Data Management – exam of 22/02/2013

Problem 1 Consider the following schedule

$$S = r_2(v)\, r_1(y)\, r_3(y)\, r_3(t)\, r_1(t)\, r_1(x)\, r_2(x)\, w_1(x)\, r_2(z).$$

1.1 Tell whether S is a 2PL schedule with only exclusive locks, explaining the answer in detail.
1.2 Tell whether S is a 2PL schedule with both shared and exclusive locks, explaining the answer in detail.
1.3 Tell whether S is a strong strict 2PL schedule with both shared and exclusive locks, explaining the answer in detail.
1.4 Tell whether S is a 2PL schedule with both shared and exclusive locks, but with no lock upgrade, explaining the answer in detail.

Problem 2 A schedule S on trasactions $T_1, \ldots, T_n$ is called *soft* if (i) the commit command c_i of every transaction in $\{T_1, \ldots, T_n\}$ appears in S, (ii) each read action in S reads only from transactions that have already committed, and (iii) no write action in S writes on another transaction in S. Prove or disprove that every soft schedule is a 2PL schedule with both shared and exclusive locks.

Problem 3 Consider the following schedule

$$S = r_2(v)\, r_1(x)\, w_4(v)\, w_3(y)\, w_2(x)\, r_1(y)\, w_4(y)\, w_3(v).$$

and tell whether S is conflict-serializable or not. If the answer is yes, then show a serial schedule that is conflict-equivalent to S. If the answer is no, then tell which is the minimal set of transactions to remove from S in such a way that the remaining schedule is conflict-serializable.

Problem 4 Assume that a system failure occurs when the log is as follows (where B means "begin", C means "commit", A means "abort", and CK means "checkpoint"):

$B(T_1), D(T_1, o_1, a_1), B(T_2), U(T_2, o_1, b_1, a_1), CK(T_1, T_2), C(T_2), B(T_3), U(T_1, o_2, a_1, c_3), I(T_3, o_3, a_3),$
$CK(T_1, T_3), B(T_4), D(T_4, o_4, a_5), C(T_3), I(T_4, o_5, b_5), A(T_4), U(T_1, o_6, a_6, a_7), B(T_5), U(T_5, o_6, a_7, a_8)$

Illustrate all the actions performed by the recovery subsystem when the failure occurs, assuming that a mixed effect strategy is adopted.

Problem 5 Consider the relation `DEPT(deptcode,chair)` that stores information about which is the chair of the various departments, and the relation `PROF(profcode,name,age)`, that stores name and age of each professor. Note that `deptcode` is the key of `DEPT`, and `profcode` is the key of `PROF`. We assume that the number of departments is 2000, the number of professors is 10.000, the size of every page is 120K, and the size of every attribute, every record id, and every pointer is 2K. Also, we assume that the following are the most important queries on the two relations:

Query 1	**Query 2**
select *	select deptcode
from PROF	from DEPT
where age > 60	where chair not in (select profcode
order by profcode	from PROF
	where age < 40)

Query 1 asks for name and age of all professors whose age is greater than 60, ordered by the value of attributes `profcode`. Query 2 asks for the `deptcode` of every department whose chair is not a professor of less than 40 years old. We assume that when executing Query 2, the system scans the relation `DEPT`, and for each of its tuples, it checks whether the value of the attribute `chair` appears among the `profcode` of the tuples of `PROF` whose value of `age` is less than 40.

Tell which method for representing the two relations you would choose in order to optimize the computation of the above queries, explaining in detail your answer. Also, based on the method chosen, tell how many pages are accessed both during the execution of Query 1, and during the execution of Query 2.