Dokumentace úlohy XQR: XML Query v Python 3 do IPP 2013/2014
Jméno a příjmení: Vojtěch Havlena
Login: xhavle03

1 Analýza řešení

Cílem zadané úlohy je implementace skriptu, který vyhodnocuje dotaz SELECT nad vstupem ve formátu XML. Výstup je možné mimo jiné ovlivňovat podmínkami v dotazu. Při řešení jsou využity znalosti formálních jazyků, a to především bezkontextových jazyků a syntaktické analýzy.

2 Implementace

Celý program je z logického hlediska rozdělen do několika spolupracujících celků. První z nich (`lex.py`) zodpovídá za lexikální analýzu dotazu. Druhý celek (`syntax.py`) provádí syntaktickou analýzu zhora dolů a zároveň vyhodnocuje dotaz. Modul `expr.py` provádí vyhodnocování podmínek dotazu za podpory funkcí z modulu `xmloperation.py`. Poslední celek (`xqr.py`) zodpovídá za zpacování vstupu a volání operací z jednotlivých modulů.

Prvním krokem při vykonávání skriptu je načítání a kontrola parametrů příkazové řádky pomocí funkce `getopt`. Taktéž je prováděna kontrola existence a oprávnění přístupu k zadaným souborům. Nemůže chybět inicializace proměnných zodpovídajících za správné vyhodnocení dotazu.

2.1 Lexikální analýza

Úvodní část zpracování dotazu je lexikální analýza, která je podstatná pro pozdější syntaktickou analýzu. Pro rozpoznávání lexémů byl implementován konečný automat. Mezi zpracovávané lexémy jsou zařazena jednotlivá klíčová slova, XML elementy, operátory a literály řetězce a celého čísla. Během lexikální analýzy probíhá také kontrola validity XML elementu, či správné formátování řetězce.

2.2 Syntaktická analýza a vyhodnocení dotazu

Pro kontrolu syntaktické správnosti dotazu byla zvolena analýza zhora dolů a pro kontrolu správnosti podmínek syntaktická analýza zdola nahoru. Během analýzy zhora dolů dochází k postupnému načítání tokenů a rekurzivním voláním funkcí reprezentující neterminály se kontroluje správnost zadání dotazu. Po načtení názvu zdrojového elementu dochází k jeho nalezení v XML stromové struktuře (použit byl algoritmus DFS). Pro práci s XML soubory byla použita knihovna `ElementTree` V dalším kroku se pomocí vestavěné funkce `iter` naleznou všechny cílové elementy a uloží se do seznamu. Pakliže byla zadána klauzule WHERE, dochází k vyhodnocení podmínky.

Kontrola syntaktické správnosti podmínek a jejich následné vyhodnocení je prováděna SA zdola nahoru pomocí precedenční tabulky (viz. Tabulka 1). V tomto tvaru je schopna vyhodnocovat i složitější podmínky, čímž bylo implementováno rozšíření LOG. Pro jednoduchost jako proměnnou i uvažujeme celou podmínku ve tvaru `<operand> <operátor> <operand>`. Kvůli umožnění vyhodnocování podmínek se do zásobníku (reprezentován seznamem) neukládá pouze typ tokenu, ale také seznam XML elementů, které vyhovují jednotlivým podmínkám. Díky tomu je možné k logickým spojkám přistupovat jako k množinovým operacím. Tedy např. při vyhodnocení operátoru AND se provede průnik seznamů, které byly vyjmuty ze zásobníku a výsledný seznam spolu s typem tokenu je uložen zpět na zásobník. Podobně pro operátor OR se jedná o sjednocení seznamů a pro operátor NOT o rozdíl všech cílových seznamů a daného seznamu. Během této analýzy dochází také ke kontrole typů operandů.

Co se týče vyhodnocování jednoduchých podmínek, postupně pro všechny dosud vyhovující elementy dochází nejprve k nalezení porovnávaného elementu/atributu a dle operátoru k příslušnému porovnání. Nevyhovující elementy jsou zahozeny. Pro správné porovnání musí také dojít k eventuálnímu přetypování hodnot elementů/atributů.

Po vyhodnocení všech podmínek je počet vyhodnocených elementů omezen parametrem klauzule LIMIT. Nakonec je výsledek dle vstupních parametrů uložen buď do zadaného souboru nebo vypsán na standardní

	AND	OR	NOT	(	)	i	$
AND	$>$	$>$	$<$	$<$	$>$	$<$	$>$
OR	$<$	$>$	$<$	$<$	$>$	$<$	$>$
NOT	$>$	$>$	$<$	$<$	$>$	$<$	$>$
(	$<$	$<$	$<$	$<$	=	$<$	
)	$>$	$>$			$>$		$>$
i	$>$	$>$			$>$		$>$
$	$<$	$<$	$<$	$<$		$<$	$

Tabulka 1: Precedenční tabulka pro vyhodnocování výrazů

výstup. Předtím je ovšem provedeno ještě eventuální obalení výsledku kořenovým elementem či přidání XML hlavičky.