CSCD43 Assignment1 report:
Analysis/interpret:
Blue dot line is for 1st query in both IndexScanQueries.sql and ScanQuries.sql

Red dot line is for 2nd query in both IndexScanQueries.sql and ScanQuries.sql

Yellow dot line is for 3rd query in both IndexScanQueries.sql and ScanQuries.sql

Black dot line is for 4th query in both IndexScanQueries.sql and ScanQuries.sql

Green dot line is for 5th query in both IndexScanQueries.sql and ScanQuries.sql

Blue line is for 6th query in both IndexScanQueries.sql and ScanQuries.sql

Red line is for 7th query in both IndexScanQueries.sql and ScanQuries.sql

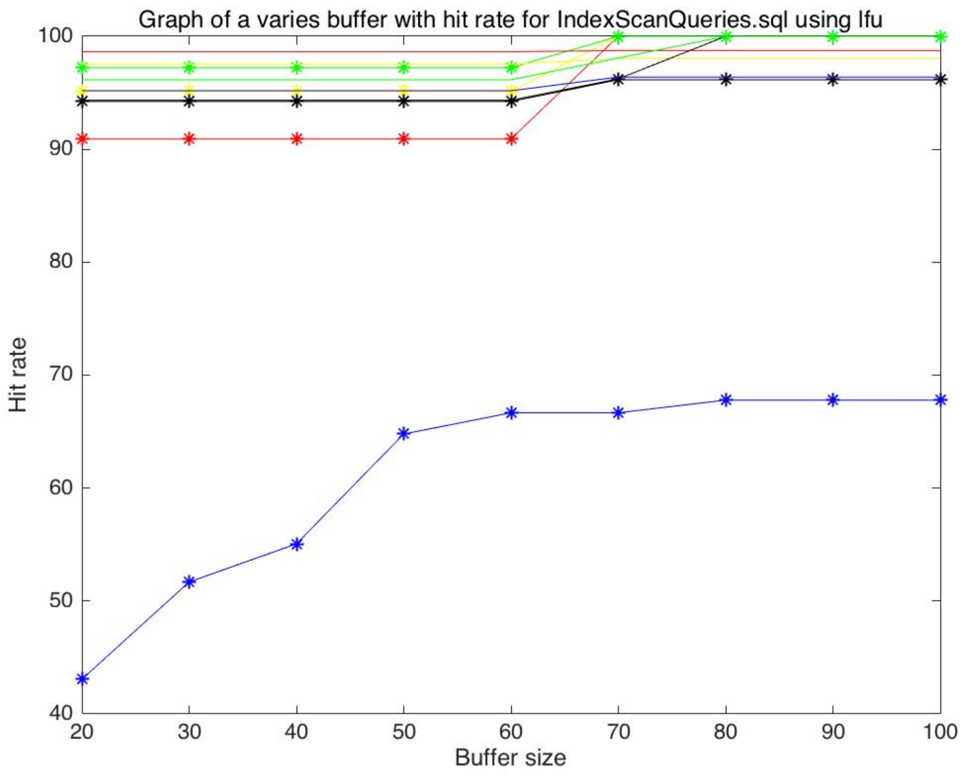
Yellow line is for 8th query in both IndexScanQueries.sql and ScanQuries.sql

Black line is for 9th query in both IndexScanQueries.sql and ScanQuries.sql

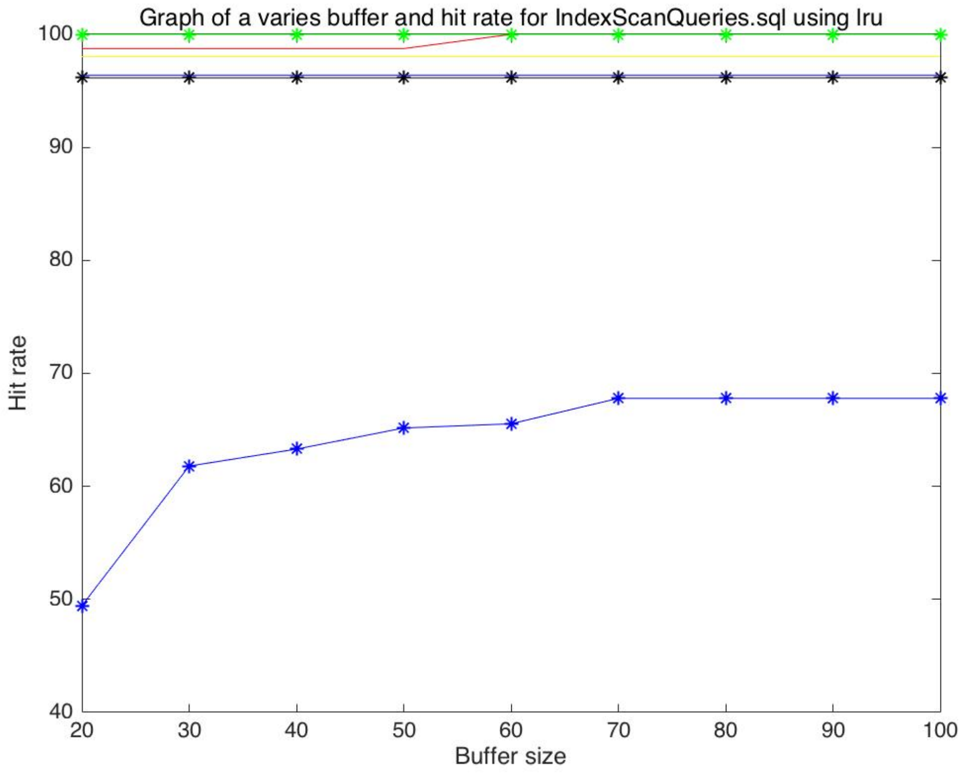
Green line is for 10th query in both IndexScanQueries.sql and ScanQuries.sql

Some graphs cannot show all the lines clearly because they are same, for example, if Green dot line and Black line are same, then the Black line would not show up, because Green dot line is drawn before Black line. Therefore, Black line is behind Green dot line. It actually depends on the above sequence of drawing the line. If 1st query give same line as 2nd query, then the 2nd query's line would not show up in graph.

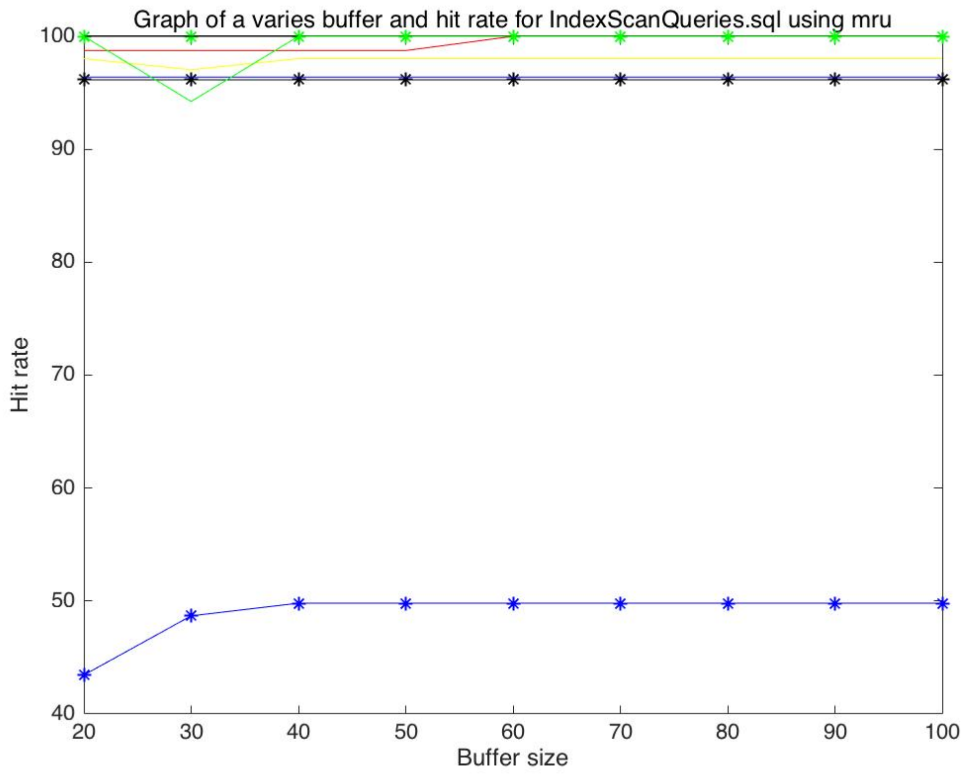
There are total 6 graphs in my reports, each graph represents 1 file using 1 specific policy. X axis is buffer size, and Y axis is hit rate.

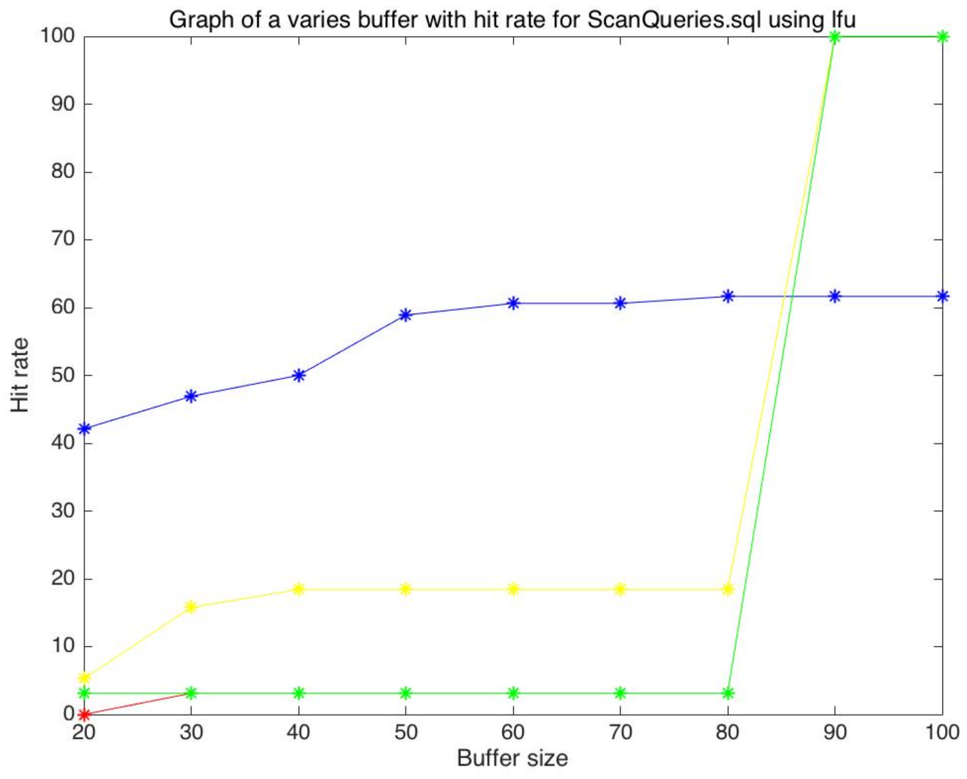


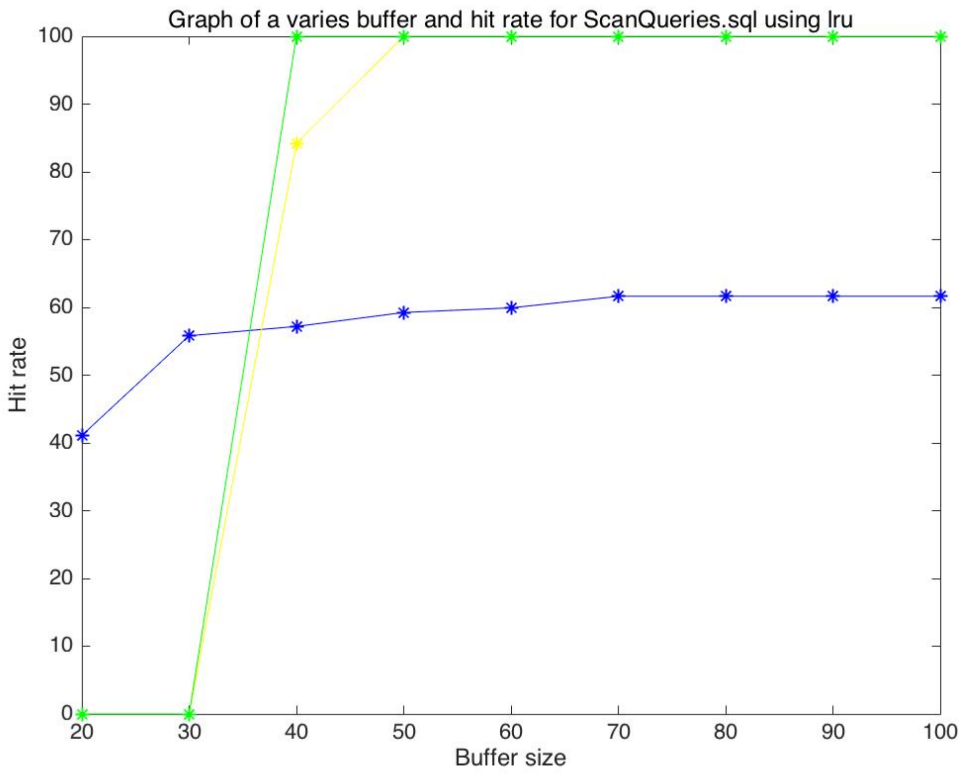
Graph of a varies buffer with hit rate for IndexScanQueries.sql using lfu
Hit rate
Buffer size

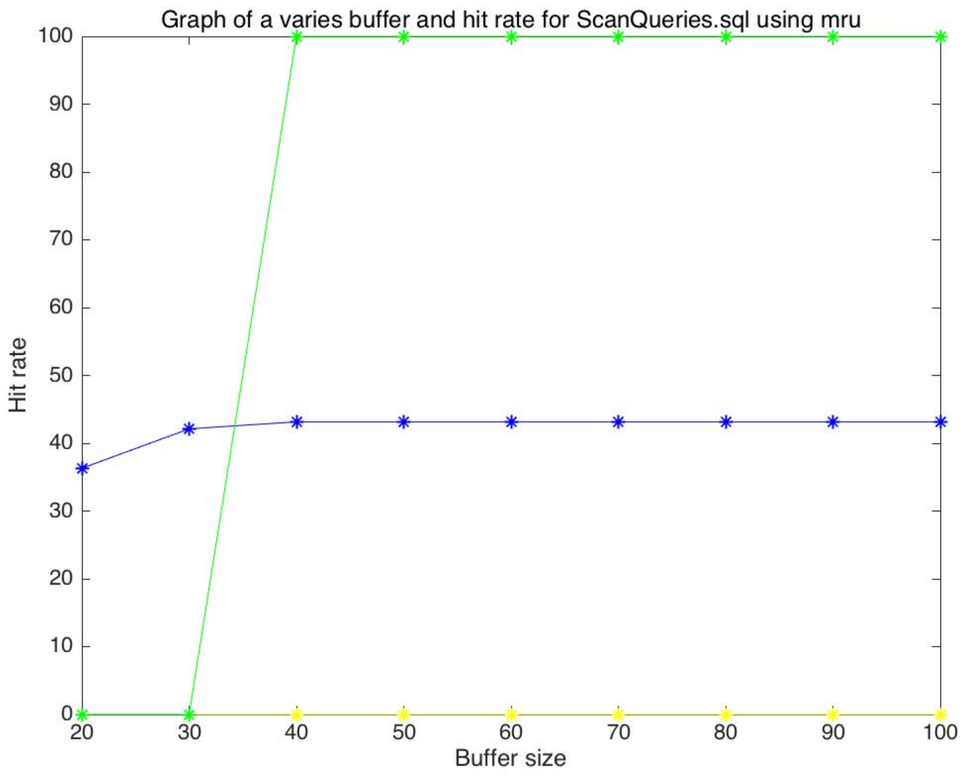


Graph of a varies buffer and hit rate for IndexScanQueries.sql using lru
Hit rate
Buffer size


Graph of a varies buffer and hit rate for IndexScanQueries.sql using mru
Hit rate
100
90
80
70
60
50
40
20
30
40
50
60
70
80
90
100
Buffer size


Graph of a varies buffer with hit rate for ScanQueries.sql using lfu
Hit rate
100
90
80
70
60
50
40
30
20
10
0
20
30
40
50
60
70
80
90
100
Buffer size


Graph of a varies buffer and hit rate for ScanQueries.sql using lru
Hit rate
Buffer size
0
10
20
30
40
50
60
70
80
90
100
20
30
40
50
60
70
80
90
100


Graph of a varies buffer and hit rate for ScanQueries.sql using mru
Hit rate
Buffer size
0
10
20
30
40
50
60
70
80
90
100
20
30
40
50
60
70
80
90
100

Interesting discoveries:

1. From all above graphs, the 1st query always generates the worst hit rate. The reason is that during the initialization of buffer pool, there is no data in it at the beginning, so after loading the amount of data, it will start hit the buffer. So with the increasing buffer size, it performs better than before, because after loading more data at first time, it will increase the rate of hitting buffer.
2. From first three graphs, the result is obviously almost same (but no exactly same) for all three policies with IndexScanQueries.sql. in my file, I start with choosing ID > 1 and ID < 10, it only counts 8 times, but due to the empty buffer at the beginning, the hit rate would be low. After the buffer pool will load a large amount of data at first time, the following queries give a very high performance after the first query, because the data is already contained in buffer pool.
   a. With the different queries except first one, they all show a much higher performance, this is due to the number 1 discoveries.
   b. . In additional, I believe that LFU has a small advantage than other two policies, because as we can see from 2nd and 3rd graphs, there are almost straight horizontal line after first query, but the 1st graph shows a little bit improvement after buffer size become 60 or bigger. The reason is that with the increasing buffer size, LFU policy wisely uses the buffer pool, only replace pages with least frequency. For example:
      Green dot line shows a huge improvement after buffer size is 60 or large, the green dot line actually finds the ID between 350 and 420, and the one before it actually finds the ID between 400 and 450, which we can use most of them for the green dot line without replacement.
      But the other two policies keep replacing most used or least used page, which has no benefits for following queries.
3. From last three graphs, the result is almost same for last two graphs which are MRU and LRU policies. After buffer size is 30, they have a huge improvement, because after buffer size is 30, the buffer pool is enough bigger to contain the data for following queries.
   a) It looks like LRU did not do well on ScanQueries.sql, as we can see that **Green dot line** always perform low hit rate until the buffer size is 80 or large, and the **Green dot line** is the 5th query, which is find C > 50 and B < 20 which is almost totally different from the one before it, which need to find B > 10 and A < 50, so the hit rate is really low, because after we replace the least frequent page, the most frequent page would stay, but that would not help us to save our performance. Moreover, the remained pages occupied our buffer pool, which might cause problem to get the correct page that we want.
   b) In this case, I would like to say LRU and MRU are generally more efficient, because the input queries are not really related to each other like IndexScanQueries.sql, then we might use LRU and MRU policies for our buffer pool.

Estimation:

Data contains 5000 lines, but the buffer cache hit rate is generally lower when we run first query, and increase after first query. For every queries in different files, the average hit rate is around 50% or 40% when we run fist query, and the number of reads is between 130 to 180, so the average is 160. Afterwards, the read dramatically decreases to a small number.
We know that read = total number of pages - pages not in buffer.
hit rate = number of hit pages/total pages in buffer.
We read 160 pages with only 50% of them is hit in buffer, and we only need read a small number of pages to get 100% of hit rate in buffer, which means we may need to read up to a number of pages from 180 to 250, the data file is containing 5000 tuples, I would consider each page to be a block to contain 5000 tuples, so the data file almost contains 180 to 250 blocks.