Data Management – exam of 09/09/2022

Problem 1
Suppose that for every schedule S on transactions $\{T_1, \ldots, T_n\}$ and for every element x of the database, we have a sequence $\sigma_{S,x}$ that is a total order on the transactions $\{T_1, \ldots, T_n\}$. Consider a scheduler R that behaves as follows when analysing action $a_i(y)$: if there is an action $b_j(y)$ (with $j \neq i$) preceeding $a_i(y)$ in S such that T_j appears after T_i in $\sigma_{S,y}$, then S is not accepted, otherwise the action $a_i(y)$ is executed and the schedule proceeds. Question: is it true that every schedule S accepted by R is conflict serializable? If your answer to the question is positive, then provide a proof. Otherwise, (i) provide a proof that the answer is negative and (ii) specify a condition on the various $\sigma_{S,x}$ under which the answer becomes true. If you do not know how to solve the problem in general, try to solve it in the case where S is on just two transactions, i.e., the case of $n = 2$.

Solution to problem 1

(i) We provide the proof that the answer is negative by exhibiting a counterexample to the statement, i.e., a schedule S that is accepted by R but is not conflict serializable. Consider $\sigma_{S,x} = \{T_1, T_2\}$ and $\sigma_{S,y} = \{T_2, T_1\}$ and $S = r_1(x)\, w_2(x)\, w_2(y)\, w_1(y)$. It is obvious that R accepts S, and it is also very clear that S is not conflict serializable. Therefore, the answer to the question is negative.

(ii) A condition on the various $\sigma_{S,x}$ under which the answer becomes true is simply that for each pair of elements x, y, we have $\sigma_{S,x} = \sigma_{S,y}$. To prove that this is true, we show that, under the above condition, if a schedule S is not conflict serializable, then it is not accepted by R. In the following, let us call σ_S the sequence $\sigma_{S,x}$ for some x (there is just one such sequence, because all the sequences associated to the elements of the database are equal).

- As a first step, we consider the case of just two transactions, i.e., the case of $n = 2$. Since S is not conflict serializable, S contains action $a_i(x)$ coming before $b_j(x)$ and action $c_j(y)$ coming before $d_i(y)$. Since $\sigma_{S,x} = \sigma_{S,y}$, we have that when R analyzes S it rejects it, because for at least one pair between $\langle a_i(x), b_j(x)\rangle$ and $\langle c_j(y), d_i(y)\rangle$ we have that an action of a transaction T_h on an element z preceeds an action of another transaction T_k such that T_h appears after T_k in $\sigma_{S,z}$.

- We now present the proof for the general case, i.e., for any n. Consider the graph G that has an edge from T_i to T_j if and only if T_i comes before T_j in the sequence σ_S. Notice that, since σ_S is a total order, for each pair of transactions T_h, T_k, either there is an edge in G from T_h to T_k or there is an edge in G from T_k to T_h. Also notice that, being S not conflict serializable, the precedence graph P_S associated to S has at least one cycle. Since G has clearly no cycle, this means that P_S has at least one edge that is not in G. Let us call e such edge, and assume that e is an edge from T_i to T_j derived from the conflicting actions o_i, o_j, with o_i coming before o_j in S. Observe that, since the edge from T_i to T_j is not in G, T_j does not appear before T_i in σ_S, and therefore, T_j appears before T_i in σ_S. It is now immediate to see that when R analyzes o_j it rejects S, because there is an action (in this case o_i) preceeding o_j in S such that T_j appears before T_i in σ_S. So, we have again shown that if S is not conflict serializable, then it is not accepted by R.

Problem 2
Let S be the schedule $\boxed{r_1(X)\, r_3(Z)\, w_1(Y)\, w_2(X)\, w_1(Z)\, w_3(U)\, w_1(V)\, w_2(U)\, w_2(V)\, r_3(T)}$ and answer the following questions: (2.1) Tell whether S is accepted by the 2PL scheduler with exclusive and shared locks. If the answer is yes, then show the 2PL schedule obtained from S by adding suitable lock and unlock commands. If the answer is no, then motivate the answer. (2.2) Tell whether we can insert into S the commit commands for all the transactions in S such that the resulting schedule S' is commit order preserving conflict serializable. If the answer is yes, then show S', otherwise motivate the answer.

Solution to problem 2

(2.1) S is accepted by the 2PL scheduler with exclusive and shared locks, as shown by the following schedule extended with appropriate locking and unlocking commands:

$sl_1(X)\, r_1(X)\, sl_3(Z)\, r_3(Z)\, xl_1(Y)\, w_1(Y)\, xl_3(U)\, sl_3(T)\, u_3(Z)\, xl_1(Z)\, xl_1(V)\, u_1(X)\, xl_2(X)\, w_2(X)\, w_1(Z)\, w_3(U)$
$u_3(U)\, w_1(V)\, u_1(V)\, xl_2(U)\, w_2(U)\, u_2(U)\, xl_2(V)\, w_2(V)\, u_2(V)\, r_3(T)\, u_3(T)$

(2.2) Yes, we can insert into S the commit commands for all the transactions in S such that the resulting schedule S' is commit order preserving conflict serializable. Indeed, it is sufficient to define S' as the schedule obtained from S by adding the commands c_3, c_1, c_2.

Problem 3
Let R(A,B,C) and S(C,D) two relations stored in two heap files with 1.000 and 5.000 pages, respectively. We have to compute the natural join between the two relations. If M is the number of frames, are there values of M for which the block nested loop algorithm is more efficient than the multi-pass algorithm based on sorting for executing the operation? If the answer is negative, then motivate the answer. If the answer is positive, then tell which are the values of M for which the block nested loop algorithm is more efficient than the multi-pass algorithm, again motivating the answer.

Solution to problem 3
It is well known that, if the buffer has M frames, then the cost of the block nested loop is $B(R) + B(R) \times B(S)/(M-2)$. We now compare the cost of the block nested loop algorithm with that of the multi-pass algorithm based on sorting, taking into account the number n of passes. As usual, we consider the cost in terms of number of page accesses.

- $n = 1$: the cost of the multi-pass algorithm (actually, the 1-pass algorithm) in this case is 6.000, and it is obvious that we can apply the 1-pass algorithm if $M \geq 1.002$. Since the cost of the block nested loop in this case is also 6.000, we conclude that for such values the block nested loop is not more efficient than the multi-pass algorithm.

- $n = 2$: we know that the cost of the multi pass algorithm (actually, the 2-pass algorithm) in this case is $3 \times (1.000 + 5.000) = 18.000$. We also know that we can apply the 2-pass algorithm if $M \times (M-1) \leq (1.000 + 5.000)$, i.e., if $M \times (M-1) \leq 6.000$, which means for $M \geq 78$. So, we need to find the values of M for which $B(R) + B(R) \times B(S)/(M-2) < 18.000$. We conclude that the condition on M for which the block nested loop algorithm is more efficient than the 2-pass algorithm is $M \geq 297$.

- $n \geq 2$: we know that the multi-pass algorithm requires more than 2 passes in the case where $M < 78$. In particular, for $M = 77$, the cost of the multi pass (actually, the 3-pass algorithm) is $5 \times 6.000 = 30.000$. For the same value, the cost of the block nested loop algorithm is $1.000 + 1.000 \times 5.000/75 = 67.667$. It is not difficult to conclude that for all values of M less than 78, the block nested loop algorithm is not more efficient that the multi-pass algorithm.

In conclusion, the values of M for which the block nested loop algorithm is more efficient than the multi-pass algorithm are those in the range $[292, 1.002]$.

Problem 4
In this exercise we assume that every value and every pointer occupies the same number of bytes and the buffer has 100 frames. Let R(A,B,C) and S(D,E,F,G,H,L) be two relations stored as heap files. R has 1.000 pages, where each page has 100 tuples, and the attribute A has 500 values uniformly distributed over the tuples of R. S has 2.000 pages and has an associated B^+-tree index using alternative 2 with search key equal to the primary key $\langle$D,E$\rangle$. Consider the query `select A, F from R,S where A=20 and B=D and C=E` and answer the following questions: (4.1) Determine which is the most efficient between the "index-based" and the "two-pass" strategies for the above query, showing and comparing the cost of the algorithms based on the two strategies. (4.2) Would the answer change if the number of pages of R were 20.000?

Solution to problem 4
As usual, we express the cost in terms of number of page accesses. Also, we refer to the two-pass strategy based on sorting.

(4.1) R has $1.000 \times 100 = 100.000$ tuples, and only $100.000/500 = 200$ of them have the value 20 for A. Since 50 tuples of S fit in one page, S has $2.000 \times 50 = 100.000$ tuples, and the associated index is

unclustering (because the heap storing S is obviously not sorted) and therefore dense. This means that the leaves of the index must store 100.000 data entries. Taking into account the 67% occupancy rule and the fact that each data entry has 3 fields (D, E and the pointer), we conclude that each leave can store 67 data entries, which implies the the leaves of the index are 100.000/67=1.493. Since the fan-out of the tree is 75, we have that the cost of accessing the index with a specific value of the search key is $log_{75}1.493 = 2$ and since we also need the value F in output, we consider the cost of 3. Since we can obviously limit the index only to the 200 tuples of R with the value 20 for A, the cost of the algorithm for evaluating the query based on the index-based strategy is $1.000 + 200 \times 3 = 1.600$.

As for the two-pass strategy, one might directly apply the known formulas and conclude that it can indeed be applied (because $100 \times 99 \geq 1.000 + 2.000$) with a cost of $3 \times 3.000 = 9.000$. However, it is not difficult to see that we can modify the algorithm following the two-pass strategy as follows: we can read R and keep only the 200 tuples satisfying the condition A=20, store 1 sorted sublists with 2 pages for such 200 tuples sorted on $\langle \mathtt{B}, \mathtt{C} \rangle$, then read S, store 20 sorted sublists (each of 100 pages) with the tuples of S sorted on $\langle \mathtt{D}, \mathtt{E} \rangle$, and then perform the merge step by reading again all the 21 sorted sublists. This means that the cost of the algorithm for evaluating the query based on this modified two-pass strategy is $1.000 + 2 + 2 + 3 \times 2.000 = 7.004$.

Observe also that we could even avoid using the two pass strategy, and adopt the one pass strategy because the 2 pages storing the 200 tuples satisfying the condition A=20 fit in the buffer. In this case the cost of the algorithm would be $1.000 + 2.000 = 3.000$, which is still greater than the cost of the index-based algorithm.

We then conclude that the index-based strategy is the most efficient one.

(4.2) R has $20.000 \times 100 = 2.000.000$ tuples, and only $2.000.000/500 = 4.000$ of them have the value 20 for A. Since we can obviously limit the index only to the 200 tuples of R with the value 20 for A, the cost of the algorithm for evaluating the query based on the index-based strategy is $20.000 + 4.000 \times 3 = 32.000$.

As for the two-pass strategy, one might directly apply the known formulas and conclude that it cannot be applied, because $100 \times 99 < 20.000 + 2.000$. However, it is not difficult to see that we can modify the algorithm as before and obtain a cost of $20.000 + 40 \times 2 + 3 \times 2.000 = 26.080$.

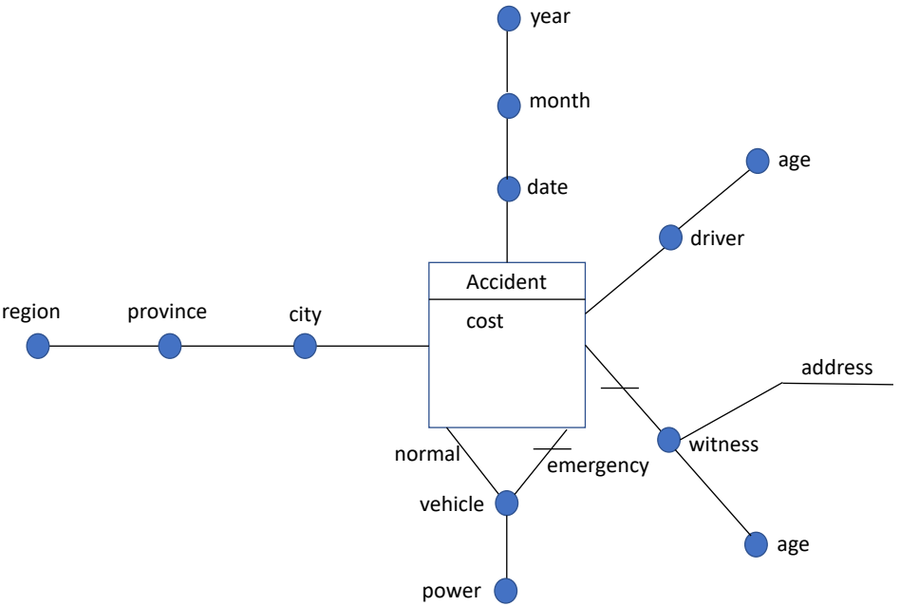
We then conclude that the two-pass strategy is the most efficient one. Observe that in this case also, we could even apply the one pass strategy, with a cost of 22.000, that is the best we can achieve.

Problem 5 (A.Y. 2021/22)

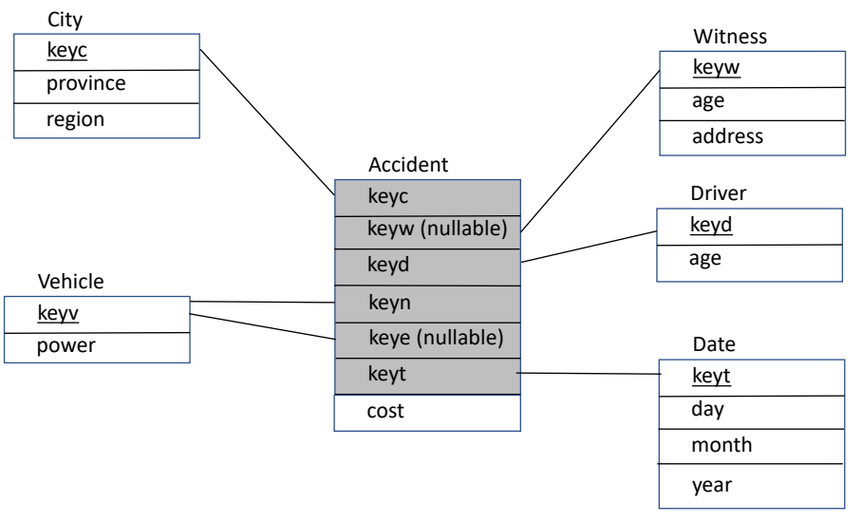
We refer to a data warehouse on car accidents. An accident (having an associated cost) is caused by a driver (with an age) driving a vehicle (with a certain power) in a date (day, month and year) and in a city (belonging to a province, which is part of a region). Sometimes the accident has a witness (again, of a certain age, but also with an address) and requires an emergency vehicle (again, with a certain power). The student is asked to (5.1) show the conceptual representation of the above domain in terms of the Dimensional Fact Model, (5.2) produce the corresponding "star schema", and (5.3) write the SQL query that computes the average cost of the accidents that required an emergency vehicle and occurred in 2021 in the Lazio region.

Solution to problem 4

We show below the conceptual representation of the above domain in terms of the Dimensional Fact Model.



The corresponding star schema is as follows.



Finally, the SQL query that computes the average cost of the accidents that required an emergency vehicle and occurred in 2021 in the Lazio region is:

```
select avg(cost)
from Accident, City, Date
where Accident.keyc = City.keyc and Accident.keyt = Date.keyt and
      keyE is not null and City.region = 'Lazio' and Date.year = 2021
```