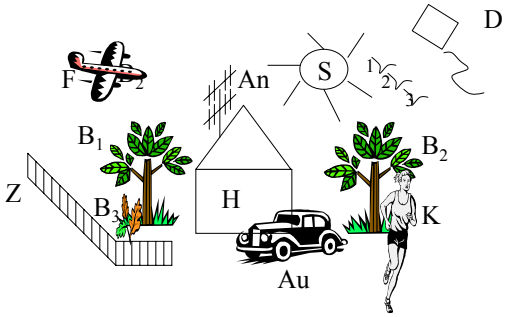
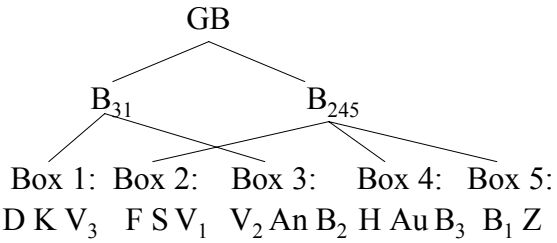
Kap. 12 R-Bäume



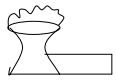


1

Problem: Dynamische Datenstrukturen
für Raum-Daten (spatia/data).
Speicherung u. Manipulation: Einfügen,
Suchen, Löschen

Anwendungen: CAD, Geo-Datenbanken,
Medizin, Gen-Technik, ..., Umweltschutz

Anfrage-Arten: Proximity Queries, z.B.
alle Objekte, die (ganz oder teilweise)
in geg. Raumausschnitt liegen.

Nearest Neighbor 

A. Guttman: A dynamic Index Structure
for Spatial Searching. Proc. ACM SIGMOD
Intl. Conference on Management of
Data, 1984, 47-57

2

Grundidee:

1. Bounding Boxes für geometrische Figuren
2. Bounding Boxes um Boxes,
   mindestens m, höchstens M
3. Höhenbalanzierung durchSpaltung u. Wachstum wie bei
   B-Bäumen. In Blättern Verweise auf Boxes, die Datenobjekte mit
   Detail-Info enthalten.

3

Objekt-Darstellung: n-dimensionales
Rechteck $J = (I_0, I_1, ..., I_{n-1})$
mit Intervall I_i für i-te Dimension
$I_i = [a,b]$ plus Verweis auf Box.

auf Blatt : (J, Box-Verweis)
wobei 1 Objekt in Box ist,

interner Knoten: (J, Kind-Verweis)
wobei Kind eine Box voller Boxes ist mit
m bis M Boxes.

Jeweils kleinste Box, die die
inneren Boxes umfaßt ("bounding box")

4

Definition R-Tree: hat folgende Eigenschaften:

(1) Jedes Blatt hat m bis M Einträge
(es sei denn Blatt = Wurzel)

(2) Für jeden Index-Eintrag (J, tup-id)
in einem Blatt ist J das kleinsteRechteck, welches das Objekt mit Identifikation tup-id umfaßt.

(3) Jeder innere Knoten (bis auf Wurzel)
hat zwischen m und M Kinder,
d.h. $(J_1,k_1), (J_2,k_2), ..., (J_l,k_l)$ 

mit $m \leq l \leq M$

(4) Für jeden Eintrag (J, Kind-Verweis)
eines inneren Knotens ist J das kleinste Rechteck, das die Rechtecke des Kindknotens umfaßt (Anpassung durch Dehnung/Schrumpfung bei Einfügung/Löschung

(5) Wurzel hat mindestens 2 Kinder, außer sie ist Blatt

(6) Alle Blätter sind auf der gleichen Baumebene.

5

Höhe: für N Objekte : $\leq \lceil \log_m N \rceil$

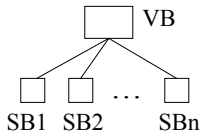
Speicher-Nutzung : mindestens $\frac{m}{M}$ (bis auf Wurzel)

Baum-Algorithmen:

Suchen: alle Objekte in einem Suchbereich S,
z.B. Rechteck, Kreis,...
verfolge **alle** Pfade mit Eigenschaft
$J \cap S \neq \emptyset$

Einfügen: wie bei B-Bäumen. Bei Suche nach Einfüge-Stelle verfolge Pfad, der die kleinsten Intervall-Anpassungen erfordert.

Box-Auswahl beim Einfügen:



{einzufügendes Objekt
mit box OB paßt in VB}

mit $m \leq n \leq M$

6

Fall 1 : OB paßt in genau 1 SB_i

Fall 2 : OB paßt in mehrere SB_j

Auswahlstrategie, abhängig von Box-Größe und Füllgrad:

- kleine Box : später präzisere Suche
- "leere" Box : weniger Gefahr für Split

Fall 3 : OB paßt in keine SB:

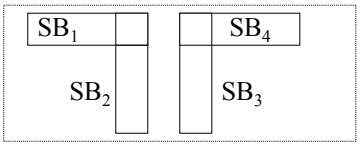
ein SB_k muß vergrößert werden,
um OB aufzunehmen.
Auswahlstrategien für SB_k z.B.:

- minimale Vergrößerung
- minimale Überlappung mit anderen SB_j

Ziel: beim Suchen möglichst wenig Parallelpfade zu verfolgen!

7

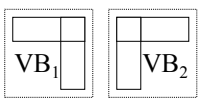
Split-Problem: Beispiel:



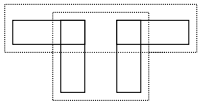
z.B.verursacht
durch Blatt-Split

VB_{alt} : 4 Einträge, M=3

Problem: Box-Inhalt von VB_{alt} in
2 Mengen aufteilen, mit Boxen umgeben



schlecht!



besser?

Wichtig: Eine Box wird immer nur in **eine** Vaterbox gesteckt,
auch wenn sie in mehrere paßt.

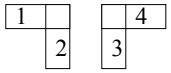
8

Ziel: minimiere bei Split
$|VB_1| + |VB_2|$

Optimale Lösung: generiere alle Split-Möglichkeiten (2^M) und wähle beste aus. Exponentieller Algorithmus

Quadratischer Alg:

1. Wähle 2 Boxen aus, die **nicht** in 1 Gruppe kommen sollten,



z.B. 1 und 3
oder 2 und 4

2. Stecke ausgewählte 2 in verschiedene Gruppen (Boxes) : VB_1, VB_2
3. Noch zu verteilen: SB_1, SB_2,..., SB_k

d_i^1 := Vergrößerung von VB_1, um Sb_i aufzunehmen

d_i^2 := Vergrößerung von VB_2, um Sb_i aufzunehmen

j := $ind_i max\ |d_i^1 - d_i^2|$

9

SB_j hat maximale "Affinität" zu einer VB,
stecke SB_j in VB_1 wenn $d_j^1 < d_j^2$
in VB_2 sonst

Hinw : Bei jedem Split werden genau M+1 Boxen verteilt, i.e. $O(M^2)$

Linearer Alg: Wähle 2 SB_i, SB_j für verschiedene Vaterboxen VB_1, VB_2
verteile restliche SB_k in beliebiger Reihenfolge mit Kriterium d_k^1, d_k^2.

Frage: Gute Heuristiken mit linearer Laufzeit für Split und anschließende Verteilung?
Heuristiken mit O(M log M) ?

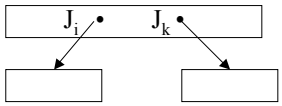
Hinw: Splits sind selten, $O(M^2)$ doch tolerierbar?

Performance Messungen: angeblich keine gravierenden Unterschiede

10

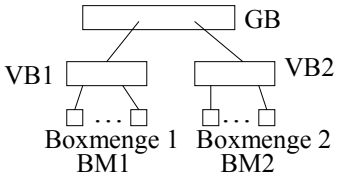
Löschalgorithmus:

- Vaterboxen "schrumpfen"
- merge bei Blättern : im Prinzip wie bei B-Bäumen:
  Autoren empfehlen Löschung und Neueinführung (reinsertion)
- merge von inneren Knoten:



J_j, J_k durch J_j' ersetzen, d.h. neueBounding Box bestimmen

- Über- oder Unterlauf:



1. BM1 u. BM2 neu auf VB1, VB2 verteilen
2. VB1, VB2 eng anpassen d.h. GB fortschreiben

11

Randbedingung: VB1, VB2 haben je ≥ m Söhne