

**Programátorská dokumentace
(experimentální metody ukládání XML do
relačních databází)
Petr Davídek**

Table of Contents

1.Úvodem (a věci obecné).....	3
2.Metoda XISS (XML Indexing and Storage systém).....	4
1.XISS.Import	4
2.XISS.xQuery	5
3.Metoda XpathAccelerator	6
1.Accel.Import	6
2.XPathAccelerator.xQuery	7
4.PathStack	8
1.PathStack.Import	8
2.PathStack.xQuery	9

1. Úvodem (a věci obecné)

Dokumentace popisuje implementaci vybraných metod ukládání XML dokumentů do relační DB. Dále popisuje implementaci xpath dotazů nad takto uloženými daty.

Implementované metody (dále jen **IM**) jsou tyto

1. XISS (adresář xiss)
2. XpathAccelerator (adresář accel)
3. PathStack (adresář path)

Pro detailnější informace o funkcích, metodách můžete použít přiloženou JavaDoc, nebo se můžete podívat přímo do kódu, je celkem bohatě komentován. Tato dokumentace si neklade za cíl vysvětlení principu používaných algoritmů, pouze popisuje jejich průběh. Pro detailní pochopení algoritmu si prosím prostudujte přiloženou práci.

Pro ukládání dat je použita relační DB Apache Derby - <http://db.apache.org/derby>. Jedná se o RDBMS napsaný v Javě. Každá z uvedených IM má svůj specifický DB model. Pro ukládání a dotazování je použito rozhraní JDBC. Pro ukládání dat do DB je použita třída preparedStatement, která by měla být lepší (ve smyslu rychlejší) při mnohonásobném vkládání dat do DB, protože ušetříme režii spojenou s voláním createStatement(). Při dotazování jsou použity metody createStatement a executeQuery.

Pokud vytváříme novou instanci DB <inst_name> (connect 'jdbc:derby:<inst_name>;create=true'), vznikne nám v aktuálním adresáři nový adresář s názvem databáze. V tomto adresáři lze pak nalézt všechny DB objekty dané DB instance.

Pro parsing XML dokumentů je použita implementace SAX od apache Xerces2, (<http://xerces.apache.org/xerces2-j>). Jedná se o implementaci SAX v Javě. Parser funguje následovně: čte proud dat na vstupu (XML) a pokud nastane událost, je vyvolána příslušná obslužná funkce. Parser hlídá, zda je dokument well-formed.

Při běhu parseru rozlišujeme 3 události (obsloužené 3 funkcemi):

1. začátek elementu - zavolá se metoda startElement (např. <kniha>). Tato metoda zároveň obsluhuje i atributy, které jsou dostupné jako pole párů „atribut:hodnota“
2. konec elementu - zavolá se metoda endElement (např. </kniha>)
3. přečtení textové hodnoty - zavolá se metoda characters (např. žlutá)

Tyto obslužné funkce se pro konkrétní implementaci IM (XISS, XPathAccelerator, PathStack) liší. Obecně to funguje tak, že se podědí třída DefaultHandler a v ní se tyto metody předefinují.

Každá z IM se dělí na 2 balíky - Import a xQuery:

1. Import čte XML dokumenty z daného adresáře, pomocí Xerces parseru je parsuje, vytváří si v paměti struktury, které následně uloží do DB. Pokud dojde k chybě při parsingu (document not well-formed), je tato

chyba vypsaná a pokračuje se dalším souborem. Parser nepoužívá validaci oproti schémátům (DTD, XS Scheme). To je zajištěno nastavením těchto features (česky fičury):
`xml.setFeature("http://apache.org/xml/features/nonvalidating/load-external-dtd" , false);`
`xml.setFeature("http://xml.org/sax/features/validation" , false);`

2. xQuery naparsuje zadaný xpath dotaz a transformuje ho na odpovídající select, který je následně proveden proti příslušnému datovému modelu.

2. Metoda XISS (XML Indexing and Storage systém)

Základem této metody je specifické kódování elementů, atributů a textových elementů. Kódováním rozumíme přiřazení číselných identifikátorů každému prvku XML dokumentu (prvkem XML míníme element/text/atribut). Na základě tohoto kódování je možno určit kdo je čí otec, kdo čí syn, potomek apod.

Základem použitého kódování je atribut order (unikátní ID každého prvku XML dokumentu, které se zvětšuje o +1 při každém přečteném prvku). Spolu s did (document_id) tvoří složený primární klíč tabulek attr_tab, elem_tab, text_tab. Dále se používá atribut depth (hloubka zanoření prvku v XML stromu), parent_id (order otce prvku), size (mohutnost prvku, tj. kolik obsahuje „podprvků“). Dále pak pomocné atributy child_id (order prvního potomka prvku), prev_id (order předchozího bratra – bratr je prvek, který má stejné parent_id (a stejnou depth)).

DATOVÝ model!!!!!!!!!!!!!!!!!!!!!!!!!!!!

1. XISS.Import

balík slouží pro nahrání XML souborů do DB instance xmldb_xiss

Balík obsahuje třídy

1. elem - třída pro ukládání vlastností elementů
2. text - třída pro ukládání vlastností textových řetězců
3. attr - třída pro ukládání vlastností atributů
4. pomzasstruct - pomocná „zásobníková“ třída pro odkládání odkazů na ještě ne úplně zpracovaných elementů.
5. XML_SAX_READER - třída implementující SAX parser (a obslužné metody pro zpracování XML)

Třídy elem, text, attr a pomzasstruct tvoří základ pro jim odpovídající zásobníky (realizováno pomocí typu Vector).

Funkce main hlavní třídy Import nejprve zavolá fci newconn, která se pokusí pomocí JDBC připojit do DB instance (v tomto případě se DB instance nazývá „xmldb_xiss“). Pokud tato DB ještě neexistuje, je vytvořena (parametr create=true). Instance DB umožňuje pouze single-

user připojení, tedy, je-li někdo k dané instanci připojen, naše připojení se nezdaří. Dále jsou postupně čteny soubory z daného adresáře a pro každý soubor se vytvoří nová instance třídy XML_SAX_READER, čímž se provede vlastní parsing XML dokumentu (myslím, že někde se o parsingu mluví jako o deserializaci, my budeme používat parsing).

V průběhu parsingu jsou rozlišovány 3 výše uvedené události.

1. Při vyvolání fce startElement se nejprve přečte vrchol zásobníku pomzastruct (kde máme uložený poslední nadřazený element), zjistí se z něj potřebné informace (child_id, parent_id) a založí se nová instance třídy elem a tato instance se uloží na konec vektoru elementů (proměnná elems). Poté se zpracují všechny atributy tohoto elementu a odkaz na tento element se uloží na vrchol zásobníku pomzasstruct. Atributy se zpracovávají obdobně – je vytvořena nová instance třídy attr a následně vložena na konec vektoru atributů (attrs).
2. Při vyvolání fce endElement je nejprve obsloužena potencionální textová hodnota (která se nám nahromadila v proměnné text). Ttext je prvek elementu podobný, jen nemá žádné potomky ani bratry ani atributy. V tomto případě je tedy založena nová instance třídy text a uložena na konec vektoru texts. Následně pak je vyzvednut vrchol zásobníku pomzasstruct a dopočítán atribut size a takto upravená hodnota je promítnuta do elems.
3. Při vyvolání fce characters se pouze upravuje proměnná text. Pro dokončení tohoto parseru máme tedy vyplněné vektory elems, attrs, texts. Nyní už je pouze projdeme a uložíme do připravených DB struktur. Pro ukládání využíváme třídu preparedStatement. Commit děláme po 1000 záznamech. Pro soubor zároveň vypíšeme statistiku toho, jak dlouho která operace trvala. A nakonec vypíšeme sumáž celého importu.

2. XISS.xQuery

balík slouží pro dotazování XML dokumentů uložených v DB instanci xmldb_xiss

Balík obsahuje tyto třídy

1. Query – slouží pro uložení fragmentů xpath dotazu (fragmenty jsou zde ty části, které vzniknou rozdělením xpath dotazu podle znaků '/' a '//').
2. Elem – slouží pro parsing jednotlivých fragmentů a k výpočtu hodnot, které budou použity pro složení finálního sql dotazu
3. xQuery – hlavní třída, která to vše řídí

Jediným vstupem procedury main třídy xQuery je xpath dotaz v uvozovkách. Procedura xpath dotaz „rozseká“ na fragmenty nejprve podle '/' a tyto fragmenty dále podle '/'. Výsledné fragmenty jsou uloženy ve vektoru query ve formě instancí třídy Query. Tento vector je následně zpracováván jako fronta fragmentů. Pro každý fragment je vytvořena nová instance třídy Elem a zavolán parser (parse_elem).

Parse_elem fragment nejprve rozdělení podle '::', následně se kontroluje,

zda fragment obsahuje znaky [] (a tedy výsledný sql dotaz bude obsahovat textovou hodnotu), zda fragment obsahuje znak @ (a tedy výsledný sql dotaz bude obsahovat atribut, případně atribut s hodnotou). Za základně zjištěných informací se upravují hodnoty et, at, tt (počty tabulek, které se zúčastní sql dotazu) a zároveň se (podle typu osy) skládá i where podmínka. Pokud budeme chtít rozšířit množinu xpath dotazů, které metoda umí, upravíme příslušně tuto funkci (parse_elem).

Skládání where podmínky mají na starosti 2 metody třídy xQuery - get_where_et a get_where_at. Pokud byste chtěli si pohrát s optimalizací výsledného sql dotazu, tak v těchto 2 fcích doporučuji se rozhlédnout.

Máme-li již zpracovány všechny fragmenty, poskládáme předmět selectu, from část a where část a máme výsledný sql dotaz (select). Připojíme se do DB (xmldb_xiss) a dotaz pustíme. Pak už si jen dokola fetchujeme výsledky a vypisuje na obrazovku. Na závěr pak vypíšeme zajímavé časové údaje.

3. Metoda XpathAccelerator

Základem této metody je opět specifické kódování elementů, atributů a textových elementů. Kódováním rozumíme přiřazení číselných identifikátorů každému prvku XML dokumentu. Na základě tohoto kódování je možno určit kdo je čí otec, kdo čí syn, potomek apod. Metoda je velmi podobná metodě XISS. Na rozdíl od předchozí metody XISS je zde použito tzv. pre/post okénko a prvky jsou uloženy v jedné tabulce (accel_main). Pro pre/post okénko platí toto: prvek b je potomkem prvku a, jestliže pre(a)<pre(b)& post(b)<post(a). Při přečtení počátku prvku je zvětšen atribut pre, při přečtení konce zase atribut post. Zároveň počítáme atribut height, který udává hloubku podstromu daného elementu. Atribut pre spolu s did tvoří složený primární klíč tabulky accel_main. (Klidně by ale klíč mohl být post, did - klíče jsou si ekvivalentní). Dále si ještě udržujeme atribut par - otec prvku.

DATOVÝ model!!!!!!!!!!!!!!!!!!!!!!!!!!!!

1. Accel.Import

balík slouží pro nahrání XML souborů do DB instance xmldb_accel

Balík obsahuje 3 třídy

1. elem - třída pro ukládání vlastností prvků (do paměti)
2. dbmanip - třída pro ukládání prvků (instancí třídy elem) do DB
3. XML_SAX_READER - třída implementující SAX parser (a obslužné metody pro zpracování XML)

Fukce main hlavní třídy Import (po kontrole vstupních parametrů) vytvoří instanci třídy dbmanip a vytvoří připojení do DB. Pokud se připojení nezdařilo (např. někdo jiný je do naší instance připojen), je program ukončen. Instance DB se v tomto případě jmenuje

„xmldb_accel“. Následně dropne existující tabulky (manip.dropold) a vytvoří nové (manip.crenew). Dále čte postupně soubory z daného adresáře a pro každý soubor vytvoří novou instance třídy XML_SAX_READER. O zbytek (míněno parsing a uložení dat do DB) se pak už stará pouze tato třída. Stejně jako v minulé IM máme i v této třídě předefinovány tyto 3 metody:

1. Při vyvolání fce startElement se vytvoří nová instance třídy elem – odpovídá novému elementu. Pokud má element nějaké atributy, použije se na jejich zpracování funkce make_leaf. Ta zpracovává nejen atributy, ale i textové hodnoty (viz dále). Nově vzniklý element je uložen na zásobník elementů. To je nezbytné kvůli dopočítávání atributů height.
2. Při vyvolání fce endElement se nejprve obslouží možný text. Jeho hodnotu máme uloženou v proměnné text. Na tuto hodnotu se zavolá fce make_leaf (vrátí nám hloubku „podstromu“ – v tomto případě vždy 1 – jedná se přeci o textovou hodnotu a ta již potomka nemá). Pak je vyzvednut vrchol zásobníku elems a dopočítán atribut post. Takto upravený elem je uložen do DB (zavolá se db.insert()) a výška tohoto elementu je propagována do otcovského elementu (na aktuální vrchol zásobníku elems).
3. Při vyvolání fce characters se pouze staví proměnná text.
- a 4 pomocná - make_leaf
4. Make_leaf slouží k zpracování (a ukládání) listů. Listy jsou v našem případě textové hodnoty nebo atributy (případně s hodnotou). Fce vrací hloubku uloženého stromu. V případě textové hodnoty je to vždy 1, v případě atributu bez hodnoty také 1, v případě atributu s hodnotou 2. Fce neudělá nic jiného, než že vytvoří novou instanci třídy elem a vloží ji do DB. (db.insert()).

Pro každý soubor zároveň vypíšeme statistiku toho, jak dlouho která operace trvala. A nakonec vypíšeme sumáž celého importu.

2. XPathAccelerator.xQuery

balík slouží pro dotazování XML dokumentů uložených v DB instanci xmldb_accel

Balík obsahuje tyto třídy

1. Query – slouží pro uložení fragmentů xpath dotazu (fragmenty jsou zde ty části, které vzniknou rozdělením xpath dotazu podle znaků '/' a '//').
2. Elem – slouží pro parsing jednotlivých fragmentů a k výpočtu hodnot, které budou použity pro složení finálního sql dotazu
3. xQuery – hlavní třída, která to vše řídí

Jediným vstupem procedury main třídy xQuery je xpath dotaz v uvozovkách. Stejně jako v minulém případě procedura xpath dotaz „rozseká“ na fragmenty nejprve podle '/' a tyto fragmenty dále podle '//'. Výsledné fragmenty jsou uloženy ve vektoru query ve formě instancí třídy Query. Tento vektor je následně zpracováván jako fronta

fragmentů. Pro každý fragment je vytvořena nová instance třídy Elem a zavolán parser (parse_elem).

Parse_elem fragment nejprve rozdělí podle '::', následně se kontroluje, zda fragment obsahuje znaky [] (a tedy výsledný sql dotaz bude obsahovat textovou hodnotu), zda fragment obsahuje znak @ (a tedy výsledný sql dotaz bude obsahovat atribut, případně atribut s hodnotou). Za základně zjištěných informací se upravuje hodnoty et (počet tabulek, které se zúčastní sql dotazu) a zároveň se (podle typu osy) skládá i where podmínka. Pokud budeme chtít rozšířit množinu xpath dotazů, které metoda umí, upravíme příslušně tuto funkci (parse_elem).

Skládání where podmínky má starosti metoda get_where_et. Pokud byste chtěli si pohrát s optimalizací výsledného sql dotazu, tak nejlépe zde.

Máme-li již zpracovány všechny fragmenty, poskládáme předmět selectu, from část a where část a máme výsledný sql dotaz (select). Připojíme se do DB (xmldb_accel) a dotaz pustíme. Pak už si jen dokola fetchujeme výsledky a vypisuje na obrazovku. Na závěr pak vypíšeme zajímavé časové údaje.

4. PathStack

Základem této metody je opět specifické kódování elementů a textových elementů. Kvůli zjednodušení rezignujeme na zpracování atributů.

Kódováním rozumíme přiřazení číselných identifikátorů každému prvku XML dokumentu. Na základě tohoto kódování je možno určit kdo je čí otec, kdo čí syn, potomek apod. Podobně jako u předchozí metody XPathAccelerator je zde použito kódování pomocí 3 atributů - leftp, rightp a level, a prvky jsou uloženy v jedné tabulce (twig_main). Rozdíl oproti XPathAccelerator je v tom, že hodnota leftp a rightp je brána ze stejné sekvence a navíc se tato sekvence nenuluje při novém dokumentu. Platí následující: prvek d je potomkem prvku c, jestliže $\text{leftp}(c) < \text{leftp}(d) < \text{rightp}(d) < \text{rightp}(c)$. Atribut level udává hloubku zanoření elementu. Při přečtení počátku prvku je zvětšen atribut leftp, při přečtení konce zase atribut rightp. Zároveň počítáme atribut level podle úrovně zanoření v XML stromu. Atribut leftp tvoří primární klíč tabulky twig_main. (Opět by klíčem mohl být rightp klíče jsou si ekvivalentní).

DATOVÝ model!!!!!!!!!!!!!!!!!!!!!!!!!!!!

1. PathStack.Import

balík slouží pro nahrání XML souborů do DB instance xmldb_twig

Balík obsahuje 3 pomocné třídy

1. elem - třída pro ukládání vlastností prvků (do paměti)
2. dbmanip - třída pro ukládání prvků (instancí třídy elem) do DB
3. XML_SAX_READER - třída implementující SAX parser (a obslužné

metody pro zpracování XML)

Funkce main hlavní třídy Import (po kontrole vstupních parametrů) vytvoří instanci třídy dbmanip a vytvoří připojení do DB. Pokud se připojení nezdařilo (např. někdo jiný je do naší instance připojen), je program ukončen. Instance DB se v tomto případě jmenuje „xmldb_accel“. Následně dropne existující tabulky (manip.dropold) a vytvoří nové (manip.crenew). Dále čte postupně soubory z daného adresáře a pro každý soubor vytvoří novou instance třídy XML_SAX_READER. O zbytek (míněno parsing a uložení dat do DB) se pak už stará pouze tato třída. Stejně jako v minulé IM máme i v této třídě předefinovány tyto 3 metody:

1. Při vyvolání fce startElement se vytvoří nová instance třídy elem - odpovídá novému elementu. Nově vzniklý element je uložen na zásobník elementů. To je nezbytné kvůli dopočítávání atributu rightp.
2. Při vyvolání fce endElement se nejprve obslouží možný text. Jeho hodnotu máme uloženou v proměnné text - uloží se do DB. Pak je vyzvednut vrchol zásobníku elems a dopočítán atribut rightp. Takto upravený elem je uložen do DB (zavolá se db.insert()).
3. Při vyvolání fce characters se pouze staví proměnná text.

Pro každý soubor zároveň vypíšeme statistiku toho, jak dlouho která operace trvala. A nakonec vypíšeme sumáž celého importu.

2. PathStack.xQuery

balík slouží pro dotazování XML dokumentů uložených v DB instanci xmldb_twig

Balík obsahuje tyto třídy

1. Stack - slouží pro ukládání pozic prvků dokumentu na zásobník
2. QueryNode - slouží pro uložení elementů xpath dotazu, jejich zásobníků, datastreamů a dalších vlastností
3. xQuery - hlavní třída

Tato dotazovací metoda se oproti minulým o něco liší. Rozdíl spočívá v tom, že je využito DB engine pouze pro vrácení záznamu pro jednotlivé elementy, „join“ těchto záznamů si řeší algoritmus sám.

Jediným vstupem procedury main třídy xQuery je xpath dotaz v uvozovkách. Xpath je opět rozsekán podle řetězců '/', '[]', 'and'. Pro každý takto vzniklý fragment je založena instance objektu QueryNode. Tím nám vznikne strom QueryNodů - QueryTree(QT). Při založení nové instance QueryNode dojde mimo jiné k otevření kurzoru, který odpovídá názvu elementu daného QueryNoda - přiřadíme k němu tzv. datastream (proud dat, který může být výsledkem xpath dotazu).

Máme-li takto vytvořený QT, začneme ho zpracovávat v cyklu. Cyklus končí, pokud všechny zásobníky na listech QT jsou prázdné. V každém kroku nejprve vybereme takový QN, který má minimální leftp na vrcholu

svého zásobníku (getMinSource). Poté vyčistíme všechny zásobníky od pozic, které nemohou být ve výsledku vypsány. Dále u vybraného QN přesuneme na vrchol zásobníku aktuální hodnoty leftp, rightp a docid. A posuneme se v otevřeném kurzoru vpřed. Pokud jako minimální QN vybereme nějaký list, pokusíme se vypsát řešení dotazu (printResults) a zahodíme vrchol zásobníku.