Dokumentace úlohy XTD: XML2DDL v Python3.6 do IPP 2016/2017
Jméno a příjmení: Miroslav Karpíšek
Login: xkarpi05

Návrh:

Program je z větší části psán proceduálně, avšak jsou zde využívány třídy a jejich metody z importované knihovny `xml.etree.ElementTree` pro analýzu zadaného vstupního souboru. Stejně tak jednotlivá XML data jsou při převodu na jednotlivé tabulky reprezentovány třídou `Element`, jejíž instance jsou uloženy do slovníku `work_dict` (případně `val_dict` - při převodu ze zadaného souboru pro validaci), který po celý chod programu uchovává potřebné informace o datech.

Zpracování argumentů:

Nejprve se provede analýza argumentů programu za pomoci funkcí z importované standartní knihovny `argparse`. V průběhu zpracování se provádí kontrola, zda-li uživatel nezadal nepovolený počet, či nepovolenou kombinaci argumentů. V případě, že ano, je program ukončen s odpovídající návratovou hodnotou. Následně se předají informace získané analýzou jednotlivým přepínačům programu.

Zpracování vstupu:

Po úspěšném otevření zvoleného vstupního souboru je za pomoci funkce `parse` z výše zmíněné importované knihovny provedena pokus o převedení zadané XML-struktury na strukturu stromu s listy. Ten je implementován pomocí třídy `ElementTree` a jeho jednotlivé listy, pomocí třídy `Element`, ty představují elementy zadané ve vstupním XML souboru. V případě, že se nejedná o validní XML soubor je zde program ukončen s chybou 4. Následně je předán ukazatel na instanci třídy `ElementTree` funkci `doXml` která rekurzivně prochazí celý strom po listech a ukládá informace o existenci tabulek, jejich atributů a odkazů do jiných tabulek (cizých klíčů) do instance třídy `TableElement` jejíž odkaz je následně vložen již zmiňovaného slovníku `work_dict`. Stejně tak ukládá do výše zmíněné instance třídy informaci o typu daného atributu, jehož určení zajištuje funkce `getType` která přijímá odkaz na vyšetřovaný objekt a vrací jeho typ specifikovaný v zadání projektu.

Vyhodnocení přepínačů:

Následuje část programu kde se prochází hierarchie tabulek a provádí se jejich úprava na základě zadaných přepínačů. V této sekci popíšu, jak probíhá vyhodnocení jednotlivých argumentů. Pouze argument "-a" je vyhodnocen již během průchodu stromem ve funkci `doXml` kdy se při jeho zadání nezaznamenávají informace o existenci atributů.

Argument --etc=n:

Pokud je tento argument zadán, je prováděna kontrola počtu výskytů jednotlivých cizích klíčů v tabulkách uložených ve `work_dict`. Pokud je počet výskytu daného cizího klíče v tabulce vyšší než-li je zadané ***n***, je informace o vazbě mezi tabulkami uložena do odkazované tabulky, před samotným uložením je provedena kontrola, zda-li by nedošlo k přepsání již existujícího záznamu - v tom případě program končí s chybovým hlášením a návratovou hodnotou 90.

Argument -b:

Pokud je tento parametr zadán, jsou stejnojmenné elementy považovány za jediný téhož jména. V případě, že zadán není, je v této části volána funkce `nameCheck`, která příjimá jako parametr ukazatel na slovník, jehož obsah je předmětem kontroly, při níž se stejnojmené elementy odliší pomocí přípony názvu. Specifický postup řešení: v případě kolize názvů, (z důvodů cyklického přiřazování přípony ***n***) zde program není ukončen s chybou kolize

názvů, protože možnost kolize odhalí a automaticky přiřadí místo ***n, n + x***, kde **x** je první možná hodnota která není v kolizi

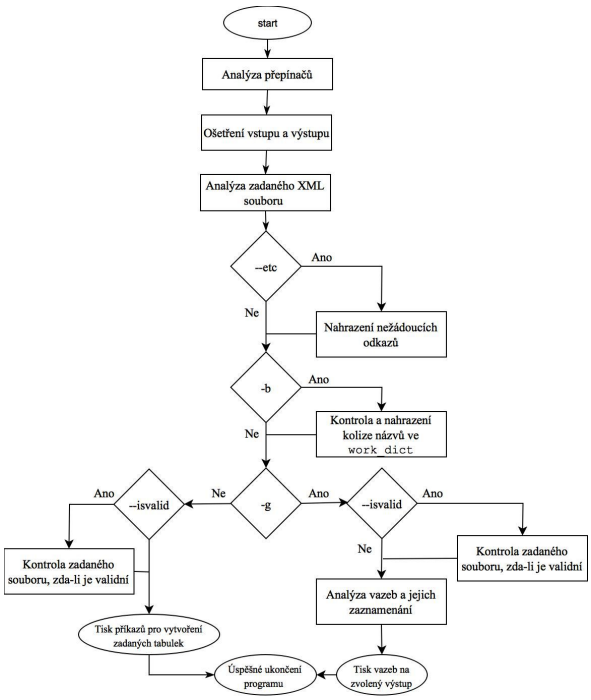
Argument -g:

Při zadání tohoto přepínače jsou na místo SQL - příkazů do výstupního souboru vypsány názvy jednotlivých tabulek a jejich vzájemné vazby včetně těch, vzniklých aplikováním tranzitivity. V případě, že je zadán soubor pro validaci je prvně provedena kontrola možnosti bezezbytku vložit data z validovaného souboru pomocí funkce `validCheck`. Ta v případě, že data uložená v souboru, který byl určený pro kontrolu, by nebylo možné bezezbytku vložit do výsledné tabulky, ukončí program s návratovou hodnotou 91. Během této funkce se také volá funkce pro kontorlu kolize názvů `nameCheck`, jejíž funkce byla již popsaná výše, avšak tentokrát je kontrolovaným slovníkem slovník s uloženými daty pro validaci.

Poté je volána funkce `printG`, která se krom výpisů jednotlivých vazeb na výstup stará i o určení kardinalit u jednotlivých vazeb mezi tabulkami a vícenásobné kontroly tranzitivity relací. A to za pomoci volání funkce `transit`, která u již existujících vazeb hledá ty, kde by se dalo uplatnit pravidlo tranzitivity a takové vazby následně zaznamenat. Pokud je zaznamenána alespoň jedna nová vazba, celý proces hledání nových vazeb se opakuje. Nakonec jsou získané informace tisknuty na zvolený výstup dle formátu popsaného v zadání. Program je poté v pořádku ukončen.

Rozšíření:

V programu je implementováno rozšíření **VAL** které krom standartních úkonů provede kontrolu, jestli datová struktura udaná vloženým souborem pro kontrolu je plně vložitelná do tabulek definovaných dříve. Funkčnost tohoto rozšíření obstarává funkce `validCheck`, která porovnává datovou strukturu uloženou ve slovníku `val_dict` s reprezentací tabulek ve `work_dict`, v případě, že data pro validaci by nešlo bezezbytku vložit do tabulky, je program ukončen s návratovou hodnotou 91 a chybovým hlášením.



obr. 1: Zjednodušený diagram popisující průchod programem na základě zadaných přepínačů