Task documentation XQR: XML query in Python 3 for IPP 2016/2017
Name and surname: Rastislav Kanda
Login: xkanda00

1 Assignment

The assignment was to create Python a script which provides an evaluation of a query, which is similar to command *SELECT* in *SQL* language, over a XML file. Output of the script is a *filtered* XML file which contains elements that meets the requirements of the query.

2 Workflow

Script flow is designed as: Parse arguments → Load files and parse query → Select elements from *FROM* part → Select elements from *SELECT* part → Select elements from *WHERE* part → Order elements if needed → Limit number of elements if needed → Print the resulting structure

3 Implementation

3.1 Parse arguments

For parsing the arguments was following command *ArgumentParser* was chosen from the *argparse* library. Every argument is also manually checked for error occurrence.

3.2 Load files and parse query

Loading content from files is made by a standard open and read Python functions. An input file is loaded into object and then parsed with the parser from *xml.dom.minidom* for an ease of manipulation with structure.
For parsing query either from the file or from the entered argument a regular expressions to check if given query corresponds to the form of expected expression was used.
If any part of query is missing an exception is made, an error will be called and program will terminate with a return code of 80.
Every part of query is loaded into a dictionary and then processed one by one. The hardest part to process is the *WHERE* part of query, which could contains uncertain numbers and patterns of words, operators on which further processing is dependant.

3.3 Filtering XML input file in right order

The input file has to be filtered in the correct order of operations. First, the *FROM* part of query has to be found, then the *SELECT* part followed by *WHERE* can be examined.
For finding the wanted elements the minidom function is used *getElementsByTagName()* in combination with the *if* statements and *for* loops. A successful find depends on, whether if only name of the element is given or if only the name of attribute is given.
If only the name of the element is entered, then every element in the structure is examined and the first occurrence is selected.
If only the name of attribute is given, every element in structure is examined and if the attribute is found, then a element with the given attribute is selected.
Combinations of elements and attributes is also possible. In that case the selected element has to have a given attribute in the structure.
Then operator is evaluated. Operators are the same as for integer numbers, float numbers and for string literals which is lexicographically compared with given value. Number values are compared as expected.
The operator *CONTAINS* can be applied only on string literals.
In script, the operator evaluation is encapsulated into a function which returns *true* if the value from the XML file meet the requirements of the operator (equals given value, contains the value,is greater than the given value or is lower than the given value).
If *NOT* is entered in WHERE part of the query, then finding will be changed and script will find every value except the given one. *NOT* operators can accumulate. Finding depends on even or uneven count. Even count means normal search, uneven count means opposite.

3.4 Ordering and limiting the resulting structure

If *ORDER BY* is entered in right form in the query, output have to be ordered by element or element and attribute or just attribute. For ordering an array of couples, in which one part is value from element which will be compared, and the second part is the element from structure which will be in the final ordered structure. If the *LIMIT* is entered in right form in the query, output has to be a limited number of elements. For limiting, a counter which compares number of written elements is used, which also stops the writing when limit is reached.

3.5 Print the resulting structure

Print is made to standard output, which is overloaded in case the argument *–output=filename* is entered and then *stdout* contains object of file which will be created. If *-n* argument is entered,the XML header will not be printed. If *–root=nameofroot* argument is entered the final structure will be enveloped with root element.

4 Registered extension

- **ORD**: Support of ORDER BY clause, ordering can be ascending or descending and order is made lexicographically.