Data Management – AA 2015/16 – exam of 06/07/2016

Problem 1
In the context of buffer management, explain the goal of the FIX operation, and describe the method used by the buffer manager for executing such operation.

Problem 2
A relation R with attributes A, B, C is stored in 8.000 pages (with 100 tuples per page), a relation S with attributes D, E is stored in 5.000 pages (with 100 tuples per page), and our DBMS has 122 free buffer frames. We want to compute the join of R and S on the condition $C = D$, and we know that the 200 different values stored in the attribute C are uniformly distributed in the tuples of R, and the 160 different values in the attribute D are uniformly distributed in the tuples of S.

2.1 Can we use the block nested-loop algorithm?
2.2 Can we use a two-pass algorithm, and if yes, in which variant?

For each of the two cases (1 and 2), (*i*) if the answer is no, then explain the answer; (*ii*) if the answer is yes, then describe in detail how the algorithm (or, for case 2, the variant chosen) works, and tell which is the cost of the algorithm for computing the above join in terms of number of page accesses.

Problem 3
Consider the following schedule

$$S = r_1(z)\, r_3(x)\, w_3(z)\, w_4(x)\, r_1(x)\, w_2(x)\, w_2(v)\, w_5(v)\, w_5(y)\, r_1(y).$$

3.1 Tell whether S is conflict-serializable or not, explaining the answer in detail.
3.2 If S is conflict-serializable, then describe all the serial schedules that are conflict-equivalent to S, otherwise tell which is the minimal number of transactions that must be deleted from S so that the resulting schedule is conflict-serializable.
3.3 Tell whether S is view-serializable or not, explaining the answer in detail.
3.4 Tell whether S is in ACR (Avoid Cascading Rollback) or not, explaining the answer in detail.

Problem 4
In the DBMS M, every transaction T has an identifier $id(T)$ which is an integer. The concurrency control mechanism of M is such that in a schedule S a transaction T_i is allowed to write on another transaction T_j (i.e., the mechanisms does not block the schedule if it contains a write action of T_j on element x such that the next action on x in S is a write action of T_i on x) if and only if $id(T_i) > id(T_j)$, and is allowed to read from another transaction T_h (i.e., the mechanisms does not block the schedule if it contains a write action of T_h on element x in S such that the next action on x is a read action of T_i on x) if and only if $id(T_i) > id(T_h)$. Prove of disprove that every schedule accepted by the concurrency control mechanism of M is conflict-serializable.

Problem 5
Consider the relation `TICKET(code,person,concert,cost)` that stores information about concert tickets, with the code of the ticket, the person who bought it, the concert for which the ticket was bought, and the cost of the ticket. The relation has 4.000.000 tuples, stored in 400.000 pages, and has 10.000 different values in the attribute `cost`, uniformly distributed in the relation. We assume that all fields and pointers have the same length, independently of the attribute. There is an ISAM index on `TICKET` with search key `cost`, using alternative 2, with one data entry for each page of `TICKET`. Consider the query that asks, for each cost in a given range of 20 values, the number of concerts whose ticket has a cost in the given range, and tell how many page accesses we need for computing the answer to the query.