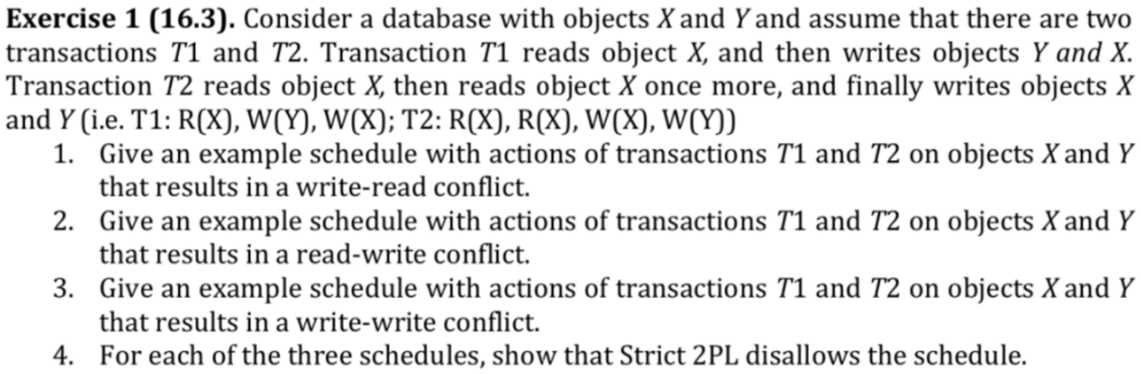
CAS CS 460: Introduction to Database Systems
Fall 2020
Lab 11

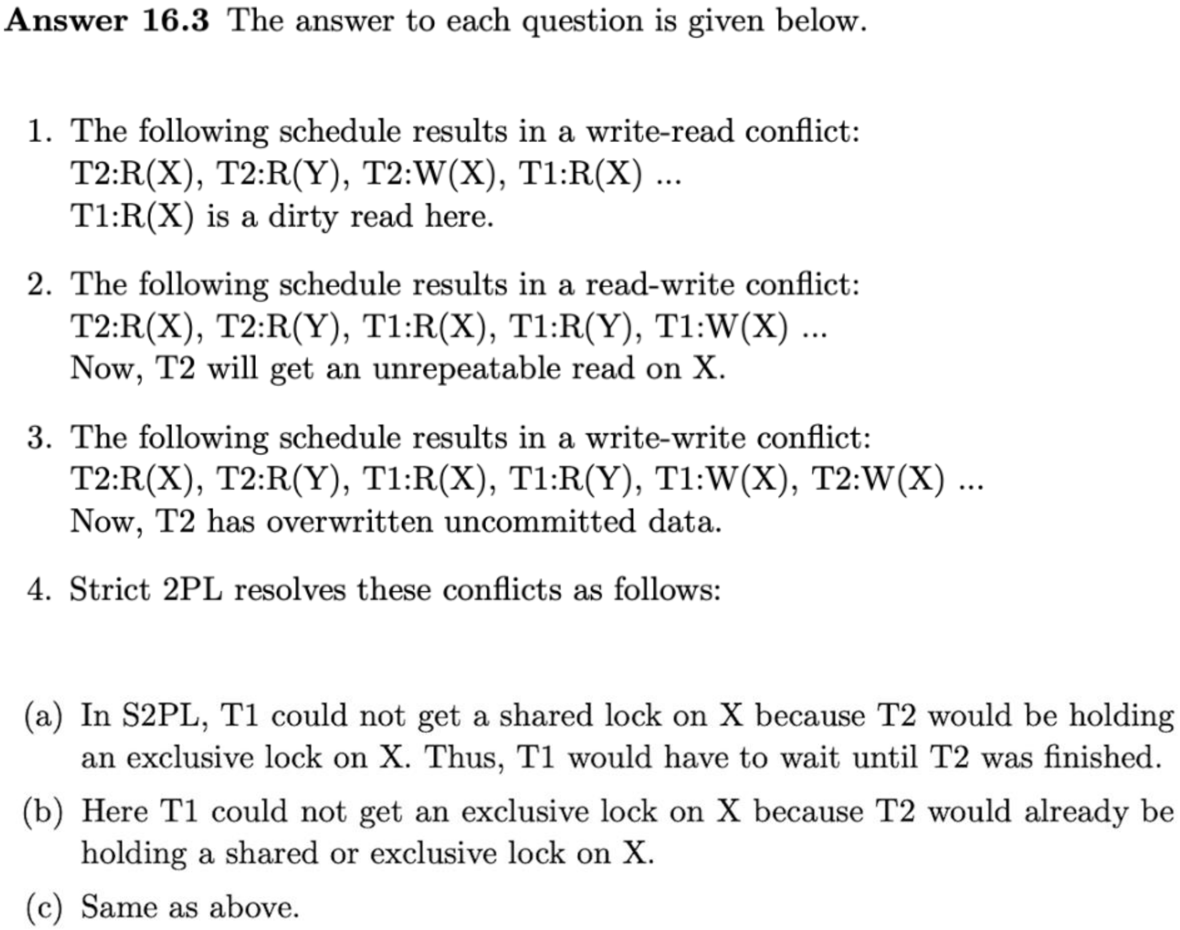
Transactions



Exercise 1 (16.3). Consider a database with objects *X* and *Y* and assume that there are two transactions *T*1 and *T*2. Transaction *T*1 reads object *X*, and then writes objects *Y and X*. Transaction *T*2 reads object *X*, then reads object *X* once more, and finally writes objects *X* and *Y* (i.e. T1: R(X), W(Y), W(X); T2: R(X), R(X), W(X), W(Y))

1. Give an example schedule with actions of transactions *T*1 and *T*2 on objects *X* and *Y* that results in a write-read conflict.
2. Give an example schedule with actions of transactions *T*1 and *T*2 on objects *X* and *Y* that results in a read-write conflict.
3. Give an example schedule with actions of transactions *T*1 and *T*2 on objects *X* and *Y* that results in a write-write conflict.
4. For each of the three schedules, show that Strict 2PL disallows the schedule.

Solution



Answer 16.3 The answer to each question is given below.

1. The following schedule results in a write-read conflict:
   T2:R(X), T2:R(Y), T2:W(X), T1:R(X) ...
   T1:R(X) is a dirty read here.

2. The following schedule results in a read-write conflict:
   T2:R(X), T2:R(Y), T1:R(X), T1:R(Y), T1:W(X) ...
   Now, T2 will get an unrepeatable read on X.

3. The following schedule results in a write-write conflict:
   T2:R(X), T2:R(Y), T1:R(X), T1:R(Y), T1:W(X), T2:W(X) ...
   Now, T2 has overwritten uncommitted data.

4. Strict 2PL resolves these conflicts as follows:

(a) In S2PL, T1 could not get a shared lock on X because T2 would be holding an exclusive lock on X. Thus, T1 would have to wait until T2 was finished.

(b) Here T1 could not get an exclusive lock on X because T2 would already be holding a shared or exclusive lock on X.

(c) Same as above.