Task documentation XTD: XML2DDL in PHP for IPP 2013/2014
Name and surname: Ján Spišiak
Login: xspisi03

1 Task Description

Write a script in PHP, which given a valid XML document on the input, outputs a DDL document describing SQL structure needed to accommodate input data. The program should also support option to output XML document describing relations between tables, and few other options affecting its behavior.

2 Implementation

2.1 Parsing arguments

After considering using of PHPs' native `getopt()`, I decided to instead implement my own method `parseArguments()`, mainly due to lack of `getopt()`s extensibility and customization. My own method simply validates and extracts values from optional arguments, and if necessary throws exception describing error and it's code (in this case 1).

2.2 Parsing XML

Input is read into memory using `file_get_contents()` and then passed to `SimpleXMLIterator`. We can use `RecursiveIteratorIterator` to iterate over all element nodes of the XML document. While iterating we do two things:

1. Merge the new value types from attributes and text sub-element of current element into old value types of elements before in array `tables` (unless option `-a` is set, then we simply ignore attributes)

2. Check the current count of elements of same name in current parent, and remember the maximum in array `counts` (unless option `-b` is set, then max is always one)

2.3 Generating Structure

After having parsed XML into two arrays describing columns and counts, we need to use this information to form array of `relations` between tables and if necessary generate additional columns (while checking if there isn't conflict in column names). The algorithm is quite simple. As we iterate over tables we:

1. Construct reflexive closure and save it to `relations` array (though original task description provides no information as to why this is needed)

2. Iterate over `counts[$tableName]` and if count is less than or equal the value specified by parameter `--etc` we generate sufficient number of columns for foreign keys to child table and add relation *parent* $\rightarrow$ *child*. If the count is more than specified we instead generate column in child table and also add relation *child* $\rightarrow$ *parent* to `relations` array.

2.4 Transitive Closure

If option `-g` is specified, we should output XML describing relations between tables. But before that we need to do the "transitive" closure. Why? I don't know, it was in the task description. Why "transitive"? Well that's also good question. The algorithm is strangely modified version of transitive closure, because we not only care if there is a relation, but there could be different relation types between tables.

I've implemented it in one loop, which checks if there weren't any changes done, and inside normal transitive `for` loops iterating start, middle and end nodes. However instead of simply checking for existence $start \to middle \to end$ we also have to check $middle \to start$ and $end \to middle$, and then decide what kind of relation to assign to $start \to end$ and $end \to start$.

3 Conclusion

The program works as intended, given XML document it can generate DDL output, containing SQL table structure for storing this document. It's also able to generate XML document describing transitive closure on relations between these tables, although in quadratic time complexity.