XML pod lupou (2)

XML dokumenty si můžeme představit jako databáze, jejichž hlavním úkolem je zpracování dotazů. Ke zjednodušení tohoto úkolu se v praxi používají tzv. modely XML dat. Které modely to jsou, jaké výhody přinášejí XML-grafy a jak vlastně XML databáze vznikají?

Dokument XML (Extensible Markup Language) či sadu XML dokumentů lze chápat jako speciální databázi. Jazykem XML konstruujeme definice DTD (Document Type Definition; obdoba databázového schématu) a vlastní XML data (obdoba databáze). Podobně jako u relační databáze, cílem provozu XML databáze je především zpracování dotazů. Ke zjednodušení problému se používají modely XML dat, které zjednodušují pohled na XML data. Jde vlastně o to, abychom neviděli XML data jako text, nýbrž jako nějakou "strukturovanější" datovou strukturu. Ve světě relací je takovou strukturou tabulka a jazykem SQL (Structured Query Language).

Modely XML dat jsou většinou stromové nebo grafové. Jde o přirozenou představu, jež vychází z prezentace XML dokumentu pomocí stromů. Prvním modelem, který bychom mohli využít, je model DOM (Document Object Model), jenž je použit při rozkladu XML dokumentu odpovídajícím parserem. Je objektově orientovaný a slouží hlavně jako rozhraní mezi XML dokumentem a aplikacemi. Existují však bohatší modely. Další přístup vychází z obecnějšího a historicky také staršího přístupu – modelu semistrukturovaných dat. Pojem schématu, tak jak jsme zvyklí z relačních či objektových databází, nemusí být u těchto dat vůbec definován. V XML nemusí být schémata (DTD) dopředu známa, nebo se v čase mění. Dané schéma DTD, existuje-li, tedy popisuje pouze část a mnohdy ještě pouze dočasně data daného informačního zdroje. Zdá se však, že budoucnost XML spočívá právě v použití DTD či nějakého obecnějšího popisu, jakým je XML schéma. Označkovaný dokument bez dohodnutého jazyka značek není totiž pro výměnu mezi různými informačními zdroji příliš užitečný.

Třetí možnost je založena na konzervativnějším přístupu – objektovém datovém modelu skupiny ODMG (Object Database Management Group), který je svojí strukturální složitostí podstatě XML velmi podobný. Rozšíření tohoto objektového návrhu ve směru k XML bylo realizováno např. v systému Ozone.

Poslední možnost přizpůsobuje semistrukturovaná data více XML. Patří sem XML-grafy, na kterých je vybudován dotazovací jazyk XML-QL (Query Language for XML).

Od modelu OEM k XML-grafům

Mezi modely semistrukturovaných dat má význačnější postavení OEM (Object Exchange Model). Data konstruovaná podle OEM nemají schéma a jsou samopopisná. Mohou být chápána jako orientovaný graf s označenými hranami. Uzly grafu jsou objekty, z nichž každý má jednoznačný identifikátor OID (Object Identifier). Atomické objekty nemají v grafu následníky a představují hodnoty daného typu. Ostatní objekty jsou složité objekty. Jména uzlů jsou speciálními aliasy objektů, mohou sloužit jako vstupní body do databáze. Příklad semistrukturované databáze (firma SOFTART) je na obr. 1.

Korespondence mezi OEM a XML je zřejmá. Objekty v OEM korespondují s elementy v XML, vztah "být podobjektem" v OEM je izomorfní s hnízděním elementů v XML. Rozdíly mezi oběma přístupy jsou v tom, že:

v XML jsou elementy uspořádané a mohou obsahovat atributy, tj. seznam dvojic jméno_atr:hodnota;

graf v OEM nemusí být stromem, tj. např. oboustranné vztahy mezi členy katedry a projekty se musí v XML explicitně specifikovat pomocí atributů ID (Identification) a IDREF (odkaz na jiný XML element přes jeho identifikační atribut);

XML dokumenty jsou obecnější, lze v nich míchat text a elementy, používat poznámky a další prvky (entity, zpracovatelné instrukce).

Část XML dokumentu korespondujícího s obr. 1 je zobrazena na obr. 2. OID &4 a &5 jsou v elementu ZAMĚSTNANEC hodnotami atributu PROJEKT (typu IDREFS). Směrem od projektů by některé OID byly hodnotami XML atributu ZAMĚSTNANEC (typu IDREFS nebo INDREF). Z obr. 1 se zdá, že tento atribut je nepovinný (IMPLIED). V praxi by se tyto atributy zřejmě nazývaly jinak, aby se nepletly se jmény elementů.

```
<zaměstnanec projekt= ”&4 &5”>
        <jméno>
              <křestní_jméno> Jiří </křestní_jméno>
              <příjmení> Dvorský </příjmení>
        </jméno>
        <adresa >
              <místo> Malostranské nám. 25, Praha 1 </místo>
              <PSČ> 118 00 </PSČ>
        </adresa>
</zaměstnanec>
```

Obr. 2. Element ZAMĚSTNANEC (firmy SOFTART)

Využít OEM pro XML znamená zavést do grafu vhodně atributy. Autoři jazyka XML-QL definují variantu OEM nazývanou XML-graf, což je orientovaný graf, jehož vnitřní uzly jsou reprezentovány pomocí OID. Hrany jsou ohodnoceny značkami elementů. Každý XML-graf má jeden uzel zvaný kořen, listy jsou označeny řetězci, které reprezentují hodnoty. Některé uzly XML-grafu jsou ohodnoceny množinou dvojic jméno_atr:hodnota. Z příkladu XML-grafu na obr. 3 je vidět, že kontaktní autor článku DBS je stejný jako autor knihy Databázová abeceda. OID je suplován buď hodnotou odpovídajícího ID atributu, nebo je generován systémem. Toho lze využít pro setření rozdílu mezi elementy a atributy typu IDREF(S) (viz obr. 4, kde však OID nejsou vyznačeny), což lze výhodně použít v dotazovacích jazycích na XML data.

XML-graf může být vytvořen na základě rozkladu XML dokumentu. Je patrné, že XML-grafy na obr. 3 a 4 jsou neuspořádané, tj. není určeno pořadí elementů tak, jak je to předepsáno v XML. Existuje ale i uspořádaná verze XML-grafu.

```
<!DOCTYPE Biblio [
<!ELEMENT Biblio(kniha | článek | monografie)*>
<!ELEMENT kniha (titul, autor)>
<!ELEMENT titul        (#PCDATA)>
<!ATTLIST kniha cena   CDATA #IMPLIED>
<!ELEMENT článek (název, autor*, kontaktní_autor)>
<!ELEMENT kontaktní_autor EMPTY>
<!ATTLIST kontaktní_autor autor_id IDREF #IMPLIED>
<!ELEMENT monografie (titul, autor, editor)>
<!ELEMENT titul        (#PCDATA)>
<!ELEMENT editor (monografie*)>
<!ATTLIST jméno_editora CDATA #REQUIRED>
<!ELEMENT autor (jméno, adresa)>
<!ATTLIST autor autor_id ID #REQUIRED>
<!ELEMENT jméno (křestní_jméno?, příjmení)>
<!ELEMENT křestní_jméno (#PCDATA)>
<!ELEMENT příjmení  (#PCDATA)>
<!ELEMENT adresa(místo, PSČ)>
<!ELEMENT místo      (#PCDATA)>
<!ELEMENT PSČ        (#PCDATA)>
]>
```

Obr. 5. DTD pro bibliografické záznamy

XML dokument popsaný XML-grafem na obr. 3 je validní vzhledem k DTD Biblio na obr. 5.

XML databáze

Databáze XML dat může vzniknout minimálně dvěma způsoby. V prvním z nich, říká se mu založený na datech, obsahuje databáze XML data, která pocházejí z databázových dat (budeme dále předpokládat, že jde o relační data). Jde o relativně jednoduchou situaci, nicméně mající různé možnosti řešení. Uvažujeme např. tabulky PROJEKTY(NÁZEV, ROZPOČET, ŘÍZEN) a ZAMĚSTNANCI(JMÉNO, ROD_Č, VĚK).

S jednou reprezentací (obr. 6) jsme se již setkali v prvním díle (Chip, srpen 2000). Data o zaměstnancích jsou umístěna “za” daty o projektech.

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

Jistě bychom našli další reprezentace, např. bez separátorů projektů (značka <projekt>) a zaměstnanců (značka <zaměstnanec>). Našly by se i sofistikovanější postupy s atributy ID a IDREF. Mohli bychom např. využít rodné číslo ne jako element, ale jako atribut typu ID a provázat projekty se zaměstnanci pomocí IDREF, podobně jako v příkladě s články a autory.

Existuje tedy více definic DTD, které popisují relační data. Problémy spojené s jejich automatickým generováním zahrnují otázky týkající se pořadí elementů, ošetření kolizí jmen apod. Nezanedbatelnou výhodou XML databází založených na datech však je, že v důsledku vlastností relačního modelu dat na uspořádání elementů příliš nezáleží.

Lze použít i speciálních šablon propojujících XML s SQL (obr. 8). Vyvoláním SQL příkazu je možné obdržet potřebný XML dokument. Pro XML dokumenty založené na datech ani obrácená transformace – zpět do relační databáze – není problémem.

Obtížnější situace nastávají s databázemi obecných XML dokumentů. O takových databázích se říká, že jsou založené na dokumentech. Převod těchto XML dat do relační databáze je dost složitý, i když zvládnutelný (někdy se ztrátou informace – např. uspořádání elementů). Mezi problémy, jak automaticky generovat schéma relační databáze z DTD, patří predikce typů dat, délek řetězců apod. Výhodu je, že k XML datům uloženým v relační databázi lze přistupovat pomocí SQL. Lépe řečeno, na XML data lze využít speciální dotazovací jazyk, jehož příkazy se překládají do příkazů SQL. Časově nejnáročnější je v tomto přístupu restrukturalizace XML dokumentů. K vygenerování eventuálně nového XML dokumentu z relačních dat asi nestačí běžná indexace tabulek a další techniky známé z relačních databází.

XML databáze založené na dokumentech zřejmě povedou k vývoji speciálních implementací. Jde o nové systémy řízení bází dat, které se nazývají anglicky Content Management Systems (např. systémy ASTORIA, Dynabase atd.). Umožňují manipulaci s fragmenty textu, řízení verzí, edici, publikační možnosti, separaci obsahu a stylu, rozšiřitelnost ve skriptování nebo programování, integraci dat z různých databází. Další variantou jsou implementace XML dat pomocí objektově relačních databází (viz ORACLE, INFORMIX a další).

Jaroslav Pokorný
pokorny@ksi.ms.mff.cuni.cz