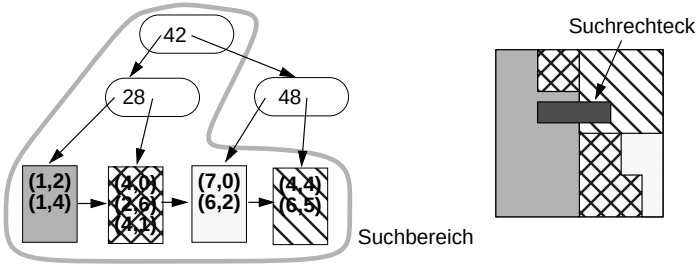
B+-Baum mit Z-Ordnung

Window Query: 1. Ansatz

- Benutze den “gewöhnlichen” Algorithmus für Bereichsanfragen im B^+-Baum:
  - Suche mit dem kleinsten Z-Wert des Suchrechtecks (entspricht dem linken unteren Eckpunkt) das zugehörige Blatt im B^+-Baum
  - Durchlaufe sequentiell die Blätter bis ein Z-Wert größer als der größte Z-Wert im Suchrechteck gefunden wurde

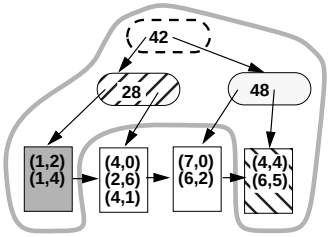


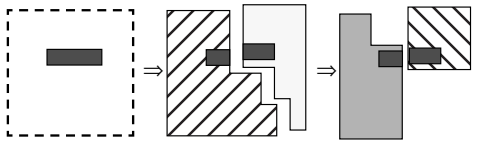
  - – ineffizient, da der Suchbereich verglichen mit dem Suchrechteck sehr groß ist

B+-Baum mit Z-Ordnung

Window Query: 2. Ansatz

- Jeder Knoten des B-Baums repräsentiert einen Bereich des Datenraums
- In jedem Knoten wird durch die Separatoren der zugehörige Bereich des Knotens in Teilbereiche zerlegt
- *Idee:* Verwende zur Beantwortung der Anfrage in einem Teilbaum nur den Teil des Suchrechtecks, der den Bereich des Teilbaums schneidet





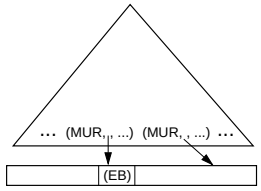
  - – Mehraufwand für das Durchlaufen der Indexseiten im B^+-Baum
  - – Teilbereiche sind nicht notwendigerweise Rechtecke
  - \+ Zugriff nur auf die tatsächlich relevanten Daten- und Directoryseiten

7.3 Räumliche Zugriffsstrukturen

Beobachtungen

- traditionelle Zugriffsstrukturen sind für die Verwaltung von geometrischen Daten nicht (ohne weiteres) geeignet
  - ⇒ *räumliche Zugriffsstrukturen (RZS)*, die geometrische Selektionen und Kombinationen unterstützen

- Geo-Objekte sind komplex
  - ⇒ Organisation von vereinfachten Geo-Objekten *(Approximationen)*, z.B. *minimal umgebende Rechtecke (MUR)*
  - ⇒ Verweis auf die exakte Beschreibung (EB) des Geo-Objektes

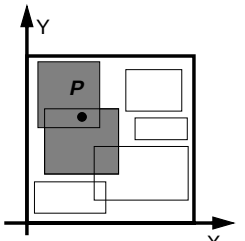


Zugriffsstruktur

Anforderungen

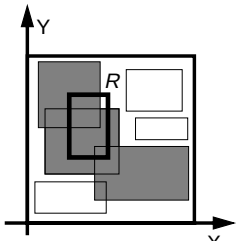
Unterstützung von geometrischen Anfragen

- Gegeben ein Anfragepunkt *P* bzw. ein Anfragerechteck *R*
- Finde die Geo-Objekte *Obj* mit



$P \in Obj.MUR$

Point Query



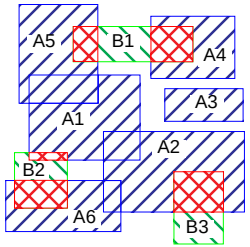
$R \cap Obj.MUR \neq \emptyset$

Window Query

Räumlicher Verbund

Unterstützung von geometrischen Anfagen (Forts.)

- Gegeben zwei Mengen von minimal umgebender Rechtecke
  $M_1 = \{MUR_{1,1}, MUR_{1,2}, \ldots, MUR_{1,m}\}$ und $M_2 = \{MUR_{2,1}, MUR_{2,2}, \ldots, MUR_{2,n}\}$
- Berechne die Menge
  $\{ (MUR_1, MUR_2) \mid MUR_1 \in M_1, MUR_2 \in M_2 \text{ und } MUR_1 \cap MUR_2 \neq \emptyset\}$



MUR-Join

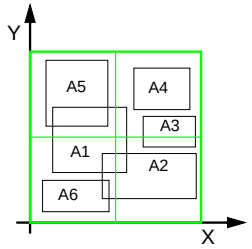
Lösungsmenge:

(A5, B1)
(A4, B1)
(A1, B2)
(A6, B2)
(A2, B3)

Prinzipielle Techniken: Clipping

Clipping

- vollständige Zerlegung des Datenraums in disjunkte Zellen
- Speicherung aller Objekte, die eine Zelle schneiden, in einer der Zelle exklusiv zugeordneten Seite (bzw. statt dessen Speicherung von Objektverweisen oder "geclippten" Teilobjekten)



4 Zellen (Seiten):

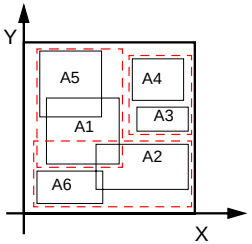
A1 A2 A3	A1 A2 A6	A1 A3 A4	A1 A5

  - Redundanz
  - schlechtes Leistungsverhalten für Einfügen, Löschen, große Window Queries

Prinzipielle Techniken: Über. Regionen

Überlappende Seitenregionen

- Aufteilung der Rechtecke in Gruppen, wobei jede Gruppe exklusiv in einer Seite abgespeichert wird



3 Gruppen (Seiten):

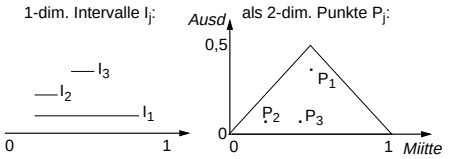
A4 A3	A5 A1	A6 A2

  - Überlappung (Point Query muß ggf. mehrere Seiten durchsuchen)

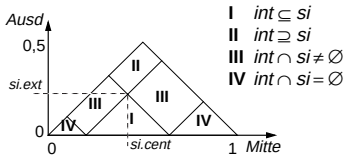
Prinzipielle Techniken: Transformation

Transformation in höherdimensionalen Raum (Punkttransformation)

- Abbildung von *n*-dimensionalen Rechtecken in 2*n*-dimensionale Punkte
- Speicherung durch 2*n*-dimensionale Punktzugriffsstruktur (z.B. Gridfile)



Mittelpunkt-Ausdehnungs-Darstellung



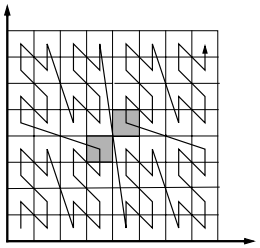
Anfrageräume zur Suche von Intervallen *int* bzgl. eines Intervalls *si* = (*si.cent*,*si.ext*)

  - Verlust von räumlicher Nähe möglich:
    - $I_1 \cap I_2 \neq \emptyset$, $I_2 \cap I_3 = \emptyset$
    - $d(P_1,P_2) > d(P_2,P_3)$

Prinzipielle Techniken: 1-dim. Transf.

Einbettung in eindimensionalen Raum

- vollständige Zerlegung des Datenraums in gleichförmige disjunkte Zellen
- Definition einer linearen Ordung auf diesen Zellen
- Organisation der Zellen über eine herkömmliche (eindimensionale) Zugriffsstruktur (z.B. B-Baum)



21	23	29	31	53	55	61	63
20	22	28	30	52	54	60	62
17	19	25	27	49	51	57	59
16	18	24	26	48	50	56	58
5	7	13	15	37	39	45	47
4	6	12	14	36	38	44	46
1	3	9	11	33	35	41	43
0	2	8	10	32	34	40	42

– zum Teil Verlust von räumlicher Nähe

7.3.1 R-Baum

- Zugriffsstruktur für die effiziente Verwaltung von Rechtecken
- basiert auf der Idee überlappender Seitenregionen
- verallgemeinert die Idee des B^+-Baums

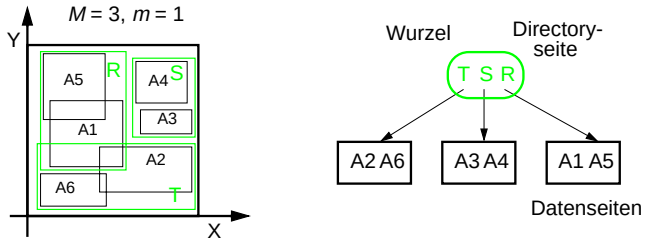
Definition

Ein *R-Baum* mit ganzzahligen Parameter m und M, $1 \le m \le \lceil M/2 \rceil$, organisiert eine Menge von Rechtecken in einem Baum mit folgenden Eigenschaften:

- Der Baum besteht aus Daten- und Directoryseiten. In einer *Datenseite* werden Rechtecke (plus weitere Informationen) und in einer *Directoryseite* Indexeinträge der Form (*R*, *Ref*) gehalten. Hier bezeichnet *R* ein Rechteck und *Ref* eine Referenz auf einen Teilbaum.
- Jedes Rechteck eines Indexeintrags überdeckt die Datenrechtecke des zugehörigen Teilbaums minimal.
- Alle Datenseiten sind Blätter des Baums. Der Baum ist vollständig balanciert, d.h. alle Pfadlängen von der Wurzel zu einem Blatt sind gleich.
- Jede Seite besitzt maximal M Einträge und, mit Ausnahme der Wurzel, mindestens m Einträge.

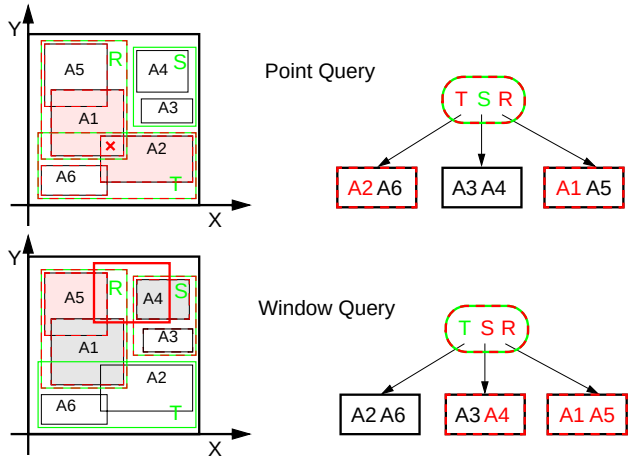
Beispiel

Beispiel



Anfragenbearbeitung im R-Baum

Suchanfragen

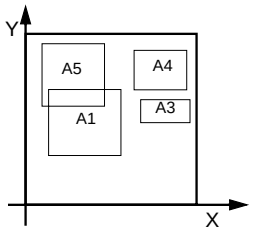


Dynamischer Aufbau (I)

Optimierungsziele

- geringe Überlappung der Seitenregionen
- Seitenregionen mit geringem Flächeninhalt (geringe Überdeckung von totem Raum)
- Seitenregionen mit geringem Umfang

Aufbau

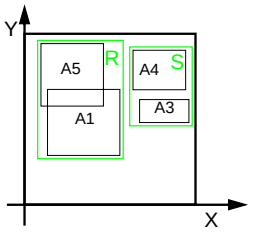


Start: leere Datenseite (= Wurzel)

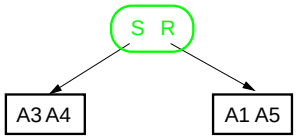
Einfügen von: A5, A1, A3, A4

A5, A1, A3, A4 * (Überlauf)

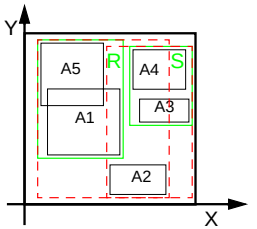
Dynamischer Aufbau (II)

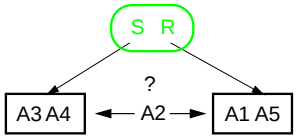


⇒ Split in 2 Seiten



Frage: Wie wird aufgeteilt? *(Splitstrategie)*





Frage: Wo wird eingefügt? *(Einfügestrategie)*

Einfügestrategie des R-Baums

Das Rechteck *R* ist in einen R-Baum einzufügen

Fälle

- Fall 1: *R* fällt vollständig in genau ein Directory-Rechteck *D*
  - ⇒ Einfügen in Teilbaum von *D*
- Fall 2: *R* fällt vollständig in mehrere Directory-Rechtecke D_1, ... , D_n
  - ⇒ Einfügen in Teilbaum von D_i, das die geringste Fläche aufweist
- Fall 3: *R* fällt vollständig in kein Directory-Rechteck
  - ⇒ Einfügen in Teilbaum von *D*, das den geringsten Flächenzuwachs erfährt (in Zweifelsfällen: ... , das die geringste Fläche hat)
  - ⇒ *D* muß entsprechend vergrößert werden

Variationsmöglichkeiten

- Einbeziehen der entstehenden Überlappung, des Umfangs, des toten Raums

Splitstrategien für R-Bäume (I)

Der Knoten *K* läuft mit $|K| = M+1$ über:

⇒ Aufteilung auf zwei Knoten K_1 und K_2, s.d. $|K_1| \geq m$ und $|K_2| \geq m$

Erschöpfender Algorithmus

- Suche unter den $O(2^M)$ Möglichkeiten die "beste" aus ⇒ sehr aufwendig ($M \approx 200$)

Linearer Algorithmus

- Suche für jede Dimension die "Extrem"-Rechtecke
- Wähle das Paar von Rechtecken mit den größten (normalisierten) Abstand bezüglich einer Dimension; diese beiden Rechtecke bilden K_1 und K_2
- Durchlaufe alle verbliebenen Rechtecke R_i und weise sie dem Knoten K_j zu, der dadurch den geringsten Flächenzuwachs erfährt
- Falls die Anzahl der verbliebenen Rechtecke = $m - |K_j|$, dann weise sie K_j zu

Quadratischer Algorithmus (Änderungen)

- Wähle das Paar von Rechtecken R_1 und R_2 mit dem größten Wert von *d* als Ausgangsmenge für K_1 bzw. K_2; d := Fläche($R_1 \cup R_2$) - Fläche(R_1) - Fläche(R_2)
- Durchlaufe alle verbliebenen Rechtecke R_i in einer Reihenfolge, so daß immer das Rechteck R_i als nächstes zugeordnet wird, wo die Differenz zwischen Fläche($K_1 \cup R_i$)-Fläche (K_1) und Fläche($K_2 \cup R_i$)-Fläche (K_2) am größten ist

Splitstrategien für R-Bäume (II)

R*-Baum-Split

- **Bestimmung der Splitdimension**
  - Sortiere für jede Dimension die Rechtecke gemäß ihrer Extremwerte

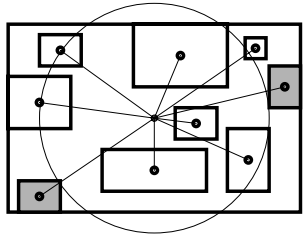
  Für jede Dimension:
  - Für jede der beiden Sortierungen werden M-2m+2 Aufteilungen der M+1 Rechtecke bestimmt, so daß die 1. Gruppe der j-ten Aufteilung m-1+j Rechtecke und die 2. Gruppe die übrigen Rechtecke enthält
  - *UG* sei die Summe aus dem Umfang der beiden MURs R_1 und R_2 um die Rechtecke der beiden Gruppen
  - *US* sei die Summe der *UG* aller berechneten Aufteilungen

  Es wird die Dimesion mit dem geringsten *US* als Splitdimension gewählt
- **Bestimmung der Aufteilung**
  - Es wird die Aufteilung der gewählten Splitdimension genommen, bei der R_1 und R_2 die geringste Überlappung haben
  - In Zweifelsfällen wird die Aufteilung genommen, bei der R_1 und R_2 die geringste Überdeckung von totem Raum besitzen

(Die besten Resultate hat bei Experimenten $m = 0{,}4*M$ ergeben)

Vermeidung von Splits (R*-Baum)

- Bevor eine Seite einem Split unterzogen wird, werden die am weitesten vom Zentrum des gemeinsamen minimalen Rechtecks entfernt liegenden Einträge (Einträge von Datensätzen oder von Teilbäumen) gelöscht und noch einmal in den R*-Baum eingefügt *(Forced Reinsert)*

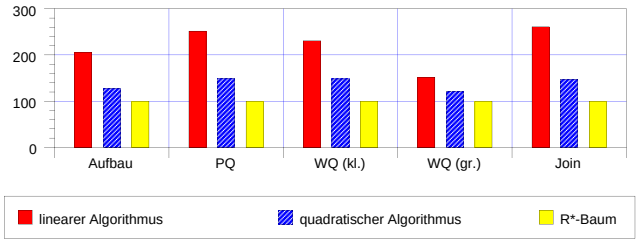


- Ziele:
  - Vermeiden von Splits ⇒ bessere Speicherplatzausnutzung
  - Anpassung des R*-Baums an die aktuelle Datenverteilung (bessere Unabhängigkeit von sortierten Einfügungen)

(Anteil der gelöschten und wieder eingefügten Rechtecke beim R*-Baum: 30%)

Leistungsvergleich

(Beckmann, Kriegel, Schneider, Seeger 1990)



- Messung der Anzahl der Seitenzugriffe
- R*-Baum (R*-Baum-Split & Forced ReInsert) auf 100 normalisiert

Zusammenfassung

Zusammenfassung

- Prinzip der überlappenden Seitenregionen
  - Rechtecke im Directory können sich überlappen
  - Point Query nicht auf einem Pfad beschränkt
- Rechtecke, die Objekte approximieren, werden genau einmal in der Struktur gespeichert
- Relativ einfach zu implementieren
- Einfüge- und Splitstrategien basieren auf heuristischen Überlegungen
- Optimierungsgesichtspunkte:
  - geringe Überlappung der Seitenregionen
  - Seitenregionen mit geringem Flächeninhalt / geringe Überdeckung von totem Raum
  - Seitenregionen mit geringem Umfang
  - Speicherplatzausnutzung