IN4331 Web Data Management - Lab assignment 1

Team:  
David Hoepelman  
1521969  
[david.hoepelman@gmail.com](mailto:david.hoepelman@gmail.com)

Gijs Kuijer  
4094107  
[gijs.kuijer@gmail.com](mailto:gijs.kuijer@gmail.com)

# Introduction

For this assignment the group had to choose chapter 5 or 6 from the Web Data Management book and complete all exercises and assignments in this chapter. In this case the group choose chapter 6. There was only one big assignment in this chapter mainly consisting of parsing and evaluating C-TP trees. A brief discussion will follow.

# Assignment discussion

The assignment in chapter 6 started by creating an algorithm to evaluate C-TP tree patterns (instructions 1). The algorithm was already largely given in the chapter but had to be implemented. During implementation we encountered that the algorithm was not very complete and that we missed implementation of some classes, such as the PatternNode. Next to completing the missing pieces we also changed the algorithm slightly in how it worked as we thought it was clearer.

The original algorithm consists of PatternNode which have stacks on them with matches. In the algorithm the stacks of all the PatternNodes are iterated to see if they match the current opening element. In our algorithm we iterate the children of the current PatterNode to see if something matches. We think this will result in a more efficient algorithm as only the PatternNode’s are iterated that actually matter. The PatternNodes that are child of the PatternNode that is last matched and is still open are relevant as in XML it is also true that when in an element only the elements inside it are relevant.

To implement instructions 2, an algorithm to compute result tuples we implemented a Boolean in the PatternNode if we wanted to have the result. If this Boolean was set to true the algorithm, which is launched after evaluating the C-TP tree patterns, will take the PatternNode into account and will output it’s preNumber into the tuple table.

Extension 1 is implemented by adding an extra criteria to the name comparison of the started element and the PatternNode’s. If the PatternNode name was a ‘\*’ a match is added for the current element. We also changed here that the name of the element is saved on the Match instead of the name of the PatternNode. By doing this we could output the actual name of the element that was matched on the end and did also not have to change the endElement method.

Extension 2 had to make optional nodes available. We implemented this feature by adjusting the PatternNode class and including an optional Boolean. We also altered the endElement method to check children on the optional feature. If this feature was set the endElement method would not require it to be in the element.

For extension 3 optional nodes had to be made available for creating the result tuples. This was simply fixed by walking through the Matches and if a Match does not have a value ‘null’ is inserted in the tuple table on the spot where the value should have been. We did not have to check if the PatternNode is optional as the evaluation algorithm takes care of this.

The values of elements and attributes are taken care of by extension 4. We extended the PatterNode and Match with a valuePredicate. In the characters method of StackEval we saved the characters to the Match which was last opened if the preNumber corresponded. If the preNumber did not correspond it may not be filled with the value as the value might then be corresponding to an element inside the last matched element. It could also be that the element inside the last matched element is already closed. This would be valid XML but for XML databases this would not be valid. Therefore we do also not consider this case. The extension is further implemented by checking if the PatternNode valuePredicate and the Match valuePredicate are equal, and if not remove the match.

Extension 5 was implemented by letting algorithm 2 get the names of the matches, then after this iterate over the attributes and get their names and values and after this end an element tag, list the children or contents (such as text values) and place an end element tag.