DBA-Zertifizierungs-AG

(Sommer 2008)

Teil 5: Kurzeinführung in XQuery

- XML-Unterstützung in DB2
- XPath
- XQuery

Literatur (1)

- C. M. Saracco: DB2 9 Fundamentals exam 730 prep, Part 7: Introducing XQuery.

  [http://www.ibm.com/developerworks/offers/lp/db2cert/db2-cert730.html?S_TACT=105AGX19&S_CMP=db2certlp]
- F. Özcan, D. Chamberlin, K. Kulkarni, J.-E. Michels: Integration of SQL and XQuery in IBM DB2.

  IBM Systems Journal, Volume 45 , Issue 2 (January 2006), Pages: 245 - 270. [http://www.research.ibm.com/journal/sj/452/ozcan.pdf]
- K. Beyer et. al.: DB2 goes hybrid: Integrating native XML and XQuery with relational data and SQL.

  IBM Systems Journal, Volume 45 , Issue 2 (January 2006), Pages: 271-298. [http://www.research.ibm.com/journal/sj/452/beyer.pdf]

Literatur (2)

- DB2 (LUW) Version 9 Information Center

  [http://publib.boulder.ibm.com/infocenter/db2luw/v9//index.jsp]
- Erhard Rahm, Gottfried Vossen (Hrsg.): Web & Datenbanken.

  dpunkt.verlag, 2003, ISBN 3-89864-189-9, 488 Seiten.
- M. Klettke, H. Meyer: XML & Datenbanken.

  dpunkt.Verlag, 2003, ISBN 3-89864-148-1, 428 Seiten.
- Georg Lausen: Datenbanken. Grundlagen und XML-Technologien.

  Spektrum Akademischer Verlag, 2005, ISBN 3827414881, 281 Seiten.

Literatur (3)

- Wolfgang Lehner, Harald Schöning: XQuery.

  dpunkt Verlag, 2004, ISBN 3-89864-266-6, 290 Seiten.
- Howard Katz (Ed.): XQuery from the Experts.

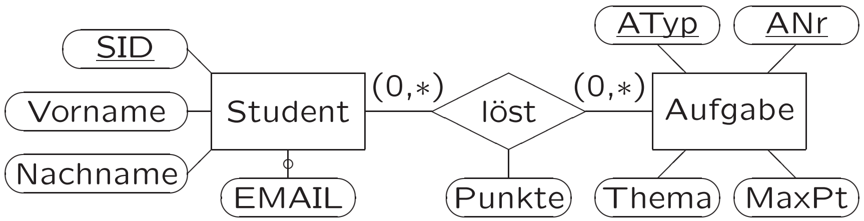
  A Guide to the W3C XML Query Language.
  Addison-Wesley, 2003, ISBN 0321180607, 512 Seiten.
- Jim Melton, Stepgen Buxton: Querying XML. XQuery, XPath, and SQL/XML in Context.

  Morgan Kaufmann, 2006, ISBN 1-55860-711-0, 815 Seiten.
- Elliotte Rusty Harold, W. Scott Means: XML in a Nutshell, A Desktop Quick Ref., 3rd Ed.

  O'Reilly, Okt. 2004, ISBN 0-596-00764-7, 689 Seiten, 37 Euro.

Inhalt

Beispiel-Datenbank: ER



- Studenten lösen Aufgaben und bekommen dafür Punkte (viele-zu-viele Beziehung).
- Aufgaben sind identifiziert über Aufgabentyp (z.B. ’H’: Hausaufgaben, ’K’: Klausur) und Aufgabennummer. Studenten über die Studenten-ID (SID).

Beispiel-Datenbank: Relational

STUDENTEN

SID	VORNAME	NACHNAME	EMAIL
101	Lisa	Weiss	···
102	Michael	Grau	NULL
103	Daniel	Sommer	···
104	Iris	Winter	···

BEWERTUNGEN

SID	ATYP	ANR	PUNKTE
101	H	1	10
101	H	2	8
101	K	1	12
102	H	1	9
102	H	2	9
102	K	1	10
103	H	1	5
103	K	1	7

AUFGABEN

ATYP	ANR	THEMA	MAXPT
H	1	ER	10
H	2	SQL	10
K	1	SQL	14

Beispiel-Datenbank: XML (1)

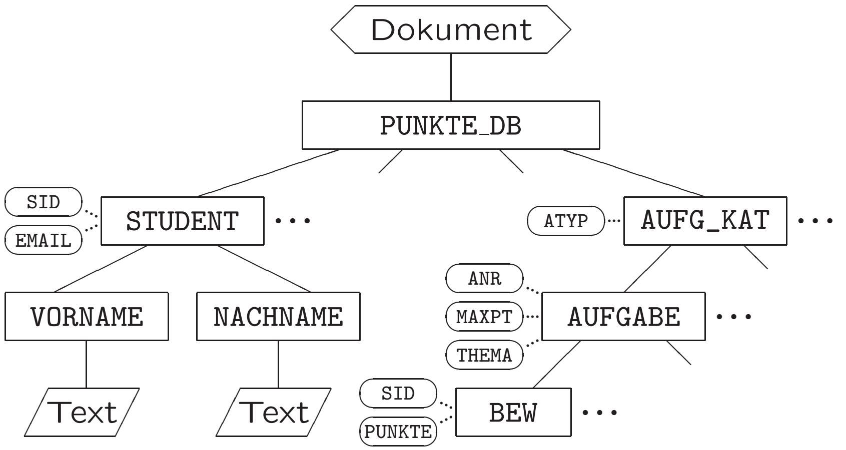
```
<?xml version=’1.0’ encoding=’ISO-8859-1’?>
<PUNKTE_DB>
  <STUDENT SID=’101’ EMAIL=’weiss@acm.org’>
    <VORNAME>Lisa</VORNAME>
    <NACHNAME>Weiss</VORNAME>
  </STUDENT>
  <STUDENT SID=’102’ EMAIL=’m.grau@gmx.de’>
    <VORNAME>Michael</VORNAME>
    <NACHNAME>Grau</VORNAME>
  </STUDENT>
  ...
```

(Fortsetzung auf nächster Folie)

Beispiel-Datenbank: XML (2)

```
    <AUFG_KAT ATYP='H'>
      <AUFGABE ANR='1' MAXPT='10' THEMA='ER'>
        <BEW SID='101' PUNKTE='10'/>
        <BEW SID='102' PUNKTE= '9'/>
        <BEW SID='103' PUNKTE= '5'/>
      </AUFGABE>
      <AUFGABE ANR='2' MAXPT='10' THEMA='SQL>
        ...
    </AUFG_KAT>
    <AUFG_KAT ATYP='K'>
      <AUFGABE ANR='1' MAXPT='14' THEMA='SQL'>
        ...
  </PUNKTE_DB>
```


Interne Darstellung: XDM (1)



Interne Darstellung: XDM (2)

- Knotentypen:
  - ◇ Element-Knoten,
  - ◇ Attribut-Knoten,
  - ◇ Text-Knoten, Dokument-Knoten,
  - ◇ Kommentar-Knoten,
  - ◇ Processing Instruction Knoten
  - ◇ ggf. Namespace Knoten (veraltet).
- Element- und Dokument-Knoten (Eltern-Knoten) haben eine geordnete Liste von Kindknoten (Element, Text, Kommentar, PI Knoten).

Interne Darstellung: XDM (3)

- Attribut-Knoten sind keine Kind-Knoten.

  Einem Elementknoten ist eine Menge von Attribt-Knoten zugeordnet (Reihenfolge nicht definiert). Der Element-Knoten ist für den Attribut-Knoten ein Elternknoten.
- In der internen Darstellung fehlen unwesentliche Details der externen Repräsentation, z.B.
  - ◇ Welche Art von Anführungszeichen für Attribute verwendet wurde (’ oder ").
  - ◇ Leerplatz innerhalb von Tags.
- Dokument-Ordnung der Knoten entsprechend dem Beginn der textuellen Repräsentation bei Ausgabe.

XQuery Software (1)

- IPSI XQ

  In Java, nichtkommerzielle Nutzung ist kostenlos.
  [http://www.ipsi.fraunhofer.de/oasys/projects/ipsi-xq/]

- AltovaXML

  Der Validator und XSLT/XQuery Auswerter, den auch XML Spy benutzt, ist kostenlos. [http://www.altova.com/altovaxml.html]

- Galax

  Open source, von einigen Herausgebern der XQuery Specification.
  [http://www.galaxquery.org/]

- Saxon

  Von Michael Kay, herausgeber der XSLT 2.0 Spezifikation.
  Die Basisversion ist Open Source. [http://saxon.sourceforge.net/]

XQuery Software (2)

- X-HIVE

  Kommerzielles XML-DBMS, Online Demo (beliebige Anfragen an vorgegebenen Datenbestand). [http://support.x-hive.com/xquery/].
  [http://support.x-hive.com/xquery/basicservlet?demo=demo0&
  xquery=xquery88&todo=showframes]

- Qizx/open

  In Java. Limited version is free. [http://www.xmlmind.com/qizx/]

- eXist (open source native XML database)

  Native XML DBMS, Open Source [http://exist.sourceforge.net/]
  Online demo: [http://demo.exist-db.org/sandbox/sandbox.xql]

XQuery-Anfragen: Aufbau

- Eine XQuery-Anfrage (“main module”) besteht aus:
  - ◇ einer optionalen Versions-Angabe,
  - ◇ einem optionalen Prolog,

    Enthält z.B. Namespace- und Funktions-Deklarationen.
  - ◇ einem Rumpf (Hauptteil der Anfrage).
- Der Rumpf ist ein Ausdruck (bzw. Folge von durch Komma getrennten Ausdrücken), z.B.
  - ◇ ein FLWOR-Ausdruck,
  - ◇ ein XPath-Ausdruck.

    Dazu gehören neben den reinen Pfad-Ausdrücken auch viele weitere Operatoren und insbesondere auch Konstruktoren.

Einfache Ausdrücke

Beispiel 1:

- Aus Konstanten und Funktionen aufgebaute Ausdrücke sind XQuery-Anfragen:

```
1+1
```

- Ergebnis:

  2

- Entsprechung in SQL (Oracle):

```
SELECT 1+1 FROM DUAL
```

- Entsprechung in SQL (DB2):

```
VALUES (1+1)
```

Konstruktoren (1)

Beispiel 2:

- Man kann Elemente in der üblichen XML-Syntax eingeben ("direct constructor").

  < beginnt Modus, bei dem Eingabe unausgewertet übernommen wird.

- Darin markiert {...} einen auszuwertenden Teil.

  Braucht man {, so ist es zu verdoppeln: {{. Entsprechend auch }.

- Beispiel:

```
<x>1+1={1+1}</x>
```

- Ergebnis:

```
<x>1+1=2</x>
```


Konstruktoren (2)

Beispiel 3:

- Daneben gibt es auch einen "computed constructor" für die verschiedenen Knotentypen.

  Nach dem Schlüsselwort für den Knotentyp kann man entweder direkt z.B. den Element-Namen angeben, oder einen Ausdruck, der den Namen berechnet, in {...}.

- Beispiel:

```
element {concat('x', 'y')}
        {attribute a {1+1},
         <z> {element leer {}} </z>}
```

- Ergebnis:

```
<xy a='2'><z><leer/></z></xy>
```


Zugriff auf Dokumente

Beispiel 4:

- Die Funktion `doc` liefert den Dokumentknoten (die Wurzel der internen Baumdarstellung) zu geg. URI.
- Beispiel:

```
doc('http://www.informatik.uni-halle.de/~brass/pdb.xml')
```

- Ergebnis:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
...
```

- Dokument wurde geparsed und wieder ungeparsed, dabei ändert sich Leerplatz, Art der Anführungszeichen, etc.

Inhalt

Pfadausdrücke (1)

- XPath ("XML Path Language") ist ein Standard für Ausdrücke, die hauptsächlich benutzt werden, um Teile von XML-Dokumenten zu selektieren.
- Allgemein ist das Ergebnis eines Ausdrucks eine endliche Folge (Sequenz) von "items", d.h.
  - ◇ Knoten der internen Baumdarstellung oder
  - ◇ atomaren Werten (Zahlen, Zeichenketten, etc.).
- XPath ist eine Teilmenge von XQuery.
- XPath wird aber z.B. auch in XSLT verwendet.

Pfadausdrücke (2)

- Pfadausdrücke werden bzgl. eines Kontexts ausgewertet, der u.a. einen Kontextknoten enthält.
- Ein Pfadausdruck besteht aus einem oder mehreren Schritten, die durch "/" getrennt werden.
- Ein Schritt besteht in der ausführlichen Form aus
  - ◇ Achse, z.B. `child::`, `descendant::`, `attribute::` (Navigation vom Kontextknoten aus)
  - ◇ Knotentest: wählt einige der durch die Achse bestimmten Knoten über Typ/ggf. Namen aus.
  - ◇ Prädikat in `[...]`: Bedingung oder Position.

Pfadausdrücke (3)

- Es gibt eine abgekürzte Syntax, mit der XPath-Pfadausdrücke ähnlich zu den Pfaden im UNIX-Dateibaum aussehen. Z.B. steht

  ```
  //AUFGABE/BEW/@PUNKTE
  ```

  für

  ```
  root(self::node()) treat as document-node()/
      /descendant-or-self::node()/child::AUFGABE/
      child::BEW/attribute::PUNKTE
  ```

- Neben den obigen Schritten ("axis step") können auch einfache oder geklammerte XPath-Ausdrücke (ggf. mit Prädikaten) Schritt in einem Pfad sein.

Pfadausdrücke (4)

Beispiel 5:

- Drucke die Nachnamen aller Studierenden:

  ```
  doc('pdb.xml')/PUNKTE_DB/STUDENT/NACHNAME
  ```

- Ergebnis:

  ```
  <NACHNAME>Weiss</NACHNAME>
  <NACHNAME>Grau</NACHNAME>
  <NACHNAME>Sommer</NACHNAME>
  <NACHNAME>Winter</NACHNAME>
  ```

- Entsprechung in SQL:

  ```
  SELECT NACHNAME FROM STUDENT
  ```

Pfadausdrücke (5)

Beispiel 6:

- Gesucht sind Bewertungen für Hausaufgabe 1:
```
doc('pdb.xml')//AUFG_KAT[@ATYP='H']
                /AUFGABE[@ANR=1]/BEW
```
- Ergebnis:
```
<BEW SID='101' PUNKTE='10'/>
<BEW SID='102' PUNKTE= '9'/>
<BEW SID='103' PUNKTE= '5'/>
```
- Entsprechung in SQL:
```
SELECT SID, PUNKTE FROM BEWERTUNGEN
WHERE ATYP='H' AND ANR=1
```


Inhalt

FLWOR-Ausdrücke (1)

- Ein wichtiges XQuery-Konstrukt sind “FLWOR”-Ausdrücke (man spricht es “Flower”, obwohl das “`order by`” falsch steht):
```
for $⟨var⟩ in ⟨Ausdruck⟩, ...
let $⟨var⟩ := ⟨Ausdruck⟩, ...
where ⟨Bedingung⟩
order by ⟨Sortierung⟩
return ⟨Ausdruck⟩
```
- Man kann `for` und `let` mehrfach in beliebiger Reihenfolge verwenden. Eins von beiden ist nötig.
- `where` und `order by` sind optional.

FLWOR-Ausdrücke (2)

Auswertung von FLWOR-Ausdrücken:

- Die Ausdrücke in den `for` und `let` Klauseln liefern (wie alles in XQuery) eine Sequenz.
- Bei `for` wird jeweils ein Element der Sequenz an die Variable gebunden (in einer Schleife).
- Bei `let` wird die Sequenz als Ganzes an die Variable gebunden.
- Insgesamt produzieren `for` und `let` also eine Folge von Variablenbelegungen.

FLWOR-Ausdrücke (3)

Auswertung von FLWOR-Ausdrücken, Forts.:

- Aus dieser Folge werden die Variablenbelegungen gestrichen, die die `where`-Bedingung nicht erfüllen.
- Der Rest wird durch `order by` sortiert.

  Die Variablenbelegungen haben auch ohne `order by` eine definierte Reihenfolge, da der Ausdruck unter `for` eine Sequenz liefert, und nicht eine Menge von Werten. Häufig ist dies die Dokumentordnung, also die Reihenfolge des Beginns im Eingabedokument.
- Am Ende wird für jede Variablenbelegung der Wert des `return`-Ausdrucks geliefert.
- Dies ergibt (natürlich) wieder eine Sequenz.

FLWOR-Ausdrücke (4)

- Vergleich mit SQL:
  - ◇ `for` entspricht `FROM`

    (Schleife über möglichen Variablenbindungen)

    In XQuery kann man statt dem Komma zwischen mehreren Variablendeklarationen auch mehrere `for`-Klauseln verwenden (ist äquivalent). In SQL gibt es nur eine `FROM`-Klausel.
  - ◇ `return` entspricht `SELECT`

    Es ist eigentlich unlogisch, daß bei SQL `SELECT` ganz am Anfang steht, obwohl es als letztes ausgewertet wird. Dies ist in XQuery wohl besser gelöst.
  - ◇ Variablen werden in XQuery mit "$" markiert.
  - ◇ XQuery hat keine reservierten Worte.

FLWOR-Ausdrücke (5)

Beispiel 7:

- Namen von Studierenden als Liste in HTML:

```
<ul>
    {for $n in doc("E:\pdb.xml")//NACHNAME
     return <li>{string($n)}</li>}
</ul>
```

- Ausgabe:

```
<ul>
    <li>Weiss</li>
    <li>Grau</li>
    ...
</ul>
```


FLWOR-Ausdrücke (6)

Beispiel 8:

- Wer hat für Hausaufgabe 1 die volle Punktzahl?

```
let $d  := doc("pdb.xml")
for $h1 in $d//AUFG_KAT[ATYP='H']/
            AUFGABE[ANR=1]
for $b  in $h1//BEW
for $s  in $d//STUDENT
where $b/@PUNKTE = $h1/@MAXPT
    and $s/@SID = $b/@SID
return <li>{string($s/NACHNAME)}</li>
```

- Ausgabe:

```
<li>Weiss</li>
```

FLWOR-Ausdrücke (7)

Beispiel 9:

- Alternative Lösung zu: Wer hat für Hausaufgabe 1 die volle Punktzahl?

```
let $d := doc("pdb.xml")
for $b in $d//AUFG_KAT[ATYP='H']/
        AUFGABE[ANR=1]/BEW[@PUNKTE=../@MAXPT]
return $d//STUDENT[@SID=$b/@SID]
```

- Ausgabe:

```
<STUDENT SID="101" EMAIL="weiss@acm.org">
  <VORNAME>Lisa</VORNAME>
  <NACHNAME>Weiss</VORNAME>
</STUDENT>
```


FLWOR-Ausdrücke (8)

Beispiel 9, Forts.:

- Entsprechung in SQL:

```
SELECT S.NACHNAME
FROM   STUDENTEN S, AUFGABEN A, BEWERTUNGEN B
WHERE  A.ATYP = 'H' AND A.ANR = 1
AND    S.SID = B.SID
```

- Dies sind 114 Zeichen.
- Die XQuery Lösung aus Beispiel 9 hat 136 Zeichen, ohne die erste Zeile mit `doc(...)` noch 111 Zeichen.

  Die Lösung aus Beispiel 8 hat 219 Zeichen, ohne die erste Zeile und ohne die Umwandlung in `li`-Elemente noch 175 Zeichen.

FLWOR-Ausdrücke (9)

Beispiel 10:

- Drucke Ergebnisse für Hausaufgabe 1 nach Punkten sortiert (als HTML Tabelle)

```
<table> {
  let $d := doc("pdb.xml")
  for $b in $d//AUFG_KAT[ATYP='H']/
      AUFGABE[ANR=1]/BEW
  for $s in $d//STUDENT[@SID=$b/@SID]
  order by number($b/@PUNKTE) descending
  (: ohne number und Schema: alphabetisch :)
  return <tr><td>{$s/NACHNAME/text()}</td>
             <td>{string($b/@PUNKTE)}</td></tr>
} </table>
```


FLWOR-Ausdrücke (10)

```
<table>
    <tr>
        <td>Weiss</td>
        <td>10</td><
    </tr>
    <tr>
        <td>Grau</td>
        <td>9</td><
    </tr>
    <tr>
        <td>Sommer</td>
        <td>5</td><
    </tr>
</table>
```

Inhalt

Aggregationen (1)

Beispiel 11:

- Was ist der Durchschnitt der Punkte für Hausaufgabe 1?

```
let $d := doc("pdb.xml")
let $a := $d//AUFG_KAT[ATYP='H']/AUFGABE[ANR=1]
return avg($a/BEW/@PUNKTE)
```

- Ausgabe: 8
- Entsprechung in SQL:

```
SELECT AVG(PUNKTE)
FROM   BEWERTUNGEN
WHERE  ATYP = 'H' AND ANR = 1
```


Aggregationen (2)

Beispiel 12:

- Punktedurchschnitt für jede Hausaufgabe:

```
let $d := doc("pdb.xml")
for $a in $d//AUFG_KAT[ATYP='H']/AUFGABE
return <li>{concat('Aufgabe ', $a/@ANR, ': ',
                   avg($a/BEW/@PUNKTE))}</li>
```

- Ausgabe:

```
<li>Aufgabe 1: 8</li>
<li>Aufgabe 2: 8.5</li>
```

- Entsprechung in SQL:

```
SELECT   ANR, AVG(PUNKTE)
FROM     BEWERTUNGEN WHERE ATYP = 'H'
GROUP BY ANR
```


Schlussbemerkung zu XQuery

- XQuery kann alles, was SQL Anfragen können.

  XQuery erlaubt die Definition von rekursiven Funktionen (hier nicht behandelt) und kann dann echt mehr.

- Wenn man komplexere Datenstrukturen als im relationalen Modell hat, ist die Anfragesprache auch komplexer.
- XPath/XQuery enthalten manche Überraschungen (u.a. ist = nicht transitiv).
- Im Wintersemester gibt es eine Vorlesung “XML und Datenbanken”.

Inhalt

XML in DB2 (1)

- IBM ist stolz darauf, daß DB2 Version 9 eine echte hybride Datenbank ist:
  - ◇ Früher wurden XML Daten in relationalen DBen entweder als CLOB oder “shredded” in vielen Zeilen (z.B. eine pro Knoten) abgespeichert.

    DB2 bietet diese Möglichkeiten auch noch (aus Kompatibilitätsgründen mit Version 8, außerdem muß man bei manchen DB2 Editionen für das neue “pureXML” extra zahlen, während der alte XML Extender kostenlos ist).
  - ◇ DB2 Version 9 enthält spezielle XML Speicherstrukturen, zusätzlich zu den normalen relationalen Speicherstrukturen.

XML in DB2 (2)

- Es gibt einen Datentyp “`XML`”, den man wie die klassischen Datentypen für Spalten verwenden kann.
- Insofern ist das obige Beispiel untypisch:
  - ◇ Man würde nicht die ganzen Daten in XML codieren, sondern nur z.B. den Aufgabentext.

    Also ein Dokument. Dies könnte eventuell auch die Maximalpunktzahl und das Thema enthalten.
  - ◇ XML wäre auch passend, wenn die Daten sehr unregelmäßig strukturiert sind, oder sich das genaue Schema häufig ändert.
  - ◇ Oder auch bei baumartig strukturierten Daten.

XML in DB2 (3)

- Beispiel:

```
CREATE TABLE AUFGABEN (
             ATYP CHAR NOT NULL,
             ANR  NUMERIC(2) NOT NULL,
             DOC  XML,
             PRIMARY KEY (ATYP, ANR))
```

- Beim Einfügen gibt man die XML Daten z.B. als Zeichenkette an:

```
INSERT INTO AUFGABEN VALUES('H', 1,
  '<AUFG MAXPT="10"><THEMA>ER</THEMA>...</AUFG>')
```

  Verwendet man ' in den Daten, muß man das Zeichen verdoppeln.

- DB2 prüft, daß das Dokument wohlgeformt ist.

XML in DB2 (4)

- Der Datentyp XML ist nicht vergleichbar mit Zeichenketten.
- Man muß ggf. eine explizite Umwandlung mit der Funktion XMLSERIALIZE vornehmen, z.B.

```
SELECT *
FROM   AUFGABEN
WHERE  XMLSERIALIZE(DOC AS VARCHAR(1000))
           = '<AUFG/>'
```

Man kann nach dem Zieltyp bei Bedarf noch INCLUDING XMLDECARATION angeben. Man beachte, daß der Vergleich so etwas unsicher ist, weil das Ergebnis ja auch '<AUFG></AUFG>' sein könnte. Diese Aufgabe würde sich besser mit XMLEXISTS (s.u.) lösen lassen.

XML in DB2 (5)

- Die umgekehrte Abbildung (von Zeichenketten auf Werte vom Typ XML) geschieht mit der Funktion XMLPARSE.
- Bei der obigen INSERT-Anweisung wurde sie implizit aufgerufen, man kann dies aber auch explizit tun:

```
INSERT INTO AUFGABEN VALUES('H', 1,
  XMLPARSE(DOCUMENT '<AUFG>...</AUFG>'))
```

- Bei Bedarf kann man PRESERVE WHITESPACE nach dem Eingabestring hinzufügen.

  Der Default ist STRIP WHITESPACE: Alle Textknoten nur als Leerzeichen werden entfernt (falls nicht xml:space='preserve' im Dokument).

XML in DB2 (6)

- In Spalten vom Typ XML können nur vollständige XML Dokumente gespeichert werden, nicht beliebige Sequenzen.

  Also z.B. nicht einzelne Text-Knoten.
- XML-Spalten können nicht Schlüssel oder Teil eines Schlüssels sein.
- In CHECK-Constraints können solche Spalten nur mit dem Prädikat "IS VALIDATED" verwendet werden.

  Die Validierung geparster XML Dokumente ist optional und geschieht mit der Funktion XMLVALIDATE. Sie liefert eine Kopie der Eingabe, in der u.a. Default-Attribute und Typ-Angaben hinzugefügt sind.

XML in DB2 (7)

- DB2 Werkzeuge wie der Command Line Processor (db2) verstehen sowohl XQuery als auch SQL.

  Es wird empfohlen, db2 mit den Argumenten -i ("display XML data with indentation") und -d ("retrieve and display XML declarations") aufzurufen (für eine hübschere Ausgabe der Anfrage-Ergebnisse).
- Zur Unterscheidung muß man XQuery Anfragen das Schlüsselwort xquery voranstellen.
- Der Zugriff auf XML Daten geschieht z.B. mit der Funktion db2-fn:xmlcolumn in der Form:

```
db2-fn:xmlcolumn('TABELLE.SPALTE')
```

XML in DB2 (8)

- Beispiel:

```
xquery db2-fn:xmlcolumn('AUFGABEN.DOC')//THEMA
```

DB2 wandelt wie Oracle intern alle Bezeichner in Großbuchstaben um. Für SQL wird so erreicht, daß die Groß-/Kleinschreibung egal ist. In XQuery und auch in dieser Funktion ist die Groß-/Kleinschreibung aber wichtig.

- Die Funktion `db2-fn:xmlcolumn` liefert eine Sequenz bestehend aus den Wurzelknoten der Bäume, die in der Spalte gespeichert sind.
- Daraus werden hier die Unterknoten mit Elementtyp `THEMA` ausgewählt.

XML in DB2 (9)

- Beispiel (Fortsetzung):
  - ◇ Sollte ein Dokument zwei `THEMA`-Knoten haben, kommen beide in die Ausgabeliste.

    Man kann in der Ausgabeliste nicht erkennen, ob zwei `THEMA`-Knoten aus dem gleichen Dokument stammen, oder aus unterschiedlichen. Natürlich kann man mit einem FLWOR-Ausdruck eine Gruppierung einführen, indem man die Ergebnisknoten pro Eingabeknoten in ein neues Element einfügt.
  - ◇ Sollte ein Dokument keinen `THEMA`-Knoten haben, fehlt es in der Ausgabeliste.

    Es erscheint auch kein Nullwert. XQuery hat keinen Nullwert bzw. der Nullwert ist die leere Liste. Da die Ergebnislisten für die Eingabedokumente konkateniert werden, verschwindet die leere Liste. Abhilfe: Container-Element wie oben oder bedingter Ausdruck.

XML in DB2 (10)

- In DB2 können sich XQuery und SQL gegenseitig aufrufen.
- Mit der Funktion `db2-fn:sqlquery` kann man eine SQL-Anfrage, die XML Werte liefert, in XQuery verwenden. Beispiel:

```
xquery db2-fn:sqlquery(
       "SELECT DOC FROM AUFGABEN WHERE ATYP='H'")
  //THEMA
```

Wenn man die Anfrage in ' einschließt, muß man dieses Zeichen in der Anfrage verdoppeln. Diese Möglichkeit, den String-Begrenzer als Daten einzugeben, existiert in XPath/XQuery, aber nicht in XML.

XML in DB2 (11)

- Mit der SQL-Funktion `xmlquery` kann man einen XQuery-Ausdruck als Term (Wertausdruck) in SQL verwenden, z.B.

```
SELECT ANR,
       xmlquery('$d//THEMA/text()'
                passing DOC as "d")
FROM   AUFGABEN
WHERE  ATYP = 'H'
```

Die Auswertung eines XPath-Ausdrucks ist relativ zu einem Kontext, in dem u.a. die Werte von Variablen definiert sind. Beim Aufruf wird hier die Variable `d` definiert, und zwar wird ihr der Wert der Spalte `DOC` zugewiesen.

XML in DB2 (12)

- Mit dem SQL-Prädikat XMLEXISTS kann man einen XQuery-Ausdruck als Bedingung in SQL verwenden, z.B.

```
SELECT ANR
FROM   AUFGABEN
WHERE  ATYP = 'H'
AND    XMLEXISTS('$d//THEMA/[text()="ER"]'
                 passing DOC as "d")
```

- XMLEXISTS ist wahr, wenn das Ergebnis der XQuery-Anfrage nicht die leere Sequenz ist.