CHAPTER 14

Indexing

Many queries reference only a small proportion of the records in a file. For example, a query like “Find all instructors in the Physics department” or “Find the total number of credits earned by the student with ID 22201” references only a fraction of the instructor or student records. It is inefficient for the system to read every tuple in the instructor relation to check if the dept_name value is “Physics”. Likewise, it is inefficient to read the entire student relation just to find the one tuple for the ID “22201”. Ideally, the system should be able to locate these records directly. To allow these forms of access, we design additional structures that we associate with files.

Bibliographical Notes

Discussions of the basic data structures in indexing can be found in [Cormen et al. (2009)]. B-tree indices were first introduced in [Bayer and McCreight (1972)] and [Bayer (1972)]. B+-trees are discussed in [Comer (1979)], [Bayer and Unterauer (1977)], and [Knuth (1973)]. [Gray and Reuter (1993)] provide a good description of issues in the implementation of B+-trees.

Index structures optimized for flash storage include the Lazy-Adaptive tree of [Agrawal et al. (2009)], and the FD tree of [Li et al. (2010)]. The Cache-Sensitive B+-tree (CSB+-tree) presented by [Rao and Ross (2000)] is a B+-tree variant designed to minimize cache misses. The Bw-tree [Levadoski et al. (2013)] is a B+-tree variant optimized for main-memory, and features latch-free traversal and updates which minimize concurrency control overheads (concurrency control is discussed later, in Chapter 18). [Faerber et al. (2017)] provide a survey of main-memory databases, including coverage of main-memory indexing techniques.

The Log-Structured Merge (LSM) tree is presented in [O’Neil et al. (1996)], while the Stepped Merge tree is presented in [Jagadish et al. (1997)]. The buffer tree is presented in [Arge (2003)]. [Vitter (2001)] provides an extensive survey of external-memory data structures and algorithms.
Several alternative tree and treelike search structures have been proposed. Skip-lists [Pugh (1990)] are a probabilistic alternative to balanced tree structures, and are used for in-memory indexing in some in-memory databases.

Bitmap indices, and variants called bit-sliced indices and projection indices, are described in [O’Neil and Quass (1997)]. They were first introduced in the IBM Model 204 file manager on the AS 400 platform. They provide very large speedups on certain types of queries, and are today implemented on most database systems. Research on bitmap indices includes [Wu and Buchmann (1998)], [Chan and Ioannidis (1998)], [Chan and Ioannidis (1999)], and [Johnson (1999)].

[Samet (2006)] provides a textbook coverage of spatial data structures. [Samet (1995)] provides an overview of the large amount of work on spatial index structures. An early description of the quad tree is provided by [Finkel and Bentley (1974)]. [Samet (1990)] and [Samet (1995)] describe numerous variants of quad trees. [Bentley (1975)] describes the k-d tree, and [Robinson (1981)] describes the k-d-B tree. The R-tree was originally presented in [Guttman (1984)]. Extensions of the R-tree are presented by [Sellis et al. (1987)], which describes the R+ tree, and [Beckmann et al. (1990)], which describes the R* tree. These structures provide better worst case complexity guarantees for search than R-trees, but at a higher space cost. [Roussopoulos et al. (1995)] describe algorithms for nearest neighbor search on R-trees.

[Lomet and Salzberg (1989)], and [Lomet and Nawab (2015)] describe index structures for temporal data. [Becker et al. (1996)] present an asymptotically optimal multiversion B-tree structure which can be used to index temporal data.

Bibliography


Credits

The photo of the sailboats in the beginning of the chapter is due to ©Pavel Nesvadba/Shutterstock.