

Dotazování XML dat

Michal Krátký, Radim Bača

Database Research Group
Katedra informatiky
VŠB-Technická univerzita Ostrava

DATAKON 2010, Mikulov, 17.10.2010

Obsah

- ▶ Úvod
- ▶ Základní pojmy
- ▶ Dotazovací jazyky pro XML data
- ▶ Metody vykonávání XQuery dotazů
- ▶ Testování XML databázových systémů

Úvod a základní pojmy

- ▶ Jazyk XML je vývojáři často chápán jako jazyk pro psaní jednoduchých konfiguračních souborů a přenos dat mezi aplikacemi.
- ▶ Můžeme jej ovšem použít pro uložení a dotazování velkých objemů slabě strukturovaných dat.
- ▶ XML určeno pro uložení a dotazování slabě strukturovaných dat případně dat kde se často mění struktura (schéma). ⇒ Pro x% aplikací nemá tedy smysl uvažovat o migraci z relačního datového modelu.
- ▶ Budeme prezentovat možnosti a slabé stránky dnešních nativních XML databází. Zaměříme se také na základní metody používané při vykonávání XQuery dotazů.

XML dokument, příklad

```
<?xml version="1.0" ?>  
<books><book id="001-00863">  
  <title>XML Data Management</title>  
  <author>  
    <first>Akmal</first>  
    <last>Chaudhri</last>  
  </author>  
  <price>59.00</price>  
</book></books>
```

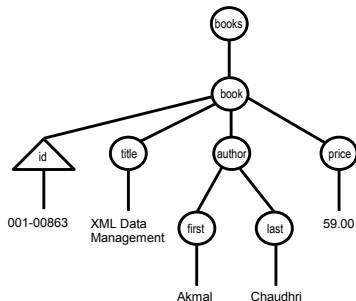
- ▶ Na příkladu vidíme XML dokument reprezentující knihy a jejich autory.
- ▶ *Dobře strukturovaný dokument* (well-formed) - splňuje základní syntaktická pravidla XML dokumentu¹:
k počáteční značce existuje koncová značka, ...

¹<http://www.w3.org/TR/REC-xml>

Datový model

XML dokument modelujeme nejčastěji jako *XML strom* (obecně se může jednat o graf).

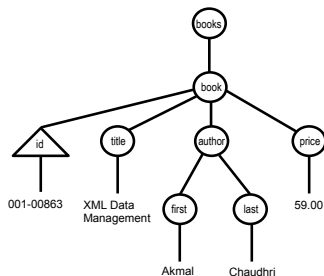
```
<?xml version="1.0" ?>
<books><book id="001-00863">
  <title>XML Data Management</title>
  <author>
    <first>Akmal</first>
    <last>Chaudhri</last>
  </author>
  <price>59.00</price>
</book></books>
```



- ▶ Elementy a atributy modelujeme jako uzly. Vztah element-podelement (tzv. rodič-dítě) modelujeme hranou mezi příslušnými uzly.

Uspořádání dokumentu

```
<?xml version="1.0" ?>
<books><book id="001-00863">
  <title>XML Data Management</title>
  <author>
    <first>Akmal</first>
    <last>Chaudhri</last>
  </author>
  <price>59.00</price>
</book></books>
```



- ▶ V případě XML rozlišujeme pořadí elementů, tzv. *uspořádání dokumentu* (document ordering), což je podstatný rozdíl oproti relačnímu datovému modelu.
- ▶ Uspořádání je ekvivalentní s *dopředným uspořádáním* (angl. preorder) elementů v XML dokumentu.

Příklad XML dokumentu

```
<?xml version="1.0" ?>
<books>
  <book id="003-04312">
    <title>XQuery from the Experts</title >
    <author><first >Jonathan</ first ><last >Robie</ last ></author>
    <author><first >Michael</ first ><last >Rys</ last ></author>
    <price >49.00</ price >
  </book>
  <book id="001-00863">
    <title >XML Data Management</ title >
    <author><first >Akmal</ first ><last >Chaudhri</ last ></author>
    <price >59.00</ price >
  </book>
  <book id="045-00012">
    <title >XML Indexing and Pattern Matching</ title >
    <author><first >Bongki</ first ><last >Moon</ last ></author>
    <price >79.99</ price >
  </book>
</books>
```

Příklad, XML strom

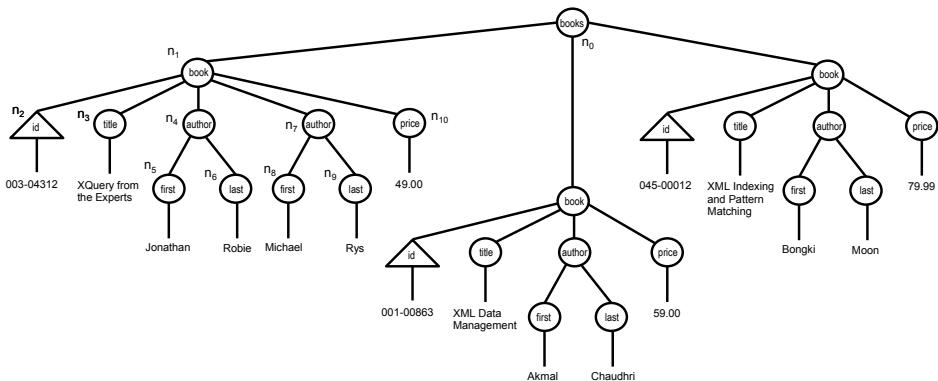


Schéma XML dokumentu

- ▶ Stejně jako v případě relačního modelu i zde můžeme mluvit o schématu, tentokrát o schématu XML dokumentu.
- ▶ Schéma tedy popisuje strukturu XML dokumentu (dětské elementy, atributy, datové typy, integritní omezení).
- ▶ Dokument, který splňuje definované schéma, označujeme jako *validní*.
- ▶ Mezi nejpoužívanější jazyky pro popis schématu patří:
 - ▶ DTD²
 - ▶ XML Schema³

²<http://www.w3.org/TR/REC-xml>

³<http://www.w3.org/XML/Schema>

Schéma XML dokumentu - DTD⁴

```
<!DOCTYPE books [  
  <!ELEMENT books(book*)>  
  <!ELEMENT book(title , author* , price)>  
  <!ATTLIST book id CDATA #REQUIRED>  
  <!ELEMENT title(#PCDATA)>  
  <!ELEMENT author(first , last)>  
  <!ELEMENT price(#PCDATA)>  
  <!ELEMENT first(#PCDATA)>  
  <!ELEMENT last(#PCDATA)>  

```

- ▶ DTD používá vlastní (poměrně úspornou) syntaxi.
- ▶ Výrazové prostředky jsou spíše omezené (např. chybí datové typy, definice počtu opakování podelementů atd.).
- ▶ I přes tyto nevýhody se jedná o nejrozšířenější způsob popisu schématu XML dokumentu.

⁴<http://www.w3.org/TR/REC-xml>

Schéma XML dokumentu - XML Schema⁵

- ▶ XML schema využívá XML syntaxi.
- ▶ XML schema nabízí celou řadu možností, které nejsou v DTD k dispozici:
 - ▶ definování nových datových typů,
 - ▶ komplikovaná integritní omezení,
 - ▶ vzory,
 - ▶ atd.
- ▶ Platíme za to značnou upovídáností (XML schema nemusí být ovšem nutně dobře čitelné po člověka).

⁵<http://www.w3.org/XML/Schema>

Schéma XML dokumentu - XML Schema, příklad 1/2

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="books">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="book" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="title" type="xsd:string"/>
              <xsd:element name="price" type="xsd:integer"/>
              <xsd:element name="author" maxOccurs="unbounded">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="first" type="xsd:string"/>
                    <xsd:element name="last" type="xsd:string"/>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Schéma XML dokumentu - XML Schema, příklad 2/2

```
<xsd:attribute name="id" type="IdType" use="required"/>
<xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>

<xsd:simpleType name="IdType">
<xsd:restriction base="xsd:string">
<xsd:length value="9"/>
<xsd:pattern value="[0-1]-[0-1]"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:schema>
```

Dotazovací jazyky pro XML

- ▶ Dotazovací jazyk je úzce spjat se způsobem jakým datový model reprezentuje data.
- ▶ Dotazovací jazyky pro XML data musí být tedy odlišné od dotazovacího jazyka SQL.
- ▶ Dotazovací jazyky používají výrazy cest pro průchod XML stromem.
- ▶ Pro dotazování XML dat používáme dotazovací jazyky jako XPath nebo XQuery (podmnožinou jazyka je XPath).

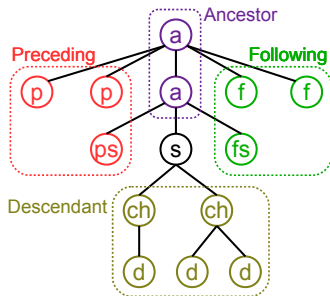
Dotazování a aktualizace dat

- ▶ Jazyk XPath umožňuje definovat výrazy cest a navíc umožňuje filtrovat uzly stromu pomocí relací mezi uzly, které se nazývají osy XPath.
- ▶ Skutečným dotazovacím jazykem využívajícím jazyk XPath je pak dotazovací jazyk XQuery.
- ▶ K aktualizaci dat je určena specifikace XQuery Update Facilities.
- ▶ K full-textovému vyhledávání slouží specifikace XQuery and XPath Full Text.

XPath

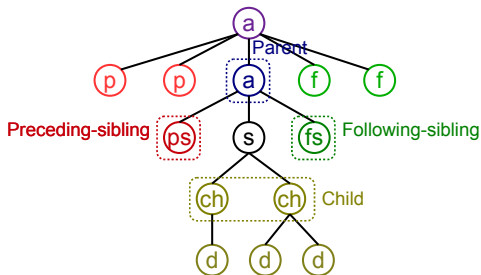
- ▶ Dotaz se skládá z výrazů `osa::znacka[podminka]` oddělených / nebo //.
- ▶ Výsledkem výrazu na kontextovém uzlu u je množina uzlů u' takových, že:
 - ▶ relace `osa` obsahuje (u, u') ,
 - ▶ značka uzlu u' je `znacka`,
 - ▶ `podminka` nabývá hodnoty `true`.

Osy XPath



| Osa | Popis |
|------------|---|
| ancestor | uzly ležící na cestě od kořene k uzlu u |
| descendant | všechny uzly pro které je u předek |
| following | uzly jenž následují uzel u (kromě potomků) |
| predecing | uzly jenž předcházejí uzel u (kromě předků) |

Osy XPath



| Osa | Popis |
|-------------------|--------------------------------------|
| parent | první uzel na cestě od u ke kořeni |
| child | přímý potomek uzlu u |
| predecing-sibling | předcházející sourozenci u |
| following-sibling | následující sourozenci u |

Osy XPath

| Osa | Popis |
|--------------------|--|
| ancestor-or-self | u a uzly ležící na cestě z u ke kořeni |
| descendant-or-self | descendant + u |
| attribute | atribut uzlu |
| self | |
| namespace | jmenný prostor |

Oddělovače výrazů:

- ▶ / – značí vztah rodič-dítě
- ▶ // – značí vztah předek-potomek

XPath, příklad

- ▶ Vrať všechny knihy:

```
//book
```

- ▶ Vrať knihu napsanou autorem s příjmením Rys:

```
/books/book[author/last='Rys']
```

- ▶ Vrať knihu, která se nachází za knihou napsanou autorem s příjmením Rys:

```
/books/book[author/last='Rys']/following::book
```

- ▶ Vrať id (hodnota atributu) první knihy:

```
/books/book[1]/@id
```

- ▶ Vrať prvního autora každé knihy:

```
/books/book/author[1]
```

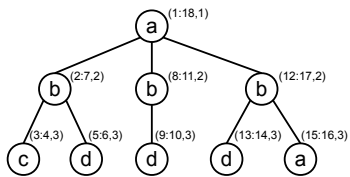
- ▶ Vrať prvního autora z celého dokumentu:

```
(/books/book/author)[1]
```

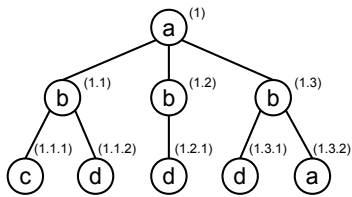
Číslovací schémata

- ▶ Číslovací schémata přiřazují jednotlivým uzlům celá čísla tak, abychom mohli rychle vyhodnotit vztah (osu XPath) mezi dvěma uzly, např.:
 - ▶ zda jeden předchází druhý,
 - ▶ zda jsou ve vztahu rodič-dítě nebo předek-potomek v XML stromu.
- ▶ V zásadě rozlišujeme dva typy číslovacích schémat:
 - ▶ číslovací schémata založená na elementech,
 - ▶ číslovací schémata založená na cestách.

Číslovací schémata, vlastnosti



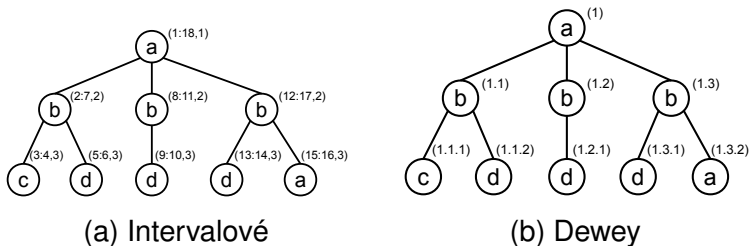
(a) Intervalové



(b) Dewey

- ▶ Číslovací schémata založená na elementech (např. intervalové):
 - ▶ (+) snadné uložení,
 - ▶ (-) nevhodné pro aktualizaci XML dokumentu.

Číslovací schémata, vlastnosti



- ▶ Číslovací schémata založená na cestách:
 - ▶ (+) ze značky přímo přečteme značky předků (můžeme urychlit vykonávání dotazu, např. TJFast [11] nebo TJDewey [4]),
 - ▶ (+) vhodné pro aktualizaci dat,
 - ▶ (-) komplikované uložení (značky mají různou délku).

XML vs relační datový model

- ▶ Datový model XML je bohatší a komplikovanější než známý relační datový model, využití relačního dotazovacího jazyka, jako je SQL, je komplikované.
- ▶ XML je modelován stromem, SQL používáme pro dotazování relací.
- ▶ Schéma XML dokumentu může obsahovat rekurzy (byť samotné dokumenty rekurzivní nejsou).
- ▶ Triviální implementace pomocí rozkladu dokumentu do relací musí nutně, pro velká data, vést k neúspěchu.

Triviální rozklad XML do relací

► Mějme schéma relace:

```
Uzel(id, atribut, nazev, hodnota)
```

Kde:

- `id` je jedinečné číslo uzlu (elementu nebo atributu) definované dle nějakého číslovacího schématu,
- `atribut` bude `true` pokud je uzel atribut,
- `nazev` je název (značka) elementu či atributu,
- `hodnota` je řetězcový obsah elementu nebo hodnota atributu.

Triviální rozklad XML do relací, dotaz

- ▶ Mějme XPath dotaz z kolekce XMARK⁶:

```
/site/closed_auctions/closed_auction/annotation/  
description/parlist/listitem/parlist/listitem/text/  
emph/keyword/
```

- ▶ SQL dotaz bude obsahovat 11 operací spojení.
- ▶ 11 operací spojení? Strašné, ale dotaz nemusí být nutně psán uživatelem.

⁶<http://monetdb.cwi.nl/xml/>, 111 MiB

Triviální rozklad XML do relací, rozklad řešení

- ▶ Druhá stránka problému je nevyhovující efektivita provádění dotazu.
 - ▶ ⇒ Velké mezivýsledky
 - ▶ ⇒ Velké časy vykonávání dotazů.
- ▶ Ačkoli je výsledkem dotazu v tomto případě 180 elementů, čas vykonání dotazu je 28s.

Triviální rozklad XML do relací, vylepšení

- ▶ Malého zlepšení dosáhneme přidáním atributu `rodic`.
`Uzel(id, atribut, rodic, nalez, hodnota)`

Kde:

- ▶ `rodic` je jedinečné číslo rodiče (uzlu, který je v XML stromu nad tímto uzlem směrem ke kořeni).
- ▶ Počet operací spojení zůstane, zmenšíme ale mezivýsledky.

Triviální rozklad XML do relací, rozklad řešení

- Výsledkem dotazu je v tomto případě 180 elementů, musíme ovšem zpracovat celkem 55 383 elementů:

| Step | Nodes | Useful |
|-----------------|---------------|------------|
| site | 1 | 1 |
| closed_auctions | 1 | 1 |
| closed_auction | 9,750 | 9,750 |
| annotation | 9,750 | 9,750 |
| description | 9,750 | 2,934 |
| parlist | 2,934 | 2,934 |
| listitem | 8,512 | 1,713 |
| parlist | 1,713 | 1,713 |
| listitem | 4,964 | 4,964 |
| text | 4,964 | 1,890 |
| emph | 2,864 | 173 |
| keyword | 180 | 180 |
| Sum | 55,383 | 36,003 |

- ⇒ Čas vykonání dotazu: 15s.

XQuery

- ▶ S pomocí XQuery můžeme při dotazování jednoho XML dokumentu vytvářet dokument jiný.
- ▶ Jednou z významných částí jazyka XQuery jsou tzv. FLWOR výrazy:
 - ▶ **F**, for klauzule: asociace jedné nebo více proměnných k výrazu,
 - ▶ **L**, let: přiřazení výsledku výrazu proměnné,
 - ▶ **W**, where: omezující podmínka,
 - ▶ **O**, order by: seřídění,
 - ▶ **R**, return: výsledek.

Příklady XQuery

Najděme názvy knih, které jsou dražší než 50\$.

```
for $b in doc('books.xml')//book  
where $b/price > 50.0  
return $b/title
```

Příklady XQuery

Najděme názvy knih, které napsal autor s příjmením Moon
setříděné dle názvu.

```
for $b in doc('books.xml')//book
where $b/author/last='Moon'
order by $b/title
return
  <result>
    { $b/title }
  </result>
```

Příklady XQuery

Najděme názvy knih, které mají více než jednoho autora.

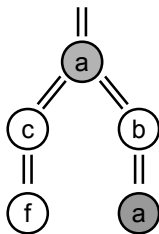
```
for $b in doc('books.xml')//book
let $c := $b/author
where count($c) > 1
return
  <result>
    { $b/title }
  </result>
```

Větvené dotazy (Twig Pattern Query)

- ▶ Mějme dotaz:

```
for $a in //a[.//c//f], $aa in $a//b//a  
return <o>$a,$aa</o>
```

- ▶ Twig pattern query:



XQuery Update Facility

- ▶ XQuery Update Facility (XUF) je rozšíření XQuery, které umožňuje specifikovat změny v původních datech.
- ▶ XUF neřeší zda-li je aktualizován pouze XML soubor či jsou-li změny provedeny v nějaké XML databázi.
- ▶ Specifikace neuvažuje transakce ani případné úrovně izolace.
- ▶ Vše je ponecháno na konkrétní implementaci.

XQuery Update Facility

- ▶ XUF definuje několik základních operací:
 - ▶ vkládání (insert),
 - ▶ mazání (delete),
 - ▶ přejmenování (rename),
 - ▶ nahrazení (replace),
 - ▶ transformaci (copy-modify-return).
- ▶ Operace se může vztahovat k jednomu nebo více uzlům.

XQuery Update Facility, Příklady

- ▶ Vložení podelementu `<year>2006</year>` do každého elementu `<book>`:

```
insert node <year>2006</year> into //book
```

- ▶ Zvýšení ceny první knihy o 10%:

```
replace value of node /books/book[1]/price  
with /books/book[1]/price * 1.1
```

XQuery Update Facility, Příklady

Vložení podelementu `<year>2006</year>` do každého elementu `<book>`:

```
for $book in //book  
return insert node <year>2006</year> into $book
```

- ▶ Tato varianta prvního příkladu ukazuje, že operace mohou být součástí XQuery výrazů.
- ▶ Příkaz je pak rozdělen na selektivní část a část, kde se specifikuje změna dat (operace XUF).
- ▶ Operace XUF se v XQuery příkazu nemohou objevit kdekoli: lze je uvést pouze za klíčovým slovem `return`.

XQuery Update Facility

- ▶ Všechny XUF operce jsou provedeny najednou (atomicky) až po provedení celého XQuery příkazu.
- ▶ I kdyby příkaz obsahoval více XUF operací, nelze o nich přemýšlet jako o transakci.
- ▶ Změny provedené jednou XUF operací jsou viditelné až po provedení celého příkazu.
- ▶ Důsledky:
 - ▶ Nezáleží na pořadí XUF operací v příkazu.
 - ▶ Nemůžeme počítat se změnami provedenými v jedné XUF operaci při psaní druhé.

XQuery Update Facility, konflikty

- ▶ Provedení všech XUF operací najednou může vést u některých operací k tzv. *konfliktům*.
- ▶ Jedná se o příkazy kdy by různé pořadí provádění XUF operací vedlo k různým výsledkům.
- ▶ V takovém případě by mělo dojít k vyvolání výjimky.

Příklad:

```
for $xmltitle in //book/title [contains(.,"XML")]  
return (  
  insert node <kw>XML<kw> after $xmltitle ,  
  delete node $xmltitle  
)
```

XQuery Update Facility, Copy-modify-return

- ▶ Všechny operace kromě `copy-modify-return` (CMR) provádějí změny přímo v datech.
- ▶ CMR vytváří kopii vybraných uzlů (operace **copy**).
- ▶ Provádí modifikaci s využitím dalších XUF operací (**modify**).
- ▶ Výsledek vrací jako výstup celé operace (**return**).
- ▶ CMR může provádět změny pouze na kopii zdrojových dat.
- ▶ Operace také jako jediná vrací po svém vykonání nějaký výstup.

XQuery Update Facility, Copy-modify-return příklad

Vložení podelementu `<year>2006</year>` do každého elementu `<book>`:

```
for $book in //book
return
  copy $bookcopy := $book
  modify insert node <year>2006</year> into $bookcopy
return $bookcopy
```

XQuery and XPath Full Text⁷

- ▶ XQuery and XPath Full Text rozšiřuje jazyky XPath a XQuery o podobnostní dotazování obsahu elementů a hodnot atributů.
- ▶ Následující příkaz vrátí autora všech knih s titulem obsahujícím slovo s kořenem dog a termem cat.

```
for $b in /books/book  
where $b/title contains text  
      ("dog" using stemming) ftand  
      "cat"  
return $b/author
```

⁷<http://www.w3.org/TR/xpath-full-text-10/>

XQuery and XPath Full Text

- ▶ Následující příkaz vrátí element title pokud bude obsahovat frázi "Web site Usability", element nebude vrácen ani v případě kdy obsahuje všechny termy, které ale nejsou uvedeny bezprostředně za sebou:

```
//book//title contains text "Web Site Usability"
```

Podpora dotazovacích jazyků

- ▶ Pokud se zaměříme na podporu popisovaných standardů ve stávajících databázových systémech, pak bezesporu nejvíce podporovanými jsou jazyky XQuery a XPath.
- ▶ I v tomto případě ovšem není jejich implementace dokonalá.
- ▶ Pro aktualizaci dat a podobnostní dotazování obsahu elementů a hodnot atributů používají databázové systémy často vlastní rozšíření dotazovacích jazyků.

Metody vykonávání XQuery dotazů

- ▶ Jednotlivé komponenty ovlivňující vykonávání XQuery dotazů:
 - ▶ Algebra XQuery dotazu.
 - ▶ Algoritmy, které se využívají pro vykonání jednotlivých operátorů (nejčastější operací jsou různé druhy spojení).
 - ▶ Datové struktury podporující použité algoritmy.

XQuery algebra

- ▶ Tři typy algeber:
 - ▶ pracující s uspořádanými posloupnostmi n-tic (NAL [5], Pathfinder [14]),
 - ▶ pracující s množinami uspořádaných stromů s podobnou strukturou (XAL [8]),
 - ▶ hybridní algebra, jež kombinuje oba předchozí přístupy (Galax [13]).
- ▶ Oproti relačním algebrám popisují navíc operátory které jsou specifické pro XQuery.
- ▶ Například navigace ve stromu.

XQuery algebra

- ▶ Algebry také nabízejí množinu přepisovacích pravidel, které umožňují dotaz zjednodušit či provést efektivněji.
- ▶ Přepisovací pravidla se zaměřují:
 - ▶ na přesunování operátorů v rámci výrazu,
 - ▶ na odstraňování vnořených výrazů (tzv. unnesting),
 - ▶ na výběr vhodných algoritmů.


Algoritmy spojení


- ▶ Nejvýznamnější operací při vykonávání dotazů je operace spojení.
- ▶ Proto je těmto operacím věnovaná ve výzkumu zvláštní pozornost.
- ▶ Ukázalo se, že spojovací algoritmy využívané v relačních databázích jsou nedostatečné.
- ▶ Jelikož se jedná o algoritmy, které provádějí navigaci v XML stromu.
- ▶ Většina dalších operací (zejména operace s hodnotami) není v zásadě odlišná od relačních databází.

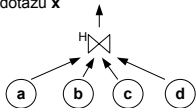
Operace spojení


- ▶ V zásadě rozlišujeme dva druhy spojení:
 - ▶ Binární strukturální spojení (např. [1]),
 - ▶ Holistické spojení (např. Holistic Twig Joins [6]).


Rozdíl ve způsobu provádění větveného dotazu $a[./b]//c/d$:

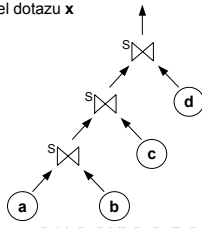
 - holistické spojení

 - množina uzlů XML stromu pro uzel dotazu x



 - strukturální spojení

 - množina uzlů XML stromu pro uzel dotazu x



Algoritmy spojení - binární

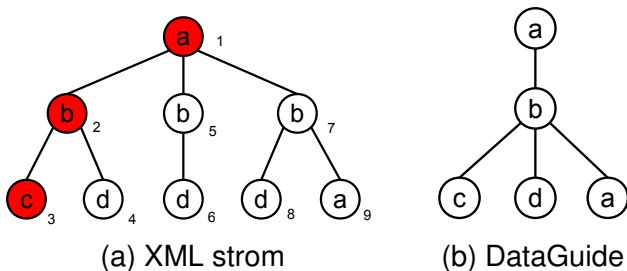
- ▶ Algoritmus řeší spojení mezi dvojicí XPath výrazů.
- ▶ Mezi takové algoritmy patří například XPath accelerator [7] nebo MPMGJN [1].
- ▶ Binární spojení odpovídá XPath modelu vykonávání dotazu:
 - ▶ ⇒ Lze je jednoduše zařadit do libovolné XQuery algebry.
 - ▶ ⇒ Také je možné vykonat jakoukoli osu XPath.
- ▶ Nevýhodou je, že mohou produkovat velké mezivýsledky.

Algoritmy spojení - holistické

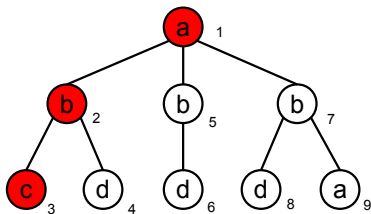
- ▶ Problém velkých mezivýsledků v některých případech řeší holistické spojovací algoritmy.
- ▶ Algoritmů je celá řada: TwigStack [6], TJFast [11], Twig²Stack, TJStrictPre, TJDewey [4], TwigStackSorting [3] atd.).
- ▶ Algoritmus vykonává dotaz tak, že bere v úvahu všechny XPath výrazy najednou.
- ▶ U některých typů dotazů dokáže garantovat lineární časovou a vstupně/výstupní složitost vzhledem k velikosti vstupu a výstupu.

Datové struktury, DataGuide

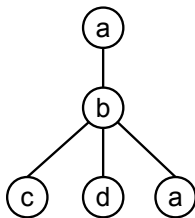
- ▶ Pro XML strom můžeme definovat strom schématu dokumentu.
- ▶ Tento strom je možné definovat více způsoby, nejznámější je DataGuide [12], který obsahuje jako cesty značkové cesty z XML dokumentu.



Datové struktury, DataGuide



(a) XML strom

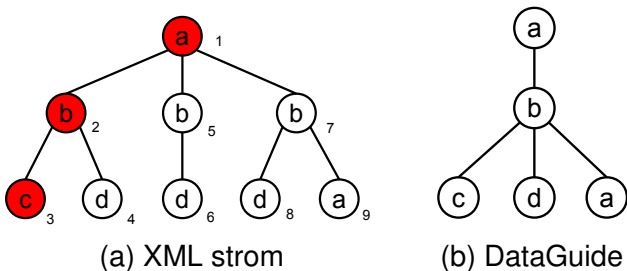


(b) DataGuide

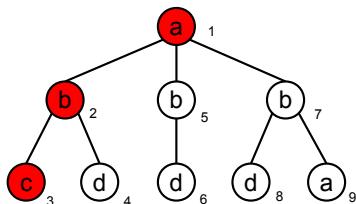
- ▶ Např. k uzlu 4 náleží značkováná cest a,b,d .
- ▶ Často mluvíme o rozkladu XML dokumentu.
- ▶ Značky stromu schématu XML dokumentu nám umožňují filtrovat nerelevantní uzly při vykonávání dotazu.

Datové struktury, rozklad dokumentu

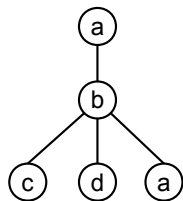
- ▶ Většina algoritmů využívá pro uložené XML dokumentu datovou strukturu invertovaný seznam kde:
 - ▶ klíčem je uzel stromu schématu dokumentu,
 - ▶ k danému klíči ukládáme čísla uzlu dle daného číslovacího schématu.



Datové struktury, rozklad dokumentu



(a) XML strom



(b) DataGuide

Rozklad dokumentu:

- ▶ Dle značky elementu (menší množství klíčů, více značek), např.: $b \rightarrow 2, 5, 7$
- ▶ Dle značkové cesty, např.: $a, b, a \rightarrow 9$
- ▶ Dle značky a úrovně elementu (větší množství klíčů, méně značek), např.: $(a, 1) \rightarrow 1$

Datové struktury, Invertovaný seznam

- ▶ Invertovaný seznam můžeme implementovat:
 - ▶ B-stromem.
 - ▶ Speciálními datovými strukturami (R-strom, XR-strom, XB-strom), které umožní přeskakovat nerelevantní uzly, např.: [9, 10].
- ▶ Tyto indexy je možné využít pro zefektivnění všech typů algoritmů spojení.

Datové struktury, rozklad dokumentu

- ▶ Pokud vezmeme v úvahu invertovaný seznam s menším počtem klíčů, např. klíčem je značka elementu, pak při provádění dotazu budeme:
 - ▶ načítat sekvenčním čtením větší počet diskových bloků obsahující značky uzlů (efektivní).
 - ▶ pracovat s větším počtem nerelevantních uzlů (neefektivní).
- ▶ Např. 1000 klíčů, každý obsahuje 1000 hodnot.

Datové struktury, rozklad dokumentu

- ▶ Pokud vezmeme v úvahu invertovaný seznam s větším počtem klíčů, např. klíčem je značkováná cesta, pak při provádění dotazu budeme:
 - ▶ načítat sekvenčním čtením menší počet diskových bloků obsahující značky uzlů a musíme použít více náhodných přístupů (neefektivní).
 - ▶ pracovat s menším počtem nerelevantních uzlů (efektivní).
- ▶ Např. 10000 klíčů, každý obsahuje 100 hodnot.

Triviální rozklad XML do relací, Vylepšení 2

- ▶ Jak můžeme obohatit naše schéma o značkové cesty?
- ▶ Přidáme atribut `lp` obsahující značkovanou cestu uzlu.
`Uzel(id, atribut, rodic, nazev, lp, hodnota)`

- ▶ Mějme dotaz:

```
/site/closed_auctions/closed_auction/  
  annotation/description/parlist/listitem/  
  parlist/listitem/text/emph/keyword/
```

- ▶ Pokud bude atribut `lp` indexován, dotaz vykonáme s logaritmickeou časovou složitostí s počtem uzlů.
- ▶ Čas vykonání dotazu bude 0.5 s (původní čas byl 28 s).

Co s dalšími dotazy?

- ▶ Co s dotazy jako:

```
/site/closed_auctions/*/emph/keyword/
```

nebo

```
//closed_auctions/*/emph/keyword/
```

- ▶ Musím použít komplikovanější metody pro vyhledávání relevantních značkových cest (B-strom umožní logaritmičticky vyhledat pouze pravé rozšíření).

Co s dalšími dotazy?

- ▶ Co s dotazy jako:

```
/site/regions/africa[item/location='United']  
/country/name/
```

- ▶ Musím vyhodnotit obě větve a pak provést binární spojení.

Co s dalšími dotazy?

- ▶ Co s dotazy, které obsahují další osy XPath:

```
for $x in //asia//item[  
    .//mail/following-sibling::mail]/name  
return <out> {$x} </out>
```

- ▶ Musím vytvořit takový plán vykonávání dotazu, který bude obsahovat různé operace spojení:
 - ▶ Část dotazu můžeme vykonat holistickým spojením nad invertovaným seznamem.
 - ▶ Část dotazu můžeme vykonat binárním spojením nad invertovaným seznamem.
 - ▶ Část dotazu můžeme vykonat nad perzistentním DOMem.

Co s dalšími dotazy?

- ▶ Potřebujeme metody založené na ceně jednotlivých operací.
- ▶ V práci [2] jsme uvedli nový typy spojení (progresivní), které můžeme použít v případech, kdy dotaz obsahuje větve s vysokou selektivitou (především větve s hodnotami).

XML kolekce

- ▶ Budeme testovat několik databází, které implementují podporu XQuery.
- ▶ Dvě velmi odlišné XML kolekce: XMARK⁸ (1.1 GB) a TreeBank⁹ (86 MB).

| Kolece | Počet uzlů | Počet značkových cest | Počet značek | Maximální hloubka |
|----------|------------|-----------------------|--------------|-------------------|
| TreeBank | 2,437,666 | 338,766 | 251 | 36 |
| XMARK | 20,532,805 | 548 | 77 | 16 |

⁸<http://www.xml-benchmark.org/>

⁹<http://www.cs.washington.edu/research/xmldatasets/www/repository.html>

XML databáze

- ▶ **MonetDB/XQuery**¹⁰ – paměťově orientovaný XQuery procesor postavený na databázovém stroji MonetDB.
- ▶ **BaseX**¹¹ – paměťově orientovaný XQuery procesor napsaný v Javě. Využívá značkových cest a DataGuide.
- ▶ **MS SQL Server** – relační SŘBD s podporou podmnožiny XQuery.
- ▶ **Oracle** – relační SŘBD, tentokrát s plnou podporou XQuery.
- ▶ **HS-LP**¹² – vlastní holistický algoritmus využívající DataGuide.

¹⁰<http://monetdb.project.cwi.nl/monetdb/XQuery/>

¹¹<http://www.inf.uni-konstanz.de/dbis/baseX/>

¹²<http://db.cs.vsb.cz/>

Vytvoření indexu

| | MonetDB/ XQuery | BaseX | Oracle 11g R2 | MS SQL Server 2008 |
|----------|----------------------------|--------------|--------------------------|-------------------------------|
| XMark | - | 220 + 75 | 80.3 + 600 | 534 + 411 |
| TreeBank | 30 | 13 + 10 | 7 | 3 + 44 |

Poznámky:

- ▶ Pro Oracle 11g R2 index obsahuje pouze nestrukturovanou komponentu, v případě TreeBank se nepodařilo vytvořit index.
- ▶ Pro MS SQL Server 2008 vytvoření primárního indexu trvalo 128s, vytvoření sekundárních indexů 283s.

Dotazy

- ▶ Záměrně vybíráme XQuery dotazy s komplikovanější strukturou.
- ▶ Dotazy obsahují většinu os XPath, FLWOR výrazy, wildcards (*), některé funkce a kvantifikátory.
- ▶ Testovali jsme také efektivitu vykonávání vnořených dotazů.
- ▶ Položka tabulky obsahuje DNF, pokud vykonávání dotazu trvalo déle než 10 minut. Pokud při provádění dotazu došlo k chybě, je položka označena E.

XMARK, Dotazy

Dotaz XM1:

```
for $x in //asia [./description [./text and  
    ./keyword]]/item[not(./parlist//parlist)]/name  
return <out> {$x} </out>
```

Velikost výsledku: 17 373

Dotaz XM4:

```
for $x in //asia//item[  
    ./mail/following-sibling::mail]/name  
return <out> {$x} </out>
```

Velikost výsledku: 4 494

XMARK

| Dotaz | MonetDB | BaseX | Oracle 11g | | SQL Server 2008 | | | HS-LP |
|-------|---------|-------|------------|-------|-----------------|----------|---------|-------|
| | | | -Index | Index | -Index | Primární | Sekund. | |
| XM1 | - | 1,2 | 7,6 | DNF | DNF | 232 | DNF | 0,19 |
| XM2 | - | 16,4 | E | E | DNF | DNF | DNF | 0,2 |
| XM3 | - | DNF | DNF | DNF | DNF | DNF | DNF | DNF |
| XM4 | - | 0,5 | 72,3 | 72,9 | - | - | - | 0,01 |
| XM5 | - | DNF | E | E | - | - | - | - |
| XM6 | - | DNF | DNF | DNF | - | - | - | - |
| XM7 | - | DNF | DNF | DNF | - | - | - | - |

- ▶ MonetDB při indexování selhal (jedná se o paměťově orientovaný SŘBD).
- ▶ SQL Server nepodporoval většinu dotazů.
- ▶ Holistické přístupy také podporují jen podmnožinu XQuery.

TreeBank, Dotazy

Dotaz TB1:

```
for $x in //VP[not(./parent::NP[./PP//JJR or ./RB])  
and ./ancestor::S//NP and ./RB]/SBAR  
return <out> {$x} </out>
```

Velikost výsledku: 551

Dotaz TB7:

```
for $p in //EMPTY[./PP[./IN and ./_LRB_]/_RRB_]  
for $pp in count($p//PP)  
let $s :=  
  for $xp in //SBAR[./WHPP/WHNP/WDT]  
  where count($xp//PP) = $pp  
  return $xp  
return <out> {$p//DT/text()}, {count($s)} </out>
```

Velikost výsledku: 28

TreeBank

| Dotaz | MonetDB | BaseX | Oracle 11g | | SQL Server 2008 | | | HS-LP |
|-------|---------|-------|------------|-------|-----------------|----------|---------|-------|
| | | | -Index | Index | -Index | Primární | Sekund. | |
| TB1 | 1,5 | 11 | 68,6 | - | - | - | - | 9,3 |
| TB2 | DNF | DNF | 68,9 | - | - | - | - | 3,7 |
| TB3 | 0,7 | 2 | 68,4 | - | - | - | - | 0,8 |
| TB4 | 2,5 | 9 | 77 | - | - | - | - | - |
| TB5 | 0,4 | 2,4 | 76,1 | - | - | - | - | - |
| TB6 | 0,3 | 2 | E | - | 4 | 2 | 2 | - |
| TB7 | 0,7 | 140 | 0,02 | - | DNF | DNF | DNF | - |

- ▶ Paměťově orientované SŘBD (MonetDB a BaseX) byly schopny vykonat většinu dotazů.
- ▶ Oracle také (jedná se o menší XML dokument s menšími mezivýsledky).
- ▶ SQL Server nepodporoval většinu dotazů.
- ▶ Holistické přístupy také podporují jen podmnožinu XQuery.

O čem jsme nemluvili?

- ▶ Pro jaké třídy dotazů jsou optimální různé holistické algoritmy (tj. v jednotlivých fázích pracují jen s uzly, které jsou ve výsledku celého dotazu).
- ▶ Transakce (uzamykací protokoly pro XML).
- ▶ Paralelní algoritmy, distribuované metody.
- ▶ Další části dotazovacích jazyků (funkce, uživatelské funkce, atd.).
- ▶ Další jazyky a specifikace pro práci s XML daty a jejich implementace pro rozsáhlá data (např. XSLT, DOM).

Závěr

- ▶ Obecně je cílem metod pro vykonávání dotazů zpracovávat v jednotlivých krocích pouze uzly, které jsou po vyhodnocení celého dotazu ve výsledku.
- ▶ Jinými slovy cílem je produkovat minimální mezivýsledky.
- ▶ Z tohoto pohledu jsou některé části dotazovacích jazyků vyřešeny.
- ▶ Co bychom ale čekali od nativní XML databáze:
Uložení libovolně rozsáhlých dat, která budeme dotazovat nějakým standardizovaným dotazovacím jazykem.
- ▶ Z tohoto pohledu nemůžeme v této chvíli hovořit o existujících nativních XML databázích.

- [1] Shurug Al-Khalifa, H. V. Jagadish, and Nick Koudas. Structural Joins: A Primitive for Efficient XML Query Pattern Matching.
In Proceedings of ICDE 2002. IEEE CS, 2002.
- [2] R. Bača and M. Krátký.
A cost-based join selection for XML twig content-based queries.
In Proceedings of the 2008 EDBT workshop on Database technologies for handling XML information on the web, DataX 2008, pages 13–20. ACM New York, NY, USA, 2008.
- [3] R. Bača and M. Krátký.
On the Efficiency of a Prefix Path Holistic Algorithm.

In *Proceedings of the 6th International XML Database Symposium on Database and XML Technologies, XSym, VLDB 2009*, pages 25–32. Springer-Verlag, 2009.

- [4] Radim Bača and Michal Krátký.
TJDewey – On the Efficient Path Labeling Scheme Holistic Approach.
In *Proceedings of Database Systems for Advanced Applications, DASFAA 2009 International Workshops*. Springer–Verlag, 2009.
- [5] M. Brantner and et al.
Full-fledged Algebraic XPath Processing in Natix.
In *Proceedings of 21st International Conference on Data Engineering (ICDE 2005)*, pages 705–716. IEEE, 2005.
- [6] N. Bruno, D. Srivastava, and N. Koudas.

Holistic Twig Joins: Optimal XML Pattern Matching.
In Proceedings of ACM SIGMOD 2002, pages 310–321.
ACM Press, 2002.

- [7] Torsten Grust.
Accelerating XPath Location Steps.
In Proceedings of ACM SIGMOD 2002, Madison, USA.
ACM Press, 2002.
- [8] H.V. Jagadish and et al.
TAX: A Tree Algebra for XML.
*In Proceedings of the 8th International Workshop on
Database Programming Languages (DBPL 2001)*, pages
149–164. Springer-Verlag, 2001.
- [9] Michal Krátký, Radim Bača, and Václav Snášel.

On the Efficient Processing Regular Path Expressions of an Enormous Volume of XML Data.

In Proceedings of the 18th International Conference on Database and Expert Systems Applications, DEXA 2007. LNCS 4653/2007, Springer-Verlag, 2007.

- [10] Michal Krátký, Jaroslav Pokorný, and Václav Snášel.
Implementation of XPath Axes in the Multi-dimensional Approach to Indexing XML Data.
In Proceedings of International Workshop on DataX, Int'l Conference on EDBT 2004, 2004.
- [11] J. Lu, T.W. Ling, C.Y. Chan, and T. Chen.
From Region Encoding to Extended Dewey: on Efficient Processing of XML Twig Pattern Matching.
In Proceedings of VLDB 2005, pages 193–204, 2005.

- [12] J. Widom R. Goldman.
DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases.
In Proceedings of VLDB 1997, pages 436–445, 1997.
- [13] C. Re, J. Simeón, and M.F. Fernández.
A Complete and Efficient Algebraic Compiler for XQuery.
In Proceedings of the 22nd International Conference on Data Engineering (ICDE 2006), pages 14–26. IEEE, 2006.
- [14] J. Teubner T. Grust, S. Sakr.
XQuery on SQL Hosts.
In Proceedings of 29th International Conference on Very Large Data Bases (VLDB 2004), pages 252–263. Morgan Kaufmann, 2004.