CS222P Fall 2018 Quiz 4

1. In project 3, an extra credit feature is to allow duplicate keys, and the number of duplicate keys may not totally fit into one page. In general, we have two approaches to handle this case, i.e.,(1) use key+RID as composite key, (2) for each key, store a list of RIDs. Explain how these two technique work, especially how you handle search/insert/delete.

   a. key+RID composite key method

   b. RID list method

2. Consider a relation R(a,b,c,d,e) containing 5,000,000 records, where each data page of the relation holds 10 records. R is organized as a sorted file with secondary indexes. Assume R.a is the primary key for R, and the values lying in range [0, 4,999,999], and that R is stored in R.a order. For each of the following four queries, state which of the following three approaches is most likely to be the cheapest:

   1. Scan the sorted file for R directly
   2. Use a (clustered) B+ tree index on attribute R.a
   3. Use a linear hashed index on attribute R.a

   a. Select * from R where a < 400,000;

   b. Select * from R where a = 400,000;

   c. Select * from R where a > 50,000 AND a < 50,100;

   d. Select * from R where a != 50,000;

3. Index only plan is often a very efficient way to execute query. Consider the following relation E(id, name, age, job). Suppose we have a secondary B+ tree index on age+name. Suppose E contains 100,000,000 records. Each data page contains 100 records, but 1000 index entries. Further assume age is uniformly distributed among [20, 59]. Now calculate the cost (in terms of page I/Os) for the following query using the methods below:

   SELECT name FROM E WHERE age = 25

For simplicity, you may ignore the cost of non-leaf pages when search the B+tree index.

   a. Scan E directly

   b. Index-only access of the secondary index on age+name

   c. Access the secondary index on age+name, sort RIDs, and retrieve names from E

1. a. key+RID composite key method

We treat key+RID as a composite key. For a search, we search the key part as the prefix of the composite key. That is, we find the leftmost matching key, and return RIDs of all matching keys. Insert/Delete is treated identically as before, since an insert/delete has to specify the whole composite key.

b. RID list method

We store a list of RIDs for each key together. When the list is too large to fit into a leaf page, we create an overflow page to store it. For search, we simply find that list of RIDs. For insert/delete, we first find the list of RID (maybe in a separate page), and update the list correspondingly.

2.
   a. 1. scan directly
   b. 3.linear hashed index
   c. 2.clustered B+Tree index
   d. 1.scan directly

3. We have 1M data pages.
   a. 1M data pages
   b. The query selects 0.025*100M = 2.5M records, which fit into 2.5M/1000 = 2.5K index pages. So only 2.5K index pages are needed
   c. Again, we need to access 2.5K index pages. Since we have to go to the data file, we need to access min(1M, 2.5M) data pages. So the total number of pages is 2.5K+1M