

**Instalační a uživatelská dokumentace
(experimentální metody ukládání XML do
relačních databází)
Petr Davídek**

Table of Contents

1.Úvodem (tux jede).....	3
2.Howto install	3
3.Howto set enviroment.....	3
4.Howto compile	3
5.Howto run Import.....	4
6.Howto run xQuery.....	4
7.Howto test it	4
8.Opakování pokusů.....	5
9.Hromadné operace.....	5
10.Příklady xpath dotazů, které by metody měly umět.....	6
11.Adresáře results.....	8

1. Úvodem (tux jede)

Nejprve se musím omluvit všem uživatelům M\$Windows – asi vám to nebude fungovat. Rozhodně ne úplně všechno. Překlad, testování (import, dotazování) je děláno pomocí bash skriptů v nějakém terminálu. Doporučuji co největší rozlišení obrazovky, aby se do terminálu pěkně vešly veškeré výstupy. Vyvíjeno a testováno na Debian GNU/Linux lenny/sid. Potřebné GNU utility (jistě budou součástí naší distribuce) – sed, tee, diff, awk, tail, head, bc. Není k dispozici žádný configure script, protože se předpokládá, že všechny tyto utility na svém linuxu máte. Pokud ne, apt-get install, pkg-get, yum install :).

Obsah CD (adresáře)

1. docs - programátorská dokumentace, tato dokumentace, bc práce
2. xiss, accel, path - jednotlivé testované metody zpracování XML, každá obsahuje adresáře :
 1. docs - javadoc dokumentace
 2. test - pro testování referenčních xpath dotazů oproti referenčním XML dokumentům
 3. results - naměřené výsledky (ukládání XML dat a dotazování nad nimi)
3. java - obsahuje instalaci JDK 1.5.11
4. derby - obsahuje instalaci derby DB ve verzi 10.2.0
5. lib - přídavné jar knihovny (zatím pouze xerces)

2. Howto install

Stačí nakopírovat obsah přiloženého CD někam na filesystem a v souboru .derby.profile (nalezneme ho v rootu CD) změním obsah proměnné INSTALL_PATH. Ta udává místo na filesystemu, kam jsme CD nakopírovali.

3. Howto set enviroment

Pokud chceme překládat javovské zdrojáky, nebo chceme pouštět Import souborů či xpath dotazy, musíme mít nejprve správně nastavené prostředí. K tomu nám slouží právě soubor .derby.profile. Měl by být spustitelný (pokud není, přidáme právo +x). Soubor spustíme (. .derby.profile), nastaví nám proměnné shellu PATH, JAVA_HOME, DERBY_HOME, CLASSPATH. Můžeme zkontrolovat pomocí příkazu env.

4. Howto compile

Je třeba mít nastavený enviroment (prostředí). Podle toho, se kterou metodou si chceme hrát se vystavíme do jejího adresáře. Pro překlad Importu pustíme příkaz

```
javac Import.java
```

Pro překlad xpath dotazovače

```
javac xQuery.java
```

Výsledkem je, že se nám v adresáři vytvoří Java binárky (classy). Co třída ve zdrojáku, to jedna samostatná classa. Jsme-li líní máme připraven skript compile.sh, který udělá vše za nás. Pustíme ho následovně:

```
./compile.sh
```

5. Howto run Import

Opět je třeba mít nastavené prostředí, vystavíme se do příslušného adresáře. Pro import xml souborů do DB pustíme příkaz

```
java Import <path_to_xml_dir>
```

kde `path_to_xml_dir` je adresář s XML soubory (soubory v adresáři musí mít příponu XML, xml). O průběhu importu jsme informováni statistikami. Při prvním spuštění Importu vznikne v adresáři podadresář `xmldb_<metod_name>` (podle toho, s jakou metodou si hrajeme - `xmldb_accel`, `xmldb_xiss`, `xmldb_twigg`). Tento adresář bude obsahovat všechny DB objekty včetně datafilů.

6. Howto run xQuery

Opět je třeba mít nastavené prostředí, vystavíme se do příslušného adresáře příslušné metody. Pokud ještě neexistuje podadresář `xmldb_<metod_name>`, je nejprve třeba nainportovat nějaká XML data - jinak se nemáme čeho ptát. Pro spuštění xpath dotazu použijeme příkaz

```
java xQuery "<xpath dotaz>"
```

A už si jen počkáme na výsledek. Výsledkem je formátovaný výstup o 3 sloupcích - název souboru, výsledek xpath dotazu a interní identifikátory elementů.

7. Howto test it

Pokud jsme např. přidali novou funkcionalitu nebo pouze čistily kód, je dobré si ověřit, že jsme nenapáchali nic špatného. K tomu nám slouží adresář test. Obsahuje adresář xml (s několika malými xml soubory), soubor `xpath.example` (vybrané testovací xpath dotazy) a referenční soubor `reffile` (ten obsahuje výsledky). Pokud tedy chceme provést tento test, pustíme následující příkaz:

```
./make_test.sh
```

Ten nejprve naimportuje referenční XML data do DB a poté spustí testovací sadu xpath dotazů, výsledky si poznamená do pomocného souboru a nakonec si diffne referenční soubor s pomocným. Pokud budou oba stejné, je celkem velká šance, že jsme nic nepokazili. Pokud jsou soubory stejné, odpovědí je „Seems OK“, jinak diff výstup rozdílů (see man diff).

8. Opakování pokusů

Pro přesnější měření časů (a do budoucna možná i pro nějaká statistická data) je dobré, pokud stejný pokus opakujeme vícekrát. Pokusem zde myslíme jeden Import adresáře (v případě importu) a jedno xQuery (v případě dotazů) do připravené DB. K účelu máme skript (v adresáři příslušné metody)

```
./make_import.sh <xml_dir> [count]
```

kde xml_dir je cesta k adresáři s XML dokumenty a count je počet opakování pokusu. Pokud není count zadáno, bere je defaultně 1. Na základě tohoto příkazu je count-krát proveden import XML souborů do DB a výsledek importu je zapsán do adresáře results do souboru <xml_dir>.import.<iterace>. Na závěr je všech pokusů vypočítána průměrná doba běhu importu.

Pro hromadné dotazování si musíme vytvořit soubor s xpath dotazy. Každý xpath dotaz začíná na nové řádce a je obalen znakem uvozovka (např. "//@*"). Potom můžeme sekvenci dotazů pustit několikrát proti DB

```
./make_xpath.sh <xpath_file> [count]
```

kde xpath_file je cesta k souboru, který obsahuje xpath dotazy. Pokud není count zadáno, bere je defaultně 1. Na základě tohoto příkazu jsou count-krát provedeny všechny xpath dotazy ze souboru. Výstupem jsou časové údaje běhu dotazů.

9. Hromadné operace

Pro naše větší pohodlí máme připraveny jednoduché skripty, které dělají něco pro všechny implementované metody. Jsou umístěny v rootu CDčka. Pokud chceme přeložit všechny java classy, pustíme

```
./compile_all.sh
```

Pokud chceme naimportovat XML soubory pomocí všech metod, pustíme

```
./make_import_all.sh <xml_dir> [count]
```

kde xml_dir je adresář s XML dokumenty a count je počet opakování pokusu. Pokud není count zadáno, bere je defaultně 1. O průběhu importu jsme informováni výpisem na obrazovku a zároveň jsou výsledky ukládány do adresářů results příslušných metod.

Pokud chceme provést sadu xpath dotazů oproti všem DB (xiss, accel, twig), pustíme

```
./make_xpath_all.sh <xpath_file> [count]
```

kde `xpath_file` je cesta k souboru, který obsahuje xpath dotazy. Pokud není `count` zadáno, bere je defaultně 1. Na základě tohoto příkazu jsou countkrát provedeny všechny xpath dotazy ze souboru oproti všem DB. Výstupem jsou časové údaje běhu dotazů.

10. Příklady xpath dotazů, které by metody měly umět

Seznam je možné nalézt v adresáři `test` v souboru `xpath.example`

1. Xiss - umí pracovat s osami `parent` (otec elementu), `child` (též `/`) (syn elementu), `descending` (též `//`) (všechny potomci elementu), `ascending` (všichni předci elementu), `preceding` (všechny elementy, které jsou (při sekvenčním čtení dokumentu) přečteny před elementem), `following` (všechny elementy, které jsou (při sekvenčním čtení dokumentu) přečteny po elementu), `preceding-sibling` (všechny elementy, které jsou (při sekvenčním čtení dokumentu) přečteny před elementem a zároveň jsou bratry elementu), `following-sibling` (všechny elementy, které jsou (při sekvenčním čtení dokumentu) přečteny po elementu a zároveň jsou bratry elementu). Umí se dotazovat na atributy (`@`, `::attribute()`), elementy (`::node()`), textové hodnoty (`::text()`). Umí používat wild `*`, která matchuje všechno.

Příklady:

```
"/@*"
"/a/b/descendant::node()"
"/***/parent::node()"
"/a*/parent::node()/b"
"/a*/**/*"
"/a*/@*"
"/@*"
"/a[@*=aaaa]"
"/**[@*=*]"
"/**[@*=*]*/h"
"/a/**[@*=*]/*"
"/**[@atta=*]"
"/i/preceding::node()"
"/b/following::node()"
"/e/preceding::node()"
"/f/preceding::node()"
"/j/preceding-sibling::attribute()"
"/j/preceding-sibling::text()"
"/**/following-sibling::attribute()"
"/**/following-sibling::text()"
"/descendant-or-self::node()/parent::node()"
"/f/descendant-or-self::node()"
"/f/descendant-or-self::node()/ancestor-or-self::node()"
```

```
"/[*[*=*]/*"  
"/[*[i=elmeIII]/*"
```

2. Accel – umí pracovat s osami parent (otec elementu), child (těž /) (syn elementu), descending (těž //) (všechny potomci elementu), ascending (všichni předci elementu), preceding (všechny elementy, které jsou (při sekvenčním čtení dokumentu) přečteny před elementem), following (všechny elementy, které jsou (při sekvenčním čtení dokumentu) přečteny po elementu), preceding-sibling (všechny elementy, které jsou (při sekvenčním čtení dokumentu) přečteny před elementem a zároveň jsou bratry elementu), following-sibling (všechny elementy, které jsou (při sekvenčním čtení dokumentu) přečteny po elementu a zároveň jsou bratry elementu). Umí se dotazovat na atributy (@, ::attribute()), elementy (::node()), textové hodnoty (::text()). Umí používat wild *, která matchuje všechno.

Příklady:

```
"//@*"  
"/a/b/descendant::node()"  
"/[*]*/parent::node()"  
"/a/*/parent::node()/b"  
"/a/*/*/*"  
"/a/*/@*"  
"/@*"  
"/a[@*=aaaa]"  
"/[*[@*=*]"  
"/[*[@*=*]*/h"  
"/a/*[@*=*]/*/*"  
"/[*[@atta=*]"  
"/i/preceding::node()"  
"/b/following::node()"  
"/e/preceding::node()"  
"/f/preceding::node()"  
"/j/preceding-sibling::attribute()"  
"/j/preceding-sibling::text()"  
"/[*]*/following-sibling::attribute()"  
"/[*]*/following::attribute()"  
"/descendant-or-self::node()/parent::node()"  
"/f/descendant-or-self::node()"  
"/f/descendant-or-self::node()/ancestor-or-self::node()"  
"/[*[*=*]/*"  
"/[*[i=elmeIII]/*"
```

3. Path – umí pracovat pouze s osami child (/) a descendant (//). Práce s textovými elementy je rozpracovaná, ale ještě nefunkční (parser dotaz rozpozná, ale neumí ho správně vyhodnotit)

Příklady:

```
"//*"  
"/*//*"  
"/a/b"  
"/a/*/b"
```

```
"/a/*/*/*"  
"//a/*/*"  
"//j/*"  
"//a//b//c"
```

11. Adresáře results

Zde jsou výsledky měření importu a dotazu. Názvy souborů odpovídají názvům adresářů na DVD s testovacími daty, přípona .import udává, že se jednalo o import, .query, že o dotaz. Poslední přípona (číslo) udává pořadové číslo pokusu. V případě xpath dotazů má soubor query 2 číselné přípony – první udává pořadové číslo xpath dotazu v souboru dotazů, druhé pak pořadové číslo pokusu