Data Management – exam of 11/06/2020

Problem 1
If S is a schedule on transactions $T_1, \ldots, T_n$, then the *partial precedence graph* $\mathsf{PPG}(S)$ associated to S is a graph that has the transactions in S as nodes, and has an edge from T_i to T_j if and only if S contains two actions of different types (i.e., one read and one write) $a_i(x)$ in T_i and $a_j(x)$ in T_j on the same element x such that $a_i(x)$ precedes (not necessarily directly) $a_j(x)$ in S. Also, the *write-on graph* $\mathsf{WOG}(S)$ associated to S is a graph that has the transactions in S as nodes, and has an edge from T_i to T_j if and only if there is an x such that $w_j(x)$ is followed by $w_i(x)$ in S, and there is no write action on x in S between $w_j(x)$ and $w_i(x)$. Prove or disprove the following claims:

1. If both $\mathsf{PPG}(S)$ and $\mathsf{WOG}(S)$ are acyclic, then S is view-serializable.
2. If $\mathsf{PPG}(S)$ is acyclic, and $\mathsf{WOG}(S)$ has no edges, then S is conflict-serializable.

Problem 2 Let `R` be a relation with 10.000.000 tuples, each with 50 attributes, occupying 1.000.000 pages, and let us consider the operation of searching for all the tuples of `R` with a given value for the non-key attribute `A`, knowing that `A` contains 100 values uniformly distributed over the tuples of `R`. We consider three methods for representing `R` in secondary storage: (1) `R` is stored as a sorted file with search key `A`, (2) `R` is stored as a heap file with an associated sorted index using alternative 2 with search key `A`, and (3) `R` is stored as a sorted file with search key `A` with an associated sorted index using alternative 2 with search key `A`. Under the assumption that each value and each pointer occupy the same space, tell which is the cost (in terms of number of page accesses) of the search operation in the cases corresponding to the three methods specified above.

Problem 3
Consider the relations `R(`$\underline{\texttt{A}}$`,`$\underline{\texttt{B}}$`,C,D,E,F)` and `Q(`$\underline{\texttt{C}}$`,D)`, where `R` is stored in 20.000 pages of a heap file with an associated B$^+$-tree index with search key $\langle$`A,B`$\rangle$, `Q` is stored in 600 pages of a heap file, each page contains 20 tuples of `R`, each attribute and each pointer occupy the same space, and we know that there are 150 available frames in the buffer. Consider the following query

```
select A from R where A not in (select C from Q)
 union
select C from Q where C not in (select A from R)
```

Show the logical query plan associated to the query, as well as the logical query plan and the physical query plan you would choose for executing the query efficiently. Also, tell which is the cost (in terms of number of page accesses) of executing the query according to the chosen physical query plan.

Problem 4
Given the two relations `R`$_1$`(A,B,C)` and `R`$_2$`(C,D)`, the following equivalences were intended to be used during the optimization of logical query plans involving `R`$_1$ and `R`$_2$:

1. If `R`$_1$ or `R`$_2$ (or both) is a bag, i.e., may contain duplicates, then $\delta(\mathtt{R}_1 \bowtie \mathtt{R}_2) = \delta(\mathtt{R}_1) \bowtie \delta(\mathtt{R}_2)$.
2. If `R`$_1$ and `R`$_2$ are sets, then $\delta(\pi_{\mathtt{A,B,C}}(\mathtt{R}_1 \bowtie \mathtt{R}_2)) = \pi_{\mathtt{A,B,C}}(\mathtt{R}_1 \bowtie \mathtt{R}_2)$.

For each of the above equivalences, prove or disprove, explaining your answer in details, that it is valid, and can indeed be used in query optimization. We remind the students that δ denotes duplicate elimination, π denotes projection (without duplicate eliminations) and $\bowtie$ denotes natural join, i.e., the join of two relations based on equality on common attributes.

Problem 5
Let `R`$_1$`(A,B,C,D)` and `R`$_2$`(A,B,C,D)` be two relations stored in two heap files with $B(\mathtt{R}_1)$ and $B(\mathtt{R}_2)$ pages, respectively. We know that $B(\mathtt{R}_1) < B(\mathtt{R}_2)$, $B(\mathtt{R}_1) > K$, and $B(\mathtt{R}_2) > K$, where K is the number of available frames in the buffer. We have to compute the intersection of `R`$_1$ and `R`$_2$, in four different scenarios: (a) both `R`$_1$ and `R`$_2$ are sets; (b) `R`$_1$ is a set and `R`$_2$ is a bag; (c) `R`$_1$ is a bag and `R`$_2$ is a set; (d) both `R`$_1$ and `R`$_2$ are bags. For each of the above scenarios, tell whether the "classical block-nested loop algorithm" can be used or not; if the answer is negative, then motivate the answer in detail, and if the answer is positive, then briefly describe the algorithm and its cost (in terms of number of page accesses). We remind the students that the "classical block-nested loop algorithm" reads all the pages of the outer relation in blocks, and for each block, it reads all the pages of the inner relation, and while doing this, it does not execute any write operation other than the writes of the pages of the result.