Boya Zhou, Yuting Liang
Project 1 for Database Management System class
Oct 1st, 2015

To GA: if there is any vague part below, please send e-mail to bzhou@WPI.EDU and I will explain to you. Thank you so much!

Introduction

A values store (a simplified version of a key-value store). The value store has just three interfaces:

```
1  void Put(int key, byte[] data); stores data under the given key,
2  byte[] Get(int key); retrieves the data and
3  void Remove(int key); deletes the key.
```

The value byte array can be arbitrarily large, up to 1 MB.

Your solution should be based on a single 5 MB file (4 MB for actual data, 1 MB for any metadata that you may need to store — you'll only need a fraction of the ! MB). Name the file `cs542.db`.
Your solution should be implemented as a library that can be linked into multiple processes. Show the results of running the following validations:

Concurrency Validation. What happens when one caller in one thread is doing a `Get()` while another caller in another thread does a Put() and replaces the data. Or when one caller does a `Remove()` and another caller does a `Get()` with the same key a millisecond later.

Durability Validation. What happens if, after a reboot of the machine after the data has been `Put()`, a caller does a `Get()`?

Fragmentation. `Put()` 4 values, byte arrays of 1 MB each, with keys A, B, C and D. Remove key B. `Put()` ½ MB in size for key E. Validate that a `Put()` 1 MB in size for key F fails. Remove C and now validate that a `Put()` 1 MB in size for key G succeeds. Remove E and try `Put()` 1 MB in size for key H. With a naive implementation, it will fail even though there is room in `store.db`. An extra bonus point if you can modify your code such that `Put("H", …)` succeeds.

Create a shell program `show <filename>` that will show the layout of data in `cs542.db`. Run the program to do the validations listed above and include a record of your interactions to prove that the validations produced the expected results.

Assumption

1. The isolation level we choose is Read committed. So we make void Put(int key, byte[] data) and void Remove(int key) synchronized but byte[] Get(int key) not.

Design Decisions

The data structure we make for the tiny database is consist of three parts:

1. byte[4][] arrayFile: this is use to put the content of database, there are 4 spaces in this array, each of them is 1MB.

2. HashMap<Integer,Integer> keyByteArray: this is use to link the key and the index of arrayFile, since the key is 'A','B','C' and so on ,we can not use the index in the 3 interface.

3. above.boolean[] occupied:this is use to record whether the space in arrayFile is occupied. This tiny database can not contain more than 4 MB.

And there are several method in this data structure:

1. protected boolean isOccupied(int index) :this is use to return whether the space is been occupied

2. public int keyToIndex(int key):link the key and index, also it will throw a exception when there is no such a key in database.

3. public static void saveToDisk(ArrayDB arrayDB) :write the obj in CS542.db(this process is like to save the content in memory in disk)

4. public static ArrayDB readFromDisk() :read from CS542.db(this process is like to load content from disk when there is need to recover data)

5. public byte[] get(int key):get the content of arrayFile using key in keyByteArray. (This will use method public int keyToIndex(int key)).

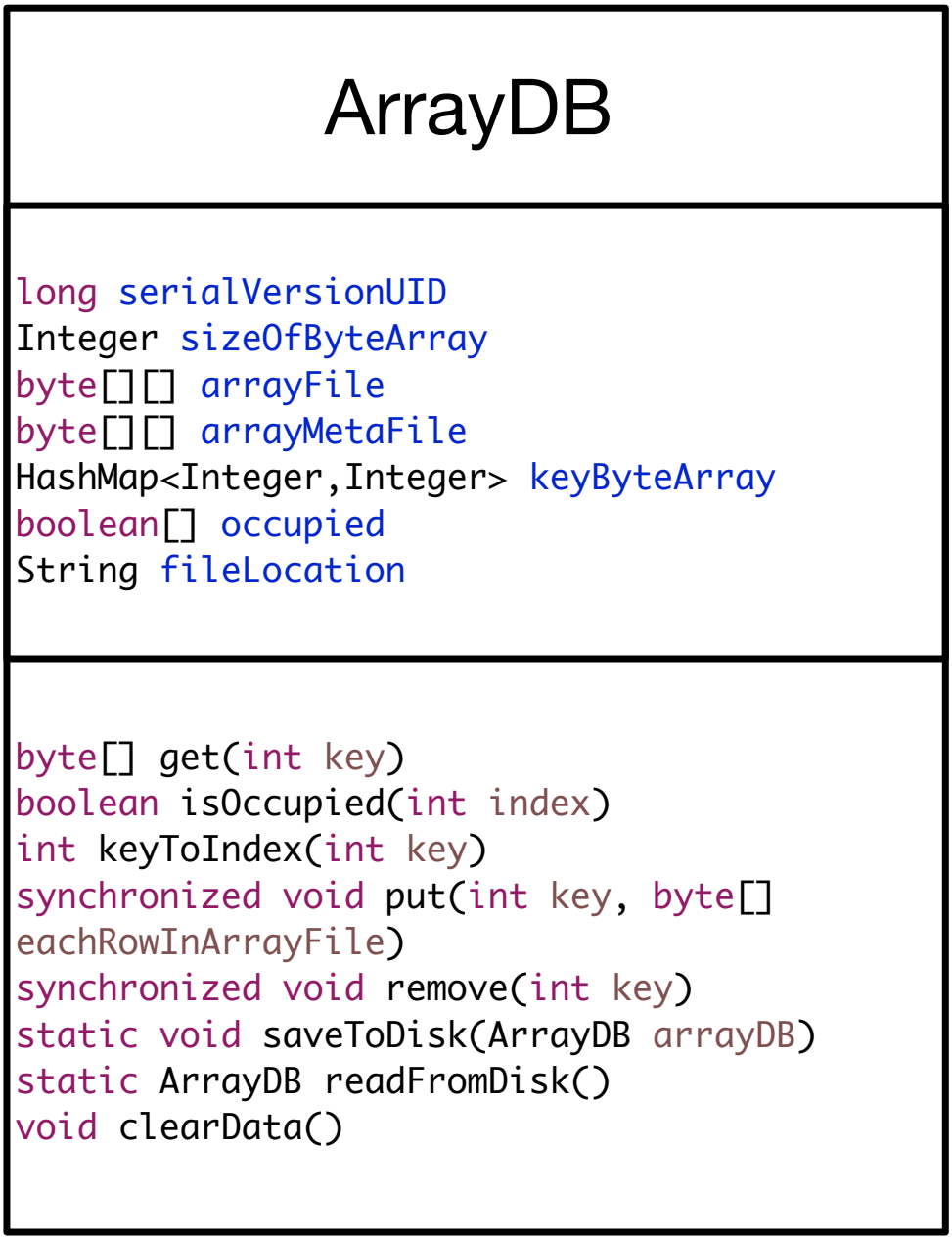
6. public synchronized void put(int key, byte[] eachRowInArrayFile): insert or update one row in database. If the key is already in database, insert; else, update.When finishing insert or update, it will save the content in memory into

disk(using method public static void saveToDisk(ArrayDB arrayDB)). When there is no space in database, it will tell us.

7. public synchronized void remove(int key): remove the row in database and save the change to disk. If there is no such key, it will tell us.

8. public void clearData(): to initialize the new, empty database.

UML Diagram



Result

when run the main.java, it will show the result of the test as follows:

```
Successfully putting A
-------------------------
-----    getAndPut   -----
Successfully putting A
false
_____________
```

1. First we use get(), to get the content of key ’A’, then we use put(), and then we get the content of key ’A’ again. This results shows that when compared the two results, the value of key’A’ has changed, it means read committed.

```
-------------------------
Successfully putting A
-----------------------------
-----    removeAndGet   -----
Successfully removing A
cannot find key for A
-----------------------------
_____________
```

2. First we remove the content of key ‘A’, and then we use get() to gain the content of key ‘A’, but the results shows that we can not find the content of key’A’;

```
---------------------------
-----    putAndGet()   -----
Successfully putting A
true
---------------------------
_____________
```

3. First we put the content of key ’A’ in database(mean time we use a variable to store this content), and we use cleardata() to initialize the database in memory, then we use put to get the content of key ‘A’ from disk. Finally, we get the result of comparing the two results, which shows true.

```
-----------------------------------------
-----    removeAndPutFragmentation   -----
Successfully putting A
Successfully putting B
Successfully putting C
```

```
Successfully putting D
Successfully removing B
Successfully putting E
There is no space for F
Successfully removing C
Successfully putting H
-----    the key with the index of array   -----
{68=3, 69=1, 65=0, 72=2}
-----------------------------------------
-----------------------------------------
--Completed-----
```

4. follow the step given in problem 2, we have get the expected result, {68=3, 69=1, 65=0, 72=2}means the key and index in HashMap keyByteArray, 72 is 'H'(ascii).

5. As for the layout of the CS542.db. we make a jar file called shellProject1.jar and you can use java -jar shellProject1.jar in bash to show it.

   and there is a quick show as follow:

```
BoyadeMacBook-Pro:Dropbox boyazhou$ java -jar shellProject1.jar
Key     Index   dbFileContent   dbFileOccuipied
D       3       [B@33afbbe3     true
E       1       [B@56584e97     true
H       2       [B@3f8fc7ca     true
A       0       [B@7885bf5f     true
BoyadeMacBook-Pro:Dropbox boyazhou$
```