Bitmap-Tree Indexing for Set Operations on Free Text

Ilias Nitsos, Georgios Evangelidis
Department of Applied Informatics
University of Macedonia
156 Egnatia Str., 54006 Thessaloniki, Greece
{nitsos, gevan}@uom.gr

Dimitris Dervos
Department of Information Technology
TEI of Thessaloniki
P.O. Box 14561, 54101 Thessaloniki, Greece
dad@it.teithe.gr

Abstract

In the present study we report on our implementation of a hybrid-indexing scheme (Bitmap-Tree) that combines the advantages of bitmap indexing and file inversion. The results we obtained are compared to those of the compressed inverted file index. Both storage overhead and query processing efficiency are taken into consideration. The proposed new method is shown to excel in handling queries involving set operations. For general-purpose user queries, the Bitmap-Tree is shown to perform as good as the compressed inverted file index.

1. Introduction

Information processing environments range from the traditional database management system (DBMS), to textbase and hypertext IR systems. The inverted file index (IF) is a popular indexing scheme for textbases. Moreover, its compressed variation is reported to excel both in space and time savings, achieving performance improvements of the order of 80% when compared to uncompressed indexes [2].

Bitmap index file organizations utilize binary patterns. Each one of the latter indexes a distinct logical unit of information (e.g. a document). In each one binary pattern, bit positions are set to 1(0) in accordance to the presence (absence) of the corresponding index term [1]. Bitmap indexing has some clear advantages over file inversion that make it the method of choice for specialized application environments.

In the present study, we report on a compressed hybrid-indexing scheme: the Bitmap-Tree. The proposed new index structure is found to combine advantages from both bitmap indexing and file inversion. Bitmap and inverted index type components are gracefully integrated into a single (hybrid) compressed index structure. The scheme may be tailored for improved efficiency in processing set operations.

2. The Bitmap-Tree Structure

The Bitmap-Tree (BT) is a binary tree structure. In the Bitmap-Tree indexing scheme, each distinct word of the textbase is mapped to a unique number in $[0, \ldots, V-1]$, where V is the total number of distinct words in the textbase. For every one document in the textbase a document bitmap is constructed. Initially all its bits are set to 0 and then the bit positions that correspond to words contained in the document are set to 1.

Once a bitmap has been created for a document d, it is inserted into the binary Bitmap-Tree structure. Starting from the root, if the number of $1s$ in the bitmap of document d is greater than the number of $0s$, then the document bitmap is stored under the root node, alongside with the corresponding document identifier d, and the algorithm terminates. On the other hand, if the number of $1s$ is less than or equal to the number of $0s$, the document bitmap is divided into two equally sized sub-bitmaps. The algorithm proceeds recursively and considers each one of the two halves against the left- and the right- subtrees of the root, respectively.

3. Experimental Results

For specialized application environments calling for I/O intensive set oriented query processing operations, BT is measured to be up to 20 times faster than IF. For queries involving terms drawn from the general textbase vocabulary, BT performs similarly to IF. The overall gain comes at the cost of a small increase in storage overhead.

References

[1] Bookstein A., Klein S.T.: "Using Bitmaps for Medium Sized Information Retrieval Systems", *Information Processing & Management*, Vol. 26, No. 4, pp. 525-533, 1990.

[2] Zobel J., Moffat A. and Ramamohanarao K., "Inverted Files Versus Signature Files for Text Indexing", *ACM Transactions on Database Systems*, Vol.23, No.4, pp. 453-490, Dec. 1998.