Project 2 (Index Library) Proposal
May Zhai
Bill Ge

a) API Documentation

Commands

1. public boolean createTree (String primaryKey, String filePath, Comparator compare)
   Following method creates a B tree structure from the given CSV file located at the path specified by filePath with the given primaryKey and uses the comparator specified
   Returns T/F for whether or not the tree was created. Could return False if the data contained a duplicate primary key.
2. public boolean insertData (Object data)
   Method allows for the insertion of new data into the B tree based upon the primaryKey specified during the createTree method
   Returns T/F for whether or not the data was inserted. Data could possibly not be inserted because of null fields, invalid fields, duplicate primary keys, etc.
3. public Object search (String primaryKey)
   Method allows for the user to search through the B tree for the primaryKey specified
   Returns Object containing the record corresponding to the primaryKey
4. public boolean remove (String primaryKey)
   Method removes the Node that represents the primaryKey from the tree
   Returns T/F for whether the object was removed from the B tree
5. public boolean deleteTree()
   Method deletes the tree structure built
   Returns T/F for whether the tree was successfully deleted
6. public ArrayList rangeQuery(Object firstKey, Object secondKey)
   Method returns data with a primary key greater than that of firstKey and less than secondKey
   Returns a set of data stored in an ArrayList

b) Design Document

Everything will be programmed in Java.

Index Tree

- B-tree, where each node is the size of a block on disk (4 kb)
  - The implementation of the tree will allow the user to specify the order of the tree and to specify the comparator used for the primary keys
- Leaves of the B-tree point to a place in the datastore

Datastore

Our datastore will consist of fixed width records stored in a binary file

- Each record is a total of 120 characters (fixed width)
  - Rank: 3 characters
  - Year: 4 characters
  - Song Title: 25 characters
  - Artist Name: 25 characters

- Features: 5 characters
- Size: 8 characters
- Time:4 characters
- Group: 5 characters
- Debut Year: 4 characters
- State: 17 characters
- Country: 20 characters

Caching/Buffering

- Whenever the user looks up a current record or adds a record, we will use a hashtable to keep track of the key and data, so we don't have to get the data again if the request is repeated

Concurrency

- When there are multiple threads inserting and deleting, the requests will be processed sequentially (the other thread must wait until the first thread is finished)

c) Timeline

Priorities are numbered where 1 = most urgent and ascending numbers represent decreasing priority

Proposal: Thursday, October 13, 2011
Goals:

- 1. implemented tree structure and be able to index keys (mzhai)
- 2. implemented the feature to create datastore from a CSV file (bge)

Key indexing or Datastore: Thursday, October 20, 2011
Goals:

- 1. caching implementation (mzhai)
- 2. buffering implementation (bge)
- 3. concurrency control (mzhai)

Record Indexing/Caching/Buffering: Thursday, October 27, 2011
Goals:

- 1. API implementation (bge)
- 2. sample application with sample data (mzhai)

- 3. documentation about how to use test program (bge)
- 4. update document describing any changes made since the original proposal (mzhai)
- 5. updated copy of the API documentation (bge)

The plan is to completely the above goals by Monday, October 31st. This way, there will be some time left over in order to fix any last minute mistakes.
Final Deadline: Thursday, November 3, 2011