CS 635 Advanced OO Design and Programming
Spring Semester, 2016
Assignment 3

San Diego State University -- This page last updated 3/15/16

Assignment 3 Version 1.1
Reactive NoSQL
Due April 5

NoSQL databases are used for unstructured data. You are going to create a mini-in-memory database. NoSQL databases tend to act like hashtables. Each piece of data has a key. When you add data to the database you provide the key and the data. To retrieve the data you use the key. To keep things on the simple side we will use strings as keys. Our data will be Strings, Numbers, “Arrays” and “Objects”. “Arrays” are ordered list of data (Strings, Numbers, “Arrays” or “Objects”). A single “Array” can hold multiple type of data at the same time. An “Object” is a hashtable with a String as a key and any data as the value. You will find the JSON library useful here. The operations on your NoSQL database will be to add new data, get the data at a given key, replace data at a given key, and remove a key and its data. Modifying data at a given key is a bit more complex, so we will not support that operation directly. Modifying data can be done with a retrieval and an add. Your first task in to implement the mini-in-memory database.

The problem with in-memory data is that it is not persistent. So when the program stops running the data is lost. Use the command and memento patterns to fix this. When you perform any operation on your in-memory database that modifies it state use a command object. You can then save all the commands in a file. You can then reconstruct the current state of the database by replaying all the stored commands. If you run the database long enough the number of stored commands will become very large. The way to handle that is to create a memento of the database, save the memento in a file, and remove all the commands. Then to recreate the database one reads the memento from the file and then applies the stored commands.

Once we are using commands to modify the database we can create transactions. Say for example our database contains bank account information. Assume that we have four companies involved a business deal. Company A first pays Company B x dollars, then has to pay Company C y dollars and then Company D has to pay Company B z dollars. All three payments must be made for the deal to be valid. If one of the payments fails all the others must be undone. Implement transactions.

Since the database is in memory it does not have to be separate from an application, like SQLite. So we could get some data and say plotting the data. If the data in the database changes we want our data to reflect the current state of the database. For example:

```
database.put(“bar”, 10);
Cursor data = databse.get(“bar”);
data.value()  // returns 10
database.put(“bar”, 20);
data.value()  // returns 20
```

Of course the data at “bar” could be more complex. Changing data.value(30) needs also change the data in the database. If we are plotting the data when the value of the data changes we want the plot to change also. So we need to be able to add an observer to the cursor in the above example.

```
database.put(“bar”, 10);
Cursor data = databse.get(“bar”);
data.addObserver(foo);
database.put(“bar”, 20);
//So here foo gets notified of the change.
```

Grading

Item	Points
Working Code	10
Unit Tests	10
Patterns	40
Quality of Code	10

Turning in your Assignment

Turn in the hard cody in class.

Version History

1.1 March 15. Added due date, grading and added observer to the Cursor.