# Data Management – exam of 12/07/2023 **(B)**

**Problem 1**
A schedule $S$ is called *dichotomic* if it contains only write actions and every element of the database is written by at most two transactions in $S$. Prove or disprove that every dichotomic schedule is view serializable if and only if it is conflict serializable.

**Problem 2**
Let `S(A,B,C)` and `T(D,E,F)` be two relations, each stored in a heap. A tuple $t$ of `S` is said to be an *upper $h$-match* with `T` if there are at most $h$ tuples of `T` whose value in the attribute `D` is equal to the value of $t$ in the attribute `B`. Let $M$ be the number of buffer frames available, let $Q$ be the query that, given a value for $h$, computes all the tuples in `S` that are upper $h$-matches with `T`, and consider the following questions.

2.1 Describe the conditions under which we can process $Q$ using the one-pass, the two-passes, and the block-nested loop methods, respectively.

2.2 For each of the methods mentioned above, illustrate the corresponding algorithm and tell which is the cost of such algorithm in terms of the number of pages of `S` and `T`.

**Problem 3**
Consider the following schedule $S$:

$$w_1(Z)\ r_2(X)\ w_3(X)\ r_3(Y)\ w_4(Y)\ w_4(X)\ r_2(Y)\ r_1(Y)\ w_2(Z)$$

and answer (with suitable motivations) the following questions: (*i*) Is $S$ conflict serializable? (*ii*) Is $S$ a 2PL schedule (with shared and exclusive locks)? (*iii*) Can we insert the commit commands in such a way that the resulting schedule is accepted by the timestamp-based scheduler (assuming that the timestamp associated to a transaction coincides with the physical time when the transaction starts)? (*iv*) In all the three cases above where the answer is no, provide the answer to the following further question: is there any action $\alpha$ in $S$ such that, if we delete $\alpha$ from $S$, then the answer to the question would switch to yes?

**Problem 4**
The friendship between a city $C_1$ and a city $C_2$ (different from $C_1$) is strong if they have signed an agreement to be twin cities, and is solid if there is a third city $C_3$ such that both $C_1$ and $C_2$ are twin cities with $C_3$. Let `S(A,B,C)` be a relation (with key ⟨`A`,`B`⟩), storing all the tuples $\langle x, y, z\rangle$ such that city $x$ and $y$ are twin cities and $z$ is the year of the agreement between them. Obviously, `S` is symmetric. We know that (*i*) no city has more than 400 twins, (*ii*) associated to `S` there is a clustering sorted index on attribute `A` whose cost for retrieving the first tuple with a certain value for `A` is 5 page accesses, (*iii*) every page has space for 900 tuples of `R`, and (*iv*) the buffer has 100 frames available. Considering the query $Q$ that, given a specific city $x$, computes the set (therefore, without duplicates) of cities with whom $x$ has a strong or solid friendship, illustrate the most efficient algorithm you can think of for answering $Q$, and tell which is the cost of the algorithm in terms of number of page accesses.

**Problem 5** (only for students enrolled in an A.Y. before 2021/22 who do **not** do the project)
Parallel algorithms for the implementation of relational operators are based on the idea of *horizontal data partitioning*, by which we can split a relation in various chunks, each one stored at a different node.

5.1 Describe the techniques that can be used for partitioning.

5.2 Among the described techniques, tell which is the most common one and explain why.

5.3 Illustrate the idea for designing a parallel algorithm that, given a table `S`, based on the partitioning technique mentioned for item 2) above, and depending on the number of buffer frames available at the various nodes, computes the relation obtained from `S` by the "group by" operator. Also, discuss the cost of the algorithm.

5.4 Illustrate the idea for designing a parallel algorithm that, based on the partitioning technique mentioned for item 2) above, and depending on the number of buffer frames available at the various nodes, computes the set intersection of two relations given in input, also discussing the cost of the algorithm.