Rollback simulation using Python

Description of features

Logical transactions that take place in a digital banking environment such as payments or transfers are usually multi-stepped in nature. Take for example, online payments, despite appearing to be only a single action performed by the end user, many additional database transactions must take place. For instance, a typical series of event may look something like the following:

- Check for authentication (not simulated)
- Verify that the purchaser has enough funds.
- Withdraw the amount in the transaction.
- Verify that the seller's account is valid and accepts the currency used in payments (not simulated)
- Deposit the amount into the seller's account.
- Mark transaction as complete

Each and every one of the above steps must have a failsafe, meaning that if any of the above steps were to fail, the buyer is protected from losing money, and the seller will not be credited money that failed to withdraw. For a such a system to take place, rollbacks and logging systems are integral to modern OLTP databases.

In this assignment, we've simulated the basic rollback and logging systems using Python, with csv files as data source.

Application Requirements

We started the assignment by figuring out how to read the csv files and overwrite these files with new/updated data. We tested the reading of the files by printing out the original data to the console. Testing the overwriting files included creating a new record with hard-coded values to the account.csv file and updating the same file with the new data as a result.

Then we needed to read in the csv files and store them in the program's memory. We used list for this purpose and created couple of enums to store constants.

For the rollback system to work, we needed a table that contained information about the transactions that took place.

To satisfy logging requirements, we devised two functions – withdrawal and deposit to simulate the steps needed to transfer funds from one account to another, while taking into consideration basic ways that such transaction can fail. These exceptions are used to trigger the rollback system. Each of these methods are to be written into the logging system.

We also have a commit function that writes changed values into the csv files, as well as a rollback function that can restore changes to a previous state. Each of these functions also updates the log table with corresponding statues.

Data Structures

We've decided to use list to store information read in from csv files. This allows for easy access to data by referencing its index and permits quick and easy iterations when needed.

Listed down below is the data type for one transaction log:

[id, table name, primary key, column number that was changed, old value, new value, status
Example:
827b85ec-14f4-477d-8c35-148a1f36febb,ACCOUNT_BALANCE,312345c,1,583624,483624,rolled back

Code design

Enum

```
class AccountType(Enum):
```

- AccountType is used to store constants for CHECKING and SAVINGS account.

```
class Table(Enum):
```

- This enum is used to identify the types of data files the program is dealing with.

Functions

Data access

To read data into our application, we've included two methods:

```
def read_csv_file():
```

- This function will read in the supplied csv files containing customer, account, and balance data into lists.

```
def write_csv_file(table: Table):
```

- This function will write a corresponding table enum to the associated csv file.

```
def getAccountNumber(fName, lName):
```

Transaction processing

We implemented the following methods to aid in processing the required transaction.

```
def startTransaction():
```

- generates a unique ID and is called before every transaction. This ID is used to keep track of the changes taking place in the application.

```
def withdrawal(transID, accountID, amount, accountType: AccountType):
```

- First reads in the csv files and populating existing records database records.
- A hardcoded failure condition (amount < 0) is implemented.
- Accountnumber is then determined from the database records. If the account is not either checking nor savings, another exception will be raised.
- The function will then attempt to subtract the amount to be withdrawn from the current account balance. If this results in a negative number, another exception will be raised.

- Finally, if the transaction completes, 1 will be returned indicating success.

```
def deposit(transID, accountID, amount, accountType: AccountType):
```

- Similar to withdrawal, if the amount to be deposited is less than 0, an exception will be raised.
- The function then finds the account to be deposited to, while checking for account type.
- Finally, the amount is added to the existing balance and 1 is returned indicating success.

Logging Sub Systems

```
def updateTable(transID, table: Table,  id, column, value):
```

- This functions essentially organizes the data that's passed in and writes them in an organized manner to the log table csv using transID, while appending the "incomplete" status.

```
def commit(transactionID):
```

- This method is only called when a transaction is complete.
- It updates the database csv files so any changes are saved to the hard drive.
- The method then sets the previous "incomplete" tag to "complete" by the transID.

```
def rollback(transactionID):
```

- This function is programmed to be called whenever an exception is raised in the main method.
- It will first read the data from csv and resets the values for accounts, account_balance, customers to its original value.
- Then it will read in the rows from the log table.
- Similar to the commit function, it will mark the transaction as "rolled back" using a transID look up in the log file and print out what's been reversed.

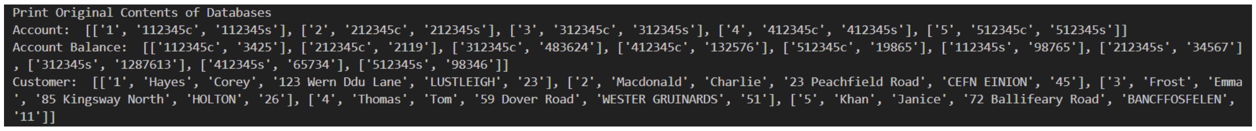
Main

```
def main():
```

- First tables are loaded into memory.
- When a transaction starts, an ID is generated.
- Each operation applied to a table is linked to this ID.
- Each time an operation is applied, a log is saved to the transaction log.
- Once commit is called, all the operations from the transaction ID are saved to disk and the records in the log are updated to complete.
- If an error occurs, the rollback function is called.
- This throws out any changes made that were not saved to disk. Transactions are only saved to disk when commit() is called.

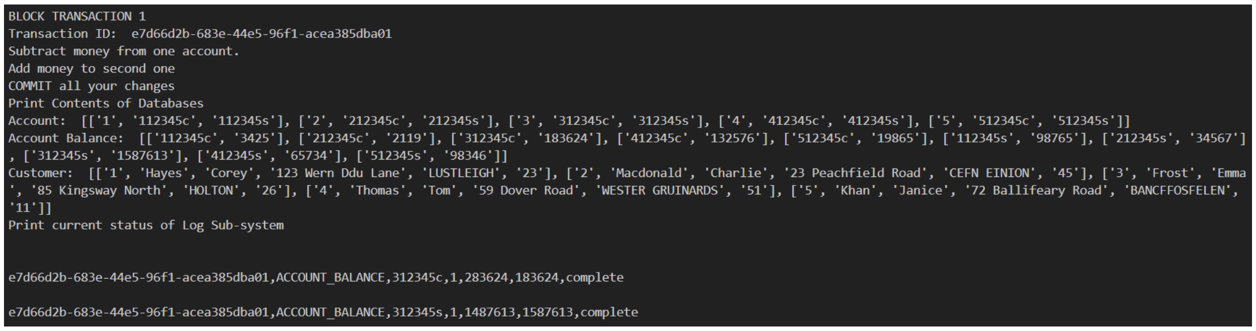
Sample Run

1. Before a transaction:
    a. Variables such as accounts, account_balance, customers, and transaction is filled using the read_csv_file() function.
    b. We query the customer of interest's account number from the customer list
    c. We then display the data in the csv file before any transaction blocks.



```
Print Original Contents of Databases
Account:  [['1', '112345c', '112345s'], ['2', '212345c', '212345s'], ['3', '312345c', '312345s'], ['4', '412345c', '412345s'], ['5', '512345c', '512345s']]
Account Balance:  [['112345c', '3425'], ['212345c', '2119'], ['312345c', '483624'], ['412345c', '132576'], ['512345c', '19865'], ['112345s', '98765'], ['212345s', '34567']
, ['312345s', '1287613'], ['412345s', '65734'], ['512345s', '98346']]
Customer:  [['1', 'Hayes', 'Corey', '123 Wern Ddu Lane', 'LUSTLEIGH', '23'], ['2', 'Macdonald', 'Charlie', '23 Peachfield Road', 'CEFN EINION', '45'], ['3', 'Frost', 'Emma
', '85 Kingsway North', 'HOLTON', '26'], ['4', 'Thomas', 'Tom', '59 Dover Road', 'WESTER GRUINARDS', '51'], ['5', 'Khan', 'Janice', '72 Ballifeary Road', 'BANCFFOSFELEN',
'11']]
```

2. Transaction 1 – success
    a. The program will produce a unique transaction ID
    b. Followed by calling the withdrawl method from the checking account, and depositing the same amount to the savings account.
    c. The program will then call the commit method to update the database files, and write the newly created log record to the log table.
    d. Finally, the database tables and log tables are displayed. Note

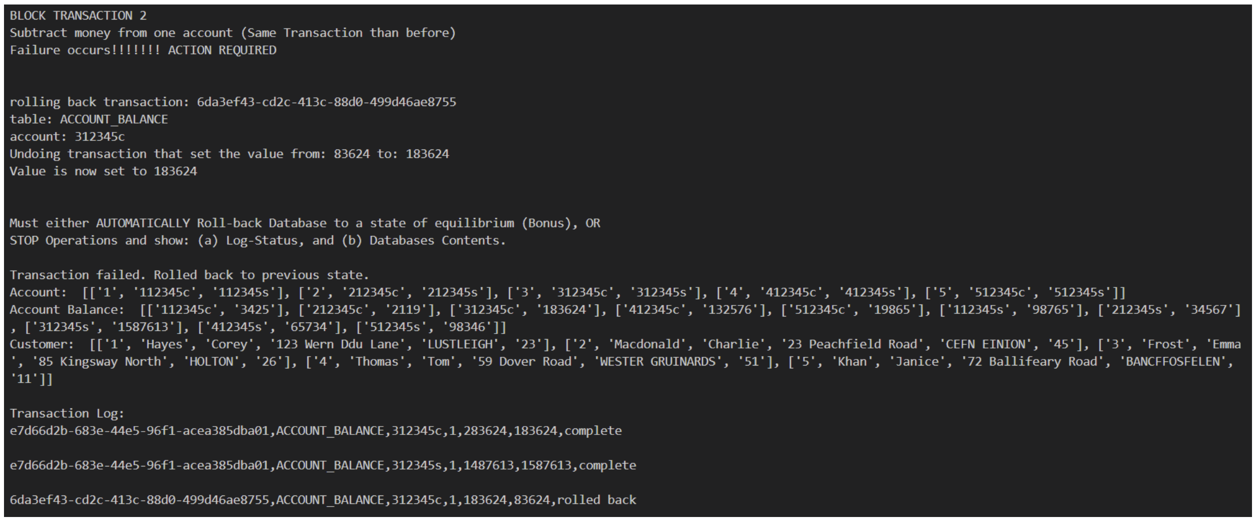


```
BLOCK TRANSACTION 1
Transaction ID:  e7d66d2b-683e-44e5-96f1-acea385dba01
Subtract money from one account.
Add money to second one
COMMIT all your changes
Print Contents of Databases
Account:  [['1', '112345c', '112345s'], ['2', '212345c', '212345s'], ['3', '312345c', '312345s'], ['4', '412345c', '412345s'], ['5', '512345c', '512345s']]
Account Balance:  [['112345c', '3425'], ['212345c', '2119'], ['312345c', '183624'], ['412345c', '132576'], ['512345c', '19865'], ['112345s', '98765'], ['212345s', '34567']
, ['312345s', '1587613'], ['412345s', '65734'], ['512345s', '98346']]
Customer:  [['1', 'Hayes', 'Corey', '123 Wern Ddu Lane', 'LUSTLEIGH', '23'], ['2', 'Macdonald', 'Charlie', '23 Peachfield Road', 'CEFN EINION', '45'], ['3', 'Frost', 'Emma
', '85 Kingsway North', 'HOLTON', '26'], ['4', 'Thomas', 'Tom', '59 Dover Road', 'WESTER GRUINARDS', '51'], ['5', 'Khan', 'Janice', '72 Ballifeary Road', 'BANCFFOSFELEN',
'11']]
Print current status of Log Sub-system


e7d66d2b-683e-44e5-96f1-acea385dba01,ACCOUNT_BALANCE,312345c,1,283624,183624,complete

e7d66d2b-683e-44e5-96f1-acea385dba01,ACCOUNT_BALANCE,312345s,1,1487613,1587613,complete
```

3. Transaction 2 – failure
    a. A new transaction is initiated with a new ID.
    b. This time, we purposely programmed the deposit amount to be negative, therefore triggering a preprogrammed failure point, which will allow us to automatically trigger the rollback() function on exception detection.
    c. As an error is detected during the deposit transaction, it was not recorded, and the previous transaction (withdrawl) is reverted automatically to the equilibrium state.



```
BLOCK TRANSACTION 2
Subtract money from one account (Same Transaction than before)
Failure occurs!!!!!!! ACTION REQUIRED


rolling back transaction: 6da3ef43-cd2c-413c-88d0-499d46ae8755
table: ACCOUNT_BALANCE
account: 312345c
Undoing transaction that set the value from: 83624 to: 183624
Value is now set to 183624


Must either AUTOMATICALLY Roll-back Database to a state of equilibrium (Bonus), OR
STOP Operations and show: (a) Log-Status, and (b) Databases Contents.

Transaction failed. Rolled back to previous state.
Account:  [['1', '112345c', '112345s'], ['2', '212345c', '212345s'], ['3', '312345c', '312345s'], ['4', '412345c', '412345s'], ['5', '512345c', '512345s']]
Account Balance:  [['112345c', '3425'], ['212345c', '2119'], ['312345c', '183624'], ['412345c', '132576'], ['512345c', '19865'], ['112345s', '98765'], ['212345s', '34567']
, ['312345s', '1587613'], ['412345s', '65734'], ['512345s', '98346']]
Customer:  [['1', 'Hayes', 'Corey', '123 Wern Ddu Lane', 'LUSTLEIGH', '23'], ['2', 'Macdonald', 'Charlie', '23 Peachfield Road', 'CEFN EINION', '45'], ['3', 'Frost', 'Emma
', '85 Kingsway North', 'HOLTON', '26'], ['4', 'Thomas', 'Tom', '59 Dover Road', 'WESTER GRUINARDS', '51'], ['5', 'Khan', 'Janice', '72 Ballifeary Road', 'BANCFFOSFELEN',
'11']]

Transaction Log:
e7d66d2b-683e-44e5-96f1-acea385dba01,ACCOUNT_BALANCE,312345c,1,283624,183624,complete

e7d66d2b-683e-44e5-96f1-acea385dba01,ACCOUNT_BALANCE,312345s,1,1487613,1587613,complete

6da3ef43-cd2c-413c-88d0-499d46ae8755,ACCOUNT_BALANCE,312345c,1,183624,83624,rolled back
```