Dokumentace úlohy XTD: XML2DDL v PHP 5 do IPP 2014/2015
Jméno a příjmení: Daniel Klimaj
Login: xklima22

Documentation for IPP 2014/2015, Project 1 - XTD: XML2DDL

1 Assignment

The purpose of this project was to create script in PHP 5, which converts data in XML file to Data Definition Language (DDL). The output data may be affected by several parameters and may be end up in defined file or on standard output (stdout).

2 Solution

2.1 Structure

Script is divided in 4 files. Each of files provides specific functions or classes:

- xtd.php is main file in which input XML file (if defined) is recursively processed and values are stored in classes provided by ddl.php.
- ddl.php provides classes which represents abstractions of entities used in DDL. These classes are DataType (subset of SQL data types – Bit, Int, Float, Nvarchar, Ntext), RelType (type of relation – Epsilon, 1:1, 1:N, N:1, N:M), Table, Column, Subelement (reference) and Relation.
- params.php – provides Params class with static methods and static variables. Static variables are set according to script parameters and are accessed via getters. Also deals with forbidden combination of parameters.
- ucs.php – provides Uniform Cost Search (UCS) algorithm which is used to find transitive relations between tables.

2.2 Data processing

The input data may be in the form of file or standard input (stdin) if --input switch is not used. Script checks if input is not empty string and on success, it creates SimpleXMLIterator and calls loop_xml function which recursively loops through each XML element.

For each element loop_xml creates Table instance and adds this instance into tables array. Table instance is created only once for elements with same name. Further loop_xml checks element's attributes (attributes are skipped if -a parameter is used), text element and child elements (subelement). When any attribute, text element or subelement is found then table instance is requested from tables array and check if table already has column (for attribute and text element) or subelement (for child elements) with given name is performed. If column or subelement does not exist then new instance of Column class or Subelement class is created and is added to table. If column already exists then it's data type may be updated if most recent occurrence's data type has higher priority. Data type determination is performed in bot cases. If subelement already exists, function counts number of it's occurences in element and updates number of maximum occurences in it's instance if most recent count of occurences is higher then previous.

When all elements are processed then check if subelement names and column names are not colliding is performed. If parameter etc=n is used then further data processing is performed on Table instances. This processing consist of removing subelements with higher number of occurences than n and creating subelement with table's namein referenced table. This is followed by another check for collisions.

2.3 Determining relations

Direct relations are found by looping through array of Table instances and checking if table has some subelements. If subelement is found then new instance of Relation is added to table's relations array and then symmetric relation if determined:

- Table A has subelement with name of Table B and Table B has subelement with name of Table A => N:M
- Table A has subelement with name of Table B and Table B hasn't subelement with name of Table A => N:1
- Table B has subelement with name of Table A and Table A hasn't subelement with name of Table B => 1:N
- 1:1 relations are automatically defined

Indirect (transitive) relations are found using UCS algorithm which finds the shortest path from starting element to end element. Search function returns path as array, which elements represents order of visited tables. If array is empty then path does not exist or one or both tables are invalid. Relation type is determined by looping path array and

checking for relation types to next table in array. Relation of first and second table in sequence is stored and if it does not change in the loop then this relation type is relation type of start and end table, if it changes during the loop then relation type is N:M.

3 Conclusion

Testing of script functionality was performed at first locally on my Linux Mint system and then on school server Merlin. I have found this project a little bit time consuming rather than difficult.