Microsoft Azure Table Storage

NoSQL Seminar @ TUT

20.11.2012

Arto Salminen

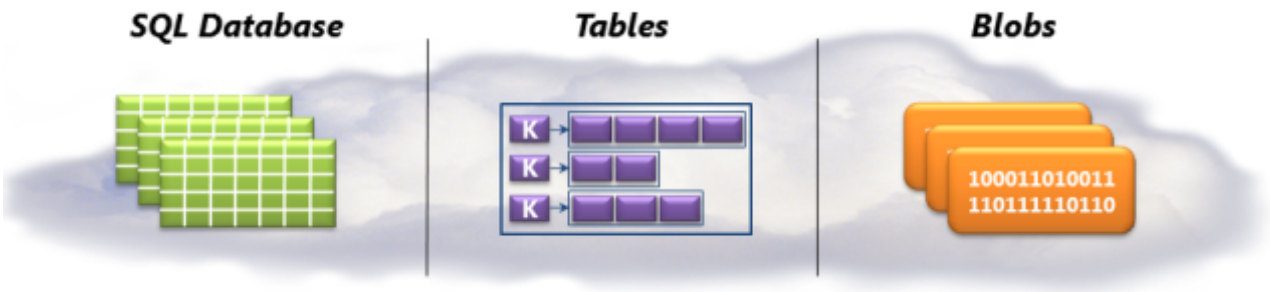
Microsoft Azure Cloud Service

- Virtual machines (IaaS)
- Cloud services (PaaS)
  - Containers of hosted applications
  - Web role - web application with PHP / .NET
  - Worker Role - back-end / application server
- Web sites (hosting)
  - Built-in support for popular applications, e.g. WordPress, Joomla, and Drupal
- SDKs started by Microsoft for Python, Java, node.js, PHP and .NET
  - Others in open source, as well
- North Europe data center in Dublin

Subscribing to Windows Azure

- Windows Live ID is required for sign-in
- Pricing
  - See
    - https://www.windowsazure.com/en-us/pricing/calculator/
    - https://www.windowsazure.com/en-us/pricing/details/#storage
  - Pay-as-you-go
  - 6-month plan / 12-month plan
- 90 days free trial
  - Credit card info needed, but not charged, hopefully

Azure Data Storing Services



- Azure has four different storage services
  - *SQL Database* for storing relational data
  - *Table Storage*, which stores large amounts of structured data.
  - *Blob* (Binary large object) for storing large amounts of unstructured data
  - Also *Queues* for for storing large numbers of messages between services

Azure Table Service

- NoSQL key-value data storage
  - Schemaless
  - Non-relational
  - Accessible from inside and outside Azure
- Size limits
  - 100 TB / storage account
  - 1 TB / table
  - 1 MB / entity
- Single account may include several tables

Table Storage Structure

- (Account)
- Tables
  - A table can have different types of rows
- Entities (rows)
- Properties
  - RowKey
  - PartitionKey
  - Timestamp
  - ...

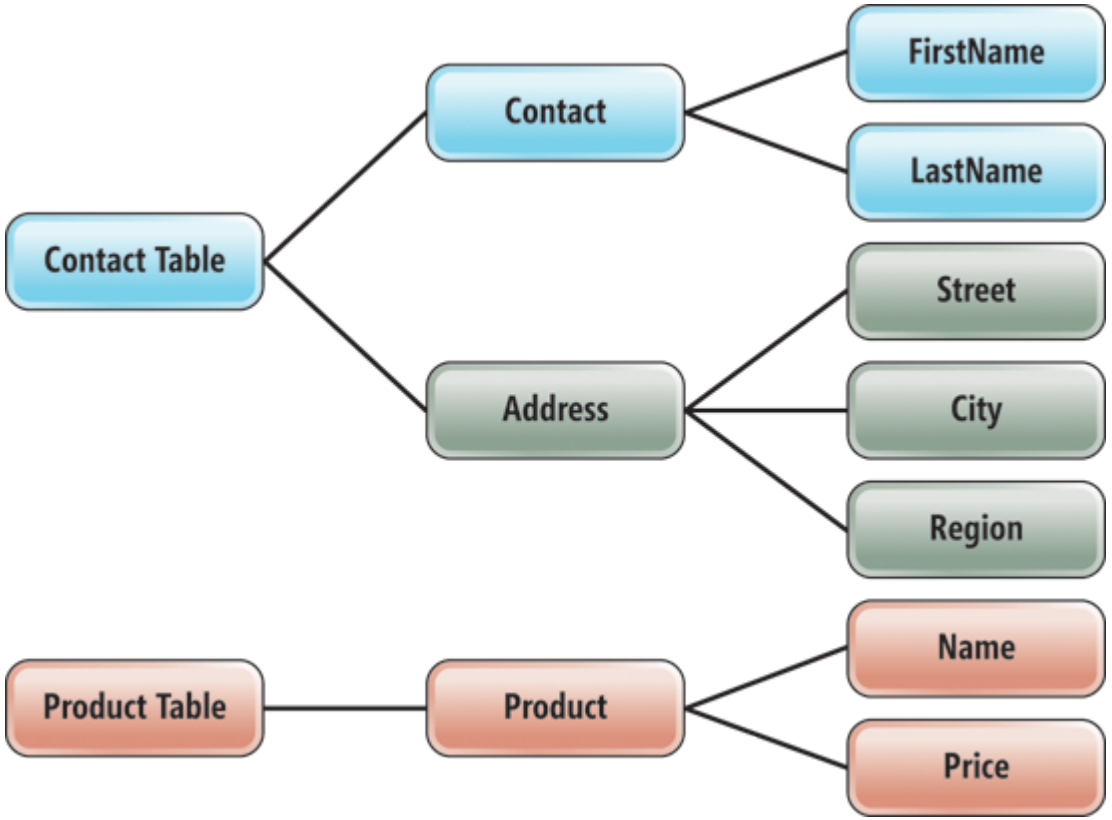


Table Storage - Partitioning

- About performance of Table Service
- For more details, see:
  - http://msdn.microsoft.com/en-us/magazine/ff796231.aspx
  - http://www.microsoftpdc.com/2009/svc09 Scaling example starting on 14:00

- Each entity in a table must have a unique PartitionKey / RowKey combination
  - Rows with the same PartitionKey value are kept together on the same server when load balancing
    - Entities with the same PartitionKey form partitions
    - Partition = unit of scale in Azure system
  - Queries with explicit PartitionKey/RowKey combination are fast.
  - Scans within a partition are fast as well, when compare to scan across partitions.
- Selecting PartitionKeys should be done carefully, as it is critical for scalability and query efficiency.

Table Storage - Continuation Tokens

- A query may return a maximum of 1 000 items at one time, and may execute for a maximum of five seconds.
- If the result set contains more than 1 000 items, if the query did not complete within five seconds, or if the query crosses the partition boundary, the response includes headers which provide the developer with *continuation tokens*.
- Continuation tokens are then used to form a new query for the next set of data
- For better performance, client can parallelize the query across partitions
  - Continuation tokens still need to be taken care of
- http://msdn.microsoft.com/en-us/library/windowsazure/dd135718.aspx

Table Service Interfaces

- REST API
  - Includes both table-level and entity-level operations, used with proper HTTP nouns
  - http://msdn.microsoft.com/en-us/library/windowsazure/dd179423.aspx
  - node.js library available :)
- .NET client library
  - Azure SDK for .NET
  - WCF service project that a client can access is typically used
- Tool for exploring: Azure Storage Explorer http://azurestorageexplorer.codeplex.com/

Creating New Table Account

- Create new account
  - Name: slmnnlocationdb
  - Region: North Europe
    - URL: http://slmnnlocationdb.table.core.windows.net/<table>
- Setup a storage connection string for client authentication
  - Storage connection string configures endpoints and credentials for accessing storage services.
  - Credentials can be imported from a file, or written directly to project configuration files

Creating Node.js project for Azure

- Use Azure PowerShell to create new node.js application instance
  - `> Get-AzurePublishSettingsFile`
  - `> Import-AzurePublishSettingsFile <file>`
  - `> C:\app> New-AzureServiceProject -ServiceName MyService`
  - `> C:\app\MyService> Add-AzureNodeWebRole MyWebRole -I 2`
- Set up git repository for node.js webrole
- Configure node.js
  - `> npm install azure`
  - `var azure = require('azure');`

Using Table Service in node.js 1/3

```
// Creating a TableService object
var tableService = azure.createTableService();

// Creating a table
tableService.createTableIfNotExists('mytable', function
(error){
    if(!error){
        // Table exists or created
    }
});
```

Using Table Service in node.js 2/3

```
// Adding new entity
var task = {
    PartitionKey : 'hometasks'
    , RowKey : '1'
    , Description : 'Take out the trash'
    , DueDate: new Date(2012, 6, 20)
};
tableService.insertEntity('mytable', task, function(error){
    if(!error){ // Entity inserted   }
});


// Updating entity
var task = {
    PartitionKey : 'hometasks'
    , RowKey : '1'
    , Description : 'Wash Dishes'
}
tableService.updateEntity('mytable', task, function(error){
    if(!error){  // Entity has been updated   }
});
```

Using Table Service in node.js 3/3

```
// Query an entity with partition and row key
tableService.queryEntity('mytable'
    , 'hometasks'
    , '1'
    , function(error, entity){
        if(!error){
            // entity contains the returned entity
        }
});

// Query a set of entities
var query = azure.TableQuery
    .select()
    .from('mytable')
    .where('PartitionKey eq ?', 'hometasks');
tableService.queryEntities(query, function(error, entities){
    if(!error){
        //entities contains an array of entities
    }
});
```

Demo Applications for the Table Storage

- My database structure
  - Table: locations
    - PartitionKey: users
    - RowKey = nick
      - Properties: Status, Lat, Lng, StatusDate
- HTML5 Client
  - Uses node.js server for interfacing with the Table Storage
- WP7 Client
  - Accesses the database directly
    - One should use the database through WCF Service Application (Windows Communication Foundation), which is implemented as a .NET project in Visual Studio. WCF project can be then included into a WP7 project as a Service Reference, and methods accessed as usual.

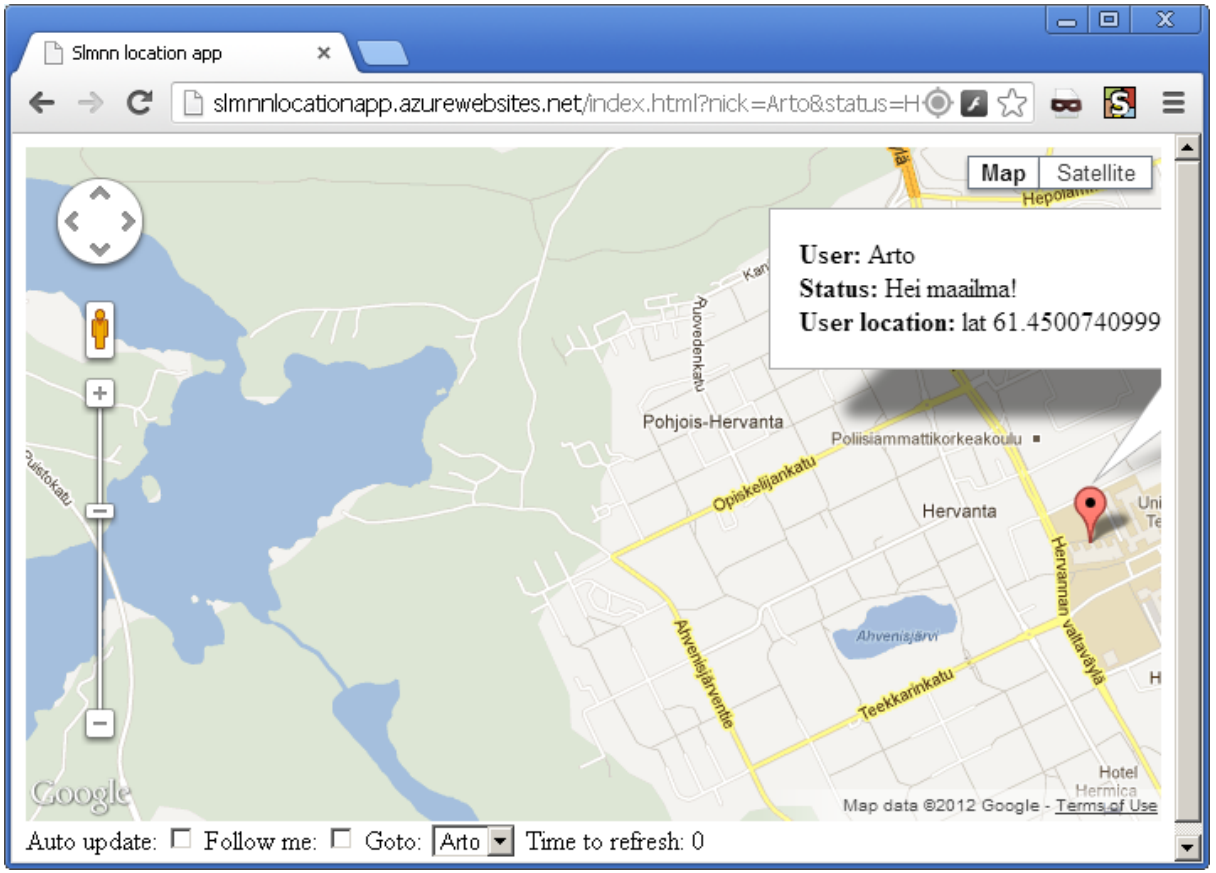
My node.js implementation for the HTML5 client

- Simple interface for querying and updating user data
  - `GET http://slmnnlocation.azurewebsites.net/user`
    - Returns data as JSON
    - One can use a nickname as query
      - http://slmnnlocation.azurewebsites.net/user?nick=Arto
  - `POST http://slmnnlocation.azurewebsites.net/user`
    - If an entity with a nickname already exists, the entity is updated. Otherwise a new one is created.
  - DELETE not supported yet :)

- See the actual code with an editor...

HTML5 client

http://slmnnlocationapp.azurewebsites.net/



HTML5 client - Features

- Created as a web site in Azure
  - Uses git publishing as well
- Geolocation for user positioning
- Google Maps for presentation
- No authentication :)

HTML5 client - Implementation

- Updating entity

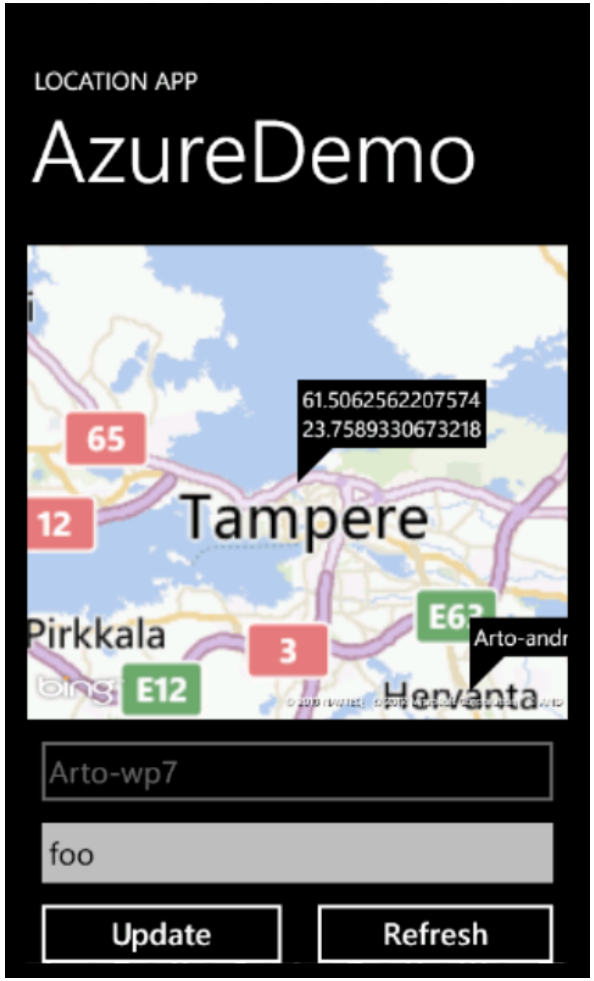
```
$.ajax({
     type: 'POST',
     url: 'http://slmnnlocation.azurewebsites.net/user',
     data: JSON.stringify(userdata),
     success: function() { alert("data posted"); },
     dataType: "json"
});
```

- Querying content

```
$.ajax({
     type: 'GET',
     url: "http://slmnnlocation.azurewebsites.net/user",
     success: function(data){
          console.log(JSON.stringify(data));
          createMarkersForAllUsers(data);
     }
     , dataType: "json"
});
```

WP7 Client

- User positioning with GPS (or other means available)
- Possibility to set nickname at startup
- Status can be changed later, as well
- Map follows the user and presents other users as pushpins



WP7 Client - Implementation

- Install Windows Azure Storage Client Library for Windows Phone
  - In Package Manager Console:
    - Install-Package Phone.Storage
- Set the credentials for database accessing
- In App_Start\StorageInitializer.cs
  - Add resolver with credentials to Azure Table Service

- See DatabaseHandler.cs for implementation details

questions / comments

arto.salminen@tut.fi