XQuery-Anfragen in Oracle

Beispieltabelle CompanyXML (aus Übung 9)

```
DEPTNO   MITARBEITERLISTE
-------------------------------------------------------------------------------
10       <Mitarbeiterliste abteilung="10">
             <Mitarbeiter id="7839">
                 <Name>King</Name>
                 <Beruf>President</Beruf>
                 <Einstelldatum>1981-11-17</Einstelldatum>
                 <Payment>
                     <Gehalt>5202</Gehalt>
                 </Payment>
             </Mitarbeiter>
             ...
         </Mitarbeiterliste>
20       <Mitarbeiterliste abteilung="20">
             <Mitarbeiter id="7566">
                 <Name>Jones</Name>
                 <Beruf>Manager</Beruf>
                 <Manager>7839</Manager>
                 <Einstelldatum>1981-04-02</Einstelldatum>
                 <Payment>
                     <Gehalt>2975</Gehalt>
                     <Kommission>300</Kommission>
                 </Payment>
             </Mitarbeiter>
             ...
         </Mitarbeiterliste>
...
```

XMLQUERY

- Extraktion von XML aus relationalen Daten
- Extraktion von XML aus anderen XML-Daten
- Rückgabe von XMLType-Instanzen

Syntax:

```
XMLQUERY (xquery-expression
          [PASSING xml-instance]
          RETURNING CONTENT)
```

xquery-expression:	XPath-Ausdruck oder vollständige FLWOR-Anweisung
PASSING:	Kontext, der die Rohdaten bereitstellt, auf die die XQuery angewendet werden soll
RETURNING CONTENT:	verpflichtende Klausel für Rückgabe der XML-Objekte

Beispiel: Alle Daten der Mitarbeiter

```
SELECT XMLQUERY('/Mitarbeiterliste/Mitarbeiter'
                PASSING mitarbeiterliste
                RETURNING CONTENT) Angestellte
FROM companyxml;
```

XQuery Funktion ora:view

- Anbindung von Tabellen an XQuery
- erzeugt eine XML-Struktur von Tabellen (SQL/XML konform)

Syntax:

```
ora:view ([db-schema STRING,] db-table STRING)
RETURNS document-node(element())
```

`db-schema` – optional, Name des Datenbank-Schemas
`db-table` – Datenbanktabelle oder -view (im Datenbank-Schema)

Beispiel: Alle Daten der Mitarbeiter

```
SELECT XMLQuery('
    <result>{
        for $e in ora:view("companyXML")//Mitarbeiter
        return $e
    }</result>
'RETURNING CONTENT ) AS Angestellte
FROM DUAL;
```

FLWOR-Ausdrücke I

- Basis für Anfragen an XML-Datenbanken
- analog zu SELECT-FROM-WHERE in SQL
- Abkürzung für for-let-where-order by-return

Beispiel: Die Namen aller Angestellten, die mehr als 2500 verdienen

```
SELECT XMLQuery('
    <result>{
      for $m in ora:view("companyXML")//Mitarbeiter
      let $g:=$m/Payment/Gehalt
      where $g>=2500
      order by $g descending
      return($m/Name)
    }</result>
'RETURNING CONTENT ) AS Angestellte
FROM DUAL;
```

FLWOR-Ausdrücke II

for-Klausel
- iteriert über alle Elemente in der angegebenen Sequenz
- bindet die Variable zu jedem einzelnen Element
- Reihenfolge der Elemente bleibt erhalten

Mehrere for-Klauseln
- erzeugen kartesisches Produkt
- Reihenfolge der Elemente im Ergebnis entsprechend den gebundenen Sequenzen von links nach rechts
- Verwendung für Joins

let-Klausel
- bindet die Variable zum Ergebnis des gesamten Ausdrucks
- iteriert nicht über die einzelnen Elemente der Sequenz

where-Klausel
- filtert von `for` bzw. `let` erzeugte Variablenbindungen
- Bedingung wird für jedes Tupel einmal überprüft

FLWOR-Ausdrücke III

order by-Klausel

- sortiert, bevor return ausgeführt wird
- Sortierung nach einem oder mehreren Kriterien möglich
- aufsteigend oder absteigend mittels `ascending` oder `descending`
- bei gleichen Werten kann die ursprüngliche Elementordnung mittels `stable` erhalten bleiben
- leere Sequenzen: Reihenfolge ist nicht definiert (analog zu NULL-Werten in SQL) Implementierungsunabhängigkeit kann mit `empty greatest` oder `empty least` garantiert werden

return-Klausel

- der return-Ausdruck wird für jedes Tupel aus dem Tupel-Stream einmal ausgewertet
- Ergebnisse werden aneinandergehängt
- ohne `order by` wird die Reihenfolge von den `for`- und `let`-Klauseln bestimmt
- jeder beliebige XQuery-Ausdruck kann im `return`-Ausdruck vorkommen
- meist werden Elementkonstruktoren verwendet

Trick: Zeilenumbruch für die Ausgabe

Eine bessere Lesbarkeit der XML-Ausgaben kann in einigen Fällen durch einen Zeilenumbruch `"
"` in der `return`-Klausel erreicht werden.
Erforderlich ist der Zeilenumbruch für das Anfrage-Ergebnis aber nicht.

Beispiel: Die Namen aller Angestellten, die mehr als 2500 verdienen

```
SELECT XMLQuery('
    <result>{
      for $m in ora:view("companyXML")//Mitarbeiter
      let $g:=$m/Payment/Gehalt
      where $g>=2500
      order by $g descending
      return($m/Name,"&#xA;")
    }</result>
'RETURNING CONTENT ) AS Angestellte
FROM DUAL;
```

Konstruktoren I

Direkte Element-Konstruktoren
- erzeugen Element-Knoten
- basieren auf Standard XML-Notation

Beispiel: Die Namen der Angestellten

```
SELECT XMLQuery('
    <result>{
      for $m in ora:view("companyXML")//Mitarbeiter
      return($m/Name)
    }</result>
'RETURNING CONTENT ) AS Angestellte
FROM DUAL;
```

Ausgabe:

```
ANGESTELLTE
----------------------------------------
<result>
     <Name>King</Name>
     <Name>Scott</Name>
     ...
     <Name>Martin</Name>
</result>
```

Konstruktoren II

Berechnete Konstruktoren

- beginnen mit dem Schlüsselwort, welches die Art des Knotens definiert: `element`, `attribute`, `document`, `text`, `comment` und `processing instruction`
- bei Typen, welche einen Namen haben, folgt auf das Schlüsselwort der Name des Knotens
- in geschwungenen Klammern folgt der Inhalt des Knotens

Beispiel: Die Namen der Angestellten mit neu definiertem Element-Knoten `Nachname`

```
SELECT XMLQuery('
    <result>{
      for $m in ora:view("companyXML")//Mitarbeiter
      return element Nachname {$m/Name/text()}
    }</result>
'RETURNING CONTENT ) AS Angestellte
FROM DUAL;
```

Ausgabe:

```
ANGESTELLTE
----------------------------------------
<result>
     <Nachname>King</Nachname>
     <Nachname>Scott</Nachname>
     ...
     <Nachname>Martin</Nachname>
</result>
```

Quantoren

- Existenz- und All-Quantoren
- bestehen aus dem Schlüsselwort `every` oder `some`, gefolgt von einer oder mehreren `in`-Klauseln, dem Schlüsselwort `satisfies` und dem zu überprüfenden Ausdruck

Beispiel: Angestellte mit dem Beruf Manager

```
SELECT XMLQuery('
   <result>{
            for $m in ora:view("companyXML")//Mitarbeiter
            where some $b in $m satisfies $b/Beruf="Manager"
            return $m/Name
   }</result>
'RETURNING CONTENT ) AS Angestellte
FROM DUAL;
```

Beispieltabelle SalegradesXML

```
GEHALTSSTUFEN
--------------------------------------------------------------------------------
<Salgrade>
     <Stufe>1</Stufe>
     <Min>700</Min>
     <Max>1200</Max>
</Salgrade>
<Salgrade>
     <Stufe>2</Stufe>
     <Min>1201</Min>
     <Max>1400</Max>
</Salgrade>
<Salgrade>
     <Stufe>3</Stufe>
     <Min>1401</Min>
     <Max>2000</Max>
</Salgrade>
<Salgrade>
     <Stufe>4</Stufe>
     <Min>2001</Min>
     <Max>3000</Max>
</Salgrade>
<Salgrade>
     <Stufe>5</Stufe>
     <Min>3001</Min>
     <Max>9999</Max>
</Salgrade>
```

Inner Join I

- Kombination mehrerer Quellen in einem Ergebnis

Beispiel: Name und Gehaltsstufe jedes Angestellten

```
SELECT XMLQuery('
   <result>{
        for $m in ora:view("companyXML")//Mitarbeiter
        for $s in ora:view("salgradesXML")//Salgrade
        where    fn:abs($m/Payment/Gehalt)<=$s/Max and
                 fn:abs($m/Payment/Gehalt)>=$s/Min
        return ( <Angestellter>
                 <Nachname>    { $m/Name/text()       }</Nachname>
                 <Stufe>       { $s/Stufe/string()    }</Stufe>
                </Angestellter>
               )
   }</result>
'RETURNING CONTENT ) AS Gehaltsstufen
FROM DUAL;
```

Hinweis: Das Gehalt muss bei dieser Anfrage in der where-Klausel mit z.B. der Funktion fn:abs() in einen numerischen Wert umgewandelt werden. Andernfalls erfolgt ein ZK-Vergleich und Gehälter wie 900 landen wegen der 9 an erster Stelle in Stufe 5.

Inner Join II

Ausgabe:

```
GEHALTSSTUFEN
--------------------------------------------------------
<result>
<Angestellter>
     <Nachname>Clark</Nachname>
     <Stufe>4</Stufe>
</Angestellter>
<Angestellter>
     <Nachname>King</Nachname>
     <Stufe>5</Stufe>
</Angestellter>
<Angestellter>
     <Nachname>Adams</Nachname>
     <Stufe>1</Stufe></Angestellter>
<Angestellter>
...
</result>
```

Gruppierung

- Gruppierung von Daten (analog zu GROUP BY in SQL)
- Aggregatfunktionen: `fn:count(), fn:avg(), fn:max(), fn:min(), fn:sum()`
- fn:distinct-values eliminiert Duplikate
- benutzerdefinierte Funktionen möglich

Beispiel: Gehaltssummen pro Abteilung

```
SELECT XMLQuery('
    <result>{
      for $a in ora:view("companyXML")//Mitarbeiterliste
      let $summe := fn:sum( for $m in ora:view("companyXML")//Mitarbeiter
                            where $a/@abteilung=$m/../@abteilung
                            return fn:abs($m/Payment/Gehalt)
                          )
      return (<Abteilung>
                 <AbtNr>{ $a/@abteilung  }</AbtNr>
                 <Summe>{ $summe         }</Summe>
              </Abteilung>)
    }</result>
'RETURNING CONTENT ) AS Gehaltssummen
FROM DUAL;
```

HTML-Ausgaben I

Beispiel: Tabellarische Ausgabe von Id und Name der Angestellten

```
SELECT XMLQuery('
  let $m := ora:view("companyXML")//Mitarbeiter
  return    <html>
            <head>
            <title>Angestellte</title>
            </head>
            <body>
            <table>
            <tr>
              <th>ID</th>
              <th>Name</th>
            </tr>
            {
              for $x in $m
              order by $x/Name
              return (<tr>
                        <td>{ data($x/@id)    }</td>
                        <td>{ data($x/Name)   }</td>
                      </tr>,"&#xA;")
            }
          </table>
          </body>
          </html>

'RETURNING CONTENT ) AS Angestellte
FROM DUAL;
```

HTML-Ausgaben II

Ausgabe:

```
ANGESTELLTE
--------------------------------------------------------------------------------
<html><head><title>Angestellte</title></head>
<body>
<table>
<tr><th>ID</th><th>Name</th></tr>
<tr><td>7876</td><td>Adams</td></tr>
<tr><td>7499</td><td>Allen</td></tr>
<tr><td>7698</td><td>Blake</td></tr>
<tr><td>7782</td><td>Clark</td></tr>
<tr><td>7902</td><td>Ford</td></tr>
<tr><td>7900</td><td>James</td></tr>
<tr><td>7566</td><td>Jones</td></tr>
<tr><td>7839</td><td>King</td></tr>
<tr><td>7654</td><td>Martin</td></tr>
<tr><td>7934</td><td>Meier</td></tr>
<tr><td>7788</td><td>Scott</td></tr>
<tr><td>7369</td><td>Smith</td></tr>
<tr><td>7844</td><td>Turner</td></tr>
<tr><td>7521</td><td>Ward</td></tr>
</table>
</body>
</html>
```

Methoden zur Extraktion des Datenwertes aus Knoten

data() - gibt typisierten Datenwert zurück
(inklusive der Werte aus Kindknoten)
string() - gibt Zeichenkette zurück
(inklusive der ZK aus Kindknoten)
text() - gibt Zeichenkette des aktuellen Knotens zurück

```
SELECT XMLQuery('
<result>{
     for $a in ora:view("companyXML")//Mitarbeiter
     return (
               <Angestellter>
               {
                    <ID>   {data($a/@id)} </ID>,
                    <EDat> {data($a/Einstelldatum)} </EDat>,
                    $a/Name,
                    $a/Beruf,
                    $a/Payment/Gehalt
               }
               </Angestellter>, "&#xA;"
                 )
}</result>
'RETURNING CONTENT ) AS Angestellte
FROM DUAL;
```