351/751 Database Systems, 2022, Semester 1

Lab 06

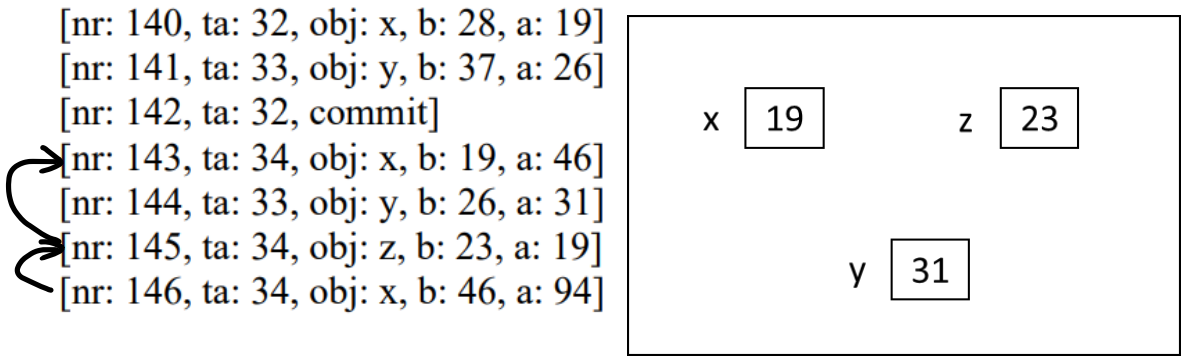
due Sat 7 May 2022 11:59pm

Name: Yixuan Li

API:yil845

1. The following diagram shows a log of a database at a certain point in time, and the current state of the database buffer to the right. You are supposed to abort and roll back transaction 34. What is the state of the database buffer after the rollback?

Result of rollback of TA34:



b) In the diagram above, no object written by the aborted transaction has been edited by any other transaction afterwards. Was this just a lucky coincidence? Could it be that in the moment of the abort, we find that an object has again been written to by a different transaction, making rollback of transaction 34 impossible?

That is impossible. According to ACID properties of isolation(I), data operation in one transaction is locked from database operations of all other transactions. All processes are updated in log buffer. In order to get rollback effective and durable, we need commit at the end of rollback transaction and write them in stable log and permanent database.

2. For the following schedule, list all the transactions in the schedule and give their local schedules.

s : r1[x], r2[y], r1[x], w1[x], r2[x], w2[y], a2, c1

simple scheduler:

TA1: r1[x], r1[x],w1[x], c1

TA2: r2[y], r2[x] ----------→, w2[y], a2(delay due to data-disjoint violation on x)

Common scheduler:

TA1: r1[x], r1[x],w1[x] c1

TA2: r2[y], r2[x], w2[y],a2

3. Can the simple scheduler produce the following schedules? Can the common scheduler produce these schedules? Give reasons for your answer.

a) s: r1[z], r2[x], r2[y], w2[y], a2, r1[y], w1[y], c1

Simple scheduler and common scheduler can both produce this schedule. There is no delay in simple and common scheduler due to mutual exclusive properties, TA2 executed first before TA1 accessing object y.

Simple scheduler:

TA1: r1[z] r1[y] w1[y],c1

TA2: r2[x], r2[y], w2[y],a2

Common scheduler:

TA1: r1[z], r1[y] w1[y],c1

TA2: r2[x], r2[y], w2[y],a2

b) s : r1[y], r2[x], r2[y], w2[z], c2, r1[y], w1[y], c1

Only common scheduler can produce this schedule. There are shared reads on Y by TA1, TA2, therefore not data-disjoint, but write-disjoint w1[y],w2[z], hence no delay for common scheduler . If changing to simple scheduler, it violates data-disjoint when TA1 and TA2 both accessing object Y.

Simple scheduler:

TA1: r1[y] r1[y]--------(deadlock due to violation of data-disjoint on y)

TA2: r2[x],r2[y]----------

Common scheduler: (serialized schedule without delay)

TA1:r1[y] r1[y],w1[y],c1

TA2: r2[x],r2[y],w2[z],c2

c) s: r1[y], r2[y], r1[z], w2[y], r2[x], a2, w1[y], c1

Only simple scheduler can produce this schedule. TA1 has lock on y and allows it to proceed before TA2 accessing. A common scheduler would run into a deadlock, because both TA1 and TA2 try to write in object y, it violates write-disjoint definition of common scheduler.

simple scheduler:

TA1:r1[y], r1[z] , w1[y],c1

TA2: r2[y]----------→, w2[y],r2[x],a2 (delayed due to prioritized write lock on y)

Common scheduler:

TA1: r1[y], r1[z],w1[y]---------- (deadlock)

TA2:r2[y], w2[y]----------(deadlock)

4. Given the following timing of operations arriving at the DB (the local schedules indicate the earliest possible point in time that an operation will be issued. Give the scheduling diagram that would be produced by the simple scheduler. Give the scheduling diagram that would be produced by the common scheduler. If an earlier operation is delayed too much, other operations will be issued later, accordingly. If a deadlock occurs, state this and list the partial schedule of executed operations):

a) r1[x],r2[y],r1[y],w2[z],w1[x],r2[x],c1,c2

Common scheduler: TA1 is delayed due to violation of write-disjoint while TA2 reads object x.

TA1: r1[x], r1[y], w1[x]--------→, c1

TA2: r2[y], w2[z], r2[x],c2

Simple scheduler: running into a deadlock because x is blocked in TA1 for further write operation and y is blocked in TA2 for further write operation.

TA1: r1[x], r1[y]----------(deadlock)

TA2: r2[y], w2[z], r2[x]--------

b) r1[x],r1[y],r2[y],w1[y],w2[z],c1,w2[y],c2

Common scheduler: running into a deadlock,because TA1 trying to read y, causing w1[y] to be not write-disjoint anymore, furthermore w2[y] causing TA2 to be not write-disjoint anymore.

TA1: r1[x], r1[y], w1[y]----------

TA2: r2[y], w2[z],w2[y]---------

Simple scheduler: (has delay since TA2 tries to read y after TA1 which causes data-disjoint violation)

TA1: r1[x], r1[y], w1[y], c1

TA2: r2[y]----------→, w2[z], w2[y], c2