GLIP: A Concurrency Control Protocol for Clipping Indexing

N.SWATHI[#1]
(M.Tech – 09491D5807)
QISCET, Ongole,
A.P., India.
nagabhairavaswathi@gmail.com
+91 96000 05683

M.RAVI KANTH[#2]
(M.Tech – 09491D5812)
QISCET, Ongole,
A.P., India.
ravi_kanth_m@yahoo.co.in
+91 99665 56723

P.V.RAVI KANTH[#2]
(M.Tech – 09491D5816)
QISCET, Ongole,
A.P., India.
pvravikanth555@gmail.com
+91 80195 94355

Abstract-The project Concurrency Control Protocol for Clipping Indexing deals with the multidimensional databases. In multidimensional indexing trees, the overlapping of nodes will tend to degrade query performance, as one single point query may need to traverse multiple branches of the tree if the query point is in an overlapped area. Multidimensional databases are beginning to be used in a wide range of applications. To meet this fast-growing demand, the R-tree family is being applied to support fast access to multidimensional data, for which the R+-tree exhibits outstanding search performance. In order to support efficient concurrent access in multiuser environments, concurrency control mechanisms for multidimensional indexing have been proposed. However, these mechanisms cannot be directly applied to the R+-tree because an object in the R+-tree may be indexed in multiple leaves. This paper proposes a concurrency control protocol for R-tree variants with object clipping, namely, Granular Locking for clipping indexing (GLIP). GLIP is the first concurrency control approach specifically designed for the R+-tree and its variants, and it supports efficient concurrent operations with serializable isolation, consistency, and deadlock-free. Experimental tests on both real and synthetic data sets validated the effectiveness and efficiency of the proposed concurrent access framework.

Key Words—Concurrency, indexing methods, spatial databases.

1 INTRODUCTION

In recent years, multidimensional databases have begun to be used for a wide range of applications, including geographical information systems (GIS), computer-aided design (CAD), and computer-aided medical diagnosis applications. As a result of this fast-growing demand for these emerging applications, the development of efficient access methods for multidimensional data has become a crucial aspect of database research. Many indexing structures (e.g., the R-tree [10] family, Generalized Search Trees (GiSTs) [11], grid files [20], and z-ordering [21]) have been proposed to support fast access to multidimensional data in relational databases. Among these indexing structures, the R-tree family has attracted significant attention as the tree structure is regarded as one of the most prominent indexing structures for relational databases. On the other hand, as an important issue related to indexing, concurrency control methods that support concurrent access in traditional databases are no longer adequate for today's multidimensional indexing structures due to the lack of a total order among key values. In order to support concurrency control in R-tree structures, several approaches have been proposed, such as Partial Locking Coupling (PLC) [25], and granular locking approaches for R-trees and GiSTs [4], [5].

In multidimensional indexing trees, the overlapping of nodes will tend to degrade query performance, as one single point query may need to traverse multiple branches of the tree if the query point is in an overlapped area. The R+-tree [23] has been proposed based on modifications of the R-tree to avoid overlaps between regions at the same level, using object clipping to ensure that point queries can follow only one single search path. The R+-tree exhibits better search performance, making it suitable for applications where search is the predominant operation. For these applications, even a marginal improvement in search operations can result in significant benefits. Thus, the increased cost of updates is an acceptable price to pay. However, the R+-tree is not suitable for use with current concurrency control methods because a single object in the R+-tree may be indexed in different leaf nodes. Special considerations are needed to support concurrent queries on R+-trees, while as far as we know, there is no concurrency control approach that specifically supports R+-trees.

Furthermore, there are some limitations in the design of the R+-tree, such as its failure to insert and split nodes in some complex overlap or intersection cases [7]. This will be discussed in Section 2.1.

This paper proposes a concurrency control protocol for R-trees with object clipping, Granular Locking for clipping indexing (GLIP), to provide phantom update protection for the R+-tree and its variants. We also introduce the Zero overlap R+-tree (ZR+-tree), which resolves the limitations of the original R+-tree by eliminating the overlaps of lead nodes. GLIP, together with the ZR+-tree, constitutes an efficient and sound concurrent access model for multi dimensional databases. The major contributions are a follows:

- The concurrency control protocol, GLIP, provides serializable isolation, consistency, and deadlock-free operations for indexing trees with object clipping.
- The proposed multidimensional access method, ZR+-tree, utilizes object clipping, optimized insertion, and reinsert approaches to refine the indexing structure and remove limitations in constructing and updating R+-trees.
- GLIP and the ZR+-tree enable an efficient and sound concurrent framework to be constructed for multi-dimensional databases.
- A set of extensive experiments on both real and synthetic data sets validated the efficiency and effectiveness of the proposed concurrent access framework.

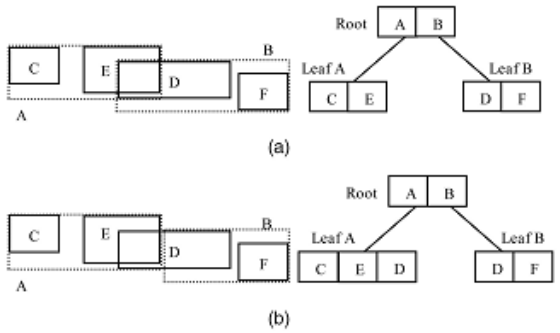

Fig. 1. Examples of R-tree and R+-tree. (a) An R-tree example. (b) An R+-tree example.

This paper is organized as follows: Section 2 reviews concurrency control methods and indexing structures in multidimensional databases. Section 3 introduces the structure and characteristics of the proposed ZR+-tree. The details of the concurrency control approaches are discussed in Section 4. Experimental results for both real and synthetic data are analyzed in Section 5. Final conclusions are drawn and future directions are suggested in Section 6.

2 RELATED RESEARCH AND MOTIVATION

In this section, we review the structure of the R-tree family, discuss some limitations that affect R+-trees, survey major concurrency control algorithms based on B-trees and R-trees, and summarize the challenges inherent in applying concurrency control to R+-trees.

2.1 The R-Tree Family

The R-tree, an extension of the B-tree, is a hierarchical, height-balanced multidimensional indexing structure that guarantees its space utilization is above a certain threshold.

In the R-tree, the root node has between 1 and M entries. Every other node, either leaf or internal node, has between m and M entries ð1 <m¼ <MÞ. The leaf node holds references to the actual data and the Minimum Bounding Rectangle (MBR), which covers all the objects stored in that node. The internal node holds references that point to its children (leaf nodes or the next level of internal nodes), the MBRs corresponding to its children, and its own MBR.

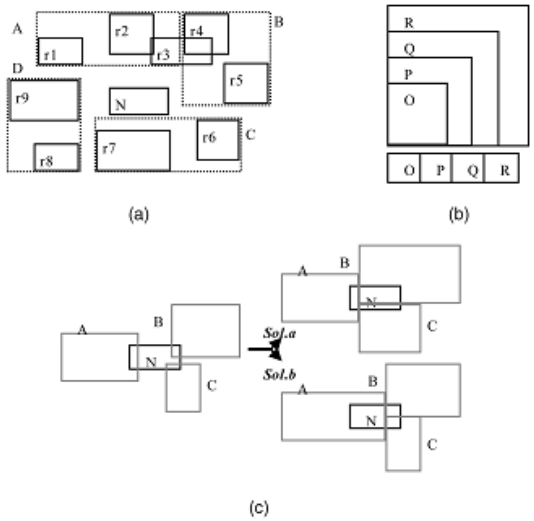

Fig. 2. Limitations in R+-trees. (a) Unable to insert. (b) Unable to split. (c) Different solutions to expand.

The R+-tree was first proposed in [23]. The R+-tree uses a clipping approach to avoid overlap between regions at the same level [7]. As a result of this policy, a point query in the R+-tree corresponds to a single path tree traversal from the root to a single leaf. For search windows that are completely covered by the MBR of a leaf node, the R+-tree guarantees that only a single search path will be traversed.

Examples of the R-tree and R+-tree are given in Fig. 1,whereA and B are leaf nodes, and C,D, E, and F are MBRs of objects. Because objects D and E overlap with each other in the data space, leaf nodes A and B have to overlap in the R-tree in order to contain the objects. In contrast, in the R+-tree, leaf nodes do not have to cover an entire object, so object Dcan be
included in both leaf nodes A and B. As a result, the R+-tree clearly has a better search performance compared to the R-tree. Experimental analyses of the relative performances of R-trees and R+-trees indicate that R+-trees generally perform better for search

operations [8], [12], although this benefit comes at the cost of higher complexity for insertions and deletions. The performance gain for search operations makes the R+-tree ideally suited for large spatial databases where search is the predominant operation.

2.2 Concurrency Controls

Several concurrency control algorithms have been pro- posed to support concurrent operations on multidimensional index structures, and they can be categorized into lock-coupling-based and link-based algorithms. The lock-coupling-based algorithms [6], [19] release the lock on the current node only when the next node to be visited has been locked while processing search operations. During node splitting and MBR updating, these approaches must hold multiple locks on several nodes simultaneously, which may deteriorate the system throughput.

Phantom updating refers to updates that occur before the commitment, in the range of a search (or a following update), and are not reflected in the results of that search (or the following update). Concurrent data access through multidimensional indexes introduces the problem of protecting a query range from phantom updates. The dynamic granular locking approach (DGL) has been proposed to provide phantom update protection in the R-tree [4] and GiST [5].

The DGL method dynamically partitions an embedded space into lockable granules that adapt to the distribution of objects. The leaf nodes and external granules of internal nodes are defined as lockable granules. External granules are additional structures that partition the no covered space in each internal node to provide protection. According to the principles of granular locking, each operation requests locks on sufficient granules such that any two conflicting operations will request conflicting locks on at least one common granule. Although the DGL approach provides phantom update protection for multidimensional access methods and granular locks can be efficiently implemented, the complexity of DGL may impact the degree of concurrency.

2.3 Challenges of Applying Concurrency Control on R+-Trees

Several efficient key value locking protocols to provide phantom update protection in B-trees have been proposed [3], [17], [18]. However, they cannot be directly applied to multidimensional index structures such as R-trees, because for multidimensional data, a total order of the key values on which these protocols are based is undefined.

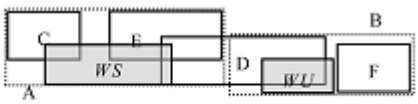

Fig. 3. Example operations for GL/R•tree on an R+•tree.

3 DEFINITION OF GLIP AND ZR+-TREE

Before proceeding to the details of the proposed concurrent access framework, we first define the notations that will be used throughout this paper.

3.1 Terms and Notations

The presence of a standard lock manager [15] is presumed to support conditional and unconditional lock requests, as well as instant, manual, and commit lock durations in GLIP.

A conditional lock request means that the requester will not wait if the lock cannot be granted immediately; an unconditional lock request means that the requester is willing to wait until the lock becomes grantable. Instant duration locks merely test whether a lock is grantable, and no lock is actually placed. Manual duration locks can be explicitly released before the transaction is completed. If they are not released explicitly, they are automatically released at the time of commit or rollback. Commit duration locks are automatically released when the transaction ends. Conventionally, five types of locks, namely, S (shared locks), X (exclusive locks), IX (Intention to set X locks), IS (Intention to set S locks), and SIX (Union of S and IX locks) [6] are used. In the proposed protocol, only S and X locks are used to support concurrent operations with relatively simple main-tenance processes.

TABLE 1
ZR+-Tree Node Attributes

Term	Description
capacity	maximum number of entries in the node
entries	number of entries in the node
mbr	minimum bounding rectangle of the node
level	level of the node in the tree
child$_i$	i_{th} child of the node
rect$_i$	MBR of the i_{th} child of the node
isLeaf	true for a leaf node

3.2 R+-Tree and ZR+-Tree

R+-trees can be viewed as an extension of K-D-B-trees [22] to cover rectangles in addition to points. The original R+-tree has the following properties [23]:

1. A leaf node has one or more entries of the form ðoid;RECTÞ, where oid is an object identifier, and RECT is the Minimum Bounding Rectangle (MBR) of a data object.
2. An internal node has one or more entries of the form ðp;RECTÞ, where p points to an R+-tree leaf or internal node R, such that if R is an internal node, then RECT is the MBR of

all the ðpi;RECTiÞ in R. However, if R is a leaf node, for each ðoidi;RECTiÞ in R, RECTi does not need to be completely enclosed by RECT; each RECTi simply needs to overlap with RECT.

3. For any two entries ðp1;RECT1Þ and ðp2;RECT2Þ in an internal node R, the overlap between RECT1 and RECT2 is zero.
4. The root has at least two children unless it is a leaf.
5. All leaves are at the same level.

Some modifications can bemade to the original R+-tree to make it suitable for the situations mentioned in Section 2.1.

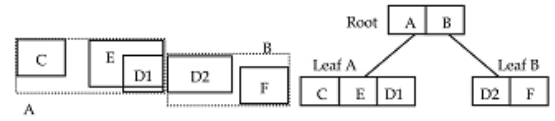

Fig. 4. An example of ZR+-tree for the data in Fig. 1.

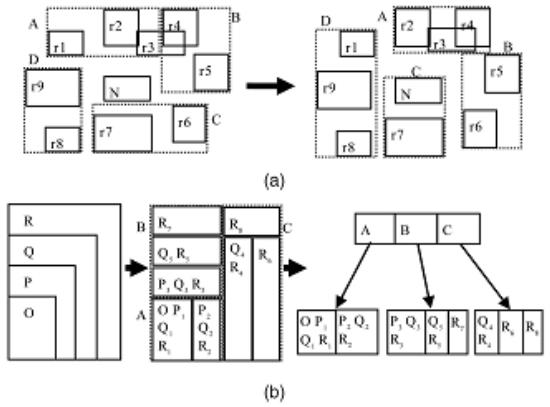

Fig. 5. ZR+-tree solution to the problems in Figs. 2a and 2b. (a) Clustering-based reinsert in ZR+-tree. (b) Object clipping in ZR+-tree.

In addition to the structure evolution, two operation strategies are proposed to improve insertions on the ZR+-tree and refine the indexing tree.

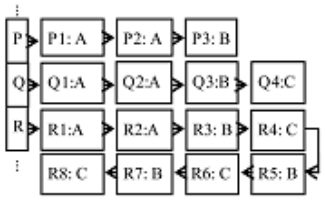

Fig. 6. A clip array for objects in Fig. 5b.

As only one MBR and several ids for each clipped object are stored in this clip array, it is feasible to store the whole array in physical memory. Based on our experiments with real data, on the average, each object is clipped into less than 1.5 segments, so it is reasonable to assume that each clipped object can use two double integers to denote the MBR and 16 integers as eight links (two ids for each link). In this case, 100,000 objects occupy only 4 Mbytes, which is small compared to the memory size available in mainstream computers.

3.3 Lockable Granules

Each leaf node in the ZR+-tree is defined as a lockable granule. We also define an external lockable granule for each ZR+-tree node as the difference between the MBR of the node and the union of the MBRs of its children. In order to reduce the overhead associated with lock maintenance, objects are not individually lockable. The clip array introduced as an auxiliary structure to store the object clipping information does not need to be locked because the locking strategies on leaf nodes ensure the serializability of access for the same object, and updating one object will not affect the other objects. Thus, in the case of the indexing tree in Fig. 3.

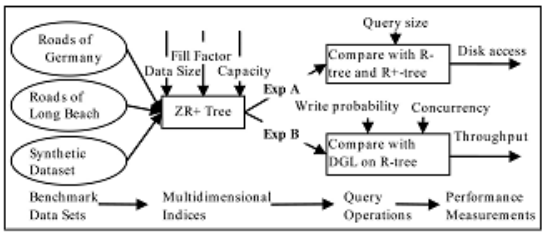

Fig. 8. Experimental design.

4 OPERATIONS WITH GLIP ON ZR+-TREE

To support concurrent spatial operations on the R+-tree and its variants, a granular locking-based concurrency control approach, GLIP, that considers the handling of clipped rectangles is proposed. The approach is designed to meet the following requirements:

1. The following concurrent operations should be supported. Select for a given search window. This is presumed to be the most frequent operation. This operation could result in the selection of a large number of objects, though this may be only a fraction of the total number of objects. Hence, it is desirable to have as few locks as possible that must be requested and released for this operation. Insert a given object. Having redefined the properties of the R+-tree with clipped objects, a new algorithm must be provided for insertion in the ZR+-tree. Delete objects intersected with a search window. Since an object in the ZR+-tree may be clipped and the search window might not select all the fragments of a given object, the algorithm is required to delete all fragments of the selected objects in order to maintain consistency.

2. The locking protocol should ensure serializable isolation for transactions, thus allowing any combination of the above operations performed.

3. The locking protocol should ensure consistency of the ZR+-tree under structure

modifications. When ZR+-tree nodes are merged or split in cases of underflow or overflow, the occasionally inconsistent state should not lead to invalid results.

4. The proposed locking protocol should not lead to additional deadlocks.

Details of the algorithms are provided in the following sections with formal algorithm descriptions.

5 EXPERIMENTS

In order to evaluate the performance of the proposed concurrency control protocol, GLIP, two sets of experiments were conducted as illustrated in Fig. 8. The first set compared the construction and query performance of the ZR+-tree, the R+-tree, and the R-tree, while the other compared the throughput of GLIP on the ZR+-tree and Dynamic Granular Locking on the R-tree. The experimental design consists of four components: selecting/generating benchmark data sets, constructing multidimensional in- dices, executing query operations, and measuring respective performance.

The second set of experiments evaluated the throughput of GLIP on the ZR+-tree by comparing it with dynamic granular locking on the R-tree [4]. The throughputs for the two trees were valuated under different write probabilities and concurrency levels.

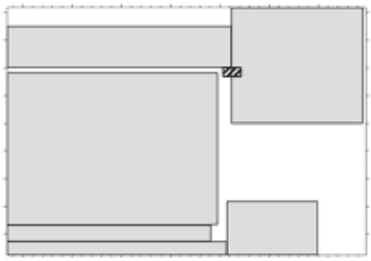

Fig. 9. Construction failure in R+-tree on Long Beach data.

6 CONCLUSION

This paper proposes a new concurrency control protocol, GLIP, with an improved spatial indexing approach, the ZR+-tree. GLIP is the first concurrency control mechanism designed specifically for the R+-tree and its variants. It assures serializable isolation, consistency, and deadlock free for indexing trees with object clipping. The ZR+-tree segments the objects to ensure every fragment is fully covered by a leaf node. This clipping-object design provides better indexing structure. Furthermore, several structural limitations of the R+-tree are overcome in the ZR+-tree by the use of a no overlap clipping and a clustering-based reinsert procedure. Experiments on tree construction, query, and concurrent execution were conducted on both real and synthetic data sets, and the results validated the soundness and comprehensive nature of the new design. In particular, the GLIP and the ZR+-tree excel at range queries in search-dominant applications. Extending GLIP and the ZR+-tree to perform spatial joins, KNN-queries, and range aggregation offer further attractive possibilities.

REFERENCES

[1] M. Abdelguerfi, J. Givaudan, K. Shaw, and R. Ladner, “The 2-3TR- Tree, a Trajectory-Oriented Index Structure for Fully Evolving Valid-Time Spatio-Temporal Datasets,” Proc. 10th ACM Int’l Symp. Advances in Geographic Information System (ACMGIS ’02), pp. 29-34, 2002.

[2] N. Beckmann, H.P. Kriegel, R. Schneider, and B. Seeger, “The R -Tree: An Efficient and Robust Access Method for Points and Rectangles,” Proc. ACM SIGMOD ’90, pp. 322-331, 1990.

[3] A. Biliris, “Operation Specific Locking in B-trees,” Proc. Sixth Int’l Conf. Principles of Database Systems (PODS ’87), pp. 159-169, 1987.

[4] K. Chakrabarti and S. Mehrotra, “Dynamic Granular Locking Approach to Phantom Protection in R-Trees,” Proc. 14th IEEE Int’l Conf. Data Eng. (ICDE ’98), pp. 446-454, 1998.

[5] K. Chakrabarti and S. Mehrotra, “Efficient Concurrency Control in Multi-Dimensional Access Methods,” Proc. ACM SIGMOD ’99, pp. 25-36, 1999.

[6] J.K. Chen, Y.F. Huang, and Y.H. Chin, “A Study of Concurrent Operations on R-Trees,” Information Sciences, vol. 98, nos. 1-4, pp. 263-300, May 1997.

[7] V. Gaede and O. Gunther, “Multidimensional Access Methods,” ACM Computing Surveys, vol. 30, no. 2, pp. 170-231, June 1998.

[8] D. Greene, “An Implementation and Performance Analysis of Spatial Data AccessMethods,” Proc. Fifth IEEE Int’l Conf. Data Eng. (ICDE ’89), pp. 606-615, 1989.

[9] S. Guha, R. Rastogi, and K. Shim, “CURE: An Efficient Clustering Algorithm for Large Databases,” Proc. ACM
SIGMOD ’98, pp. 73-84, 1998.

[10] A. Guttman, “R-Trees: A Dynamic Index Structure for Spatial Searching,” Proc. ACM SIGMOD ’84, pp. 47-57, 1984.