Concurrency control and scheduling

CSE 444, summer 2010 — section 5 worksheet

July 22, 2010

Our notation for actions in a schedule:

- st_k: transaction T_k begins
- $r_k(X)$: T_k reads database element X
- $w_k(X)$: T_k writes database element X
- com_k: T_k commits

Other notation will be introduced as needed.

1 Two-phase locking

For each of the following schedules, suppose that we add one lock action ($L_k(X)$) and one unlock action ($U_k(X)$) for each database element that is used.

1. $r_1(A)$, $w_1(B)$
2. $r_2(A)$, $w_2(A)$, $w_2(B)$

For each schedule, please answer the following questions:

1. Say when each lock and unlock action can appear relative to the other actions (both read/write and lock/unlock). Don't worry about ensuring two-phase locking for now, but make sure that every element is locked before use and unlocked after use.

2. How many possible schedules are allowed by the rules you just gave?

3. If we enforce two-phase locking, how do your rules change, and how many schedules are now allowed?

2 Timestamps

Each of the following schedules is presented to a timestamp-based scheduler. Assume that the read and write timestamps of each element start at 0 ($\mathrm{RT}(X) = \mathrm{WT}(X) = 0$), and the commit bits for each element are set ($\mathrm{C}(X) = 1$). Please tell what happens as each schedule executes.

Correction: In the handout the commit bits were listed as initially set to 0 rather than 1; this has been corrected above.

1. st_1, st_2, st_3, $r_1(A)$, $r_2(B)$, $w_1(C)$, $r_3(B)$, $r_3(C)$, $w_2(B)$, $w_3(A)$

2. st_1, st_3, st_2, $r_1(A)$, $r_2(B)$, $w_1(C)$, $r_3(B)$, $r_3(C)$, $w_2(B)$, $w_3(A)$

3. st_1, st_2, st_3, $r_1(A)$, $r_2(B)$, $r_2(C)$, $r_3(B)$, com_2, $w_3(B)$, $w_3(C)$

4. st_1, st_2, $r_1(A)$, $r_2(B)$, $w_2(A)$, com_2, $w_1(B)$

5. st_1, st_3, st_2, $r_1(A)$, $r_2(B)$, $r_3(B)$, $w_3(A)$, $w_2(B)$, com_3, $w_1(A)$

6. st_1, $r_1(A)$, $w_1(A)$, st_2, $r_2(C)$, $w_2(B)$, $r_2(A)$, $w_1(B)$

3 Multi-version timestamps

Tell what happens during the following schedules if we use a *multi-version* timestamp scheduler. What happens if the scheduler does not maintain multiple versions?

Correction: Commit actions have been added after the last action of each transaction; solutions changed accordingly.

1. st_1, st_2, st_3, st_4, $w_1(A)$, com_1, $w_2(A)$, $w_3(A)$, com_3, $r_2(A)$, com_2, $r_4(A)$, com_4

2. st_1, st_2, st_3, st_4, $w_1(A)$, com_1, $w_3(A)$, com_3, $r_4(A)$, com_4, $r_2(A)$, com_2

3. st_1, st_2, st_3, st_4, $w_1(A)$, com_1, $w_4(A)$, com_4, $r_3(A)$, com_3, $w_2(A)$, com_2

4 Validation

For the following schedules:

- $R_k(X)$ means "transaction T_k starts, and its read set is the list of database elements X,"
- V_k means "T_k tries to validate," and
- $W_k(x)$ means "T_k finished, and its write set was X."

Clarification: Remember that each transaction must inform the scheduler of *both* its read and write sets when it begins, or when it validates (at the latest). While the notation we use implies otherwise, and hence is slightly confusing, we use it to be consistent with your textbook's notation.

Tell what happens when each schedule is processed by a validation-based scheduler.

1. $R_1(A,B)$, $R_2(B,C)$, $R_3(C)$, V_1, V_2, V_3, $W_1(A)$, $W_2(B)$, $W_3(C)$

2. $R_1(A,B)$, $R_2(B,C)$, $R_3(C)$, V_1, V_2, V_3, $W_1(C)$, $W_2(B)$, $W_3(A)$

3. $R_1(A,B)$, $R_2(B,C)$, $R_3(C)$, V_1, V_2, V_3, $W_1(A)$, $W_2(C)$, $W_3(B)$

4. $R_1(A, B)$, $R_2(B, C)$, V_1, $R_3(C, D)$, V_3, $W_1(C)$, V_2, $W_2(A)$, $W_3(D)$