Universität Freiburg
Institut für Informatik
Dr. Fang Wei-Kleiner

Georges-Köhler Allee, Geb. 51
D-79110 Freiburg
fwei@informatik.uni-freiburg.de

Advanced Databases and Information Systems Project II
Implementation of Join Algorithm for SPARQL Query Processing
Summer Term 2022

Deadline: 17.7.2022

Submission Guidelines:
For this experiment you need to submit a project report in pdf.
Upload your report to ILIAS (by Programming Projects):
`https://ilias.uni-freiburg.de/goto.php?target=crs_2631216&client_id=unifreiburg`.
Check this link for a template in Latex writing a report:
`https://fachschaft.tf.uni-freiburg.de/informationen/dokumentvorlagen`
The report should consist of the following parts: Problem statement, Algorithm description, Dataset description, Experiment and analysis, Conclusion.
We encourage you upload your code to Github.
You may work in a team of two.

The task is to implement two most popular join algorithms, namely Hash join and Sort-merge join algorithms for a specific SPARQL query over an RDF dataset. The Waterloo SPARQL Diversity Test Suite WatDiv dataset from University of Waterloo `https://dsg.uwaterloo.ca/watdiv/` consists of diverse RDF triple stores in different size. Based on two selected datasets from it, the SPARQL query of the form:

`(?a)---follows--->(?b)---friendOf--->(?c)---likes--->(?d)---hasReview--->(?e)`

is to be evaluated. In the query, `(?a), (?b), (?c), (?d), (?e)` are variables, and `follows`, `friendOf`, `likes`, `hasReview` are properties. The answer of the query is the list of mapped values of all the variables `(?a), (?b), (?c), (?d), (?e)`.

The dataset consists of two triple WatDiv triple stores, one with the size of 100 thousand, and another with 10 million triples. Both files can be downloaded from ILIAS.

a) The first task is to pre-process the data. It is required to partition the triples into relations by using vertically partitioned approach, namely for each distinct property, set up a table with 'Subject' and 'Object' as columns. Assume there are n properties in the triple store, then you need to construct n tables. One optional step before the pre-processing, is to build up a dictionary of all strings occurring in the triple store and transform the string values into integers. Since the comparison of integer is much faster than the comparison on string values, this optional step helps improve the efficiency of the join algorithm.

b) The second task is to design and implement hash join and sort-merge join algorithms for the query evaluation. Obviously, our running query can be expressed in the form of SQL given the data set yield by vertically partitioned approach. The corresponding SQL expression is as follows.

```
SELECT follows.subject, follows.object, friendOf.object, likes.object, hasReview.object
FROM follows, friendOf, likes, hasReview
WHERE follows.object = friendOf.subject
      AND friendOf.object = likes.subject
      AND likes.object = hasReview.subject
```

The query in the form of relational algebra is as follows:

$\texttt{follows} \bowtie_{follows.object=friendOf.subject} \texttt{friendOf} \bowtie_{friendOf.object=likes.subject} \texttt{likes} \bowtie_{likes.object=hasReview.subject} \texttt{hasReview}$

Your task is to implement two procedures to obtain the answer of the query. One is based on Hash join and the other is based on Sort-merge join. There is no restriction on the programming language, however the preferred language is C++ or Python. Run both procedure on the two datasets provided, and record the time cost of both processes. Analyse the time costs and given possible explanation in your report.

c) The third task is to design and implement an improvement algorithm regarding the running time. There is no restrictions on the approaches. Possible candidates are: use radix join algorithm, use a different hash function or hashing scheme, partition the data before the join operation, or use parallel sorting algorithms. Other options are for instance building indexes on the data before the query evaluation. Of course you might even invent a completely new join algorithm. In summary everything is allowed as long as the result is correct. Describe your improvement approach in the report.