Physical Data Organization and Indexing

Concepts and Definitions that you need to understand:

1. What are the components of access time to a particular sector?
2. What is a heap file? What is a sorted file? What are the advantages and disadvantages of head files comparing to sorted files?
3. What is the access cost for insertion/deletion in heap files? in sorted files?
4. What are equality search and range search?
5. What is an index? Why are indices good?
6. List all the types of indices. What is the access cost of a data record given certain type of index?
7. B^+ tree, ISAM, hash indexes.

Summary:

1. Components of access time (to a particular sector): seek time, retational latency, and transfer time. Most of the time, the number of I/O operations (transfer) is the major component in response time to a query.
2. **Page**: the number of bytes transferred in each I/O transfer; Number of page transfers is often used to compare the efficiency between different queries.
3. What is a heap file? Unsorted file. Rows are appended to the end of the file. Need on average **F**/2 page transfers to determine if a given key exists; Successful insertion needs **F** page reads and one page write. Similar for deletion. Sometime needs to recompact the file as deletion might leave holes in pages.
4. What is a sorted file? Sorted by a key and stored accordingly. Needs log_2**F** page reads to locate a key; Sometime, sequential search is used (**F**/2) if seek time is large. Needs to maintain order when insertion is done; *fillfactor* and *overflow* can be used to compensate for sorted order.
5. What are equality search and range search? Equality search is a search for a single row; Range search is a search for a set of rows. Heap files are bad for range search. Sorted files are good if the range is a prefix of the key.

   Example of equality search:

   ```
   SELECT Id, Name, Address FROM Students WHERE Id=123456789
   ```

   where `ID` is the key of `Students`

   Example of range search:

   ```
   SELECT Id, Name, Address FROM Students WHERE Id>123456789
   ```

   where `ID` is the key of `Students`;
6. What is an index? A set of index entries and a location mechanism. Why are indices good? The size of an index entry is usually much smaller than the size of a data record. This means that the size of index file is much smaller comparing to the size of the data file. This leads to more efficient data location.
7. List all the types of indices. Integrated vs separate index file. Clustered vs unclustered. Sparse vs dense. Multilevel indexes: ISAM, B^+ tree, and hash indexes.
8. Clustered index: good for range search.

9. Multiple attributes index: provides finer granularity search and different types of range search. *Partial-key search*: search condition with unknown values for some attributes of the search key;

   If the search keys are $sk_1, sk_2, \ldots, sk_n$, a query that specifies the values of some keys and leaves others unknown is a *partial-key search*. For example, the query

   ```
   SELECT Id, CrsCode, Semester FROM Transcript WHERE CrsCode = 'CS482'
   ```

   is a partial search key if `Transcrip` has an index created by

   ```
   CREATE INDEX xyz ON Transcript (ID, CrsCode, Semester)
   ```

   For partial-key search, the index is good if a *prefix* of the key is specified; otherwise, it is not good.

10. What is the access cost of a data record given certain type of index? The cost depends on the size of the index and the location mechanism.

11. Multi-level index: second and higher level contains separator entries; sparse index only; reduces cost significantly.

12. ISAM: (we did not go over this in the class) main imdex. Each entry consists of a pointer and a search key that is the separator entry. Good for range search. Need compact operation from time to time. Static index. Not very good if the database changes a lot. The access cost depends on the number of levels and the fan-out number (denoted by ϕ). It is given by $log_\phi(Q) + 1$ where Q is the number of leaf pages.

13. B^+ tree: balanced tree. Pointers to data record at any level. Add *sibling pointers*. Dynamic index (change if insert/delete/update of keys). If the minimum number of separators stored in a page is $\phi/2$, the maximal cost of searching through a B^+ tree having Q leaf pages is $log_{\phi/2}Q + 1$.

14. Hash: good for equality search. Not good with range search. The choosing of the hash function is important.