Zadání úlohy do projektu z předmětu IPP 2011/2012

Zbyněk Křivka a Dušan Kolář
E-mail: {krivka, kolar}@fit.vutbr.cz, {54 114 1313, 54 114 1238}

XQR: XML Query

Zodpovědný cvičící: Martin Čermák (icermak@fit.vutbr.cz)

1 Detailní zadání úlohy

Skript provádí vyhodnocení zadaného dotazu, jenž je podobný příkazu SELECT jazyka SQL, nad vstupem ve formátu XML. Výstupem je XML obsahující elementy splňující požadavky dané dotazem. Dotazovací jazyk má zjednodušené podmínky a syntaxi.

Tento skript bude pracovat s těmito parametry:

- `--help` viz společné zadání všech úloh
- `--input=filename` zadaný vstupní soubor ve formátu XML
- `--output=filename` zadaný výstupní soubor ve formátu XML s obsahem podle zadaného dotazu
- `--query='dotaz'` zadaný dotaz v dotazovacím jazyce definovaném níže (v případě zadání tímto způsobem nebude dotaz obsahovat symbol apostrof)
- `--qf=filename` dotaz v dotazovacím jazyce definovaném níže zadaný v externím textovém souboru (nelze kombinovat s `--query`)
- `-n` negenerovat XML hlavičku na výstup skriptu
- `--root=element` jméno párového kořenového elementu obalující výsledky. Pokud nebude zadán, tak se výsledky neobalují kořenovým elementem, ač to porušuje validitu XML.

Dotazovací jazyk Nejprve uveďme neformální zápis struktury dotazovacího jazyka:

```
SELECT element LIMIT n FROM element|element.attribute|ROOT WHERE condition
                      ORDER BY element|element.attribute ASC|DESC
```

Celá bezkontextová gramatika (včetně povolených rozšíření) je definována takto (neterminály jsou v úhlových závorkách, `<QUERY>` je startující neterminál, tokeny jsou odděleny bílým znakem (je-li to nezbytné) a jazyk je case-sensitive[1]):

```
<QUERY> --> SELECT element <LIMITn> FROM <FROM-ELM> <WHERE-CLAUSE> <ORDER-CLAUSE>
<LIMITn> --> empty
<LIMITn> --> LIMIT number
<FROM-ELM> --> <ELEMENT-OR-ATTRIBUTE>
<FROM-ELM> --> ROOT
<WHERE-CLAUSE> --> empty
```

[1] *case-sensitive* znamená, že v dotazech záleží na velikosti písmen u klíčových slov i u identifikátorů.

```
<WHERE-CLAUSE> --> WHERE <CONDITION>
<CONDITION> --> ( <CONDITION> )
<CONDITION> --> NOT <CONDITION>
<CONDITION> --> <CONDITION> AND <CONDITION>
<CONDITION> --> <CONDITION> OR <CONDITION>
<CONDITION> --> <ELEMENT-OR-ATTRIBUTE> <RELATION-OPERATOR> <LITERAL>
<LITERAL> --> string
<LITERAL> --> number
<RELATION-OPERATOR> --> CONTAINS
<RELATION-OPERATOR> --> =
<RELATION-OPERATOR> --> >
<RELATION-OPERATOR> --> <
<ELEMENT-OR-ATTRIBUTE> --> element
<ELEMENT-OR-ATTRIBUTE> --> element.attribute
<ELEMENT-OR-ATTRIBUTE> --> .attribute
<ORDER-CLAUSE> --> empty
<ORDER-CLAUSE> --> ORDER BY <ELEMENT-OR-ATTRIBUTE> <ORDERING>
<ORDERING> --> ASC
<ORDERING> --> DESC
```

Ke splnění základního zadání není nutné podporovat klauzuli `ORDER BY` a skládání podmínek pomocí klíčových slov `AND` a `OR`. Jednoduchou negaci podmínky pomocí klíčového slova `NOT` považujte za součást základního zadání.

Lexémy jsou definovány takto: `empty` je prázdný řetězec. `number` je celé číslo v běžném 32-bitovém celočíselném rozsahu implementačního jazyka. `element` resp. `attribute` jsou identifikátory elementu resp. atributu jazyka XML (bez ohraničujících znaků `<` a `>`). `string` je řetězec zapsaný v uvozovkách, který neobsahuje žádné netisknutelné znaky, escape sekvence, konec řádku, ani uvozovky (nebude testováno).

Sémantika dotazovacího jazyka: Dotaz v klauzuli `FROM` definuje zdrojový element (viz neterminál `<FROM-ELM>`), kde se následně hledají vnořené výstupní elementy z klauzule `SELECT`, které splňují podmínky dané v klauzulí `WHERE`. Poté může být výsledný seznam elementů seřazen klauzulí `ORDER BY` a ořezán omezením `LIMIT` na požadovaný maximální počet elementů.

Hledání zdrojového elementu provádějte hledáním do hloubky, dokud nenarazíte na první výskyt zdrojového elementu dle následujících podmínek pro klauzuli `FROM`:

- Pokud je klauzule tvaru `FROM element`, je hledán první výskyt elementu `element`.
- Pokud je klauzule tvaru `FROM element.attribute`, je hledán první výskyt elementu `element` obsahující atribut `attribute`.
- Pokud je klauzule tvaru `FROM .attribute`, je hledán první element obsahující atribut `attribute`.

Teprve zde bude prováděno další zpracování (již se neuvažuje další nepřekrývající zdrojový element jinde na vstupu). Vzájemné zanoření totožných výstupních elementů neřešte. Nicméně uvažujte, že výstupní element, může být pokaždé zanořen do jiné úrovně ve zdrojovém elementu (nepřekrývajícím způsobem). Klíčové slovo `ROOT` zastupuje virtuální kořenový element zastupující celý XML

dokument, který pak obsahuje skutečný kořenový element. Entita (element nebo atribut elementu) z podmínky v klauzuli `WHERE` se hledá opět do hloubky po první svůj výskyt elementu dle stejných vlastností pro výběr vhodné entity jako v případě hledání zdrojového elementu s tím rozdílem, že `.attribute` a `element.attribute` vrací hodnotu atributu, nikoliv elementu. Není-li hledaná entita v aktuálně kontrolovaném výstupním elementu nikde nalezena nebo je její první výskyt špatného typu, je výsledek porovnání (viz neterminál `<RELATION-OPERATOR>`) nepravdivý.

Výstupní elementy jsou na výstup kopírovány v nezměněné podobě (včetně všech atributů, hodnot i podelementů[2]), případně obaleny kořenovým elementem v závislosti na parametrech skriptu.

V případě kolize jmen atributů či elementů uvažujte první načtený.

Pokud bude v dotazu syntaktická nebo sémantická chyba, tak ukončete skript s chybovou hláškou vypsanou na standardní chybový výstup a vraťte návratový kód 80.

Příklady k definici sémantiky dotazovacího jazyka:

```
SELECT book FROM library WHERE title CONTAINS "XML"
```

Projdi všechny elementy `<book>` v elementu `<library>` a vyber ty knihy, které v prvním podelementu `<title>` obsahují podřetězec `XML` (case-sensitive[3]). Pokud `<title>` obsahuje další elementy místo textové hodnoty, tak skonči s chybou. Vybrané knihy vypisujte na výstup jako kopie vybraných elementů `<book>` včetně všech atributů a podelementů (až na formátování). Podle argumentů skriptu případně doplňte párový kořenový element a XML hlavičku.

```
SELECT book LIMIT 8 FROM library WHERE title CONTAINS "Duna" AND
                                 (.name = "Frank Herbert" OR year > 1980)
                           ORDER BY year DESC
```

Vypište XML elementy (vč. podelementů a atributů) pro 8 knih, jejichž název obsahuje podřetězec `Duna` a jejichž autor uvedený v libovolném prvně načteném atributu `name` uvnitř libovolného elementu uvnitř elementu `<book>` (v libovolném zanoření, tedy i přímo v elementu `<book>`) je `Frank Herbert` nebo rok (element `<year>`) je větší jak číslo `1980`. Výsledek seřaďte sestupně podle obsahu elementu `<year>` a až poté vyber prvních 8 knih (dáno klauzulí `LIMIT`). Pokud dojde ke kolizi jmen například atributu `name` ve více elementech, tak uvažujte první nalezený atribut v elementu `<book>` nebo jeho podelementech. Relační operátor pracující s neexistujícím (nedefinovaným) elementem nebo atributem vždy vrací nepravdivou hodnotu.

```
SELECT title FROM library WHERE NOT title CONTAINS "Duna"
```

Tento dotaz vypíše pomocí XML všechny názvy knih z knihovny, které neobsahují v názvu řetězec `Duna`.

```
SELECT book FROM library WHERE author.name CONTAINS "Herbert"
```

Tento dotaz vypíše knihy z knihovny, kde první nalezený element `author` obsahující atribut `name` obsahuje ve jménu řetězec `Herbert`.

```
SELECT author FROM library WHERE (year > 1980 OR year = 1980) AND year < 1990
```

Poslední příklad vypíše autory z knihovny, kteří publikovali v letech 1980 až 1989 včetně. Na celočíselný literál nelze aplikovat relační operátor `CONTAINS`.

[2] Není třeba kopírovat komentáře.

[3] *case-sensitive* vyhledávání řetězců vyžaduje shodu i ve velikosti všech písmen.

2 Bonusová rozšíření

- **ORD** (1 bod): Jedním z rozšíření je podpora klauzule `ORDER BY`, která lexikograficky seřadí vracené elementy, přičemž ke každému elementu přidá atribut `order` s hodnotou odpovídající pozici v seřazeném seznamu. První vracený element má index 1.
- **LOG** (2 body): Dalším z možných rozšíření je podpora skládání podmínek pomocí závorek a klíčových slov `AND`, `OR` a `NOT`. Nejvyšší prioritu má operátor `NOT`, nižší má `AND` a nejnižší `OR`. Operátory `AND` a `OR` jsou levě asociativní.

3 Poznámky k hodnocení

Výsledný XML soubor bude porovnáván nástrojem pro porovnání XML souborů, který se umí správně vypořádat s rozdílným uspořádáním podelementů v rámcí stejného elementu, různým odsazením či s vypuštěním komentářů z původního vstupního XML souboru.