Dokumentace úlohy XTD: XML2DDL v Python3 do IPP 2013/2014
Jméno a příjmení: David Kašpar
Login: xkaspa34

1 Úvod

V této dokumentaci je řešena implementace úlohy XTD v předmětu IPP[1], na *Fakultě informačních technologií Vysokého učení technického v Brně*, pro akademický rok 2013/2014. Cílem této úlohy bylo vytvořit skript ve skriptovacím jazyku Python[2] (verze 3.2.3), který převádí vstupní soubor v serializačním formátu XML[3] na sadu příkazů dotazovacího jazyka SQL[4].

1.1 Zdroje informací

K analýze a řešení projektu byly využity převážně informace nabité v předmětech Databázové systémy[5] a Informační systémy[6]. Seznámení se skriptovacím jazykem Python proběhlo v rámci Red Hat[7] *Python Labs*, následně pak pomocí neoficiálních[8][9] i oficiálních[10] tutoriálů, a oficiální dokumentace jazyka Python[11].

V rámci řešení bylo ještě potřeba najít vhodný algoritmus pro hledání transitivního uzávěru v orientovaném grafu[12]. Pro tyto účely byla použita anglická Wikipedie[13].

Fórum předmětu bylo využíváno pro řešení některých sporných bodů a nejasností zadání XTD.

1.2 Prostředí a testování

Jako referenční prostředí pro testování a následné hodnocení byl zadáním vybrán školní server Merlin[14]. Jedná se o linuxovou distribuci *CentOS*, čemuž byly dále přizpůsobeny další formální požadavky na výsledný skript, jako například vstupní i výstupní kódování v UTF-8, standardní Unixové konce řádků `'\n'`, apod.

Společně se zadáním byla zveřejněna základní sada testů, která představovala možnosti použití skriptu, použití jeho parametrů, jednotlivých vstupů a k nim odpovídajících výstupů.

2 Rozbor zadání

Zadání vyžadovalo, aby výsledný skript měl podobu tzv. textového filtru příkazové řádky. Jinými slovy, skript musel být schopen načítat vstupní data ze standardního vstupu a výsledek svého konání vypisovat na standardní výstup. To umožňuje přidávat skript do Unixových kolon (zřetězit ho) pomocí rour (angl. *pipelines*).

Skript musel být naprosto soběstačný s využitím pouze standardních a povolených knihoven. Bylo zakázáno spouštět jakékoliv další vlákna/procesy nebo příkazy operačního systému. Při jakékoliv chybě se všechny chybové hlášení musely vypisovat na standardní chybový výstup a skript musel skončit s patřičným chybovým kódem, který byl určen zadáním.

2.1 Převod formátu XML na formát SQL

Na tvar výstupního souboru byl kladen značný důraz. Zadání tedy obsahovalo i gramatiku pro způsob generování odpovídajících SQL příkazů, kterou lze považovat za modifikaci Backus-Naurovy formy[15].

Samotný proces převodu potom zahrnoval vytvoření příkazu pro generování nové SQL tabulky, obsahující všechny náležité sloupce této tabulky. Tyto sloupce byly generovány z atributů a obsahu jednotlivých elementů a lze je rozdělit do tří kategorií:

1. Sloupec pro textový obsah daného elementu, při zanedbání výskytů jakýchkoliv pod-elementů.
2. Sloupce generované z obsahu atributů daného elementu.
3. Sloupce pro pod-elementy nacházející se v daném elementu. Jednalo se o tzv. *cizí klíče* (angl. foreign keys), i když SQL příkaz, pro jejich označení, do výsledného souboru přidáván není.

Chování skriptu dále mělo být možno ovlivnit parametry, zadávanými při spuštění. Některé z těchto parametrů pak přímo ovlivňovaly základní způsob generování výstupu, umístění cizích klíčů, apod.

2.2 Převod formátu SQL na speciální XML reprezentaci

Zadání dále vyžadovalo změnu výstupního obsahu skriptu při použití přepínače `-g` na XML výstup, který bude reprezentovat relační vztahy mezi jednotlivými SQL tabulkami a kardinality těchto vztahů[16].

3 Návrh řešení

Z analýzy řešení vyplývaly dvě možnosti řešení pro generování SQL výstupu:

1. Vytvořit algoritmus pro postupné zanořování do hloubky v rámci zpracování XML, jelikož ho lze považovat za stromovou strukturu.
2. Využít vestavěné funkce jazyka Python v podobě iterátorů, procházet vstupní XML krok za krokem a vytvořit patřičné obalovací funkce pro zpracování jednotlivých XML elementů.

Jelikož je jazyk Python značně flexibilní, díky možnosti využití *objektově orientovaného paradigmatu*, bylo rozhodnuto použít řešení z bodu číslo 2.

Mimo standardního zapouzdření, dědičnosti, mnohotvárnosti a přetížení operátorů totiž jazyk Python nabízí právě ještě tzv. *Duck-typing*[17], snadné vytváření a použití výjimek a ve výsledku tak má méně strmou křivku učení v porovnání s jinými OO jazyky.

4 Způsob řešení

Při vytváření skriptu byl kladen důraz na co možná největší dodržování všech PEP[18] zásad týkajících se způsobu implementace a formálních náležitostí. Samotný obsah skriptu byl rozdělen na 10 modulů pro snadnější testování, kde každý modul zastává svojí prací určitou logickou část z celého algoritmu. Toto rozdělení taktéž umožnilo postupný inkrementální vývoj celého skriptu.

V rámci řešení části pro převod SQL na XML se nabízela možnost pouze transformovat vstupní XML soubor na odpovídající výstupní soubor. Avšak po patřičné úvaze byla tato možnost zamítnuta. Stále by totiž existovala potřeba provádět analýzu vstupního souboru a ve skriptu by se tak nacházely části, které by dělaly téměř to samé. Výsledné řešení tedy používá vygenerované SQL příkazy (s dodatečnými informacemi) pro vytvoření výstupu pro přepínač `-g`.

4.1 Vybrané části

Zde jsou uvedeny některé části výsledného skriptu s popisem jejich chování nebo zvláštností:

- všechny možné návratové kódy skriptu se nacházejí v souboru `errors.py`
- zpracování parametrů, jejich kontrolu a případný výpis nápovědy řeší modul `parameters.py` s využitím standardní knihovny `argparse`
- rozhraní pro vstup/výstup obstarává modul `input_output.py`, který rovněž zpracovává vstupní XML soubor/soubory pomocí standardní knihovny `xml.etree.ElementTree`
- zpracovaný vstupní XML soubor je následně analyzován v modulu `analyser.py`, který provádí dodatečné kontroly vstupního formátu, a zároveň při své analýze produkuje interní reprezentaci příkazů pro vytvoření patřičných SQL tabulek
- modul `analyser.py` taktéž provádí validaci obsahu souboru zadaného přepínačem `--isvalid`
- interní reprezentaci příkazů pro vytvoření SQL tabulek uchovává třída `TablesBuilder` z modulu `tables_builder.py`, která se chová jako kontejner a na požádání je schopna vrátit celý svůj obsah v řetězci, který lze použít pro výstup skriptu
- převod SQL příkazů pro vytvoření tabulek na odpovídající XML reprezentaci (přepínač `-g`) je obstaráváno v modulu `database_to_xml.py`

  (Pro tento převod je vždy potřeba nalézt a vytvořit reflexivní, symetrický a transitivní uzávěr v orientovaném grafu, který představuje relační vazby mezi výslednými tabulkami v SQL databázi. Nejsložitější v tomto ohledu je nalezení transitivního uzávěru v tomto grafu. Existuje několik algoritmů, které tento netriviální problém řeší. Ve výsledném skriptu byl pro svoji jednoduchost implementace použit *Floyd-Warshallův algoritmus*[19], který byl vymyšlen již v roce 1962. Jako příklad velmi efektivního algoritmu lze uvést *STACK_TC* [20] Fina Esko Nuutila, PhD.)

5 Rozšíření

V rámci projektu bylo naimplementováno rozšíření **VAL**, které umožňuje zkontrolovat, zdali obsah zadaného souboru, lze bezezbytku vložit do struktury tabulek generovaných pro normální vstupní soubor. Jméno souboru pro tento test je zadáváno pomocí přepínače `--isvalid=filename` a v případě neúspěšného testu skript končí s návratovým kódem `91`. Řešení je založeno na vytvoření druhé interní reprezentace příkazů pro tvorbu SQL tabulek a porovnání s již dříve vytvořenou vnitřní strukturou pro normální vstupní soubor. K tomuto porovnávání je využito přetížených operátorů porovnání nad objekty těchto struktur, což značně zjednodušuje samotný test.

Reference

[1] Principy programovacích jazyků a OOP, 2014. [Online]
https://www.fit.vutbr.cz/study/courses/index.php?id=9384

[2] Python (programming language) - Wikipedia, 2014. [Online]
http://en.wikipedia.org/wiki/Python_(programming_language)

[3] Extensible Markup Language (XML) 1.0 (Fifth Edition), 2014. [Online]
http://www.w3.org/TR/REC-xml/

[4] SQL - Wikipedia, 2014. [Online]
http://en.wikipedia.org/wiki/SQL

[5] Databázové systémy, 2013. [Online]
https://www.fit.vutbr.cz/study/courses/index.php?id=8648

[6] Informační systémy, 2013. [Online]
https://www.fit.vutbr.cz/study/courses/index.php?id=9360

[7] Red Hat Czech, 2012. [Online]
http://cz.redhat.com/

[8] Python for Newbies – Part1, 2014. [Online]
http://temporaryland.wordpress.com/2011/01/26/python-for-newbies/

[9] Python for Newbies – Part2, 2014. [Online]
http://temporaryland.wordpress.com/2011/01/27/python-for-newbies-part2/

[10] The Python Tutorial, 2014. [Online]
https://docs.python.org/release/3.2.3/tutorial/index.html

[11] Python v3.2.3 documentation, 2014. [Online]
https://docs.python.org/release/3.2.3/

[12] Transitive closure - Wikipedia, 2014. [Online]
http://en.wikipedia.org/wiki/Transitive_closure#Algorithms

[13] Wikipedia, the free encyclopedia, 2014. [Online]
http://en.wikipedia.org/wiki/Main_Page

[14] Server of Faculty of Information technology, Brno University of Technology, 2014. [Online]
http://merlin.fit.vutbr.cz/

[15] Backus–Naur Form - Wikipedia, 2014. [Online]
http://en.wikipedia.org/wiki/Backus–Naur_Form

[16] Cardinality (data modeling) - Wikipedia, 2014. [Online]
http://en.wikipedia.org/wiki/Cardinality_(data_modeling)

[17] Duck typing - Wikipedia, 2014. [Online]
http://en.wikipedia.org/wiki/Duck_typing

[18] PEP 0 - Index of Python Enhancement Proposals (PEPs), 2014. [Online]
http://legacy.python.org/dev/peps/

[19] Floyd-Warshall algorithm - Wikipedia, 2014. [Online]
http://en.wikipedia.org/wiki/Floyd_Warshall

[20] Efficient Transitive Closure Computation in Large Digraphs, 2014. [Online]
http://www.cs.hut.fi/~enu/thesis.html