XQuery Einführung

Stephan Karrer

Was ist XQuery

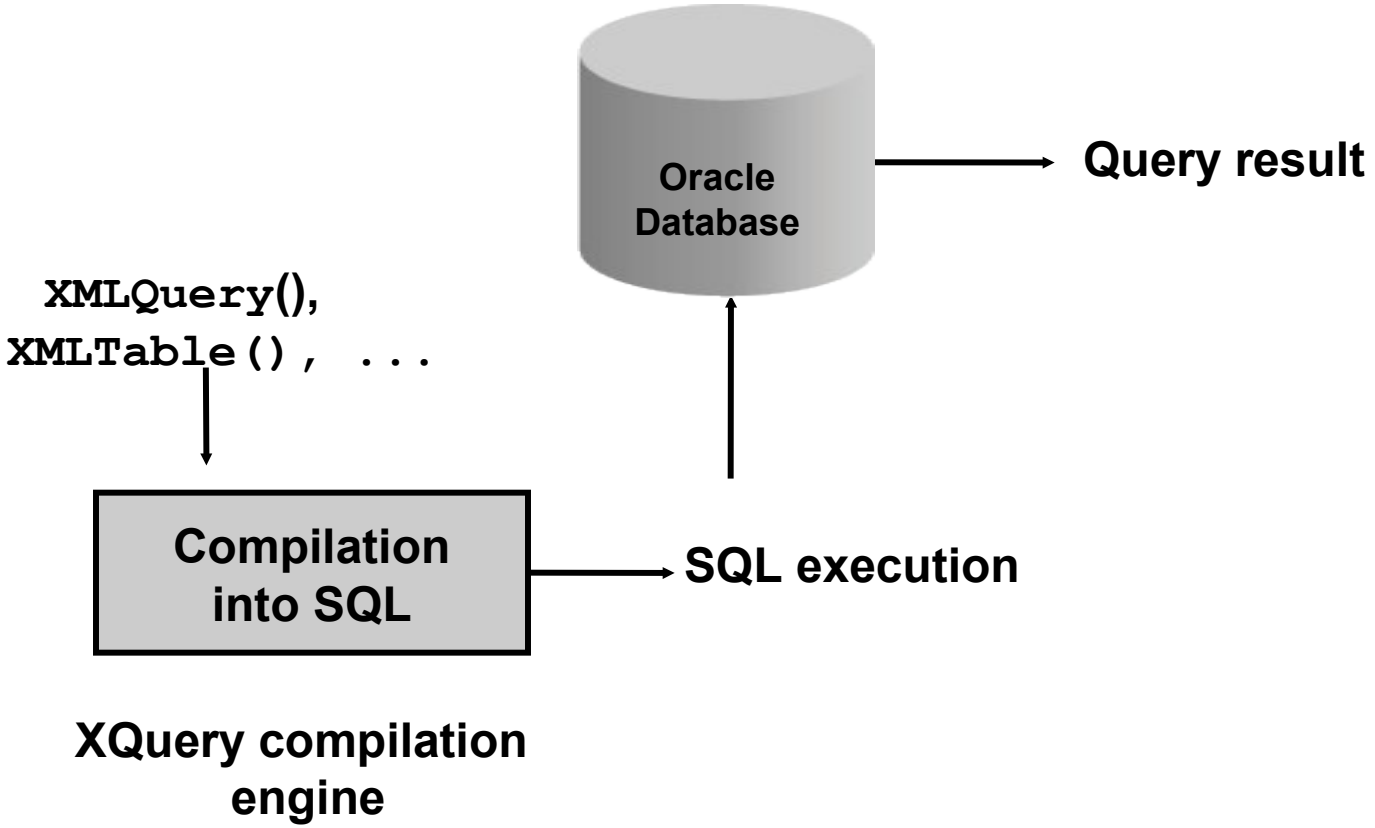
- XQuery ist die allgemeine Abfragesprache für XML-Daten (wie SQL in der relationalen Welt)
- XQuery ist offizieller W3C-Standard und wird durch eine Vielzahl von Produkten unterstützt
- XQuery erweitert XPath: XPath-Ausdrücke sind auch XQuery-Ausdrücke
- Via XQuery kann auch XML erzeugt bzw. verändert werden (Update)
- Oracle unterstützt folgende Standards:
  - XQuery 1.0 Recommendation
  - XQuery Update Facility 1.0 Recommendation
  - XQuery and XPath Full Text 1.0 Recommendation

XQuery: Merkmale

- Kann prinzipiell für alle möglichen XML-Quellen benutzt werden:
  - Relationale Datenbanken
  - XML-Dokumente
  - Andere Datenquellen mit XML-Sicht auf die Daten
- Unterstützt den XMLType

```
SELECT XMLQUERY(
    'fn:collection("oradb:/HR/DEPARTMENTS")'
    RETURNING CONTENT) AS EMP_DEPARTMENTS
 FROM dual;
```

XQuery Support in Oracle XML DB

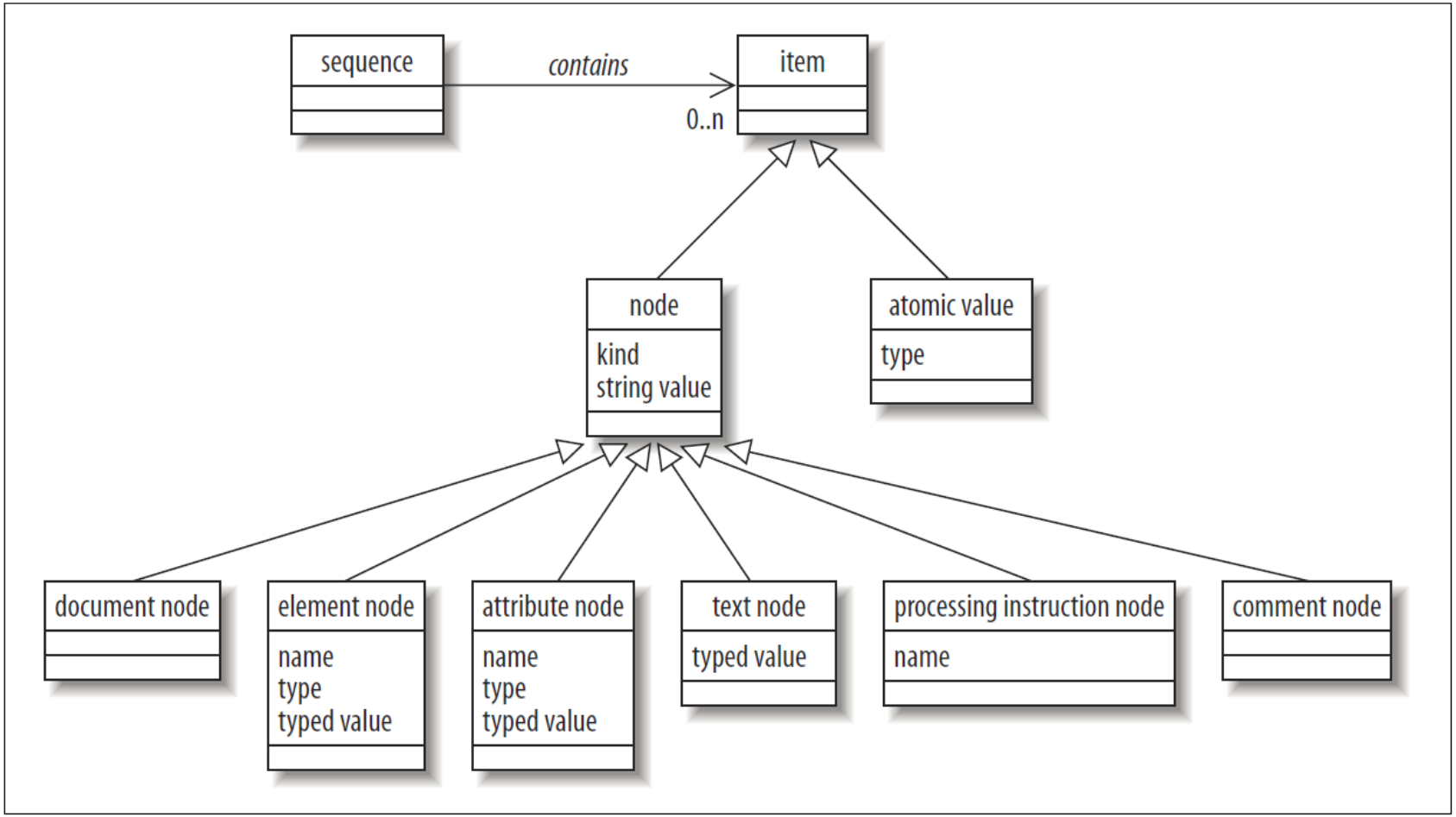


Das Datenmodell von XQuery

Das Datenmodell von XQuery

- Basiert auf Sequenzen von atomaren Werten oder XML-Konoten
- Ist eine abstrakte Repräsentaion eines oder mehrerer XML-Dokumente
- Repräsentiert ein Dokument als Baum von Knoten
- Unterscheidet sich durch die Hierarchie vom relationalen Modell
- Unterstützt die Identität von Knoten

Datentypen in XQuery



XQuery-Funktionen bei Oracle

- Oracle XML DB implementiert XQuery-Unterstützung durch SQL/XML-Funktionen (gemäß dem SQL/XML-Standard):
  - XMLQuery()
  - XMLTable ()
  - XMLExists()
  - XMLCast()
- SQL*Plus-Kommando XQUERY:
  Direkte Eingabe eines XQuery-Ausdrucks
- Alle Funktionen werden gleichermaßen optimiert (Query Rewrite)
- Es gibt funktionale Überschneidungen mit den SQL/XML-Generierungsfunktionen (XMLElement(), XMLAgg(), ...) sowie der Verwendung von XSLT

XMLQuery(): Einfache Beispiele

`XMLQuery()` wird benutzt zur Abfrage von XML-Daten bzw. deren Generierung

```
-- Query-Ausdruck kann eine Sequenz von atomaren
-- Werten sein
SELECT XMLQUERY('(1,2+3, "a")'
   RETURNING CONTENT) as ergebnis
FROM dual;
```

```
-- Query-Ausdruck kann eine XML-Knoten sein
SELECT XMLQuery ('<TEST>34</TEST>'
RETURNING CONTENT) AS ergebnis
FROM DUAL;
```

Funktion XMLTable( )

`XMLTable()`

- Bildet das Ergebnis einer XQuery-Auswertung auf relationale Zeilen und Spalten ab
  - damit kann das Ergebnis in der relationalen Welt weiterverarbeitet werden
- Wird typischerweise in der FROM-Klausel verwendet

```
SELECT * FROM XMLTABLE(
  'fn:collection("oradb:/OE/WAREHOUSES")/ROW/LOCATION_ID');
```

Abfrage relationaler Daten via XQuery

Oracle-spezifische Funktion **`ora:view()`** (deprecated ab 11.2)

```
SELECT XMLQuery(
    'ora:view("DEPARTMENTS")'
   RETURNING CONTENT) AS EMP_DEPARTMENTS
FROM dual;
```

Ab 11.2 sollte stattdessen **`fn:collection`** mit dem Oracle URI-Schema **`oradb`** verwendet werden:

```
SELECT XMLQUERY(
   'fn:collection("oradb:/HR/DEPARTMENTS")'
   RETURNING CONTENT) AS EMP_DEPARTMENTS
FROM dual;
```

XQuery-Ausdrücke

XQuery-Ausdrücke können sein:

- Primäre Ausdrücke: Literale, Variablen, Konstruktoren
- Sequenzen
- FLWOR-Ausdrücke (For-Let-Where-OrderBy-Return)
- Pfad-Ausdrücke
- Arithmetische Ausdrücke (Berechnungen)
- Logische Ausdrücke
- Bedingungen
- Quantifizierende Ausdrücke

Variablen und Literale

- Variable-Namen beginnen mit $-Zeichen
- Als Literale sind Zeichenketten, Zahlen (Ganzzahlen und Gleitpunktzahlen), Datums- bzw. Zeitstempel möglich
- Typkonvertierung erfolgt implizit, sofern möglich
- Es gibt auch Kommentare

```
SELECT XMLQUERY(' let $num := 1
                  (: dies ist ein Kommentar :)
                  let $erg := $num + 2
                  let $str := "Hallo"
                  let $d   := xs:date("2017.1.1")
                  return ($num, $str, $erg, $d)'
 RETURNING CONTENT) as output FROM dual;
```

Konstruktor-Ausdrücke

Konstruktor-Ausdrücke generieren XML-Konstrukte:

- Direkter Konstruktor-Ausdruck
- Berechneter Konstruktor-Ausdruck (computed constructor)

```
SELECT XMLQuery(' ( <A>33</A>  (: direkt :),
                  element book {
                    attribute isbn {"isbn-0060229357" },
                    element title { "Harold"},
                    element author {
                        element first { "Crockett" },
                        element last {"Johnson" }
                     }
                  }   (: berechnet :)
                ) '
RETURNING CONTENT) AS output
FROM DUAL;
```

Sequenzen

- XQuery manipuliert Sequenzen.
- Eine Sequenz ist eine Liste von Elementen, die XML-Knoten oder einfacher Inhalt sein können
- Der Komma-Operator trennt die einzelnen Elemente
- Sequenzen werden in der Regel in runde Klammern eingeschlossen
- Eine Sequenz aus einem Element entspricht diesem Element
- Geschachtelte Sequenzen werden flach geklopft

```
SELECT XMLQuery('(1, 2 + 3, "a", 100 to 102, ("x", "y"),
                 <A>33</A>)'
RETURNING CONTENT) AS output
FROM DUAL;
```

FLOWR-Ausdrücke

Ein FLWOR-Ausdruck ist die mächtigste Form in XQuery:

- Das Akronym FLWOR steht für:
  FOR - LET - WHERE - ORDER BY - RETURN
- Unterstützen Variablen und Iteration
- Bauen Sequenzen in gewünschter Ordnung auf

```
for $identifier in expr(expr)
let $identifier := expr(expr)
where conditional_expr
order by identifier("ascending" |
   "descending")
return expr_containing_result
```

Beispiel für FLOWR-Ausdruck

```
SELECT XMLQuery( '
        for $i in
          fn:collection("oradb:/HR/DEPARTMENTS")/ROW
        let $j := 20
        where $i/DEPARTMENT_ID > $j
        order by $i/DEPARTMENT_NAME
        return $i/DEPARTMENT_NAME/text() '
  RETURNING CONTENT)AS DEPARTMENT_NAME
  FROM DUAL;
```

Pfad-Ausdrücke

- Ein Pfad-Ausdruck selektiert Knoten im XML-Baum via XPath
- Die Rückgabe ist stets eine Sequenz von Knoten in Dokumentordnung
- Ein Pfad besteht aus eine Folge von Schritten
- Jeder Schritt besteht aus der Angabe einer Achse, einem Knotentest und optionalen Prädikaten zur Einschränkung der Knotenmenge

```
axis::node-test[predicates]
```

XPath: Achsen

- child: Kindelemte.
- parent: Direkter Vorfahr.
- descendant: Benannter oder erster Nachfahre.
- ancestor: Benannter oder erster Vorfahre.
- following: Menge aller Nachfahren.
- preceding: Menge aller Vorfahren.
- following-sibling: Nachfolgenden Geschwister.
- preceding-sibling: Vorherigen Geschwister.
- attribute: Attribute des Knotens.
- namespace: Namensraumkürzel.
- self: Der aktuelle Knoten.
- descendant-or-self: descendant und der aktuelle Knoten.
- ancestor-or-self: ancestor und der aktuelle Knoten.

Knotentest

Die Knoten können über folgende Typprüfungen eingeschränkt werden (Knotentest):

- <node-name> Knoten mit diesem Namen
- @<node-name> Attribute mit diesem Namen
- * Alle Knoten bzgl. des Achsentyp
- text() Alle Textknoten
- comment() Alle Kommentarknoten
- node() Alle Knoten
- processing-instruction(<name>) Alle Verarbeitungsanweisungsknoten mit dem optionalen Namen „name“

Beispiele zu Pfadausdrücken

child::text()

- Dieser Location-Path wählt alle Textelemente aus, die direkte Nachkommen des Kontextknotens sind

/child::department

- Wählt department-Elemente aus,die Kinder des Wurzelknotens sind

/descendant-or-self::node()/child::department

- Wählt alle department-Elemente im Dokument aus

/parent::node()/child::table/attribute::name

- Wählt die name-Attribute aller table-Elemente aus, die Kinder des übergeordneten Knotens sind

following-sibling::*/child::tr/child::td

- Wählt alle td-Elemente aus, die innerhalb eines tr-Elements stehen, das in einem nachfolgendem Geschwisterknoten des aktuellen Elements steht

Kurzformen für Pfadausdrücke

Einige oft benutzte Teilausdrücke lassen sich auch in Kurzform schreiben:

- (nichts) statt child:: (weglassen)
- @ statt attribute::
- . statt self::node()
- .. statt parent::node()
- // statt /descendant-or-self::node()/
- [number] statt [position()=number]

Beispiele zu Pfadausdrücken in Kurzform

text()

- Dieser Location-Path wählt alle Textelemente aus, die direkte Nachkommen des Kontextknotens sind

/department

- Wählt department-Elemente aus,die Kinder des Wurzelknotens sind

//department

- Wählt alle department-Elemente im Dokument aus

../table/@name

- Wählt die name-Attribute aller table-Elemente aus, die Kinder des übergeordneten Knotens sind

following-sibling::*/tr/td

- Wählt alle td-Elemente aus, die innerhalb eines tr-Elements stehen, das in einem nachfolgendem Geschwisterknoten des aktuellen Elements steht

Komplexere Schritte

Beispiele starten mit **doc("catalog.xml")/catalog/)**

Example	Meaning
`product/(number | name)`	All `number` AND `name` children of product.
`product/(* except number)`	All children of `product` except `number`. See "Combining Results" in Chapter 9 for more information on the | and `except` operators.
`product/` `(if (desc) then desc else name)`	For each `product` element, the `desc` child if it exists; otherwise, the `name` child.
`product/substring(name,1,30)`	A sequence of `xs:string` values that are substrings of product names.

Einschränkung der Knotenmenge durch Prädikate

Die Prädikate werden für jeden Knoten der aktuellen Menge ausgewertet. Falls das Prädikat zu „true“ evaluiert, verbleibt der Knoten in der Menge.

- Positionsangabe bzgl. der Reihenfolge (Zählung beginnt mit 1)
  - /departments/department[2]
- Jeder Schritt im Pfad kann Prädikate verwenden
  - //department[@num<3]/department_id[.="10"]
- Prädikate können mit „and“ und „or“ kombiniert werden
  - //department[@num>2 and @num<=4]/department_name
- Prädikate können auch durch Reihung kombiniert werden
  - //department[@num>2][@location<=10]/department_name

Positionale Prädikate

Beispiele starten mit **doc("catalog.xml")/catalog/)**

Example	Description
`product[2]`	The second `product` child of `catalog`
`product[position( ) = 2]`	The second `product` child of `catalog`
`product[position( ) > 1]`	All `product` children of `catalog` after the first one
`product[last( )-1]`	The second to last `product` child of `catalog`
`product[last( )]`	The last `product` child of `catalog`
`*[2]`	The second child of `catalog`, regardless of name
`product[3]/*[2]`	The second child of the third `product` child of `catalog`

Bedingungen

```
select XMLQuery(
'for $i in ora:view("DEPARTMENTS")/ROW
where $i/DEPARTMENT_ID < 30
return <Result>
       <department_name> {$i/DEPARTMENT_NAME/text()}
                 </department_name>
       <location> { if ($i/LOCATION_ID = 1700)
                                    then "Seattle"
                                 else "Elsewhere" }
                 </location>
          </Result>'
RETURNING CONTENT) AS XMLRES
FROM DUAL;
```

Quantifizierende Ausdrücke

- XQuery 2 Quantifizierende Operatoren:
  - every
  - some
- Der Wert eines quantifizierenden Ausdrucks ist immer ein Wahrheitswert (true, false)
- Quantifizierende Ausdrücke sind Abkürzungen für FLOWR-Ausdrücke

```
SELECT XMLQuery(
'some $i in (1 , 2), $j in (22, 4, 2)
satisfies $i = $j'
RETURNING CONTENT) AS SOME
FROM DUAL;
```

Unterstützte XQuery Funktionen

Oracle XML DB unterstützt die spezifizierten Funktionen aus XQuery 1.0 und XPath 2.0. Dazu gehören beispielsweise:

- Konstruktor-Funktionen
- Numerische Funktionen: fn:abs(), fn:ceiling(), fn:floor(), fn:round()
- String-Funktionen: fn:concat(), fn:substring(), fn:lower-case(), fn:upper-case()
- Aggregatsfunktionen: fn:count(), fn:avg(), fn:max(), fn:min(), fn:sum()