

# Indexing: Part IV

CPS 216  
Advanced Database Systems

---

---

---

---

---

---

---

---

## Announcements

2

- ❖ Homework #2 due in two days (February 26)
  - Typo corrected in Problem 5
  - You may work in groups of three, but then you must complete the optional part of either 8(c) or 8(d)
- ❖ Midterm next Monday (March 3)
  - Everything up to (including) today's lecture
  - Open-book, open-notes
- ❖ Course project proposal due in 9 days (March 5)
  - By email to [junyang@cs.duke.edu](mailto:junyang@cs.duke.edu)
- ❖ Reading assignment
  - Two papers on cache-sensitive indexing, by Rao and Ross, *VLDB* 1999 and *SIGMOD* 2000

---

---

---

---

---

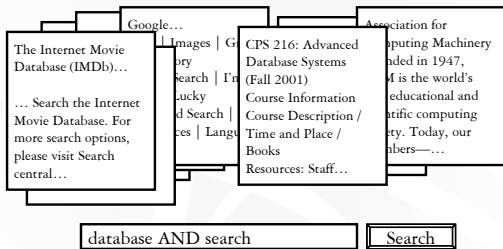
---

---

---

## Keyword search

3



What are the documents containing both "database" and "search"?

---

---

---

---

---

---

---

---

## Keywords × documents

4

All documents

All keywords

	Document 1	Document 2	Document 3	...	Document #
"a"	1	1	1	...	1
"cat"	1	1	0	...	0
"database"	0	0	1	...	0
"dog"	0	1	0	...	1
"search"	0	0	1	...	0
...	...	...	...	...	...

1 means keyword appears in the document  
0 means otherwise

- ❖ Inverted lists: store the matrix by rows
  - ❖ Signature files: store the matrix by columns
- With compression, of course!

---

---

---

---

---

---

---

---

---

---

## Inverted lists

5

- ❖ Store the matrix by rows
- ❖ For each keyword, store an inverted list
  - $\langle \text{keyword}, \text{doc-id-list} \rangle$
  - $\langle \text{"database"}, \{3, 7, 142, 857, \dots\} \rangle$
  - $\langle \text{"search"}, \{3, 9, 192, 512, \dots\} \rangle$
  - It helps to sort *doc-id-list* (why?)
- ❖ Vocabulary index on keywords
  - B<sup>+</sup>-tree or hash-based
- ❖ How large is an inverted list index?

---

---

---

---

---

---

---

---

---

---

## Using inverted lists

6

- ❖ Documents containing "database"
  - Use the vocabulary index to find the inverted list for "database"
  - Return documents in the inverted list
- ❖ Documents containing "database" AND "search"
  - Return documents in the intersection of the two inverted lists
- ❖ OR? NOT?

---

---

---

---

---

---

---

---

---

---

## What are “all” the keywords?

7

- ❖ All sequences of letters (up to a given length)?
  - ... that actually appear in documents!
- ❖ All words in English?
- ❖ Plus all phrases?
  - Alternative: approximate phrase search by proximity
- ❖ Minus all stop words
  - They appear in nearly every document; not useful in search
  - Example: a, of, the, it
- ❖ Combine words with common stems
  - They can be treated as the same for the purpose of search
  - Example: database, databases

---

---

---

---

---

---

---

---

## Frequency and proximity

8

- ❖ Frequency
  - $\langle \text{keyword}, \{ \langle \text{doc-id}, \text{number-of-occurrences} \rangle, \langle \text{doc-id}, \text{number-of-occurrences} \rangle, \dots \} \rangle$
- ❖ Proximity (and frequency)
  - $\langle \text{keyword}, \{ \langle \text{doc-id}, \langle \text{position-of-occurrence}_1, \text{position-of-occurrence}_2, \dots \rangle \rangle, \langle \text{doc-id}, \langle \text{position-of-occurrence}_1, \dots \rangle \rangle, \dots \} \rangle$
  - When doing AND, check for positions that are near

---

---

---

---

---

---

---

---

## Signature files

9

- ❖ Store the matrix by columns and compress them
- ❖ For each document, store a  $w$ -bit signature
- ❖ Each word is hashed into a  $w$ -bit value, with only  $s < w$  bits turned on
- ❖ Signature is computed by taking the bit-wise OR of the hash values of all words on the document

$\text{hash}(\text{"database"}) = 0110$       Does  $\text{doc}_3$  contain  
 $\text{hash}(\text{"dog"}) = 1100$        $\text{doc}_1$  contains "database": 0110 "database"?  
 $\text{hash}(\text{"cat"}) = 0010$        $\text{doc}_2$  contains "dog": 1100  
 $\text{doc}_3$  contains "cat" and "dog": 1110

☞ Some false positives; no false negatives

---

---

---

---

---

---

---

---

## Bit-sliced signature files

10

### ❖ Motivation

- To check if a document contains a word, we only need to check the bits that are set in the word's hash value
- So why bother retrieving all  $w$  bits of the signature?

doc	signature
1	000011100
2	000011100
3	000111100
4	011011100
...	...
$n$	000011100

Slice 7 ... Slice 0

Bit-sliced signature files

Starting to look like an inverted list again!

### ❖ Instead of storing $n$ signature files, store $w$ bit slices

### ❖ Only check the slices that correspond to the set bits in the word's hash value

### ❖ Start from the sparse slices

---

---

---

---

---

---

---

---

## Inverted lists versus signatures

11

### ❖ Inverted lists are better for most purposes (*TODS*, 1998)

### ❖ Problems of signature files

### ❖ Saving grace of signature files

---

---

---

---

---

---

---

---

## Suffix arrays (*SODA*, 1990)

12

### ❖ Another index for searching text

### ❖ Conceptually, to construct a suffix array for string $S$

- Enumerate all  $|S|$  suffixes of  $S$
- Sort these suffixes in lexicographical order

### ❖ To search for occurrences of a substring

- Do a binary search on the suffix array

---

---

---

---

---

---

---

---

## Suffix array example

13

$S = \text{mississippi}$        $q = \text{sip}$

Suffixes:	Sorted suffixes:	Suffix array:	
mississippi	i	10	
ississippi	ippi	7	
ssissippi	issippi	4	No need to store
sissippi	issippi	1	the suffix strings;
issippi	mississippi	0	just store where
ssippi	⇒ pi	9	they start
sippi	ppi	8	
ippi	⇒ sippi	6	
ppi	⇒ sissippi	3	
pi	ssippi	5	
i	ssissippi	2	

---

---

---

---

---

---

---

---

---

---

## One improvement

14

- ❖ Remember how much of the query string has been matched

$q = \text{sisterhood}$

...		
<i>low</i> :	⇒ sissipi...	Matched 3 characters
...		
<i>middle</i> :	⇒ sisterhood...	Start checking from the 4 <sup>th</sup> character
...		
<i>high</i> :	⇒ sistering...	Matched 5 characters
...		

---

---

---

---

---

---

---

---

---

---

## Another improvement

15

- ❖ Pre-compute the longest common prefix information between suffixes
  - For all (*low*, *middle*) and (*middle*, *high*) pairs that can come up in a binary search

$q = \text{sisterhood}$

...		
<i>low</i> :	⇒ sissipi...	Matched 3 characters
...		
<i>middle</i> :	⇒ sisterhood...	Start checking from the 6 <sup>th</sup> character
...	... } Matched 5 characters (pre-computed)	
<i>high</i> :	⇒ sistering...	Matched 5 characters
...		

---

---

---

---

---

---

---

---

---

---

## Suffix arrays versus inverted lists

- ❖ Suffix arrays are more powerful because they index all substrings (not just words)
  
- ❖ Suffix arrays use more space than inverted lists?
  - Check out compressed suffix arrays (*STOC 2000*)

---

---

---

---

---

---

---

---