Week 11 Notes

Organisation of Records in Files

- Unordered file (heap) - Records can be placed anywhere in the file
- Ordered file (sequential file) - store records in sequential order based on the value of an ordering field
- Hashed file - A hash function determines which block the file gets placed into

> For each type of organisation, some access is faster

DB Access Types

- Retrieval
  - Point query
  - Range query
- Update
  - Deletion, insertion and modification

Ordered Files

- Kept ordered by sorting the values of an ordering field
- Searching on the ordering field is efficient as we can use binary search
- Range query is efficient
- Insertion is expensive as the records must be inserted in correct order
- Modification is expensive

Hashed Files

- Fast for point queries
- File blocks are divided into M buckets B0 -> BM
- Typically one bucket is one disk block
- Designate field as hash key
- Use hash function to place and retrieve the record in the file

> The hash function is denoted h(K) where h is the function and K is the key to be hashed

Hashed Files Performance

- Slow for range queries
- Insertion is efficient | If the bucket is full you need a bucket chain
- Modification is more efficient than ordered but less efficient compared to heap
- Point queries are very fast

Indexes

Search Key - Attribute or set of attributes used to lookup records in a file An **index file** consists of records (called index entries) in the form [Search key | Pointer] The pointer points to the physical address of the record

Dense vs Sparse Indexes

Dense indexes have a pointer for every single **distinct** value Sparse indexes have index entries for some values but not for all

> If the data file is not ordered than you cannot use a sparse index

Primary Index

Defined on the primary key Primary indexes are sparse indexes Typically point to the block where the value occurs The pointers point to the first record of each block

Clustering Index (clustered)

Define on an ordered data file

- The data file is ordered on a non key value
- Clustering indexes are dense as each index entry points to the first occurrence of the key

Secondary Index

Primary and clustering index files used only for searching records based on the ordering fields One data file can only have one Primary or Clustering Index Secondary index entries allow searching of non ordering attributes as if they where ordered

Multilevel Index

If the index file is too large we can index the index file This is due to the desire of loading the entire index in memory to speed up searching Search the top level index to find the first level index value. Then the first level index value will point to the actual data.

Index structures in Commercial Systems

Most commercial DBMS systems support a multi-level index called **B+tree** index