S

Secondary Index

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

Non-clustering index

Definition

A tree-based index is called a *secondary index* if the order which it maintains on the search-key values is *not* the same as the order of the file which it indexes. For example, consider a relation R with some numeric attribute A taking values over an (ordered) domain D. Assume that relation R is *not* physically stored on the values of attribute A (i.e., relation R is either stored as a heap – an unordered file, or is ordered on another attribute). Furthermore, assume that a tree-based index (e.g., B+ -tree) has been created on attribute A. Then this index is secondary.

Key Points

Tree-based indices are built on numeric attributes and maintain an order among the indexed search-key values. They are further categorized by whether their search-key ordering is the same with the file's physical order (if any). Note that a file may or may not be ordered. Ordered is a file whose records are stored in pages according to the order of the values of an attribute. Obviously, a file can have at most a single such order since it is physically stored once. For example, if the *Employee* relation is ordered according to the *name* attribute, the values in the other attributes will not be in order. A file stored without any order is called an unordered file or heap. An index built on any non-ordering attribute of a file is called *secondary* (or non-clustering) while an index built on the ordering attribute of a file is called *primary* (clustering).

Since the actual data record can be anywhere in the file, the secondary index needs an extra level of indirection, namely, a pointer to the actual position of a record with a given value in the relation file. In other words, a secondary index only clusters references to records (in the form of <value, pointer>fields), but *not* the records themselves. This extra indirection from a leaf page of a secondary index to the actual position of a record in a file has important subsequences on optimization. Consider, for example, a secondary index (B+ -tree) on the *ssn* attribute of

L. Liu, M.T. Özsu (eds.), *Encyclopedia of Database Systems*,
DOI 10.1007/978-1-4899-7993-3_758-2

the *Employee* relation (which assume is ordered by the *name* attribute). A query that asks for the salaries of employees with *ssn* in the range *[x, y]* can facilitate the B+ -tree on *ssn* to retrieve references to all records in the query range. Assume there are 1,000 such *ssn* values in the *Employee* file. Since the actual *Employee* records must be retrieved (so as to report their salaries), each such reference needs to be materialized by possibly a separate page I/O (since the actual records can be in different pages of the *Employee* file).

A relation can have several indices, on different search-keys; among them, at most one is primary (clustering) index while the rest are secondary ones.

Cross-References

- Access Methods
- B+ -Tree
- Index Sequential Access Method (ISAM)
- Indexing

Recommended Reading

1. Elmasri RA, Shamkant NB. 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.