Hands-on 2

Write Ahead Log System Report

Sequence 1

After executing the commands in sequence 1, the output is as follows:

```
Opening new log
> Beginning id: 1
> Initializing account studentA to 1000
> Committing id: 1
> Ending id: 1
> Beginning id: 2
> Initializing account studentB to 2000
> Beginning id: 3
> Initializing account studentC to 3000
> Crediting account studentC by 100
> Debiting account studentA by 100
> Committing id: 3
>
_______________________
On-disk DB contents:
Account: studentA Value: 1000
_______________________


_______________________
LOG contents:
type: START action_id: 1
type: UPDATE action_id: 1 variable: studentA redo: "1000" undo: NULL
type: OUTCOME action_id: 1 status: COMMITTED
type: END action_id: 1
type: START action_id: 2
type: UPDATE action_id: 2 variable: studentB redo: "2000" undo: NULL
```

```
type: START action_id: 3
type: UPDATE action_id: 3 variable: studentC redo: "3000" undo: NULL
type: UPDATE action_id: 3 variable: studentC redo: "3100" undo: "3000"
type: UPDATE action_id: 3 variable: studentA redo: "900" undo: "1000"
type: OUTCOME action_id: 3 status: COMMITTED
```

```
> crashing ...
```

Question 1

Wal-sys displays the current state of the database contents after you type show_state. Why doesn't the database show studentB?

Answer 1

The answer is the command of creating account "studentA" is in the package between begin 1 and end 1. So after the end 1 line, the account 1 was written to the DB. However those creations after that wasn't ended even before the final crash. So the later command wasn't actually written into DB.

Question 2

When the database recovers, which accounts should be active, and what values should they contain?

Answer 2

Here's the result of recovery process:

```
Recovering the database ..
Starting rollback ...
  The log was rolled back 11 lines
Rollback done
Winners: id: 3  Losers: id: 2  Done: id: 1
Starting forward scan ...
  REDOING: type: UPDATE action_id: 3 variable: studentC redo: "3000" undo: NULL
  REDOING: type: UPDATE action_id: 3 variable: studentC redo: "3100" undo: "300
  REDOING: type: UPDATE action_id: 3 variable: studentA redo: "900" undo: "1000"
  Logging END records for winners
Forward scan done
```

```
Recovery done
```

After that, the DB would contain such stuff here:

```
studentC 3100
studentA 900
```

Question 3

Can you explain why the ”DB” file does not contain a record for studentC and contains the pre-debit balance for studentA?

Answer 3

Because the studentB creation operation was declared in begin 2 phase, however it wasn’t committed nor ended before the final crash. So it won’t be in the LOG nor the DB.

Sequence 2

Question 4

What do you expect the state of ”DB” to be after walsys recovers? Why?

Answer 4

After the crash, walsys should be able to recover the operations that we have ended or committed, in this case, the committed 1 and 3. However the studentB creation wasn’t committed at all, so there’s no way it could be recovered.

Question 5

Run walsys again to recover the database. Examine the ”DB” file. Does the state of the database match your expectations? Why or why not?

Answer 5

It matches my expectation. Because that’s how I expect it works.

Question 6

During recovery, walsys reports the action_ids of those recoverable actions that are ”Losers”, ”Winners”, and ”Done”. What is the meaning of these categories?

Answer 6

Winners ID refers to those who aren’t ended (written into DB) but successfully recovered according to the LOG files. Losers ID refers to those who never commits, which makes it unable to recover. Done ID refers to those who are already written to the DB, so there’s no need to do any recovering stuffs.

Checkpoints

After running the reset

```
Opening new log
> Beginning id: 1
> Initializing account studentA to 1000
> Committing id: 1
> Ending id: 1
> Beginning id: 2
> Initializing account studentB to 2000
> Doing checkpoint
> Beginning id: 3
> Initializing account studentC to 3000
> Crediting account studentC by 100
> Debiting account studentB by 100
> Committing id: 3
>
_______________________
On-disk DB contents:
Account: studentA Value: 1000
_______________________


_______________________
LOG contents:
type: START action_id: 1
type: UPDATE action_id: 1 variable: studentA redo: ”1000” undo: NULL
type: OUTCOME action_id: 1 status: COMMITTED
```

```
type: END action_id: 1
type: START action_id: 2
type: UPDATE action_id: 2 variable: studentB redo: "2000" undo: NULL
type: CHECKPOINT PENDING: id: 2 COMMITTED:  DONE: id: 1
type: START action_id: 3
type: UPDATE action_id: 3 variable: studentC redo: "3000" undo: NULL
type: UPDATE action_id: 3 variable: studentC redo: "3100" undo: "3000"
type: UPDATE action_id: 2 variable: studentB redo: "1900" undo: "2000"
type: OUTCOME action_id: 3 status: COMMITTED
```

```
> crashing ...
```

The major difference between Sequence 2 and Sequence 1 is one more "checkpoint" keyword. And the last debit_account was operated on transaction 2. It is specially pointed out that it's not a typo. Yes because there are other typos everywhere else in the handson.

Question 7

Why are the results of recoverable action 2's create_account 2 studentB 2000 command not installed in "DB" by the checkpoint command on the following line? Examine the "LOG" output file. In particular, inspect the CHECKPOINT entry. Also, count the number of entries in the "LOG" file. Run walsys again to recover the database.

Answer 7

Because the transaction 2 wasn't committed to the LOG at all, so the afterwards operation wasn't committed into the LOG. In the checkpoint line, we can see that transaction 1 was "done", however the transaction 2 was still pending anyway.

Question 8

How many lines were rolled back? What is the advantage of using checkpoints?

Note down the action_ids of "Winners", "Losers", and "Done". Use the show_state command to look at the recovered database and verify that the database recovered correctly. Crash the system, and then run walsys again to recover the database a second time.

Answer 8

Only back 6 lines were rolled back.

```
REDOING: type: UPDATE action_id: 3 variable: studentC redo: ”3000” undo: NULL
REDOING: type: UPDATE action_id: 3 variable: studentC redo: ”3100” undo: ”300
```

Compared to the former recovery state, there are only two redoing declarations.

Question 9

Does the second run of the recovery procedure (for sequence 2) restore ”DB” to the same state as the first run? What is this property called?

Answer 9

Not really. See the redoing log in Answer 8, It didn’t do the studentA(B) redo, because studentB wasn’t created at all. Two redoing declaration is executed as follows:

```
REDOING: type: UPDATE action_id: 3 variable: studentC redo: ”3000” undo: NULL
REDOING: type: UPDATE action_id: 3 variable: studentC redo: ”3100” undo: ”300
```

Question 10

Compare the action_ids of ”Winners”, ”Losers”, and ”Done” from the second recovery with those from the first. The lists are different. How does the recovery procedure guarantee the property from Question 9 even though the recovery procedure can change? (Hint: Examine the ”LOG” file).

Answer 10

In my tests, the WLD list is still Winners: id: 3 Losers: id: 2 Done: id: 1. The lists are NOT different.