T-106.5220 Transaction Management in Databases

Exam, March 12th, 2009

Write the following clearly on top of every paper you submit: "T-106.5220, March 12th, 2009", your full name, student ID and study programme, and the **total number of papers** you submit.

1. (6p) Explain the following concepts briefly:
   (a) "steal" buffering policy
   (b) unrepeatable read
   (c) fuzzy checkpoint
   (d) space map
   (e) latch-coupling
   (f) Commit-LSN

2. (6p) We apply the key-range locking protocol. What locks are acquired by the transactions in the following histories and when are these locks released? Are the histories possible under the key-range locking protocol? What isolation anomalies (dirty writes, dirty reads, unrepeatable reads) do the histories contain? In all cases, the database is initially empty.
   (a) $B_1 I_1[4] B_2 I_2[6] C_2 B_3 I_1[2] C_3 C_1$
   (b) $B_1 I_1[4] C_1 B_2 R_2[4, >1] B_3 I_3[3] C_3 I_2[7] C_2$
   (c) $B_1 I_1[4] C_1 B_2 I_2[9] D_2[4] B_3 I_3[4] C_3 C_2$

3. (a) (3p) The database applies the multi-granular key-range locking protocol using the following three granules: database, relation, and tuple. The database contains a relation $r(\underline{X}, V)$. Transaction T executes the following SQL statements:
   insert into r **values** $(4, 4)$;
   select * **from** r;
   commit.
   Which locks does the transaction need to acquire while executing the statements? Note that the statements are executed one by one in the above order, and that when the **insert** statement is executed, it is not known that the next statement will be the **select**.
   (b) (3p) Describe briefly the isolation level called cursor stability. Which isolation anomalies does it allow and which ones does it prevent? Give a short example of a situation where this isolation level can be used.

4. (A short answer is sufficient for all of the questions.)
   (a) (2p) How can the database management system detect a deadlock caused by some transactions?
   (b) (1p) How does the wait-or-die (aka wait-die) protocol for preventing deadlocks work?
   (c) (1p) How does the wound-or-wait (aka wound-wait) protocol for preventing deadlocks work?
   (d) (1p) Can the protocols in (b) and (c) cause starvation?
   (e) (1p) How do update-mode locks (U-locks) help in preventing deadlocks?

5. (6p) The contents of the log on disk at the time of a system crash are the following:
   101: $\langle$*begin-checkpoint*$\rangle$
   102: $\langle$*transaction-table*, $\{\}\rangle$
   103: $\langle$*page-table*, $\{\}\rangle$
   104: $\langle$*end-checkpoint*$\rangle$
   105: $\langle T_1, B\rangle$
   106: $\langle T_1, I, p, i_1, 7, 3, 105\rangle$
   107: $\langle T_2, B\rangle$
   108: $\langle T_2, I, p, i_2, 18, 6, 107\rangle$
   109: $\langle T_3, B\rangle$
   110: $\langle T_3, I, p, i_3, 22, 6, 109\rangle$
   111: $\langle T_3, C\rangle$
   112: $\langle T_2, D, p, i_4, 14, 2, 108\rangle$
   113: $\langle T_2, A\rangle$
   114: $\langle T_2, D^{-1}, p, i_4, 14, 2, 108\rangle$
   115: $\langle T_1, I, p, i_5, 24, 1, 106\rangle$
   What actions are involved in restart recovery when using the ARIES algorithm? What log records are generated and when is the log forced to disk? Assume that Page-LSN = 108 in the disk version of p. We assume that all inverse operations in the undo pass can be performed physically.