

Chapter 7

Index Structures for Files

Indexes

- ❖ File already exists with some primary organization such as unordered, ordered, or hashed.
- ❖ **An Index** is a critical data structure which allows fast searching over large volumes of data.
- ❖ **Indexing** is used to speed up access to desired information from document collection as per users query
- ❖ But **searching** scans document corpus to find relevant documents that matches users query
- ❖ Different index structures might be used, but the most popular are the **inverted file, signature file, suffix tree, etc.**

Cont...

The user first **specifies an information need** in terms of queries which is then represented for searching.

Then with the help of matching module a comparison is made to find documents that are related with users' query.

Before the retrieved documents are sent to the user, the retrieved documents are **ranked** according to the likelihood of relevance

The user then **examines** the set of ranked documents in the search for useful information

Cont...

❖ Indexing and searching

- ❖ You cannot search that was not first indexed in some manner or other
- ❖ Indexing of documents or objects is done in order to be searchable
 - ❖ There are many ways to do indexing
- ❖ To index, one needs an indexing language
 - ❖ There are many indexing languages
 - ❖ Even taking every word in a document is an indexing language
- ❖ Knowing searching is knowing indexing

Cont...

- ❖ Indexing enhances efficiency in terms of time for retrieval.
 - Relevant documents are searched and retrieved quickly
- ❖ An **index file** consists of records, called **index entries**.
 - An index file of a document is a file consisting of a list of index terms and a link to one or more documents that has the index term
 - A good **index file** maps each keyword to a set of documents that contain the keyword.

Cont...

- ❖ Index file usually has index terms in a sorted order.
 - The sort order of the terms in the index file provides an order on a physical file
- ❖ An index file is list of search terms that are organized for associative look-up, i.e., to answer user's query:
- ❖ For organizing index file for a collection of documents, there are various data structure and/or file structure to use.
 - inverted file,
 - signature file
 - sequential file, etc.

Cont...

❖ Inverted file

- ❖ An indexing mechanism based on sorted list of keywords, with each keyword having links to the documents containing it
 - Building and maintaining an inverted index is a relatively low cost risk.
- ❖ An *inverted index* consists of two files: vocabulary and posting files
 - ❖ A **vocabulary file**:
 - Stores all of the **distinct** terms (keywords) that appear in any of the documents and
 - For each term a pointer to posting file

Cont...

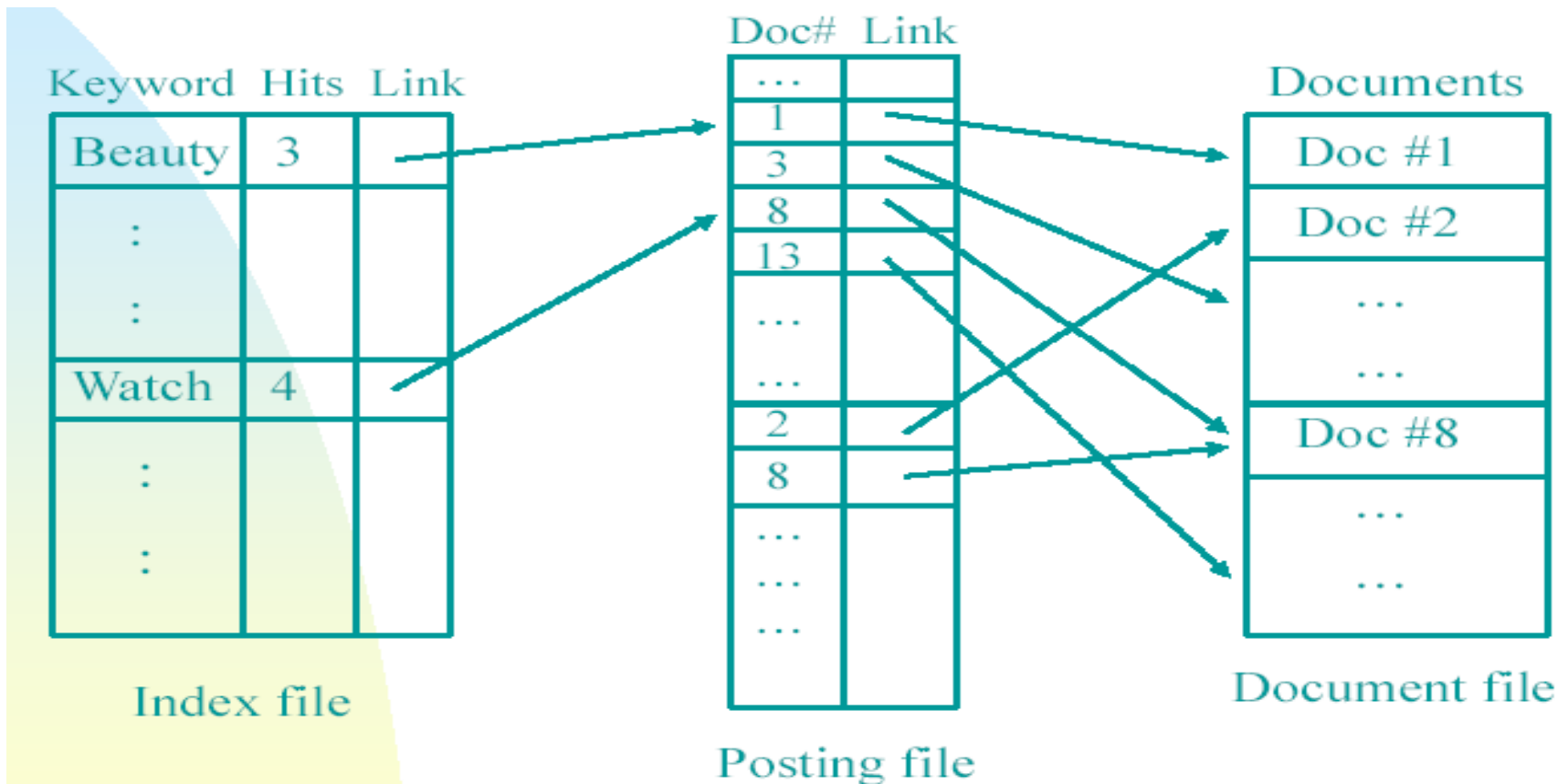
- ❖ Records kept for each term j in the vocabulary contains the following:
 - ❖ term j
 - ❖ number of documents in which term j occurs (n_j)
 - ❖ Total frequency of term j
 - ❖ pointer to inverted (postings) list for term j

❖ Postings File

- ❖ *For* each distinct term in the vocabulary, stores a list of *pointers* to the documents that contain that term.
- ❖ Each element in an inverted list is called a **posting**, i.e., the occurrence of a term in a document

Cont...

- ❖ The following figure shows the general structure of inverted index file.



Cont...

- ❖ Example: Given a collection of documents, they are parsed to extract words and these are saved with the Document ID.

Doc1

I did enact Julius
Caesar I was saved
i' the Capitol;
Brutus saved me.

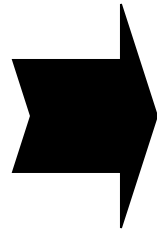
■ Doc2

So let it be with
Caesar. The noble
Brutus hath told you
Caesar was ambitious

- **Sorting the Vocabulary**: After all documents have been parsed the inverted file is sorted by terms Inverted index may record term locations within document during parsing

Cont...

Term	Doc #
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
Saved	1
i'	1
the	1
capitol	1
brutus	1
Saved	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2



Term	Doc #
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
Saved	1
Saved	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

Cont...

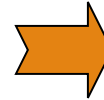
👉 **Remove duplicate terms & add frequency**

👉 Multiple term entries in a single document are merged and frequency information added

👉 Counting number of occurrence of terms in the collections helps to compute TF

Cont...

Term	Doc #
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2



Term	Doc #	Freq
ambitious	2	1
be	2	1
brutus	1	1
brutus	2	1
capitol	1	1
caesar	1	1
caesar	2	2
did	1	1
enact	1	1
hath	2	1
I	1	2
i'	1	1
it	2	1
julius	1	1
killed	1	2
let	2	1
me	1	1
noble	2	1
so	2	1
the	1	1
the	2	1
told	2	1
you	2	1
was	1	1
was	2	1
with	2	1

Cont...

❖ Vocabulary and postings file

- ❖ The file is commonly split in to a dictionary and a posting file

Cont...

Term	Doc #	Freq
ambitious	2	1
be	2	1
brutus	1	1
brutus	2	1
capitol	1	1
caesar	1	1
caesar	2	2
did	1	1
enact	1	1
hath	2	1
I	1	2
i'	1	1
it	2	1
julius	1	1
killed	1	2
let	2	1
me	1	1
noble	2	1
so	2	1
the	1	1
the	2	1
told	2	1
you	2	1
was	1	1
was	2	1
with	2	1



Term	N docs	Tot Freq
ambitious	1	1
be	1	1
brutus	2	2
capitol	1	1
caesar	2	3
did	1	1
enact	1	1
hath	1	1
I	1	2
i'	1	1
it	1	1
julius	1	1
killed	1	2
let	1	1
me	1	1
noble	1	1
so	1	1
the	2	2
told	1	1
you	1	1
was	2	2
with	1	1

Doc #	Freq
2	1
2	1
1	1
2	1
1	1
1	1
2	2
1	1
1	1
2	1
1	2
1	1
2	1
1	1
2	1
1	1
2	1
1	1
2	1
2	1
1	1
2	1
2	1
1	1
2	1
2	1

Pointers

Cont...

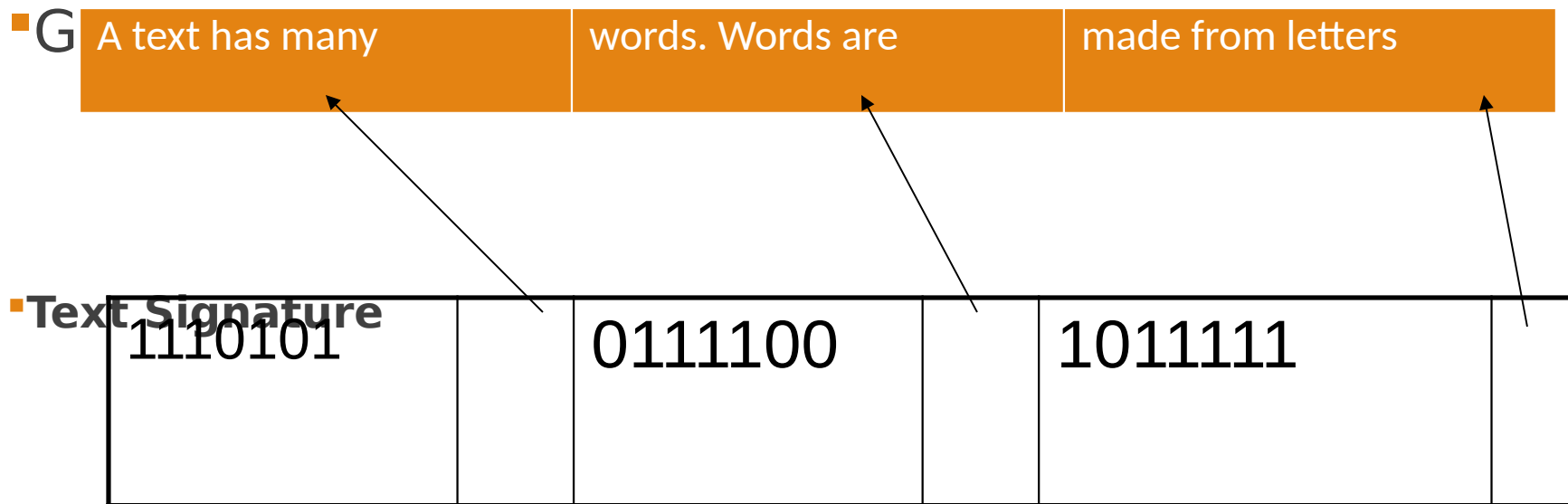
❖ Signature file

- ❖ Term-oriented index structures based on **hashing**
- ❖ How to build signature file
 - Hash each term to allocate fixed sized **F-bits** vector (term signature)
 - Divide the document in **blocks** of **N** terms each
 - Assign **F-bits** masks for each document block of size **N** (document signature)
 - This is obtained by bitwise **ORing** the signatures of all the terms in the document block.

Cont...

- Hence the signature file is no more than the sequence of bit masks of all blocks (plus a pointer to each block).

- Example



Cont...

Signature (hash) function:

$h(\text{text}) = 1000101$

$h(\text{many}) = 0110101$

$h(\text{word}) = 0111100$

$h(\text{made}) = 0010111$

$h(\text{letter}) = 1001011$

Block 4: 001100

OR 100001

101101

Cont...

Multilevel Indexes Using B-Trees and B+-Trees

- ▶ It is adopted for reducing index insertion and deletion problems by leaving some space in each block for appropriate insertion/deletion when the data file grows and shrinks.
- ▶ Implemented by using data structures called B-trees and B+-trees.
- ▶ B-trees and B+-trees is the well-known search data structure.
- ▶ B-trees and B+-trees, which are data structures that are commonly used in DBMSs to implement dynamically changing multilevel indexes

Cont...

- ▶ A **tree** is formed from **nodes**.
- ▶ Each node in the tree, except **root**, has one **parent** node and zero or more **child** nodes.
- ▶ The root node has no parent.
- ▶ A node that does not have any child nodes is called a **leaf** node; a non leaf node is called an **internal** node.
- ▶ The **level** of a node is always one more than the level of its parent, with the level of the root node being *zero*.

Cont...

901331_Fundamentals_of_Database_Systems_6th_Edition_(0136086209).pdf - Foxit Reader 2.2 - [901331_Fundamentals_of_Database_Systems_6th_Edition_(0136086209).]

File Edit View Language Document Tools Advanced Window Help

Merge and split PDF files

Find: 647

18.3 Dynamic Multilevel Indexes Using B-Trees and B+-Trees 647

Subtree for node B

Root node (level 0)

Nodes at level 1

Nodes at level 2

Nodes at level 3

(Nodes E, J, G, H, and K are leaf nodes of the tree)

Figure 18.7
A tree data structure that shows an unbalanced tree.

In Section 18.3.1, we introduce search trees and then discuss B-trees which can be used as dynamic multilevel indexes to guide the search for records in a data file. B-tree nodes are kept between 50 and 100 percent full, and pointers to the data blocks are stored in both internal nodes and leaf nodes of the B-tree structure. In Section 18.3.2 we discuss B+-trees, a variation of B-trees in which pointers to the data blocks of a file are stored only in leaf nodes, which can lead to fewer levels and higher-capacity indexes. In the DBMSs prevalent in the market today, the common structure used for indexing is B+-trees.

18.3.1 Search Trees and B-Trees

A search tree is a special type of tree that is used to guide the search for a record, given the value of one of the record's fields. The multilevel indexes discussed in Section 18.2 can be thought of as a variation of a search tree; each node in the multilevel index can have as many as fo pointers and fo key values, where fo is the index fan-out. The index field values in each node guide us to the next node, until we reach the data file block that contains the required records. By following a pointer, we restrict our search at each level to a subtree of the search tree and ignore all

Ready

676 of 1201

67.9%

File: 901331_Fundamentals_o

EN

5:14 PM
12/17/13

Thank you!!!