DOTAZOVACÍ JAZYKY - XML

slajdy k přednášce NDBI001

J. Pokorný

KSI MFF UK

Obsah

- Část 1: Základy - dokumenty (SGML/HTML) a databáze (strukturovaná a semistrukturovaná data)
- Část 2: Datový model XML
- Část 3: XML a dotazovací jazyky
  - příklady: XML-QL, Xpath
- Část 4: Indexace XML dat

Část I: Základy

Dokumenty vs. databáze

Svět dokumentů	Svět databází
> mnoho malých dokumentů	> několik rozsáhlých databází
> obvykle statický	> obvykle dynamický
> implicitní struktura sekce, paragraf, věta,	> explicitní struktura (schéma)
> značkování	> záznamy
> přizpůsobený člověku	> přizpůsobený stroji
> obsah formát/rozvržení na médiu, anotace	> obsah schéma, data, metody
> paradigmata “ulož jako”, wysiwyg	> paradigmata atomicita, souběžnost, izolace, trvanlivost
> metadata autor jméno, datum, předmět	> metadata popis schématu

Co s nimi dělat

Dokumenty

- editace
- tisk
- lexikální kontrola
- počítání slov
- výběr informací (IR)
- prohledávání

Databáze

- aktualizace
- čištění dat
- dotazování
- skládání/transformování

Hranice mezi dokumenty a db

- Hranice mezi světem dokumentů a světem databází je nejasná.
- V některých návrzích jsou legitimní oba přístupy.
- Někde uprostřed jsou
  - formátovací jazyky
  - semistrukturovaná data

výzkum
z 2. pol. 90. let

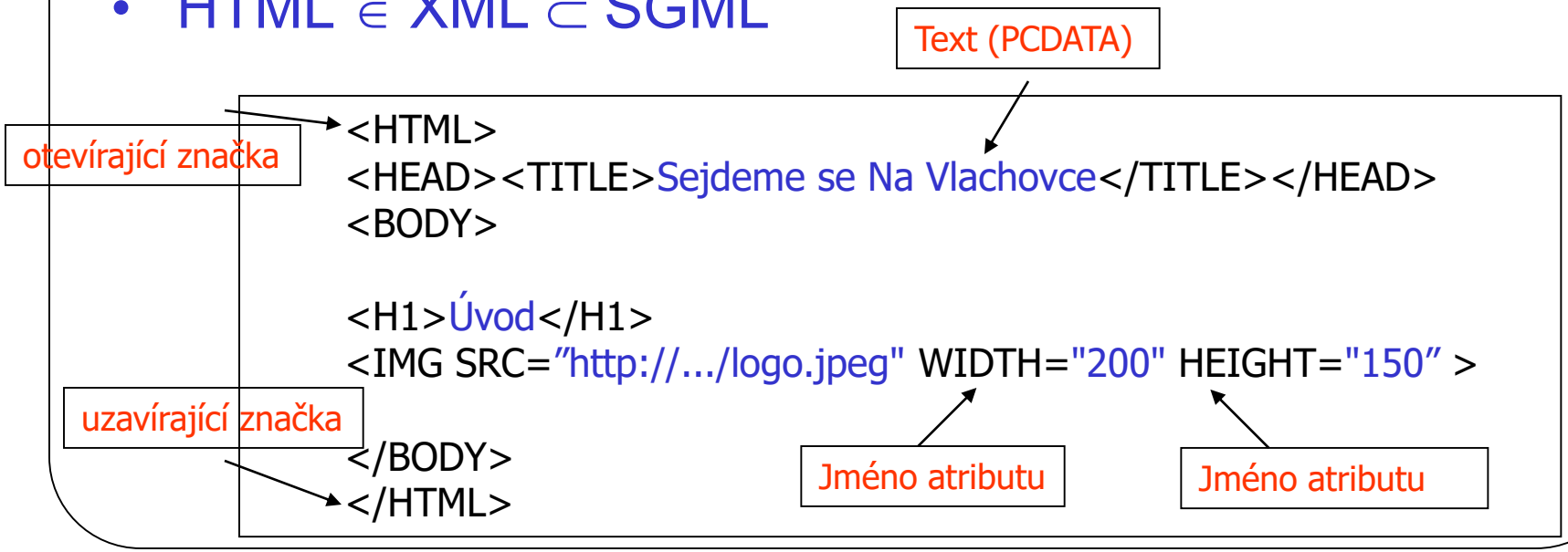
Semistrukturovaná data jsou definována jako data, která jsou neuspořádaná či neúplná, jejich struktura se může měnit, dokonce nepredikovatelným způsobem.

Př.: data ve webových zdrojích, HTML stránky, Bibtexovské soubory, biologická data.

XML data jsou instancí semistrukturovaných dat.

HTML

- Lingua franca pro publikování hypertextu na WWW
- Navržen pro popis, jak by měl webovský prohlížeč uspořádat text, obrázky a tlačítka na stránce.
- Jednoduchý k učení, ale neudržující strukturu.
- Pevná množina značek.
- HTML $\in$ XML $\subset$ SGML



Struktura XML

- XML sestává ze značek a textu
- značky jsou v párech <datum> ...</datum>
- musí být správně hnízděné

  <datum> <den> ... </den> ... </datum> --- dobře

  <datum> <den> ... </datum>... </den> --- špatně

(nelze dělat ... ...)

XML text

XML má pouze jeden "základní" typ -- text.

Text je ohraničen značkami, např.

<název> Databázová abeceda </název>

<rok> 1999 </ rok> --- 1999 je text

XML text se označuje PCDATA (angl. parsed character data). Používá 16-bitové kódování (Unicode).

Později uvidíme, jak jsou pomocí XML dat specifikovány nové typy.

XML struktura

Hnízděné značky mohou být použity k vyjádření různých struktur. Např. n-tice (řádek):

```
<osoba>
  <jméno> Jana Dejmková </jméno>
  <tel> 2191 4264 </tel>
  <email> dejmkova@ksi.ms.mff.cuni.cz </email>
</osoba>
```

XML struktura (pokr.)

- Seznam můžeme reprezentovat opakovaně použitím *stejné* značky:

```
<adresy>
   <osoba> ... </osoba>
   <osoba> ... </osoba>
   <osoba> ... </osoba>
   ...
</adresy>
```

Terminologie

Segment XML dokumentu s počáteční a korespondující koncovou značkou se nazývá *element.* Text mezi značkami je obsah elementu.

Element může být prázdný (nemá obsah - kromě atributů)

<jméno> </jméno> nebo zkráceně *<jméno/>*

Terminologie

Počáteční značka elementu může obsahovat atributy.
Jsou používány typicky k popsání obsahu element.

```
<položka>
  <slovo jazyk = “A”> cheese </slovo>
  <slovo jazyk = “F”> fromage </slovo>
  <slovo jazyk = “N”> Käse </slovo>
  <význam> potravina vytvořená … </význam>
</položka>
```

Atributy

Další využití - vyjádření dimenze nebo typu

```
<obrázek>
    <výška dim = “cm”> 2400 </výška>
    <šířka dim = “in”> 96 </ šířka>
    <data kódování = “gif” komprese = “zip”>
        M05-.+C$@02!G96YE<FEC ...
    </data>
</obrázek>
```

Smíšený obsah

element může obsahovat směs elementů a dat typu PCDATA

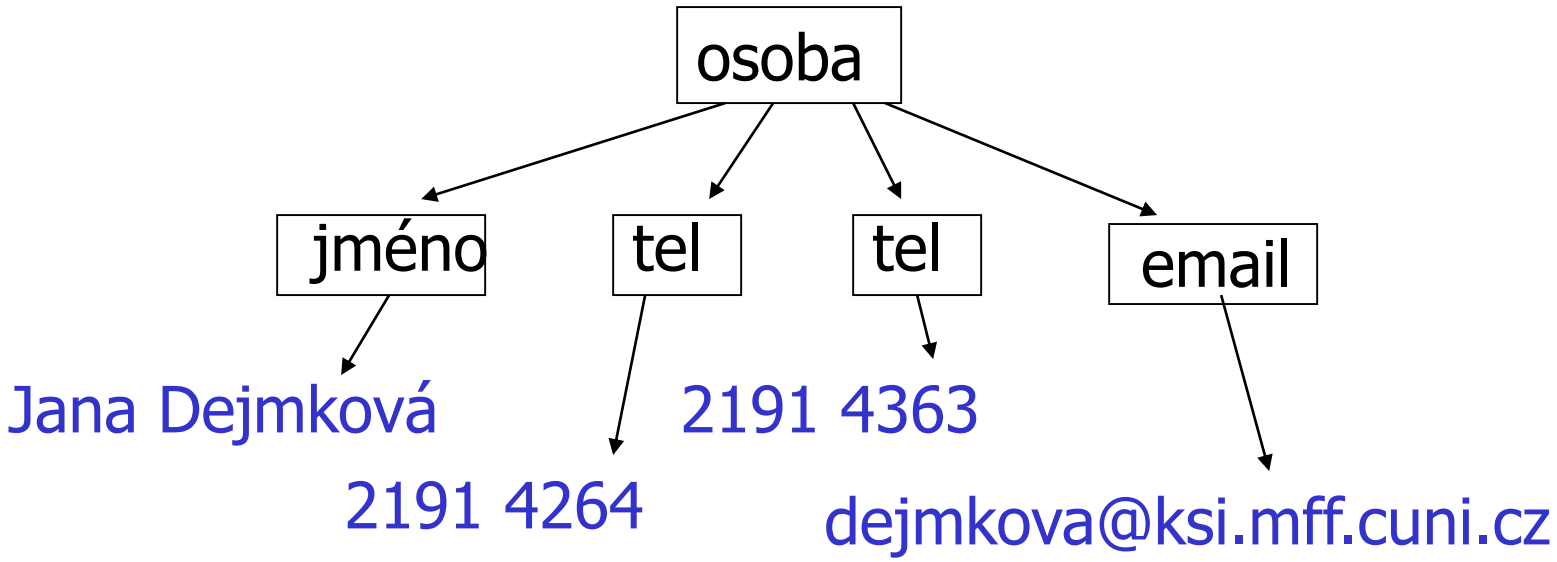
```
<praní>
    <jméno> Persil 1.2 </jméno>
    <motto>
      Světový <pochybný> favorit </pochybný> prášků
    </motto>
</praní>
```

Data tohoto tvaru nejsou typicky generována z (relačních) databází.

Úplný XML Dokument

```
<?xml version="1.0"?>
<osoba>
  <jméno> Jana Dejmková </jméno>
  <tel> 2191 4264 </tel>
  <email> dejmkova@ksi.mff.cuni.cz </email>
</osoba>
```

XML má stromovou strukturu



Pz.:

- na obrázku je model XML textu
- rozdíly vzhledem k modelům semistrukturovaných dat, které používají typicky ohodnocení hran

Příklad: reprezentace relační DB

projekty:

název	rozpočet	řízen

zaměstnanci:

jméno	rod_č	věk

Relace projekty a zaměstnanci v XML

projekty a zaměstnanci jsou smíchaní

```
<db>
  <projekt>
    <název> Vyhledávání </název>
    <rozpočet> 100000 </rozpočet>
    <řízen> Kopecký, M. </řízen>
  </projekt>
  <zaměstnanec>
    <jméno> Dvorský, J. </jméno>
    <rod_č> 700321/1423 </rod_č>
    <věk> 29 < /věk>
  </zaměstnanec>
  <zaměstnanec>
    <jméno> Mikulová,L. </jméno>
    <rod_č> 715512/0132 </rod_č>
    <věk> 38 </věk>
  </zaměstnanec>
  <projekt>
    <název> Třídění </název>
    <rozpočet> 700000 </rozpočet>
    <řízen> Mikulová,L. </řízen>
  </projekt>
  :
</db>
```

Relace projekty a zaměstnanci v XML(pokr.)

zaměstnanci jsou “za” projekty

```
<db>
   <projekty>
     <projekt>
          <název> Vyhledávání </název>
          <rozpočet> 100000 </rozpočet>
          <řízen> Kopecký, M. </řízen>
     </projekt>
     <projekt>
          <název> Třídění </název>
          <rozpočet> 700000 </rozpočet>
          <řízen> Mikulová,L. </řízen>
     </projekt>
    :
   </projekty>
```

```
   <zaměstnanci>
     <zaměstnanec>
        <jméno> Kopecký, M. </jméno>
        <rod_č> 640802/3200</rod_č>
        <věk> 35 </věk>
     </zaměstnanec>
     <zaměstnanec>
        <jméno> Mikulová,L. </jméno>
        <rod_č> 715512/0132 </rod_č>
        <věk>38 </věk>
     </zaměstnanec>
    :
   </zaměstnanci>
</db>
```

Relace projekty a zaměstnanci v XML(pokr.)

nebo bez značky “separátoru” ...

```
<db>
  <projekty>
    <název> Vyhledávání </název>
    <rozpočet> 100000 </rozpočet>
    <řízen> Kopecký, M. </řízen>
    <název> Třídění </název>
    <rozpočet> 700000 </rozpočet>
    <řízen> Mikulová,L </řízen>
    :
  </projekty>
  <zaměstnanci>
    <jméno> Kopecký, M. </jméno>
    <rod_č> 640802/3200 </rod_č>
    <věk> 35 </věk>
    <jméno> Mikulová, L</jméno>
    <rod_č> 715512/0132 </rod_č>
    <věk> 38 </věk>
    :
  </zaměstnanci>
</db>
```

Více o atributech

```
<db>
  <film id="f1">
    <název>Turbína</název>
    <režisér>Novák A.</režisér>
    <obsazení idrefs="h1 h2"></obsazení>
    <rozpočet>100000</rozpočet>
  </film>
  <film id="f2">
    <název>Batalion</název>
    <režisér>Buřita S.</režisér>
    <obsazení idrefs="h2 h9 h21"></obsazení>
    <rozpočet>110000</rozpočet>
  </film>
  <film id="f3">
    <název>Gabriela</název>
    <režisér>Vrchota J.</režisér>
    <obsazení idrefs="h1 h8"></obsazení>
    <rozpočet>90000</rozpočet>
  </film>
  :
```

```
  <herec id="h1">
    <jméno>M. Glázrová</jméno>
    <hrající_v idrefs="f1 f3 f78" >
    </hrající_v>
  </herec>
  <herec id="h2">
     <jméno>K. Höger</jméno>
     <hrající_v idrefs="f1 f2 f11">
     </hrající_v>
     <věk>38</věk>
  </herec>
  <herec id="h3">
      <jméno>H. Vítová</jméno>
      <hrající_v idrefs="f2 f35">
      </hrající_v>
  </herec>
   :
</db>
```

Kdy použít atributy

Není vždy jasné, kdy použít atributy. Atributy nejsou „vidět“.

```
<osoba rod_č= “780730/0013”>
  <jméno> J. Blatný </jméno>
  <email>
     blatný@ksi.mff.cuni.cz
  </email>
  ...
</osoba>
```

```
<osoba>
  <rod_č> 780730/0013 </rod_č>
  <jméno> J. Blatný </jméno>
  <email>
        blatný@ksi.mff.cuni.cz
  </email>
  ...
</osoba>
```

Dokument, který vyhovuje pravidlu “hnízdění značek” a nemá stejné atributy uvnitř značky, se nazývá dobře vytvořený.

Část II: Datový model XML

- Popis typu dokumentu pomocí DTD
- Vztah k objektovému datovému modelu
- Vztah k modelu semistrukturovaných dat
- Rozšíření datového modelu XML

Popis typu dokumentu pomocí DTD

- Document Type Descriptors (DTDs) přiřazují XML dokumentu strukturu.
- existuje *jistý* vztah mezi DTD a schématem databáze,
- DTD je a *syntaktická* specifikace.

Příklad: Osobní adresář

<osoba>

<jméno> Říha Antonín </jméno> } přesně jedno jméno

<s_titulem> Dr. A. Říha </ s_titulem> } Max. 1

<adresa>Malostranské 25 </adresa> } tolik řádků pro adresy, kolik je třeba

<adresa> Praha, 100 00 </adresa>

<tel> 2191 4268 </tel> } smíchané telefony a faxy

<fax> 2191 4323 </fax>

<tel> 2191 4323 </tel>

<email> riha@ksi.mff.cuni.cz</email> } kolik jich je třeba

</osoba>

Specifikace struktury

- jméno — specifikuje element jméno
- s_titulem? — specifikuje nepovinné (0 nebo 1) elementy s_titulem
- jméno, s_titulem? — specifikuje jméno následované nepovinně s_titulem
- adresa* — specifikuje 0 nebo více řádků adresa
- tel | fax — tel *nebo* fax element
- (tel | fax)* — 0 nebo více tel nebo fax
- email* — 0 nebo více elementů email

Specifikace struktury (pokr.)

celá struktura elementu osoba je specifikována

jméno, s_titulem?, adresa*, (tel | fax)*, email*

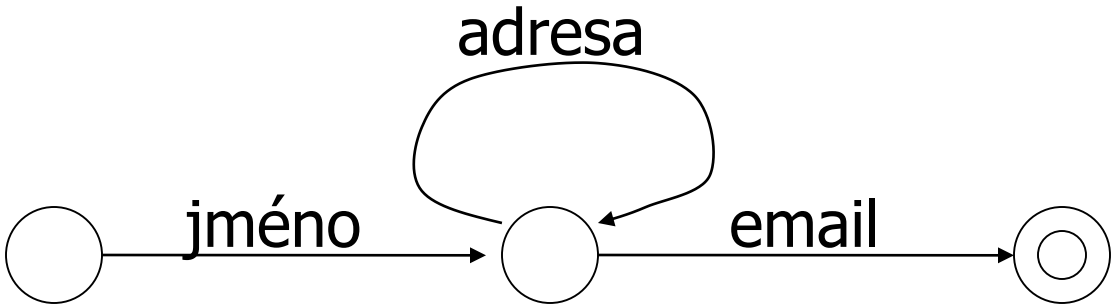
Jde o regulární výraz. Proč je důležitý?

Regulární výrazy

Každý regulární výraz určuje korespondující *konečný automat.*
Příklad:

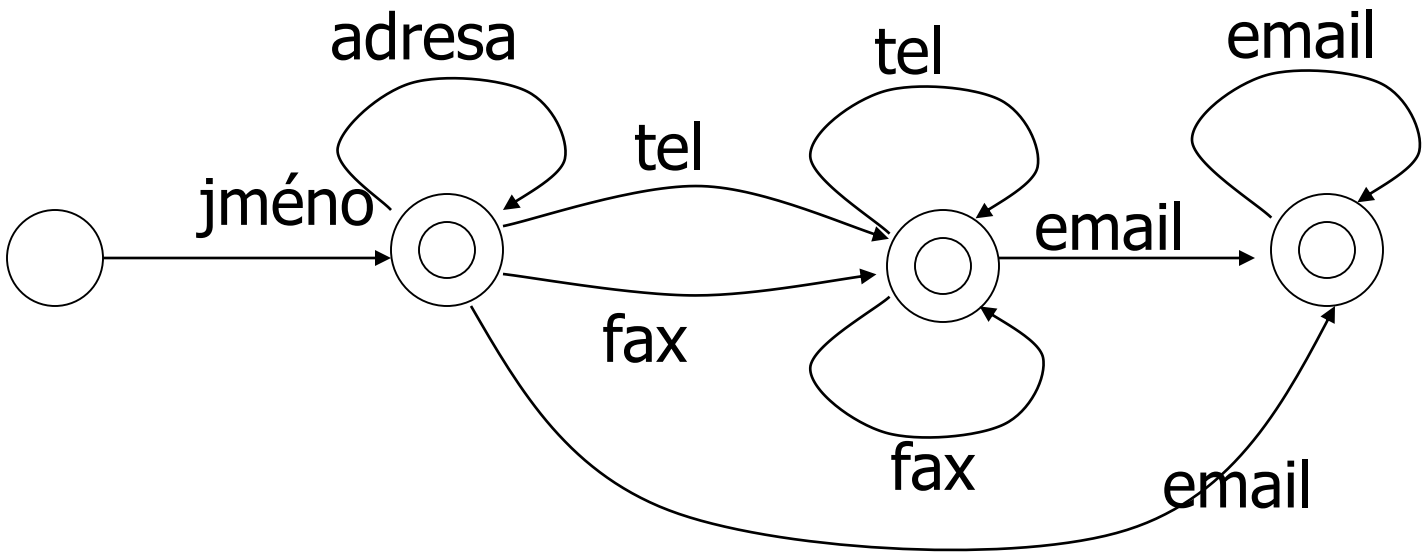
jméno, adresa*, email

Odpovídající jednoduchý program (parser)



Další příklad

jméno, adresa*, (tel | fax)*, email*



Přidání nepovinného s_titulem vede ke komplikacím s velikostí automatu

DTD pro adresář

```
<!DOCTYPE adresář [
   <!ELEMENT adresář (osoba*)>
   <!ELEMENT osoba
       (jméno, s_titulem?, adresa*, (fax | tel)*, email*)>
   <!ELEMENT jméno   (#PCDATA)>
   <!ELEMENT s_titulem    (#PCDATA)>
   <!ELEMENT adresa  (#PCDATA)>
   <!ELEMENT tel     (#PCDATA)>
   <!ELEMENT fax     (#PCDATA)>
   <!ELEMENT email   (#PCDATA)>
]>
```

Revidovaná relační DB

projekty:

název	rozpočet	řízen

zaměstnanci:

jméno	rod_č	věk

Dva DTD relační DB

```
<!DOCTYPE db [
  <!ELEMENT db            (projekty,zaměstnanci)>
  <!ELEMENT projekty      (projekt*)>
  <!ELEMENT zaměstnanci (zaměstnanec*)>
  <!ELEMENT projekt       (název, rozpočet, řízen)>
  <!ELEMENT zaměstnanec   (jméno, rod_č, věk)>
  ...
]>
```

```
<!DOCTYPE db [
  <!ELEMENT db            (projekt | zaměstnanec)*>
  <!ELEMENT projekt       (název, rozpočet, řízen)>
  <!ELEMENT zaměstnanec (jméno, rod_č, věk)>
  ...
]>
```

Rekurzivní DTD

```
<!DOCTYPE genealogie [
    <!ELEMENT genealogie (osoba*)>
    <!ELEMENT osoba (
                    jméno,
                    datum_nar,
                    osoba,                  -- matka
                    osoba   )>              -- otec
    ...
]>
```

Kde je problém? Povinní rodiče. Pořadí.

Rekurzivní DTD (pokr.)

```
<DOCTYPE genealogie [
    <!ELEMENT genealogie (osoba*)>
    <!ELEMENT osoba (
                    jméno,
                    datum_nar,
                    osoba?,                    -- matka
                    osoba?  )>                 -- otec
    ...
]>
```

Kde je nyní problém? Pořadí.

Lepší řešení: s ID, IDREF, IDREFS

Některé věci je obtížné specifikovat

Každý zaměstnanecký element obsahuje elementy jméno, věk a rod_č v nějakém pořadí.

```
<!ELEMENT zaměstnanec
    ( (jméno, věk, rod_č) | (věk, rod_č, jméno) |
      (rod_č, jméno, věk) | ...
    )>
```

Předpokládejte situaci, kdy existuje více atributů zaměstnance!

Přehled regularních výrazů v XML

- A — značka A se vyskytuje
- e1,e2 — výraz e1 následovaný e2
- e* — 0 nebo více výskytů e
- e? — nepovinné -- 0 nebo 1 výskytů
- e+ — 1 nebo více výskytů
- e1 | e2 — buď e1 nebo e2
- (e) — seskupování

Specifikace atributů v DTD

```
<!ELEMENT výška (#PCDATA)>
<!ATTLIST výška
    dimenze CDATA #REQUIRED
    přesnost CDATA #IMPLIED >
```

Atribut dimenze je požadován; atribut přesnost je nepovinný.

CDATA je “typ” atributu -- znamená string.

Specifikace atributů ID a IDREF

```
<!DOCTYPE rodina [
  <!ELEMENT rodina   (osoba)*>
  <!ELEMENT osoba  (jméno)>
  <!ELEMENT jméno    (#PCDATA)>
  <!ATTLIST osoba
                    id        ID       #REQUIRED
                    matka     IDREF    #IMPLIED
                    otec      IDREF    #IMPLIED
                    děti      IDREFS   #IMPLIED>
]>
```

Dobře vytvořený dokument mající DTD a vyhovující tomuto DTD se nazývá platný (validní).

Některá validní data

```
<rodina>
    <osoba  id="jana"  matka="marie" otec="josef">
        <jméno> Jana Nováková </jméno>
    </osoba>
    <osoba id="josef" děti="jana vít">
        <jméno> Josef Novák </jméno>
    </osoba>
    <osoba id="marie" děti="jana  vít">
        <jméno> Marie Nováková </jméno>
    </osoba>
        <osoba  id="vít"  matka=”marie" otec="josef">
        <jméno> Vít Novák </jméno>
    </osoba>
</rodina>
```

Konzistence hodnot ID a IDREF(S) atributů

- Je-li atribut deklarován jako ID
  - asociované hodnoty musí být v dokumentu všechny různé
- Je-li atribut deklarován jako IDREF
  - asociovaná hodnota musí existovat jako hodnota nějakého ID atributu (žádné „visící ukazatele”) v daném dokumentu
- podobně pro všechny hodnoty atributu IDREFS
- Atributy ID, IDREF a IDREFS *nejsou typovány*

DTD vs. db schémata (nebo typy)

- Z hlediska standardů databází (nebo programovacího jazyka) jsou DTD spíše slabé specifikace.
  - pouze jeden základní typ -- PCDATA
  - žádné užitečné “abstrakce” (např. množiny)
  - IDREF jsou netypované. Ukazuje se na něco, ale neví se na co!
  - žádná IO, např., dítě je inverzní k rodičům
  - žádné metody
- Návrhy na rozšíření XML: schémata, IO
  - DCD (Document Content Desription)
  - XML Schema (návrh W3C)
  - Microsoft v Exploreru 5.
- Dnes populární JSON: „odlehčená“ a jednodušší alternativa k XML

Část III: Dotazovací jazyky

XML-QL

http://www.w3.org/TR/NOTE-xml-ql

http://db.cis.upenn.edu/XML-QL/

XPath

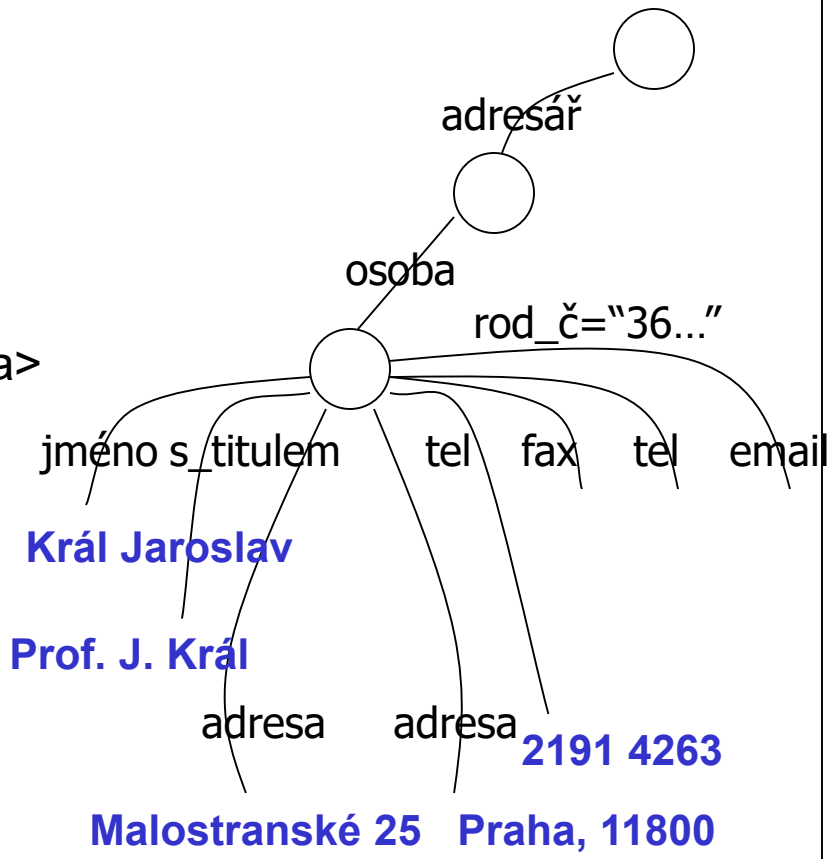
http://www.w3.org/TR/xpath

XML-QL: datový model

- orientovaný ohodnocený graf
- značky jsou reprezentovány jako ohodnocení *hran*
- ohodnocení uzlů jsou množinami dvojic jméno atributu/hodnota
- 2 modely: uspořádaný a neuspořádaný

XML-QL - datový model (pokr.)

```
<osoba rod_č="360528/281"
      <jméno> Král Jaroslav </jméno>
      <s_titulem> Prof. J. Král </s_titulem>
      <adresa> Malostranské 25 </adresa>
      <adresa> Praha, 100 00 </adresa>
      <tel> 2191 4263 </tel>
      <fax> 2191 4323 <fax>
      <tel> 2191 4323 </tel>
      <email> kral@ksi.mff.cuni.cz </email>
</osoba>
```



XML-QL (XML Query Language)

- Proč dotazovací jazyk? Extrakce, restrukturalizace, integrace, listování…
- požadavky:
  - vyhledávání pomocí klíčových slov (podobně jako v systémech úplných textů),
  - navigace podle struktury značek XML,
  - silný přístup ke strukturovaným datům podobný např. možnostem SQL,
  - techniky založené na pojmu podobnost dokumentů či vzdálenosti (proximity) mezi termy.

XML-QL (XML Query Language)

- návrh konsorcia W3, August 1998

- autoři:
  - Mary Fernandez — AT&T
  - Dana Florescu — INRIA
  - Alon Levy — Univ. of Washington
  - Dan Suciu — AT&T
  - Alin Deutsch — Univ. of Pennsylvania

XML-QL: výraz určující cestu

jednoduchá cesta specifikuje jeden krok v navigaci v db.

x/l

cesta je seznam jednoduchých cest.

x/l/k

Klíčový pojem: regulární výraz určující cestu

Př.: biblio/(zpráva | článek) -- větvení

autor/křestní_jméno? -- částečná informace

zpráva/reference*/autor -- Kleeneho uzávěr

biblio/_*/autor

Pz.: R+, kde R je regulární výraz, je ekvivalentní R/R*

Revidovaný adresář

```
<adresář>
  <osoba rod_č="420529/102">
      <jméno> Říha Antonín </jméno>
      <s_titulem> Dr. A. Říha </s_titulem>
      <adresa> Malostranské 25 </adresa>
      <adresa> Praha, 100 00 </adresa>
      <tel> 2191 4268 </tel>
      <fax> 2191 4323 <fax>
      <tel> 2191 4323 </tel>
      <email>riha@ksi.mff.cuni.cz </email>
  </osoba>
</adresář>
```

XML-QL: porovnávání vzorků

D1.: Nalezni Říhovu e-mailovou adresu:

```
where  <adresář>
          <osoba>
                <jméno> Říha Antonín </jméno>
                <email>$e</email>
          </osoba>
       </adresář> in “http://kocour.mff.cuni.cz/~honza/adresa.xml”
construct  $e
```

<XML> riha@ksi.mff.cuni.cz </XML>

Selekce (extrakce) dat

XML-QL: konstrukce nových XML dat

D2: Koho můžeme elektronicky kontaktovat?

```
where     <adresář>
            <osoba>
              <s_titulem>$g</s_titulem>
              <email>$e</email>
            </osoba>
          </adresář> in "http://..."
construct <e-kontakt>
            <kdo>$g</kdo>
            <kde>$e</kde>
          </e-kontakt>
```

```
<XML>
  <e-kontakt>
    <kdo> Dr. A. Říha </kdo>
    <kde>riha@ksi.mff.cuni.cz
    </kde>
  </e-kontakt>
  <e-kontakt>
    <kdo>Prof. J. Král</kdo>
    <kde> kral@ksi.mff.cuni.cz
    </kde>
  </e-kontakt>
  ...
</XML>
```

Restrukturalizace dat

XML-QL: spojení

D3: Kdo z našich kontaktů byl zapojen ve filmu?

```
where   <adresář>
          <osoba>
            <s_titulem>$g</s_titulem>
            <email>$e</email>
          </osoba>
        </adresář> in "http://…adresa.xml"
        <film><název>$t</>
                <jméno>$g</>   //jde o jméno herce
        </film>  in "http://www.barrandov.com"
construct  <film-kontakt>
             <kdo>$g</kdo>
             <film>$t</film>
             <kde>$e</kde>
           </film-kontakt>
```

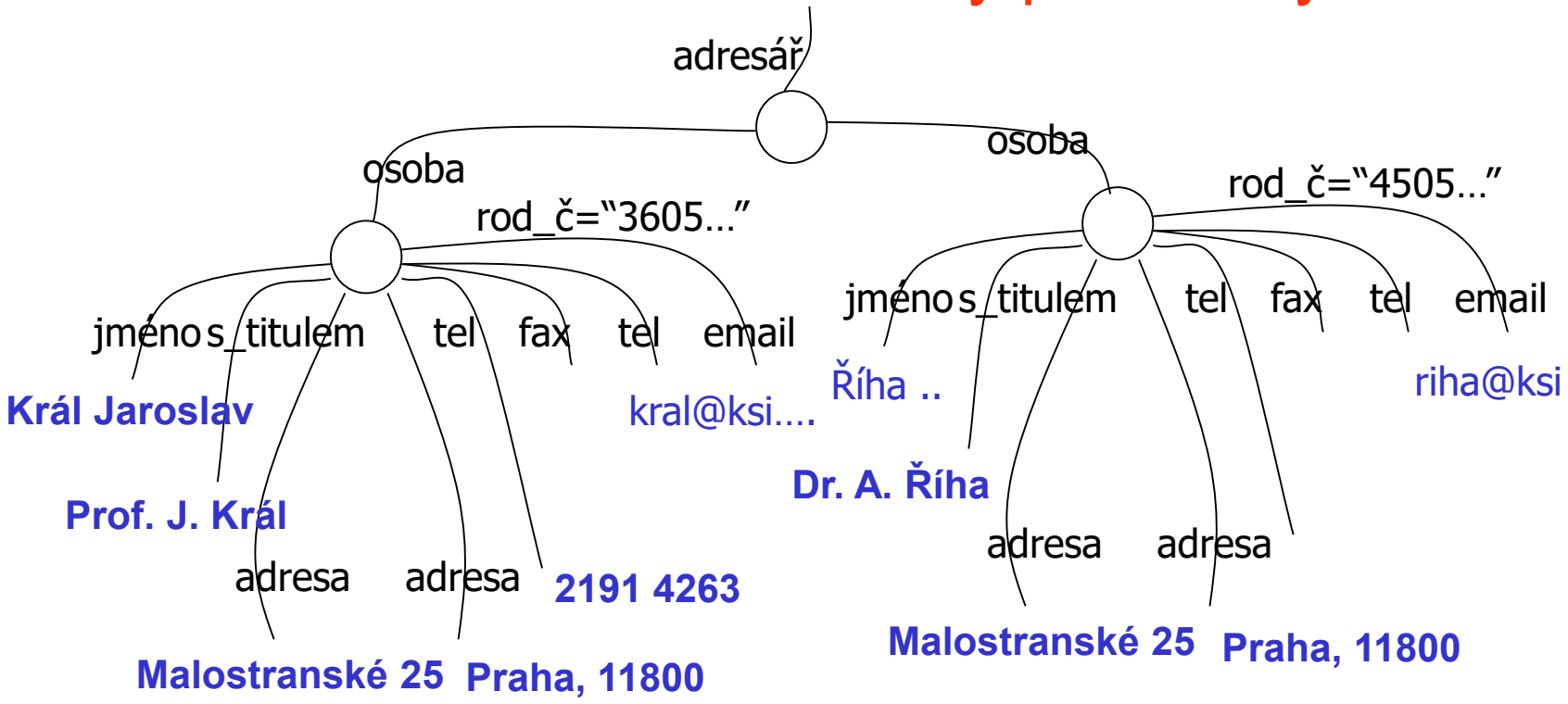
XML-QL: spojení (pokr.)

```
<XML>
  <film-kontakt>
    <kdo> Prof. J. Král </kdo>
    <kde> kral@ksi.mff.cuni.cz </kde>
    <film>Král Šumavy</film>
  </film-kontakt>

  <film-kontakt>
    <kdo>Dr. D. Húsek</kdo>
    <kde>husek@ui.cas.cz</kde>
    <film>Jakpak je dnes u nas doma</film>
  </film-kontakt>
...
</XML>
```

Integrace dat

XML-QL sémantika: vazby proměnných




```
where <adresář>
        <osoba>
          <jméno>$n</>
          <email>$e</>
        </>
      </>
```

$n	$e
Král Jaroslav	kral@ksi.mff…
Říha Antonín	riha@ksi.mff….

XML-QL pro pokročilé

Značkové a atributové proměnné

D4.: Najdi značky podelementů elementů osoba :

```
where      <adresář.osoba.$tag></>
             in "http://kocour.mff.cuni.cz/~honza/adresa.xml"
construct  <elementy_osoby>$tag</>
```

D5.: Najdi atributy elementů osoba:

```
where      <_*.osoba $jm_atr=$h></>
             in "http://kocour.mff.cuni.cz/~honza/adresa.xml"
construct  <atr_osoby>
             <jméno>$jm_atr</>
             <hodnota>$h</>
           </>
```

Listování ve schématu

XML-QL pro pokročilé

Regulární výrazy určující cestu
D6: Najdi všechny emailové adresy a faxová čísla:

```
where      <adresář._*.(email | fax)>$ef</>
             in "http://kocour.mff.cuni.cz/~honza/adresa.xml"
construct  <email_Fax>$ef</>
```

Integrace heterogenních dat

XML-QL pro pokročilé

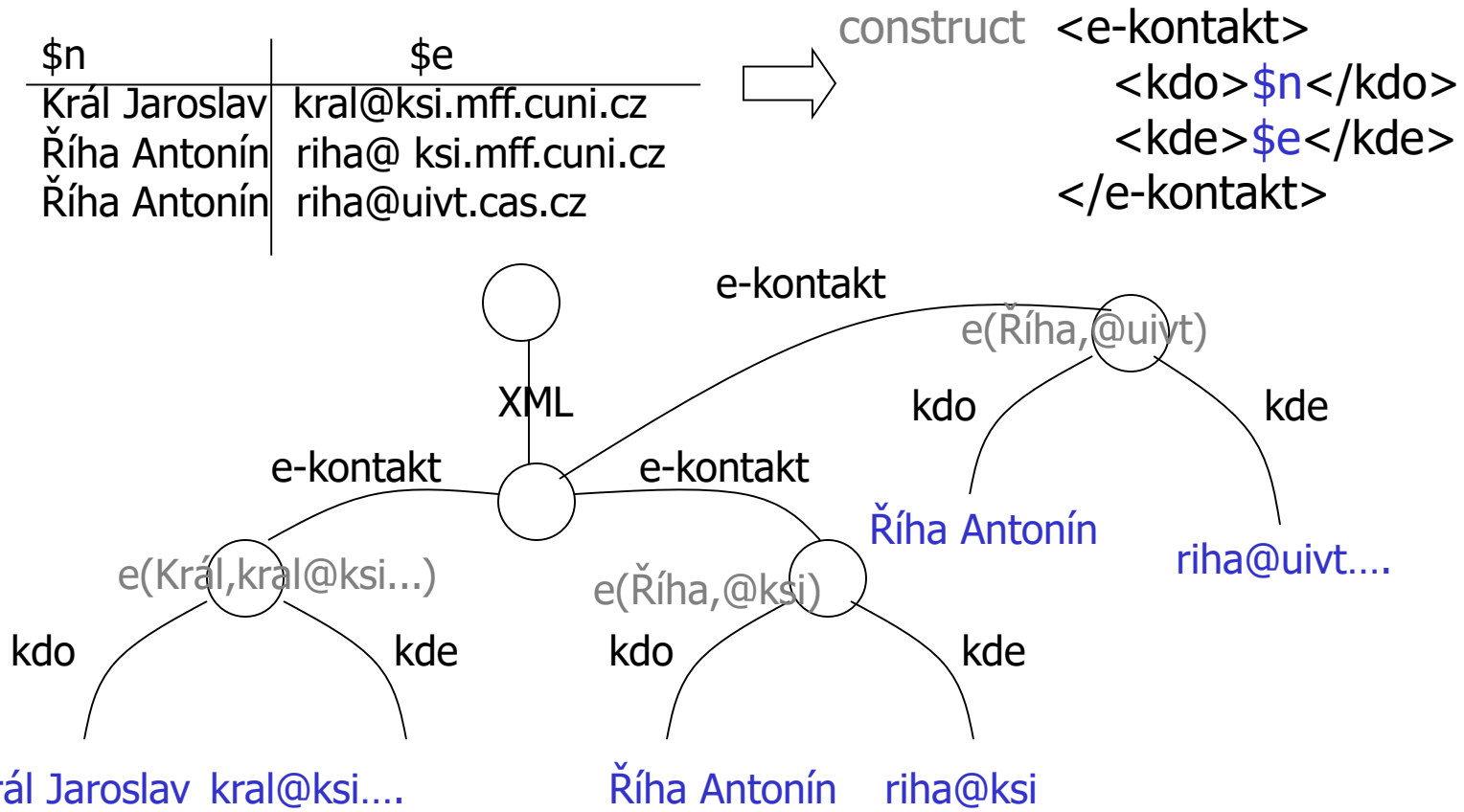
D3‘: Kdo z našich kontaktů byl zapojen ve filmu?

```
where     <adresář>
            <osoba>
              <s_titulem>$g</>
              <email></> ELEMENT_AS $e
            </osoba>
          </adresář> in “http://…adresa.xml”
          <film><název></> ELEMENT_AS $t
                  <jméno>$g</>   //jde o jméno herce
          </film>  in “http://www.barrandov.com”
construct  <film-kontakt>
                  <kdo>$g</kdo>
                  $t $e
          </film-kontakt>
```

Přenos elementů

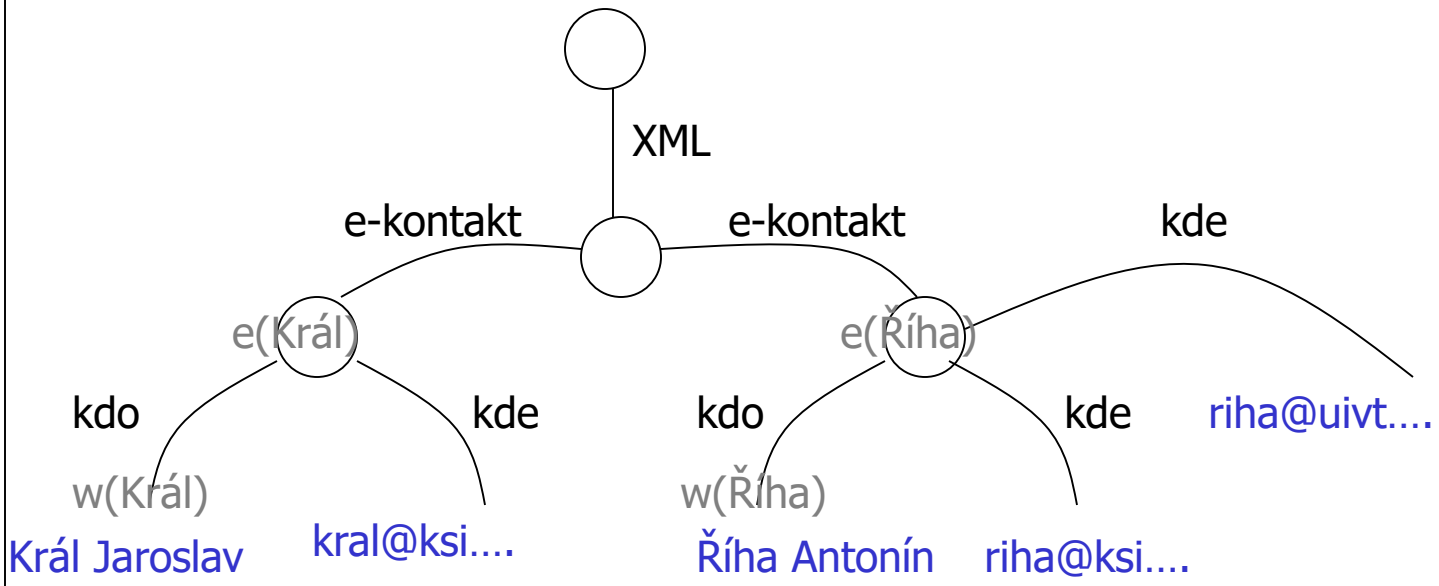
XML-QL sémantika: výstup v XML

$n	$e
Král Jaroslav	kral@ksi.mff.cuni.cz
Říha Antonín	riha@ ksi.mff.cuni.cz
Říha Antonín	riha@uivt.cas.cz



XML-QL: Skolemovy funkce

```
construct  <e-kontakt ID=e($n) >
              <kdo ID=w($n) >$n</kdo>
              <kde>$e</kde>
           </e-kontakt>
```



XPath

- XPath je syntaxe použitá pro výběr částí XML dokumentu
- Možnosti výsledku vyhodnocení výrazu XPath:
  - množina uzlů,
  - číslo,
  - boolská hodnota,
  - řetězec.

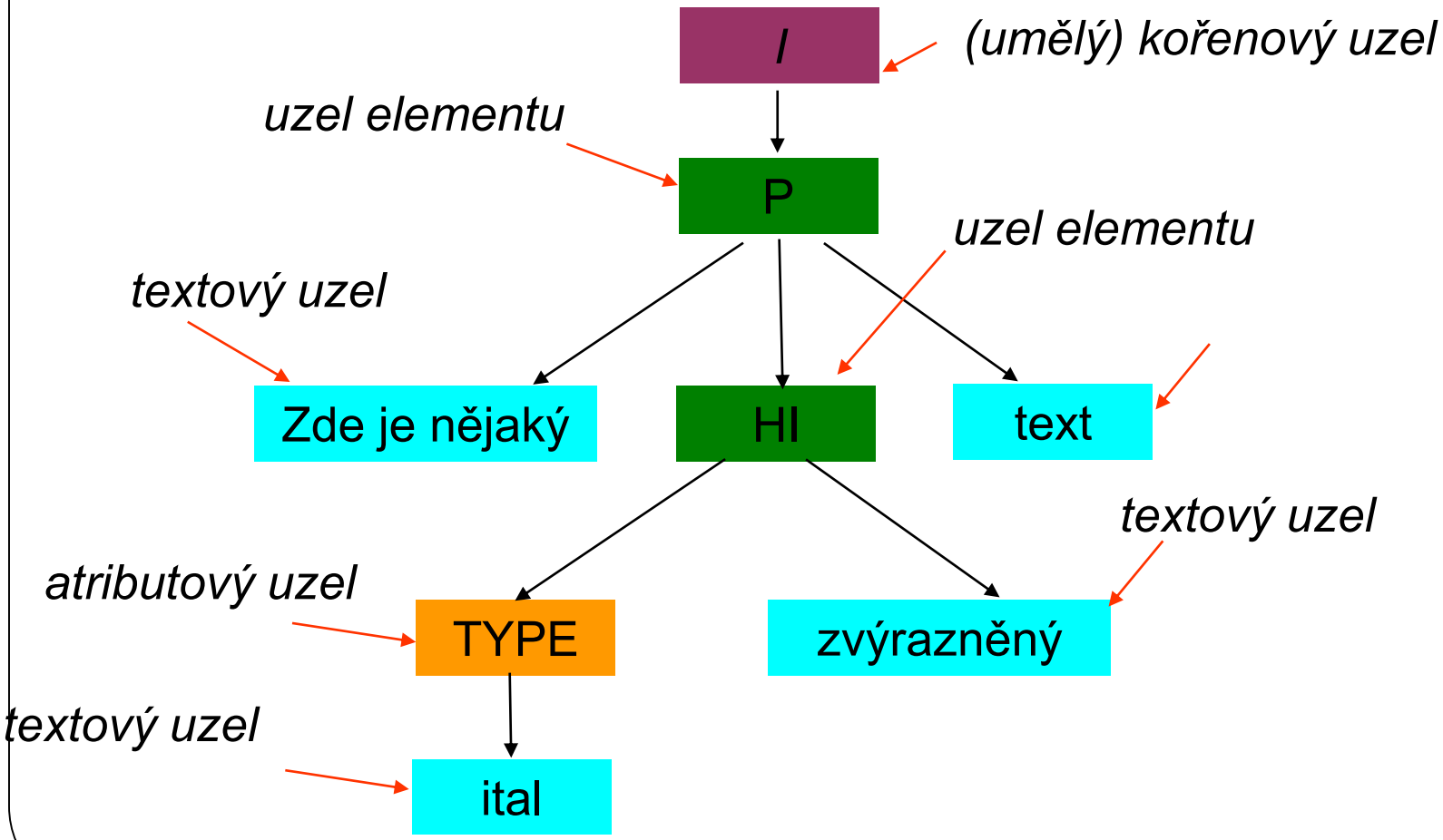
jednoduchá cesta založená na relaci předci-potomci

cesta je seznam jednoduchých cest, např. x/l/k

  - Každý krok je vyhodnocen v rámci nějakého kontextu.
  - Výsledkem každého kroku je množina uzlů. Výsledky nějakého kroku vstupují jako kontext do následujícího kroku.

- Pz: vyhodnocovací algoritmy v PTIME v $|D|$ a $|DB|$

Model používaný v XPath



XPath

Dotazy využívají různé relace mezi uzly (osy v XPath)

ancestor (předci) – uzly ležící na cestě od *u* do kořenu,

ancestor-or-self (předci včetně mne) – *u* a uzly ležící na cestě od *u* do kořenu,

parent (rodič) – první uzel ležící na cestě od *u* do kořenu,

child (děti) – bezprostřední následníci uzlu *u*,

descendant (potomci) - všechny uzly, pro které je uzel *u* předkem,

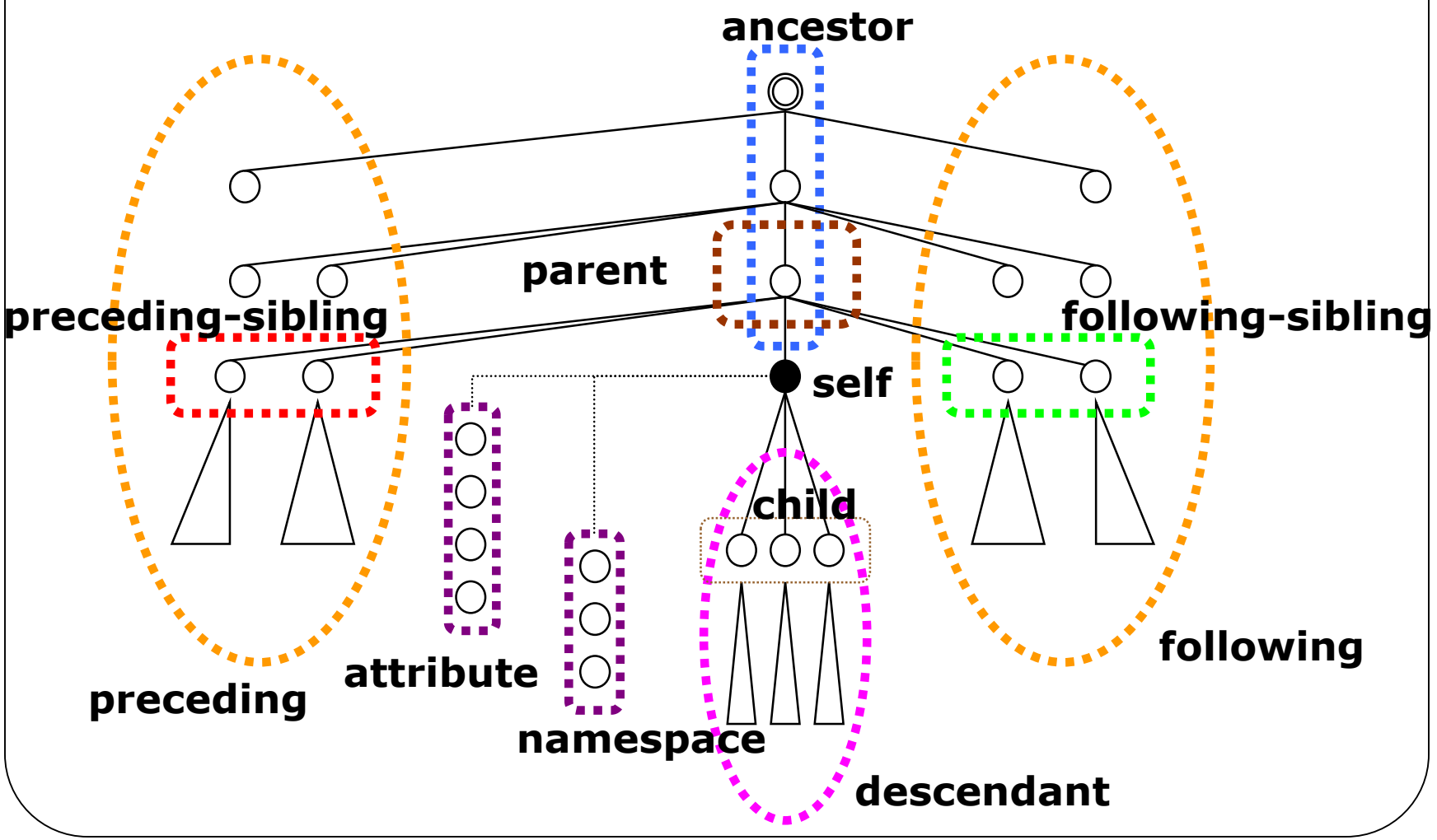
preceding-sibling (předcházející sourozenci) – sourozenci uzlu *u* předcházející *u* při průchodu stromem doleva-do-hloubky,

following-sibling (následující sourozenci) – sourozenci uzlu *u* následující *u* při průchodu stromem doleva-do-hloubky,

preceding (předchůdci) – uzly předcházející *u* (kromě jeho předků) při průchodu stromem do doleva-do-hloubky,

following (následníci) – uzly následující po *u* (kromě jeho potomků) při průchodu stromem doleva-do hloubky.

XPath


ancestor
parent
preceding-sibling
following-sibling
self
child
preceding
attribute
namespace
following
descendant

XPath

Klíčový pojem: regulární výraz určující cestu

podobně jako v XML-QL

biblio/(zpráva | článek) -- větvení

autor/křestní_jméno? -- částečná informace

zpráva/reference*/autor -- Kleeneho uzávěr

- Zde: jde o relativní cestu začínající z běžného (kontextového) elementu.
- Cesta, která začíná / reprezentuje absolutní cestu, začínající z kořene XML dat.

  /kniha
- Cesta začínající // může začínat *kdekoliv* v dokumentu.

  //nadpis vybere každý element nadpis, který se vyskytuje v dokumentu

XPath

Příklady dotazů:

/článek/*/odstavec --- * jako zástupný symbol

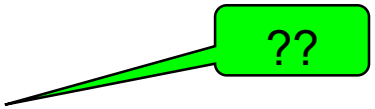
článek//obrázek --- // zkratka za relaci potomci

//článek[autor=’Jiří Kosek’]

Složitější:

//článek[název = ’Lehký úvod do XML’]/autor

článek[autor]//název



obrázek/ancestor::kapitola/following-sibling::kapitola

XPath

Příklady dotazů:

/článek/*/odstavec --- * jako zástupný symbol

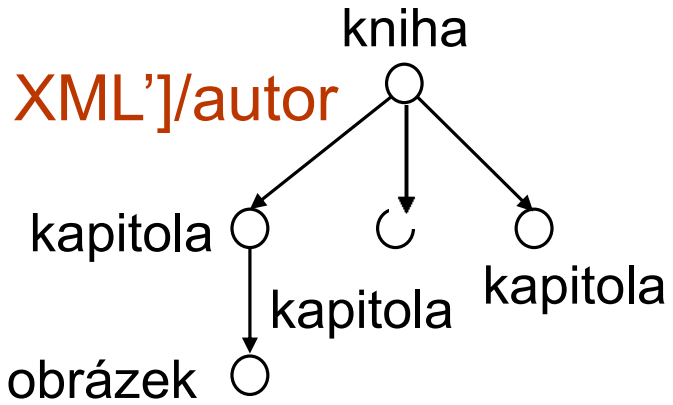
článek//obrázek --- // zkratka za relaci potomci

//článek[autor='Jiří Kosek']

Složitější:

//článek[název = 'Lehký úvod do XML']/autor

článek[autor]//název



obrázek/ancestor::kapitola/following-sibling::kapitola

Vybere kapitolu, která je následujícím sourozencem kapitoly, ve které je obrázek

Část IV: Indexace XML dat

Lore

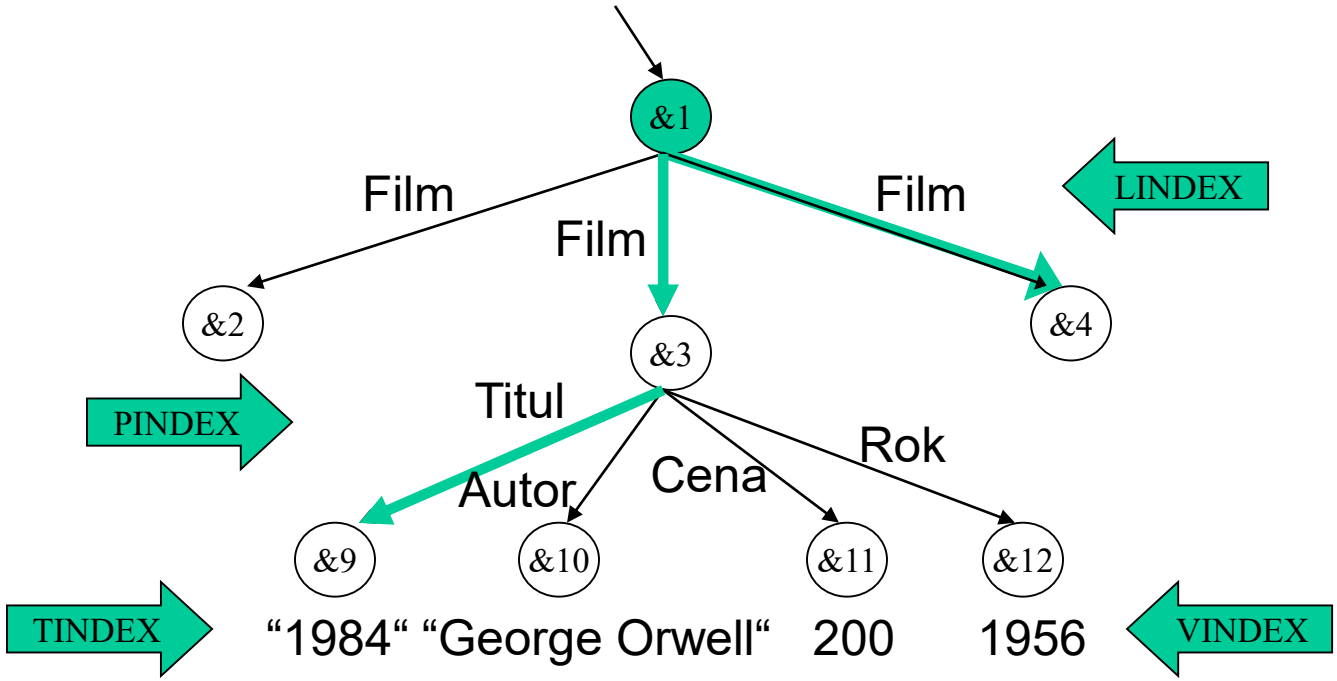
http://infolab.stanford.edu/lore/

Metody indexace semistrukturovaných dat

- Indexace úplného textu

  Nevýhoda: nelze dotazovat podle struktury
- Indexace relací klasicky (Lore)
- Indexování založené na pozicích
  - Použití absolutních resp. relativních adres pro reprezentaci pozic slov a značek v XML dokumentu
- Indexování založené na cestách
  - kódují se cesty podle typu průchodu stromem
  - kóduje všechny cesty vedoucí ke všem slovům
    - je možné se dotazovat na obsah i strukturu
  - …

Indexace v Lore



Value Index (VINDEX)

- Vstup: značka T, porovnávací operátor Θ, hodnota v
- Výstup: všechny atomické objekty mající vstupující hranu T a hodnotu v' vyhovující Θ a hodnotě v.
- Př.: (*Cena*, >, 150)

  Výsledek je {&11, &15}
- Vindex lze implementovat např. B+-stromů;

Link Index (LINDEX)

- Vstup: oid o a značka T
- Výstup: všechny rodičovské objekty mající hranu T vstupující do o.
  - Je-li T vynechána, vrací všechny rodiče a jejich značky.
- Př.: nalezení rodiče pomocí Lindex pro objekt &4 podle značky Film;

  vrací rodičovský objekt &1
- Lindex lze implementovat např. lineárním hašováním

Link Index (LINDEX)

Př.: db/A/B[C=5]

Využije Vindex i Lindex:

- C komponenta pomocí Vindex a (C, =, 5)
- Zkouší, zdali existuje cesta A/B z db do tohoto objektu pomocí dvou volání Lindex
- Zpětné vyhodnocení

Text Index (TINDEX)

- Vstup: Tindex provádí hledání podle klíčových slov na hodnotách typu text.
- Výstup: seznam souřadnic tvaru <*o*, *n*>
- Lze implementovat pomocí invertovaných seznamů, mapujících slovo w a značku T do seznamu atomických hodnot v se vstupující hranou T, přičemž v obsahuje w.
- Značka může být vynechána.

Př.: nahlédnutí pomocí Tindex na všechny objekty s atomickou hodnotou obsahující slovo “Ford“ a mající
Výsledek: {<&17, 2><&21, 2>}

Path Index (PINDEX)

- Vstup: objekt o a výraz p označující cestu
- Výstup: všechny objekty dosažitelné z o po cestě p
- omezení: obvykle pouze jednoduché cesty, tj. ty začínající v pojmenovaných objektech a neobsahující žádné regulární výrazy

Př.: db/Film/Titul

Pindex k nalezení všech objektů dosažitelných pomocí db/Film/Titul

Výsledek: {&5, &9, &14}

Vyhodnocování shora dolů přímé

Př.: db.Film[Cena < 200]

- prohledají se všechny podelementy elementu Film v db a pro každý se nahlédne na obsah podelementu Cena, jestli je jeho hodnota menší než 200
- To vede na prohledávání stromu do hloubky (depth-first) podle hran, které se vyskytují ve výrazech cest

Vyhodnocování zdola-nahoru s indexy

Př.: db.Film[Cena < 200]

- Nejprve se naleznou všechny objekty vyhovující podmínce použitím vhodného Vindex indexu
- Pro každý z těchto objektů se prochází zpět ve stromě pomocí Lindex směrem k jejich rodičům

  Výhoda: vyhne se cestám, které nesplňují podmínku.

Vyhodnocování hybridní s indexy

Př.: db.Film[Cena < 200]

- Vyhodnotí se něco (ne nutně všechno) co se týká podmínky pomocí shora-dolů
- Pak se naleznou přímo ty objekty, které vyhovují podmínce pomocí Videx a prochází se dál pomocí Lindex do stejného bodu jako při shora-dolů přístupu
- Výsledek dotazu se nalezne jako průnik množiny objektů a kombinace procházení cest.

XML - rodina standardů