

# Zadání úlohy do projektu z předmětu IPP 2013/2014

Zbyněk Křivka a Dušan Kolář

E-mail: {krivka, kolar}@fit.vutbr.cz, {54 114 1313, 54 114 1238}

## XTD: XML2DDL

**Zodpovědný cvičící:** Ondřej Navrátil (inavra@fit.vutbr.cz)

### 1 Detailní zadání úlohy

Skript na základě XML souboru s daty vytvoří sadu SQL příkazů generujících příslušnou strukturu tabulek v SQL databázi, ve které by se data obsažená ve vstupním souboru mohla nacházet.

Vstupním souborem může být libovolný XML soubor bez speciálních znaků<sup>1</sup> v názvech elementů a atributů. Skript provede analýzu vstupního souboru a následně pro každý netextový element (mimo kořenového) vytvoří SQL příkaz generující tabulku pojmenovanou právě podle elementu. Výsledná tabulka bude obsahovat sloupce reprezentující jednotlivé atributy, textové elementy a odkazy na tabulky (implementovaných formou cizích klíčů) reprezentované dalšími podelementy. V případě shody jmen elementů bude vytvářena jediná tabulka obsahující sloupce pro sjednocení atributů a podelementů všech těchto elementů. Například pro elementy `<book title="Automata and Languages"/>` a `<book author="Karel Čapek" title="Robot"/>` bude výsledkem jedna tabulka `book` obsahující sloupce `author` a `title`. Mimo sloupce, vzniklé z atributů, netextových i textových podelementů (název sloupců vzniklých z textových elementů bude `value`), bude každá tabulka obsahovat sloupec daný jménem tabulky s předponou „PRK\_“ a příponou „\_ID“ reprezentující tzv. primární klíč.

**Tvar výstupního souboru** bude case-insensitive<sup>2</sup> a je dán následujícím předpisem (neterminály jsou v úhlových závorkách, `<OFILE>` je startující neterminál, tokeny jsou odděleny bílým znakem, zbývající řetězce jsou terminální s tím, že terminály psané velkými písmeny reprezentují klíčová slova, kdežto terminály psané malými písmeny jsou literály definované níže):

```
<OFILE> --> <HEADER> <DDL>
<HEADER> --> empty
<HEADER> --> header
<DDL> --> <CREATE-TABLE> <DDL>
<DDL> --> empty
<CREATE-TABLE> --> CREATE TABLE table_name ( <ID> , <DECLARATIONS> ) ;
<ID> --> PRK_table_name_ID INT PRIMARY KEY
<DECLARATIONS> --> <DECLARATION> , <DECLARATIONS>
<DECLARATIONS> --> <DECLARATION>
<DECLARATION> --> column_name data_type
```

**Datové typy** (`data_type`) vyskytující se ve výsledném souboru budou `BIT`, `INT`, `FLOAT`, `NVARCHAR` a `NTEXT`. Datové typy budou přiřazovány sloupcům tak, že:

- pokud daný atribut, nebo textový podelement bude obsahovat ve všech jeho výskytech u stejnojmenných elementů hodnotu 0, 1, *True*, *False*, nebo hodnota nebude zadána vůbec, použije se datový typ `BIT`,

<sup>1</sup>Speciálním znakem je myšlen libovolný znak, který by porušoval strukturu výstupního souboru definovanou níže.

<sup>2</sup>Nebudou se rozlišovat malá a velká písmena.

- pokud daný atribut, nebo textový podelement bude obsahovat celé číslo, pak datový typ sloupce bude **INT**,
- pokud daný atribut, nebo textový podelement bude obsahovat reálné číslo (dle standardu C99), pak datový typ sloupce bude **FLOAT**,
- pokud daný atribut bude obsahovat textový řetězec, pak datový typ sloupce bude **NVARCHAR**,
- pokud daný textový podelement neodpovídá výše určeným podmínkám pro datové typy **BIT**, **INT**, **FLOAT**, pak bude výsledný sloupec datového typu **NTEXT**.

Navíc, pokud podelementem je další netextový podelement a ten se objeví ve všech elementech stejného jména maximálně  $n$ -krát ( $n$  je dáno parametrem **etc=n** viz definice parametrů), vytvoří se  $m$  sloupců typu **INT**, přičemž název bude dán konkatencí názvu podelementu s *num* a řetězcem „\_ID“, kde  $num \in \{\text{empty}, 1, 2, \dots, m\}$  a  $m$  je maximální počet stejnojmenných podelementů daného elementu. Za předpokladu, že  $m = 1$ , není potřeba sloupec indexovat a *num* pak považujte za prázdný řetězec. Pokud se v elementu objeví podelement stejného názvu vícekrát jak  $n$ , tak se naopak vytvoří jeden nový sloupec typu **INT** v tabulce určené názvem podelementu. Název sloupce bude určen konkatencí názvu elementu a řetězce „\_ID“. Tedy, pro `<book><author/><author/></book>` a **--etc=2** budou vytvořeny sloupce **author1\_ID** a **author2\_ID** v tabulce **book**. Naopak pro `<book><author/><author/></book>` a **--etc=1** bude vytvářen sloupec s názvem **book\_ID** v tabulce **author**. Odkazy<sup>3</sup>, které by vedly na tabulku vzniklou z kořenového elementu se, přirozeně, negenerují.

**Ostatní lexémy** jsou definovány následovně: **table\_name**, resp. **column\_name** jsou identifikátory tabulek, resp. jejich sloupců, **empty** reprezentuje prázdný řetězec, **num** je libovolné dekadické, nezáporné, celé číslo a **header** je hlavička výstupního souboru, kde se bude nacházet na prvním řádku uvozena dvojznakem „--“ a zakončena dvěma konci řádku. Zbývající obsah hlavičky bude určen volitelným parametrem **--header='hlavička'** tak, že **hlavička** (bez ohraničujících apostrofů, či uvozovek) bude vložena bezprostředně za „--“. V případě, že nebude hlavička parametrem skriptu specifikována, do výstupního souboru se nevkládá ani dvojznak „--“, ani dvojí konec řádku.

### Definice parametrů skriptu:

- **--help** viz společné zadání všech úloh
- **--input=filename** zadaný vstupní soubor ve formátu XML
- **--output=filename** zadaný výstupní soubor ve formátu definovaném výše
- **--header='hlavička'** na začátek výstupního souboru se vloží zakomentovaná hlavička
- **--etc=n** pro  $n \geq 0$  určuje maximální počet sloupců vzniklých ze stejnojmenných podelementů
- **-a** nebudou se generovat sloupce z atributů ve vstupním XML souboru
- **-b** pokud element bude obsahovat více podelementů stejného názvu, bude se uvažovat, jako by zde byl pouze jediný takový, ale s tím že bude uvažován ten s nejvyšším datovým typem dle odstavce Řešení konfliktů (tento parametr nesmí být kombinován s parametrem **--etc=n**)

---

<sup>3</sup>Sloupce vzniklé konkatencí názvu tabulky a \_ID

- `-g` lze jej uplatnit v kombinaci s jakýmikoliv jinými přepínači vyjma `--help`. Při jeho aktivaci bude výstupním souborem pouze XML tvaru

```
<tables>
  <table name="nazev_tabulky">
    <relation to="nazev_cizi_tabulky" relation_type="vztah" />
    ...
  </table>
  ...
</tables>
```

kde mezi `<tables>` a `</tables>` budou zahrnuty všechny, za normálních okolností, generované tabulky a mezi tagy `<table>` a `</table>` všechny relace (včetně cyklu). Vztah bude nabývat pouze jednu z hodnot 1:1, 1:N, N:1 nebo N:M, v závislosti na typu relace (viz obrázek 9 ve 4. snímcích k přednáškám IDS). Relace jsou tranzitivní a symetrické - tj. pokud tabulka *A* je v relaci s *B* a *B* je v relaci s *C*, pak bude do výstupu zahrnuta i relace z *A* do *C*, z *B* do *A* a z *C* do *A*. Určení kardinality je detailněji popsáno v sekci *Určení kardinality relací*.

Při nezadání parametru `--input=filename`, resp. `--output=filename`, uvažujte standardní vstup resp. výstup a při nezadání `--etc=n` považujte *n* neomezené.

### Příklad:

```
<Earth>
  <country name="Slovenská republika">
    <capital>Bratislava</capital>
  </country>
  <country name="Česká republika" population="8900000">
    <capital>Praha</capital>
    <city>Brno</city>
    <city>Olomouc</city>
  </country>
</Earth>
```

Je-li skript spuštěn s parametrem `--etc=1`, generují se tabulky `country`, `city` a `capital`. V tabulce `country` budou sloupce `PRK_country_ID`, `name`, `capital_ID` a `population`. Tabulka `city` bude obsahovat sloupce `PRK_city_ID`, `country_ID` a `value`, a konečně tabulka `capital` bude složena ze sloupců `PRK_capital_ID` a `value`.

**Řešení konfliktů:** Vstupní XML soubor bude vždy validní. Na druhou stranu, během zpracování můžete narazit na atributy stejného názvu ale jiných datových typů. V tomto případě výsledný datový typ bude roven nejvyššímu zúčastněnému datovému typu podle uspořádání `BIT < INT < FLOAT < NVARCHAR < NTEXT`. Při konfliktu názvu sloupců vznikajících z atributů nebo textových elementů a sloupců reprezentující odkaz do tabulky negenerujete žádný výstupní soubor, ale ukončete skript s chybovým hlášením vypsáním na standardní chybový výstup a vraťte návratový kód 90. Pokud nějaký element obsahuje atribut `value` a současně textový podelement, pak se generuje pouze jeden sloupec `value` s vyšším datovým typem dle uvedeného uspořádání.

**Určení kardinality relací:** Následující algoritmus definuje postup pro korektní určení kardinality vztahů mezi tabulkami. Jednotlivé kroky algoritmu jsou záměrně popsány s použitím matematických formalismů a záleží na studentovi, nakolik je schopen tyto metody aplikovat a reflektovat ve své implementaci.

Předpokládáme, že již bylo pro daný vstup vytvořeno korektní databázové schéma. Toto schéma budeme reprezentovat množinou  $T$  všech tabulek a dále relací vzájemného odkazování  $F \subseteq T \times T$ . Pokud  $(a, b) \in F$ , tedy  $aFb$ , potom tabulka  $a$  obsahuje (alespoň jeden) sloupec odkazující se do tabulky  $b$  (v klasickém případě nazvaný `b_id`). V dalším budeme používat syntaxi  $a \rightarrow b$ .

Výstupem algoritmu bude zobrazení  $R : T \times T \rightarrow \{1:N, 1:1, N:1, N:M, \varepsilon\}$  reprezentující požadované kardinality (hodnota  $\varepsilon$  vyjadřuje, že mezi tabulkami není žádný vztah). Zobrazení získáme následujícím postupem:

1. (výchozí bod)  $R = \emptyset$ .
2. (inicializace)  $\forall a, b \in T$ :
  - (a) Pokud  $a = b$ , pak  $R(a, b) = 1:1$ .
  - (b) Pokud  $a \neq b, a \rightarrow b$  a  $b \rightarrow a$ , pak  $R(a, b) = N:M$ .
  - (c) Pokud  $a \neq b, a \rightarrow b$  a neplatí  $b \rightarrow a$ , pak  $R(a, b) = N:1$ .
  - (d) Pokud  $a \neq b, b \rightarrow a$  a neplatí  $a \rightarrow b$ , pak  $R(a, b) = 1:N$ .
  - (e) Jinak  $R(a, b) = \varepsilon$ .
3. (tranzitivita vztahů  $1:N$ ) Dokud je možné provést nějakou změnu, proved'  $\forall a, b \in T, R(a, b) = \varepsilon$ : Pokud  $\exists c \in T : R(a, c) = 1:N, R(c, b) = 1:N$ , pak  $R(a, b) = 1:N$ .
4. (tranzitivita vztahů  $N:1$ ) Dokud je možné provést nějakou změnu, proved'  $\forall a, b \in T, R(a, b) = \varepsilon$ : Pokud  $\exists c \in T : R(a, c) = N:1, R(c, b) = N:1$ , pak  $R(a, b) = N:1$ .
5. (vznik a tranzitivita vztahů  $N:M$ ) Dokud je možné provést nějakou změnu, proved'  $\forall a, b \in T, R(a, b) = \varepsilon$ : Pokud  $\exists c \in T : R(a, c) \neq \varepsilon, R(c, b) \neq \varepsilon$ , pak  $R(a, b) = R(b, a) = N:M$ .

Výsledné zobrazení  $R$  popisuje všechny relace v tomto schématu.

## 2 Bonusová rozšíření

**VAL** (až 2 body): Jako bonusové rozšíření lze implementovat ověření, zda XML soubor určený parametrem `--isvalid=filename` obsahuje data, která lze beze zbytku vložit do databázové struktury tabulek vzniklé pro soubor daný parametrem `--input=filename`. V případě že ano, bude se generovat výstup standardním způsobem. Pokud ne, skript se ukončí chybou s návratovou hodnotou 91. Volitelné parametry `-a` a `-b` aplikujte i na testovaný XML soubor. Tedy, pokud XML soubor sloužící jako vstup i testovaný soubor současně má nějaké atributy a skript je spouštěn s parametrem `-a`, pak bude test v pořádku a výstup se bude generovat standardním způsobem.

## 3 Poznámky k hodnocení

Výstup bude automaticky analyzován (je nezbytné, aby byl syntakticky správně dle zadané gramatiky) a bude z něj následně vytvářen XML soubor. Ten pak bude porovnáván nástrojem JEx-XML pro porovnání XML souborů. Při generování SQL příkazů není tedy třeba brát zřetel na uspořádání generovaných deklarací tabulek, ani na pořadí jednotlivých sloupců v rámci jedné tabulky.

**Revize zadání:**

2014-03-05: Doplněn popis parametru -g.