Dokumentace úlohy XTD:
XML2DDL v PHP5 do IPP 2016/2017
Jméno a příjmení: Michael Halinár Login: xhalin01

Úvod
Mojí úlohou byla implementace aplikace která bude zpracovávat vstupný soubor v tvare .xml na výsledné SQL příkazy – konkrétně vytváření tabulek v databázi.

Návrh
Nejprve je nutné načíst vstupní soubor do objektu, na tento účel dobře poslouží funkce simplexml_load_string(). Vzhledem k tomu že php verze 5.6 nepodporuje struktury, pro lepší zacházení s daty bude dobré když využijeme možnost objektového programováni. Tímto si zabezpečíme neomezený přístup k atributům třídy, jelikož skoro celý program bude napsán v jeho metodách. Dále bude nutné oddělit atributy, koncové hodnoty a cizí klíče do ostatních tabulek. Když toho dosáhneme bude možné vytvářet sql příkazy.

Implementace
Parsováni argumentů
Uživatel má možnost zadání několika vstupných argumentů. Tyto argumenty jsou zpracovávány funkcí *parseArgs() ,* která kontroluje zadané argumenty a vrací je v asociativním poli. Tato funkce zároveň kontroluje i validitu argumentů a chybné kombinace.

Získáni vstupu:
Podle toho jaké zadal uživatel argumenty, aplikace rozhodne zda se má číst vstup z terminálu nebo ze souboru.

Třída sqlTables
Obsahuje atributy a metody které budou zpracovávat xml. Mezi atributy patří 5 polí a to *nodes, nodeCount, simpleTables, tables, attributes.* Pole *nodes* bude uchovávat jednotlivé tabulky, přičemž každá bude obsahovat dále cizí klíče na odkaz do dalších tabulek. *NodeCount* pomocná proměnná pro počítaní počtu pod-elementů pro každou tabulku. *SimpleTables* obsahuje pole tabulek vzniklých z koncových hodnot ve vstupním xml. *Tables* bude obsahovat všechny tabulky které se budou vytvářet. *Attributes* obsahuje atributy příslušných tabulek.

Hledání atributů a koncových hodnot
Po převodu vstupního souboru na objekt funkcí *simplexml_load_string()*, si převedeme tento objekt na pole pomocí sekvence příkazy *json_encode()* a následně *json_decode()*. Tímto získáme pole z které budeme dále parsovat. Funkcí *findAttributes* najdeme koncové elementy a atributy. Tato funkce prochází celé a obsahuje 3 základní větve, pokud se právě procházené pole jmenuje *@attributes* znamená to že se nacházíme v atributech elementu a uložíme si jej do pole *sqlTables->attributes.* Pokud je obsah právě procházeného prvku pole značí to že se tady nachází pod-element a zavoláme na tento pod-element tuto stejnou funkci rekurzivně. Jinak se nacházíme v koncovém elementu xml a poznačíme si to taktéž do atributů ale tento krát se značkou #TAB# která nám označí že z toho elementu bude generována konečná tabulka. Musíme jí ještě přiřadit datový typ. Pro tyto účely poslouží funkce *getOldType()* pokud už hodnotě někdy byl nastavený datový typ, nebo *getNewType()* pokud ne. *SetType()* pak nastaví výslednou hodnotu typu.

Hledání uzlů a cizích klíčů
Pro tento účel nám poslouží funkce *getNodes()* jejím vstupem je objekt obsahující xml vstup a název rodičovského uzlu. Tato funkce pak prohledává všechny elementy a pod-elementy. Jestliže právě procházený prvek není uložen v poli *sqlTables->nodes* pak se tam vloží. Pokud právě procházený element má nějakého rodiče, pomocné pole uchová případný počet všech těchto elementů. Stejná funkce se rekurzivně zavolá na všechny pod-elementy. Získáme tak pole elementů a jejich pod-elementů – odkazů do dalších tabulek.

Etc correction
Uživatel má možnost upravovat počty sloupců které se budou vytvářet ze stejnojmenných elementů. Pokud počet stejnojmenných elementů je menší než zadal uživatel, tak se vytvoří cizí klíč v tabulce na kterou elementy ukazovali, a původní elementy se vymažou.

Vytvoření koncových tabulek
Elementy které jsme si označili v značkou *#TAB#* v poli *sqlTables->attributes,* budou tvořit jednoduché tabulky se sloupcem pro primární klíč a sloupcem pro koncovou hodnotu s příslušným datovým typem.

Přidání atributů a spojení tabulek

Pokud si uživatel zvolil přepínač *-a* do výsledných tabulek se nebudou přidávat sloupce z atributů, funkce *addAttributes()* se v tomto případe nezavolá. Následně se přidají všem cizím klíčům koncovky "*_id*". Dále se spojí pole složených a jednoduchých tabulek. Výsledné pole *sqlTables->tables* tak obsahuje všechny potřebné údaje.

Výpis

Pokud uživatel nezadal přepínač *-g* program přejde k výpisu do požadovaného formátu, jinak se v poslední fázi vypočítají vztahy tabulek. A to tak že se porovnává každá tabulka s každou. Jestliže se názvy rovnají tento vztah bude označen jako 1:1, jestliže se první tabulka nachází nachází v druhé, nebo v jejich potomcích je tento vztah označen jako 1:N, jestli toto platí naopak tento vztah je N:1. Jinak je vztah označen jako N:M.