Out From Under the Trees

Christopher Jermaine, Edward Omiecinski, Wai Gen Yee
Georgia Institute of Technology, College of Computing
{jermaine, edwardo, waigen}@cc.gatech.edu

Abstract

We introduce the linear file template, which is a generic data organization suitable for use with many different types of data. The linear file is specifically designed to handle intense database update loads concurrently with processing of analytic queries.

1 Introduction

Hard disk storage capacity continues to increase exponentially, at the same time that there is little change in hard disk seek time. This means that per byte of storage, performing a seek becomes exponentially more expensive over time.

We believe that this trend will make popular tree-based structures (such as the B-Tree and the R-Tree) less and less useful in the long run. Hierarchical structures typically require two seeks per insertion: one to read the leaf page which will receive the insert, and one to write it back again to disk. Since pages are often located randomly on disk, multiple seeks can be required to evaluate large range queries. In environments with intense update/analytic query loads (such as a modern data warehouse) those seeks may be debilitating.

However, while seeks have become more expensive, sequential disk I/O capability has done a much better job keeping up with ever-increasing hard disk capacities. In this work, we introduce the *linear file template* for database storage and access, which aims to remove seeks from such an environment.

2 Basic Linear File Template

The linear file is a generic data organization (like GiST [1]). However, unlike tree-based structures, the linear file uses sequential I/O almost exclusively. Data are organized into *subindexes*, which are storage structures that cover contiguous ranges of key values for numerical data, or "related" key values for more general data.

Since subindexes are essentially ongoing external memory sorts, updates are performed through a set of merge/sort/pack operations that use very few seeks. Since subindexes are considered atomic units in a linear file, they are stored contiguously on disk. Because even large range queries usually access data located entirely within a single subindex, random seeks are often entirely removed from evaluation of large range queries.

3 Instantiating the Linear File

Our prototype of the linear file is a formal template, written in C++, which provides full concurrency control for instantiations of the linear file. In order to customize the template for use with a specific type of data, the user needs to customize several functions for the type of data that the linear file is to manage. The functions to instantiate include *Pack*, *Partition*, *ChooseSubindexForInsertion*, and *IsConsistent*, among others. The majority of the work in processing updates is accomplished by the *Pack* operation, which is called by the linear file template to sort a set of data objects into pages, and then order those pages based on data semantics.

An example instantiation of the linear file is the T2SM storage manager for multidimensional or spatial data [3]. This access method provides many advantages over traditional access methods like the R-Tree and R*-Tree [1].

4 Advantages of the Linear File

Other data organizations have been proposed to handle a workload of intense queries and updates, but the linear file provides two key innovations: the partitioning of the data into subindexes, and the fact that the linear file is a generic template, suitable for many types of data.

The use of many subindexes allows the linear file to be easily customized, and allows the linear file template to use a unique algorithm that associates buffer memory with those subindexes that are actively receiving updates. This greatly speeds update processing by the structure.

References

[1] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, Bernhard Seeger: The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. *SIGMOD Conference 1990*: 322-331

[2] Joseph M. Hellerstein, Jeffrey F. Naughton, Avi Pfeffer: Generalized Search Trees for Database Systems. *VLDB 1995*: 562-573

[3] Christopher Jermaine, Edward Omiecinski, Wai Gen Yee: Maintaining a Large Spatial Database with T2SM. *ACM GIS Conference 2001*: 121-127

Proceedings of the 18th International Conference on Data Engineering (ICDE'02)
1063-6382/02 $17.00 © 2002 **IEEE**