WIKIPEDIA

Inverted index

In computer science, an **inverted index** (also referred to as **postings file** or **inverted file**) is an index data structure storing a mapping from content, such as words or numbers, to its locations in a database file, or in a document or a set of documents (named in contrast to a forward index, which maps from documents to content). The purpose of an inverted index is to allow fast full text searches, at a cost of increased processing when a document is added to the database. The inverted file may be the database file itself, rather than its index. It is the most popular data structure used in document retrieval systems,[1] used on a large scale for example in search engines. Additionally, several significant general-purpose mainframe-based database management systems have used inverted list architectures, including ADABAS, DATACOM/DB, and Model 204.

There are two main variants of inverted indexes: A **record-level inverted index** (or **inverted file index** or just **inverted file**) contains a list of references to documents for each word. A **word-level inverted index** (or **full inverted index** or **inverted list**) additionally contains the positions of each word within a document.[2] The latter form offers more functionality (like phrase searches), but needs more processing power and space to be created.

Contents

Applications

The inverted index data structure is a central component of a typical search engine indexing algorithm. A goal of a search engine implementation is to optimize the speed of the query: find the documents where word X occurs. Once a forward index is developed, which stores lists of words per document, it is next inverted to develop an inverted index. Querying the forward index would require sequential iteration through each document and to each word to verify a matching document. The time, memory, and processing resources to perform such a query are not always technically realistic. Instead of listing the words per document in the forward index, the inverted index data structure is developed which lists the documents per word.

With the inverted index created, the query can now be resolved by jumping to the word ID (via random access) in the inverted index.

In pre-computer times, concordances to important books were manually assembled. These were effectively inverted indexes with a small amount of accompanying commentary that required a tremendous amount of effort to produce.

In bioinformatics, inverted indexes are very important in the sequence assembly of short fragments of sequenced DNA. One way to find the source of a fragment is to search for it against a reference DNA sequence. A small number of mismatches (due to differences between the sequenced DNA and reference DNA, or errors) can be accounted for by dividing the fragment into smaller fragments—at least one subfragment is likely to match the reference DNA sequence. The matching requires constructing an inverted index of all substrings of a certain length from the reference DNA sequence. Since the human DNA contains more than 3 billion base pairs, and we need to store a DNA substring for every index and a 32-bit integer for index itself, the storage requirement for such an inverted index would probably be in the tens of gigabytes.

See also

- Index (search engine)
- Reverse index
- Vector space model

Bibliography

- Knuth, D. E. (1997) [1973]. "6.5. Retrieval on Secondary Keys". *The Art of Computer Programming* (Third ed.). Reading, Massachusetts: Addison-Wesley. ISBN 0-201-89685-0.
- Zobel, Justin; Moffat, Alistair; Ramamohanarao, Kotagiri (December 1998). "Inverted files versus signature files for text indexing". *ACM Transactions on Database Systems*. New York: Association for Computing Machinery. **23** (4): 453–490. doi:10.1145/296854.277632 (https://doi.org/10.1145%2F296854.277632).
- Zobel, Justin; Moffat, Alistair (July 2006). "Inverted Files for Text Search Engines". *ACM Computing Surveys*. New York: Association for Computing Machinery. **38** (2): 6. doi:10.1145/1132956.1132959 (https://doi.org/10.1145%2F1132956.1132959).
- Baeza-Yates, Ricardo; Ribeiro-Neto, Berthier (1999). *Modern information retrieval*. Reading, Massachusetts: Addison-Wesley Longman. p. 192. ISBN 0-201-39829-X.
- Salton, Gerard; Fox, Edward A.; Wu, Harry (1983). "Extended Boolean information retrieval". *Commun. ACM*. ACM. **26** (11): 1022. doi:10.1145/182.358466 (https://doi.org/10.1145%2F182.358466).
- *Information Retrieval: Implementing and Evaluating Search Engines* (http://www.ir.uwaterloo.ca/book/). Cambridge, Massachusetts: MIT Press. 2010. ISBN 978-0-262-02651-2.

References

1. Zobel, Moffat & Ramamohanarao 1998
2. Baeza-Yates & Ribeiro-Neto 1999, p. 192

External links

- NIST's Dictionary of Algorithms and Data Structures: inverted index (https://xlinux.nist.gov/dads/HTML/invertedIndex.html)
- Managing Gigabytes for Java (http://mg4j.di.unimi.it) a free full-text search engine for large document collections written in Java.
- Lucene (http://lucene.apache.org/java/docs/) - Apache Lucene is a full-featured text search engine library written in Java.
- Sphinx Search (http://sphinxsearch.com/) - Open source high-performance, full-featured text search engine library used by craigslist and others employing an inverted index.
- Example implementations (http://rosettacode.org/wiki/Inverted_Index) on Rosetta Code
- Caltech Large Scale Image Search Toolbox (https://web.archive.org/web/20101203074412/http://www.vision.caltech.edu/malaa/software/research/image-search/): a Matlab toolbox implementing Inverted File Bag-of-Words image search.