

Erweiterungen des R-Baums für räumliche Datenbankanfragen

Der R*-Baum

Patrick Schulz & Simon Hötten

Seminar Geodatenbanken
Dozent: Prof. Dr.-Ing. Jan-Henrik Haunert
Institut für Geoinformatik und Fernerkundung
Universität Osnabrück
Sommersemester 2015

Schlüsselwörter: Geodatenbanken, R*, Spatial Access, Index, R-Tree

1 Motivation

Herkömmliche eindimensionale Indexstrukturen bieten keine Möglichkeiten, mehrdimensionale räumliche Daten effizient zu durchsuchen. Die Reduzierung auf Punkte, um Objekte mit Point access methods (PAM) abzufragen, ist mit gewissen Einbußen möglich, aber insbesondere für komplexere Anfragen unzureichend. Der 1984 von Guttman entwickelte R-Baum (Guttman, 1984) versucht dieses Problem zu lösen, in dem der Index direkt auf den räumlichen Eigenschaften basiert. Mittlerweile existieren unzählige Varianten und Verwandte des R-Baums, dessen Einsatzgebiet weit über die klassische Geoinformatik hinaus geht.

Eine dieser Varianten ist der R*-Baum, welcher die (teils unbegründeten) Annahmen in der ursprünglichen Veröffentlichung hinterfragt und so die Datenstruktur weiter optimiert. Im Folgenden gehen wir auf die Verfahren und Eigenheiten des regulären R-, als auch des R*-Baums ein, stellen allgemeine Optimierungskriterien auf und schließen mit einem Vergleich.

Im Prinzip ist der R*-Baum für n-dimensionale Daten geeignet. Diese Arbeit beschränkt sich allerdings auf Geodaten, insbesondere sind alle Beispiele im zwei-dimensionalen Raum. Hier liegt auch das Haupteinsatzgebiet von R*-Bäumen. Für höher-dimensionale Daten sind andere Indexstrukturen, wie der X-Baum, zu bevorzugen (vgl. Berchtold u. a., 1996, S. 28-29).

2 Prinzipien eines R-Baums

Der R-Baum ist eine räumliche Indexstruktur für die effiziente Bereichsabfragen von Geometrien im zweidimensionalen Raum. Der R-Baum teilt in jeder Ebene seiner Struktur die beinhalteten Geometrien in Partitionen auf. Die Gesamtheit der Geometrien einer Partitionen werden durch minimal umschließende Rechtecke (kurz MBR) repräsentiert. Innerhalb einer Partition werden die Geometrien in

weitere Partitionen unterteilt, bis die Anzahl der Geometrien den Schwellwert M innerhalb einer Partition nicht mehr ueberschreitet. Der Vorteil von dieser Strukturierung sorgt dafuer, dass bei Bereichabfragen - zB. Was befindet sich in der Umgebung von Polygon p ? - nicht zwingend alle Geometrien in der Ebene betrachtet werden muessen: Wenn in Abbildung 1

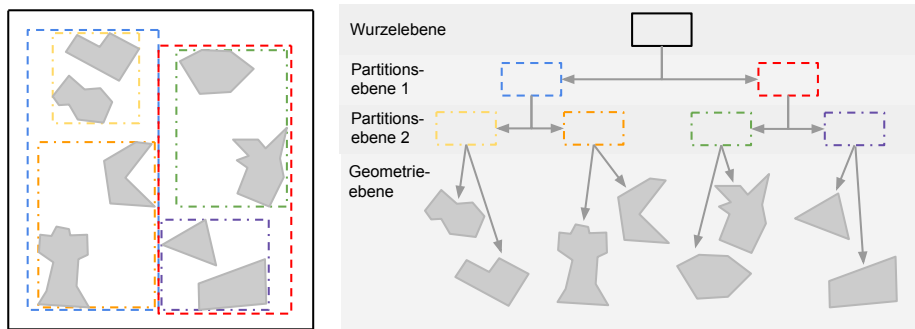


Abb. 1. Beispiel fuer einen R-Baum. ($M=3$, $m=2$)

3 Optimierungskriterien

Bei dem herkömmlichen R-Baum wird, sowohl beim Hinzufügen neuer Elemente als auch beim Split, lediglich die Fläche der umschließenden Rechtecke minimiert (vgl. Guttman, 1984, S. 50-51). Einige der daraus resultierenden Probleme wurden bereits im vorherigen Abschnitt dargelegt. Im Folgenden werden weitere mögliche Optimierungen und ihre Wechselwirkungen aufgeführt. Ein R-Baum muss mit unterschiedlichen Geometrien und Anfragen umgehen können, daher wirken sich einige Kriterien in einigen Situationen stärker aus als andere.

Flächenausnutzung maximieren

Die Fläche, welche von dem umschließenden Rechteck, aber nicht von den in ihm enthaltenen Rechtecken, überdeckt wird, soll minimiert werden. Es soll also möglichst wenig Platz „verschwendet“ werden. (vgl. Beckmann u. a., 1990, S. 323)

Überlappung minimieren

Die Überlappung der umschließenden Rechtecke soll minimiert werden. Dadurch müssen ebenfalls weniger Pfade im Baum traversiert werden. Liegt ein angefragter Punkt beispielsweise in einer Region, in der sich viele Rechtecke überschneiden, müssen alle Möglichkeiten weiter verfolgt werden, was zu erhöhtem Rechenaufwand führt.

Summe der Kantenlänge minimieren

Die Summe der Kantenlänge der Verzeichnisrechtecke („margin“) soll möglichst klein sein. Quadrate werden also bevorzugt. Da Quadrate auf den jeweils höheren Ebenen im Baum besser zusammengefasst werden können, reduziert sich so die benötigte Fläche. Außerdem profitieren Anfragen mit großen, quadratischen Elementen von dieser Optimierung. (vgl. ebd., S. 323)

Speichernutzung maximieren

Eine geringe Höhe des Baumes wirkt sich positiv auf die Kosten einer Abfrage aus. Das kann durch eine möglichst gleichmäßige Verteilung der Blattknoten erreicht werden. Insbesondere für große Abfragerechtecke ist dies relevant, da hier, auch abgesehen von den ersten drei genannten Optimierungen, mehrere Pfade traversiert werden müssen. (vgl. ebd., S. 323-324)

Wechselwirkungen

Um die Flächenausnutzung zu maximieren und die Überlappung zu minimieren, bedarf es einer größeren Freiheit bei der Wahl der Formen und der Anzahl an Rechtecken pro Knoten. Die Kriterien stehen also in Konkurrenz mit einer geringen Kantenlänge und hohen Speicherausnutzung. Auf der anderen Seite können quadratischere Rechtecke besser zusammengefasst werden, was sich wiederum positiv auf die Speichernutzung auswirkt. (vgl. ebd., S. 323-324)

4 Der R*-Baum**5 Fazit**

Zunächst lässt sich festhalten, dass der R*-Baum alle vorgestellten Optimierungskriterien berücksichtigt. Das hat einen etwas erhöhten Implementierungsaufwand gegenüber herkömmlichen R-Bäumen zur Folge.

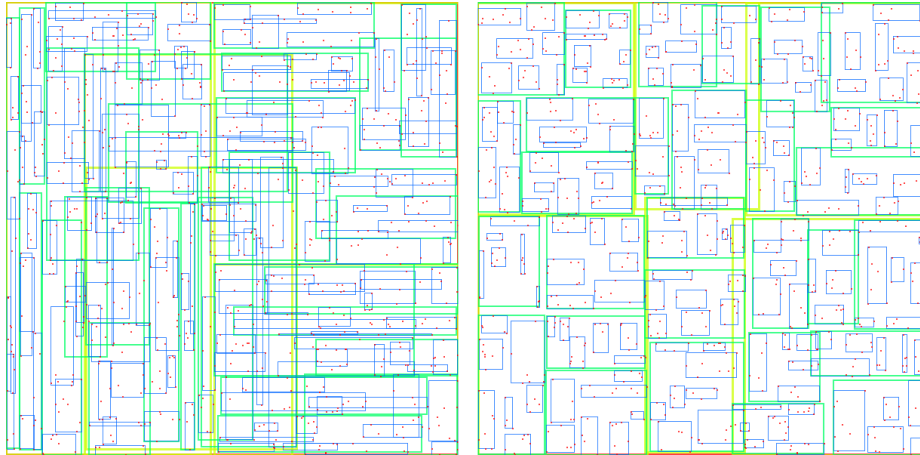


Abb. 2. Quadratischer- (links) und R*-Split (rechts) im Vergleich. (Bildquelle: <https://github.com/davidmoten/rtree>)

Der R*-Baum betreibt sehr hohen Aufwand beim Hinzufügen und Löschen von Elementen, um eine gute Struktur zu bewahren. Abbildung 2 zeigt den gleichen Datensatz (mit gleicher Einfügereihenfolge), aber unterschiedlichen Split-Verfahren. Das Beispiel zeigt, wie positiv sich der Mehraufwand auswirkt. Mit R*-Split überschneiden sich die Verzeichnisrechtecke deutlich weniger, was schnellere Abfragen ermöglicht.

Benchmarks

Weiterentwicklungen

Die Effizienz des R*-Baums nimmt ab fünf Dimensionen rapide ab (vgl Kriegel u. a., 2008, S. 29). Um auch höher dimensional Daten gerecht zu werden, existieren daher zahlreiche Weiterentwicklung. Dazu gehört, wie eingangs erwähnt, der X-Baum, welcher darauf ausgelegt ist auch in höheren Dimensionen Überlappungen zu vermeiden (Berchtold u. a., 1996, vgl.). Andere Indizes bilden Näherungen der tatsächlichen Daten und führen Anfragen zunächst auf diesen aus („*Vector Approximation*“, siehe Gibas und Ferhatosmanoglu, 2008 oder Daoudi u. a., 2008).

Anhang

Abkürzungsverzeichnis

m	Variable: minimale Anzahl der Geometrien in einer Partition
M	Variable: maximale Anzahl der Geometrien in einer Partition
MBR	Minimum bounding Rectangle
SAM	Spatial access methods
PAM	Point access methods

Literatur

- Beckmann, Norbert, Hans-Peter Kriegel, Ralf Schneider und Bernhard Seeger (1990). „The R*-tree: An Efficient and Robust Access Method for Points and Rectangles“. In: *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*. SIGMOD '90. Atlantic City, New Jersey, USA: ACM, S. 322–331. DOI: 10.1145/93597.98741 (siehe S. 2 f.).
- Berchtold, Stefan, Daniel A. Keim und Hans-Peter Kriegel (1996). „The X-tree: An Index Structure for High-Dimensional Data“. In: *Proceedings of the 22nd VLDB Conference*. Mumbai, India, S. 28–39 (siehe S. 1, 4).
- Daoudi, I., S.E. Ouatik, A. El Kharraz, K. Idrissi und D. Aboutajdine (2008). „Vector Approximation based Indexing for High-Dimensional Multimedia Databases“. In: *Engineering Letters* 16.2 (siehe S. 4).
- Gibas, MICHAEL und Hakan Ferhatosmanoglu (2008). „High Dimensional Indexing“. In: *Encyclopedia of GIS*. Springer US, S. 502–507. DOI: 10.1007/978-0-387-35973-1_1148 (siehe S. 4).
- Guttman, Antonin (1984). „R-trees: a dynamic index structure for spatial searching“. In: *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*. SIGMOD '84. New York, NY, USA: ACM, S. 47–57. DOI: 10.1145/602259.602266 (siehe S. 1 f.).
- Kriegel, Hans-Peter, Peter Kunath und Matthias Renz (2008). „R*-Tree“. In: *Encyclopedia of GIS*. Springer US, S. 987–992. DOI: 10.1007/978-0-387-35973-1_1148 (siehe S. 4).