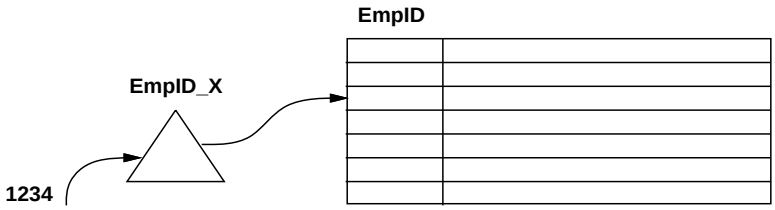
Contents

1 Indexing and Inverted Files **1**

1.1 Indexing on Relational Databases 1
1.2 Indexing on Documents 1
1.3 Inverted Index 1
1.4 Boolean Retrieval on Inverted Indexes 2
1.5 Advantages and Disadvantages of Inverted Indexes 2
1.6 Extensions on Inverted Indexes 2
1.7 Distance Constraints 2
1.8 Term Weights 3
1.9 Synonyms 3
1.10 Term Truncation 3
1.11 Conclusion 4

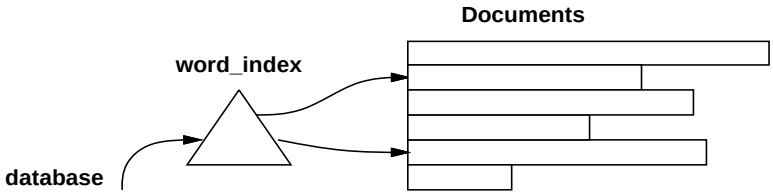
1 Indexing and Inverted Files

1.1 Indexing on Relational Databases



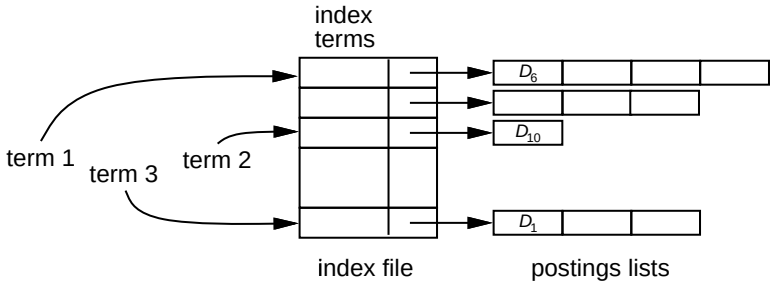
- Index structures: hashing, dynamic and linear hashing, B-tree, B+-trees.
- Index whole attribute value.
- Search whole attribute value.

1.2 Indexing on Documents



- Index structures: hashing, dynamic and linear hashing, B-tree, B+-trees, tries. Tries are good for dictionary-based searching.)
- Index a word within a document; if a document is considered as an attribute, then indexing is done on part of an attribute value.
- Partial match: '%database%', wildcards.
- Question: compare Oracle's SQL*TextRetrieval and OODBMS in handling text.

1.3 Inverted Index



- Index file can be implemented in any file structure (e.g., B-tree).
- Number of terms in the index file is the vocabulary of the document collection.

1.4 Boolean Retrieval on Inverted Indexes

- A Boolean query consists of n terms connected with Boolean operators. Parenthesis can be used to denote precedence.
- Each term returns a postings list from the inverted index. The postings list could be empty.
- Results are combined based on:
  - AND: set intersection
  - OR: set merge
  - NOT: set subtraction
- Standard optimization techniques apply: start with AND on the shortest lists; keep intermediate results as small as possible.

1.5 Advantages and Disadvantages of Inverted Indexes

Advantages:

- Fast retrieval.
- Flexible: different kinds of information can be stored.
- Support complex retrieval operations if enough information is stored.

Disadvantages:

- Large storage overhead (50% – 150%).
- High maintenance costs on updates, insertions and deletions.
- Processing cost increases with the number of Boolean operators.
- Questions: Why storage overhead could exceed 100%? How could some systems claim to have 2-3% storage overhead?

1.6 Extensions on Inverted Indexes

1.7 Distance Constraints

- Adjacency conditions are often needed. E.g.,
  - "database" immediately followed by "systems".
  - "database" and "systems" no more than 3 words apart.
  - "database" and "architecture" in the same sentence.
- Keep location information in the inverted index:
  - keeping sentence locations:

database	$\longrightarrow$	$R_{345}, 25$	$R_{348}, 37$	$R_{350}, 8$	... ...
systems	$\longrightarrow$	$R_{123}, 5$	$R_{128}, 25$	$R_{345}, 25$	... ...

  - keeping paragraph, sentence and word locations:

database	$\longrightarrow$	$R_{345}, 2, 3, 5$	$R_{348}, 37, 5, 9$	$R_{350}, 8, 12, 1$	... ...
systems	$\longrightarrow$	$R_{123}, 5, 4, 3$	$R_{128}, 25, 1, 12$	$R_{345}, 2, 3, 6$	... ...

1.8 Term Weights

- Occurrence frequencies can be stored in the index to support statistics-based retrieval:

database	$\longrightarrow$	$R_{345}, 10$	$R_{348}, 20$	$R_{350}, 1$	... ...
systems	$\longrightarrow$	$R_{123}, 82$	$R_{128}, 8$	$R_{345}, 12$	... ...

- The second component of a postings could be a weight by itself (e.g., a number between 0 and 1) or frequency information based on which a weight can be computed.
- Term frequency: Number of times that a term appears in a document.
- Document frequency: Number of documents containing a term.

1.9 Synonyms

- Synonyms are important for increasing the coverage of a query.
- Synonyms can be added to the index with pointers pointing to the same postings list.

1.10 Term Truncation

- Suffix truncation is a simple form of stemming:
  - comput* : computer, computing, computation, etc.
  - Can be handled easily on an inverted index.
- Prefix truncation and infix truncation:

  - *symmetry: symmetry, asymmetry, etc.
  - wom*n: woman, women.
  - Can be handled by a rotated index. Given a term abcde:
    * append word terminator: abcde#
    * generate all possible rotations: abcde#, #abcde, e#abcd, de#abc, etc
    * try to search for (in UNIX convention): ^abc*, cde$, *bcd*, b*e etc.
  - note the difficulty in implementing prefix-truncation, or in general, doing partial matching on the index.

1.11 Conclusion

- The overall effect of including more and more information into the index is higher and higher storage overhead. It has been reported in the literature that the storage overhead could reach 300%.
- The post-processing stage (after the postings are retrieved) becomes more and more expensive.
- Inverted indexes are good for relatively static environment.
- Research issues:
  - Compression of index and postings files.
  - Fast insertion methods.