P

Primary 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

Clustering index; Sparse index

Definition

A tree-based index is called a *primary index* if its order is the same as the order of the file which it indexes Consider a relation R with some numeric attribute A taking values over an (ordered) domain D. Assume that relation R has been physically stored as an *ordered* file, following the order of the values of attribute A. Furthermore, assume that a tree-based index (e.g., B +-tree) has been created on attribute A. Then this index is primary.

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. 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*. An index built on any non-ordering attribute of a file is called *secondary*.

Note that a primary index also controls the physical placement of the data records in a file. Since such an index also clusters the values of the file, the term *clustering* index has alternatively been used. In contrast, the order in the leaf pages of a secondary index is not the same as the order of the records in the file. Hence, a secondary index (also called *non-clustering*) 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

L. Liu, M.T. Özsu (eds.), *Encyclopedia of Database Systems*,
DOI 10.1007/978-1-4899-7993-3_757-2

index (B+ -tree) on the *ssn* attribute of 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). If instead the file was ordered on the *ssn* attribute, the B+ -tree on *ssn* would have clustered (as primary index) the *Employee* records on *ssn*, and thus the 1,000 records that need to be retrieved would be located within few pages.

In practice, the search-key of a primary index is usually the file's primary key, however this is not necessary. That is, primary indexes need not be on the primary keys of relations. (In the above example, it was first assumed that the *Employee* file is ordered according to the *name* attribute and not according to the primary key *ssn* attribute). 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.

Cross-References

- Access Path
- Index Creation and File Structures
- Index Sequential Access Method (ISAM)
- Secondary Index

Recommended Reading

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