A Survey of R-Trees

Aaron Gorenstein
University of Wisconsin - Madison
agorenst@cs.wisc.edu

Rebecca Lam
University of Wisconsin - Madison
rjlam@cs.wisc.edu

March 13, 2013

1 Introduction

What was the motivation for this survey? What was the point of the original paper?

An overview of the rest of the paper here.

2 Overview of R-Trees

To solve the problem of performing efficient searchs on spatial data, Guttman proposed the R-tree, which inspired a variety of different variations analagous to the family of B-trees. In Section 2.1 we outline the original R-tree paper, and in Section 2.2 we examine the variants and draw appropriate comparisons.

2.1 R-Trees

In 1984, Guttman first proposed the idea of using minimum bounding rectangles (MBR) as a way to restrict the search space during a lookup for spatial data.

2.1.1 Search

2.1.2 Insert

2.1.3 Delete

2.2 R-Tree Variants

Much like its cousin, the B-tree, the R-tree has a few main variants such as the R^+-tree and the R^*-tree, which we discuss in the following sections.

2.2.1 R+-Trees

2.2.2 R*-Trees

3 Implementation Challenges

While we have seen a complete description of the behavior of the R-tree, in a sense our description is far from complete. In this and the next section we will place R-trees in broader contexts.

As the R-tree stores spatial data, implicitly it is designed to support queries based on the spatial relationships of the data. Thus, in this section we discuss a variety of spatial queries the R-tree supports, and to what degree of efficiency. In moving towards the natural goal of efficient R-trees, we will discuss various successful strategies in optimizing the core data structure. As often happens, different settings lead to further specializations and design decisions, so a discussion on the tweaks and other optimizations of R-trees will follow.

In the next section we explore in particular the optimality and formal analysis of R-trees. Spatial data is much slipperier than ordered data, and developing a model to explain the behavior of R-trees is a natural motivation after we have seen its various forms. On this note we will conclude our discussion on R-tree's state-of-the-art.

3.1 Basic Queries, Data Model

The prototypical R-tree is introduced as storing axis-aligned rectangles: basic spatial data. As R-trees store spatial data according to their spatial relationships, it follows that R-trees support spatial queries. Just what is a spatial query? The quintessential example is the

"intersect-range" query, where the user supplies an arbitrary, axis-aligned rectangle, and the R-tree returns a list of all rectangles with a non-empty intersection. However, a surprising zoo of basic queries have appeared in the literature, and we list them here for better understanding of what may want our R-tree to support. [SEE [**?**]., and spatial join paper, and [1]].

3.2 General Optimizations

Hopefully it is clear from these diverse applications that it is not enough to pour all our efforts into optimizing for one query. It seems difficult to improve an R-tree design over all these objectives, but some general successes have been made: [R*, Front-loading, node splitting, space utilization (grafting), and more.]

3.3 Data-Dependent Optimizations

As one considers the nature of the stored data and the nature of the queries, we are able to specialize more. [R+, static data, Hilbert gives data ordering (b-tree linearity), using not just distribution but fractal stuff]

3.4 Extensions to the R-Tree

Polygon, X-tree, maybe R+ tree again. DEFINITELY CONCURRENCY.

4 Database Related Challenges

The flexibility of the R-tree has no doubt contributed to its success. [But what about understanding how it actually works? Develop a theory of understanding how it behaves. Measures of optimality. And more.]

5 Modern Applications

cool topics like parallelism, concurrency, emerging applications

6 Conclusion

References

[1] Hanan Samet. Spatial data structures. In Won Kim, editor, *Modern Database Systems*, pages 361–385. ACM Press and Addison-Wesley, 1995.