## Chapter 3.1 The basics of Index Structure

***Definition:***

*Storage Structure:*

The Storage Structure is constructed by files, here the concept of file similar with the concept in Operation System.

*Example:*

One data file can be used to store one Relation. One data file can own one or multi - index files, each index file build the connection between the Query Key and Data Record. The pointer of the Query Key points to the record with the same property.

*Dense Index:*

The index could be Dense, which is to say that each record in data file will set it’s corresponding index in the index file.

*Sparse Index:*

The index could be Sparse, which is to say that some of the records in the index file will be present in the index file, normally each data block will be set one index in the index file.

*Main Index:*

The Main Index could be used to identify the location of the records in the data file.

*Auxiliary Index:*

The Auxiliary Index could not be used to identify the location of the records in the data file.

*(Normally, we will create the main index on the main key among Relation, and construct the Auxiliary Index on other attributes.)*

*Inverted Index:*

Given one or several key words, then through using Inverted Index, then we can get information from texts more efficiently.

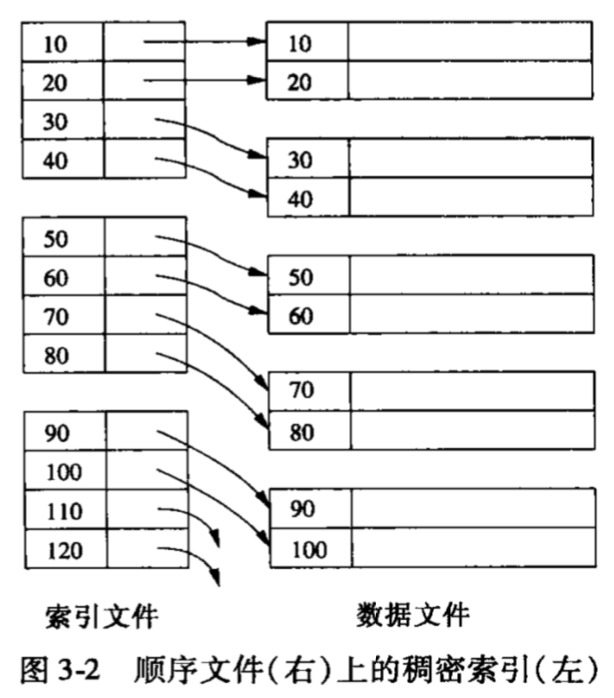
### Chapter 3.1.1 Sequential File

***Definition:***

Sequential File is the file that sorted all tuples of Relation according to the Main Key. The tuples are distributed among multi - blocks.

***Example:***

* In the image below, the sequential file has been displayed. Here assume that the Main Key is integer and there only exist two records in each storage block.
* The first block of the file just stored two records whose key value are 10 and 20. Also we use the continuous multiple as the Main Key.



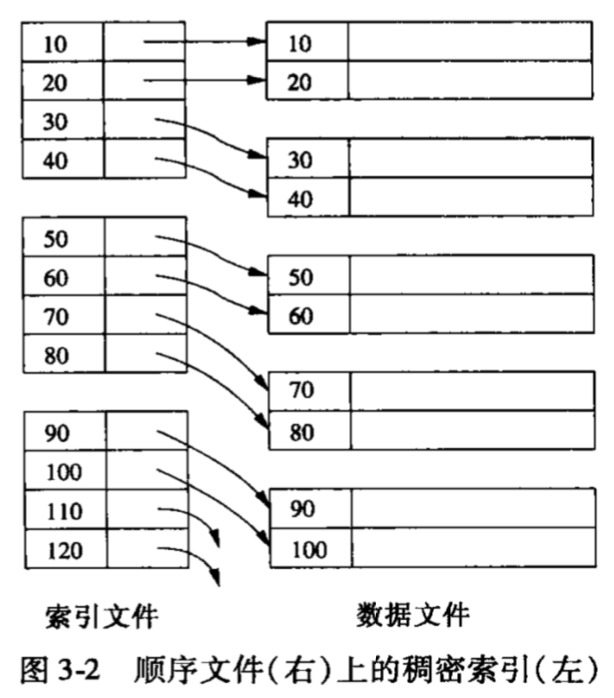
### Chapter 3.1.2 Dense Index

***Definition:***

* If the records are sorted, then here we can build Dense Index on records. Dense Index is such kind of storage block. Only save the record key and the pointers that point to records. The pointer is address. *(The Key Value Sequence of Dense Index is the same with the Sorted Sequence in file.)*
* Since the total size of Main Key and pointer are much less than the record, therefore we can assume that the blocks used for store index file is much smaller than store data file.
* When main memory can only be used to keep the index file, then it can increase the speed to query. Through using the index, then we can query only once to find the record with the given value.

***Example:***

Below is the Dense Index build on the Sequential File.



1. The first index block, four pointers point to the first four records.
2. The second index block, four pointers point to the next four records.

***Query based on Dense Index:***

Given the key value k of the record, then first search in the index block. After we find the K, then according to the corresponding pointer find the record from the data file.

Before we find the specific record, we need to search each storage block or almost half of the storage block.

***Advantage:***

* The number of index block is much less than data block.
* Since the Key are sorted, we can use Binary Search to find K. If there exist n index block, we only need to find log2(n).
* The index file may be small enough so that it can be saved in the main memory permanently. Then when we query, it only relates to the main memory access but no need other I/O operation.

### Chapter 3.1.3 Sparse Index

### Chapter 3.1.4 Multi - Level Index

### Chapter 3.1.5 Auxiliary Index

### Chapter 3.1.6 The Usage of Auxiliary Index

### Chapter 3.1.7 The Auxiliary Index’ Indirect

### Chapter 3.1.8 File Search and Inverted Index