XML Query Language

aus GlossarWiki, der Glossar-Datenbank der Fachhochschule Augsburg
Wechseln zu:Navigation, Suche

Inhaltsverzeichnis

1 Definition

Die **XML Query Language** ist eine Abfragesprache für XML-Datenbanken und wird von der „XML Query Working Group“, einer Untergruppe der „XML Activity“ des W3C, entwickelt. Derzeit im Status „Working Draft“ soll XQuery bald zum W3C-Standard werden.

Als funktionale Sprache arbeitet XQuery ohne Seiteneffekte. Das bedeutet, dass weder Daten dauerhaft oder auch nur temporär gespeichert, noch Inhalte modifiziert werden können.

2 Sprachelemente

Die Navigation innerhalb der Dokumente erfolgt mit Hilfe von XPath. Durch XPath kann jeder Knoten des durch das XML-Dokument erstellten Baums erreicht werden. Die in XPath vorhandenen Funktionen und elementaren Ausdrücke können ebenfalls verwendet werden. Man spricht deshalb von XPath als einer Untermenge von XQuery.

2.1 Das FLWOR-Muster

Komplexere Ausdrücke werden nach dem **FLWOR**-Muster (**f**or-**l**et-**w**here-**o**rder-**b**y-**r**eturn) aufgebaut:

```
for $var1 at $pos in EXPR1, ...
let $var2 :=         EXPR2, ...
where                EXPR3
order by             EXPR4 ascending/descending
return               EXPR5
```

Entsprechend der SELECT-Anweisung in SQL kann auch in XQuery zur Abfrage der Daten eine Schleife (`for`) durchlaufen werden. Das Element des jeweiligen Schleifenschritts wird dabei in einer Konstanten (`$var1`, s. o.) gespeichert, über die in den folgenden Anweisungen auf dessen Inhalt zugegriffen werden kann. In einer Konstanten (`$pos`) kann die Position des Elementes gespeichert und auf diese, beispielsweise in der where-Bedingung, zugegriffen werden (z.B. `where $posvar1 mod 2 = 0` bedeutet jeder 2te).

Im `let`-Abschnitt können Konstanten definiert, mit Werten belegt und Berechnungen durchgeführt werden. In jedem Schritt werden diese Konstanten neu definiert, ihr Inhalt wird also nicht über den Schleifenschritt hinaus gespeichert.

Es ist zudem möglich mehrere weitere `let`-Anweisungen vor dem Aufruf der for-Schleife(n) zu platzieren. Auf diese Daten kann während des gesamten Schleifendurchlaufs zugegriffen werden. Um Berechnungen mit Daten aus einer Abfrage auszuführen, müssen diese jedoch nach der Zuweisung für `$var1` durchgeführt werden.

In der `where`-Bedingung wird jedes Element auf die angegebenen Kriterien geprüft. Hier können mehrere Bedingungen, die entweder mit `and` oder `or` verknüpft werden, angegeben werden. Im Gegensatz zu SQL können mit Xquery keine negativen Bedingungen (`not`) angegeben werden.

Erfüllt dieses die Bedingungen, so wird es in der `return`-Anweisung verarbeitet und alle gefundenen Daten nach dem `order-by`-Kriterium geordnet zurück gegeben.

Im folgenden Beispiel werden aus einem XML-Dokument die Nach- und Vornamen aller Personen ausgewählt und als neues XML-Dokument ausgegeben.

personen.xml

```
<personen>
  <person gender="f">
    <last>Mustermann</last> <first>Meike</firste>
  </person>
  <person gender="m">
    <last>Mustermann</last> <first>Max</firste>
  </person>
<personen>
```

xquery

```
<namen>
{
   for $person in  document("personen.xml")//person
   let $l:=string($person/last),
       $f:= string($person/first)
   return
     <name>{$f}{" "}{$l}</name>
}
</namen>
```

Resultat

```
<namen>
   <name>Meike Musterfrau</name>
   <name>Max Musterman</name>
 </namen>
```

Die Konstanten in XQuery-Ausdrücken werden jeweils mit einem vorangehenden $-Zeichen gekennzeichnet.

Den Konstanten, aber auch den Inhalten der Knoten, können explizit bestimmte Datentypen zugewiesen werden. Diese können als Bezeichner, zur Typüberprüfung oder auch für explizite Typecasts verwendet werden (Bsp.: `string($person/last)`).

Bei der Rückgabe der Elemente muss jeder neue Knotentyp (Attribute, Elemente, Text, etc.) in geschweifte Klammern eingefasst werden. Ausschließlich XML-Tags können ohne diesen Rahmen ausgegeben werden. (z.B.: `<name>{"Frau"} {$last}</name>`) Hierbei muss man darauf achten, wiederum wohlgeformtes XML zu produzieren.

Beim Schreiben der where-Klauseln muss eine Besonderheit beachtet werden. Da im XML-Umfeld

stets zwischen dem kompletten Knoten und dessen Inhalt unterschieden wird, werden auch hier unterschiedliche Vergleichsoperatoren angeboten.

2.2 Vergleichsoperatoren

Operatoren für die Prüfung des Wertes eines Elements:
: `eq` (equal), `ne` (not equal), `t` (less than), `le` (less or equal), `gt` (greater than), `ge` (greater or equal)

Operatoren für die Prüfung der Existenz eines Elements:
: =, !=, <, <=, >, >=

Operatoren für die Prüfung der Abfolge von Elementen:
: `is`, <<, >>

2.3 Weitere Elemente

Neben den bereits genannten FLWOR-Ausdrücken verfügt XQuery über weitere Elemente, die seine Funktionalität und Möglichkeiten erweitern. So können auch Listen (z.B.: `(1, 2, 4)`) oder Integerränge (z.B.: `1 to 3`, d. h. die Zahlen 1, 2 und 3) definiert werden.

Um mehrere Knotenfolgen zu kombinieren, können, wie in SQL auch, die Operatoren `union`, `intersect` oder `except` verwendet werden. Die Auswertung der Knoten und ihres Inhalts anhand von Bedingungen ist in XQuery möglich. Hierzu kann das `if-then-else`-Konstrukt verwendet werden. Knotenmengen können auch anhand quantifizierender Ausdrücke überprüft werden (`some-in-satisfies`, `every-in-satisfies`).

In XQuery können Kommentare, auch geschachtelt, verwendet werden. Diese werden jeweils durch einen Smilie eingeleitet und abgeschlossen (z.B.: `(: Kommentar :)`). Auch Kommentare im XML-Format sind erlaubt (z.B.:).

2.4 Benutzerdefinierte Funktionen

In XQuery können XPath-Funktionen verwendet werden.

Ein Besonderheit, die XQuery bietet, ist jedoch die Möglichkeit zur Definition benutzerdefinierter Funktionen:

```
declare function namespace:function (para1 as type1, ...)
as ret
{
  <expr>
}
```

In diesen Funktionen können weitere Funktionen aufgerufen werden. Auch der rekursive Aufruf von Funktionen ist an dieser Stelle möglich. Gerade die Möglichkeit zur Verwendung von Rekursion in den benutzerdefinierten Funktionen unterscheidet XQuery grundlegend von SQL.

Das folgende Beispiel gibt aus einem XML-Dokument, in dem Familienmitglieder über eine ID verknüpft sind, den Stammbaum aller Ottos aus. Dazu wird zuerst eine lokale benutzerdefinierte Funktion erstellt, die sich rekursiv selbst aufruft, bis keine Vorfahren mehr gefunden werden können. Diese Funktion kann anschließend unter Angabe ihres Namespaces beispielsweise aus einer

for-Schleife aufgerufen werden.

```
declare function local:getfamilie($id as xs:string) as node()*
{
   <mitglied>
     {document("personen.xml")//person[id=$id]/last}
     {document("personen.xml")//person[id=$id]/first}
     <mutter>
     {  let $mid as xs:string := //person[id=$id]/mutter
        return if ($mid) then local:getfamilie(string($mid)) else ()
     }
     </mutter>
     <vater>
     {  let $vid as xs:string := //person[id=$id]/vater
        return if ($vid) then local:getfamilie(string($vid)) else ()
     }
     </vater>
   </mitglied>
}

for $e in document("personen.xml")//person[first="Otto"]
  return <familie>{local:getfamilie(string($e/id))}</familie>
```

3 Beispiele

siehe XML_Query_Language/Beispiele

4 Quellen

- http://www.w3.org/TR/xquery
- http://www.w3.org/XML/Query
- http://www.w3schools.com/xquery/xquery_intro.asp
- http://www.jeckle.de/vorlesung/xml/script.html
- http://demo.exist-db.org/xquery/functions.xq
- http://de.wikipedia.org/wiki/XQuery
- Lehner, Wolfgang. Schöning, Harald. „XQuery" – Ein Überblick. Datenbank-Spektrum. Ausgabe 11/2004.
- Sasaki, Felix. Milde, Jan-Torsten. Fragestunde. Neuerungen in XQuery und XSLT.XML & Web Services Magazin. Ausgabe 4.05. Software und Support Verlag GmbH.

- XML
- Glossar

Versteckte Kategorien:

- GlossarWiki-konformer Artikel
- Anmerkung
- Quellenangaben verbessern
- Quellenangaben verbessern: Formatierung

- Links auf diese Seite
- Änderungen an verlinkten Seiten
- Spezialseiten
- Permanenter Link
- Seiteninformationen
- Attribute anzeigen

- Diese Seite wurde zuletzt am 17. Mai 2019 um 14:46 Uhr bearbeitet.
-

- Datenschutz
- Über GlossarWiki
- Lizenzbestimmungen

-
-
-