XQuery

XML Query Language

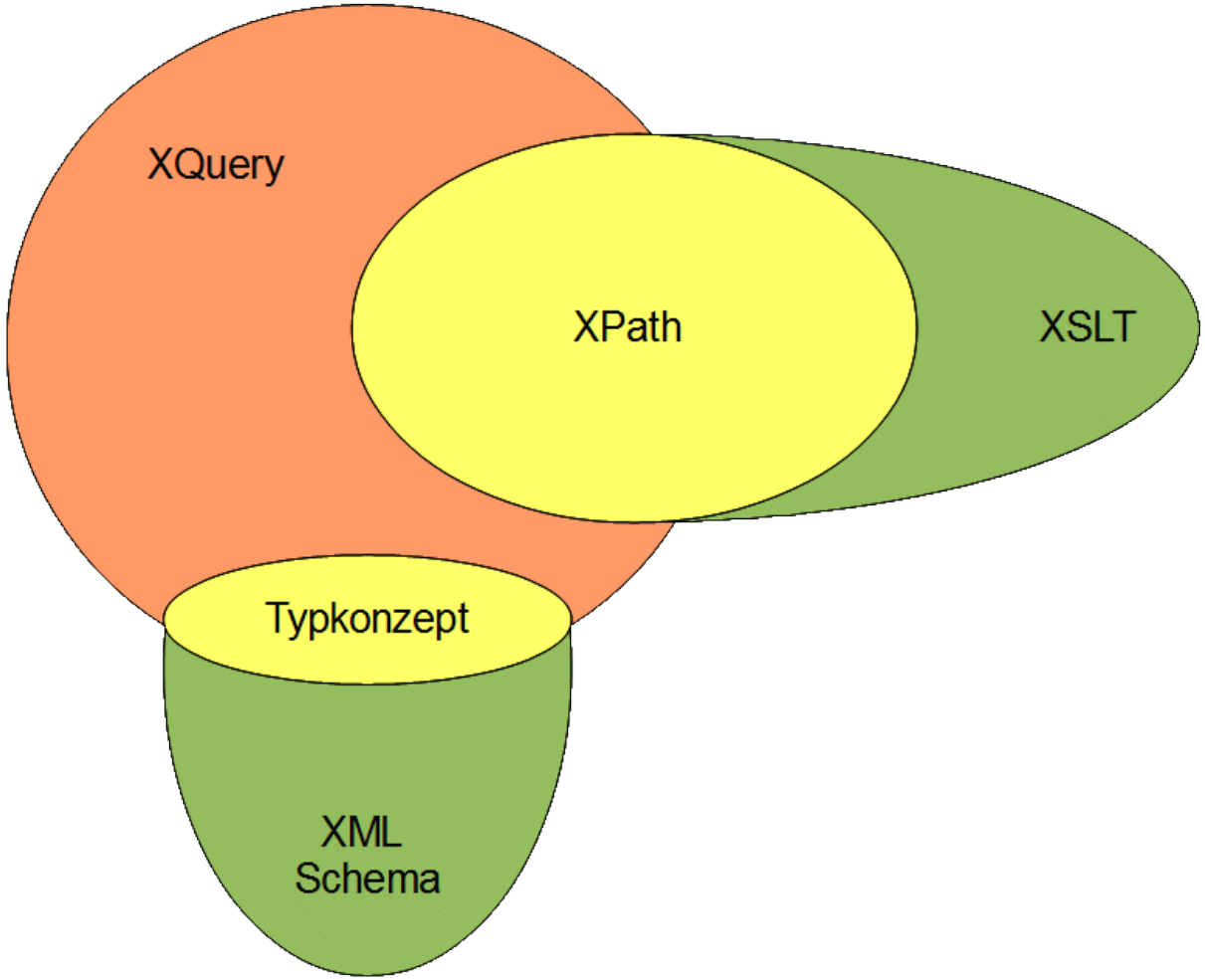
eine Abfragesprache für XML

Ulrike Henny
ulrike.henny@uni-koeln.de

XQuery – XML Query Language

- Abfragesprache für XML
- Entwicklung seit 1999, aus Quilt, XQL und XML-QL hervorgegangen
- 2007/2010 W3C-Recommendation
- Funktionen:
  - Auswahl
  - Sortierung
  - Umstrukturierung
  - Verknüpfung
  - Rückgabe
  - Aktualisierung von Daten?

XQuery in der XML-Welt



XQuery - Datenmodell

- XQuery 1.0 und XPath 2.0 Datenmodell (XDM)
- Bestandteile:
  - Knoten — *Element, Attribut, Textknoten,…*
  - Atomarer Wert — *String, Zahl,…*
  - Item — *Knoten oder atomarer Wert*
  - Sequenz — *Liste von Items*
    (geordnet, nicht hierarchisch)

XQuery - Sequenz

- Beispiele für Sequenzen:
  - ()
  - (1, 2, 3)
  - ("Chemnitz", ("Köln", "Berlin"))
  - (<ort>
    <name>Chemnitz</name>
    </ort>,
    <ort>
    <name>Köln</name>
    </ort>)
  - (<ehrenbuerger>Stefan Heym</ehrenbuerger>, "Bora Ćosić")

XQuery - Sequenz

- Wie entstehen Sequenzen?
  - direkt konstruiert:
    - (… , …)
  - als Ergebnis von Pfadausdrücken:
    - doc("edition.xml")//p → (… , … , …)
  - Als Rückgabe von Funktionen:
    - tokenize(//s[1], "\s") → ("Dies", "war", "ein", “Satz")

XQuery - Ausdrücke

- Ein Query setzt sich aus Ausdrücken zusammen:
  - XPath-Ausdrücke
  - XQuery-Ausdrücke

- Im einfachsten Fall besteht ein Query aus einem einfachen Ausdruck
  - Z.B.
    - „Hallo Welt“
    - <h1>Hallo Welt</h1>
    - //title

XQuery - Ausdrücke

- **=, !=, <, >, <=, >=**
  - Vergleichsausdrücke
  - einzelne Werte werden verglichen, aber auch Sequenzen
  - Beispiele:
    - 1 > 2 → falsch
    - (“Andrea“) = (“Andrea“, “Anna“) → wahr

- **eq, ne, lt, gt, le, ge**
  - equal, not equal, less than, greater than, less than or equal to, greater or equal to
  - nur zum Vergleich einzelner Werte
  - Beispiele:
    - 3 gt 4 → falsch
    - “abc“ lt “cde“ → wahr

XQuery - Ausdrücke

- **or, and**
  - Logische Verknüpfung von Ausdrücken, vor allem in Bedingungen
  - Beispiel:
    - //w[. = “glücklich“ or . = “unglücklich“]

- **+, -, *, div, idiv, mod**
  - Arithmetische Ausdrücke
  - idiv: gibt vom Ergebnis der Teilung nur die Ganzzahl zurück
  - mod: gibt den Rest zurück, der nach der Teilung übrig bleibt
  - Beispiele:
    - (3+4+5) *2 → 24
    - 14 idiv 4 → 3
    - 23 mod 2 → 1

XQuery - Ausdrücke

- **Pfadausdrücke**:
  - Beispiel:
    - //buch/kapitel[3]/absatz[4]

- **direkte Konstruktoren**:
  - Elemente und Attribute, die direkt notiert werden
  - XQuery-Ausdrücke darin in geschweiften Klammern
  - Beispiel:

```
<div class="maerchen">
    <p>Es war einmal ein {data(//maerchen/figuren/hauptfigur)}</p>
</div>
```

XQuery - Ausdrücke

- **if-Ausdruck**
  - Konditionaler Ausdruck
  - Beispiel:
    - if (contains(//absatz[1], "glücklich“))
      then "ja“
      else "nein“

- **For-Schleife**
  - Eine Sequenz wird durchlaufen und für jedes Item etwas getan
  - Beispiel:
    - for ($p in doc(“edition.xml“)//absatz)
      return substring($p, 1, 1)

XQuery - Ausdrücke

- **to, union (|), intersect, except**
- Sequenz-Ausdrücke, um Sequenzen zu bilden und zu kombinieren
  - *Zahl* to *Zahl*:
    - Sequenz von Zahlen
  - union:
    - Verknüpfung zweier Sequenzen
  - intersect:
    - gibt nur die Items zurück, die in BEIDEN Sequenzen vorkommen
  - exept:
    - gibt alle Items der ersten Sequenz zurück AUSSER denen, die auch in der zweiten vorkommen

XQuery - Ausdrücke

- **to, union (|), intersect, except**
  - Beispiele:
    - 1 to 10 → (1,2,3,4,5,6,7,8,9,10)
    - //div/(persName union placeName) → alle persName- und placeName-Elemente in div
    - //person[@type = “biblisch“] intersect //person/name[starts-with(., “A“)] → alle biblischen Personen, die mit A anfangen
    - //person[@type = “biblisch“] except //person/name[ends-with(., “E“)] → alle biblischen Personen außer denen, die mit E anfangen

XQuery - Ausdrücke

- **some/every $… in …satisfies (…)**
  - Quantifizierende Ausdrücke:
  - some: gibt es EIN Item in der Sequenz, das eine bestimmte Bedingung erfüllt?
  - every: erfüllen ALLE Items in der Sequenz eine bestimmte Bedingung?

  - Beispiele:
    - some $bibl in doc(“bibliografie.xml“)//bibl
      satisfies ($bibl/@ISBN = “978-0-596-00634-1)
    - every $bibl in doc(“bibliografie.xml“)//bibl
      satisfies ($bibl/@ISBN != ““)

XQuery - Ausdrücke

- **Variablen**
  - Syntax:
    - Erstellen: let *$variablenname* := *expression*
    - Aufrufen: *$variablenname*
  - Beispiele:
    - let $docImprint := doc(“tei-letter.xml“)//tei:docImprint
    - let $closer := “Hochachtungsvoll…“

XQuery - Ausdrücke

- **Funktionsaufrufe**
  - XPath-Funktionen
  - Benutzerdefinierte Funktionen
  - eXist-spezifische Funktionen
  - Beispiele:
    - replace(//p[1], “ä“, “ae“) → alle ä‘s im Absatz durch ae ersetzt
    - string-join((“a“,“b“,“c“), “; “) → “a; b; c“

XQuery - Ausdrücke

- **Kommentare**:
  - Syntax: (: *Kommentar* :)
  - können überall dort stehen, wo XQuery steht
  - werden im Ergebnisdokument nicht ausgegeben
  - Beispiele:
    - (: Hier wird eine Variable belegt :)
      let (: Name :) $autor := (: Wert:) //teiHeader//titleStmt/author
    - (: Hier wird für jedes Graphic-Element etwas getan :)
      for $graphic in //facsimile//graphic
      (: und zwar wird jeweils das Attribut @url zurückgegeben :)
      return $graphic/@url

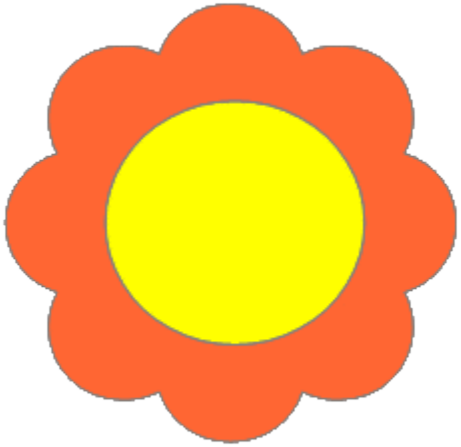
XQuery - Ausdrücke

- Das ist eine vollständige XQuery-Datei:
  - “Hallo Welt!“ → “Hallo Welt!“
- Dieses auch:
  - <h1>Unterm Birnbaum</h1> → <h1>Unterm Birnbaum</h1>
- Und auch dies:
  - <h1>{data(//tei:title)}</h1> → <h1>Unterm Birnbaum</h1>
- ABER: es gibt einen „typischen“ Aufbau für eine XQuery-Datei…

XQuery - Abfrage

- Eine „typische“ Datenbankabfrage:
  - **hole** bestimmte Daten aus der Datenbank
  - **filtere** sie: nur relevante Daten/Daten, die eine bestimmte Bedingung erfüllen sollen verarbeitet werden
  - **ordne** sie nach einem bestimmten Kriterium
  - **verarbeite** sie in irgendeiner Form weiter (ergänze sie, strukturiere sie um)
  - **gib** sie in einer bestimmten Form **zurück**

XQuery - FLWOR



for

let

where

order by *ascending* | *descending*

return

XQuery - FLWOR

- Beispiel:

```
for $absatz in doc("edition.xml")//p
let $initiale := substring($absatz, 1, 1)
let $rest := substring($absatz, 2)
where contains($absatz, "glücklich")
order by $initiale
return
    <p>
        <span class= "init">{$initiale}</span> {$rest}
    </p>
```

→ für jeden Absatz der Edition wird ein neuer Absatz zurückgegeben, der die Initiale in einem eigenen Element enthält und dann den restlichen Inhalt

XQuery - FLWOR

- let $initiale := substring(//p[1], 1, 1)
  return $initiale
  → die Initiale des ersten Absatzes wird zurückgegeben

- for $datum in doc("edition.xml")//date
  return {data($absatz)}
  → jedes Datum wird als Listeneintrag zurückgegeben

- for $ort in collection("register")//ort
  let $name := data($ort/name)
  for $ref in collection("edition")//ort[@ref = $ort/@id]
  return ("Ort:", $name, "Seite:", $ref/preceding::pb)

  → für jeden Ort aus der Sammlung „Register“ werden die Vorkommen in der Sammlung „Edition“ zurückgegeben (mit Seitenzahl)

XQuery - Syntax

- Kommentare: (: *String* :)
- Variablennamen ist ein *$* vorangestellt, der Name muss ein valider XML-Name sein
- die Zuweisung von Variablen erfolgt durch *:=*
- Wenn XQuery-Ausdrücke innerhalb von Elementkonstruktoren stehen, werden sie von geschweiften Klammern umschlossen: <element>{ *XQuery* }</element>

Aufbau einer XQuery-Datei

Prolog

- XQuery-Deklaration
- Namensraum-Deklarationen
- Schemaimport
- Variablen-Deklarationen
- ...

```
xquery version "1.0";

declare namespace tei="http://www.tei-c.org/ns/1.0";

import schema namespace tei="http://www.tei-c.org/ns/1.0"
at "tei.xsd";

(: globale Variablen und Funktionen :)
declare variable $local:transcript := doc("tei-transcript.xml");

...
```

Hauptteil

- Element-Konstruktor
- XPath
- FLWOR
- ...

→ Sequenz von Ausdrücken

```
(<h1>{local:transcript//tei:head[1]}</h1>,

//tei:div[1],

for $absatz in $local:transcript//tei:p
let $initiale := substring($absatz, 1, 1)
let $rest := substring($absatz, 2)
where contains($absatz, "glücklich")
order by $initiale
return
   <p id="{data($absatz/@xml:id)}">
     <span class="init">{$initiale}</span>
     {$rest}
   </p>)
```

Aufbau einer XQuery-Datei: Prolog

Prolog

- XQuery-Deklaration
- Namensraum-Deklarationen
- Schemaimport
- Variablen-Deklarationen
- ...

```
xquery version "1.0";

declare namespace tei="http://www.tei-c.org/ns/1.0";

import schema namespace tei="http://www.tei-c.org/ns/1.0"
at "tei.xsd";

(: globale Variablen und Funktionen :)
declare variable $local:transcript := doc("tei-transcript.xml");

...
```

Aufbau einer XQuery-Datei: Hauptteil

Hauptteil

- Element-Konstruktor
- XPath
- FLWOR
- ...

→ Sequenz von Ausdrücken

```
(<h1>{local:transcript//tei:head[1]}</h1>,

//tei:div[1],

for $absatz in $local:transcript//tei:p
let $initiale := substring($absatz, 1, 1)
let $rest := substring($absatz, 2)
where contains($absatz, "glücklich")
order by $initiale
return
   <p id="{data($absatz/@xml:id)}">
     <span class="init">{$initiale}</span>
     {$rest}
   </p>)
```

XQuery – Literatur

- W3Schools Tutorial: http://www.w3schools.com/xquery/default.asp
- W3C XQuery 1.0 Recommendation: http://www.w3.org/TR/xquery/
- W3C XQuery 1.0 and XPath 2.0 Functions and Operators Recommendation: http://www.w3.org/TR/xquery-operators/

- Jansen, Rudolf, *XQuery. Eine praxisorientierte Einführung*, Frankfurt 2004.
- Lehner, Wolfgang / Schöning, Harald, *XQuery. Grundlagen und fortgeschrittene Methoden*, Heidelberg 2004.
- Walmsley, Priscilla, *XQuery. Search across a variety of XML Data*, Sebastopol 2007.

Übungen: XQuery-Ausdrücke

Ausgangsdaten: hamlet.xml, neue XQuery-Datei (in Oxygen erstellen)

- Aufruf in eXist: localhost:8080/exist/rest/db/*…(Pfad zur Collection)…/*hamlet.xquery
- Aufruf in Oxygen: Transformationsszenario einrichten; hamlet.xml als Basis-XML

- Lösungen (kommentiert): hamlet.zip/hamlet.xquery

Übung 1: Titelliste

1. Titelliste aus einem Text generieren
   - Geben Sie mit XPath alle <TITLE>-Elemente aus
     - Achtung: im Browser sieht man die Ausgabe nur im Quelltext!
   - Konstruieren Sie ein -Element und geben Sie darin die <TITLE>-Elemente aus
     - Konstruktor!
   - Geben Sie für jedes-<TITLE>-Element ein -Element mit dem Inhalt des <TITLE>-Elements zurück
     - For-Schleife!
     - XPath-Funktion data() zur Ausgabe des Element-Inhalts

Übung 2: Sprecherstatistik

1. Wie oft sprechen die Figuren?
   - Geben Sie eine Liste der verschiedenen Sprecher (<SPEAKER>) aus
     - For-Schleife!
     - XPath-Funktion distinct-values()
   - Sortieren Sie die Liste alphabetisch aufsteigend
     - Order by
   - Geben Sie hinter jedem Sprecher die Anzahl seiner Reden (<SPEECH>) aus
     - XPath-Funktion count()
     - XPath-Bedingung
   - Sortieren Sie die Liste nach Häufigkeit der Rede um

Übung 3: Suchen

- Geben Sie alle Bühnenanweisungen (<STAGEDIR>) zurück, in denen Ophelia vorkommt
  - FLWOR; XPath-Funktion contains()
- Geben Sie alle Zeilen (<LINE>) zurück, in denen „love“ (groß oder klein geschrieben) vorkommt
  - FLWOR; XPath-Funktion contains()
- Geben Sie an, wie viele Treffer es sind
  - XPath-Funktion count()
- Geben Sie an, aus welchem Akt (ACT) und welcher Szene (SCENE) die Treffer stammen
  - XPath-Achsen, XPath-Funktion count()

Übung 4: Fragen

- Kommt Hamlet in allen Szenen vor?
  - every… in … satisfies
- Kommt jemand nur in einer Szene vor?
  - some… in… satisfies

Danke für die Aufmerksamkeit!