Tree-Based Indexing

Yannis Manolopoulos[1], Yannis Theodoridis[2], and Vassilis J. Tsotras[3]
[1]Aristotle University, Thessaloniki, Greece
[2]University of Piraeus, Piraeus, Greece
[3]University of California-Riverside, Riverside, MA, USA

Synonyms

B+-tree; Index Sequential Access Method (ISAM); R-tree

Definition

Consider a relation R with some numeric attribute A taking values over an (ordered) domain D. A *range query* retrieves all tuples in R whose attribute A has values in the interval [*low, high*]. That is, $low \le R.A \le high$. To enable fast processing of range selection queries, an access method that maintains order is needed. Such an index has the form of a tree, where each node corresponds to a page. Leaf nodes contain (or index) the actual values of A, while index nodes provide ordered access to the nodes underneath. Examples of tree-based indexing are the B+-tree and the R-tree (for single- and multidimensional ranges, respectively).

Key Points

A major performance goal of a database management system is to minimize the number of I/Os (i.e., blocks or pages transferred) between the disk and main memory. One way to achieve this goal is to minimize the number of I/Os when answering a query. Consider, for example, relation *Employee* (*ssn*, *name*, *salary*, *dept.*, *address*); the query, "Find the employees who reside in Santa Monica," references only a fraction of *Employee* records. It would be very inefficient to have the database system sequentially read all the pages of the *Employee* file and check the *address* field of each employee record for the value "Santa Monica." Instead the system should be able to locate the pages with "Santa Monica" employee records directly.

To allow such fast access, additional data structures called access methods (or indices) are designed per database file. The term *search-key* is used to identify the attribute(s) on which the access method is built. There are two fundamental access methods, namely, tree-based and hash-based indexing. They differ on the kind of queries that they can efficiently address. Tree-based indexing maintains the order of the search-key values. It is thus applicable to attributes that are *numeric*, and hence it can be used to address *range* queries (and also *equality* queries, when the range query interval is reduced to one value). Hash-based indexing on the other hand does not assume any ordering; rather it is based on mapping the search-key values on a collection of

© Springer Science+Business Media LLC 2017
L. Liu, M.T. Özsu (eds.), *Encyclopedia of Database Systems*,
DOI 10.1007/978-1-4899-7993-3_755-2

buckets. Therefore it can only address *equality* (or *membership*) queries. Since a tree-based index maintains the order of the indexed values in its leaf pages, a (one-dimensional) range query is implemented as a search for the leaf page with the lower value of the range interval, followed by the accessing of sibling pages until a page that contains the higher value of the range interval is reached.

Tree-based indices are further categorized by whether their search-key ordering is the same with the file's logical order (if any). Note that a file may or may not be ordered according to the value of an (a sequence of) attribute(s). A file stored without any logical order is called an unordered file or heap. If the search-key of a tree-based index is the same as the ordering attribute of a (ordered) file, then the index is called *primary*. Since such an index also clusters the values of the file, the term *clustering* index has also been used. The search-key of a primary index is usually the file's primary key; however this is not necessary. An index built on any non-ordering attribute of a file is called *secondary*. A relation can have several indices, on different search-keys; among them, at most one is primary (clustering) index, and the rest are secondary ones (obviously, a file can have at most a single logical order since it is physically stored once).

Historically, the first tree-based index used in relational databases was the ISAM file (a static tree). Currently, the most widely used tree-based index is the B+-tree which is a dynamic structure (i.e., its size increases or decreases as the size of the indexed file changes by record insertions/deletions). As with any data structure, the performance of an access method is characterized by three costs, namely, query time, update time, and space. Query time corresponds to the number of page accesses (I/Os) needed to answer a query. Update time counts the number of I/Os needed for updating the method when a record is updated, inserted, or deleted. Space measures the number of pages occupied by the method's data structures. The B+-tree uses logarithmic update and query costs, while its space requirements are linear to the size of the indexed file.

Cross-References

▶ Access Path
▶ Hash-Based Indexing
▶ Index Creation and File Structures
▶ Primary Index
▶ Secondary Index
▶ Signature Trees
▶ Spatial Indexing Techniques
▶ Suffix Tree
▶ Text Indexing Techniques

Recommended Reading

1. Elmasri RA, Navathe SB. Fundamentals of database systems. 5th ed. Reading: Addisson-Wesley; 2007.
2. Manolopoulos TY, Tsotras Y, Vassilis J. Advanced database indexing. Dordecht: Kluwer; 1999.
3. Ramakrishnan R, Gehrke J. Database management systems. 3rd ed. New York: McGraw-Hill; 2003.