OData Extension for XML Data

A Directional White Paper

Introduction

This paper documents some use cases, initial requirements, examples and design principles for an OData extension for XML data. It is non-normative and is intended to seed discussion in the OASIS OData TC for the development of an OASIS standard OData extension defining retrieval and manipulation of properties representing XML documents in OData.

XML has achieved widespread adoption since it was published as a W3C Recommendation in 1998. XML documents were initially stored in databases as character strings, character large objects (CLOBs), or shredded into numerous rows in several related tables. Many databases now have native support for XML documents. Support for XML was added to SQL in 2003 and then extended in 2008.

XML properties are not currently supported in OData. We suggest that an OData extension be defined to add this support. Properties that contain XML documents will be identified as such, and additional operations will be made available on such properties.

Status

Version 1.0 (May 18, 2012)

Authors

Andrew Eisenberg, IBM
Ralf Handl, SAP
Michael Pizzo, Microsoft

Background

SQL/XML allows XML columns to be defined in tables:

```
CREATE TABLE employees
   (empid     INTEGER PRIMARY KEY,
    lastname  VARCHAR(30) NOT NULL,
    resume    XML)
```

The XMLQUERY function can be used to evaluate an XQuery expression on an XML document and return an XML document as a result:

```
SELECT   XMLQUERY ('<jobs> {$r/resume/jobHistory} </jobs>'
                   PASSING resume AS "r")
FROM     employees
WHERE    empid='166549'
```

XMLCAST can be used to cast a value of an XML type into an SQL type:

```
SELECT   XMLCAST (XMLQUERY ('($r/resume/jobHistory)[1]/jobTitle'
                            PASSING resume AS "r")
                  AS VARCHAR(30))
FROM     employees
WHERE    empid='166549'
```

XMLEXISTS can be used to evaluate an XQuery expression, returning true if the result contains any nodes (elements, attributes, etc.).

```
SELECT   empid
FROM     employees
WHERE    XMLEXISTS ('$r/resume/jobHistory[contains(., "Marketing")]'
                    PASSING resume AS "r")
```

Some products have extended SQL/XML with a function that allows an XML document to be transformed using an XSLT stylesheet.

```
SELECT   XSLTRANSFORM (resume USING SELECT   stylesheet
                                    FROM     stylsheets
                                    WHERE    xsltid='res_summary')
FROM     employees
WHERE    empid='166549'
```

Motivation

An OData service might publish an Employees entity set, with a resume property providing an XML document:

```
<Schema xmlns="http://schemas.microsoft.com/ado/2008/09/edm"
        Namespace="Personnel">
   <EntityContainer Name="MyCompany">
      <EntitySet Name="Employees" EntityType="Employee"/>
   </EntityContainer>
   <EntityType Name="Employee">
      <Key>
         <PropertyRef Name="empid"/>
      </Key>
      <Property Name="empid" Type="Edm.Int32" Nullable="false"/>
      <Property Name="lastname" Type="Edm.String" Nullable="false"
         MaxLength="30" FixedLength="false" Unicode="true"/>
      <Property Name="resume" Type="Edm.Stream" Nullable="true"
         MaxLength="Max" FixedLength="false"/>
   </EntityType>
</Schema>
```

A client might wish to query this entity set in a number of ways.

1. Retrieve only those employees that have “Marketing” in the job history located in their resume.
2. Return every employee, ordering the result based on the state in which they live, where that state is located in their resume.
3. Return the name and phone number of every employee, where the phone number has been taken from his or her resume.

4. Replace an old resume with a new resume for a specific employee.
5. Return a summarized resume for each employee.

Requirements

The following capabilities must be supported in this extension to OData:

- An OData Stream data type may be annotated to represent an XML data type
- XML properties may be returned separately from non-XML properties
- Entities may be filtered based on the content of their XML properties
- XML values that have been derived from XML properties may be retrieved
- Scalar values that have been derived from XML properties may be retrieved
- XMLQUERY, XMLEXISTS, and XMLCAST operations may be applied to XML properties
- The values of XML properties may be updated

Examples

The following examples describe possible annotations and extensions to OData to support XML data. Although concrete annotations, functions, and behavior are described, they are intended to be purely illustrative and not prescriptive. The Employees entity set might now be published as:

```
<Schema xmlns="http://schemas.microsoft.com/ado/2008/09/edm"
        Namespace="Personnel">
   <EntityContainer Name="MyCompany">
      <EntitySet Name="Employees" EntityType="Employee"/>
   </EntityContainer>
   <EntityType Name="Employee">
      <Key>
         <PropertyRef Name="empid"/>
      </Key>
      <Property Name="empid" Type="Edm.Int32" Nullable="false"/>
      <Property Name="lastname" Type="Edm.String" Nullable="false"
            MaxLength="30" FixedLength="false" Unicode="true"/>
      <Property Name="resume" Type="Edm.Stream" Nullable="true"
            MaxLength="Max" FixedLength="false">
         <ValueAnnotation Term="OData.ContentType" String="text/xml"/>
      </Property>
   </EntityType>
</Schema>
```

To retrieve a single employee, one might submit:

```
http://www.example.com/mycompany/Employees(166549)
```

The might return the following entry:

```
<entry xml:base=http://www.example.com/mycompany/Employees ...>
   <id>http://www.example.com/mycompany/Employees(166549)</id>
   <link
      rel=http://schemas.microsoft.com/ado/2007/08
                  /dataservices/mediaresource/resume
      type="application/xml"
      title="resume"
      href="resume166549.xml"/>
   <link
      rel=http://schemas.microsoft.com/ado/2007/08
                  /dataservices/edit-media/resume
      type="application/xml"
      title="resume"
      href="resume166549.xml"/>
   <m:properties>
      <d:empid>166549</d:empid>
      <d:lastname>Heinlein</d:lastname>
   </m:properties>
</entry>
```

To retrieve only those employees that have “Marketing” in the job history located in their resume, one might submit:

```
http://www.example.com/mycompany/Employees
?$filter=resume/XML.XMLEXISTS
                  (query = '$r//jobHistory[contains(., "Marketing")]',
                   vname= 'r')
```

Or, if the variable name can be defaulted to “it”:

```
http://www.example.com/mycompany/Employees
?$filter=resume/XML.XMLEXISTS
                  (query = '$it//jobHistory
                                       [contains(., "Marketing")]')
```

To return every employee, ordering the result based on the state in which they live, where that state is located in their resume, one might submit:

```
http://www.example.com/mycompany/Employees
?$orderby=resume/XML.XMLQUERY_STRING
                       (query = '$r/resume/address/state',
                        vname = 'r')
```

To return the name and phone number of every employee, where the phone number has been taken from his or her resume, one might submit:

```
http://www.example.com/mycompany/Employees
?$select=lastname,
         resume/XML.XMLQUERY_STRING
                       (query = '$r/resume/phone',
                        vname = 'r')
```

This query would require an extension to OData, allowing an expression to appear in a $select query option.

To replace an old resume with a new resume for a specific employee, one might submit:

```
PUT /resume166549.xml HTTP/1.1
Host: host
Content-Type: application/xml
DataServiceVersion: 1.0
MaxDataServiceVersion: 3.0
If-Match: ...Etag...
Content-Length: ####

<resume>
   <ssn>1234</ssn>
   <lastname>Handl</lastname>
   <address>
      <street>ABC St.</street>
      <zipcode>10022</zipcode>
   </address>
   <experience>excellent</experience>
</resume>
```

To return a summarized resume for each employee, one might submit:

```
http://www.example.com/mycompany/Employees
     /resume/XML.XMLQUERY
                     (query = '<summary>
                                 { $r/lastname,
                                   <experience>
                                      { $r/jobHistory/company }
                                   </experience>
                                 }
                               </summary>',
                      vname = 'r',
                      version='1.0',
                      XMLDeclaration=true)
```

With defaults for the variable name and serialization parameters, this could be shortened to:

```
http://www.example.com/mycompany/Employees
     /resume/XML.XMLQUERY
                     (query = '<summary>
                                 { $it/lastname,
                                   <experience>
                                      { $it/jobHistory/company }
                                   </experience>
                                 }
                               </summary>')
```

OData implementations should be able to analyze a query and raise an error if one of these XML functions is applied to a property that does not have a content type of text/xml or application/xml.

Design Principles

OData is an application-level protocol for interacting with data via RESTful web services. An OData Service's contract is defined by simple, well-defined conventions and semantics applied to the data model exposed by the service, providing a high level of semantic interoperability between loosely coupled clients and services.

The design principles of OData are to:

- Make it easy to implement and consume a basic OData service over a variety of data sources. Rather than try and expose the full functionality of all stores, define common features for core data retrieval and update scenarios and incremental, optional features for more advanced scenarios.
- Leverage Data Models to guide clients through common interaction patterns rather than force clients to write complex queries against raw data
- Define consistency across the protocol and a single way to express each piece of functionality

The design principles of OData extensions are to:

- Ensure extensions do not violate the core semantics of OData
- Avoid defining different representations for common concepts across extensions
- Ensure independent extensions compose well
- Ensure clients can ignore extended functionality and still query and consume data correctly

Technical Direction

The design of this extension to OData should take the following direction:

- An OData vocabulary for XML shall be defined.
- An annotation from a common vocabulary defining the content type should be applied to a Stream property that represents XML documents.
- The XML vocabulary will define functions that can be applied to XML properties.
- These functions will be based on functions found in SQL/XML:2011, and by extension will be based on XQuery 1.0.

Open questions, issues and work items

- Support may be provided for transforming an XML property by applying an XSLT stylesheet.
- The XML annotation may contain additional properties describing the XML document. These properties might include the location of the schema used to validate the document.
- Support may be provided for updating only a portion of an XML property.

- XML operations could define a default variable name for the document being processed. The document could be assigned as the context item for that query.
- OData could be extended to allow expressions in the $select query option, allowing derived values to be returned along with the properties of an entity.
- OData could be extended with an operator that returns the content of a Stream as either a String or Binary value.
- The OData.ContentType value annotation could be defined to allow multiple content types as its value.
- Support for a function similar to XSLTRANSFORM could be added.

References

1. Extensible Markup Language (XML) 1.0 (Fifth Edition), W3C Recommendation, Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau, 26 November 2008, http://www.w3.org/TR/xml/.
2. ISO/IEC 9075-14:2003 Information technology - Database languages - SQL - Part 14: XML-Related Specifications, (SQL/XML).
3. ISO/IEC 9075-14:2008 Information technology - Database languages - SQL - Part 14: XML-Related Specifications, (SQL/XML).
4. ISO/IEC 9075-14:2011 Information technology - Database languages - SQL - Part 14: XML-Related Specifications, (SQL/XML).
5. XQuery 1.0: An XML Query Language (Second Edition), W3C Recommendation, 14 December 2010, Scott Boag, Don Chamberlin, Mary F. Fernández, Daniela Florescu, Jonathan Robie, Jérôme Siméon, http://www.w3.org/TR/xquery/.
6. XSL Transformations (XSLT) Version 2.0, W3C Recommendation, Michael Kay, 23 January 2007, http://www.w3.org/TR/xslt20/.