HW 4

Name: Jiacheng Zhao

Resources. (All people, books, articles, web pages, etc. that have been consulted when producing your answers to this homework)

Textbook and instructor's slides

On my honor, as an Aggie, I have neither given nor received any unauthorized aid on any portion of the academic work included in this assignment. Furthermore, I have disclosed all resources (people, books, web sites, etc.) that have been used to prepare this homework. This work is my own and is written in my own words.

Signature: JIACHENG ZHAO

Problem 1. Exercise 16.4

Solution.

The balance of Z should be 20 in the end.

Transactions of T and U is not a serially equivalent example. The conflicting operations in T and U are NOT executed in the same order for all the data they both access. If U is executed first then execute T, the update of all U's operations will be lost, since we can not deposit money to an account that does not exist; if T is executed first, then the final deposit should be 30. In that case T and U are not serially equivalent.

10

Problem 2. Exercise 16.9

Solution.

(a) serially equivalent but not with two-phase locking

(b) serially equivalent and with two-phase locking

(c) serially equivalent and with two-phase locking

(d) serially equivalent but not with two-phase locking

10

Problem 3. Exercise 16.10

Solution.

T	U
read(i)	
	write(i, 55)
	write(j, 66)
read(j)	

A read only transaction can have inconsistent retrievals. Take the scenario above as an example. When read(i) release the read-only lock, U calls write-only lock and release it only after changing the number of j. After that T is able to read the value of j. Since the pairs of conflict operation in T and U are different so these transactions have inconsistent retrievals. They are NOT serially equivalent.

10

Problem 4. Exercise 16.16

Solution.

Scenario i)

T	U
X:=read(i);	
write(j,44);	
	write(i,55);
	write(j,66);

i) In this scenario, because it's backward validation we only check rule 2. Since U has not committed when T perform read(i), rule 2 is satisfied.

Scenario ii)

T	U
	write(i,55);
	write(j,66);
X:= read(i)	
Abort	

ii) In this scenario, U commits first. And since it's backward validation as well ,we only need check rule 2. Rule 2 forbids T form reading object written by U. Obviously, this rule is violated because T commits later and performs read(i), which was written by U before. In this case T should abort after performing read(i). The backward validation fails.

10

Scenario iii)

T	U
x: = read (i);	
write(j,44);	
	write(i,55);
	write (j,66);

iii) In this scenario we only need to check rule 1 since it's forward validation. Rule 1 forbids U from reading objects written by T. Since U does NOT perform any reading action, this rule is automatically satisfied.

Scenario iv)

T	U
x:= read(i)	Abort
write(j.44)	

iv) In this scenario, this case satisfy Rule 1 which forbids U form reading objects written by T. However, T commits later and performs read(i), which was written by U before. This action violates the Rule 2, so U has to abort. The forward validation fails.

Problem 5. Exercise 16.19

Solution.

(a) Initially:
$a_i = 10$, write timestamp = max read timestamp = t_0
$a_j = 20$, write timestamp = max read timestamp = t_0
First, T: read(i); T timestamp = t_1;

According to the read rule, t_1 > write timestamp on committed version, which is t_0 and $D_{selected}$ is committed. In this case, allow T to read x = 10 and set max read timestamp(a_i) as t_1.

Next, U: write(i, 55); U timestamp = t_2;

According to write rule, t_2 >= max read timestamp, which is t_1, and t_2 >write timestamp committed version(t_0). In this case, allow U to write a_i's value as 55 and set write timestamp as t_2.

Next, T: write(j, 44); T timestamp = t_1;

According to write rule, t_1 >= max read timestamp, which is t_0, and t_1 >write timestamp committed version(t_0). In this case, allow T to write a_j's value as 44 and set write timestamp as t_1.

Finally, U: write(j, 66); U timestamp = t_2;

According to write rule, t_2 >= max read timestamp, which is t_0, and t_2 >write timestamp committed version(t_0). In this case, allow U to write a_j's value as 66 and set write timestamp as t_2.

After U commits:

a_i: committed version: value = 55; write timestamp = t_2; max read timestamp = t_1

a_j: committed version: value = 66; write timestamp = t_2; max read timestamp = t_0

(b)Initially as (a);

First, T: read(i); T timestamp = t_1;

According to the read rule, t_1 > write timestamp on committed version, which is t_0 and $D_{selected}$ is committed. In this case, allow T to read x = 10 and set max read timestamp(a_i) as t_1.

Next, T: write(j, 44); T timestamp = t_1;

According to write rule, t_1 >= max read timestamp, which is t_0, and t_1 >write timestamp committed version(t_0). In this case, allow T to write a_j's value as 44 and set write timestamp as t_1.

Next, U: write(i, 55); U timestamp = t_2;

According to write rule, t_2 >= max read timestamp, which is t_1, and t_2 >write timestamp committed version(t_0). In this case, allow U to write a_i's value as 55 and set write timestamp as t_2.

Finally, U: write(j, 66); U timestamp = t_2;

According to write rule, t_2 >= max read timestamp, which is t_0, and t_2 >write timestamp committed version(t_0). In this case, allow U to write a_j's value as 66 and set write timestamp as t_2.

After U commits:

a_i: committed version: value = 55; write timestamp = t_2; max read timestamp = t_1

a_j: committed version: value = 66; write timestamp = t_2; max read timestamp = t_0

(c) Initially as (a)

First, U: write(i, 55); U timestamp = t_1;

According to write rule, t_1 >= max read timestamp, which is t_0, and t_1 >write timestamp committed version(t_0). In this case, allow U to write a_i's value as 55 and set write timestamp as t_1.

Next, U: write(j, 66); U timestamp = t_1;

According to write rule, t_1 >= max read timestamp, which is t_0, and t_1 >write timestamp committed version(t_0). In this case, allow U to write a_j's value as 66 and set write timestamp as t_1.

Next, T: read(i); T timestamp = t_2;

According to the read rule, $t_2 >$ write timestamp on committed version, which is t_0, but $D_{selected} = t_1$ is NOT committed yet. In this case, WAIT U to commit or abort.

U commits:

a_i: committed version: value = 55; write timestamp = t_1; max read timestamp = t_0

a_j: committed version: value = 66; write timestamp = t_1; max read timestamp = t_0

Finally, T: write(j, 44); T timestamp = t_2;

According to write rule, t_2 >= max read timestamp, which is t_0, and t_2 >write timestamp committed version(t_1). In this case, allow T to write a_j's value as 44 and set write timestamp as t_2.

After T commits:

a_i: committed version: value = 55; write timestamp = t_1; max read timestamp = t_2

a_j: committed version: value = 44; write timestamp = t_2; max read timestamp = t_0

(d) Initially as (a)

First, U: write(i, 55); U timestamp = t_1;

According to write rule, t_1 >= max read timestamp, which is t_0, and t_1 >write timestamp committed version(t_0). In this case, allow U to write a_i's value as 55 and set write timestamp as t_1.

Next, T: read(i); T timestamp = t_2;

According to the read rule, $t_2 >$ write timestamp on committed version, which is t_0, but $D_{selected} = t_1$ is NOT committed yet. In this case, WAIT U to commit or abort.

Next, U: write(j, 66); U timestamp = t_1;

According to write rule, t_1 >= max read timestamp, which is t_0, and t_1 >write timestamp committed version(t_0). In this case, allow U to write a_j's value as 66 and set write timestamp as t_1.

U commits:

a_i: committed version: value = 55; write timestamp = t_1; max read timestamp = t_0

a_j: committed version: value = 66; write timestamp = t_1; max read timestamp = t_0

Finally, T: write(j, 44); T timestamp = t_2;

According to write rule, t_2 >= max read timestamp, which is t_0, and t_2 >write timestamp committed version(t_1). In this case, allow T to write a_j's value as 44 and set write timestamp as t_2.

After T commits:

a_i: committed version: value = 55; write timestamp = t_1; max read timestamp = t_2

a_j: committed version: value = 44; write timestamp = t_2; max read timestamp = t_0

10