xquery introduction

Bob Beauchemin

Outline

- Input and Output
- XQuery data model - Sequences
- Parts of an XQuery program
- Path and FLWOR
- Functions
  - Constructors
  - Accessors
  - Built-Ins

XQuery - A Query Language

- **What is XQuery?**
  - a declarative query language
  - works with a new data model
  - uses XPath to address nodesets
  - SQL-like expressions (FLWOR expressions)
  - W3C standard
  - a case-sensitive language

Input and Output

- **XQuery standard uses input functions**
  - doc() - single document
  - collection() - collection of documents
- **SQL Server uses XML data type methods**
  - query - xml output
  - exist - bit output
  - value - SQL type output
  - nodes - table output
  - modify - mutator

Data Models

- **XQuery accompanied by a data model**
  - XQuery 1.0 and XPath 2.0 data model
  - Closed data model
  - Can process untyped and typed XML
    - XML Infoset
    - Post Schema-Validation Infoset
  - Goes beyond nodes
    - Nodes and atomic values
    - Sequences

XQuery and XML Schema

- **XQuery can use XML Schema**
  - Schema is optional
    - well-formed or schema valid input
  - XML Schema is base type system
  - Additional XQuery types
    - xdt: untyped
    - xdt:untypedAtomic

Strong Typing

- **XQuery is a strong-typed language**
  - data type
  - cardinality
  - all expressions and functions use types

```
Definition of XQuery number() function

fn:number() as xs:double?
fn:number($arg as node()?) as xs:double?
```

Static Type-Checking

- **Static type-checking optional in XQuery**
  - SQL Server XQuery does static type-checking
  - two query phases
    - static evaluation phase - does not use data
    - dynamic execution phase - evaluations query against data
  - if errors in phase one, no phase two

Sequences

- **XQuery works with sequences**
  - can be nodes or atomic values
    - no heterogeneous sequences in SQL Server XQuery
  - sequences do not nest
  - input and output are sequences

Parts of an XQuery Program

- **XQuery program consists of**
  - prolog
    - namespace bindings
    - function, variables, xmlspace - not supported in SQL Server XQuery
  - body
    - query expressions

XPath and FLWOR

- **XQuery uses XPath for selection**
  - any XPath 2.0 is valid XQuery
- **XQuery uses FLWOR for extension**
  - iteration
  - sorting
  - construction

```
(: this is a valid XQuery :)
/people/person[age > 30]

(: so is this FLWOR expression :)
for $p in /people/person
where $p/age > 30
return $p
```

XPath Introduction

- **XPath got its name from its path-based syntax**
  - /books/book/title/chapter
- **Why a path-based syntax?**
  - Familiar
    - c:\Program Files\Microsoft SQL Server
    - http://www.microsoft.com/SQLServer
  - Intuitive for navigation of hierarchical structures
  - Compact (consider URLs or attribute values)

Location Step Evaluation

Example:
/books/book[1]/title/chapter

- A location step is evaluated against each node in the context node-set to produce a result node-set (the union of all node-sets identified)
- The result node-set then becomes the context node-set for the subsequent location step
- This is repeated for all location steps in the expression
- The result node-set produced by the final location step is the outcome of the expression

Axes

- **XPath defines several axes relative to a context node**
  - an axis identifies a group of nodes that have a specific relationship to the context node
  - axes allow you to move in different directions
  - axes are used in location steps to reduce the initial set of nodes in the context node-set

self	context node
child (default)	context node's child nodes
parent	context node's parent node
descendant[-or-self]	context node's descendant nodes (child of child of child, etc.)
attribute	context node's attribute nodes

FLWOR

- **FLWOR is the core of XQuery**
  - for - iteration
  - let - assignment
  - where - filtering
  - order by - sorting
  - return – results
- **FLWOR is similar to SQL in structure**
  - Easier than XSLT for SQL programmers to learn

SQL and FLWOR

SQL

```
SELECT p.name, p.job
FROM people AS p
WHERE p.age > 30
ORDER BY p.age
```

FLWOR

```
for $p in /people/person
where $p/age > 30
order by $p/age[1]
return ($p/name, $p/job)
```

Scope of FLWOR

- FLWOR starts at first for or let
- FLWOR ends at return

```
for $p in /people/person
for $e in /emp
where $p/age > 30
order by $p/age[1]
return ($p/name, $e/job)
```

for clause

- ***for* clause creates iterator over a nodeset**
  - set of nodes is bound to a variable
  - variable references **one node from set** during any given iteration
  - variable can be used in subsequent clauses
  - used with construction to transform XML
  - *for* cannot be used on constructed XML

```
(: one return for each person :)
for $p in /people/person
return $p
```

let clause

- ***let* clause assigns a nodeset to a variable**
  - set of nodes is bound to a variable
  - variable **always references entire set** of nodes
  - variable can be used in subsequent clauses
  - executed each time statement is referenced
  - let cannot be used on constructed XML
  - note: let not supported until SQL Server 2008

```
(: one sequence of person returned :)
let $p := /people/person
return $p
```

where clause

- **A *where* clause filters variables defined by *for* or *let***
  - defines condition that a variable binding must satisfy
  - if condition evaluates to true the current node is retained
  - otherwise it is dropped
  - works like an XPath predicate
    - XPath predicates can be used instead

```
(: using where  :)
for $p in /people/person
where $p/age[1] gt 30
return $p/name
```

Comparison Operators

- **XQuery has different types of comparison operators**
  - general (existential) comparison
    - =, !=, >, <, >=, <=
  - value comparison
    - eq, ne, gt, lt, ge, le
  - node comparison
    - is
  - order comparison
    - <<, >>

order by clause

- XQuery can easily sort by using order by
  - similar to SQL ORDER BY clause
  - must resolve to a scalar value
    - nodesets not comparable

```
(: using order by  :)
for $p in /people/person
where $p/age[1] gt 30
(: order by requires one age :)
order by $p/age[1]
return $p/name
```

return clause

- **A *return* clause defines the query result**
  - brackets - *for* or *let*
  - can reference variable bindings
  - can emit static XML
  - can emit dynamic XML using constructors

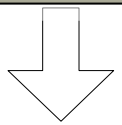
```
(: one sequence of text nodes returned :)
let $p := /people/person
return $p/name/givenName/text()
```

Construction

- **XQuery can construct new content**
  - Direct constructors
    - Direct use of XML-like syntax
    - Static constructors
    - Dynamic constructors
  - Computed constructors
    - For element, attribute, text
  - Constructor functions
    - For atomic values

Direct Constructors

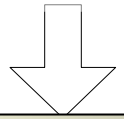
```
return <person><name>

{$p/name[1]/givenName[1]/text()}
        </name></person>
```



```
<person>
    <name>Martin</name>
</person>
<person>
    <name>Simon</name>
</person>
```

Computed Constructors

```
return element person
   {
    attribute name
    {data($p/name[1]/givenName[1]/text()[1])}
   }
```



```
<person name="Martin"/>
<person name="Simon"/>
```

Constructor Functions

```
(: construct dateTime atomic values for comparison :)

for $PD in  /root/ProductDescription
where
  xs:dateTime(data(($PD/@DateCreated)[1]))
    < xs:dateTime("2001-01-01T00:00:00Z")
return
  element Product
  {
    ( attribute ProductID {data($PD/@ProductID)},
      attribute DateCreated
                          {data(($PD/@DateValue)[1])}
    )
  }
```

Constructor Limitations

- **Cannot use any type of constructor to construct element or attribute name**
- **No computed constructor for**
  - document
  - comment
  - processing instruction
- **Computed attribute constructors cannot be used to declare new namespaces**
- **No constructor functions for**
  - Time duration data types
  - xs:QName, xs:NMTOKEN, xs:NOTATION

XQuery Expressions

- **XQuery supports rich sets of operators**
  - Sequence operators
    - Construct, filter, combine sequences
  - Arithmetic operators
    - +, -, *, div
    - Workaround for idiv : xs:integer (5 div 3)
  - Comparison operators - discussed earlier
  - Logical operators
    - and, or
  - Conditional expressions
    - if-then-else
  - Quantified expressions
    - some/every *variable* in *expression* satisfies ....
  - Sequence type expressions
    - instance of

XQuery Function Library

- **SQL Server implements XQuery functions**
  - constructors - discussed earlier
  - string functions
  - data accessors
  - context functions
  - numeric functions - ceiling, floor, round
  - aggregate functions - min, max, avg, sum, count
  - sequence functions - empty, distinct-values, id
  - node functions - number, local-name, namespace-uri
  - qname functions- expanded-QName, local-name-from-QName, namespace-uri-from-QName

String Functions

Function	Description
concat	returns the concatenation of the arguments
contains	returns true if the first string contains the second
substring	returns the specified substring
string-length	returns the length of the string
upper-case	converts string to all upper case*
lower-case	converts string to all lower case*

* Available in SQL Server 2008

Data Accessors

Function	Description
data	returns the atomized value
string	returns value of a string

Context Functions

Function	Description
last	returns the last member of sequence
position	returns sequence member at this position

Note: These can only be used inside brackets (in a predicate)

XQuery Functions

- **SQL Server does not implement all functions**
  - trace
  - error
    - Runtime error returns an empty sequence
    - XML methods (XQuery) require T-SQL arithabort option
  - date time functions
  - subset of
    - numeric
    - string
    - node test
  - …are not implemented

sql: User-defined functions

- **XQuery allows user-defined functions**
  - functions using T-SQL items are provided
    - sql:column - column value from the outer FROM clause
    - sql:variable - T-SQL variable value
  - user-defined functions not supported
  - modules not supported

Combining documents

- **XML.Query method returns XML**
  - on XML variable, returns one answer
  - on XML table, returns one answer/row
  - one combined answer using FOR XML
- **XML.Nodes method returns table**
  - one column in table
  - one row in table per matching node

SQL Server XQuery

- Aligned with July 2004 standard
- Static type checking
- Schema support
- XML methods (XQuery) require T-SQL arithabort option
- Subset of standard
- Subset of functions and operators
- Not all options supported on operations

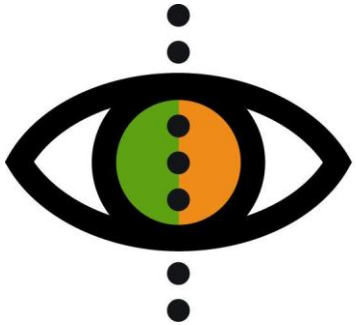
Review

- **XQuery is a new SQL-like query language**
  - Exposed in SQL Server through XML methods
- **Uses a new data model that adds**
  - Atomic Values
  - Sequences
- **XQuery is a strongly typed language**
- **Static type checking is optional**
  - SQL Server has this feature
- **Queries can contain a prolog and body**
- **Vendors can add user-defined functions**

References

- A Developer's Guide to SQL Server 2005 - Ch 10, Addison-Wesley, Bob Beauchemin and Dan Sullivan
- Pro SQL Server 2008 XML, APress, Michael Coles
- XQuery From the Experts, Addison-Wesley, Howard Katz, ed.
- XQuery: The XML Query Language, Addison-Wesley, Michael Brundage
- XQuery in Relational Database Systems - Michael Rys
- XQuery Implementation in a Relational Database System, Shankar Pal, et al., Conference on VLDB

For more in-depth online developer training visit



pluralsight

see what you can learn

on-demand content from authors you trust