CS430 Study Guide Name _____

Module 8

1. What is the difference between a dense index and a sparse index?

   Dense Index: An index record appears for every search-key value in the file.
   Sparse Index: Contains an index record for only some search-ley values.

2. What are the advantages of an B+ tree for indexing? Disadvantages?

   Automatically reorganizes itself with small, local changes. A minor disadvantage is that there is some overhead costs of insertion and deletions.

3. What are the properties of a B+ tree?

   All leaves are at the same level.
   The root has at least two children.
   Each node except root can have a maximum of m children and at least m /2 children.
   Each node can contain a maximum of m - 1 keys and a minimum of m/2 - 1 keys.

4. What does a node in a B+ tree look like? What is a leaf node? A non-leaf node?

   A node is simply an index flanked by pointers to other indices that are either less than or greater than it.

   A leaf node is where the data is stored (or a pointer to the data) and also exists as a sequential linked list.

5. What is the cost advantage of a B+ tree over a binary tree?

   The path from the root to any leaf node is exactly the same

6. How do you insert into a B+ tree?

7. How do you delete from a B+ tree?

8. What problems does prefix compression solve?

   N number of first characters are unique enough to distinquish unique index values. The name can then be compressed to the first 4 characters (for example) and leaf nodes can share common prefixes.

9. What is a bucket with respect to data records? With hash algorithms, how do we obtain the bucket of a record?

A bucket is a table where the data (or pointers to the data) is stored and can be accessed by hashing the search index of what you want to access.

10. What are the traits of an ideal hash algorithm?

Produces uniform distribution and random distribution of the search index

11. What are some of the deficiencies of static hashing?

As the files/database grow / shrink there is considerable overhead resizing or refactoring the whole hash system to fit. Either you're wasting space (hash is too big) or you're too dense and your hash has too much bucket overflow.

12. How does dynamic hashing solve those deficiencies (i.e.extendible) ?

Periodically rehashes the entries of the database by creating a new hash table / hash function.

13. What is the basic premise of extendible hashing?

Tailored more for disk based hashing, where buckets are shared by multiple hash values. Doubles the numbers in the hash table without doubling the number of buckets.

Grows selective parts of the hash table based on their occupancy.

14. What is the global index? The local index?

A global index is a shared partition from multiple buckets, allowing you to (for example) only need to query a single partition when doing a lookup across multiple buckets / tables. A local index is a single partition for a single table and so doing a lookup across an entire database may require traversing many partitions.

15. When the directory is doubled, what happens to the pointers not pointing to the node being split?

16. What are the benefits of extendible hashing? Disadvantages?

Data retrieval is less expensive (in terms of computing).
No problem of Data-loss since the storage capacity increases dynamically.
With dynamic changes in hashing function, associated old values are rehashed w.r.t the new hash function.

The directory size may increase significantly if several records are hashed on the same directory while keeping the record distribution non-uniform.
Size of every bucket is fixed.
Memory is wasted in pointers when the global depth and local depth difference becomes drastic.
This method is complicated to code.