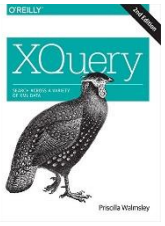
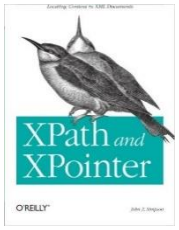
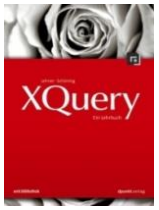
XQuery
XML Query Language

Josef Altmann

```
for $instructorNr in //@instructorNumber
let $pNr := replace($instructorNr, 'p','')
where starts-with($pNr, '22')
order by $pNr descending
return $pNr
```





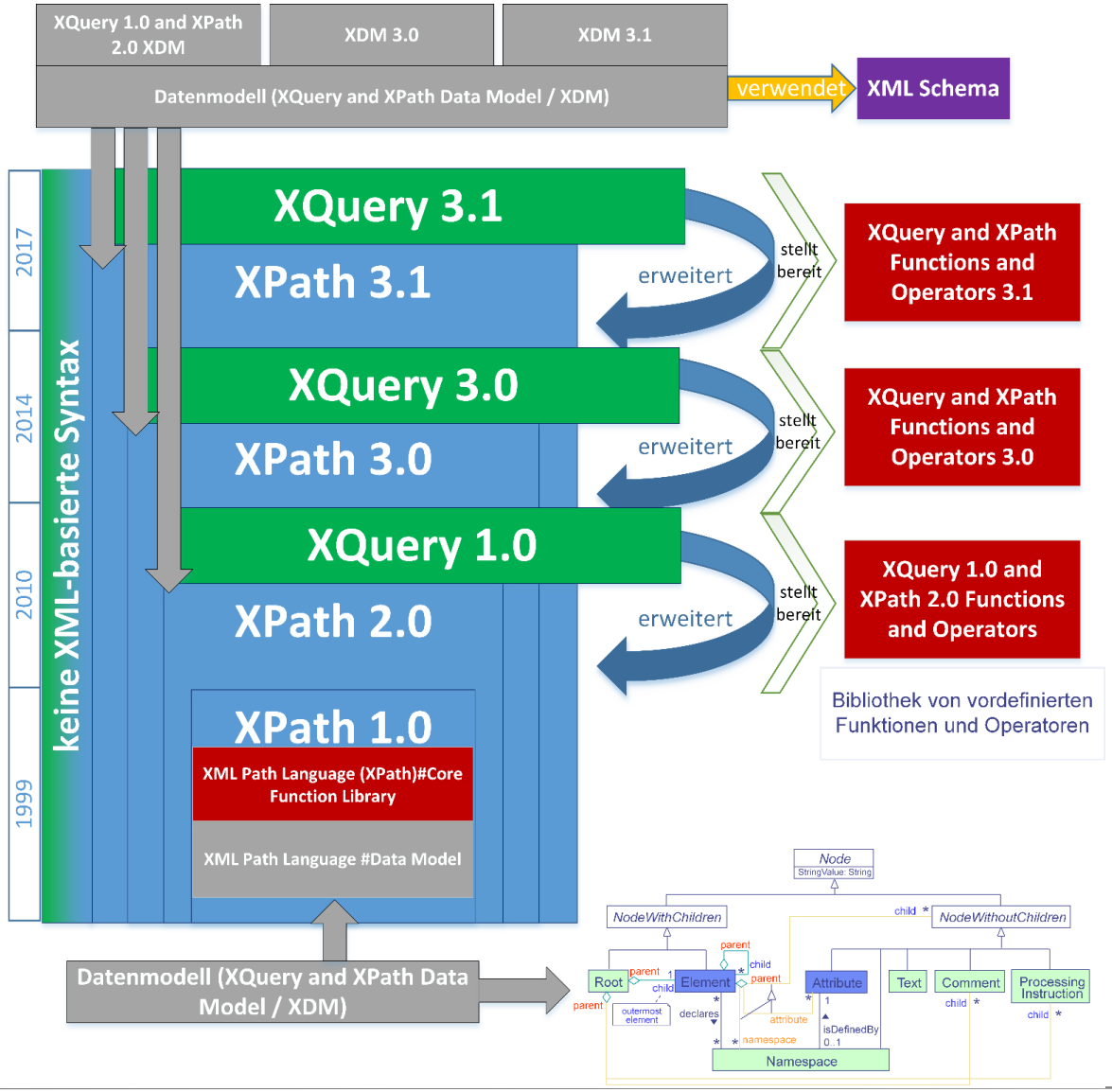


Der vorliegende Foliensatz basiert vorwiegend auf:

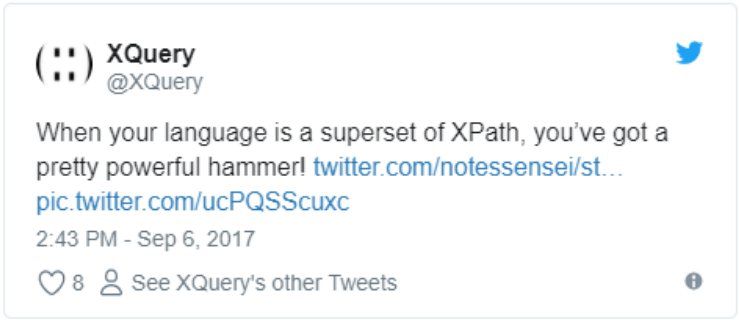
Kay, M.: XPath 2.0 Programmer's Reference (3rd ed.), Wiley, 2004.
Lehner, W., Schöning, H.: XQuery, dpunkt.verlag, 2004.
Simpson, J.: XPath and XPointer, O'Reilly, 2002.
Walmsley, P.: XQuery (2nd ed.), O'Reilly, 2015/2017.

Inhalt

XQuery
Versionen



XQuery 1.0
Sprachumfang (Auswahl)



[https://twitter.com/XQuery/status/905426178311376896]

- Bedingte Ausdrücke
- Arithmetische Ausdrücke
- Quantifizierende Ausdrücke
- Viele eingebaute Funktionen (> 100)
- Unterstützung von XML-Schema-Datentypen
- Verwendung mehrerer Dokumente
- Datenmodell: Knoten***sequenz***

- FLWOR-Ausdrücke
- Restrukturierung
- Element-Konstruktoren
- XQuery-Prolog
- Benutzerdef. Funktionen

XQuery 1.0

XPath 2.0

XPath 1.0

XSLT 2.0

- Stylesheets
- Templates
- Literale Ergebnis-elemente
- Benutzerdef. Funktionen

„SQL des 21. Jahrhunderts“

- Pfadausdrücke (Extraktion & Reduktion)
- Knotentests und Prädikate (Selektion)
- Vergleichsausdrücke
- Einige eingebaute Funktionen (27)
- Datenmodell: Knoten***menge***

Inhalt

- Einführung
- XQuery 1.0
  - Grundlagen
  - `for`- und `let`-Klausel
  - Hinzufügen von Elementen/Attributen
  - Konditionale Ausdrücke
  - Verbund
  - Quantifizierende Ausdrücke
  - Gruppierung
  - Sortierung und Aggregation
  - Programmstruktur
  - Dokumentation
- Anhang: XQuery Update Facility 1.0

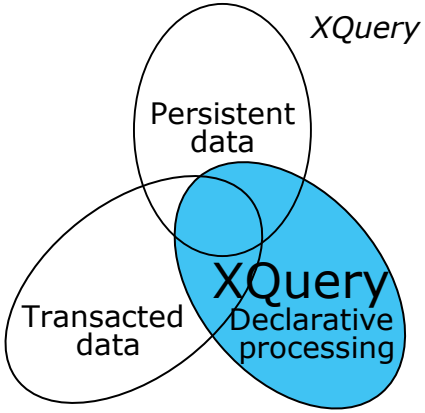
80% der Sprachkonzepte von XPath 2.0

Ähnlichkeit zu SQL

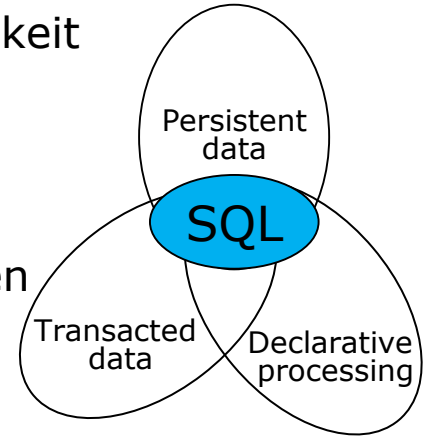
W3C-Standards:

- XQuery 1.0, März 2007
  - viele Zwischenschritte: 2003/2004/2005
- XQuery 3.0, April 2014
- XQuery 3.1, März 2017

XQuery
Grundlagen – Warum XQuery?



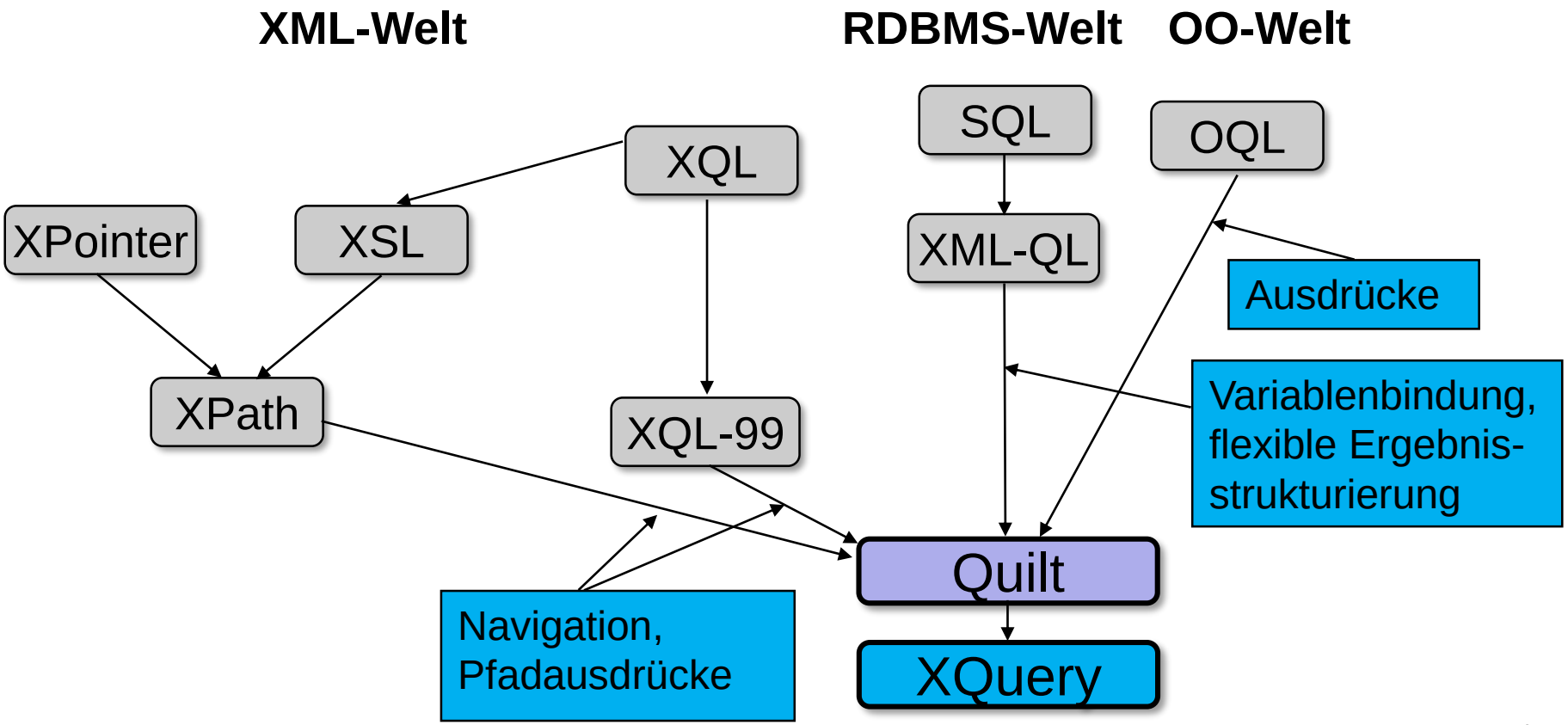
- Warum eine Abfragesprache für XML?
  - Logische/physische Datenunabhängigkeit
    - Abstraktes Datenmodell gewährleistet Unabhängigkeit von konkreter physischer Speicherung
  - Deklarative Programmierung
    - Beschreibe das *WAS*, nicht das *WIE*
    - Gemeinsamkeiten mit funktionalen und imperativen Programmiersprachen sowie mit Abfragesprachen



- Warum eine native Abfragesprache? Warum nicht SQL?
  - Eigenheiten von XML müssen berücksichtigt werden
  - Hierarchisch, geordnet, Text-basiert, ev. Schema-lose Struktur

XQuery 1.0
Grundlagen – Historie

- Ausgangsbasis: **Quilt** und Bausteine anderer Sprachen
- Hauptinitiatoren: IBM, INRIA, Software AG



XQuery 1.0
Grundlagen – Bestandteile der Spezifikation

- W3C-REC
  - XQuery 1.0 and XPath 2.0 Functions and Operators
    - Funktionen, die in XPath-Ausdrücken aufgerufen werden
    - Operatoren, die auf XPath-Datentypen ausgeführt werden
  - XQuery 1.0 and XPath 2.0 Data Model (XDM)
    - Gemeinsames Datenmodell von XPath 2.0 und XQuery
  - XSLT 2.0 and XQuery 1.0 Serialization
    - Ausgabe von XSLT 2.0- und XQuery-Ergebnissen in XML, HTML oder Text
  - XML Syntax for XQuery 1.0 (XQueryX)
    - XML-basierte Syntax für XQuery
  - XQuery 1.0 and XPath 2.0 Formal Semantics
    - Typsystem von XQuery definiert auf Basis von Xpath
  - XQuery Update Facility 1.0 (W3C REC; 17.3.2011)
    - Ändern, Einfügen, Löschen oder Ersetzen von Knoten
  - XQuery and XPath Full Text 1.0 (W3C REC; 17.3.2011)
    - Volltextsuchfuntion

- Java Community Process
  - **JSR 225:** XQuery API for Java - XQJ (~ JDBC)

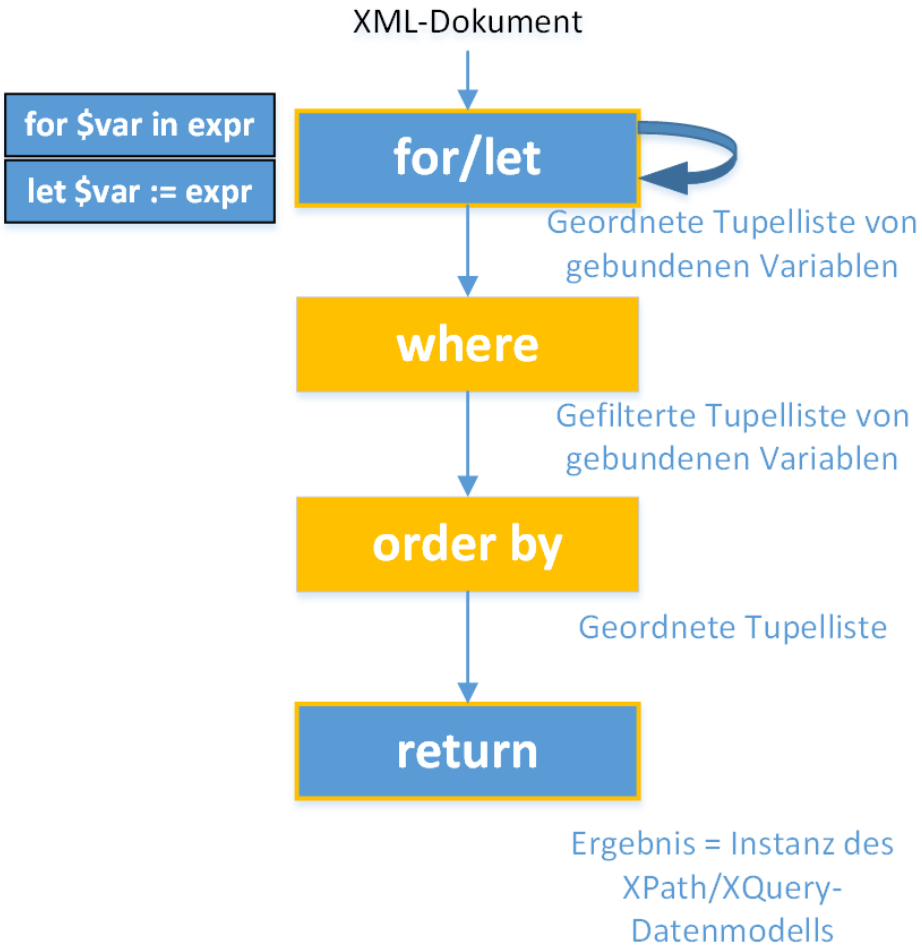
XQuery 1.0

Grundlagen – XQuery = **80 %** XPath 2.0 + **20 %** ...

- FLWOR (`for-let-where-order-return`)-Ausdruck
  - entspricht `SELECT-FROM-WHERE` in SQL
- Generieren von (u.a.) XML
  - Element- und Attributkonstruktoren, Transformation, ...
- Sortieren u. Gruppieren von Ergebnissen
- Operatoren für Typen
  - Typprüfung während statischer Analyse- und / oder dynamischer Evaluierungs-Phase
- Benutzerdefinierte Funktionen
  - Modularisierung von umfangreichen Anfragen
  - Rekursive Verarbeitung von Daten
- Streng typisiert
  - statisch wie dynamisch

XQuery 1.0

Grundlagen – FLWOR ['floʊer] Ausdruck 1/2



Iteration (vgl. `FROM` in SQL) und Var. Bindung

Variablen werden an Ausdruckswerte gebunden (basiert auf XPath)

Selektion (vgl. `WHERE` in SQL)

Filterung von Tupeln auf Basis von Prädikaten (optional)

Sortierung (vgl. `ORDER BY` in SQL)

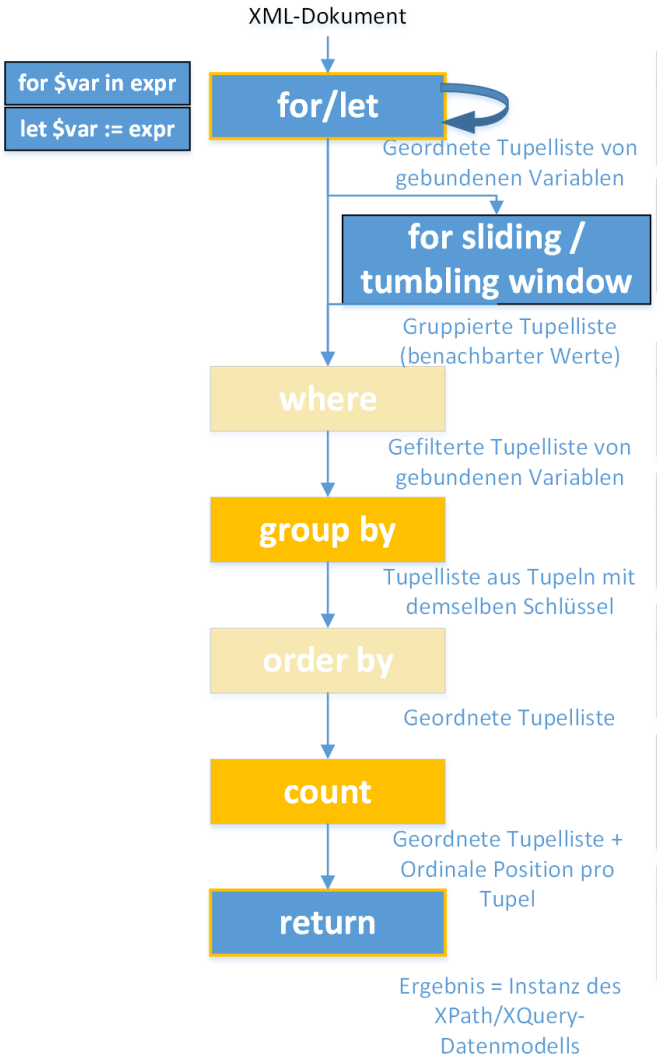
Sortierung von Tupeln auf Basis von Prädikaten (optional)

Generierung (vgl. `SELECT` in SQL)

Ergebnisgenerierung (einzelne Knoten, XML-Fragment oder atomare Werte)

XQuery 3.0

Grundlagen – FLWOR ['flo͡ər] Ausdruck



Iteration (vgl. `FROM` in SQL) und Var. Bindung

Variablen werden an Ausdruckswerte gebunden (basiert auf XPath)

(Sliding/Tumbling) Window

„Gruppieren" von benachbarten Werten (einer Sequenz) zu einzelnen „Fenstern"

Selektion (vgl. `WHERE` in SQL)

Filterung von Tupeln auf Basis von Prädikaten (optional)

Gruppierung (vgl. `GROUP BY` in SQL)

Gruppierung von Tupeln auf Basis von Gruppierungs-Schlüssel

Sortierung (vgl. `ORDER BY` in SQL)

Sortierung von Tupeln auf Basis von Prädikaten (optional)

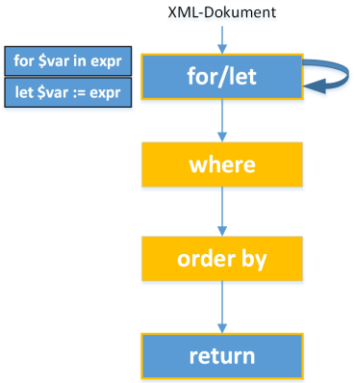
Count

Einführung einer neuen Variable, die pro Tupel die ordinalte Position in der Tupelliste enthält.

Generierung (vgl. `SELECT` in SQL)

Ergebnisgenerierung (einzelne Knoten, XML-Fragment oder atomare Werte)

XQuery
Grundlagen – FLWOR ['flɔwer] Ausdruck 2/2



Variablenbindung Funktionsaufruf *XPath*-Ausdruck

```
for $course in doc("CourseCatalog.xml")//Course
where contains($course/Title, 'XML')
return $course/Title
```

Variablenreferenzierung (+ XPath)

Vordefinierte Funktion (aus „Functions and Operators“)

```
<Title>Introduction to semi-structured data models and XML</Title>
<Title>Introduction to semi-structured data models and XML</Title>
```

FLWOR-Ausdrücke

unterstützen
- Sortierung
- Verbund und Gruppierung
- Hinzufügen von Elementen und Attributen zum Ergebnis

```
doc("CourseCatalog.xml")//Course[contains(Title, 'XML')]/Title
```

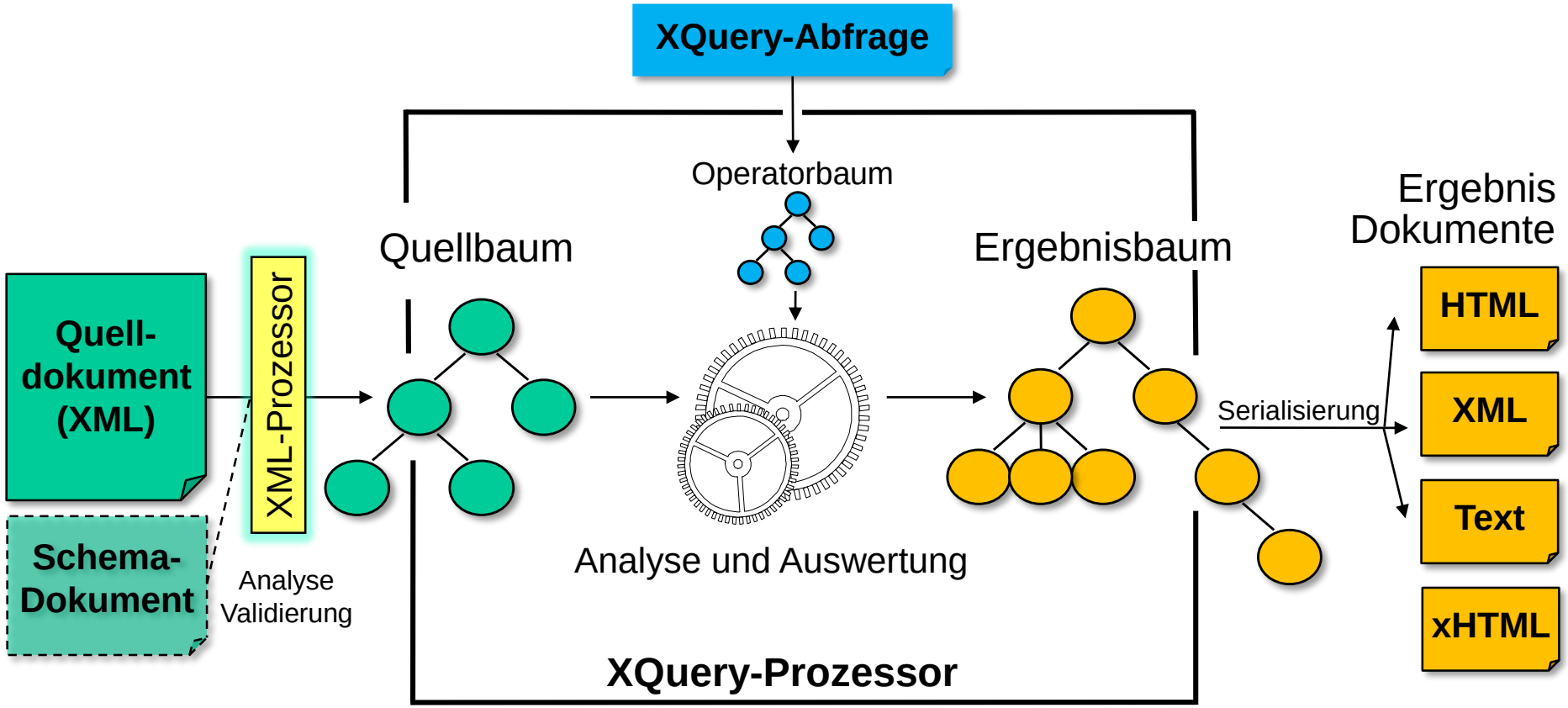
Pfadausdrücke

- Gute Unterstützung, wenn in Abfragen ausgewählte Elemente und Attribute **unverändert** in das Ergebnis übernommen werden!

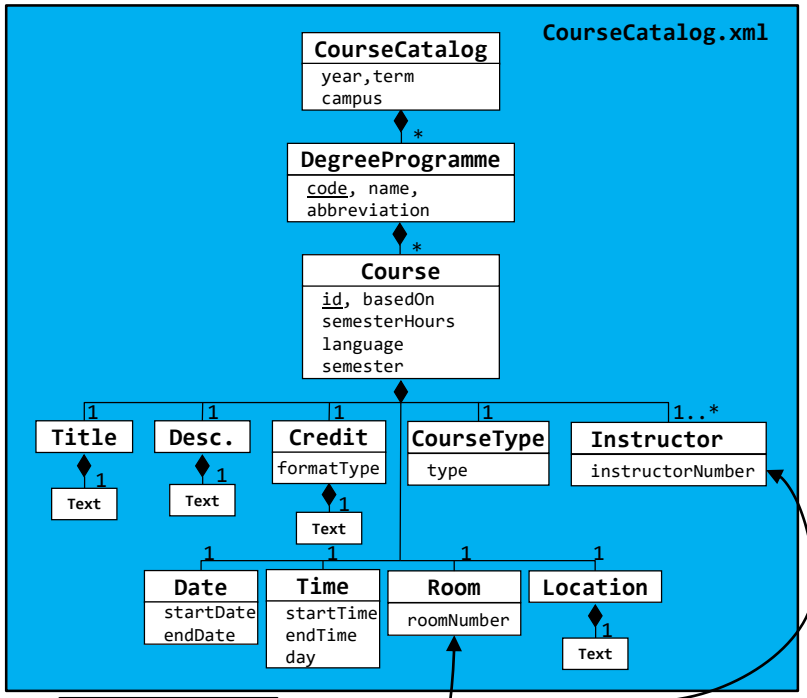
XQuery
Grundlagen – Syntax

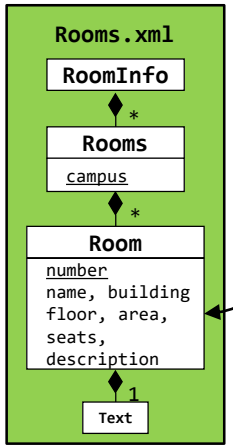
- Geschachtelte FLWOR-Ausdrücke
- Kompakt, keine XML-Syntax!
- ABER Namen müssen gültige XML-Namen sein
  - Variablen, Funktionen, Elemente, etc.
  - Verwendung von Namensräumen
- Keine reservierten Wörter
- Case-sensitive
  - Schlüsselwörter mit Kleinbuchstaben
- Kein spezielles End-Of-Line-Zeichen
- Kommentar beginnt mit **(:** und endet mit **:)**
  - beliebig tiefe Schachtelung
  - über mehrere Zeilen
  - erscheint nicht im Ergebnis
- Whitespace-Zeichen
  - haben keine Bedeutung

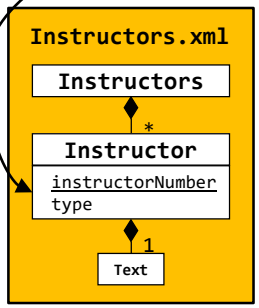
XQuery
Grundlagen – Verarbeitungsmodell



Beispiel
CourseCatalog








```
<Instructors>
  <Instructor instructorNumber="p20621" type="HBL">Josef Altmann</Instructor>
  <Instructor instructorNumber="p22080" type="NBL">Julian Haslinger</Instructor>
  <Instructor instructorNumber="p20622" type="NBL">Norbert Niklas</Instructor>
  <Instructor instructorNumber="p20623" type="HBL">Barbara Traxler</Instructor>
</Instructors>
```

```
<RoomInfo>
  <Rooms campus="Hagenberg">
    <Room name="LBS1" number="3.008" building="FH 3" floor="0"
          description="SE Labor 1" area="77.72" seats="24"/>
    <Room name="LBS2" number="3.009" building="FH 3" floor="0"
          description="SE Labor 2" area="77.72" seats="24"/>
<!-- ... -->
    <Room name="HS3" number="2.025" building="FH 2" floor="0"
          description="bet-at-home.com Hörsaal 3" area="156.05" seats="120"/>
    <Room name="HS4" number="1.004" building="FH 1" floor="0"
          description="CELUM HS2" area="95.20" seats="90"/>
<!-- ... -->
</RoomInfo>
```

```
<CourseCatalog year="2019" term="summer" campus="Hagenberg">
  <DegreeProgramme code="0307" name="Software Engineering" abbreviation="SE">
    <Course id="cID_8314" semesterHours="1" language="en" semester="4">
      <Title>Introduction to semi-structured data models and XML</Title>
      <Description>Introduction of skills related to XML.<Content>Includes DTD, Schema,
                   XPath, XQuery, XSLT, JSON</Content> <Exam>Final Exam
                   required.</Exam>Participation without any previous knowledge.
      </Description>
      <Credit formatType="ECTS">1</Credit>
      <CourseType type="Lecture"/>
      <Date startDate="--02-28" endDate="--05-03"/>
      <Time startTime="08:00:00" endTime="10:25:00" day="THU"/>
      <Room roomNumber="1.004"/>
      <Instructor instructorNumber="p22080"/>
    </Course>
<!-- ... -->
```

XQuery
for / let-Klausel

{} Auswertungskontext

- let-Klausel mit to-Bereichsoperator

```
let $i := (1 to 3)
return <value>{$i}</value>
```

```
<value>1 2 3</value>
```

- for-Klausel mit to-Bereichsoperator

```
for $i in (1 to 3)
return <value>{$i}</value>
```

```
<value>1</value>
<value>2</value>
<value>3</value>
```

- Schachtelung von for-Klauseln

```
for $i in (1 to 2)
for $j in ('a','b')
return
  <result>
      $i is {$i} and $j is {$j}
  </result>
```

```
<result>$i is 1 and $j is a</result>
<result>$i is 1 and $j is b</result>
<result>$i is 2 and $j is a</result>
<result>$i is 2 and $j is b</result>
```

- for-Klausel mit mehreren Variablenbindungen

```
for $i in (1 to 2), $j in ('a','b')
return
  <result>
    $i is {$i} and $j is {$j}
  </result>
```

Sequenz mit 2 Items: a, b

XQuery
for / let-Klausel

- Einfache `for` / `let`-Klausel

```
for $course in doc("CourseCatalog.xml")//Course
let $courseType := $course/CourseType/@type
where $courseType = 'Training'
return $course/Title
```

```
<Title>Intercultural Communications</Title>
```

- Mehrere `for` / `let`-Klauseln

```
let $doc := doc("CourseCatalog.xml")
for $course in $doc//Course
let $courseType := $course/CourseType/@type
let $courseTitle := $course/Title
where $courseType = 'Training'
return $courseTitle
```

Reihenfolge könnte zusätzlich geändert werden

```
<Title>Intercultural Communications</Title>
```

XQuery
for / let-Klausel – Namespaces 1/2

- Einfache `for` / `let`-Klausel – Elemente in bestimmten Namespace

```
declare default element namespace "http://www.fh-ooe.at/CourseCatalog";
for $course in doc("CourseCatalogWithSchema.xml")//Course
let $courseType := $course/CourseType/@type
where $courseType = 'Training'
return $course/Title
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Title xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns="http://www.fh-ooe.at/CourseCatalog"
       xmlns:cc="http://www.fh-ooe.at/CourseCatalog"
> Intercultural Communications</Title>
```

Default-Namespace wird übernommen

Andere Namespaces aus dem Quell-Dokument

XQuery
for / let-Klausel – Namespaces 2/2

- Einfache `for` / `let`-Klausel – Elemente in bestimmten Namespace

```
declare namespace catalog = "http://www.fh-ooe.at/CourseCatalog";
for $course in doc("CourseCatalogWithSchema.xml")//catalog:Course
let $courseType := $course/catalog:CourseType/@type
where $courseType = 'Training'
return $course/catalog:Title
```

Namespace-Deklaration
(Namespace, aus dem die zu selektierenden Elemente stammen)

```
<?xml version="1.0" encoding="UTF-8"?>
<Title xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns="http://www.fh-ooe.at/CourseCatalog"
       xmlns:cc="http://www.fh-ooe.at/CourseCatalog"
> Intercultural Communications</Title>
```

Default-Namespace wird übernommen

Andere Namespaces aus dem Quell-Dokument

XQuery
Sortierung (order by)

- order by-Klausel

```
for $room in doc("CourseCatalog.xml")//Room
order by $room/@building
return $room
```

```
<Room roomNumber="1.004"/>
<Room roomNumber="2.010" building="FH2">Audimax</Room>
<Room roomNumber="3.108" building="FH3">XORTEX LBS3</Room>
```

- Mehrere Sortierkriterien

```
for $room in doc("CourseCatalog.xml")//Room
order by $room/@building descending, $room descending
return $room
```

```
<Room roomNumber="3.108" building="FH3">XORTEX LBS3</Room>
<Room roomNumber="2.010" building="FH2">Audimax</Room>
<Room roomNumber="1.004"/>
```

XQuery
Hinzufügen von Elementen/Attributen

(1) Unveränderte Übernahme aus Eingangsdokument
- Einfache Elemente
- Komplexe Elemente – zusätzlich Attribute und Sub-Elemente
- ➢ Elemente, Attribute, etc. nicht änderbar

(2) Direkter Element-/Attributkonstruktor – Mischung aus ...
- Literale – direkte Übernahme in das Ausgabedokument
- Ausdrücke in **{}** werden zu Knoten (Elemente, Attribute, etc.) und atomare Werte ausgewertet
- Direkter Elementkonstruktor
- ➢ muss auf XML-Syntax basieren

(3) Berechneter Konstruktor
- Berechnung des Namens und des Inhalts von Elementen
- Transformation eines Eingangsdokuments in ein Ausgangsdokument (mit geänderten Elementen, Attributen und Inhalten)

XQuery
(1) Unveränderte Übernahme aus Eingangsdokument

- Übernahme von einfachen Elementen

```
for $course in
doc("CourseCatalog.xml")//Course[@id="cID_8314"]
return $course/Title
```

```
<Title>Introduction to semi-structured data models and XML</Title>
```

- Übernahme von komplexen Elementen

```
for $course in
doc("CourseCatalog.xml")//Course[@id="cID_8314"]
return $course
```

```
<Course id="cID_8314" semesterHours="1" language="en" semester="4">
   <Title>Introduction to semi-structured data models and XML</Title>
   <Description>Introduction of skills related to XML.<Content>Includes DTD, Schema, XPath,
        XQuery, XSLT, JSON</Content><Exam>Final Exam required.</Exam>Participation without
        any previous knowledge. </Description>
   <Credit formatType="ECTS">1</Credit>
   <CourseType type="Lecture"/>
   <Date startDate="--02-28" endDate="--05-03"/>
   <Time startTime="08:00:00" endTime="10:25:00" day="THU"/>
   <Room roomNumber="1.004"/>
   <Instructor instructorNumber="p22080"/>
</Course>
```

XQuery
(2) Direkter Konstruktor 1/7

- Schließe Ergebnis (`Title`-Elemente) in `ul`-Element ein

```
<ul>
{
for $course in
doc("CourseCatalog.xml")//Course[@id="cID_8314"]
return $course/Title
}
</ul>
```

Literaler Inhalt

```
<ul>
   <Title>Introduction to semi-structured data models and XML</Title>
</ul>
```

- Schließe jedes `Title`-Element in ein `li`-Element ein

```
<ul>
{
for $course in
doc("CourseCatalog.xml")//Course[@id="cID_8314"]
return <li>{$course/Title}</li>
}
</ul>
```

Literaler Inhalt

```
<ul>
   <li>
      <Title>Introduction to semi-
structured data models and XML</Title>
   </li>
</ul>
```

XQuery
(2) Direkter Konstruktor 2/7

- Neue Attribute hinzufügen, Attributwert/Elementinhalt übernehmen

```
<ul type="Courses">
{
for $course in doc("CourseCatalog.xml")//Course
order by $course/Time/@startTime descending
return <li time="{$course/Time/@startTime}">{data($course/Title)}</li>
}
</ul>
```

Neuer Attributname & -wert

Neuer Attributname & Übernahme von existierendem Wert

Übernahme Elementinhalt via data()-Funktion

```
<ul type="Courses">
   <li time="11:20:00">Introduction to semi-structured data models and XML</li>
   <li time="10:30:00">Data modelling and database design</li>
   <li time="09:00:00">Intercultural Communications</li>
   <li time="08:00:00">Introduction to semi-structured data models and XML</li>
</ul>
```

XQuery
(2) Direkter Konstruktor 3/7

- Elementinhalt als Attributwert mit Präfix (ECTS_) übernehmen

```
for $course in doc("CourseCatalog.xml")//Course
order by $course/Time/@startTime
return
  <new_course
    ects="ECTS_{$course/Credit}"
    title="{$course/Title}">
  </new_course>
```

data()-Funktion nicht notwendig – Automatische **Atomisierung**

```
<new_course ects="ECTS_1"
            title="Introduction to semi-structured data models and XML"/>
<new_course ects="ECTS_0.5" title="Intercultural Communications"/>
<new_course ects="ECTS_2" title="Data modelling and database design"/>
<new_course ects="ECTS_1.5"
            title="Introduction to semi-structured data models and XML"/>
```

XQuery
(2) Direkter Konstruktor 4/7

- Attribute/Element übernehmen und bestimmte Attribute/Elemente eliminieren

```
for $course in doc("CourseCatalog.xml")//Course
where contains($course/Title, "Intro")
order by $course/Time/@startTime
return
  <new_course>
     {$course/(@* except @id, * except (Credit, CourseType, Location,
                                        Description, Instructor, Date))
     }
  </new_course>
```

course-**Attribute** in new_course-Attribute übernehmen
Ausnahme: Attribut @id wird nicht übernommen.

course-**Elemente** übernehmen und
als Sub-Elemente in new_course-Element aufnehmen

Eliminiere bestimmte Sub-Elemente von
course-Element

```
<new_course semesterHours="1" language="en" semester="4">
   <Title>Introduction to semi-structured data models and XML</Title>
   <Time startTime="08:00:00" endTime="10:25:00" day="THU"/>
   <Room roomNumber="1.004"/>
</new_course>
<new_course semesterHours="2" language="en" semester="4" basedOn="cID_8314">
   <Title>Introduction to semi-structured data models and XML</Title>
   <Time startTime="11:20:00" endTime="14:35:00" day="SAT"/>
   <Room roomNumber="3.108" building="FH3">XORTEX LBS3</Room>
</new_course>
```

XQuery
(2) Direkter Konstruktor 5/7

- Aggregatfunktion – Tags von Eingangsdokument werden nicht übernommen

```
<html>
 <h1>Kurse</h1>
 <p>Es gibt insgesamt {count(doc("CourseCatalog.xml")//Course)} Kurse</p>
</html>
```

```
<html>
   <h1>Kurse</h1>
   <p>Es gibt insgesamt 4 Kurse</p>
</html>
```

XQuery
(2) Direkter Konstruktor 6/7

- Eingeschlossener Ausdruck erzeugt Elemente

```
for $course in doc("CourseCatalog.xml")//Course
return
  <li>ECTS: {$course/Credit}</li>
```

```
<li>ECTS: <Credit formatType="ECTS">1</Credit></li>
<li>ECTS: <Credit formatType="ECTS">1.5</Credit></li>
<li>ECTS: <Credit formatType="ECTS">0.5</Credit></li>
<li>ECTS: <Credit formatType="ECTS">2</Credit></li>
```

- Eingeschlossener Ausdruck erzeugt Attribute

```
for $course in doc("CourseCatalog.xml")//Course
return
 <li>{$course/@id}ECTS: {$course/Credit}</li>
```

```
<li id="cID_8314">ECTS: <Credit formatType="ECTS">1</Credit></li>
<li id="cID_8315">ECTS: <Credit formatType="ECTS">1.5</Credit></li>
<li id="cID_7555">ECTS: <Credit formatType="ECTS">0.5</Credit></li>
<li id="cID_8840">ECTS: <Credit formatType="ECTS">2</Credit></li>
```

XQuery
(2) Direkter Konstruktor 7/7

- Ausdruck mit mehreren Sub-Ausdrücken

```
for $course in doc("CourseCatalog.xml")//Course
return
  <li>
    {$course/@id, 'ECTS:', 9*4, $course/Credit}
  </li>
```

Element

Attribut

Element-Inhalt

```
<li id="cID_8314">ECTS: 36<Credit formatType="ECTS">1</Credit></li>
<li id="cID_8315">ECTS: 36<Credit formatType="ECTS">1.5</Credit></li>
<li id="cID_7555">ECTS: 36<Credit formatType="ECTS">0.5</Credit></li>
<li id="cID_8840">ECTS: 36<Credit formatType="ECTS">2</Credit></li>
```

XQuery
(3) Berechneter Konstruktor 1/4

- Neues Element erzeugen (dynamisch berechneter Element-Name)

```
for $course in doc("CourseCatalog.xml")//Course
where contains($course/@id, '8314')
return
 element {$course/@id} {data($course/Title)}
```

```
<cID_8314>Introduction to semi-structured data models and XML</cID_8314>
```

- Neues Element erzeugen

```
for $course in doc("CourseCatalog.xml")//Course
where contains($course/@id, '8314')
return
 element Kurs {data($course/Title)}
```

```
<Kurs>Introduction to semi-structured data models and XML</Kurs>
```

XQuery
(3) Berechneter Konstruktor 2/4

- Neues Attribut erzeugen (dynamisch berechneter Name)

```
for $course in doc("CourseCatalog.xml")//Course
where contains($course/@id, '8314')
return
  element Kurs { attribute {$course/CourseType/@type} {$course/@id}}
```

```
<Kurs Lecture="cID_8314"/>
```

- Neues Attribut erzeugen

```
for $course in doc("CourseCatalog.xml")//Course
where contains($course/@id, '8314')
return
  element Kurs { attribute ID {$course/@id}}
```

```
<Kurs ID="cID_8314"/>
```

XQuery
(3) Berechneter Konstruktor 3/4

- Mehrere Attribute und Elemente hinzufügen

```
let $a := true()
return
element Kurs
  {
    attribute A {"A"},
    attribute B {12},
    element KursDetails
      {"Details"}
  }
```

```
<Kurs A="A" B="12">
   <KursDetails>Details</KursDetails>
</Kurs>
```

XQuery
(3) Berechneter Konstruktor 4/4

- Inhalt in XML-Markup transformieren
  - Attributinhalt ⇨Elemente
  - Expliziter Elementkonstruktor

```
for $wd in distinct-values(doc("CourseCatalog.xml")//@day)
return
  element {$wd}
          { doc("CourseCatalog.xml")//Course[Time/@day = $wd]/Title}
```

```
<THU>
   <Title>Introduction to semi-structured data models and XML</Title>
</THU>
<SAT>
   <Title>Introduction to semi-structured data models and XML</Title>
</SAT>
<MON>
   <Title>Data modelling and database design</Title>
</MON>
```

XQuery
Konstruktoren und Namespaces

- Neue Elemente können zusätzlich einem bestimmten Namespace zugeordnet werden
  - Deklaration eines Präfix und Verwendung bei der Erstellung der Ergebnis-Sequenz

```
declare namespace new = "http://www.fh-ooe.at/NewCourse";

for $course in doc("CourseCatalogXQ.xml")//Course
let $courseType := $course/CourseType/@type
where $courseType = 'Training'
return element new:entry {data($course/Title)}
```

auch: `return <new:entry>{data($course/Title)}</new:entry>`

```
<new:entry xmlns:new="http://www.fh-ooe.at/NewCourse">Intercultural
Communications</new:entry>
```

XQuery
Konditionale Ausdrücke - if

- Einfacher konditionaler Ausdruck
  - Teil von Return-Klausel

```
for $course in doc("CourseCatalog.xml")//Course
let $type := $course/CourseType/@type
return
  if ($type = 'Lecture') then
    <VO>{$course/@id, data($course/Title)}</VO>
  else if ($type ='LabSession') then
    <UE>{$course/@id, data($course/Title)}</UE>
  else
    <Sonstiges>{$type, data($course/Title)}</Sonstiges>
```

Rückgabe-Wert muss einem XQuery-Datentyp entsprechen

```
<VO id="cID_8314">Introduction to semi-structured data models and XML</VO>
<UE id="cID_8315">Introduction to semi-structured data models and XML</UE>
<Sonstiges id="cID_7555"/>
<Training>Intercultural Communications</Training>
<Location>White House and Moskovskij Kreml</Location>
<VO id="cID_8840">Data modelling and database design</VO>
```

XQuery
Konditionale Ausdrücke - if

- Konditionaler Ausdruck mit mehreren Ausdrücken als Rückgabewert

```
for $course in doc("CourseCatalog.xml")//Course
let $type := $course/CourseType/@type
return
  if ($type = 'Lecture') then
    <VO>{$course/@id, data($course/Title)}</VO>
  else if ($type ='LabSession') then
    <UE>{$course/@id, data($course/Title)}</UE>
  else
    (
     <Sonstiges>{$course/@id}</Sonstiges>,
     element {$type} {data($course/Title)},
     element Location {data($course/Location)}
     )
```

Item-Sequenz (inkl. berechnete Elementkonstruktoren)

```
<VO id="cID_8314">Introduction to semi-structured data models and XML</VO>
<UE id="cID_8315">Introduction to semi-structured data models and XML</UE>
<Sonstiges id="cID_7555"/>
<Training>Intercultural Communications</Training>
<Location>White House and Moskovskij Kreml</Location>
<VO id="cID_8840">Data modelling and database design</VO>
```

XQuery

Konditionale Ausdrücke – switch (XQuery 3.0)

```
for $course in doc("CourseCatalog.xml")//Course
return
  switch ($course/CourseType/@type)
    case "LabSession"
      return <übung>{data($course/Title)}</übung>
    case "Lecture"
      return <vorlesung>{data($course/Title)}</vorlesung>
    default
      return <anderes>{data($course/Title)}</anderes>
```

```
<vorlesung>Introduction to semi-structured data models and XML</vorlesung>
<übung>Introduction to semi-structured data models and XML</übung>
<anderes>Intercultural Communications</anderes>
```

XQuery
Verbunde 1/3

```
<info>Course Introduction to
semi-structured data models and
XML is in room 3.108</info>
```

- Verbund mit Verbundprädikat

```
for $course in doc("CourseCatalog.xml")//Course
for $room in doc("Rooms.xml")//Room[@number = $course/Room/@roomNumber]
return
<info>
  {
    concat('Course ', $course/Title, ' is in room ', $room/@number)
  }
</info>
```

- Verbund mit Verbundprädikat in der where-Klausel

```
for $course in doc("CourseCatalog.xml")//Course
for $room in doc("Rooms.xml")//Room
where $room/@number = $course/Room/@roomNumber
return
<info>
  {
    concat('Course ', $course/Title, ' is in room ', $room/@number)
  }
</info>
```

XQuery
Verbunde 2/3

- Verbund mit 3 XML-Dokumenten in where-Klausel

```
for
  $course in doc("CourseCatalog.xml")//Course,
  $room in doc("Rooms.xml")//Room,
  $instructor in doc("Instructors.xml")//Instructor
where
  $room/@number = $course/Room/@roomNumber and
  $course/Instructor/@instructorNumber = $instructor/@instructorNumber
return
  <info>
  {
    concat('Course "', $course/Title, '" is in room FH',
           $room/@number, ' (lecturer: ', $instructor, ')')
  }
  </info>
```

```
<info>Course "Introduction to semi-structured data models and XML" is in room FH3.108
(lecturer: Barbara Traxler)</info>
```

XQuery
Verbunde 3/3

- Äußerer Verbund

```
for $course in doc("CourseCatalog.xml")//Course
return
<info course = "{$course/Title}">
{
  attribute room {
    for $room in doc("Rooms.xml")//Room
    where $room/@number = $course/Room/@roomNumber
    return $room/@number
  }
}
</info>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<info course="Introduction to semi-structured data models and XML" room=""/>
<info course="Introduction to semi-structured data models and XML"
      room="3.108"/>
<info course="Intercultural Communications" room=""/>
<info course="Data modelling and database design" room=""/>
```

XQuery
Quantifizierende Ausdrücke

- Existenzielle Quantifizierung - some

```
some $course in doc("CourseCatalog.xml")//Course satisfies
$course//CourseType/@type = "LabSession"
```

true

- Universelle Quantifizierung - every

```
every $course in doc("CourseCatalog.xml")//Course satisfies
$course//Credit[@formatType="ECTS"] > 1
```

false

- not-Funktion mit some

```
not(some $course in doc("CourseCatalog.xml")//Course satisfies
$course//CourseType/@type = "LabSession")
```

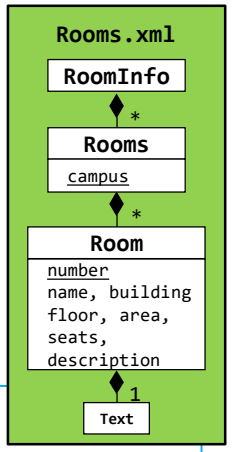
false

- Quantifizierende Ausdrücke mit mehreren Variablen

```
some $course in doc("CourseCatalog.xml")//Course,
     $room in doc("Rooms.xml")//Room
satisfies contains($room/@number, $course/Room/@roomNumber)
```

true

XQuery
Duplikat-Eliminierung und Gruppierung

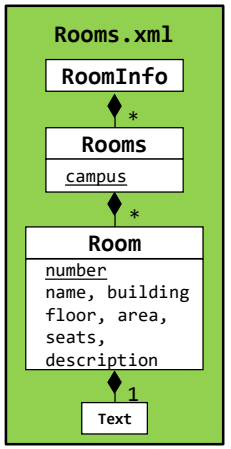


- Gruppierung der Räume nach Gebäude (**@building**)

```
for $fh in distinct-values(doc("Rooms.xml")//Room/@building)
let $rooms := doc("Rooms.xml")//Room[@building = $fh]
order by $fh
return
  <fh id="{$fh}">
    {
      for $room in $rooms
      order by $room/@number
      return
        <room>{$room/@name}</room>
    }
  </fh>
```

```
<fh id="FH 2">
   <room name="AM"/>
   <room name="HS3"/>
   <room name="SRC1"/>
   <room name="LBB2"/>
   <room name="LBB1"/>
   <room name="LBS5"/>
   <room name="SRC2"/>
</fh>
<fh id="FH 3">
   <room name="LBS1"/>
   <room name="LBS2"/>
   <room name="LBS3"/>
   <room name="LBS4"/>
</fh>
```

XQuery
Gruppierung – group by (XQuery 3.0)



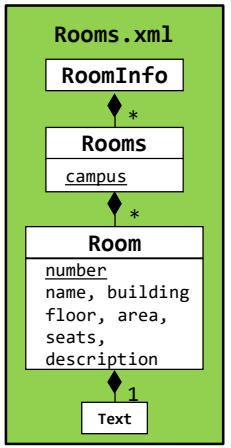
- Gruppierung (group by) der Räume nach Gebäude (@building)

```
for $rooms in doc("Rooms.xml")//Room
let $fh := $rooms/@building
group by $fh
order by $fh
return
<rooms fh="{$fh}">
{
  for $room in $rooms
  order by $room/@number
  return <room>{$room/@name}</room>
}
</rooms>
```

```
<fh id="FH 2">
   <room name="AM"/>
   <room name="HS3"/>
   <room name="SRC1"/>
   <room name="LBB2"/>
   <room name="LBB1"/>
   <room name="LBS5"/>
   <room name="SRC2"/>
</fh>
<fh id="FH 3">
   <room name="LBS1"/>
   <room name="LBS2"/>
   <room name="LBS3"/>
   <room name="LBS4"/>
</fh>
```

return-Ausdruck wird für jede Gruppe einmal ausgeführt

XQuery
Duplikateliminierung, Sortierung und Aggregation 1/2



- Aggregation – **sum**

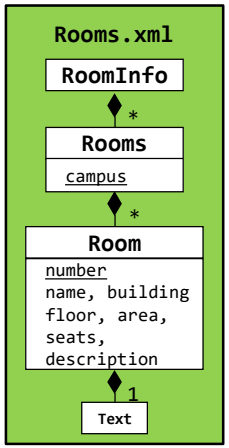
```
for $fh in distinct-values(doc("Rooms.xml")//Room/@building)
let $rooms := doc("Rooms.xml")//Room[@building = $fh]
order by $fh
return
  <fh
      id="{$fh}"
      totalArea="{sum($rooms/@area)}"
      seats="{sum($rooms/@seats)}"
   />
```

```
<fh id="FH 1" totalArea="95.2" seats="90"/>
<fh id="FH 2" totalArea="842.61" seats="497"/>
<fh id="FH 3" totalArea="310.88" seats="92"/>
```

- Aggregation – **sum** (mit **group by**)

```
for $rooms in doc("Rooms.xml")//Room
group by $building := $rooms/@building
order by $building
return
  <fh
    id="{$building}"
    totalArea="{sum($rooms/@area)}"
    seats="{sum($rooms/@seats)}”
  />
```

XQuery
Duplikateliminierung, Sortierung und Aggregation 2/2



- Aggregation von mehreren Werten – `count, sum`

```
for $fh in distinct-values(doc("Rooms.xml")//Room/@building)
let $rooms := doc("Rooms.xml")//Room[@building = $fh]
order by $fh
return
  <building
      id="{$fh}"
      totalRooms="{count($rooms)}"
      distinctFloors="{count(distinct-values($rooms/@floor))}"
      seats="{sum($rooms/@seats)}"
   />
```

```
<?xml version="1.0" encoding="UTF-8"?>
<building id="FH 1" totalRooms="1" distinctFloors="1" seats="90"/>
<building id="FH 2" totalRooms="7" distinctFloors="4" seats="497"/>
<building id="FH 3" totalRooms="4" distinctFloors="2" seats="92"/>
```

XQuery
Quantifizierende Ausdrücke

- `where`-Klausel mit mehreren Ausdrücken und einer `exists`-Quantifizierung

```
for $course in doc("CourseCatalog.xml")//Course
let $id := $course/@id where
   starts-with($course/Title, "Intro") and
   $course/Credit/@formatType = 'ECTS' and
   exists($course/@basedOn)
return
   (data($course/Title), ':', data($course/@id))
```

Es werden nur Kurse, die auf anderen Kursen aufbauen, in das Ergebnis aufgenommen.

```
Introduction to semi-structured data models and XML
cID_8315
```

Ergebnis: Sequenz mit drei Items (*kein* XML-Konstrukt!)

Inhalt

Programmstruktur - Module

- Hauptmodul (*main module, *.xq*)
  - Prolog
    - Import anderer Module (Bibliothekmodule, *library modules*)
  - Body
    - Implementierung der Funktionalität
- Bibliothekmodul (*library module, *.xqm*)
  - Unterschied zu Hauptmodulen:
    - Kein Body vorhanden
    - Beginnen mit einer Modul-Deklaration
- Beide Arten von Modulen können weitere Bibliothekmodule importieren
  - Danach können die Variablen und Funktionen des importierten Moduls verwendet werden

Programmstruktur – Beispiel

Verwendung einer Bibliothek mit rekursiven Funktionen

library.xq*m*

```
xquery version "1.0";
module namespace rec = "http://www.fh-ooe.at/xquery/recursion";

declare function rec:factorial($x as xs:integer)
as xs:integer
{
  if ($x <= 0) then
    0
  else if ($x = 1) then
    1
  else
    $x * rec:factorial($x - 1)
};
```

```
xquery version "1.0";
import module namespace r = "http://www.fh-
ooe.at/xquery/recursion" at "library.xqm";

for $i in -10 to 4
let $fac := r:factorial($i)
return
  if ($i >= 0) then
    <val i='{$i}' fac='{$fac}'/>
  else
    ()
```

main.xq

```
<val i="0" fac="0"/>
<val i="1" fac="1"/>
<val i="2" fac="2"/>
<val i="3" fac="6"/>
<val i="4" fac="24"/>
```

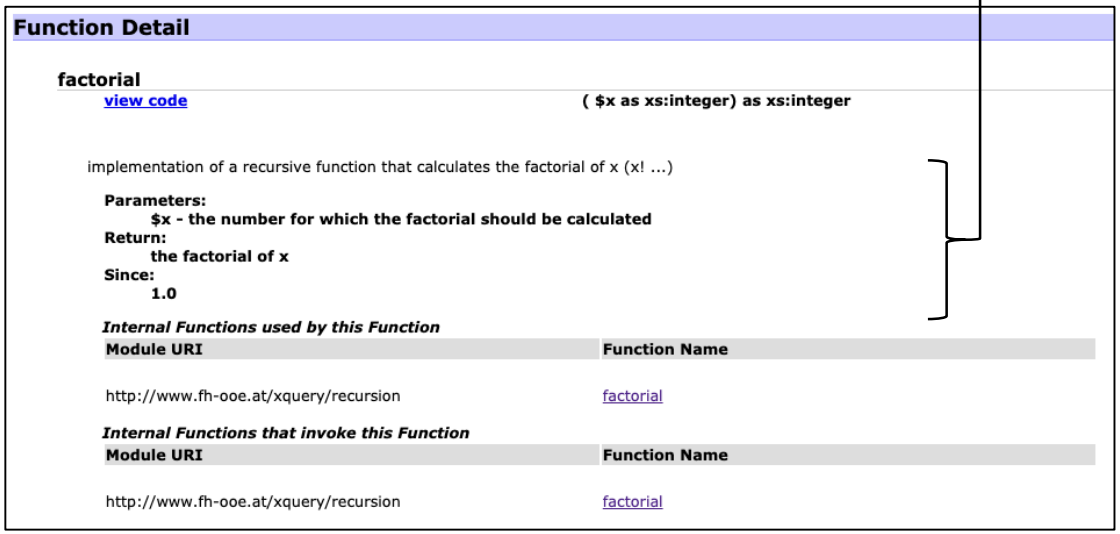
Programmstruktur – Dokumentation

Verwendung von xqDoc

library.xq*m*

```
(:~
 : implementation of a recursive function that calculates the factorial of x
 : (x! ...)
 : @param $x - the number for which the factorial should be calculated
 : @return the factorial of x
 : @author Instructor Name - Pxy0000
 : @since 1.0
:)
declare function rec:factorial($x as xs:integer)
```

- http://xqdoc.org/
- Tool zur Erstellung von lesbarer Dokumentation
- Hersteller-unabhängig
- Teilweise in XML-IDEs integriert



Function Detail

factorial

view code ($x as xs:integer) as xs:integer

implementation of a recursive function that calculates the factorial of x (x! ...)

Parameters:
$x - the number for which the factorial should be calculated
Return:
the factorial of x
Since:
1.0

Internal Functions used by this Function

Module URI	Function Name
http://www.fh-ooe.at/xquery/recursion	factorial

Internal Functions that invoke this Function

Module URI	Function Name
http://www.fh-ooe.at/xquery/recursion	factorial

Inhalt

Anhang

XQuery Update Facility 1.0

W3C Recommendation, 17.3.2011, http://www.w3.org/TR/xquery-update-10/

Exkurs: XQuery Update Facility 1.0

Instructors.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Instructors>
  <Instructor instructorNumber="p20621" type="HBL">Josef Altmann</Instructor>
  <Instructor instructorNumber="p22080" type="NBL">Julian Haslinger</Instructor>
  <Instructor instructorNumber="p20622" type="NBL">Norbert Niklas</Instructor>
  <Instructor instructorNumber="p20623" type="HBL">Barbara Traxler</Instructor>
</Instructors>
```

https://www.w3.org/TR/xquery-update-10/

XQuery Update Facility 3.0: https://www.w3.org/TR/xquery-update-30/

XQuery Update Facility 1.0

- Löschen
  - Knoten, der durch den Ausdruck selektiert wird, wird gelöscht

```
delete node doc("Instructors.xml")//Instructor[@instructorNumber = 'p22080']
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Instructors>
  <Instructor instructorNumber="p20621" type="HBL">Josef Altmann</Instructor>
  <Instructor instructorNumber="p22080" type="NBL">Julian Haslinger</Instructor>
  <Instructor instructorNumber="p20622" type="NBL">Norbert Niklas</Instructor>
  <Instructor instructorNumber="p20623" type="HBL">Barbara Traxler</Instructor>
</Instructors>
```

XQuery Update Facility 1.0

- Einfügen

```
insert node
  <Instructor>Julian Haslinger</Instructor>
before
  doc("Instructors.xml")//Instructor[1]
```

Mögliche Positionen
before / after / into
as first / as last

```
<?xml version="1.0" encoding="UTF-8"?>
<Instructors>
  <Instructor>Julian Haslinger</Instructor>
  <Instructor instructorNumber="p20621" type="HBL">Josef Altmann</Instructor>
  <Instructor instructorNumber="p20622" type="NBL">Norbert Niklas</Instructor>
  <Instructor instructorNumber="p20623" type="HBL">Barbara Traxler</Instructor>
</Instructors>
```

XQuery Update Facility 1.0

- Ersetzen

```
replace value of node doc("Instructors.xml")//Instructor[1] with "Julian-Paul
Haslinger"
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Instructors>
  <Instructor>Julian-Paul Haslinger</Instructor>
  <Instructor instructorNumber="p20621" type="HBL">Josef Altmann</Instructor>
  <Instructor instructorNumber="p20622" type="NBL">Norbert Niklas</Instructor>
  <Instructor instructorNumber="p20623" type="HBL">Barbara Traxler</Instructor>
</Instructors>
```

XQuery Update Facility 1.0

- Umbenennen

```
rename node doc("InstructorsXQU.xml")//Instructor[1] as "LVALeiter"
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Instructors>
  <LVALeiter>Julian-Paul Haslinger</LVALeiter>
  <Instructor instructorNumber="p20621" type="HBL">Josef Altmann</Instructor>
  <Instructor instructorNumber="p20622" type="NBL">Norbert Niklas</Instructor>
  <Instructor instructorNumber="p20623" type="HBL">Barbara Traxler</Instructor>
</Instructors>
```

XQuery Update Facility 1.0

- Transformation (copy, modify, return)
  - Kopiere Original-Dokument, lösche mehrere LVA-Leiter aus der Kopie und gib beide Dokumente zurück

```
for $original in doc("InstructorsXQU.xml")
return
  copy $copy := $original
  modify delete nodes ($copy//Instructor[1], $copy//Instructor[2])
  return
    ($copy, $original)
```

```
<Instructors>
   <Instructor instructorNumber="p20622" type="NBL">Norbert Niklas</Instructor>
   <Instructor instructorNumber="p20623" type="HBL">Barbara Traxler</Instructor>
</Instructors>
<Instructors>
   <Instructor instructorNumber="p20621" type="HBL">Josef Altmann</Instructor>
   <Instructor instructorNumber="p22080" type="NBL">Julian Haslinger</Instructor>
   <Instructor instructorNumber="p20622" type="NBL">Norbert Niklas</Instructor>
   <Instructor instructorNumber="p20623" type="HBL">Barbara Traxler</Instructor>
</Instructors>
```