Hands-on 1: Write Ahead Log System (1)

Complete the following hands-on assignment. Do the activities described, and submit your solutions for following questions. Please upload your answers to Canvas in a pdf file. You are free to use either Chinese or English. Due time is 2021-11-2 23:59.

Intro to wal-sys

This and the next hands-on assignment will give you some experience using a **Write Ahead Log (WAL)** system. This system corresponds to the WAL scheme described in Section 9.3 of the textbook. You should carefully read that section before attempting this assignment. You can do this hands-on on any computer that has a Python language interpreter, but we will be able to answer your questions more easily if you run this on a Linux. Please download a Python script named wal-sys. Before trying to run it, change its permissions to make it executable, for example by typing:

```
$ chmod +x wal-sys.py
```

The wal-sys script can be run as follows:

```
$ ./wal-sys.py [-reset]
```

Alternatively, you can run the script as:

```
$ python ./wal-sys.py [-reset]
```

Wal-sys is a simple WAL system that models a bank's central database, implementing redo logging for error-recovery. Wal-sys creates and uses two files, named LOG and DB, in the current working directory. The "LOG" file contains the log entries, and the "DB" file contains all of the installed changes to the database.

After you start wal-sys, you can enter commands to manage recoverable actions and accounts. There are also commands to simulate a system crash and to print the contents of the "LOG" and "DB" files. All the commands to wal-sys are case sensitive. Since wal-sys uses the standard input stream, you can use the system in batch mode. To do this, place your commands in a file ("cmd.in" for example) and redirect the file to wal-sys's standard input:

```
$ ./wal-sys.py -reset < cmd.in
```

When using batch mode, make sure that each command is followed by a newline character (including the last one). The -reset option tells wal-sys to discard the contents of any previous "DB" and "LOG" files so that it can start with a clean initial state.

After a system crash, you can restart wal-sys using the following command without any parameters:

```
$ ./wal-sys.py
```

It will perform a log-based recovery of the "DB" file using the "LOG" file it finds in the current working directory.

Commands interpreted by wal-sys

The following commands are used for managing recoverable actions and accounts:

- `begin action_id`: Begin a recoverable action denoted by `action_id`. The action_id is a positive integer that uniquely identifies a given recoverable action.
- `create_account action_id account_name starting_balance`: Create a new account with the given `account_name` and `starting_balance`. The first argument specifies that this operation is part of a recoverable action `action_id`. The `account_name` can be any character string with no white spaces.
- `credit_account action_id account_name credit_amount`: Add `credit_amount` to `account_name`'s balance. This command logs the credit and holds it in a buffer until an `end` command is executed for recoverable action `action_id`.
- `debit_account action_id account_name debit_amount`: Reduce `account_name`'s balance by `debit_amount`. Like credit, this command logs the debit and holds it in a buffer until an `end` command is executed for recoverable action `action_id`.
- `commit action_id`: Commit the recoverable action `action_id`. This command logs a `commit` record.
- `checkpoint`: This command logs a `checkpoint` record.
- `end action_id`: End recoverable action `action_id`. This command installs the results of recoverable action `action_id` to the "DB". It also logs an `end` record.

The following commands help us understand the dynamics of the WAL system:

- `show_state`: Print out the current state of the database. This command displays the contents of the "DB" and "LOG" files.
- `crash`: Crash the system. In this hands-on, we are only concerned about crash recovery, so this is the only command we will use to exit the program.

Using wal-sys

In this part, we will initialize the database using some commands and then make the system crash. Start wal-sys with a reset:

```
$ ./wal-sys.py -reset
```

and run the following commands:

```
begin 1
create_account 1 studentA 1000
commit 1
end 1
begin 2
create_account 2 studentB 2000
credit_account 2 studentA 100
begin 3
create_account 3 studentC 3000
commit 2
debit_account 3 studentC 100
show_state
crash
```

Wal-sys should print out the contents of the "DB" and "LOG" files, and then exit.

Use a text editor to examine the "DB" and "LOG" files and answer the following questions (do not run wal-sys again until you have answered these questions):

Question 1: Wal-sys displays the current state of the database contents after you type `show_state`. What do you observe in on-disk DB contents? Why doesn't the database show studentC?

Question 2: Which accounts should exist, and what values should they contain when the database recovers?

Question 3: Can you explain why the "DB" file does not contain a record for studentB and contains the balance for studentA before `credit_account` is issued?

Recovering the database

When you run wal-sys without the `-reset` option, it recovers the database "DB" using the "LOG" file. To recover the database and then look at the results, type:

```
$ ./wal-sys.py
> show_state
> crash
```

Question 4: What do you expect the state of "DB" to be after wal-sys recovers? Why?

Question 5: If you issue another wal-sys command to recover the database again, what would the "DB" file contain after the second recovery? Why?

Question 6: During recovery, wal-sys reports the `action_ids` of those recoverable actions that are "Losers", "Winners", and "Done". What is the difference between these categories?