Explanation Assignment 5

XML File Parsing and file loading

In the beginning the given dataset in XML format must be parsed to RDF triples, which can be inserted into the graph database. I used a SAX (incremental) parser for this task. I wrote a content handler for the parser, which collects all necessary data (document id, classifications, author ids, keywords and publication year) for answering the problems. If an XML element is part of two consecutive feeds, it gets corrupted. There are approximately 200 feeds for the whole dataset, and over four million elements, so the probability of a corruption is small but not zero. After the whole dataset has been parsed, I created multiple RDF Graph, added the triples, and saved them into files named triplesX, where X is a sequential number. Afterwards I loaded each file once into the database.

Coauthors

The query for the coauthors problems will select all distinct coauthors of the given author and the author and sorted them. I collected all coauthors into a set and excluded the author.

Msc-Intersection

I used multiple queries, one query per given classification. Each query will select all document ids if the document has the given (sub-)classification. These ids were inserted into a set. Afterwards a calculated the intersection of the sets, so each document in the intersection must match at least one (sub-)classification. Next, I checked if the documents in the intersection match all constraint. The remaining documents are the solution to the problem. I used many queries for this problem type, there is a better solution, but my way should also work.

Top-3-keywords

For the top-3-keyword problems only one query was necessary. The query selects the keyword paired with its number of occurrences if the document meets the given constraints of author and if present, publication year. I inserted all tuples into a list, sorted the list descending by the number of occurrences and only returned the first 3 elements. These are the top-3-keywords.

Coauth-dist

I used a query that will get bigger until a solution has been found. There first query checks if there is a chain of size one between the “from” and the “to” author given in the problem, so a direct link. If there is none the next query will check if there is a chain of size two, so if a random author links the two given authors. Next a chain of size three and so on until a solution has been found.

Writing the solution

After all solutions had been calculated I created a DOM tree containing all solutions and saved it as the solution file.