Comparing Z-order B-tree and R-tree Family for Spatial Query Processing

C. T. Lu, S. Shekhar
Computer Science Department, University of Minnesota
200 Union Street SE, Minneapolis, MN-55455
[*ctlu*, *shekhar*]@cs.umn.edu TEL:(612) 6248307 FAX:(612)6250572
http://www.cs.umn.edu/Research/shashi-group

December 3, 2000

Abstract

Spatial objects in multidimensional space can be managed by one-dimensional access method, point access method, and spatial access method. We give a brief survey of one-dimensional access method Z-order B-tree compared with spatial access method R-tree in the original space and transformed space, and discuss the performance issues for spatial query processing.

1 Summary of results

We summary the main results related to Z-order B-tree in the original space Z-order and transformed space.

1.1 Original space

The spatial queries [11] include point queries, range queries, aggregate queries, e.g., nearest-neighbor queries, and spatial join queries. Some examples of the predicate θ used in the spatial join queries are intersect, contain, is_closed_by, distance, northwest, adjacent, meet, and overlap. Z-order B-tree can be used to support all of these queries, and is is well explored in literature.

In the PROBE project, Orenstein [8] constructed Z-order B-tree as the access method to process spatial queries, such as point and range query using spatial predicates of overlap. They showed that the approximate geometry, i.e., grid representation of spatial data, can be supported with very minor modification of the existing DBMS implementation. The formal model for spatial query processing using Z-order B-trees was proposed by Volker et al. [4] The spatial query operations they defined in the model include intersection, enclosure, containment, distance, adjacency, northeast, nearest, and farthest. In their GIS GODOT database system, they chose Z-order B-tree rather than multidimensional index method since the query optimizer of the underlying database system cannot take advantage of the spatial index and does not support concurrency and recovery for spatial index. They proposed the concept of the ϕ function which enables the optimization and processing of user-defined functions. They showed that the Z-order B-tree technique is general enough to be used in combination with current database systems. Recent work includes star-join in data warehouse using Z-order B-tree [7].

To handle nearest-neighbor query using Z-order B-tree, first, we compute the Z-value of the query point P_i and search the data point P_j with the closest Z-value from the B-tree. Then, we compute the distance D between P_i and P_j and issue a range query centered at P_i with radius D. We check all the retrieved points and return the one with the shortest distance to the query point.

One disadvantage using Z-order B-tree is that the index partitions can not be joined if the grid structure is not compatible, and one of the indexes has to be recomputed for join processing [13]. Another problem of Z-order as space filling curve is the long diagonal jumps, where the consecutive Z-value points connecting these jumps are far apart in X-Y space. The spatial clustering of Z-ordering can be improved by using the Hilbert curve [2, 6]. However, algorithms to process range query and spatial join using Hilbert curve need to be worked out. Recursive definition of Hilbert curve may pose some challenges here. Distance between two points in original space are often different from those space-filling curve for both Z-order and Hilbert curve.

1.2 Transformed space

Extended objects, e.g., minimum bounding boxes for polygons, are often transformed into points in a higher dimensional space. These points can thus be managed by multi-dimensional point access methods or one-dimensional access methods which use space filling curve to define order. Range queries on the original space often translated to a convex region in the higher dimensional space.

Orenstein evaluated the spatial query within original-space and transformed-space using zkd b+-tree [9]. The transformed space searching is simple and can be supported on top of any point access method. He compared the performance of one-to-many and many-to-many overlap queries in the original space and transformed space. Their experimental results can be used as a guidance while choosing spatial query processing strategies.

Yu et al. investigated the performance of two generation of object-transformation schemes in both low- and high-dimensional spaces [14]. The first-generation object transformation scheme simply uses endpoints to represent the minimum bounding rectangle of an object, the second-generation object transformation scheme tries to restrict the search region in the transformed space by considering the properties of the actual data set. The spatial queries they used in the experiment are equal, covers, covered_by and not_disjoint. They used a variant of kdB-tree as the underlying point access method in the transformed space and compared the I/O cost with R*-tree, topological R*-tree and QSF tree.

The data distribution is skewed in the transformed space, so the underlying point access methods must adapt to this non-uniform data distribution. The kdB-tree is a perfectly balanced tree which adapts well to the distribution of the data. The LSD tree adapts well to the non-uniform distribution of the data and can be used in connection with the transformation technique.

This transformation approach has some disadvantages. First, the formulation of range queries in the transformed space is usually much more complicated than in the original space. Second, the distribution of points in the transformed space may be highly non-uniform. Third, it is difficult to perform nearest-neighbor and spatial-join queries, since the distance relation in the original space are not reflected in the transformed space. The distance between two close objects in the original space may be arbitrarily far apart from each other in the transformed space [13]. Special transformation techniques and split strategies have to be used to overcome these problems [2, 5, 10].

2 Discussion

2.1 Do we need transformations on the dimensions?

Mapping to higher-dimensional space is not needed if one is willing to decompose or replicate extended objects into a collection of cells compatible with the resolution of Z-order grid. See [3, 4] for details. R-tree, R+tree, and R*tree do the same (replicate or decompose) for large objects.

2.2 Is it sufficient to implement Z-order B-trees only, Or are R-trees truly required? What optimizer smarts are required for Z-order B-trees? What overall performance loss will we need to tolerate?

Z-order B-tree can be used to process all spatial queries, i.e., R-tree is not necessary. Needed optimizer smarts include a preprocessor to transform spatial queries (e.g., MBR range queries) into a collection of queries (e.g., interval range queries) onto B-tree. See [3, 8] for details. Also, a post-processing is needed to assemble results of B-tree interval range queries to answer original spatial query.

Performance relative to R-tree: There should not be any loss for point queries. Performance differences for range query and spatial join will be due to clustering efficiency and duplicate elimination (to assemble bits of polygon in original space). R-tree primary index will provide better clustering, i.e., higher P_r [O_i and spatial neighbor(O_i) being in the same disk page] relative to use Z-order B-tree as the primary index assuming same blocking factor. This gap can be reduced via many methods:

- Smaller index records: Since Z-values are smaller than rectangles and can be compressed easily (relative to rectangle), there is plenty of room to reduce index record size and improve blocking factor.

- Spatial clustering: Clustering records into pages via a spatial method stronger than Z-order will reduce I/O. A simple approach is to separate clustering of records into pages from indexing. This approach will use Z-order B-tree as a secondary index while using a spatial clustering algorithm as data file. A naive spatial clustering algorithm is regular grid based tiling, which is often used in GIS products. Main problem with this method is potentially large variation in number of object across tiles which will lead low utilization of some of the pages. Better spatial clustering algorithm can be based on smarter algorithm, e.g., graph-partitioning [12] or heuristics e.g., sharing disk-pages across sparsely populated tiles. An alternate approach would index tiles via B-tree and maintain a list of object-id for each tile in the leaves (first level data). This is similar to indexing column where values are not unique.

- Large page sizes: Large page sizes can improve blocking factor and indirectly reduce number of seeks/latency to be paid.

- Buffer space : Additional buffers are useful in spatial-joins to reduce I/Os as inner table.

Lastly, we should look at the total cost of GIS operations to compare spatial indexing methods, which primarily affect the filter step but do not impact the cost of refinement step. Since cost of refinement step dominates the performance, the difference of spatial access methods for filter step is less significant in the overall picture. The spatial join refinement step has been studied with Z-order B-tree [1], but not with R-tree.

References

[1] David J. Abel, Volker Gaede, Robert A. Power, and Xiaofang Zhou. Resequencing and clustering to improve the performance of spatial joins. In *http://www.wiwi.hu-berlin.de/ gaede/vg.pub.html*, 1996.

[2] Christos Faloutsos and Yi Rong. Dot: A spatial access method using fractals. In *Proceedings of the Seventh International Conference on Data Engineering, April 8-12, 1991, Kobe, Japan*, pages 152–159.

[3] Volker Gaede. Optimal redundancy in spatial database system. In *Proc. 4th Int. Symposium on Spatial Databases (SSD'95), http://www.wiwi.hu-berlin.de/ gaede/vg.pub.html*, 1995.

[4] Volker Gaede and Wolf-Fritz Riekert. Query evaluation in the object-oriented gis godot, iss-36, december 96. In *http://www.wiwi.hu-berlin.de/ gaede/vg.pub.html*.

[5] Andreas Henrich, Hans-Werner Six, and Peter Widmayer. The lsd tree: Spatial access to multidimensional point and nonpoint objects. In *Proceedings of the Fifteenth International Conference on Very Large Data Bases, August 22-25, 1989, Amsterdam, The Netherlands*, pages 45–53.

[6] H. V. Jagadish. Linear clustering of objects with multiple atributes. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Atlantic City, NJ, May 23-25, 1990*, pages 332–342.

[7] Volker Markl and Rudolf Bayer. Processing relational olap queries with ub-trees and multidimensional hierarchical clustering. In *Proceedings of the Second Intl. Workshop on Design and Management of Data Warehouses, DMDW 2000.*

[8] J. Orenstein. Spatial query processing in an object-oriented database system. In *Proceedings of the 1986 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 28-30, 1986*, pages 326–336.

[9] Jack Orenstein. A comparison of spatial query processing techniques for native and parameter spaces. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 19(2):343–352, 1990.

[10] Bernd-Uwe Pagel, Hans-Werner Six, Heinrich Toben, and Peter Widmayer. Towards an analysis of range query performance in spatial data structures. In *Proceedings of the Twelfth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 25-28, 1993, Washington, DC*, pages 214–221.

[11] S. Shekhar, S. Chawla, S. Ravada, A. Fetterer, X. Liu, and C.T. Lu. Spatial databases: Accomplishments and research needs. *IEEE Transactions on Knowledge and Data Engineering(TKDE)*, 11(1):45–55, 1999.

[12] S. Shekhar and D. R. Liu. Ccam: A connectivity-clustered access method for networks and network computations. In *IEEE Trans. on Knowledge and Data Engineering, Vol. 9, No. 1, Jan. 1997.*

[13] V. Gaede V. and O. Gunther. Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231, 1998.

[14] B. Yu and R. Orlandic. Object and query transformation: Supporting multi-dimensional queries through code reuse. In *Proceedings of the 2000 ACM CIKM International Conference on Information and Knowledge Management*. ACM.