Denis Dwornitzak
denis.dwornitzak@studentpartners.de

A cloudy approach – SQL Data Services

Imagine that you have your own company. Business is going on well, your enterprise grows. But what would your networking infrastructure look like? There are a lot of business challenges, like growth, flexibility, storage and innovation which needs the appropriate infrastructure. Today, there is an approach called cloud computing which seems to be the answer for these challenges. Such a concept needs not only the correct hardware, but also proper software –SQL Data Services (SDS).

Cloud Computing

Cloud computing is a new way of supporting applications. In the concept of cloud computing there is a cloud platform which lets developers write applications that run in the cloud, or use services provided from the cloud, or both.

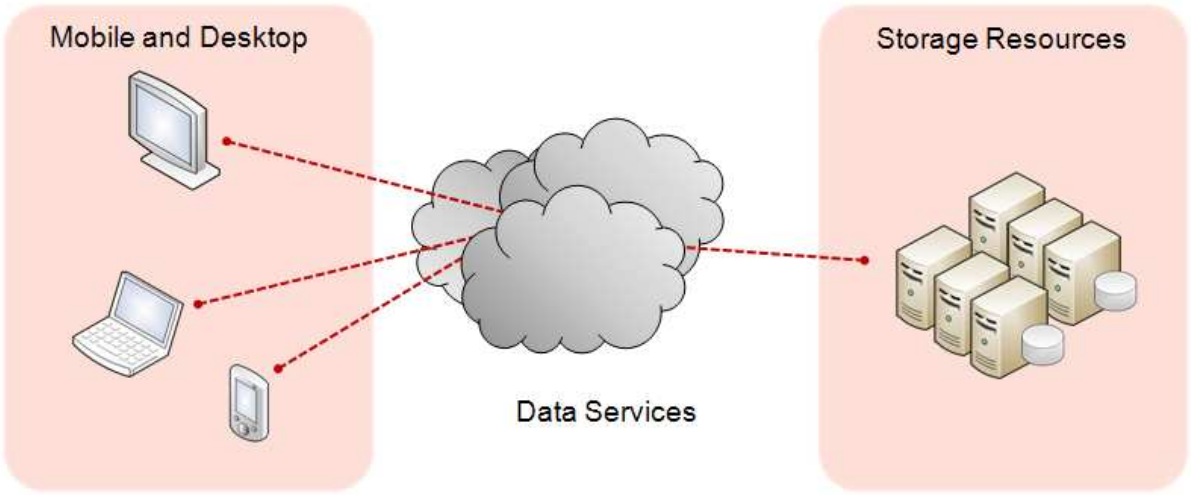

Figure 1: Services in the cloud

Unlike on-premises applications, applications in the cloud have different architectures. While on-premises platforms support enterprise scale applications, cloud platforms operate in internet scale, handling many more simultaneous users. This high scalability forces a cloud platform to provide functions in a different way.

SDS is such a cloud platform. It is based on HTTP and uses the SOAP and REST protocol to make queries and transfer data between the cloud and client. SDS is very flexible, because of its service interface and simple data model. To easily process and operate on data, there is also a query language which works with the service interface and data model. Do we really need a query language? Isn't it just overhead?

On a cloud platform there are some limitations. For example, it's not possible to access underlying operating system functions to process data, but why?

"One of the things that makes cloud computing so attractive is its potential for scalability, but to make an application built on a cloud foundation handle Internet-size loads requires limiting it in some ways." [DavChap08]

These limitations lead to the fact that the platform has to provide the functions itself and this in a flexible and efficient way to handle a lot of user requests and to be scalable. Therefore access to data storage is done by a query language.

There's also a different way to store data. SDS uses a three-containment level – the ACE Concept. This is a flexible data model where no schema is needed to store data. There are three levels – authority, container and entity. The authority is a unit of geo-location and has a collection of containers. A container has a collection of entities and is unit of consistency and search. An entity is a property bag of name-value pairs and therefore a unit of update and change. This data model lets the cloud arbitrarily scale.

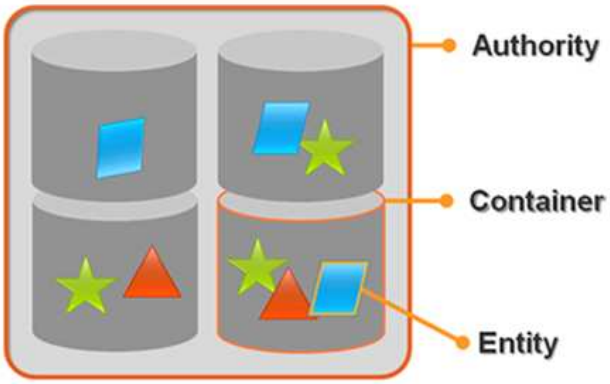

Figure 2: The ACE Concept

Using SQL Data Services with REST

First of all, what you need is to register for SDS. Once you got an account, you need to download the SQL Data Services SDK. There you will find an application called SDS Explorer, which I will use for this demo.

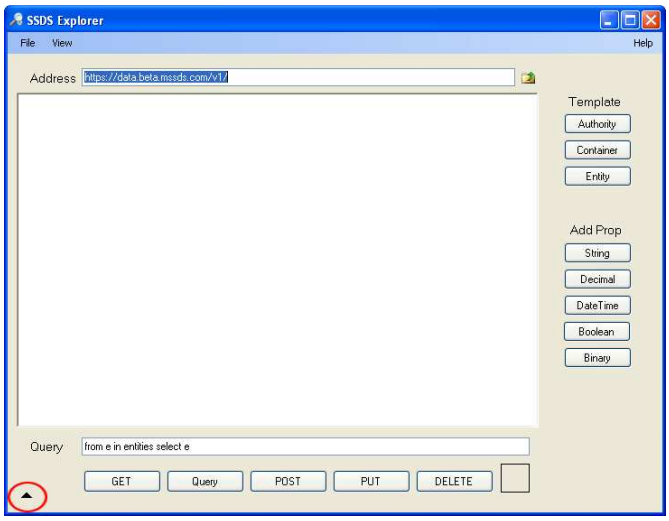

Figure 3: SDS Explorer

After you opened the SDS Explorer you will see the following UI. At the top is the address bar which you need to navigate between the three-containment levels. Then there are some buttons for the support of templates and for interaction with the service. There is also a textbox for your queries and if you press the arrow button at the bottom left corner you will get additional information about your requests and responses.

NOTE: You have to choose your authority id wisely, because you can't delete or undo it at the moment.

The SDS Explorer uses the REST protocol to connect to the service. I first create an authority which is at the top of the ACE Concept. Therefore I use the authority template and name its ID *msp-goes-sds*. Then I send this statement to the service by pressing the post button. You can see the request and response in the bottom of the window. Here you will recognize that REST is based on HTTP.

The next step is to create a container; therefore I choose the appropriate template button. I call this container *actiongames* and post it. Now I'm a step deeper in the three-containment level. You can see it in the URI or by comparing these statements.

https://{authority id}.data.beta.mssds.com/v1/{container id}

https://msp-goes-sds.data.beta.mssds.com/v1/actiongames

It's time to add some entities; there is also a particular template button for it. In the bottom box you see the basic structure of an entity. An entity can have two types of properties, metadata properties and flexible properties. The ID-tag is a metadata property and the other two, *Name* and *ReleaseDate,* are flexible properties.

```
<Entity xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:x="http://www.w3.org/2001/XMLSchema"
        xmlns:s="http://schemas.microsoft.com/sitka/2008/03/">
  <s:Id>1</s:Id>
  <Name xsi:type="x:string">Halo 3</Name>
  <ReleaseDate xsi:type="x:dateTime">2008-10-18T10:54:27.2712064Z</ReleaseDate>
</Entity>
```

In the picture you see two entities. The flexible properties of these entities differ in their kind; types and one of these entities has more properties than the other.

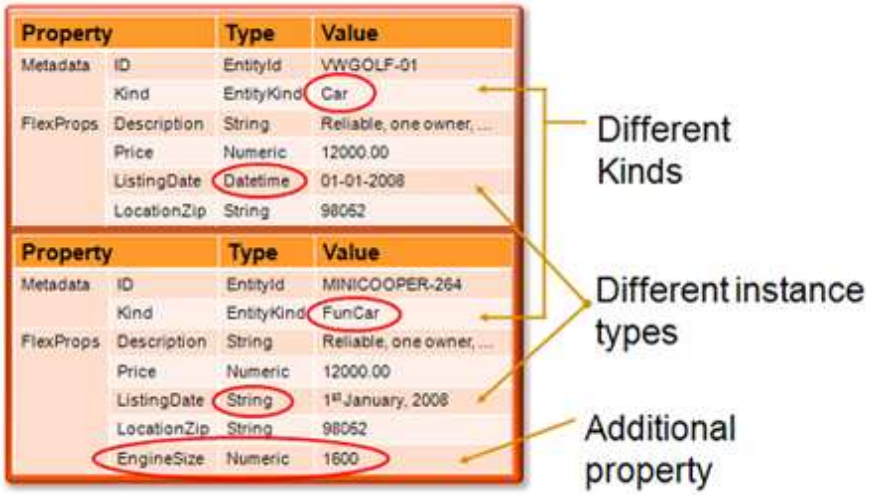

Figure 4: Flexible Properties

I repeated this step several times to have a few games in my *actiongames* container. Now I can query for them by using the query language. But first I have to go back to the container level. What does this mean? After I created a new entity the URI automatically move on to the entity level. **1** is the Id of the currently created entity. If you want to move back to the container you just have to delete this 1.

https://{authority id}.data.beta.mssds.com/v1/{container id}/{entity id}

https://msp-goes-sds.data.beta.mssds.com/v1/actiongames/**1**

Now, I can query the container for entities. I first create a query where I use a metadata property. You should recognize the difference of how to access the property data.

Metadata property:	from e in entities where **e.Id**=="1" select e
Flexible property:	from e in entities where **e["Name"]**=="Halo 3"

Using SDS with SOAP

What you need for this part is Visual Studio 2005 / 2008. I create a new console application in Visual Studio and want to query data from the SDS service. After I created the Console Application I need a Service Reference to SDS, so I go to the project in the Solution Explorer and add a Service Reference. At next I have to find the service by enter the address and press GO. I call this Service Reference SDSClient and press Ok.

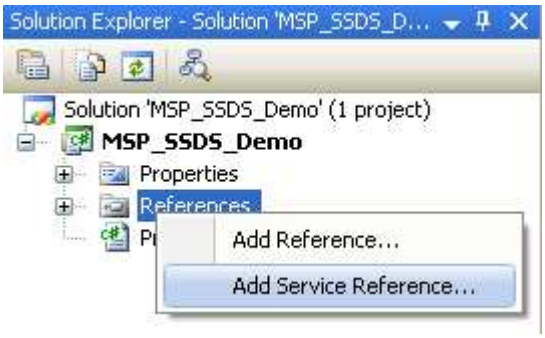

Figure 5: Add the SDS service

Service address: http://data.beta.mssds.com/soap/v1?wsdl

After this, I have to add the namespace by adding a using statement at the top of the source code and add the following code in the main block to query the SDS service.

```
using MSP_SDS_Demo.SDSClient;

static void Main(string[] args)
{
    using (SitkaSoapServiceClient proxy = new
           SitkaSoapServiceClient("SitkaSoapEndpoint"))
    {
        // 1 - Access to SDS
        proxy.ClientCredentials.UserName.UserName = _YourUserName;
        proxy.ClientCredentials.UserName.Password = _YourPassword;

        // 2 - Definition of the scope
        Scope myContainerScope = new Scope();
        myContainerScope.AuthorityId = "msp-goes-sds";
        myContainerScope.ContainerId = "actiongames";

        // 3 - Sample Query
        string sampleQuery = @"from e in entities where e.Id >
                               ""1"" select e";

        // 4 - Process Query
        IEnumerable<Entity> games = proxy.Query(myContainerScope,
        sampleQuery);

        foreach (Entity e in games)
            Console.WriteLine(e.Properties["Name"]);

        Console.Write("Press any key...");
        Console.ReadKey(true);
    }
}
```

At first I created a using block in which I instantiated a SitkaSoapServiceClient object. This object is the proxy for my service. Now I have to give the proxy my credentials to get access to SDS. At next I need a scope, because of the ACE concept. I define this scope at point 2. With `myContainerScope.AuthorityId` and `myContainerScope.ContainerId` I determine explicitly that I want to look for entities in the *actiongames* container. After that I create the query. I want all games where the ID is greater than 1. So I use the query language to build such a construct. So for now I just need to process this query. I do this at point 4. In the end I list the response to the output.

NOTE: In this article you will see sometimes the abbreviation SSDS instead of SDS, especially in pictures of the demo or in URLs of the service. This is due to a change of the name from Microsoft SQL Server Data Services (SSDS) to Microsoft SQL Data Services (SDS). It can also be that some URLs will be different after the beta status.

[DavChap08] Chappell, David: Cloud Platforms – An enterprise-oriented overview. Chappell & Associates, 2008

[JasLee08] Lee, Jason: Microsoft SQL Data Services – Under the Hood. Microsoft, 2008

[MSDN08] MSDN Library – SQL Data Services Primer http://msdn.microsoft.com/de-de/library/cc512417(en-us).aspx. Version: 2008. [last checked on 11/05/2008]