Exercises for the book

Semantic Web Databases

by Sven Groppe

Chapter 6 – Physical Optimization

1. In this exercise, we want to determine the number of page accesses for the block-based nested-loop join for computing $R \bowtie S$ without any optimization like discussed in this chapter. We assume b_R to be the number of pages of R and b_S the number of pages of S. Furthermore, m is the number of pages in main memory reserved for this join, and k is the number of pages for R, where $1 \leq k \leq m$. How many page accesses are then necessary for the block-based nested-loop join without any optimization? How should k be chosen and which input, R or S, should be taken as R, in order to optimize the number of page accesses of the block-based nested loop join? How many page accesses are there for a join $A \bowtie B$, when A contains 8 000 000 and B contains 30 000 000 solutions, each solution of A has a size of 100 bytes, the solution size of B is 50 bytes, a page contains 8 Kilobytes, and for joining is 0.5 Gigabytes of main memory available? How do we have to modify our formula for the number of page accesses for the optimized variant of the block-based nested loop join?

2. Determine the number of page accesses of the merge join algorithm for computing $R \bowtie S$, when the number of pages for operand R is b_R and the number of pages for operand S is b_S. We assume that all values bound to the join variables are distinct, which is an optimal case for the merge join algorithm.

3. Determine the average number of page accesses of the hash join algorithm for computing $A \bowtie B$. For this purpose, we assume that the number of pages for A is b_A, the number of pages for B is b_B. The number of pages reserved for the hash join is m. a) Determine the number x_i of pages of a partition after i partition rounds. Assume that the used hash functions are perfect hash functions, i.e., the created partitions have equal size. This will never be reached in practice. However, the difference will not be significantly for large datasets and randomly chosen attributes of a hash function. b) How many partition rounds are necessary? c) How many page accesses are necessary for the whole hash join?

4. Use dynamically restricting triple patterns for the join computation between the triple patterns *(?Z, rdf:type, ub:Course)*, *(?Y, ub:teacherOf, ?Z)* and *(?X, ub:advisor, ?Y)* when the po-index is given as follows:

Key *po*	**Triples**
rdf:type ub:Course	*{(Course1, rdf:type, ub:Course), (Course2, rdf:type, ub:Course), (Course3, rdf:type, ub:Course)}*
ub:teacherOf Course1	*{(Teacher1, ub:teacherOf, Course1)}*
ub:teacherOf Course3	*{(Teacher2, ub:teacherOf, Course3), (Teacher3, ub:teacherOf, Course3)}*
ub:advisor Teacher1	*{(Student1, ub:advisor, Teacher1), (Student2, ub:advisor, Teacher1)}*
ub:advisor Teacher3	*{(Student3, ub:advisor, Teacher3), (Student4, ub:advisor, Teacher3), (Student5, ub:advisor, Teacher3), (Student6, ub:advisor, Teacher3)}*

5. Build the query plans for the following SPARQL query with and without using our sorting numbering scheme, and compare both query plans. How many hash joins can be replaced with merge joins when using our sorting numbering scheme?

```
PREFIX rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
PREFIX dc:     <http://purl.org/dc/elements/1.1/>
PREFIX bench:  <http://localhost/vocabulary/bench/>

SELECT ?yr ?name ?document ?document2
WHERE {
  ?document rdf:type bench:Incollection .
  ?document dc:creator ?author .
  ?author foaf:name ?name .
  ?author rdf:type foaf:Person .
  ?document2 dc:creator ?author .
  ?document2 rdf:type bench:Article .
```

```
  Filter( ?document!=?document2)
}
```

6. Add the remaining information in the histogram index given below and compute a histogram with two intervals for the variable *?v* and the key (5, *?v*, *?o*).