Devin Roberts  
Tejbir Singh  
Xinjie Hao

**CS 542, Project 4: Undo/Redo Logging**

**Introduction**

The focus of this project was twofold: First, to update the city and country relations by applying a 2% increase in city and country population for each tuple in the corresponding relations. Each tuple update is captured in a log file to mimic the transactional behavior of typical DBMS found in real-world. As in real DBMS systems, logs can be useful to recover data that may have been lost due to database failure. Second, we used only the log files to update older copies of the city and country relations.

**Design**

The Relation class is a modified version of the Relation class used in project 3. The class now keeps track of a log, which is a list of LogElement objects. LogElement is a simple object which is used to keep track of changes made to the Relation. When a “COMMIT” message is received, a .log file is generated and the Relation writes all of its logged changes to disk.

LogElement

* Transaction – transaction which made the modification
* Element – element which was modified
* newValue – value which the element was changed to
* oldValue – value which the element was previously set to

The UpdateOperator class is used to iterate through each of the Relation objects, increment their population by 2% and write the new value back. The constructor requires the Relation to work with as well as the index of the population column in the tuple, enabling it to be used generically for any Relation. For each modified value, UpdateOperator adds a log message to the Relation indicating the required information. When all modifications are complete, a “COMMIT” message is outputted and the logs are generated.

From there, the main function calls ApplyLogsToStore. This takes a string containing the location of the log, the Relation to apply the logged changes to, and the index of the population in the tuple. The class then iterates through the log changing each element in the Relation to the specified value.

LoggingUtil is a small utility class that was created to help manipulate the tuples in UpdateOperator and ApplyLogsToStore. getTupleValues() converts the byte array implementation native to the Relation object into a String array, allowing it to be manipulated as text. Unsplit() does the opposite: it takes the String array and formats it as a byte array ready to be re-added to the Relation.

The Main class is the runnable portion of the project which calls upon the others to perform the larger task. The main function begins by initializing the Relation objects, which ultimately reads the .csv files and adds them to the values store. It then calls upon the UpdateOperator to perform its task and create the .log files, followed by ApplyLogsToStore to re-apply the changes to an untouched copy of the original Relation.

**Assumptions**

* The population of each city and country change is always positive.
* The population of each city and country is increased by 2%. That is, the population of the population of the country relation will always increase by a full 2% rather than calculating the change based on each of its cities.

**Testing**

Because we were given a specific task to complete, testing was relatively straightforward. After both UpdateOperator and ApplyLogsToStore run, the main function verifies that each tuple in the Relation modified by UpdateOperator is equal to the matching tuple in the Relation which was modified by ApplyLogsToStore.