COS221 - L26 - Indexing

Linda Marshall

8 May 2023

Access Structures

- In the previous 2 lectures, unordered, ordered and hashing organisation of files were discussed.
- This lecture and the next assume that these files already exist and additional auxiliary access structures, called indexes, will be discussed.
  - Index files provide secondary access paths to records and do not affect the physical placement of the records.
  - Indexing fields are used to construct the index. Any field and any number of fields of a record can be indexed.
  - There are different index implementations, each using a different data structure to speed up the search.
  - The most common type of index is based on ordered files (single-level indexes) and tree data structures (multi-level indexes such as ISAM and B^+-trees). Other indexes are based on hashing or other search-based data structures, such as bitmap indexes.

Single-level Ordered Indexes

- A single-level ordered index is similar to the index in a textbook. A term is listed in the index with a list of page numbers where the term appears.
- A single-level ordered index is defined on a single field of a file (the indexing field/attribute). The index stores the index field value and a list of pointers to all disk blocks which contain a record with the particular field value.
- By ordering the index values, a binary search can be conducted on the index.

Single-level Ordered Indexes

- Several types of ordering indexes can be defined.
  - A *primary index* on the ordering key field. This is the unique field used to order the records on disk.
  - A *clustering index* is used when the ordering field is not unique. Clustering indexes are placed in a clustering file.
  - A *secondary index* is used on any non-ordering field of a file.

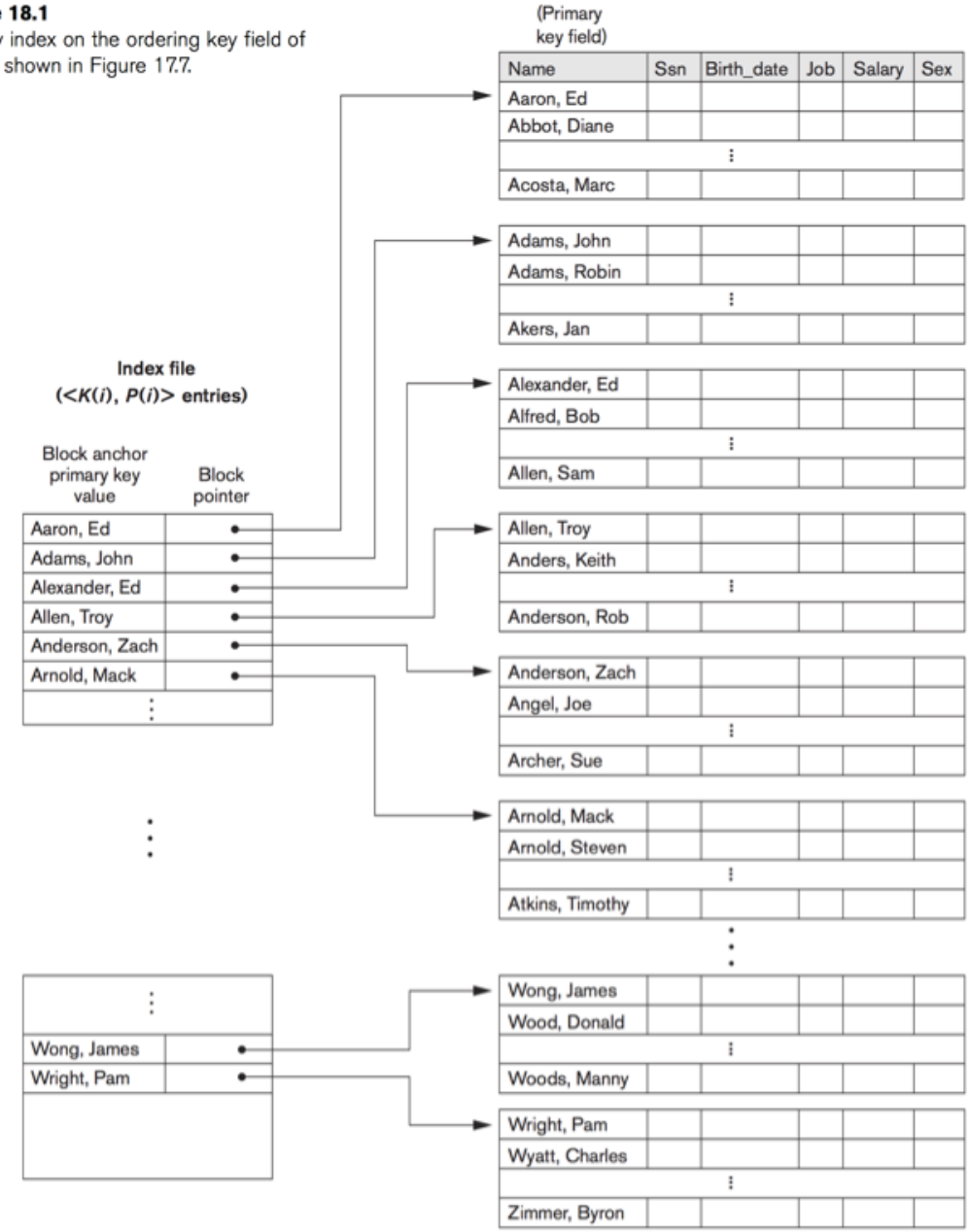
  Note, a file may have at most one primary index or clustering index. A file may have multiple secondary indexes.
- Indexes can be dense or sparse.
  - *Dense indexes* have an index entry for every record in the data file.
  - *Sparse indexes* have entries for some of the data file record entries.

Single-level Ordered Indexes - Primary indexes

- A primary index is on an ordered file where each entry in the file is a fixed length and comprises of 2 fields.
  - The first field is the same as the ordering key and has the value the same as the ordering key of the first record in the block.
  - The second field is the address of the block - that is the first record in the block.
- This is an example of a sparse index.

Single-level Ordered Indexes - Primary indexes

Figure 18.1
Primary index on the ordering key field of the file shown in Figure 17.7.



Single-level Ordered Indexes - Primary indexes

- The main problem with the primary index, or any ordered file for that matter, is with insertion and deletion.
- Insertion of a record in the data file, will require the moving of records in the data file. This moving may change the records at the beginning of blocks which in turn requires anchor fields to be updated. Using an unordered overflow file, as previously discussed, can alleviate the problem.
- Deletion is handled using deletion markers.

Single-level Ordered Indexes - Clustering indexes

- If the records are physically ordered on a non-key field, without distinct values, the ordering field is called a clustering field.
- The clustering index is an ordered file with two fields.
  - The first field is the same as the clustering field of the data file.
  - The second field is a disk block pointer to the first block in the data file that has a record with the first field value in the data file.

Single-level Ordered Indexes - Clustering indexes

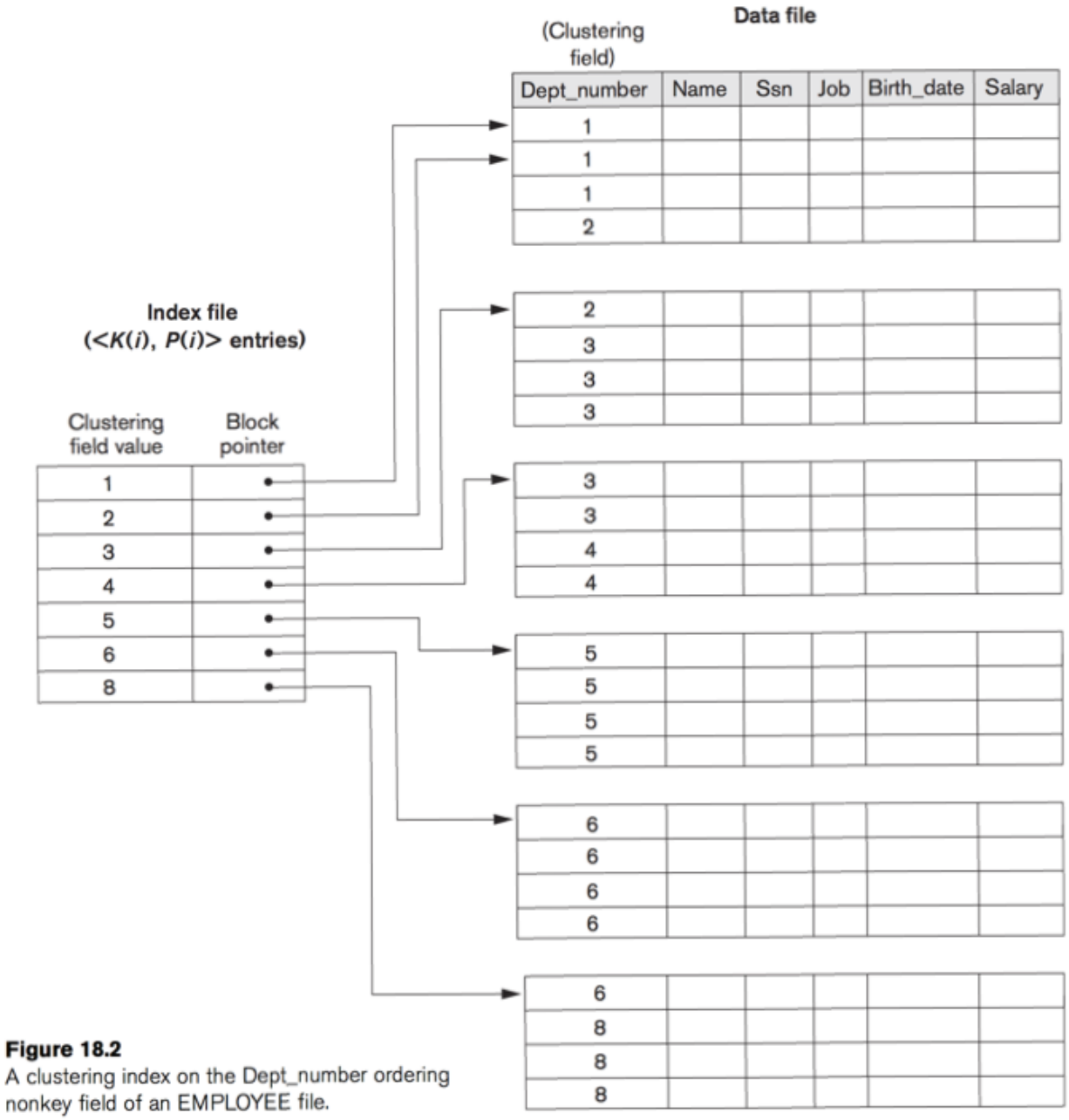


Figure 18.2
A clustering index on the Dept_number ordering nonkey field of an EMPLOYEE file.

Single-level Ordered Indexes - Clustering indexes

- Record insertion and deletion suffers from the same problems as primary indexes do.
- To alleviate this problem, a block is reserved per clustering field value. This means a record with a specific value for the field is placed in a specific block.
- Deletion requires the record to be removed from the block only.
- Clustering indexes are also an example of sparse indexes.

Single-level Ordered Indexes - Clustering indexes

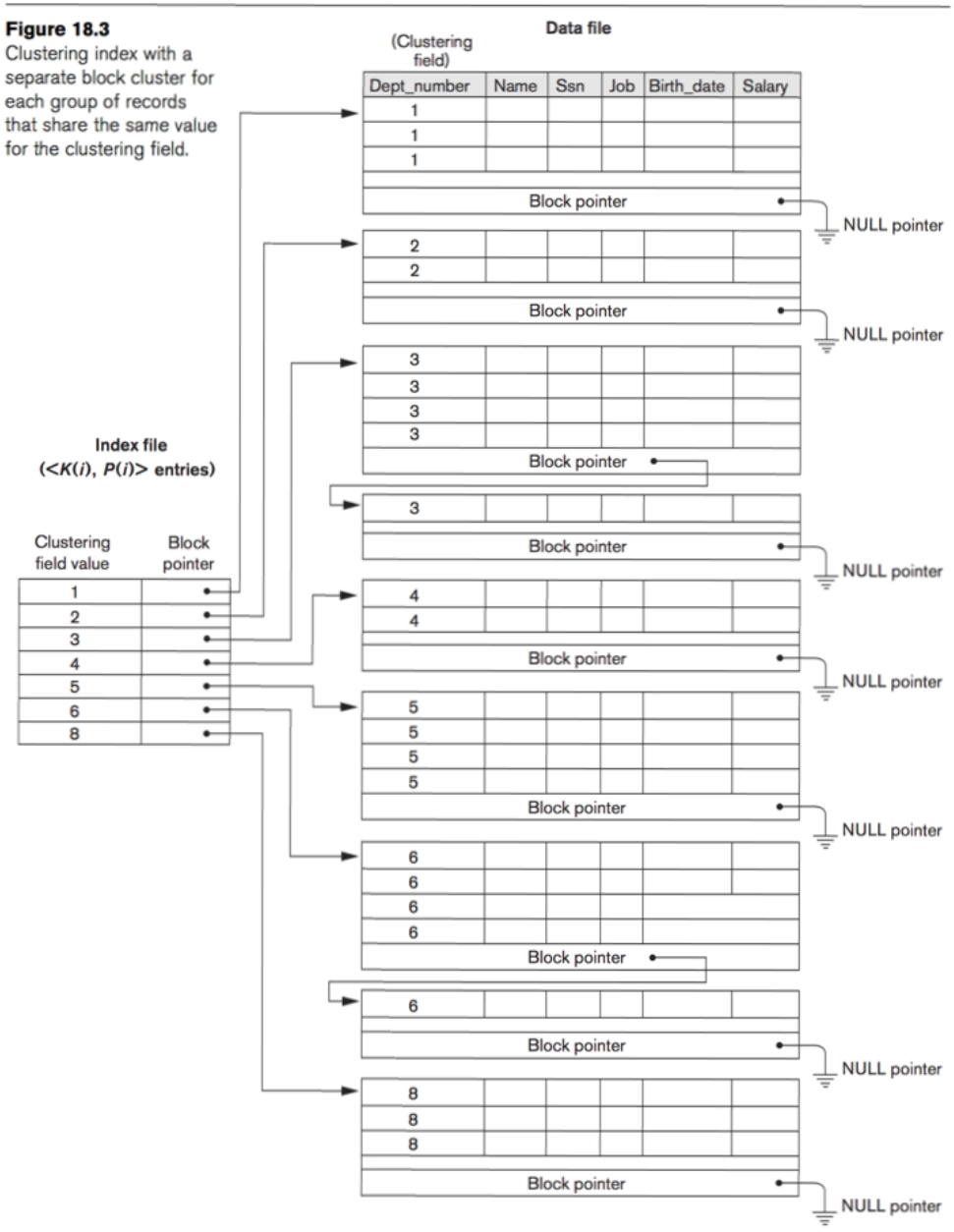


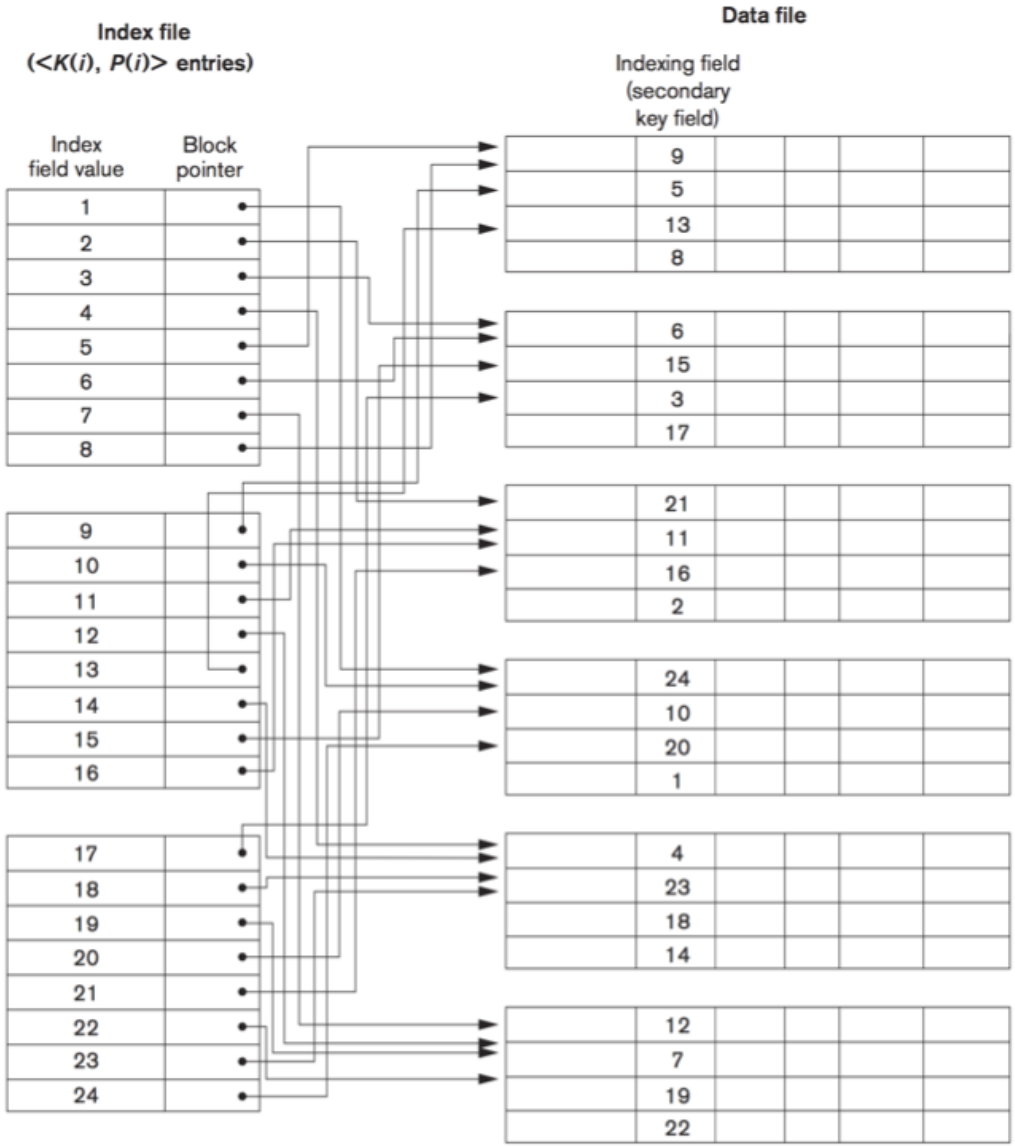
Figure 18.3
Clustering index with a separate block cluster for each group of records that share the same value for the clustering field.

Single-level Ordered Indexes - Secondary indexes

- Secondary indexes provide a second index for records which already have either a primary or a clustering index. In this case the data files, in terms of this indexing field, may be ordered, unordered or hashed. The secondary index may be created on a field that is unique (candidate key) or not unique.
- The entry in the secondary index consists of two fields.
  - The first field represents the non-ordering index
  - The second field is either a pointer to a block or record.
- A single data file may be indexed by multiple secondary indexes.
- Consider a secondary index on a (candidate) key - defined with UNIQUE.
  - This secondary index will require an entry for each instance of the field and therefore a one-to-one mapping exists between the index and the record of the data file.
  - The second field of the index will be a pointer to the record holding the field value.
  - This is an example of a dense index.

Single-level Ordered Indexes - Secondary indexes

Figure 18.4
A dense secondary index (with block pointers) on a nonordering key field of a file.



Single-level Ordered Indexes - Secondary indexes

- A secondary index usually requires more storage and longer search time than a primary index. Searching for an arbitrary field using a secondary index is way better than a linear search if the index had not existed.
- Creating a secondary index on a nonkey, unordered field can result in numerous records in the data file having the same value for the indexing field. Several options exist for implementing the index.
  - Include multiple entries for the index values - one per record. This will result in a dense index.
  - Create variable length entries for the index entries. The pointer will be represented by a repeating field.
  - Include an additional level of indirection. Have a pointer that points to a disk block comprising of record pointers. If the pointers for the index exceed a block, cluster or linked list of pointers is created. This is a sparse index.

Single-level Ordered Indexes - Secondary indexes

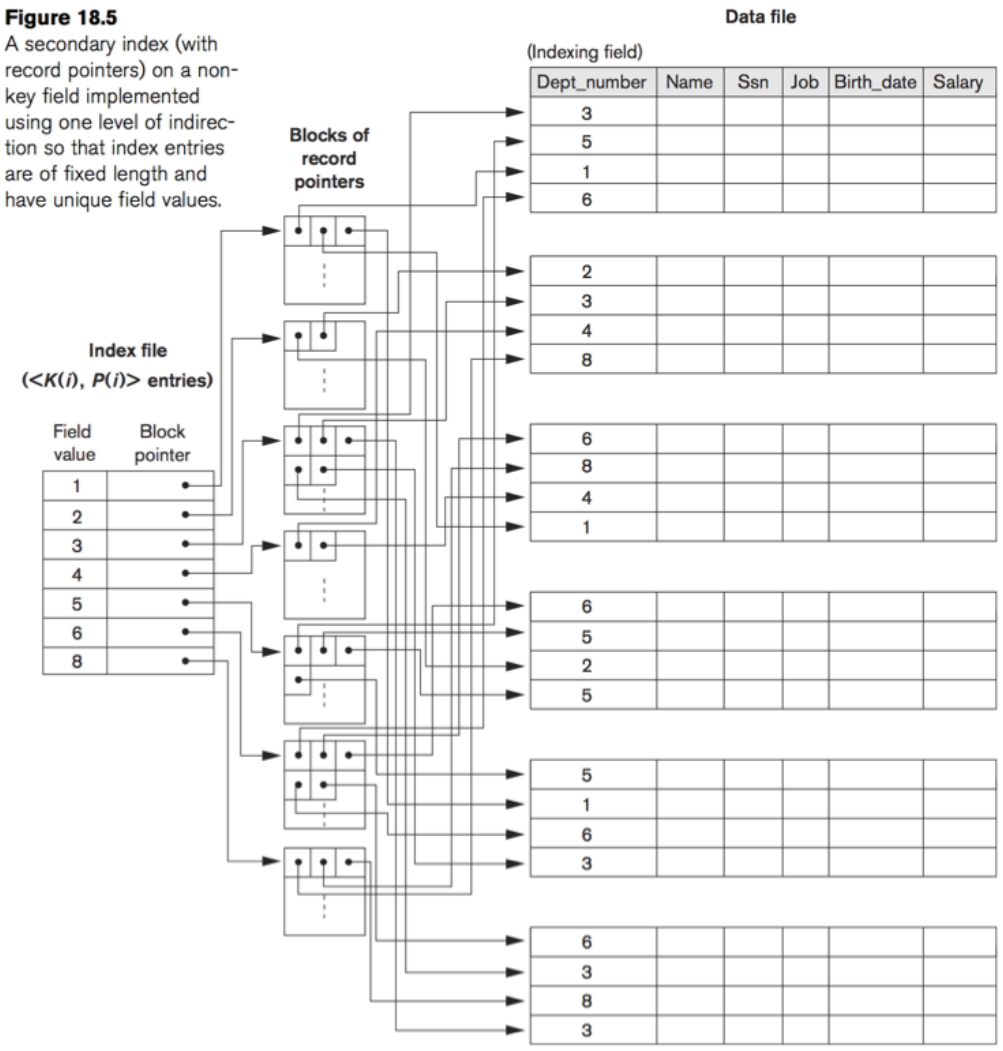


Figure 18.5
A secondary index (with record pointers) on a non-key field implemented using one level of indirection so that index entries are of fixed length and have unique field values.

What is to come?

- Multilevel Indexes
  - Want more than a binary search to reduce the search space quickly.
  - Make use of multilevel linear structures for indexing
  - Make use of tree-based structure, such as B-trees and B^+-trees, for indexing
- Indexes on multiple indexes, a key comprising of multiple values
  - Partitioned Hashing
  - Grid File
  - Other - Hash, Bitmap and Function-based indexes