INDEXING AND HASHING

* An index is a data structure that allows the DBMS to locate particular records in a file more quickly

Eg: Index works in the same way as index at the end of the textbook to locate various topics covered in a book

* An attribute or a set of attributes used to search records in a file called a search key.

Types of Indices

1) Ordered Indices: Based on sorted ordering of values
2) Hash Indices: Based on dividing the total data to be stored into a series of organized "buckets". The bucket to which a value is assigned is determined by a function, called a hash function.

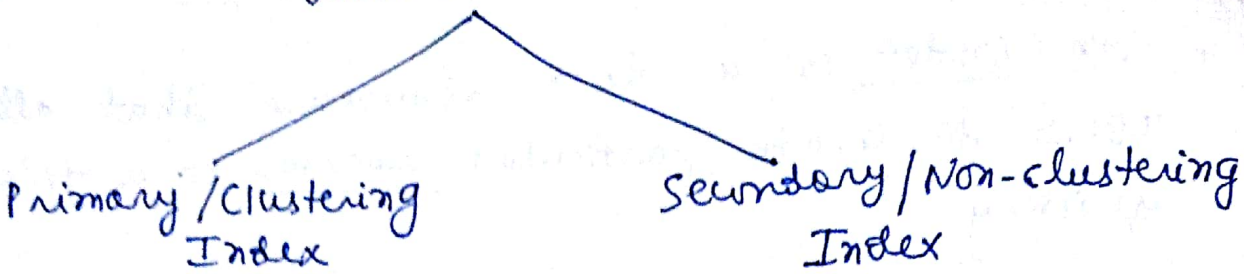
ORDERED INDICES

* To gain fast access to records in a file, an index structure is used.
* Each index structure is associated with a particular search key
* Just like index of a book, an ordered index stores the values of the search keys in sorted order and associates with each search key the records that contain it.
* An index file consists of records of the form:

search key	Pointer

* Index files are much smaller than the original file.

Ordered Indices



Primary / Clustering Index

* In a sequentially ordered file, the index whose search key specifies the sequential order of the file is called a primary index
* The search key of a primary index is usually the primary key.

Secondary / Non-Clustering Index

* An index whose search key specifies an order different from the sequential order of the file is called secondary index

Assumption

We assume that all files are ordered sequentially on some search key.

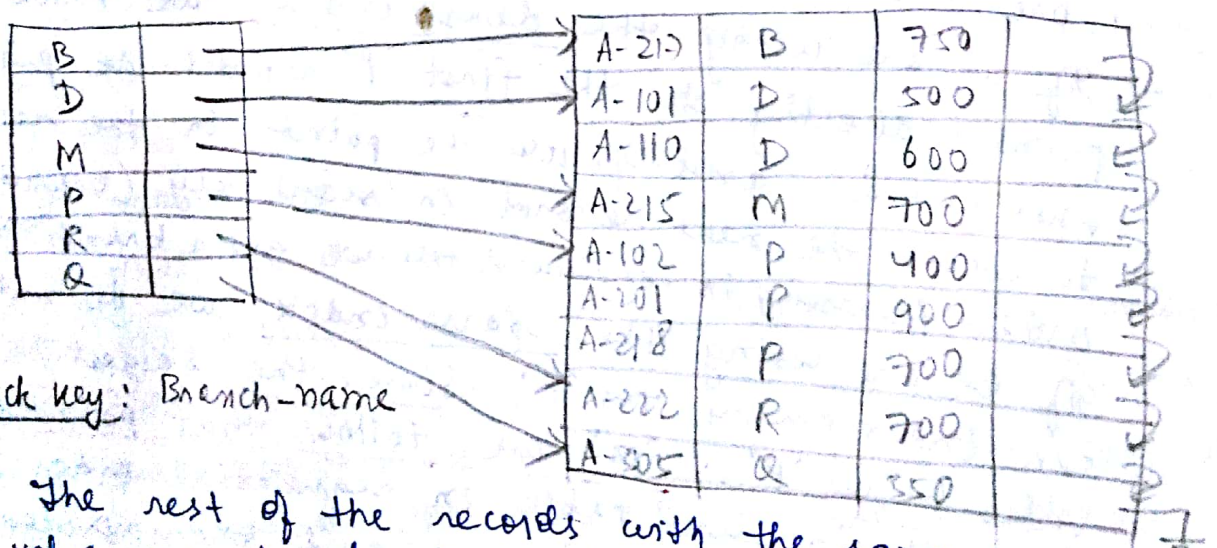
Such files with a primary / clustering index on the search key are called index-sequential files. It is one of the oldest indexing schemes used in database systems.

Dense and Sparse Indices

* The ordered indices are also classified as:
  1) Dense Index
  2) Sparse Index
* An index record / index entry consists of a search-key value and pointers to one or more records with that value as their search-key value.
* The pointer to a record consists of the identifier of a disk block and an offset value to identify records within the block.

Dense Index

* An index record appears for every search-key value in the file.
* In a dense clustering index, the index record contains the search key value and a pointer to the first record with that search key value.

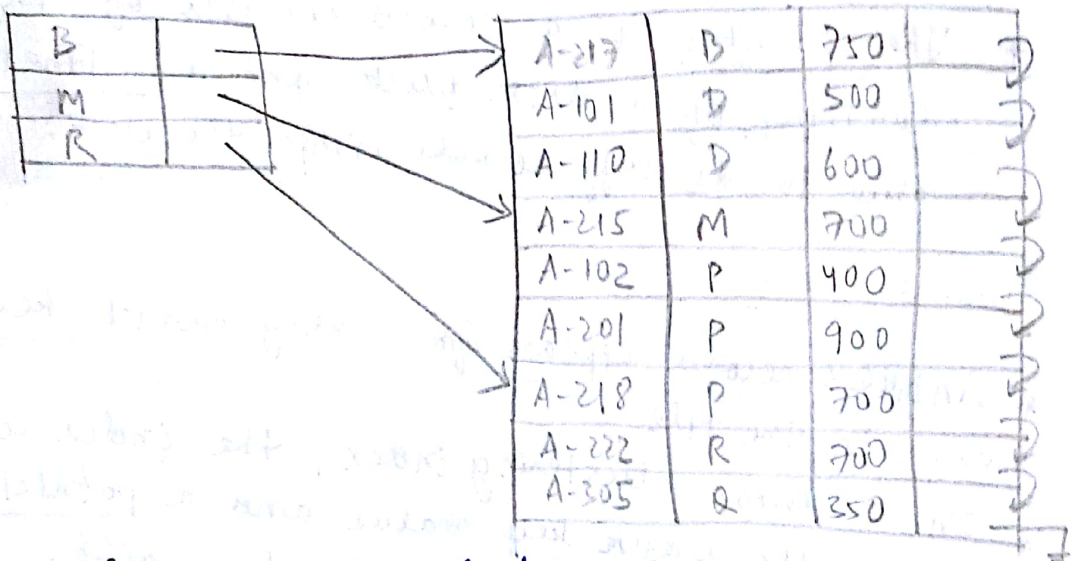


Search key: Branch-name

* The rest of the records with the same search-key value are stored sequentially after the first record
* As the index is clustered, records are sorted on the same search key.

Sparse Index

* An index record appears for only some of the search-key values
* To locate a record, we find the index entry with the largest search-key value that is less than or equal to the search-key value for which we are searching. We start at the record pointed to by that index entry and follow the pointers in the file until we find the desired record.



Eg: Suppose we are looking for records for the "p" branch.

- If we are using the dense index, we follow the pointer directly to the first 'P' record. We process this record, and follow the pointer in the record to locate the next record in search-key (branch-name) order. Processing is continued till we get a branch other than 'P'.
- If we are using the sparse index, we do not find an index entry for "P". Since the largest search-key value before 'P' is 'M', we follow that pointer. We read the "account" table in sequential order until we find the first "P" record and begin processing at that point.

Comparison of Dense Index and Sparse Index

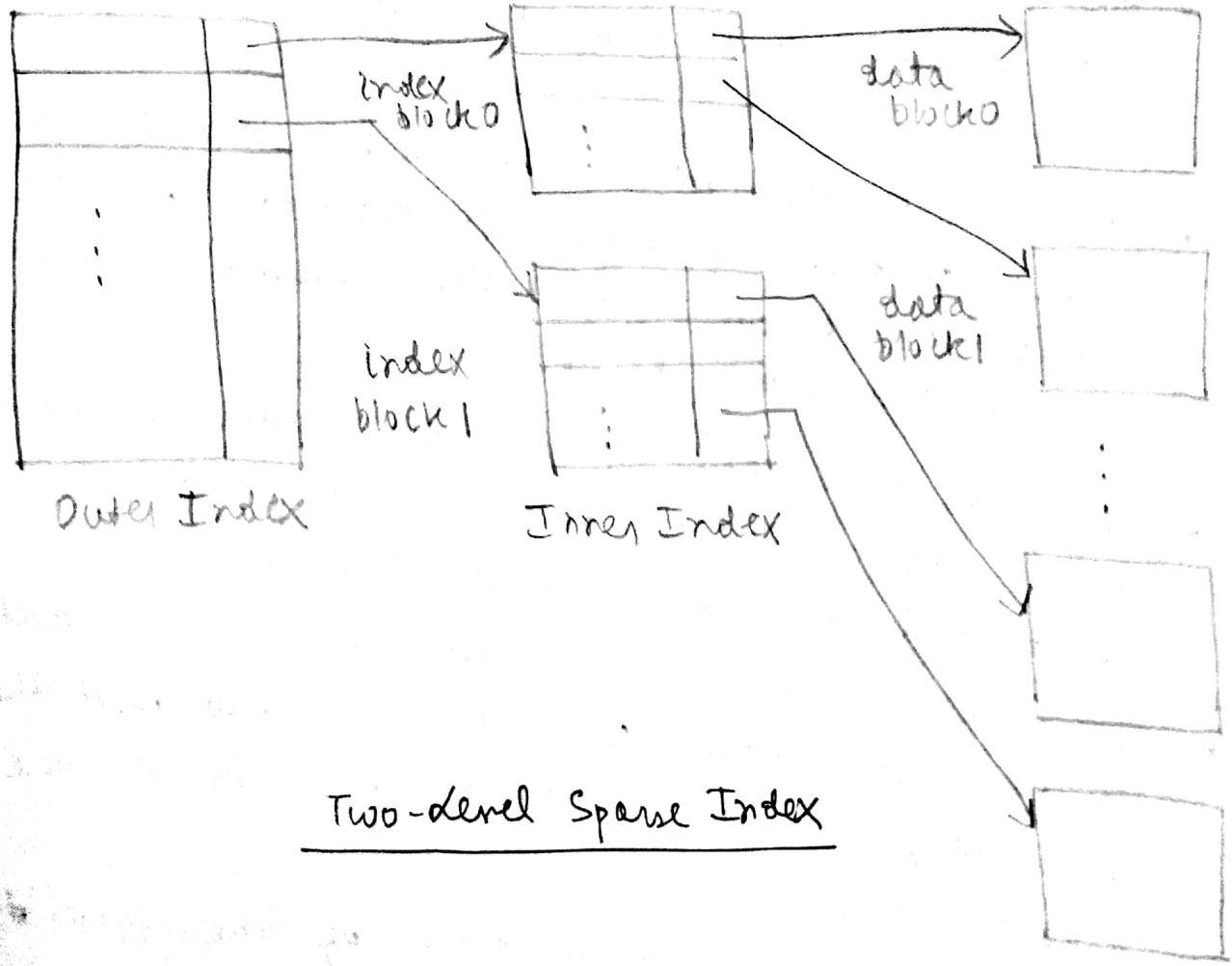
* It is generally faster to locate a record if we have a dense index rather than a sparse index.
* However, sparse indices have advantages over dense indices in that they require less space and impose less maintenance overhead for insertions and deletions.

Multilevel Indices

* If an index is sufficiently small to be kept in main memory, the search time to find a record is low.
* However, if the index is so large that it must be kept on disk, a search for an entry requires several disk-block reads. The process of searching a large index may be costly.
* To deal with this problem, the index is considered as a sequential file and a sparse index is constructed on the clustering index.
* To locate a record, we first use binary search on the outer index to find the record with the largest search-key value less than or equal to the one that we desire.
* The pointer points to a block of the inner index.
* The inner block is scanned to find the record that has the largest search key value less than or equal to the one that we are searching.
* The pointer in the inner block record points to

the block of the file that contains the record for which we are looking.

* If the file is extremely large, then the outer-index may become too large to fit in main memory. In such case, we can create another level of index.
This process can be repeated as many times as necessary.
* Indices with two or more levels are called multilevel indices.



Two-level Sparse Index

B^+ Tree Index Files

* The main disadvantage of the index - sequential file organization is that the performance degrades as the file grows, both for index lookups and for sequential scans through the data

* The B^+- tree index structure is the most widely used among several index structures that maintain their efficiency inspite of insertions and deletions

* A B^+- tree index takes the form of a balanced tree in which every path from the root of the tree to a leaf of a tree is of same length.