Y36XML – Technologie XML

Přednáší:

Irena Mlýnková (mlynkova@ksi.mff.cuni.cz)

Martin Nečaský (necasky@ksi.mff.cuni.cz)

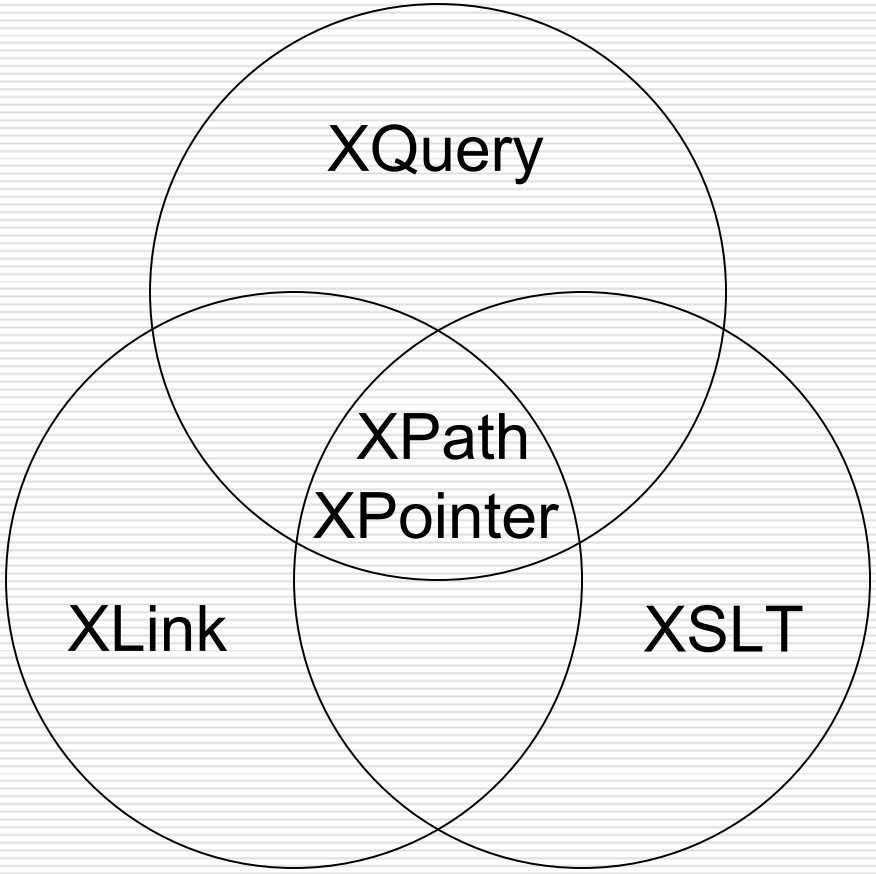
ZS 2009

Stránka přednášky:
http://www.ksi.mff.cuni.cz/~mlynkova/Y36XML/

Osnova předmětu

- Úvod do principů formátu XML, přehled XML technologií, jazyk DTD
- Datové modely XML, rozhraní DOM a SAX
- Úvod do jazyka XPath
- Úvod do jazyka XSLT
- XPath 2.0, XSLT 2.0
- Úvod do jazyka XML Schema
- Pokročilé rysy jazyka XML Schema
- Přehled standardních XML formátů
- Úvod do jazyka XQuery
- Pokročilé rysy jazyka XQuery, XQuery Update
- Úvod do XML databází, nativní XML databáze, číslovací schémata, structural join
- Relační databáze s XML rozšířením, SQL/XML

Vztahy mezi XML jazyky



Dotazovací jazyky nad XML daty

- Cíle: dotazování, pohledy, transformace, případně aktualizace XML dat
  - Od r. 1998 XML-QL, XQL, …
  - Vývoj v konsorciu W3C se ustálil/pokračuje v jazycích XSLT 1.0, XSLT 2.0, XPath 1.0, XPath 2.0, XQuery 1.0
    - XSLT je jazyk pro transformace, využívá XPath, zápis transformací hodně využívá XML
    - XQuery vhodnější pro dotazování – uživatelsky orientovaná syntaxe

Pz.: XPath 2.0 $\subset$ XQuery

Základem je XPath

- Základním stavebním kamenem dotazovacích jazyků nad XML daty
- Výběr částí XML dokumentů
- Podrobně viz. přednáška o XPath

XML dotazovací jazyky (1)

- Zpřístupnění potenciálně rozsáhlých dat nezávisle na jejich skutečné reprezentaci
- Ideální dotazovací jazyk by šlo použít na dotazování do
  - nestrukturovaných dat (text)
  - semistrukturovaných dat (web, xml)
  - silně strukturovaných dat (RDBMS)
  - objektových dat

XML dotazovací jazyky (2)

- Takový jazyk by byl příliš komplikovaný

→ specializace:

  - řetězcové masky, regulární výrazy
  - SQL
  - XPath, XQuery
  - OQL

XML dotazovací jazyky (3)

- Historické
  - XML-QL, Lorel, XQL, Quilt, ...
- Běžně používané
  - XPath 1.0
- Nastupující
  - XPath 2.0, XQuery 1.0
  - i XSLT je často používán pro dotazování

XQuery

- Aktuálně XQuery 1.0
- Stejný datový model s XPath 2.0
- XQuery 1.0 je nadmnožinou XPath 2.0
  - Každý dotaz v XPath 2.0 je i dotaz v XQuery 1.0
- XPath 1.0 a XPath 2.0 (a tedy i XQuery 1.0) nejsou vzájemně zcela kompatibilní
  - Ddatový model XPath 1.0 není kompatibilní s modelem XML Infoset

XQuery

- Větší vyjadřovací síla než XPath 1.0, XQL, atd.
- Čistší sémantika (XQuery Core model)
- Využití XML Schema
  - popis struktury
  - datové typy
- Kompatibilita datového modelu s XML Infoset
- Přístup založen na příkladech použití

XQuery

```
<?xml version="1.0"?>
<katalog>
  <kniha rok=”2002”>
    <titul>Šéfkuchař bez čepice</titul>
    <autor>Jamie Oliver</autor>
    <isbn>80-968347-4-6</isbn>
    <kategorie>kuchařky</kategorie>
    <stran>250</stran>
  </kniha>
  <kniha rok=”2007”>
    <titul>Modrá, nikoli zelená planeta</titul>
    <podtitul>Co je ohroženo: klima nebo svoboda?</podtitul>
    <autor>Václav Klaus</autor>
    <isbn>978-80-7363-152-9</isbn>
    <kategorie>společnost</kategorie>
    <kategorie>ekologie</kategorie>
    <stran>176</stran>
  </kniha>
```

XQuery

```
<kniha rok=”2006”>
  <titul>Jamie po italsku</titul>
  <original>
    <titul>Jamie's Italy</titul>
    <preklad>Vladimir Fuksa</preklad>
  </original>
  <autor>Jamie Oliver</autor>
  <isbn>80-89189-18-0</isbn>
  <kategorie>kuchařky</kategorie>
  <stran>319</stran>
</kniha>
```

XQuery

```
  <kniha rok="2007">
    <titul>Nepříjemná pravda</titul>
    <podtitul>Naše planeta v ohrožení – globální oteplování a
   co s ním můžeme udělat</podtitul>
    <original>
      <titul>An incovenient Truth</titul>
      <preklad>Jitka Fialová</preklad>
    </original>
    <autor>Al Gore</autor>
    <isbn>978-80-7203-868-8</isbn>
    <kategorie>ekologie</kategorie>
    <stran>329</stran>
  </kniha>

<katalog>
```

XQuery

- XQuery je funkcionální jazyk
  - Dotaz je výrazem
  - Výrazy lze libovolně kombinovat
- Forma dotazu v XQuery:
  - Deklarace jmenných prostorů (nepovinné)
  - Definice funkcí (nepovinné)
  - Vlastní výraz dotazu

XQuery

- XPath výrazy
  - `//katalog/kniha[autor="Jamie Oliver"]`
- Konstruktory
  - `element kniha {element autor}`
- FLWOR výrazy
  - `FOR ... LET ... WHERE ... ORDER BY ... RETURN`
- Podmíněné výrazy
  - `IF ... THEN ... ELSE`

XQuery

- Kvantifikátory
  - `EVERY var IN expr SATISFIES expr`
  - `SOME var IN expr SATISFIES expr`
- Operace s typy
  - `TYPESWITCH typeexpr CASE ... DEFAULT`
- Operátory a funkce
  - `x + y, z = x, fce(x,y,z)`
- Proměnné a konstanty
  - `$x, "Novák", 256`
- Porovnávání

XQuery - Konstruktory

- Přímé konstruktory

```
<html>
  <body>
    <h1>Výpis z doc("katalog.xml")//kniha</h1>
    <h2>Titul: {doc("katalog.xml")//kniha[1]/titul}</h2>
    <h3>Podtitul: {doc("katalog.xml")//kniha[1]/podtitul}</h3>
    <h2>
      Titul: {fn:data(doc("katalog.xml")//kniha[2]/titul)}
    </h2>
    <h3>
      Podtitul: {fn:data(doc("katalog.xml")//kniha[2]/podtitul})
    </h3>
  </body>
</html>
```

XQuery - Konstruktory

- Počítané konstruktory

```
element html {
  element body {
    element h1 {"Výpis z doc("katalog.xml")//kniha"},
    element h2 {
      text{"Titul: "},
      {doc("katalog.xml")//kniha[1]/titul}
    },
    ...
  }
}
```

XQuery - konstruktory

- Výsledek

```
<html>
  <body>
    <h1>Výpis z doc(”katalog.xml”)//kniha</h1>
    <h2>Titul: <titul>Šéfkuchař bez čepice</titul></h2>
    <h3>Podtitul: </h3>
    <h2>Titul: Modrá, nikoli zelená planeta</h2>
    <h3>Podtitul: Co je ohroženo: klima nebo svoboda?</h3>
  </body>
</html>
```

XQuery - FLWOR

- Základní konstrukce jazyka XQuery
- Klauzule for (for $var in expr) (FLWOR)
  - Vyhodnocuje výraz expr jehož výsledkem je seznam n-tic
  - n-tice iterativně přiřazuje do proměnné $var
- Klauzule let (let $var := expr) (FLWOR)
  - vyhodnotí výraz expr a přiřadí výsledek do proměnné $var
- Klauzule where (where expr) (FLWOR)
  - filtr na jednotlivé n-tice z klauzule for

XQuery - FLWOR

- Klauzule order by (order by expr) (FLWOR)
  - třídí n-tice, které prošly filtrem klauzule where podle daného kritéria
- Klauzule return (return expr) (FLRWOR)
  - závěrečná klauzule, ve které je zkonstruován výsledek výrazu ze získaných n-tic

XQuery - FLWOR

- Pro každou knihu, která má více než 300 stran vypiš titul a autora seřazené podle roku vydání

```
for      $kniha in doc("knihy.xml")//kniha
where    $kniha/stran > 300
order by $kniha/@rok
return
  <kniha>
    {$kniha/titul},
    {$kniha/autor}
  </kniha>
```

XQuery - FLWOR

- Pro každou knihu, která má cizojazyčný originál vypiš orignální i český titul a autora

```
for       $kniha in doc("knihy.xml")//kniha
where     $kniha/originál
return
  <kniha>
    {$kniha/titul}
    <orignálnítitul>
      {data($kniha/originál/titul)}
    </originálnítitul>
    {$kniha/autor}
  </kniha>
```

XQuery - FLWOR

- FLWOR výrazy také umožňují výrazně transformovat původní strukturu dat, např.:
  - Převod do XHTML a dalších formátů
- XHTML tabulka knih
  - Přehazování předků a potomků (swap)
    - kniha / autor → autor / seznam knih
  - Seskupování (group by)
    - seskupení knih podle kategorie
  - Spojování XML dat z různých zdrojů (join)
    - knihy v katalogu doplníme o recenze z jiného zdroje

XQuery - FLWOR

- Vypiš HTML tabulku knih z kategorie kuchařek se sloupečky titul, autor a počet stran

```
<table>
  <tr><th>Titul</th><th>Autor</th><th>Stran</th></tr>
  {
    for       $kniha in doc("knihy.xml")//kniha
    where     $kniha/kategorie = "kuchařka"
    return
      <tr>
        <td>{data($kniha/titul)}</td>
        <td>{data($kniha/autor)}</td>
        <td>{data($kniha/stran)}</td>
      </tr>
  }
</table>
```

XQuery - FLWOR

- Pro každého autora vypiš seznam jeho knih

```
<autori>
{
  for    $jmeno in distinct-values(doc("knihy.xml")//autor)
    return
    <autor>
      <jmeno>{$jmeno}</jmeno>
      {
        for    $kniha in doc("knihy.xml")//kniha
        where  $kniha/autor = $jmeno
        return
          <kniha>{$kniha/titul}</kniha>
      }
    </autor>
}
</autori>
```

XQuery - FLWOR

- Rozstřiď knihy podle kategorií, pro každou kategorii vytvoř samostatný element s názvem v atributu

```
<seznam-kategorií>
{
  for    $kategorie in distinct-
   values(doc("knihy.xml")//kategorie)
  return
    <kategorie nazev="{$kategorie}">
      {
        for    $kniha in doc("knihy.xml")//kniha
        where  $kniha/kategorie = $kategorie
        return
          <kniha>{$kniha/titul}</kniha>
      }
    </kategorie>
}
</seznam-kategorií>
```

XQuery - FLWOR

- Ke každé knize připoj seznam prodaných kusů ze zdroje prodej.xml (vnitřní spojení)

```
<knihy>
{
  for     $kniha in doc("knihy.xml")//kniha,
          $prodej in doc("prodej.xml")//kniha
  where   $kniha/ISBN = $prodej/ISBN
  return
    <kniha>
      {$kniha/titul},
      {$kniha/autor},
      {$prodej/stav},
    </kniha>
}
</knihy>
```

XQuery - FLWOR

- Ke každé knize připoj recenze ze zdroje recenze.xml (vnější spojení)

```
<knihy>{
  for    $kniha in doc("knihy.xml")//kniha
  return
    <kniha>
      {$kniha/titul},
      {$kniha/autor},
      {
        for $recenze in doc("recenze.xml")//recenze
        where $recenze/ISBN = $kniha/ISBN
        return $recenze/text
      }
    </kniha>
}</knihy>
```

XQuery – Podmíněné výrazy

- Klauzule if (if expr)
  - Vyhodnocuje výraz expr jehož hodnotou je true nebo false
- Klauzule then (then expr)
- Klauzule else (else expr)

XQuery – Podmíněné výrazy

- Pro každou knihu vypiš její název a první kategorii, pokud patří i do dalších kategorií, nahraď je prázdným elementem <dalsi-kategorie/>

```
for    $kniha in doc("knihy.xml")//kniha
return
  <kniha>
    {$kniha/titul}
    {$kniha/kategorie[1]}
    {
      if (count($kniha/kategorie) > 1 )
      then return <dalsi-kategorie/>
    }
  </kniha>
```

XQuery – Podmíněné výrazy

- Pro každou knihu vypiš její název a první kategorii, pokud patří i do dalších kategorií, nahraď je prázdným elementem <dalsi-kategorie/>

```
for    $kniha in doc("knihy.xml")//kniha
return
  <kniha>
    {$kniha/titul}
    {$kniha/kategorie[1]}
    {
      if (count($kniha/kategorie > 1) )
      then return <dalsi-kategorie/>
    }
  </kniha>
```

XQuery – Kvantifikátory

- Klauzule every/some (every/some var in expr)
  - vhodnotí výraz expr a požaduje aby každá/nějaká n-tice ve výsledku splňovala podmínku
- Klauzule satisfies (satisfies expr)
  - expr je podmínka kvantifikátoru

XQuery – Kvantifikátory

- Autoři, kteří nepíší české knihy

```
for $autor in distinct-values(doc("katalog.xml")//autor)
where every $autorova-kniha in
        for    $kniha in $doc("knihy.xml")//kniha
        where  $kniha[autor = $autor/jmeno]
        return $kniha
      satisfies
        $autorova-kniha/original
return $autor
```

XQuery – Funkce

- Zabudované funkce
  - distinct, distinct-value, empty, name, ...
  - Agregační funkce max, min, avg, count, ...
  - Další: řetězcové, numerické
  - ... je jich hodně
  - namespace fn
- Uživatelsky definované funkce
  - Přímo pomocí syntaxe XQuery
  - I rekurzivní, typované
  - Podpora knihoven, rozšiřitelné

XQuery – Zabudované funkce

- Některé už jsme poznali:
  - uzel dokumentu podle daného uri:

    ```
    fn:doc($uri as xs:string?) as document-node()?
    ```

  - sekvence atomických hodnot ze sekvence položek

    ```
    fn:data($arg as item()*) as xs:anyAtomicType*
    ```

  - počet položek v sekvenci

    ```
    fn:count($arg as item()*) as xs:integer
    ```

  - odstranění duplicit (jen atomické hodnoty)

    ```
    fn:distinct-values($arg as xs:anyAtomicType*) as xs:anyAtomicType*
    ```

XQuery – Uživatelsky definované funkce

- Syntaxe

  ```
  define function name(parameters) as type
  ```

- Kde
  - *name* je jméno funkce
  - *parameters* je seznam parametrů (typovaných i netypovaných)
  - *type* je typ návratové hodnoty funkce

XQuery – Uživatelsky definované funkce

- Funkce vracející názvy knih od daného autora (podle jména a příjmení, jedna kniha může mít i více autorů), seřazené podle názvu

```
module "http://ksi.mff.cuni.cz/xquery/knihy"

define function knihy-autora($jmeno, $prijmeni) as element()*
[
  for $kniha in doc("knihy.xml")//kniha
  where some      $autor in $kniha/autor
        satisfies $autor/prijmeni = $prijmeni and
                  $autor/jmeno = $jmeno
  order by $kniha/nazev
  return $kniha/nazev
]
```

XQuery – Uživatelsky definované funkce

- Import knihovny s přiřazením prefixu určitého prostoru jmen

```
import module namespace ksi =
     "http://ksi.mff.cuni.cz/xquery/knihy"
     at "file://home/novak/xquery/lib/knihy.xq"

<autor>
  <jmeno>Jan</jmeno>
  <prijmeni>Novák</prijmeni>
  <publikace>
    {ksi:knihy-autora("Jan", "Novák")}
  </publikace>
</autor>
```

XQuery – Uživatelsky Definované Funkce

- Funkce procházející strukturu knihy (sekce – rekurzivně) a počítající počet podsekcí dané sekce

```
module "http://ksi.mff.cuni.cz/xquery/knihy"

define function podsekce($kniha-or-sekce) as element()*
[
  for $podsekce in $kniha-or-sekce/sekce
  return
    <sekce>
      {$podsekce/nazev}
      <pocet>{fn:count($podsekce/sekce}</pocet>
      <podsekce>{podsekce($podsekce)}</podsekce>
    </sekce>
]
```

XQuery – Uživatelsky Definované Funkce

- Import knihovny s přiřazením prefixu určitého prostoru jmen

```
import module namespace ksi =
     "http://ksi.mff.cuni.cz/xquery/knihy"
     at "file://home/novak/xquery/lib/knihy.xq"

for $kniha in fn:doc("katalog.xml")//kniha
return
    <kniha>
      {$kniha/nazev}
      <pocet>{fn:count($kniha/sekce)}</pocet>
      {ksi:podsekce($kniha)}
    </kniha>
```

XQuery – Porovnání
Hodnotová

- Operátory
  - lt, gt, le, ge, eq, ne ve významu “menší”, “větší”, “menší rovno”, “větší rovno”, “rovno”, “nerovno”
- Postup porovnání operandů
  - Atomizace
  - Implicitní konverze na stejný datový typ
  - Porovnání upravených operandů

XQuery – Porovnání
Hodnotová

- Netypové operandy jsou implicitně přetypovány na řetězce
- Pokud je některý z operandů převeden na prázdnou sekvenci je výsledkem porovnání prázdná sekvence
- Pokud je některý z operandů převeden na sekvenci delší než 1 je vyvolána chyba

XQuery – Porovnání
Hodnotová

- `1 le 2 => true`
- `(1) le (2) => true`
- `(1) le (2,1) => chyba`
- `(1) le () => ()`
- `<a>5</a> eq <b>5</b> => true`
- `$kniha/autor eq "Jamie Oliver"`

  => `true` pouze pokud $book má právě jeden podelement autor s hodnotou "Jamie Oliver"

XQuery – Porovnání
Obecná

- Operátory
  - <, >, <=, >=, =, !=
- I na sekvence
- Postup porovnání operandů
  - Atomizace
    - Vzniknou sekvence atomických hodnot
  - Hledá se položka z levého operandu a položka z pravého operandu, které nabývají pro operátor hodnotu true
    - Pokud existuje, pak true
    - Pokud neexistuje, pak false

XQuery – Porovnání
Obecná

- Při hledání páru položek opět konverze
  - Obě netypové – konverze na xs:string
  - Jedna netypová, druhá numerická – konverze na xs:double
  - Jedna netypová, druhá typovaná ale ne řetězcová ani numerická – převod na tento typ

XQuery – Porovnání
Obecná

- `1 < 2` `=> true`
- `(1) < (2)` `=> true`
- `(1) < (2,1)` `=> true`
- `(1) < ()` `=> false`
- `(0,1) = (1,2)` `=> true`
- `(0,1) != (1,2)` `=> true`
- `$kniha/autor = "Jamie Oliver"`
  `=> true` pokud $book má nějaký podelement autor s hodnotou "Jamie Oliver"

XQuery – Porovnání
Uzlová

- Operátory
  - is, << a >>
- Postup porovnání operandů
  - Vyhodnocení operandů
  - Pokud je některý z operandů prázdná sekvence je výsledkem porovnání prázdná sekvence
  - Pokud je některý z operandů sekvence s délkou větší než 1 je vyvolána chyba

XQuery – Porovnání
Uzlová

- is je true, pokud oba operandy jsou uzly se stejnou identitou
- << je true, pokud levý operand předchází pravý operand (podle pořadí dokumentu)
- >> je true, pokud levý operand následuje pravý operand (podle pořadí dokumentu)

XQuery – Porovnání
Uzlová

```
/katalog/kniha[isbn="80-968347-4-6"]
                    is
/katalog/kniha[titul="Jamie Oliver"]
```

- `true`, pouze pokud se oba operandy vyhodnotí na ten samý uzel

XQuery – Porovnání
Uzlová

- Uvažujte program konference. Napište dotaz, který pro každý dotaz vrátí přednášky, které se konají první den před první přestávkou na kávu.

```
let $program-dne := doc("program.xml")/program/den[1]
let $ranni-kavicka := $program-dne/prestavka[@type="coffee"][1]
for $prednaska in $program-dne/prednaska
where $prednaska << $ranni-kavicka
return $prednaska
```

XQuery – Integritní omezení

- XML Schema poskytuje nástroje pro specifikaci různých integritních omezení
  - např. kardinality, klíče, datové typy, ...
- Neposkytuje ale vhodné nástroje pro specifikaci složitějších IO
  - např. “Pokud autor píše v cizím jazyce musí každá jeho kniha obsahovat i název v tomto jazyce a jméno překladatele do češtiny”

XQuery – Integritní omezení

- IO jsou v XML datech v určitých případech stejně i více důležitá než v RDBMS
  - Při integraci dat z různých externích zdrojů je potřeba kontrolovat velké množství různých IO
  - Při řízení toku dat v rámci organizace i mezi různými organizacemi
  - Taková IO vycházejí z podnikové logiky, mohou být poměrně složitá a týkají se často různých zdrojů/XML dokumentů
  - Důležitost poroste (SOA, webové služby, ...)

XQuery – Integritní omezení

- XQuery je dostatečně silný jazyk pro specifikaci IO
  - paralela CHECK v SQL
- Kontrola je vlastně speciální dotaz vracející hlášení o kontrole jako XML data, např.:
  - Pokud jsou data O.K.
    - `<ok no=”cislo IO”/>`
  - Pokud data porušují integritní omezení
    - `<error no=”cislo IO”>Hlášení o chybě</error>`

XQuery – Integritní omezení

- IO(1001): Pokud autor píše v cizím jazyce musí každá jeho kniha obsahovat i název v tomto jazyce a jméno překladatele do češtiny

```
let $autori := doc("autori.xml")//autor[jazyk != "cs"]
return
  {
    if every $autor in $autori
       satisfies every $autorova-kniha in
                          for    $kniha in
   $doc("knihy.xml")//kniha
                          where  $kniha[autor = $autor/jmeno]
                          return $kniha
                satisfies
                          $autorova-kniha/original
    then return <ok no="1001" />
    else return <error no="1001" />
  }
```

XQuery – Integritní omezení

```
let $autori := doc("autori.xml")//autor[jazyk != "cs"]
let $spatniautori
            := for $autor in $autori
               where some $autorova-kniha in
                           for    $kniha in
   $doc("knihy.xml")//kniha
                           where  $kniha[autor = $autor/jmeno]
                           return $kniha
                     satisfies
                           count($autorova-kniha/original) = 0
               return $autor
return {
  if exists($spatniautori)
  then return <error no="1001">
                Autor {data($autor)} má knihu bez původního
                názvu a překladatele!
              </error>
  else return <ok no="1001" />}
```

XQuery – Integritní omezení

```
let $spatniautori :=
  for $autor in doc("autori.xml")//autor[jazyk != "cs"]
  let $autorovyspatneknihy :=
    for    $kniha in $doc("knihy.xml")//kniha
    where  $kniha[autor=$autor/jmeno] and count($kniha/original)=0
    return $kniha/nazev
  where count($autorovyspatneknihy)>0
  return <autor>{$autor/jmeno}
           <knihy>{$autorovyspatneknihy}</knihy></autor>
return {
  if exists($spatniautori)
  then return <error no="1001">{
                for $autor in $spatniautori
                return <suberror>Autor {$autor/jmeno} nemá u knih
                         {for $nazev in $autor/knihy/nazev
                          return {$nazev},} uveden původní název a
                         překladatele.</suberror>
              }</error>
  else return <ok no="1001" />}
```

XQuery – Integritní omezení

- Upravená hláška o chybě:

```
<suberror>
  Autor {$autor/jmeno} nemá u knih
  {
    for $nazev in $autor/knihy/nazev[position()<last()]
    return {$nazev},
  } a {
    $autor/knihy/nazev[last()]
  }
  uveden původní název a překladatele.
</suberror>
```

XQuery – podpora schémat

- Podpora schémat je významným přínosem oproti ostatním XML dotazovacím jazykům
  - XQuery musí být schopno pracovat s dokumenty bez známé struktury
  - XQuery musí využívat vlastnosti schématu, je-li známé (rozšířená implementace)
  - Implementace může umožňovat statické typování
- Typový systém založen na XML Schema

XQuery – formální sémantika

- XQuery obsahuje velké množství redundancí
- XQuery Core definuje syntaktickou podmnožinu jazyka XQuery, která má stejnou vyjadřovací sílu jako původní jazyk
  - Součástí definice XQuery jsou i přepisovací pravidla do XQuery Core
- XQuery Core má význam především z teoretického hlediska, příliš se nehodí k optimalizaci dotazů

Konec