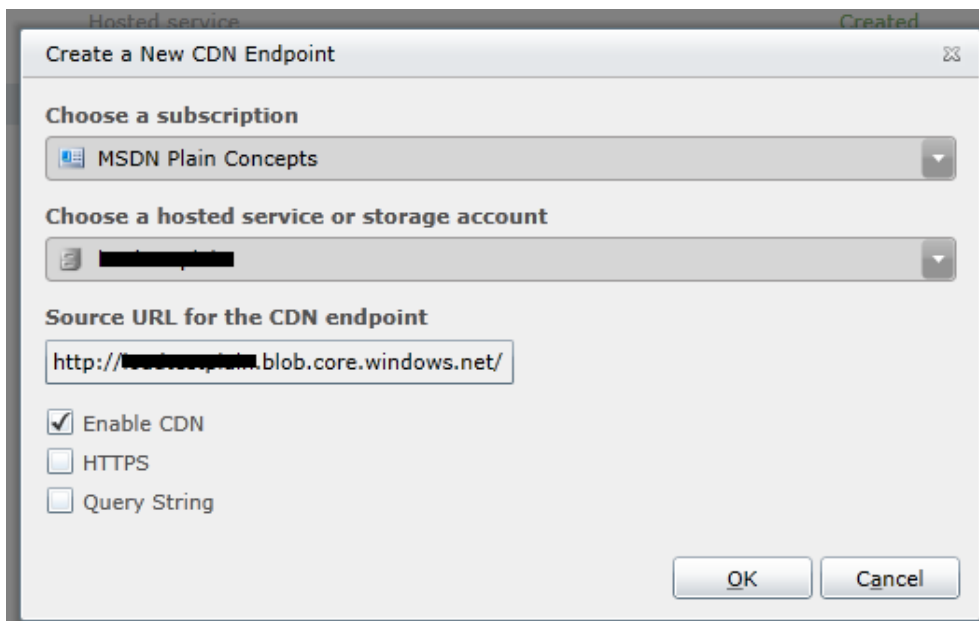


Figura 2.21.- Activar la característica de CDN

Esta característica puede de ser de gran utilidad en aplicaciones que deben estar accesibles desde ubicaciones diferentes y que tenga un componente de descarga que puede afectar al rendimiento.

No sólo hay que pensar en una aplicación típica de descargas, un ejemplo habitual podría ser una aplicación Silverlight.

Al acceder a la aplicación Web hecha con Silverlight, éste componente debe descargarse al equipo que realiza la petición para poder ejecutarse. Si por ejemplo la aplicación es accesible desde diferentes países siempre obtendrá un mejor rendimiento cuando más cerca esté el servidor web. Es un escenario tradicional, el componente Silverlight (xbap) residirá en el Web Role dónde esté desplegada la aplicación, junto con el resto de la aplicación web; html, aspx etc...

Otra posible solución podría ser incluir el componente Silverlight (xbap) dentro de Windows Azure Storage, como un blob, activando la característica de Content Delivery Network. De esta forma el componente Silverlight se distribuiría por la red de servidores.

La aplicación web, que reside en un Web Role, dentro de la página aspx o html haría referencia al componente Silverlight a través de la URI única a través de la cual se puede acceder al contenido de Windows Azure Storage.

De esta forma, aunque la aplicación web resida dentro de un determinado Data Center, la descarga del componente Silverlight se haría desde la red de servidores consiguiendo de esta manera un mejor rendimiento.

10.-WINDOWS AZURE GEO-REPLICATION

Una de las últimas novedades de la plataforma es la característica de geo-replicación. Con esta característica la información almacenada en los blob y tablas del storage se almacenarán de forma transparente y sin coste adicional en otro datacenter de la misma región.

Aunque a nivel del propio datacenter Windows Azure Storage ya dispone de mecanismos para evitar la pérdida de información, con esta característica ganamos todavía mucho más desde el punto de vista de recuperación ante desastres, ya que la información se duplicará en un datacenter diferente al que hayamos elegido en la creación del storage.

Cuando se crea el storage se elige la ubicación para el storage, siendo Windows Azure de forma automática el que elegirá la ubicación del datacenter de respaldo. En esta primera versión hay una relación 1 a 1 entre el datacenter primario y el secundario, el que hace de respaldo. En un futuro se podrá elegir cuál será el que haga de secundario e incluso se podrá forzar un intercambio de roles.

Tabla 2.1.- Relación entre datacenters

Principal	Secundario
North Central US	South Central US
South Central US	North Central US
North Europe	West Europe
West Europe	North Europe
South East Asia	East Asia
East Asia	South East Asia

La operación de sincronización es un proceso asíncrono, transparente para el usuario y que no implica una pérdida de rendimiento en las operaciones que se hagan con el storage. Si ocurriese un fallo el primer punto de restauración sería la que se encuentra el mismo datacenter, si el datacenter principal se viese comprometido de forma transparente el secundario pasaría a darnos el servicio.

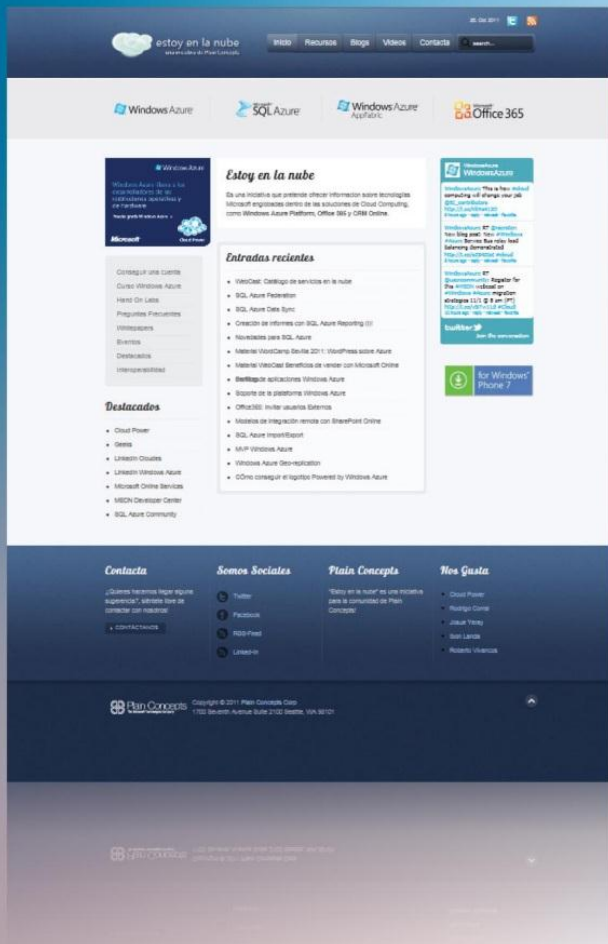
Este paso es transparente para la aplicación porque únicamente se realiza una actualización del DNS para que las URLs que usan la aplicación apunten al datacenter secundario en lugar de al primario.

Por último, aunque la opción es gratuita, podría deshabilitarse esta característica contactando con el soporte de Windows Azure.

estoy en la nube

INICIATIVA DE PLAIN CONCEPTS

www.estoyenlanube.com



 Windows Azure

www.plainconcepts.com

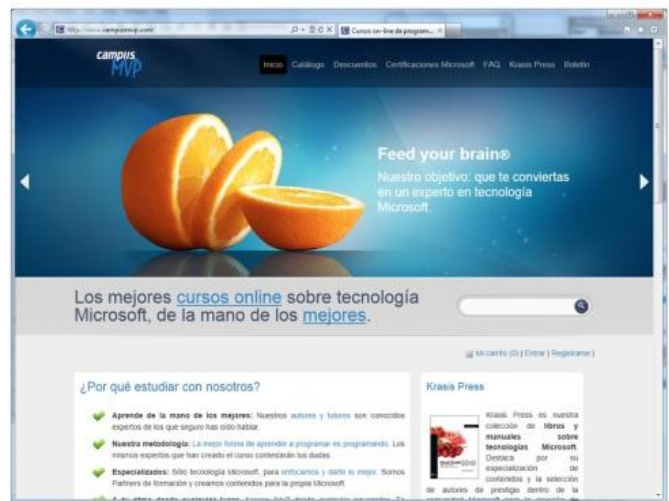
Plain Concepts is a company specialized in Microsoft technologies, agile methodologies, Application Lifecycle Management, performance tuning, advanced debugging, software architecture and User Experience.

Plain Concepts focuses on delivering high quality consulting, mentoring and training as well as in being an effective and reliable team resolving all type of software development issues.

¿Aún quieres más?

**campus
MVP**

Formación online especializada
en tecnologías Microsoft.



**krasis
PRESS**

Los libros que lo saben todo sobre
tecnologías Microsoft.

Síguenos y descubrirás los mejores trucos y recursos:

 facebook.com/campusmvp  twitter.com/campusmvp

 **feed your brain®**

- ☑ Sin tener que desplazarse
- ☑ Sin romper el ritmo de trabajo
- ☑ Preguntándole a los que más saben

infórmate ya:

902 876 475
www.campusmvp.com

<http://www.krasis.com>



krasis

Microsoft Partner
Silver Learning
Silver Software Development

