Homework Sheet 2 : Costing and Modelling

Question 1: What is the least size in terms of number of pages required as buffer in the main memory to perform a GRACE hash-join of relations R and S? You can assume that $|R|$ is the total number of pages in the hash table for R. (Hint: Think about maximum number of partitions of R and maximum size of each partition).

(a) $|R|^2$

(b) $\log |R|$

(c) $\sqrt{|R|}$

(d) $|R|$

Solution to Question 1: c

Question 2: In a Block Nested Loops Join, we use as the outer relation:

(a) the smaller relation

(b) the larger relation

(c) size does not matter but it is efficient to pick a relation that is sorted, if any

(d) neither size nor sortedness matters

Solution to Question 2: a

Question 3: The best way to split the pool of available buffer pages in Block Nested Loops Join is:

(a) one page for the outer relation and all the rest for the inner relation

(b) split them evenly between the outer relation and the inner relation

(c) as much as possible for the outer relation and just one page for the inner relation

(d) it depends on whether the relations are sorted or not

Solution to Question 3: c

Question 4:
Consider the following two relations:

```
Student(name: String, stid: Int)
(Peter,1)
(Kate, 2)
(Maggie, 3)
(John, 4)

TakesCourse(studid: Int, courseName: String)
(1, Database Systems)
(2, Database Systems)
(2, Quantitative Risk Assessment)
(4, Diversity Management)
```

Write down in which order a Nested Loop Join considers the pairs of these tuples (i.e. you are going to write 16 pairs of the form “`(Peter,1) x (1, Database Systems)`”). Afterwards, write down in which order Block Nested Loop Join considers the pairs, given that you have 2 pages of main memory at your disposal and one page can hold 2 tuples from any of the two relations. How many times does the tuple ”(2, Quantitative Risk Assessment)“ have to be brought into the main memory in the NLJ and how many times in the BNL?

Solution to Question 4:
NLJ:

```
(Peter, 1) x (1, Database Systems)
(Peter, 1) x (2, Database Systems)
(Peter, 1) x (2, Quantitative Risk Assessment)
(Peter, 1) x (4, Diversity Management)
(Kate, 2) x (1, Database Systems)
(Kate, 2) x (2, Database Systems)
```

```
(Kate, 2) x (2, Quantitative Risk Assessment)
(Kate, 2) x (4, Diversity Management)
(Maggie, 3) x (1, Database Systems)
(Maggie, 3) x (2, Database Systems)
(Maggie, 3) x (2, Quantitative Risk Assessment)
(Maggie, 3) x (4, Diversity Management)
(John, 4) x (1, Database Systems)
(John, 4) x (2, Database Systems)
(John, 4) x (2, Quantitative Risk Assessment)
(John, 4) x (4, Diversity Management)
```

BNL:

```
(Peter, 1) x (1, Database Systems)
(Peter, 1) x (2, Database Systems)
(Kate, 2) x (1, Database Systems)
(Kate, 2) x (2, Database Systems)
(Peter, 1) x (2, Quantitative Risk Assessment)
(Peter, 1) x (4, Diversity Management)
(Kate, 2) x (2, Quantitative Risk Assessment)
(Kate, 2) x (4, Diversity Management)
(Maggie, 3) x (1, Database Systems)
(Maggie, 3) x (2, Database Systems)
(John, 4) x (1, Database Systems)
(John, 4) x (2, Database Systems)
(Maggie, 3) x (2, Quantitative Risk Assessment)
(Maggie, 3) x (4, Diversity Management)
(John, 4) x (2, Quantitative Risk Assessment)
(John, 4) x (4, Diversity Management)
```

Question 5: Given:
- Relation X contains 10,000 tuples and 10 tuples can fit into one page
- Relation Y contains 2,000 tuples and 10 tuples can fit into one page too.
- 51 buffer pages.

Compute the cost of the join X $\bowtie_{X.a=Y.b}$ Y in terms of number of pages transferred from disk using a:

(a) Simple nested loop join

(b) Page-oriented simple nested loop join

(c) Block nested loop join

(d) Grace Hash join

Note: Think carefully about which relation should be the outer one.

Solution to Question 5:
X pages: 10,000/10= 1000 pages
Y pages=2,000/10= 200 pages
Y is the smaller relation and therefore used as outer relation.

(a) Cost $= |Y| + \|Y\| * |X| = 200 + 2000 * 1000 = 2,000,200$

(b) Cost $= |Y| + |Y| * |X| = 200 + 200 * 1000 = 200,200$

(c) Use 51-1 = 50 pages for buffering Y. Cost $= |Y| + \frac{|Y|}{50} * |X| = 200 + 4 * 1000 = 4,200$

(d) $|Y| < B^2$. So each partition fits into memory. Cost $= 3 * (|X| + |Y|) = 3 * (1000 + 200) = 3,600$

Question 6: (Final Exam 2014) Assume your system has a disk, a main memory, and a CPU cache (which is much smaller than the available main memory) with a least-recently-used (LRU) cache replacement policy. Assume it is a single-core system. Assume that there are no other active processes or threads. An access to the cache is much faster than a cache miss (which requires to fetch the item from the main memory).

Write the pseudocode of a cache-conscious version of the block-nested loop (BNL) join (a version that greatly reduces the number of cache misses compared to the classical BNL join that does not make provisions for the presence of a cache). Pick suitable names for the parameters you are using (such as page size, size of available main memory [in pages], and cache size). Briefly explain your design.
(Hint: the classical BNL join uses two nested loops over data on disk to fetch data into main memory buffers; then, all the tuples currently in main mem get joined. One could do that by building in-memory hash tables, but let's assume it is simply done by looping. How do you do that looping well given that your cache is much smaller than your main mem?)

Solution to Question 6: Let us assume that R is the smaller relation, M is the size of R on disk (in pages), B is the size of available main memory (in pages), and C is the cache size.

```
foreach block of B–1 pages of R do
 foreach page of S do
  foreach block of size C/2 from R pages in memory do
   foreach block of size C/2 from S pages in memory do
    for all matching in-cache tuples r in R
                     and in-cache tuples s in S
     add <r,s> to result
```

By maximizing the number of pages of R in memory, we reduce the total number of iterations over the S relation ($\lceil \frac{M}{B-1} \rceil$). If we take into account the blocked access to several pages, by allocating more pages for S in memory, we could drastically reduce the number of seeks for S. We reserve one page for the output tuples.
By placing the data in cache, we avoid expensive main memory accesses. The cache sizes for R and S are equal ($\frac{C}{2}$). (Actually, the cache sizes are slightly smaller, as we need to dedicate some space in cache for the output tuples.) We opted for this value ($\frac{C}{2}$) in order to maximize the covered area of the join matrix using only the in-cache tuples.