COMP207 Tutorial Exercises
Week 5 (2^{nd}/4^{th} November)

The exercises below provide the opportunity to practice the concepts and methods discussed during previous week's videos/reading material. If you haven't done so, it is worthwhile to spend some time on making yourself familiar with these concepts and methods. Don't worry if you cannot solve all the exercises during the tutorial session, but try to tackle at least one or two of them. If at some point you do not know how to proceed, you could review the relevant material from the videos/reading material and return to the exercise later.

Recoverable, Cascadeless, and Strict Schedules

Exercise 1 (Exercise 20.24 in [1])**.** For each of the following schedules, determine if the schedule is (A) recoverable, (B) cascadeless, (C) strict, (D) non-recoverable. Try to determine the strictest recoverability condition that each schedule satisfies.

(a) $S_1 : r_1(X);\ r_2(Z);\ r_1(Z);\ r_3(X);\ r_3(Y);\ w_1(X);\ c_1;\ w_3(Y);\ c_3;\ r_2(Y);\ w_2(Z);\ w_2(Y);\ c_2$

(b) $S_2 : r_1(X);\ r_2(Z);\ r_1(Z);\ r_3(X);\ r_3(Y);\ w_1(X);\ w_3(Y);\ r_2(Y);\ w_2(Z);\ w_2(Y);\ c_1;\ c_2;\ c_3$

(c) $S_3 : r_1(X);\ r_2(Z);\ r_3(X);\ r_1(Z);\ r_2(Y);\ r_3(Y);\ w_1(X);\ c_1;\ w_2(Z);\ w_3(Y);\ w_2(Y);\ c_3;\ c_2$

Exercise 2 (Exercise 19.1.1 in [2])**.** What are all the ways to insert lock operations (of the simple lock type only), unlock operations, and commit operations into

$$r_1(X);\ r_1(Y);\ w_1(X);\ w_1(Y)$$

so that the transaction T_1 is:

(a) Two-phase locked, and strict two-phase locked.

(b) Two phase locked, but not strict two-phase locked.

Timestamp-based deadlock detection

As pointed out in the video "No cascading-rollbacks!", deadlocks may arise even if transactions are scheduled using strict two-phase locking (strict 2PL). The video "Detecting deadlocks" covered several techniques for deadlock detection, among them two techniques based on timestamps: *wait-die* and *wound-wait*.

Exercise 3. Consider the following schedules:

- $S_1 : xl_1(X);\ r_1(X);\ sl_2(Y);\ r_2(Y);\ xl_2(X);\ w_2(X);\ u_2(X);\ u_2(Y);\ w_1(X);\ u_1(X)$
- $S_2 : sl_1(X);\ r_1(X);\ xl_2(Y);\ r_2(Y);\ xl_1(Y);\ r_1(Y);\ w_2(Y);\ u_2(Y);\ w_1(Y);\ u_1(X);\ u_1(Y)$

For each of the schedules, decide if a lock request is denied, and if so give the first lock request that is denied and say what happens in this case under the

(a) wait-die scheme;

(b) wound-wait scheme.

Assume that T_1 arrives earlier than T_2.

Timestamp-based scheduling

Exercise 4 (Exercise 18.8.1 in [2])**.** Below are several sequences of start events and read/write operations (here, st_i means that transaction T_i starts):

(a) $st_1;\ st_2;\ r_1(X);\ r_2(Y);\ w_2(X);\ w_1(Y)$

(b) $st_1;\ r_1(X);\ st_2;\ w_2(Y);\ r_2(X);\ w_1(Y)$

(c) $st_1;\ st_2;\ st_3;\ r_1(X);\ r_2(Y);\ w_1(Z);\ r_3(Y);\ r_3(Z);\ w_2(Y);\ w_3(X)$

(d) $st_1;\ st_3;\ st_2;\ r_1(X);\ r_2(Y);\ w_1(Z);\ r_3(Y);\ r_3(Z);\ w_2(Y);\ w_3(X)$

Tell what happens as each of the sequences executes under a (basic) timestamp-based scheduler. Assume that the read and write times of all items are 0 at the beginning of the sequence.

Exercise 5 (Exercise 18.8.2 in [2])**.** Tell what happens during the following sequences of events if a multiversion, timestamp scheduler is used. What happens instead, if the scheduler does not maintain multiple versions?

(a) $st_1; st_2; st_3; st_4; w_1(A); w_2(A); w_3(A); r_2(A); r_4(A);$

(b) $st_1; st_2; st_3; st_4; w_1(A); w_3(A); r_4(A); r_2(A);$

(c) $st_1; st_2; st_3; st_4; w_1(A); w_4(A); r_3(A); w_2(A);$

References

[1] Ramez Elmasri and Shamkant B. Navathe. *Fundamentals of Database Systems*. Pearson Education, 7th edition, 2016.

[2] Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. *Database Systems - The Complete Book*. Pearson Education, 2nd edition, 2009.