Chapter LXIX
An Overview on Signature File Techniques

Yangjun Chen
University of Winnipeg, Canada

INTRODUCTION

An important question in information retrieval is how to create a database index which can be searched efficiently for the data one seeks. Today, one or more of the following four techniques have been frequently used: full text searching, B-trees, inversion and the signature file. Full text searching imposes no space overhead, but requires long response time. In contrast, B-trees, inversion and the signature file work quickly, but need a large intermediary representation structure (index), which provides direct links to relevant data. In this paper, we concentrate on the techniques of signature files and discuss different construction approaches of a signature file.

The signature technique cannot only be used in document databases, but also in relational and object-oriented databases. In a document database, a set of semistructured (XML) documents is stored and the queries related to keywords are frequently evaluated. To speed up the evaluation of such queries, we can construct signatures for words and superimpose them to establish signatures for document blocks, which can be used to cut off non-relevant documents as early as possible when evaluating a query. Especially, such a method can be extended to handle the so-called containment queries, for which not only the key words, but also the hierarchical structure of a document has to be considered. We can also handle queries issued to a relational or an object-oriented database using the signature technique by establishing signatures for attribute values, tuples, as well as tables and classes.

BACKGROUND

The signature file method was originally introduced as a text indexing methodology (Faloutsos, 1985; Faloutsos, Lee, Plaisant & Shneiderman, 1990). Nowadays, however, it is utilized in a wide range of applications, such as in office filing (Christodoulakis, Theodoridou, Ho, Papa & Pathria, 1986), hypertext systems (Faloutsos, Lee, Plaisant & Shneiderman, 1990), relational and object-oriented databases (Chang & Schek, 1989; Ishikawa, Kitagawa & Ohbo, 1993; Lee & Lee, 1992; Sacks-Davis, Kent, Ramamohanarao, Thom & Zobel, 1995; Yong, Lee & Kim, 1994, Tousidou et al., 2002; Chen, 2004), as well as in data mining (Andre-Joesson & Badal, 1997). It requires much smaller storage space than inverted files, and can handle insertion and update operations in databases easily.

A typical query processing with the signature file is as follows: When a query is given, a query signature is formed from the query value. Then each signature in the signature file is examined over the query signature. If a signature in the file matches the query signature, the corresponding data object becomes a candidate that may satisfy the query. Such an object is called a drop. The next step of the query processing is the false drop resolution. Each drop is accessed and examined whether it actually satisfies the query condition. Drops that fail the test are called false drops while the qualified data objects are called actual drops.

A variety of approaches for constructing signature files have been proposed, such as bit-slice files (Ishikawa, Kitagawa & Ohbo, 1993), S-trees (Deppisch, 1986), and signature trees (Chen, 2002; Chen, 2005), as well as their different variants. In the following, we overview all of them and analyze their computational complexities.

SIGNATURE FILES AND SIGNATURE FILE ORGANIZATION

Signature Files

Intuitively, a signature file can be considered as a set of bit strings, which are called signatures. Compared to the inverted index, the signature file is more efficient in handling new insertions and queries on parts of words. But the scheme introduces information loss. More specifically, its output usually involves a number of false drops, which may only be identified by means of a full text scanning on every text block short-listed in the output. Also, for each query processed, the entire signature file needs to be searched (Faloutsos, 1985; Faloutsos, 1992). Consequently, the signature file method involves high processing and I/O cost. This problem is mitigated by partitioning the signature file, as well as by exploiting parallel computer architecture (Ciaccia & Zezula, 1996).

When creating a signature file, each word is processed separately by a hashing function. The scheme sets a constant number (m) of 1s in the $[1..F]$ range. The resulting binary pattern is called the word signature. Each text is seen to consist of fixed size logical blocks and each block involves a constant number (D) of non-common, distinct words. The D word signatures of a block are superimposed (bit OR-ed) to produce a single F-bit pattern, which is the block signature stored as an entry in the signature file.

Fig. 1 depicts the signature generation and comparison process of a block containing three words (then $D = 3$), say "John", "12345678", and "professor". Each signature is of length $F = 12$, in which $m = 4$ bits are set to 1. When a query arrives, the block signatures are scanned and many non-qualifying blocks are discarded. The rest are either checked (so that the "false drops" are discarded; see below) or they are returned to the user as they are. Concretely, a query specifying certain values to be searched for will be transformed into a query signature s_q in the same way as for word signatures. The query signature is then compared to every block signature in the signature file. Three possible outcomes of the comparison are exemplified in Fig. 1: (1) the block matches the query; that is, for every bit set in s_q, the corresponding bit in the block signature s is