XML Databáze

Petr Adámek

Obsah

XML Databáze

Co je to XML databáze

XML databáze je prakticky kolekce XML dokumentů uložených v nějakém úložišti.

Tímto úložištěm může být například

- souborový systém (nebo jiné perzistentní úložiště)
- relační databáze
- objektová databáze
- hierarchická nebo postrelační databáze
- nativní XML databáze

V přeneseném slova smyslu se pojmem XML databáze označují systémy řízení báze dat, které umožňují ukládat kolekce XML dokumentů (podobně jako se pojmem relační databáze nepesně označují systémy řízení báze dat pracující s relačním modelem). Někdy se také význam pojmu XML databáze omezuje pouze na nativní XML databáze.

Proč XML databáze

- XML je univerzální formát pro výměnu a reprezentaci informací.
- Transformace do jiných formátů je snadná.
- Ideální pro ukládání dokumentů.
- Univerzální formát pro datové modelování.
- Snadná a přirozená reprezentace objektů (agregace, dědičnost, asociace) - viz XML Schema.

Typickým případem kdy je vhodné použít XML databázi je situace, kdy pořizujeme nebo získáváme data přímo ve formátu XML a potřebujeme vzniklé dokumenty ukládat.

Kdy nepoužít XML databázi

- Když vyhovuje jiný datový model (většinou relační).
- U aplikací se specifickými požadavky (výkon).
- Když potřebujeme vlastnosti, které zatím běžně dostupné XML databáze neposkytují

XML databáze jsou zatím v plenkách

- Nedosahují robusnosti relačních databázových systémů (transakce, souběžný přístup, škálovatelnost).
- Chybí plná podpora standardů.
- Metody pro indexování a optimalizaci vyhodnocování dotazů se teprve vyvíjí.

XML databáze nejsou a nikdy nebudou následníkem relačních databází; tvoří k nim alternativu vhodnou pro určité typy dat a aplikací.

XML a relační databáze

XML dokumenty je možné ukládat v relačních databázích pomocí

- polí typu TEXT nebo BLOB
- specializovaných schémat pro konkrétní aplikace
- univerzálních schémat bez indexování struktury
- univerzálních schémat s indexováním struktury

Mnoho existujících relačních SŘBD poskytuje většinou proprietární podporu pro ukládání XML (rozšíření SQL).

Při indexování XML dokumentů nalézá uplatnění mnoho algoritmů a technik z relačních SŘBD.

Nativní XML databáze

Existuje řada nativních XML databází, mezi nejznámější patří

- Tamino [http://www.softwareag.com/tamino/News/tamino_41.htm], což je zřejmě nejznámější komerční nativní XML databáze.
- eXist [http://exist-db.org/], což je open source podporující mimo jiné XQuery [http://www.w3.org/XML/Query], XUpdate [http://xmldb-org.sourceforge.net/xupdate/], XML:DB API [http://xmldb-org.sourceforge.net/xapi/] a XML-RPC. Jde dle mého názoru o nejlepší existující open source databázi. Je šířena pod licencí GNU LGPL [http://www.gnu.org/copyleft/lesser.html].
- Apache Xindice [http://xml.apache.org/xindice/], cože je open source nativní XML databáze od Apache Software Foundation [http://www.apache.org/]. Podporuje XPath [http://www.w3.org/TR/xpath], XUpdate [http://xmldb-org.sourceforge.net/xupdate/], XML:DB API [http://xmldb-org.sourceforge.net/xapi/] a XML-RPC.

Přímo na fakultě Informatiky Masarykovy univerzity vzniká experimentální systém XIQE [http://www.xiqe.org/], jehož cílem je poskytnout platformu pro experimentální ověření vlastností různých indexačních metod. Tento systém není v současné době prakticky použitelný, nicméně jeho vývoj intenzivně probíhá a vývojový tým doufá, že se v dohledné době podaří implementovat všechny potřebné vlastnosti.

Indexování XML dokumentů

Efektivní ukládání a vyhledávání

Jak již bylo zmíněno, kolekce XML dokumentů se dají ukládat různým způsobem, které se liší zejména efektivitou různých operací. Ve všech zmíněných případech lze však aplikovat pomocné indexační struktury pro zvýšení jejich efektivity.

Indexování XML dokumentů umožňuje

- efektivní vyhledávání v kolekcích XML dokumentů,
- efektivní provádění XML transformací,
- efektivní aktualizaci dokumentů,
- efektivní navigaci v rámci dokumentu.

Nejčastěji je cílem indexování efektivní vyhodnocování XPath dotazů.

Indexování pro vyhodnocování XPath výrazů

- Indexování textových (hodnotových) informací
  - hodnoty textových uzlů;
  - hodnoty atributů;
  - jména elementů a atributů.
- Indexování strukturálních vztahů (osy XPath)
  - Vyhodnocení relace je na ose/není na ose (např. je uzel *x* potomkem uzlu *y*?);
  - Které uzly leží na dané ose (vrať mi všechny potomky uzlu *x*).

Pro indexování textových informací se používají invertované soubory (B+ strom, hešovací tabulky) nebo plnotextové indexy (fulltext). Pro indexování strukturálních vztahů byla vyvinuta řada metod, každá z nich má však nějaká slabá místa. Obvykle jsou tyto metody založeny na nějakém číslovacím schématu.

Vyhodnocování XPath dotazů

Pro vyhodnocení XPath dotazů je nutné

- vyhodnotit všechny predikáty a testy uzlů,
- vyhodnotit všechny strukturální vztahy,
- spojit (*join*) výsledky.

Pořadí operací může výrazně ovlivnit efektivitu zpracování.

Optimalizace XPath výrazů

- Zjednodušování a transformace dotazů
  - s využitím znalosti struktury dokumentu;
  - s využitím znalosti složitosti jednotlivých operací;
  - s využitím statistických informací;
  - eliminiace zbytečných predikátů.
- Výběr vhodného pořadí pro zpracování.

- Výběr vhodné metody pro vyhodnocení.

V současné době jsou bohužel schopnosti optimalizátorů v XML databázích omezené a nepříliš účinné.

Efektivita vyhodnocování XPath výrazů

Různě zapsané XPath/XQuery dotazy se stejným významem se mohou vyhodnotit různě efektivně. Dokonalý optimalizátor dotazů by měl toto eliminovat. Jenomže

- dokonalý optimalizátor (zatím?) neexistuje,
- mnoho implementací ani žádný nemá (XSLT procesory),
- optimizer mnohdy nemá k dispozici všechny informace.

Jak psát efektivní XPath dotazy?

- Pokud to není nezbytně nutné, efektivitu neřešte a dejte přednost přehlednosti.
- Efektivita jednotlivých operací silně závisí na konkrétní implementaci.
- Proto je pro dosažení efektivity vhodné znát způsob vyhodnocování XPath dotazů konkrétním XPath procesorem.
- Alternativou může být experimentální metoda.

Co je efektivnější?

- `/people/person` nebo `//person`
- `//person[@id=1]/name` nebo `//name[parent::*/@id=1]`
- `//person[number(@id)=$a]` nebo `//person[@id=$a]`
- `{ for $p in //person[@id=$a] return $p }` nebo `{ for $p in //person where $p/@id=$a return $p }`