Exercises for the book

Semantic Web Databases

by Sven Groppe

Solutions

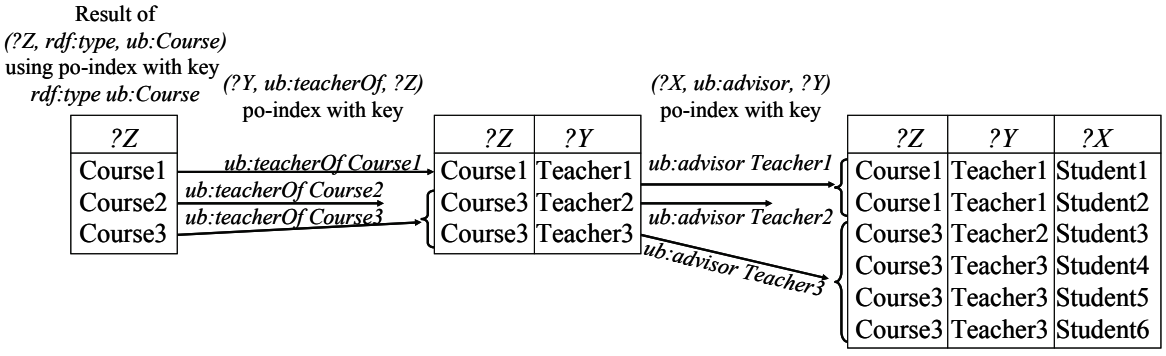
Chapter 6 – Physical Optimization

1. The formula to determine the number of page accesses is $b_S + \lceil b_S / (m\text{-}k) \rceil * b_R$, as b_S pages are read in for the intermediate result S, and $\lceil b_S / (m\text{-}k) \rceil$ times the inner loop is processed in which the whole R consisting of b_R pages is read. To reduce the number of page accesses, $m\text{-}k$ should be made as large as possible, which is the case for k=1. Furthermore, we can choose the smaller intermediate result for the outer loop as S, such that the summand b_S is minimized. In our example, $\lfloor 8192/100 \rfloor = 81$ intermediate results of A fit in a page, i.e., $b_A = \lceil 8\,000\,000 / 81 \rceil = 98\,766$. Likewise, $\lfloor 8192/50 \rfloor = 163$ intermediate results of B fit in a page, i.e., $b_B = \lceil 30\,000\,000 / 163 \rceil = 184\,050$. The number of pages in main memory, m, is $\lfloor (0.5*1024*1024)/8 \rfloor = 65\,536$. Thus, the overall number of page accesses of the block-based nested loop join is $b_S + \lceil b_S / (m\text{-}k) \rceil * b_R = 98\,766 + \lceil 98\,766 / (65\,536\text{-}1) \rceil * 184\,050 = 466\,866$ for this example. For the optimized version of the block-based nested loop join, we have to read k pages less in each (except of the first) iteration of the outer loop. Thus we have to modify our formula for the number of page accesses slightly to $b_S + \lceil b_S / (m\text{-}k) \rceil * (b_R\text{-}k) + k$. In our previous example, the number of page accesses would be reduced to $b_S + \lceil b_S / (m\text{-}k) \rceil * (b_R\text{-}k) + k = 98\,766 + \lceil 98\,766 / (65\,536\text{-}1) \rceil * (184\,050\text{-}1) + 1 = 466\,865$.

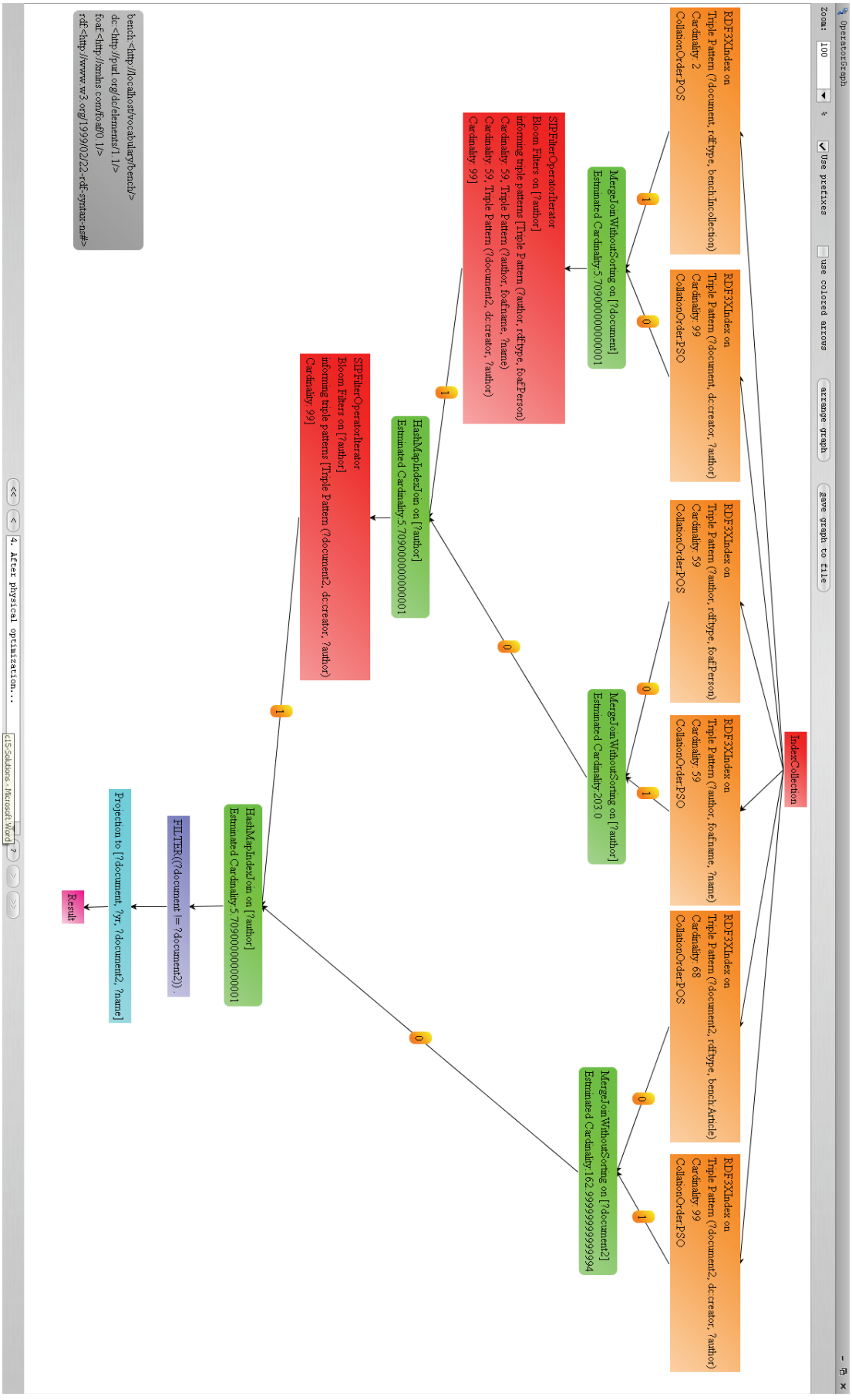
2. If all values are distinct of the bound values for join variables, then the merge join algorithm only reads the intermediate results of R as well as the intermediate results of S one time, such that $b_R + b_S$ page accesses are necessary. The number of page accesses typically increase only slightly if not all bound values for join variables are distinct. The reason for increasing the page accesses is that the cursor must be reset several times to the position where the same values start.

3. a) Before the partition starts, we have $x_0 = b_A$. The number of pages for a partition is reduced by a factor of m-1 for each partition round. Therefore, we have $x_i = b_A / (m\text{-}1)^i$. To be more precise, we would have to round after each partition round, which we ignore here, as the error is small, but it would complicate the calculation too much. b) Partitioning rounds are done until $x_i \leq m\text{-}1 \Leftrightarrow b_A / (m\text{-}1)^i \leq m\text{-}1 \Leftrightarrow i \geq log_{m-1}(b_A) - 1$. Therefore, $\lceil log_{m-1}(b_A) - 1 \rceil$ partition rounds are necessary. c) For each partition round, we have to read and write each partition one time. As all partitions of one partition round are together as large as the whole intermediate results of the considered operand, we have $2*b_A$ page acesses for A for each partition round ($2*b_B$ page accesses for B respectively). The number of partition rounds is $\lceil log_{m-1}(b_A) - 1 \rceil$ for both, A and B. Computing the join of each partition finally needs b_A+b_B page accesses. Thus, the total number of page accesses is $2*(b_A + b_B)*\lceil log_{m-1}(b_A) - 1 \rceil + b_A + b_B = (b_A + b_B)*(2*\lceil log_{m-1}(b_A) \rceil - 1)$.

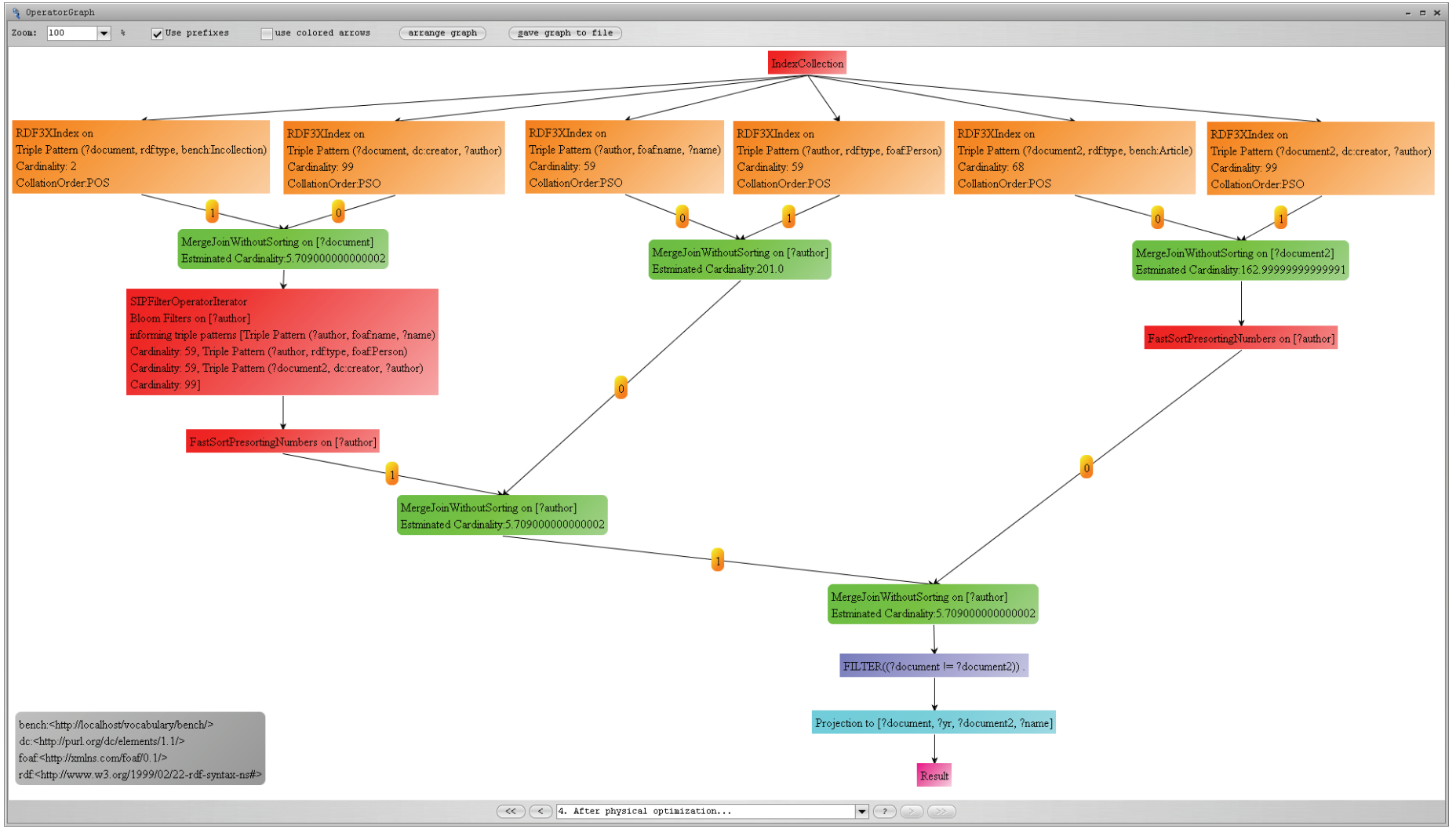
4. The join computation can be done as follows:

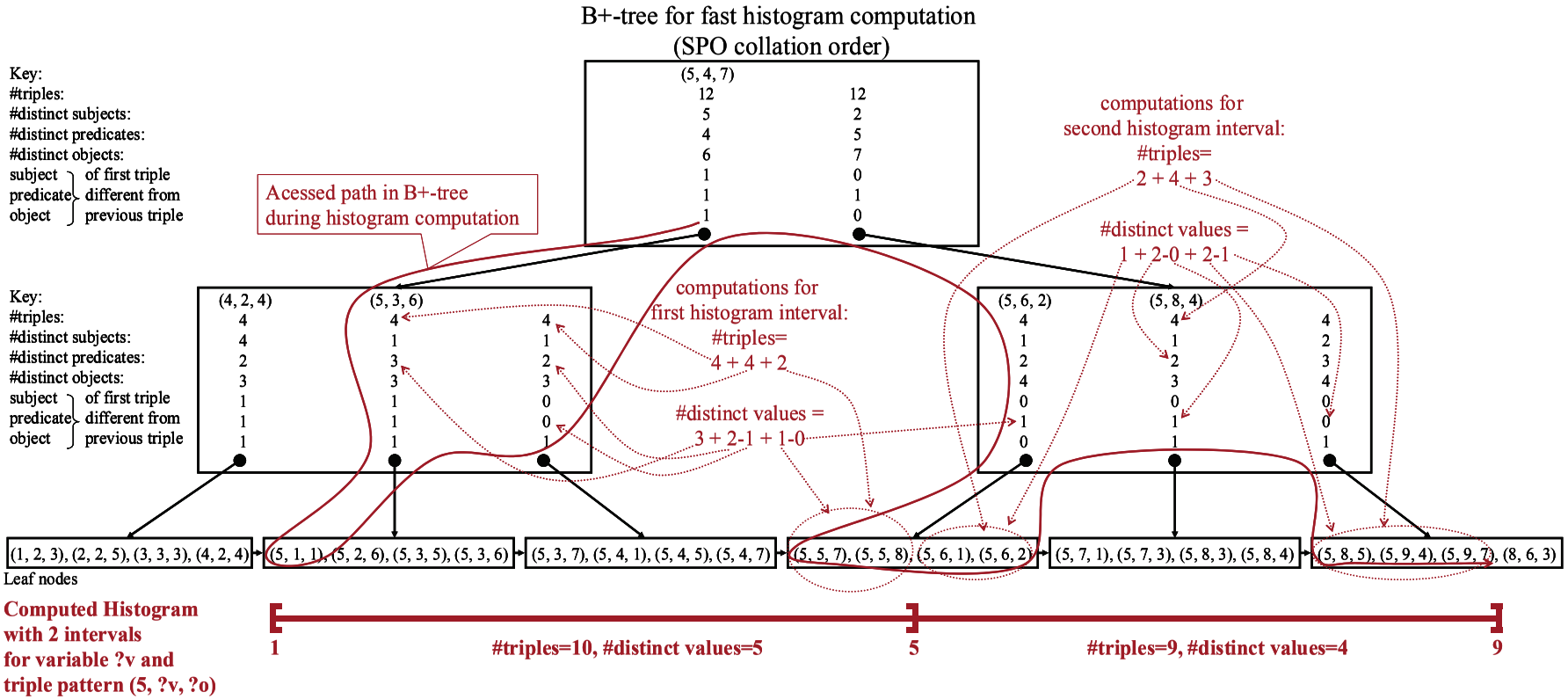


5. Without using our sorting numbering scheme, the following query plan is generated:



With using our sorting numbering scheme, the following query plan is generated, i.e., two hash joins have been replaced with merge joins.


OperatorGraph
Zoom: 100 %
Use prefixes
use colored arrows
arrange graph
save graph to file
IndexCollection
RDF3XIndex on
Triple Pattern (?document, rdf:type, bench:Incollection)
Cardinality: 2
CollationOrder:POS
RDF3XIndex on
Triple Pattern (?document, dc:creator, ?author)
Cardinality: 99
CollationOrder:PSO
RDF3XIndex on
Triple Pattern (?author, foaf:name, ?name)
Cardinality: 59
CollationOrder:PSO
RDF3XIndex on
Triple Pattern (?author, rdf:type, foaf:Person)
Cardinality: 59
CollationOrder:POS
RDF3XIndex on
Triple Pattern (?document2, rdf:type, bench:Article)
Cardinality: 68
CollationOrder:POS
RDF3XIndex on
Triple Pattern (?document2, dc:creator, ?author)
Cardinality: 99
CollationOrder:PSO
MergeJoinWithoutSorting on [?document]
Estminated Cardinality:5.709000000000002
MergeJoinWithoutSorting on [?author]
Estminated Cardinality:201.0
MergeJoinWithoutSorting on [?document2]
Estminated Cardinality:162.99999999999991
SIPFilterOperatorIterator
Bloom Filters on [?author]
informing triple patterns [Triple Pattern (?author, foaf:name, ?name)
Cardinality: 59, Triple Pattern (?author, rdf:type, foaf:Person)
Cardinality: 59, Triple Pattern (?document2, dc:creator, ?author)
Cardinality: 99]
FastSortPresortingNumbers on [?author]
FastSortPresortingNumbers on [?author]
MergeJoinWithoutSorting on [?author]
Estminated Cardinality:5.709000000000002
MergeJoinWithoutSorting on [?author]
Estminated Cardinality:5.709000000000002
FILTER((?document != ?document2)) .
Projection to [?document, ?yr, ?document2, ?name]
Result
bench:<http://localhost/vocabulary/bench/>
dc:<http://purl.org/dc/elements/1.1/>
foaf:<http://xmlns.com/foaf/0.1/>
rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
4. After physical optimization...



6. The histogram index and the computed histogram are: