RMIT University

COSC2406/2407 – Database Systems

Assignment #2

**MongoDB, Apache Derby, Java**

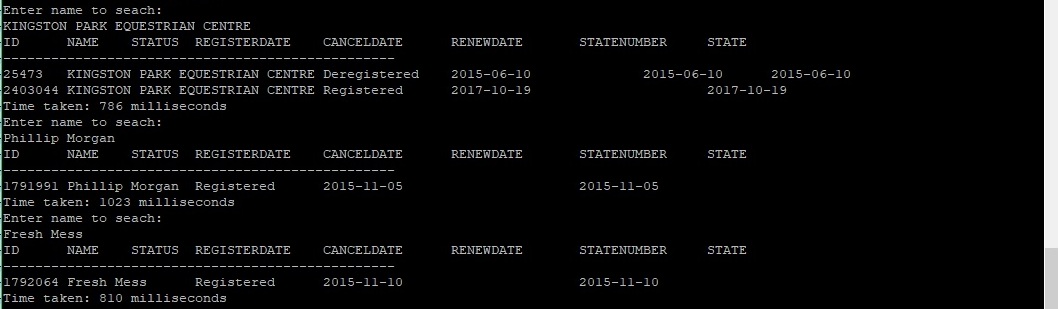
# Task 1: Derby

## Summary

I created a Java app that reads data from the database created in assignment 1. This app searches for exact name given by the user.

String statement = ("select \* from businessNames where name = '"+ seachName +"'");

After a few searched the results are the following:



Query 1: 786 milliseconds

Query 2: 1023 milliseconds

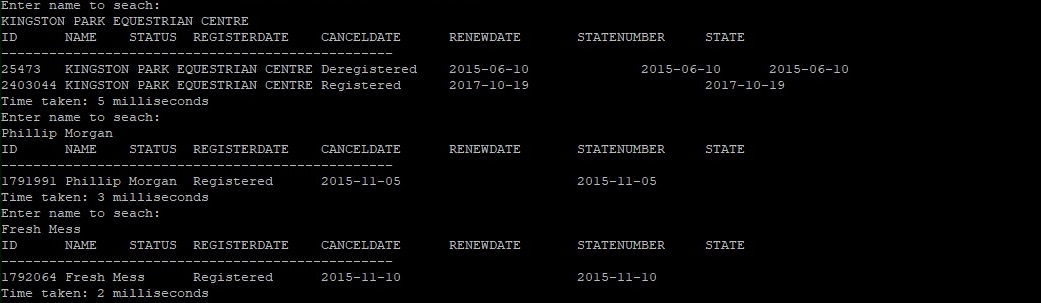
Query 3: 810 milliseconds

**Average time: 873 milliseconds**

After this, I created index on the name field in the database with:

CREATE INDEX index\_name ON businessNames (name);

Then, conducted same searches:



Query 1: 5 milliseconds

Query 2: 3 milliseconds

Query 3: 2 milliseconds

**Average time: 3.3 milliseconds**

## Result

Adding index to derby database greatly improved the performance of a search against the database, however only work with exact matches.

# Task 2: MongoDB

## Summary

Queries with the same business name were done on mongoDB. To display query statistics I used

.explain("executionStats") at the end of query.

Before adding an index the results are the following:



Query 1: 1380 milliseconds

Query 2: 1270 milliseconds

Query 3: 1290 milliseconds

**Average time: 1313 milliseconds**

Then I added index to be in ascending order with the command:

db.business.createIndex({Name:1});

And then ran the queries again:



Query 1: 5 milliseconds

Query 2: 0 milliseconds

Query 3: 0 milliseconds

**Average time: 1.6 milliseconds**

Impressively, a lot of queries would return in 0-1 milliseconds.

# Task 3: Implement Heap File in Java

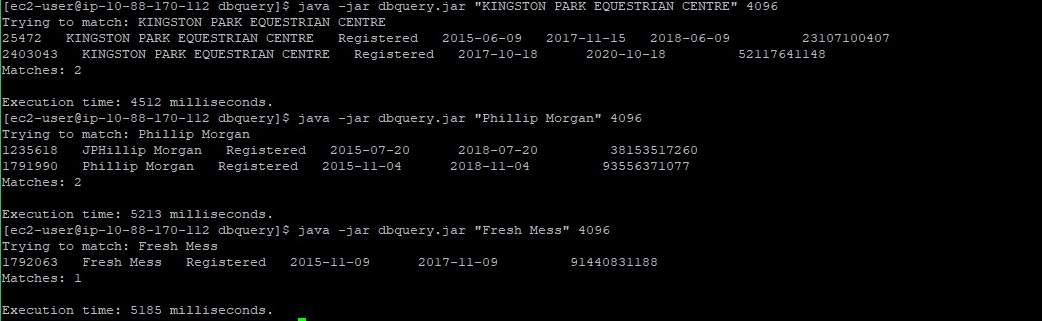
## Index file structure

Index file in just pairs of integers: hashID (hashCode \* 3800000) of the business name and position in the heap file. As we have about 2.5 million record and to achieve about 70% occupancy I used 3800000 as a size of the index file. I used RandomAccessFile class in java to write to index file in particular position which can be calculated from hashID.

Record structure in an index file if the following:

{[hashID][ position ]}{[ hashID][ position]}{[ hashID ][ position ]}{[hashID][ position]}…

I ran the same queries as on derby and mongo:



Query 1: 4512 milliseconds

Query 2: 5213 milliseconds

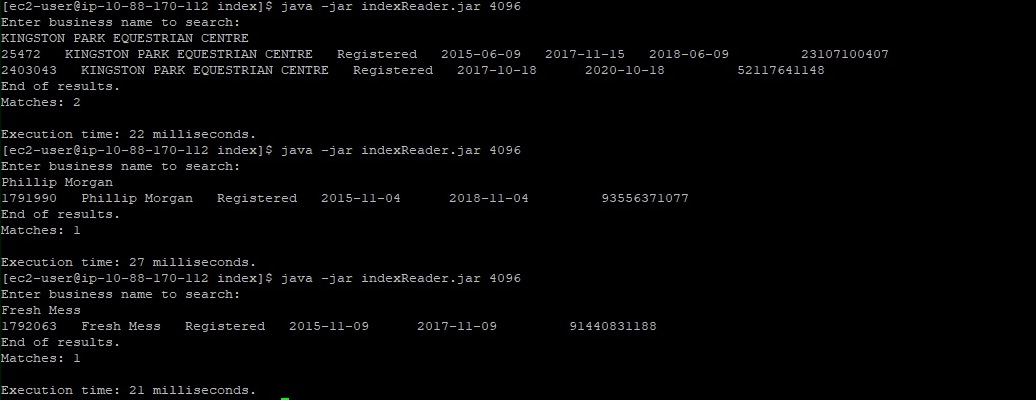
Query 3: 5185 milliseconds

**Average time 4870 milliseconds**

Creating index file:



Searched using index reader:



Query 1: 22 milliseconds

Query 2: 27 milliseconds

Query 3: 21 milliseconds

**Average time 23.3 milliseconds**