# Oracle XML DB - Verwendung von XQuery

Stephan Karrer

## Abfrage von Daten via XQuery

- Zur Abfrage werden XQuery-Ausdrücke in die XMLQuery-Funktion eingepackt
- Über die Benutzung von Views können auch relationale Daten via XQuery abgefragt werden bzw. im XML-Format bereit gestellt werden
  - Hierbei werden die Zeilen in zusätzliche Wrapper (default ROW) eingepackt.

```
SELECT XMLQuery(
      'for $i in fn:collection("oradb:/HR/DEPARTMENTS")
       where $i/ROW/DEPARTMENT_NAME
             = "Human Resources"
        return $i'
        RETURNING CONTENT) AS HR
 FROM dual;
```

## Join via XQuery

- Durch Schachtelung der FOR-Ausdrücke in XQuery entsteht der JOIN
- Weniger deskriptiv als in SQL

```
SELECT XMLQuery(
  'for $i in fn:collection("oradb:/HR/EMPLOYEES")/ROW
   return <Employee id ="{$i/EMPLOYEE_ID/text()}">
          <Department>
          {for $j in
    fn:collection("oradb:/HR/DEPARTMENTS")/ROW
          where $j/DEPARTMENT_ID eq $i/DEPARTMENT_ID
          return ($j/DEPARTMENT_NAME, $j/LOCATION_ID)}
          </Department>
         </Employee>'
RETURNING CONTENT) AS res FROM DUAL;
```

## Abfrage von XML-Daten via XQuery

- Wir können natürlich die VIEW-Technik nutzen, ist aber überflüssig
- Stattdessen wird üblicherweise die XML-Spalte übergeben

```
-- Abfrage via VIEW
SELECT XMLQuery(
     'fn:collection("oradb:/OE/PURCHASEORDER")'
    RETURNING CONTENT) AS res
 FROM dual;

-- direkte Abfrage
SELECT XMLQuery(
    'for $i in /PurchaseOrder return $i'
       PASSING OBJECT_VALUE
    RETURNING CONTENT) AS res
 FROM OE.PURCHASEORDER;
```

## Nutzung von XMLTable() zur Abfrage

- Via XMLTable-Funktion ist ebenfalls ein JOIN XML-basierter Daten möglich
- Es kann sich hierbei um virtuelle Tabellen handeln

```
SELECT xtab.column_value
FROM purchaseorder p,
    XMLTable('for $i in /PurchaseOrder
              where $i/CostCenter eq "A10"
              return
                 <A10po pono="{$i/Reference}"/>'
              PASSING OBJECT_VALUE
) xtab;
```

## XMLTable() zur Generierung relationaler Ergebnisse

- Die häufigste Verwendung besteht in der Erzeugung relationaler Ergebnisse

```
SELECT xtab.poref, xtab.usr, xtab.requestor
FROM purchaseorder,
    XMLTable(
     'for $i in /PurchaseOrder
      where $i/CostCenter eq "A10"
      return $i'
    PASSING OBJECT_VALUE
    COLUMNS
     poref VARCHAR2(20) PATH 'Reference',
     usr   VARCHAR2(20) PATH 'User'
                         DEFAULT 'Unknown',
     requestor VARCHAR2(20) PATH 'Requestor'
  ) xtab;
```

## XMLExists() zur Filterung

- XMLExists  prüft, ob ein XQuery-Ausdruck eine nicht-leere Sequenz liefert und gibt in disem Fall true zurück, sonst false
- Soll statt der „deprecated“ Funktion ExistsNode verwendet werden

```
SELECT OBJECT_VALUE FROM purchaseorder
WHERE
XMLExists('/PurchaseOrder[SpecialInstructions="Expedite"]'
PASSING OBJECT_VALUE);

-- mit Ergebnis als Zeichenkette
SELECT XMLSerialize(CONTENT OBJECT_VALUE) FROM purchaseorder
WHERE
XMLExists('/PurchaseOrder[SpecialInstructions="Expedite"]'
PASSING OBJECT_VALUE);
```

## XMLCast() zur Konvertierung

- Damit können via XQuery gelieferte skalare Abfragewerte in den entsprechenden Oracle-Typ konvertiert werden

```
SELECT XMLCast(XMLQuery('/PurchaseOrder/Reference'
                 PASSING OBJECT_VALUE
                 RETURNING CONTENT)
          AS VARCHAR2(100)) "REFERENCE"
FROM purchaseorder
WHERE
  XMLExists('/PurchaseOrder[SpecialInstructions="Expedite"]'
             PASSING OBJECT_VALUE);
```

## Generierung mit XMLQuery

- Selbstverständlich kann XQuery auch zur Generierung von XML-Daten benutzt werden
- Überschneidungen mit den sionstigen Möglichkeiten zur Generierung

```
SELECT XMLQuery(
 'for $j in 1
  return (
  <EMPLOYEES> {
   for $i in ora:view("HR", "employees")/ROW
   where $i/EMPLOYEE_ID <= 102
   return (<EMPLOYEE>
                <EMPNO>{xs:string($i/EMPLOYEE_ID)}</EMPNO>
                 <ENAME>{xs:string($i/LAST_NAME)}</ENAME>
                 <SAL>{xs:integer($i/SALARY)}</SAL>
           </EMPLOYEE>)} </EMPLOYEES>)'
  RETURNING CONTENT)  FROM DUAL;
```

# Transformation unter Nutzung von XMLTable und XQuery

```
SELECT ttab.COLUMN_VALUE AS OrderTotal FROM
   purchaseorder,
 XMLTable(
 'for $i in /PurchaseOrder
  where $i/User = "EABEL"
   return  <OrderTotal> {$i/Reference}
    <Total>
     {fn:sum(for $j in $i/LineItems/LineItem/Part
     return ($j/@Quantity*$j/@UnitPrice))}
    </Total>
    </OrderTotal>'
    PASSING OBJECT_VALUE) ttab;
```

## Abfrage von Repository-Dokumenten

- können sowohl via XQuery-Funktion collection() als auch doc() abgefragt werden

```
SELECT XMLQuery(
   'for $i in
    fn:doc("/public/employees/employees.xml")//EMPLOYEE
 order by $i/@ENAME return $i'
 RETURNING CONTENT)as xml FROM DUAL;

SELECT XMLQuery(
  'for $i in fn:collection("/public/employees")
   return $i//DEPARTMENT/DNAME'
 RETURNING CONTENT)as XML FROM DUAL;
```

## Benutzung von Namensräumen mit XQuery

```
-- explizite Verwendung mit XMLNAMESPACES
SELECT * FROM XMLTable(
XMLNAMESPACES('http://emp.com' AS "e"),
    'for $i in doc("/public/empsns.xml")
    return $i/e:emps/e:emp'
    COLUMNS name VARCHAR2(6) PATH '@ename',
    id NUMBER PATH '@empno');

-- Nutzung des XQuery-Prologs
SELECT * FROM XMLTable(
'declare default element namespace "http://emp.com";
    (: :)
   for $i in doc("/public/empsns.xml")
   return $i/emps/emp'
   COLUMNS name VARCHAR2(6) PATH '@ename',
     id NUMBER PATH '@empno' );
```