Chapter XIV

An Efficient Concurrency Control Algorithm for High-Dimensional Index Structures

Seok Il Song, Chungju National University, Korea
Jae Soo Yoo, Chungbuk National University, Korea

ABSTRACT
This chapter introduces a concurrency control algorithm based on link-technique for high-dimensional index structures. In high-dimensional index structures, search operations are generally more frequent than insert or delete operations and need to access many more nodes than those in other index structures, such as B*-tree, B-tree, hashing techniques, and so on, due to the properties of queries. This chapter proposed an algorithm that minimizes the delay of search operations in all cases. The proposed algorithm also supports concurrency control on reinsert operations for the high-dimensional index structures employing reinsert operations to improve their performance. The authors hope that this chapter will give helpful information for studying multidimensional index structures and their concurrency control problems to researchers.

INTRODUCTION
In the past couple of decades, multi-dimensional index structures have become the crucial component of multi-dimensional feature vectors-based similarity search systems.
such as GIS, content-based image retrieval systems, multimedia database systems, moving object database systems, and so on. To satisfy the requirements of the modern database applications, various multi-dimensional index structures have been proposed. There are space-partitioning methods like Grid-file (Nievergelt, Hinterberger, & Sevcik, 1984), K-D-B-tree (Robinson, 1981), and Quad-tree (Finkel & Bentley, 1974) that divide the data space along predefined or predetermined lines regardless of data distributions. On the other hand, R-tree (Guttman, 1984), R+-tree (Sellis, Roussopoulos, & Faloutsos, 1987), R*-tree (Beckmann, Kornacker, Schneider, & Seeger, 1990), X-tree (Berchtold, Keim, & Kriegel, 1996), SR-tree (Katayama & Satoh, 1997), M-tree (Ciaccia, Patella, & Zezula, 1997), TV-tree (Lin, Jagadish, & Faloutsos, 1994), and CIR-tree (Yoo et al., 1998) are data-partitioning index structures that divide the data space according to the distribution of data objects inserted or loaded into the tree. In addition, Hybrid-tree (Chakrabarti & Mehrotra, 1999a) is a hybrid approach of data-partitioning and space-partitioning methods; VA-file (Weber, Schek, & Blott, 1998) uses flat-file structure, and that described by Indyk and Motwani (1998) uses hashing techniques.

In order for the multi-dimensional index structures to support the modern database applications, they should be integrated into existing database systems. Even though the integration is an important and practical issue, not much previous work on it exists. To integrate an access method into a data base management system (DBMS), we must consider two problems, namely, concurrency control and recovery. The concurrency control mechanism contains two independent problems. First, techniques must be developed to ensure the consistency of the data structure in the presence of concurrent insertions, deletions, and updates. Several methods that use lock-coupling techniques and link techniques have been proposed for multi-dimensional index structures (Chen & Huang, 1997; Kornacker & Banks, 1995; Kornacker, Mohan & Hellerstein, 1997; Ng & Kamada, 1993; Ravi, Kanth, Serena & Singh, 1998; Song, Kim, & Yoo, 2004). Second, phantom protection methods that protect searchers’ predicates from subsequent insertions, and the rollbacks of deletions before the searchers commit must be developed (Chakrabarti & Mehrotra, 1998; Chakrabarti & Mehrotra, 1999b; Kornacker, Mohan, & Hellerstein, 1997).

In this chapter, we propose a concurrency control method that ensures the consistency of the data structure in the presence of multiple running transactions. Concurrency control methods for multi-dimensional index structures should consider the different properties of multi-dimensional index structures from B+-tree or B-tree. Usually, multi-dimensional index structures used as access methods in the similarity search system have the following properties:

- First, search operations are generally more frequent than insert or delete operations.
- Second, when processing the search operations, they need to access many more nodes than other index structures, such as B*-Tree, B-Tree, hashing techniques, and so on, due to the characteristics of queries (Range Search, K-NN Search).
- Finally, some of them employ forced reinsert operations to reorganize index structures efficiently and to gain high search performance.

We need to add the above properties to the design requirements of the concurrency control algorithm of multi-dimensional index structures.
Related Content

Database High Availability: An Extended Survey
Moh'd A. Radaideh and Hayder Al-Ameed (2009). Selected Readings on Database Technologies and Applications (pp. 21-43).
www.irma-international.org/chapter/database-high-availability/28571/

Experimental Study of a Self-Tuning Algorithm for DBMS Buffer Pools
www.irma-international.org/article/experimental-study-self-tuning-algorithm/3329/

Similarity Search in Time Series
www.irma-international.org/chapter/similarity-search-time-series/20713/

Big Data in Massive Parallel Processing: A Multi-Core Processors Perspective
www.irma-international.org/chapter/big-data-in-massive-parallel-processing/198767/

On the Adaptation of an Agile Information Systems Development Method
www.irma-international.org/article/adaptation-agile-information-systems-development/3340/