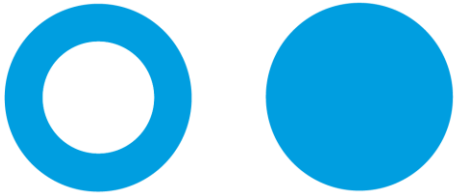
SQL Server vs. Azure DocumentDB

ein battle zwischen <xml> und {JSON}

SQL saturday #409

Organizer



Hochschule
Bonn-Rhein-Sieg

Bronze Sponsor

Silver Sponsor

redgate
ingeniously simple

idera™

SIOS

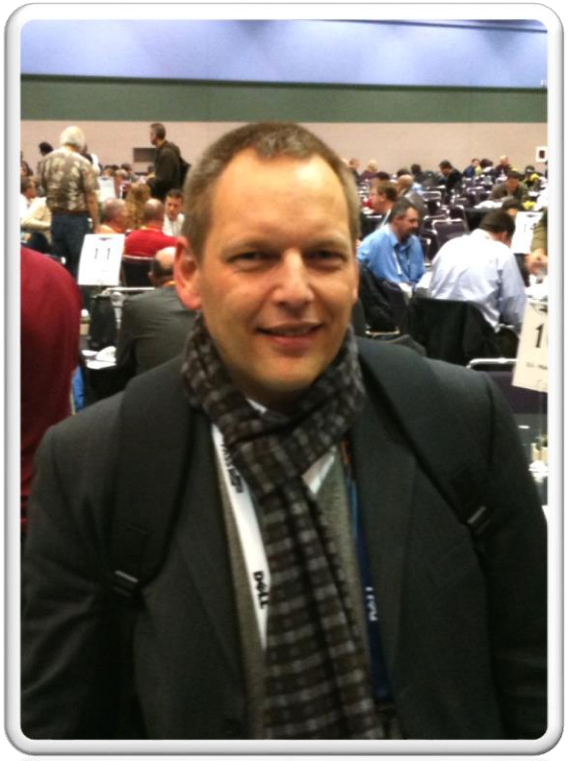
COZYROC™
Go to the next level

PYRAMID
ANALYTICS

Gold Sponsor

You rock!

About me



Alexander Karl

.net - CDE **SQL + BI Consultant**

Microsoft CERTIFIED Trainer

Microsoft CERTIFIED IT Professional

Database Administrator 2008
Server Administrator on Windows Server® 2008
Database Administrator on SQL Server® 2005

... and „2012er“ SQL MCSE

About me

Sascha Dittmann

EY Building a better working world

Sr. Software Developer
Sr. Solutions Architect

Microsoft CERTIFIED Trainer

MVP Microsoft® Most Valuable Professional

Twitter: @SaschaDittmann
Blog: http://www.sascha-dittmann.de

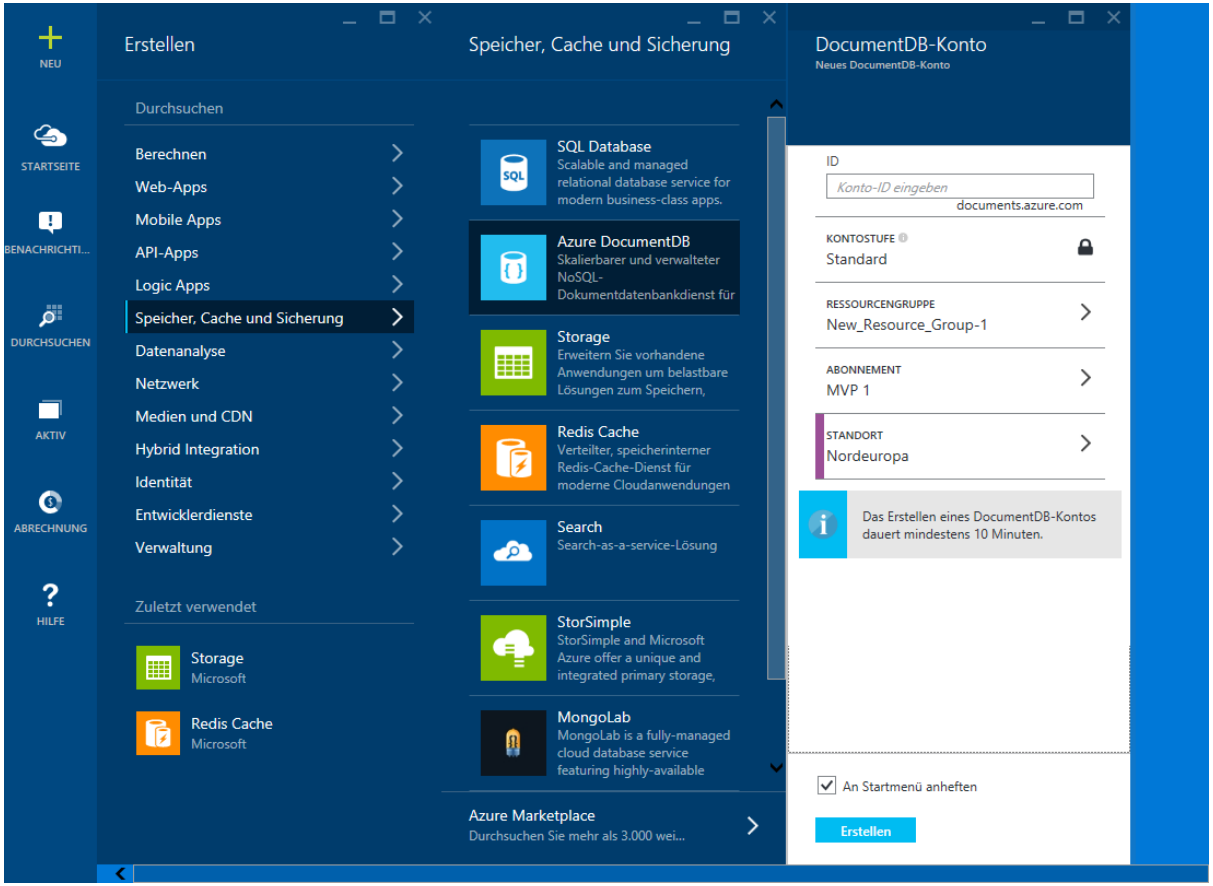
Agenda <xml/> {JSON}

- Setup
- Tooling
- Consistency
- CRUD
- Indexing
- Schema validation <xml> only
- Server side code
- Complex queries and transformation <xml> only

<xml/> Setup

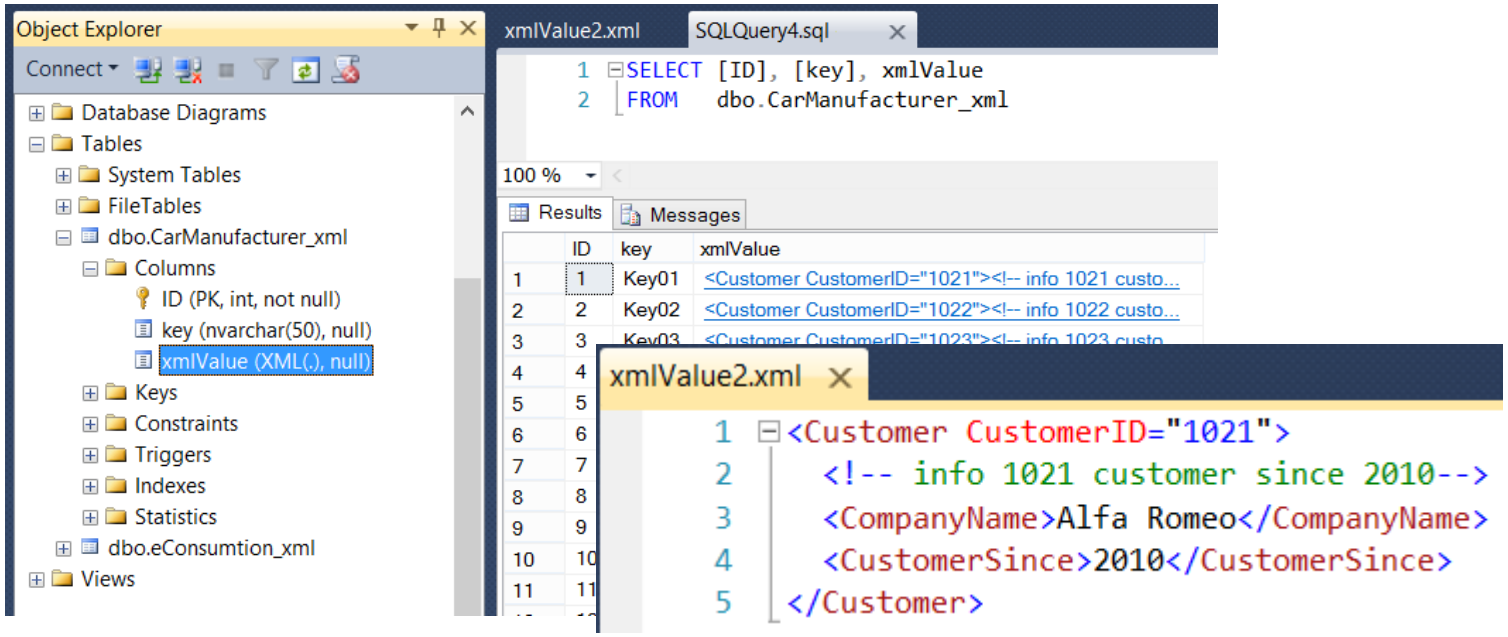
- SQL Server Setup
  erfolgt mittels Wizard oder via cmd-Line
- native **xml** Unterstützung
  „out of the Box“

Setup

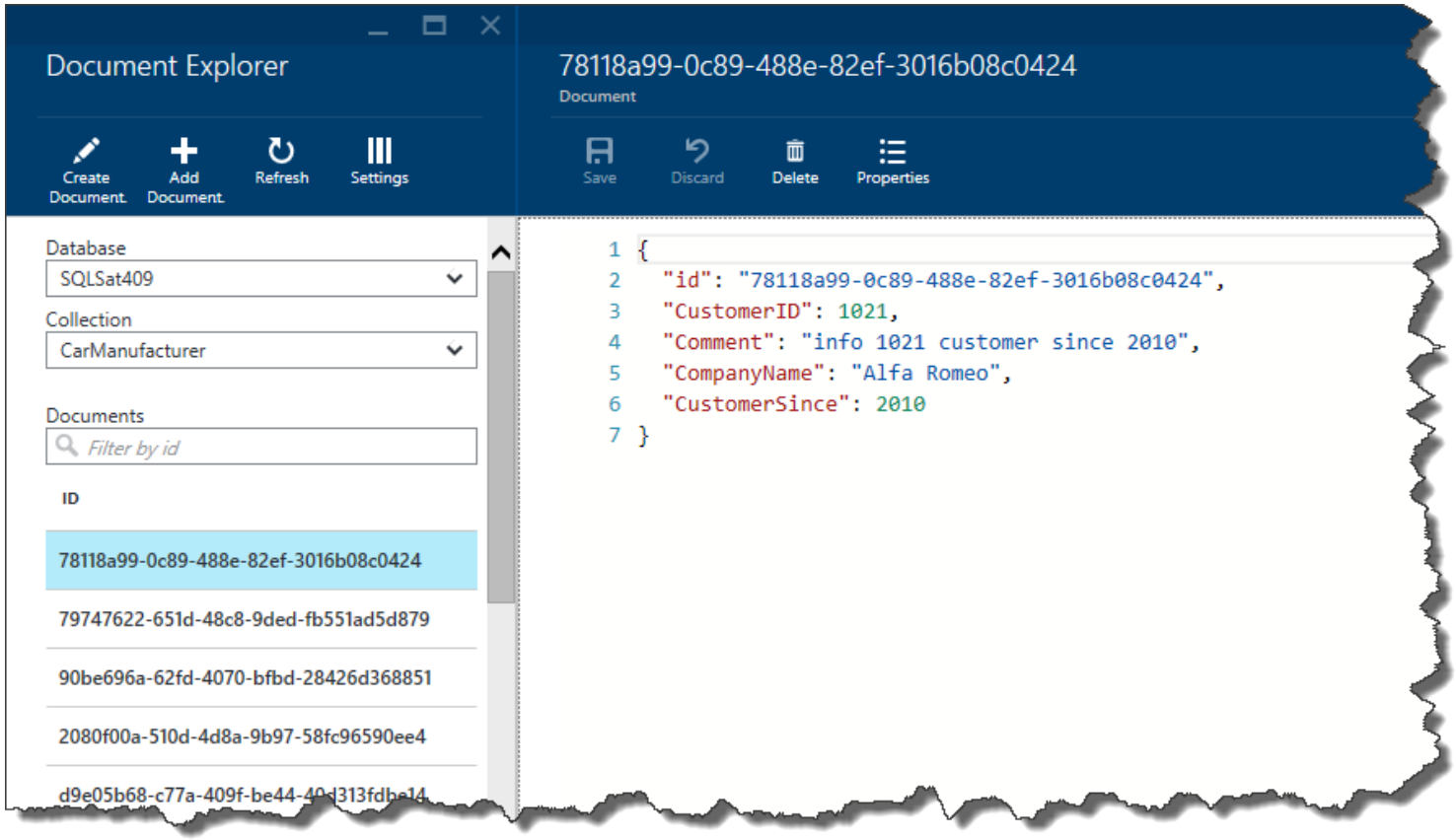

NEU
STARTSEITE
BENACHRICHTI...
DURCHSUCHEN
AKTIV
ABRECHNUNG
HILFE
Erstellen
Durchsuchen
Berechnen
Web-Apps
Mobile Apps
API-Apps
Logic Apps
Speicher, Cache und Sicherung
Datenanalyse
Netzwerk
Medien und CDN
Hybrid Integration
Identität
Entwicklerdienste
Verwaltung
Zuletzt verwendet
Storage
Microsoft
Redis Cache
Microsoft
Speicher, Cache und Sicherung
SQL Database
Scalable and managed relational database service for modern business-class apps.
Azure DocumentDB
Skalierbarer und verwalteter NoSQL-Dokumentdatenbankdienst für
Storage
Erweitern Sie vorhandene Anwendungen um belastbare Lösungen zum Speichern,
Redis Cache
Verteilter, speicherinterner Redis-Cache-Dienst für moderne Cloudanwendungen
Search
Search-as-a-service-Lösung
StorSimple
StorSimple and Microsoft Azure offer a unique and integrated primary storage,
MongoLab
MongoLab is a fully-managed cloud database service featuring highly-available
Azure Marketplace
Durchsuchen Sie mehr als 3.000 wei...
DocumentDB-Konto
Neues DocumentDB-Konto
ID
Konto-ID eingeben
documents.azure.com
KONTOSTUFE
Standard
RESSOURCENGRUPPE
New_Resource_Group-1
ABONNEMENT
MVP 1
STANDORT
Nordeuropa
Das Erstellen eines DocumentDB-Kontos dauert mindestens 10 Minuten.
An Startmenü anheften
Erstellen

<xml/> Tooling

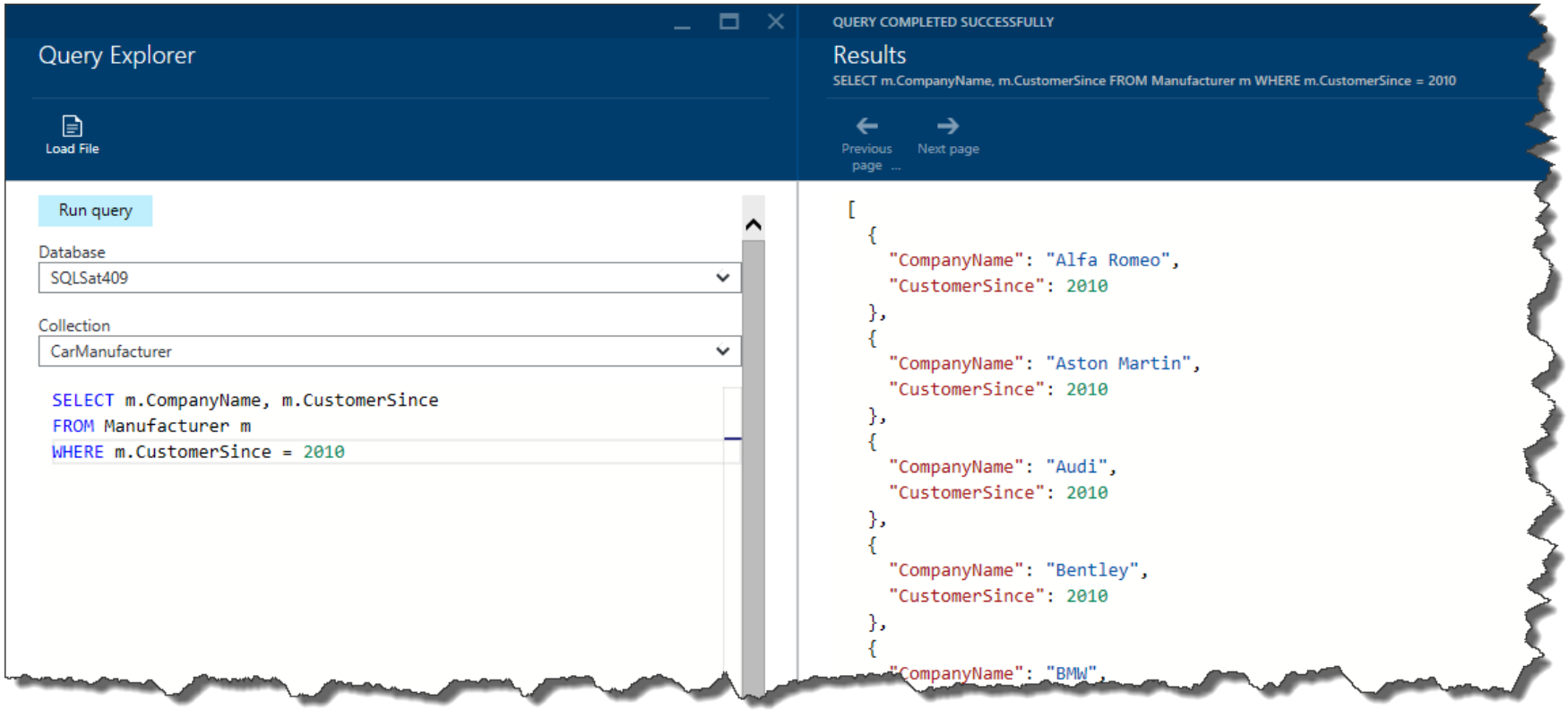
- SQL Server Management Studio
  >> für queries
  >> für „xml-results“



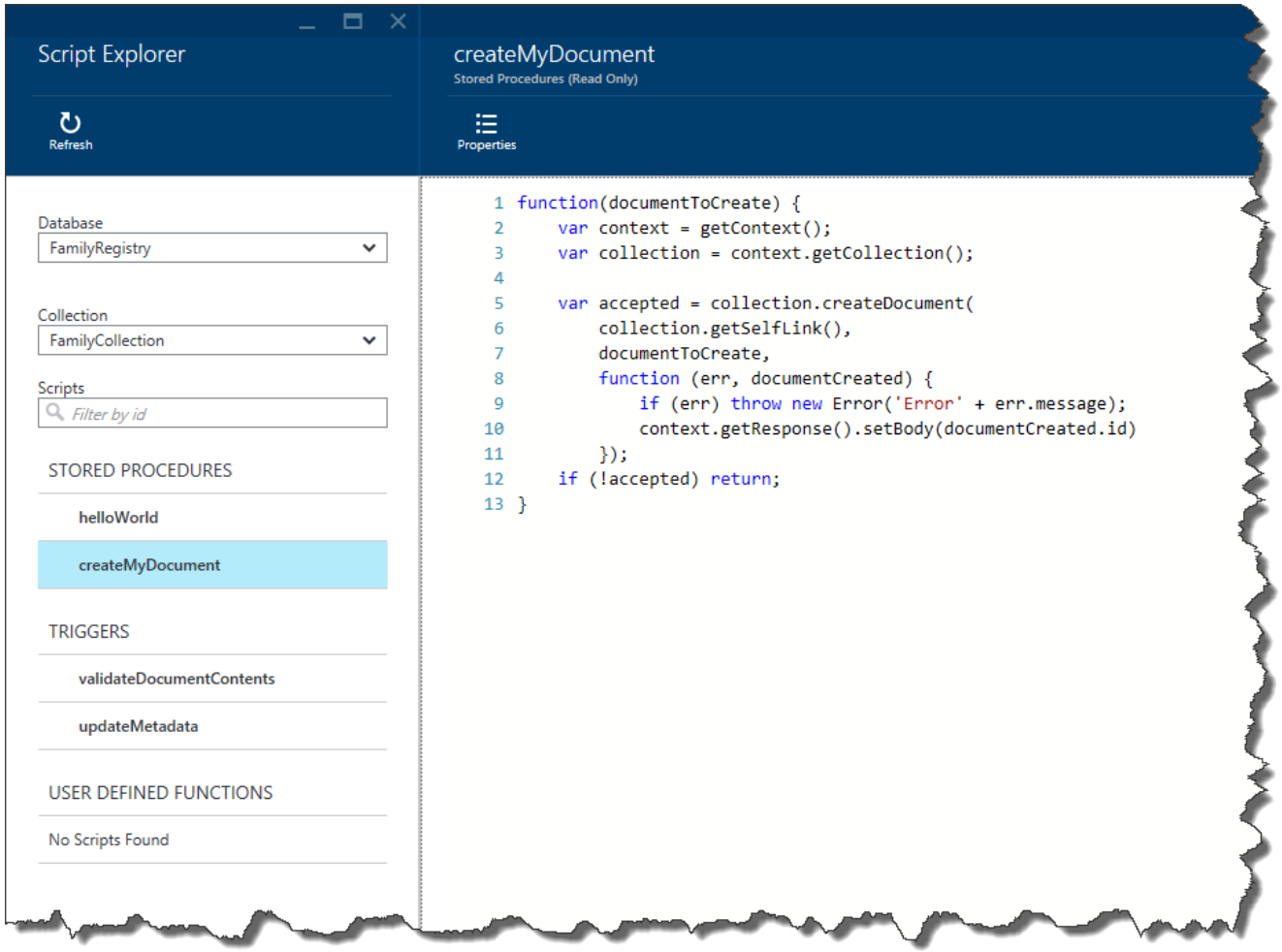
Tooling


Document Explorer
Create Document
Add Document
Refresh
Settings
Database
SQLSat409
Collection
CarManufacturer
Documents
Filter by id
ID
78118a99-0c89-488e-82ef-3016b08c0424
79747622-651d-48c8-9ded-fb551ad5d879
90be696a-62fd-4070-bfbd-28426d368851
2080f00a-510d-4d8a-9b97-58fc96590ee4
78118a99-0c89-488e-82ef-3016b08c0424
Document
Save
Discard
Delete
Properties
{
"id": "78118a99-0c89-488e-82ef-3016b08c0424",
"CustomerID": 1021,
"Comment": "info 1021 customer since 2010",
"CompanyName": "Alfa Romeo",
"CustomerSince": 2010
}

Tooling


Query Explorer
Load File
Run query
Database
SQLSat409
Collection
CarManufacturer
SELECT m.CompanyName, m.CustomerSince
FROM Manufacturer m
WHERE m.CustomerSince = 2010
QUERY COMPLETED SUCCESSFULLY
Results
SELECT m.CompanyName, m.CustomerSince FROM Manufacturer m WHERE m.CustomerSince = 2010
Previous page
Next page
[
{
"CompanyName": "Alfa Romeo",
"CustomerSince": 2010
},
{
"CompanyName": "Aston Martin",
"CustomerSince": 2010
},
{
"CompanyName": "Audi",
"CustomerSince": 2010
},
{
"CompanyName": "Bentley",
"CustomerSince": 2010
},
{
"CompanyName": "BMW",

Tooling


Script Explorer
createMyDocument
Stored Procedures (Read Only)
Refresh
Properties
Database
FamilyRegistry
Collection
FamilyCollection
Scripts
Filter by id
STORED PROCEDURES
helloWorld
createMyDocument
TRIGGERS
validateDocumentContents
updateMetadata
USER DEFINED FUNCTIONS
No Scripts Found
1 function(documentToCreate) {
2 var context = getContext();
3 var collection = context.getCollection();
4
5 var accepted = collection.createDocument(
6 collection.getSelfLink(),
7 documentToCreate,
8 function (err, documentCreated) {
9 if (err) throw new Error('Error' + err.message);
10 context.getResponse().setBody(documentCreated.id)
11 });
12 if (!accepted) return;
13 }

<xml/> Consistency (ACID)

A tomicity

C onsistency

I solation

D urabilty

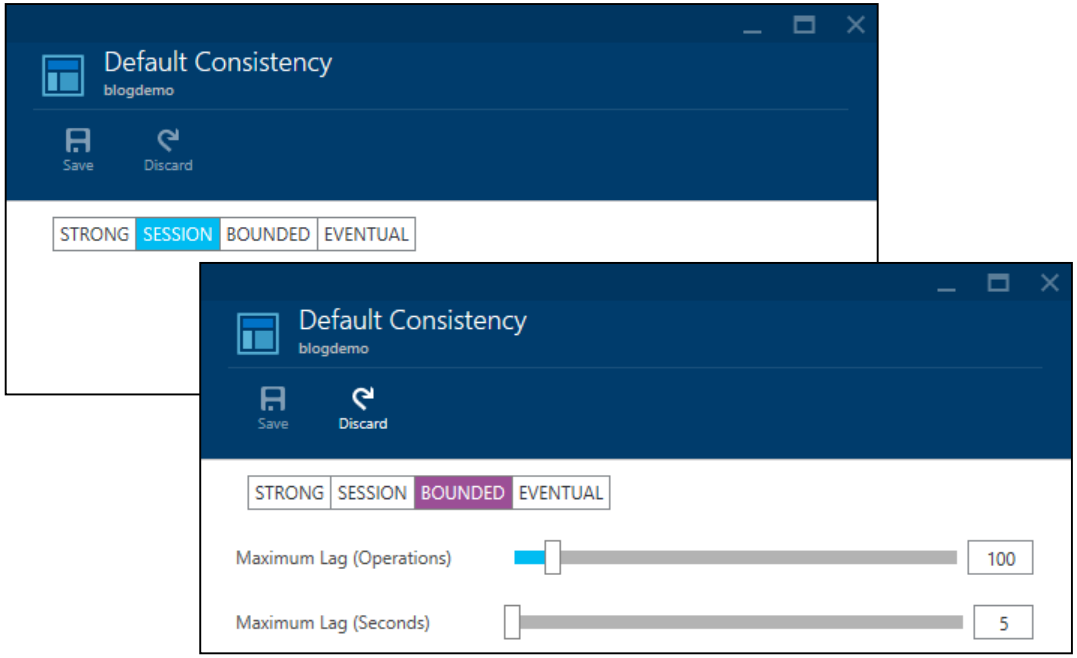
Consistency (BASE)

B asically

A vailable

S oft state

E ventual consistency



<xml/> Select & Ins/Upd/Del

- T-SQL conformer Ins/Upd/Del
- T-SQL XQuery
  - <column>.query (return xml)
  - <column>.value (return sql-Type)
  - <column>.exist (return bit)
  - <column>.nodes

<xml/> Select & Ins/Upd/Del

- T-SQL Xquery Navigation im <xml>
  - child
  - descendant
  - parent
  - Attribut
  - self
  - descendant-or-self

  - .node()
  - .text()

<xml/> CRUD Samples

SQLQuery1.sql

```sql
----  1st Query
SELECT [key], xmlValue.query('*') as Complete_Sequence
FROM   dbo.CarManufacturer_xml

----  2nd Query
SELECT [key], xmlValue.query('data(*)') as Complete_Data
FROM   dbo.CarManufacturer_xml

----  3rd Query  /node()   &   /comment()
SELECT [key]
     , xmlValue.query('/Customer/CompanyName/node()') as CompanyName
     , xmlValue.query('/Customer/comment()')
FROM   dbo.CarManufacturer_xml

```

Select & Ins/Upd/Del

- LINQ, SQL
- REST API over HTTPS
  - .NET
  - Node.js
  - JavaScript
  - Python
  - ...

CRUD Samples

```
var manufacturers = client
    .CreateDocumentQuery<CarManufacturer>(collectionLink)
    .Where(c => c.CompanyName == name);

foreach (var manufacturer in manufacturers)
{
    Console.WriteLine("{0} is a customer since {1}",
        manufacturer.CompanyName, manufacturer.CustomerSince);
}
```

```
SELECT m.CompanyName, m.CustomerSince
FROM Manufacturer m
WHERE m.CustomerSince = 2010
```

<xml/> Indexing

- PRIMARY XML INDEX IX_primaryXml

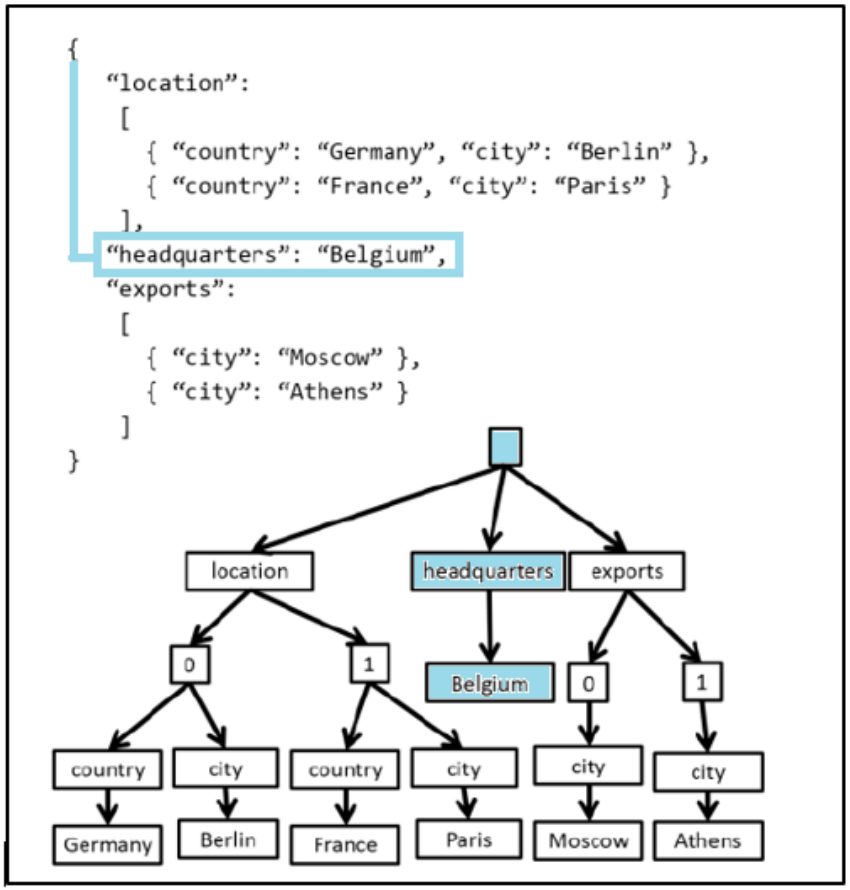
- CREATE XML INDEX IX_name
  ON sch.table (xmlCol)
  USING XML INDEX IX_primaryXml
  - FOR PATH;
  - FOR VALUE;
  - FOR PROPERTY;

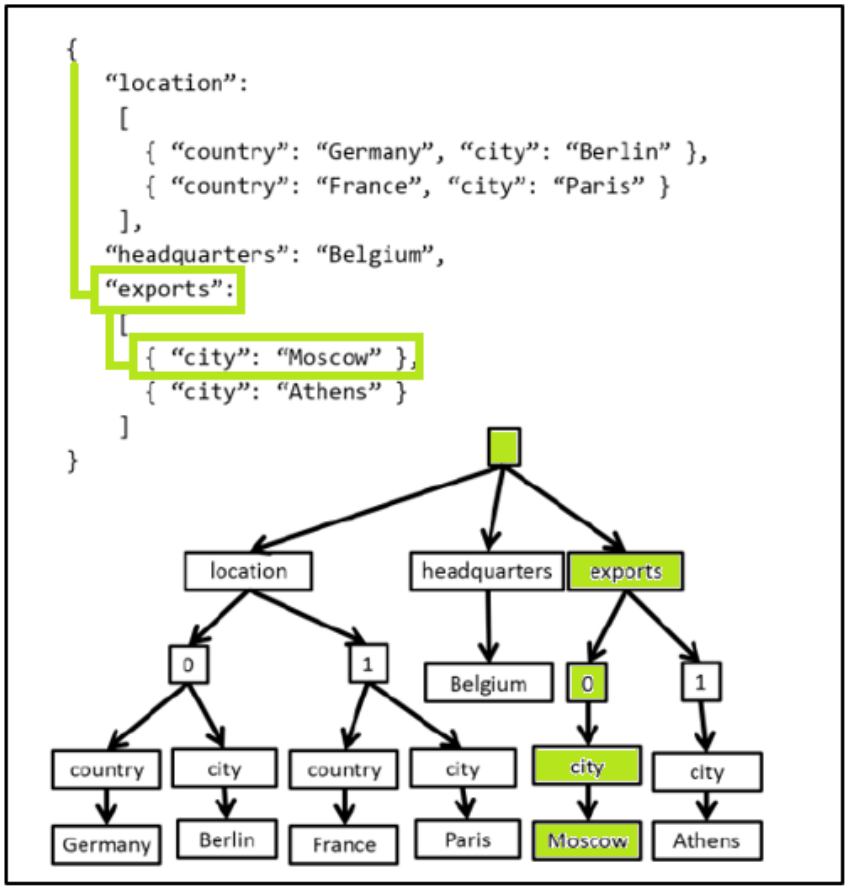
<xml/> Indexing

xml_Indexes.sql

```
1  CREATE PRIMARY XML INDEX IX_primaryXml
2  ON  dbo.eConsumtion_xml  ( xmlValue );
3  GO
4
5
6  CREATE XML INDEX IX_Xml_PATH
7  ON  dbo.eConsumtion_xml  ( xmlValue )
8  USING XML INDEX  IX_primaryXml
9  FOR PATH;
10
11 CREATE XML INDEX IX_Xml_PROPERTY
12 ON  dbo.eConsumtion_xml  ( xmlValue )
13 USING XML INDEX  IX_primaryXml
14 FOR PROPERTY;
15
16 CREATE XML INDEX IX_Xml_VALUE
17 ON  dbo.eConsumtion_xml  ( xmlValue )
18 USING XML INDEX  IX_primaryXml
19 FOR VALUE;
20 GO
```

Indexing


{
"location":
[
{ "country": "Germany", "city": "Berlin" },
{ "country": "France", "city": "Paris" }
],
"headquarters": "Belgium",
"exports":
[
{ "city": "Moscow" },
{ "city": "Athens" }
]
}
location
headquarters
exports
0
1
Belgium
0
1
country
city
country
city
city
city
Germany
Berlin
France
Paris
Moscow
Athens


{
"location":
[
{ "country": "Germany", "city": "Berlin" },
{ "country": "France", "city": "Paris" }
],
"headquarters": "Belgium",
"exports":
[
{ "city": "Moscow" },
{ "city": "Athens" }
]
}
location
headquarters
exports
0
1
Belgium
0
1
country
city
country
city
city
city
Germany
Berlin
France
Paris
Moscow
Athens

Indexing Policies

```
{
    "id":"customIndexCollection",
    "indexingPolicy":
    {
        "automatic":true,
        "indexingMode":"Consistent",
        "IncludedPaths":
        [
            {
                "IndexType":"Hash",
                "Path":"/"
            }
        ],
        "ExcludedPaths":
        [
            "/\"nonIndexedContent\"/*"
        ]
    }
}
```

<xml/> Schema Validation

- XML SCHEMA COLLECTION schemaName

- (table) xmlColumn
  xml (DOCUMENT schemaName)

<xml/> Schema Validation

xml_Schema.sql

```
1  CREATE XML SCHEMA COLLECTION  dbo.bookSchemaCollection
2  AS
3  N'<?xml version="1.0"?>
4  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
5      targetNamespace="http://www.test-fabrik.de/BooksSchema"
6      xmlns="http://www.test-fabrik.de/BooksSchema"
7      elementFormDefault="qualified">
8      <xs:element name="Book">
9          <xs:complexType>
10             <xs:attribute name="Title" type="xs:string"/>
11             <xs:attribute name="Price" type="xs:decimal"/>
12         </xs:complexType>
13     </xs:element>
14 </xs:schema>';
15 GO
16
17 CREATE TABLE  dbo.Books_xml(
18                ID        int Identity(1,1)
19              , [key]     nvarchar(50)
20              , xmlRaw    xml
21              , xmlValue  xml(DOCUMENT bookSchemaCollection)
22              );
23 GO
24
```

<xml/> Schema Validation

xml_Schema.sql

```
1  INSERT dbo.Books_xml
2          ( [key] , xmlValue )
3  VALUES ( 'key02', '<Book xmlns="http://www.test-fabrik.de/BooksSchema"
4                           Title="the booksample2"
5                           Price="49.90" />')
6
7  --------------
8  -- !! decimal ,
9  INSERT dbo.Books_xml
10         ( [key] , xmlValue )
11 VALUES ( 'key03', '<Book xmlns="http://www.test-fabrik.de/BooksSchema"
12                          Title="! v ! Book"
13                          Price="49,90" />')
14
15 --Msg 6926, Level 16, State 1, Line 40
16 --XML Validation: Invalid simple type value: '49,90'. Location: /*:Book[1]/@*:Price
```

Schema Validation

- Triggers
- Client-Side Code

server side code

```
function(documentToCreate) {
    var context = getContext();
    var collection = context.getCollection();

    var accepted = collection.createDocument(
        collection.getSelfLink(),
        documentToCreate,
        function (err, documentCreated) {
            if (err) throw new Error('Error' + err.message);
            context.getResponse().setBody(documentCreated.id)
        });
    if (!accepted) return;
}
```

<xml/> compl. queries & transformation

- XML *„FLWOR“*
  for **l**et **w**here **o**rder **r**eturn

- Aggregat functions
  - ✓ {count($j)}
  - ✓ {sum($j)}
  - ✓ {avg($j)}

<xml/> compl. queries & transformation

xml_FLWOR.sql

```sql
-- sample FLWOR
SELECT ID, [key], xmlValue
     , xmlValue.value('(CustomerData/Customer/CustID)[1]', 'nvarchar(40)' )  as 'nvarchar_CustID'
     , xmlValue.query(' for $i in CustomerData/Customer/CustID
                        let $j := CustomerData/Customer/Consumtion/consumtionValue
                        where  avg($j) >= 5
                        return
                        <result>
                           {data($j)}
                        </result>
                      ') as 'data'
     , xmlValue.query(' for $i in CustomerData/Customer/CustID
                        let $j := CustomerData/Customer/Consumtion/consumtionValue
                        where  avg($j) >= 5
                        return
                        <result>
                           <count>  {count($j)} </count>
                           <sum>    {sum($j)}   </sum>
                           <avg>    {avg($j)}   </avg>
                        </result>
                      ') as 'summary'
FROM   dbo.eConsumtion_xml;
```

compl. queries

```
SELECT c.id, v.ConsumptionDay
FROM Customer c
JOIN v IN c.Consumptions
WHERE v.ConsumptionValue = 5
```

???

Save the date!