ASSIGNMENT #1 WINTER 2024

COMP 4522 ADVANCED DATABASES

GROUP: ANDREW W., RUTU K., BRANDON K.

DUE: FEBRUARY 15, 2024 @ 5 pm

DOCUMENTATION

1. **Why did we choose to go with the data structure of a ‘Python List’ instead of other options?**
   1. Simple implementation to create, modify, and locate values in a list. As a team, we all felt that an array was a stronger skillset of ours than the other options.
   2. Lists allowed for variable types to be inserted / instantiated into it with no additional libraries. This allowed for a better mimic of what a database table structure would function like. A standard list data type also helped us avoid implementing something from another library such as a NumPy array for different stored data types.
   3. Moving to grab data from lists can easily be implemented and run via indexing, which makes it easier to retrieve the data and specific records too. This is especially true in this case because all the data falls in the same layout via the first header index pulled from the csv file, hence we know for example, index 2 will always be ‘Last\_Name’ in the ‘data\_base’ list variable.
   4. Printing out a list to the screen and/or csv file for testing and debugging is also a plus point.
   5. We did consider linked lists as a potential data structure for this assignment, but we quickly decided against it because of the complex implementation and maintenance of it. From traversing the data to linking the data together with heads and tails, it seemed overly complex, especially for this assignment.

Overall, a list seemed like a straightforward data structure to implement and maintain hence that is what we went with.

1. **What are the attributes that we decided to go with and why?**
   1. **'**transId' 🡪 Identification key of the specific transaction.
      1. Needed for: Tracking and referring to each transaction. Currently implemented by a custom function that generates an “ID” out of variable ASCII characters and ASCII Numbers of a chosen string size appended to a CRUD character. Other concept options we had debated on could be a string made up of a few attributes in the ‘DB\_Log’ appended to the same CRUD char.
   2. 'table' 🡪 Table the transaction is attempting to change.
      1. Needed for: Identifying the target table for modification and maintenance.
      2. Candidate composite key for primary key in ‘transId.’ For example, Employees would be ‘E.’
   3. 'attribute' 🡪 The requested attribute to be changed for the transaction.
      1. Needed for: Specifying the attribute that will be modified in the target table.
   4. 'targetTable' 🡪 The data table being changed.
      1. Needed for: Indicating the specific table that will be modified.
   5. 'empId' 🡪 The Id of the employee data being changed.
      1. Needed for: Connecting the transaction with the employee.
      2. Candidate composite key for primary key in ‘transId.’ For example, if ‘empId’ is 2, then append 2 to the ‘transId.’
   6. 'valueBefore' 🡪 The current attribute value in the table of selected transaction.
      1. Needed for: Recording the original attribute value before modification for rollback purposes.
   7. 'valueAfter' 🡪 The new attribute value pending change of the selected transaction.
      1. Needed for: Storing the updated attribute value after a successful change.
   8. 'success' 🡪 The state of the transaction. 'S' = Committed, 'P' = Not-Executed, 'F' = Rolled-Back.
      1. Needed for: Tracking the outcome of each transaction (committed, not-executed, or rolled-back).
   9. 'ownerTransId' 🡪 The id of the owner who ran the transaction.
      1. Needed for: Identifying the initiator of the transaction for accountability.
      2. Candidate composite key for primary key in ‘transId.’ For example, if ‘ownerTransId’ is 13, then append 13 to the ‘transId.’

**NOTE**: Concept ‘transId’ key from composites could be ‘U-2E13.’

1. **Are there any special installations needed?**
   1. No, there are no third party libraries in use for this program so no special installations are needed.