

Kapitel 4: Räumliche Indexstrukturen

Skript zur Vorlesung
Geo-Informationssysteme

Wintersemester 2013/14

Ludwig-Maximilians-Universität München

(c) Peer Kröger 2011, basierend auf dem Skript von Christian Böhm aus dem
SoSe 2009



4.1 Einführung

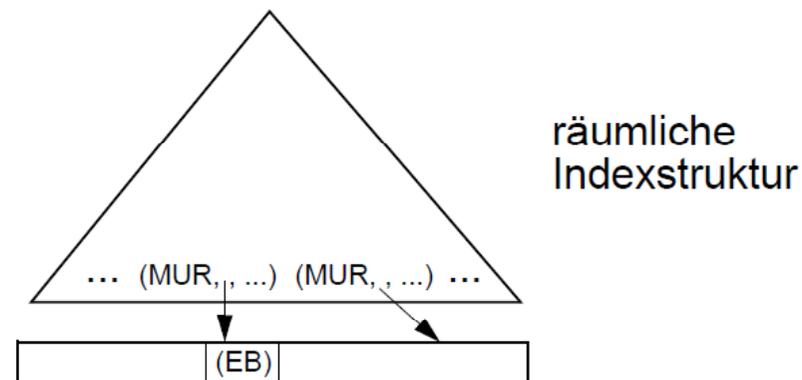
4.2 Z-Ordnung

4.3 R-Bäume

4.4 Quadrees

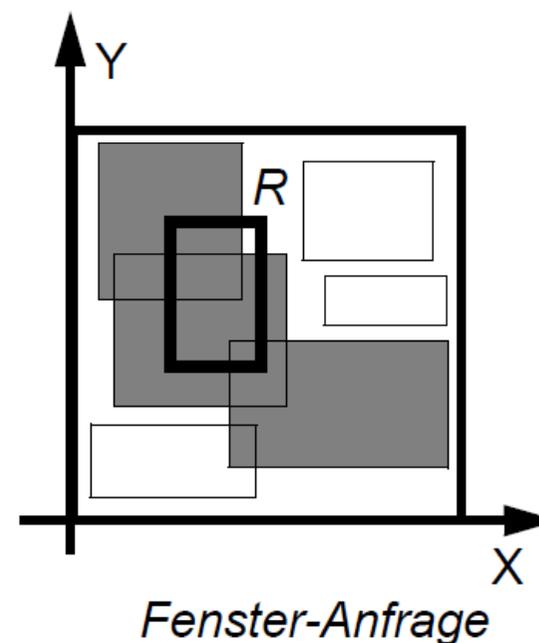
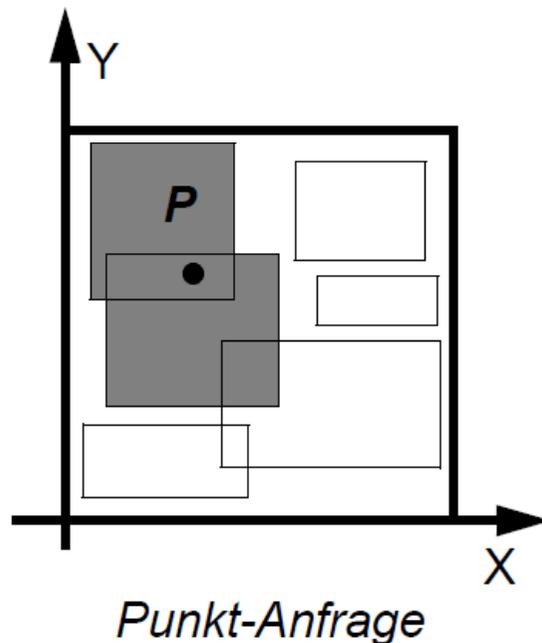
Grundlegende Ideen

- konventionelle Zugriffsstrukturen sind für die Verwaltung von geometrischen Daten schlecht geeignet
 ⇒ *räumliche Indexstrukturen*, die die Ordnung des multidimensionalen Datenraums auf dem Sekundärspeicher möglichst gut erhalten
- Geo-Objekte variieren stark in ihrer Größe, sind aber auf Seiten fester Größe abzuspeichern
 ⇒ Organisation von vereinfachten Geo-Objekten (*Approximationen*), z.B. *minimal umgebende Rechtecke (MUR)*
 ⇒ Verweis auf die exakte Beschreibung (EB) des Geo-Objektes



Zu unterstützende Anfragen

- Gegeben ein Anfragepunkt P bzw. ein Anfragerecteck R
 - *Punkt-Anfrage*: Finde die Geo-Objekte Obj mit $P \in Obj.MUR$.
 - *Fenster-Anfrage*: Finde die Geo-Objekte Obj mit $R \cap Obj.MUR \neq \emptyset$.



Zu unterstützende Anfragen (Forts.)

- Gegeben zwei Mengen minimal umgebender Rechtecke

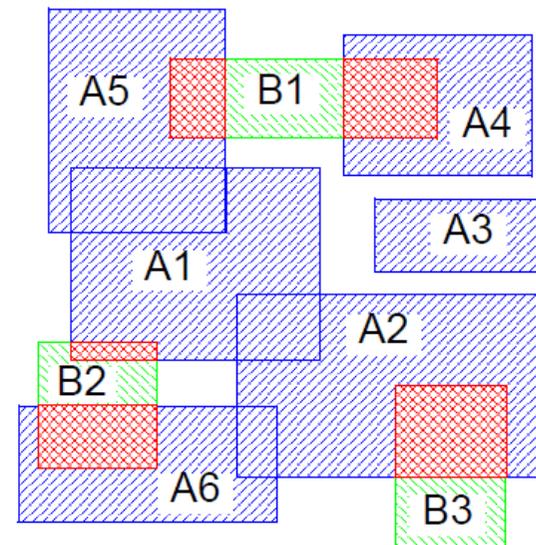
$$M_1 = \{MUR_{1,1}, MUR_{1,2}, \dots, MUR_{1,m}\}$$

$$M_2 = \{MUR_{2,1}, MUR_{2,2}, \dots, MUR_{2,n}\}$$

- *Spatial-Join*: Berechne die Menge $\{(MUR_1, MUR_2) \mid MUR_1 \in M_1, MUR_2 \in M_2 \text{ und } MUR_1 \cap MUR_2 \neq \emptyset\}$.

⇒ auch andere räumliche

Prädikate möglich



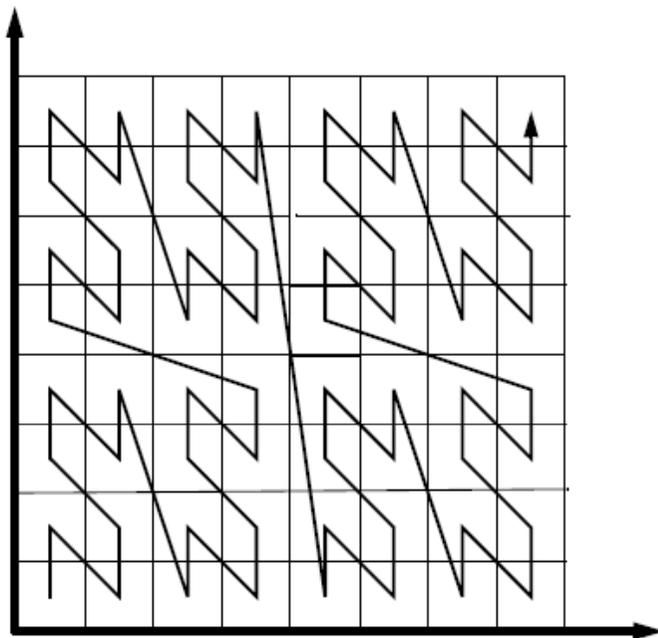
Antwortmenge:

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

Spatial-Join

Einbettung in eindimensionalen Raum

- vollständige Zerlegung des Datenraums in gleichförmige disjunkte Zellen
- Definition einer linearen Ordnung auf diesen Zellen (z.B. Z-Ordnung)
- Organisation der Zellen über eine konventionelle (eindimensionale) Indexstruktur (z.B. B-Baum)

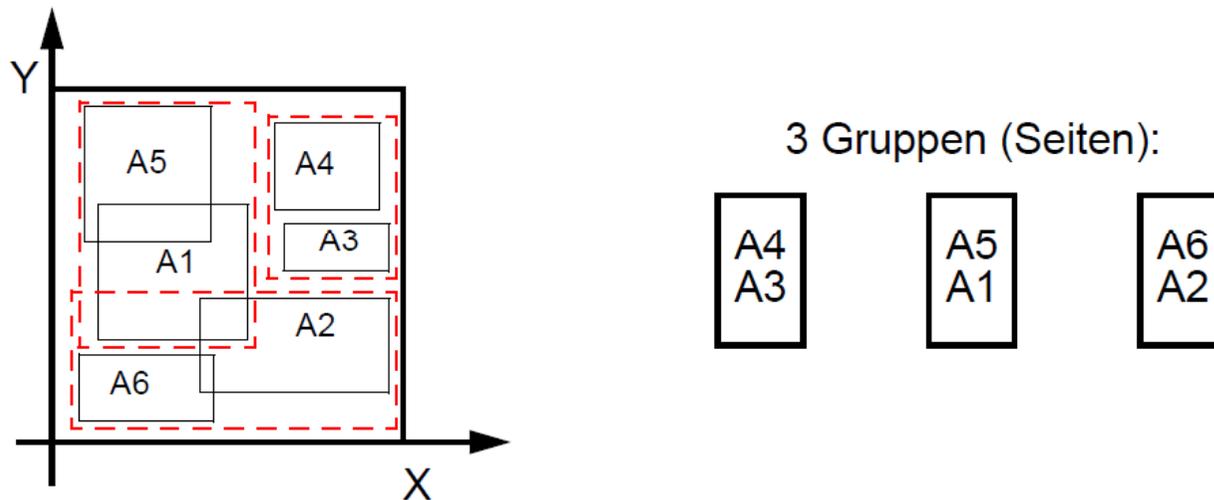


| | | | | | | | |
|----|----|----|----|----|----|----|----|
| 21 | 23 | 29 | 31 | 53 | 55 | 61 | 63 |
| 20 | 22 | 28 | 30 | 52 | 54 | 60 | 62 |
| 17 | 19 | 25 | 27 | 49 | 51 | 57 | 59 |
| 16 | 18 | 24 | 26 | 48 | 50 | 56 | 58 |
| 5 | 7 | 13 | 15 | 37 | 39 | 45 | 47 |
| 4 | 6 | 12 | 14 | 36 | 38 | 44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

Z-Ordnung

Organisation des 2D-Raums (Gruppierung und Speicherung in MURs)

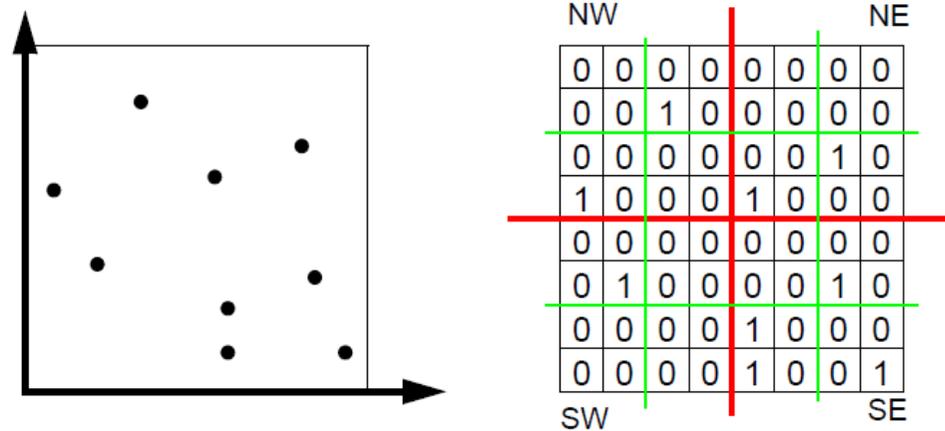
- Aufteilung der Rechtecke (MURs) in disjunkte Gruppen, wobei jede Gruppe exklusiv in einer Seite abgespeichert wird



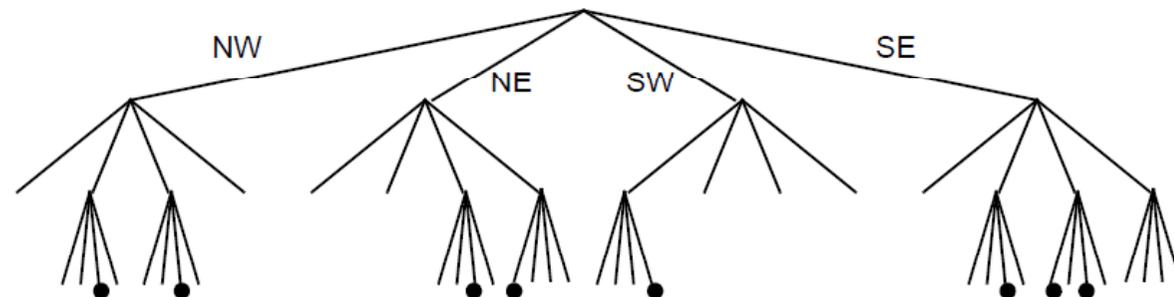
- Im Directory MURs der im Teilbaum enthaltenen MURs abspeichern.
- Split des Datenraums je nach abgespeicherten Daten.
⇒ R-Baum

Organisation des 2D-Raums (hierarchische Zerlegung des Raums)

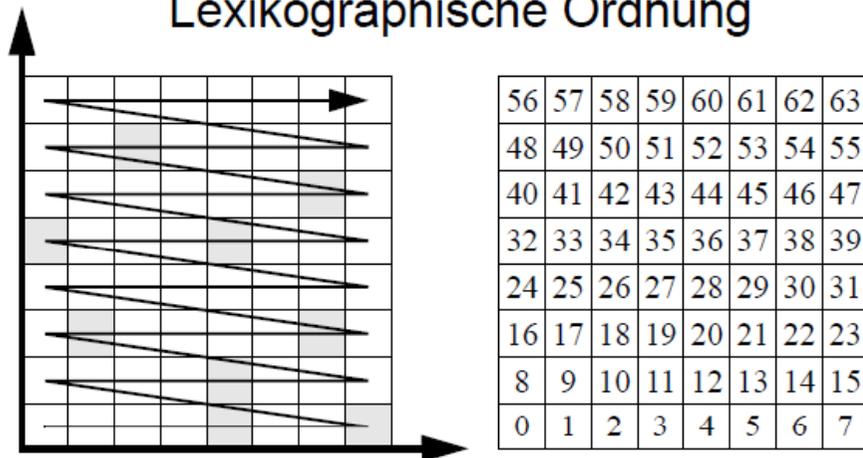
- Split des Datenraums immer in der Mitte einer Dimension
- Punkte = Pixel in dem gerasterten Datenraum



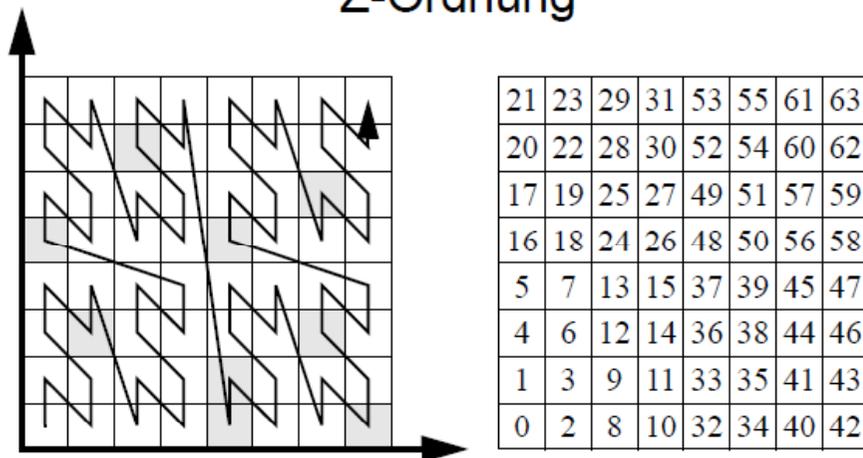
⇒ Quadrees



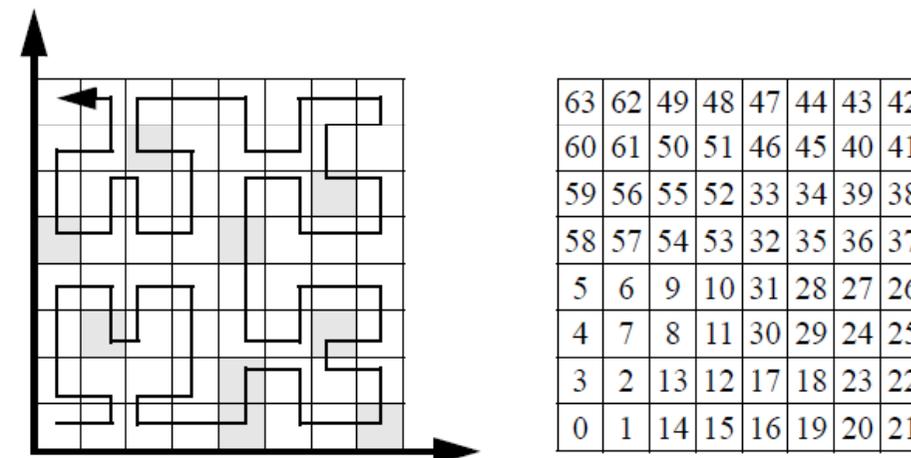
Lexikographische Ordnung



Z-Ordnung

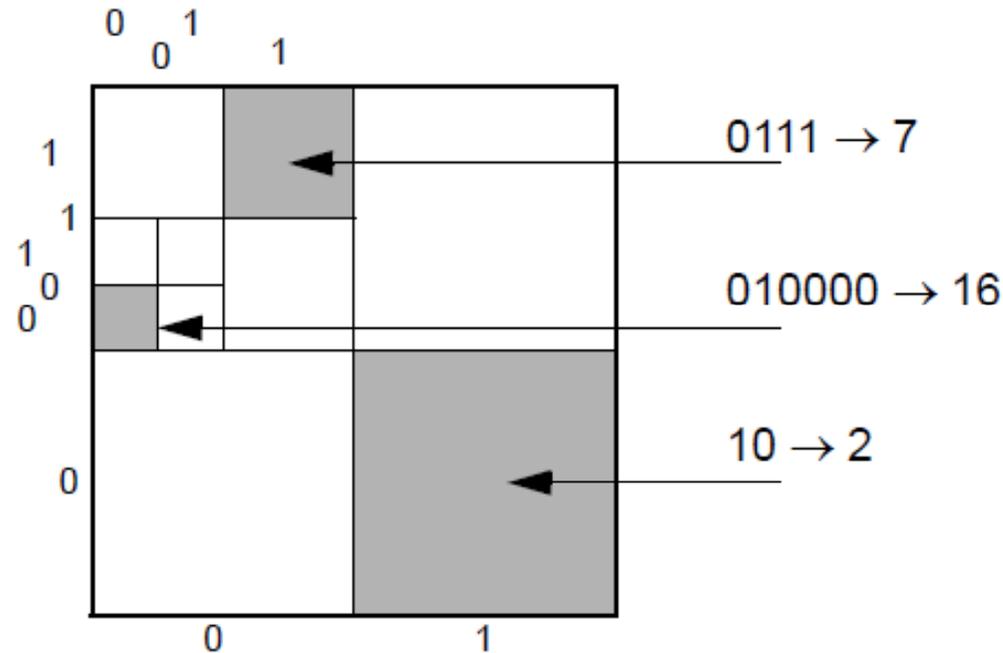


Hilbert-Kurve



- Z-Ordnung erhält die räumliche Nähe relativ gut
- Z-Ordnung ist effizient berechenbar

Codierung von Zellen



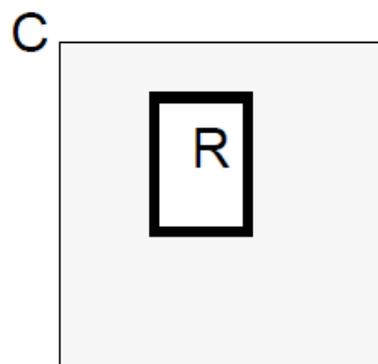
abwechselnd rechts/links und oben/unten partitionieren

Z-Werte

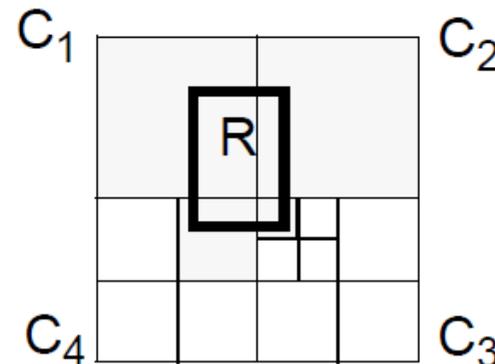
- *Level* eines Codes = Anzahl der Bits
- *Z-Wert* = (Dezimalwert des Codes, Level)

Codierung von Rechtecken (MURs)

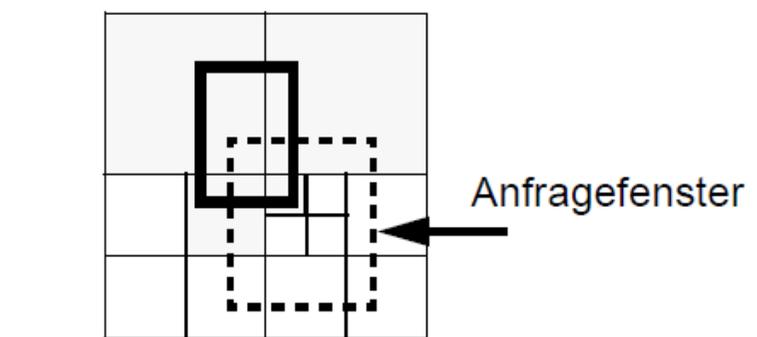
- durch die minimal umgebende Zelle
⇒ Probleme mit Rechtecken, die auf den Schnittlinien liegen
- durch mehrere Zellen
⇒ bessere Approximation des Rechtecks
⇒ redundante Abspeicherung
⇒ redundante Antworten



Codierung von R durch eine Zelle



Codierung von R durch mehrere Zellen



Anfrage liefert mehrmals die gleiche Antwort

Abbildung auf B⁺-Baum

- Lineare Ordnung zur Verwaltung im B⁺-Baum
 - Seien (c_1, l_1) und (c_2, l_2) zwei Z-Werte und sei $l = \min \{l_1, l_2\}$.
 - Dann ist die Ordnungsrelation \leq_z wie folgt definiert:

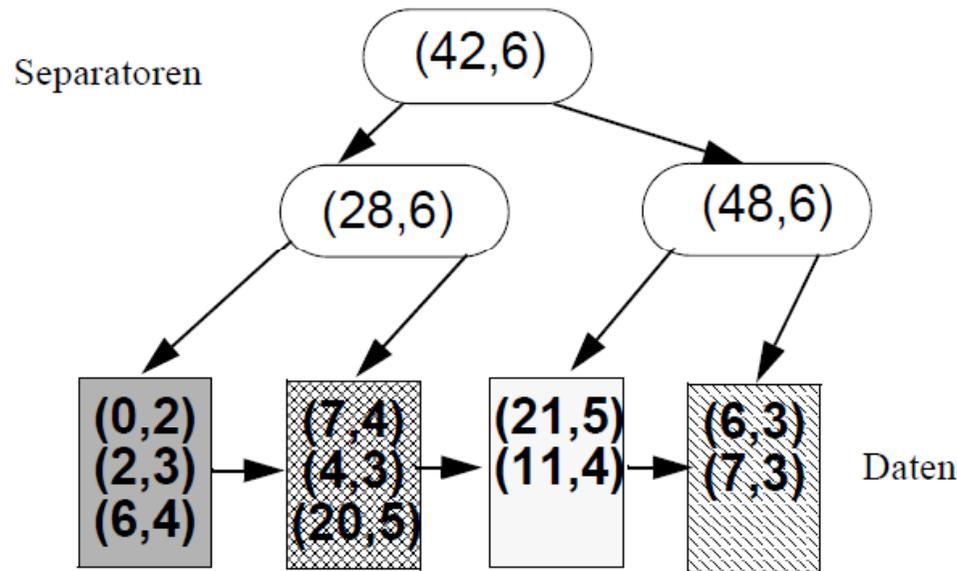
$$(c_1, l_1) \leq_z (c_2, l_2) \quad \text{falls} \quad c_1 \text{div } 2^{l_1-l} \leq c_2 \text{div } 2^{l_2-l}$$

- Beispiele:

$$(1,2) \leq_z (3,2), \quad (3,4) \leq_z (3,2), \quad (1,2) \leq_z (10,4)$$

- Wenn ein Blatt des B⁺-Baums überläuft, dann Split der Seite in 2 Seiten gemäß B⁺-Baum Strategie.

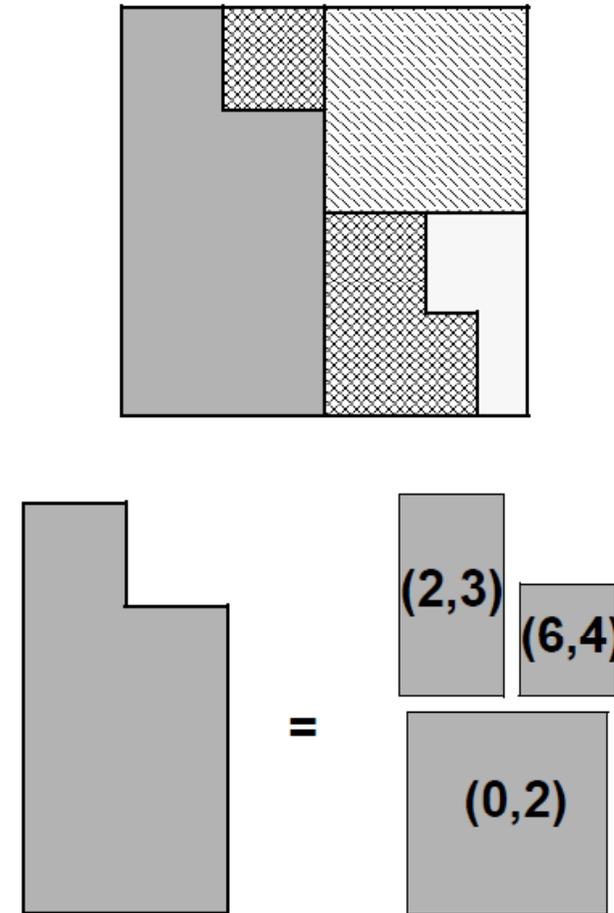
Beispiel



$$(0,2) \leq (28,6) \leq (7,4) \leq (42,6)$$

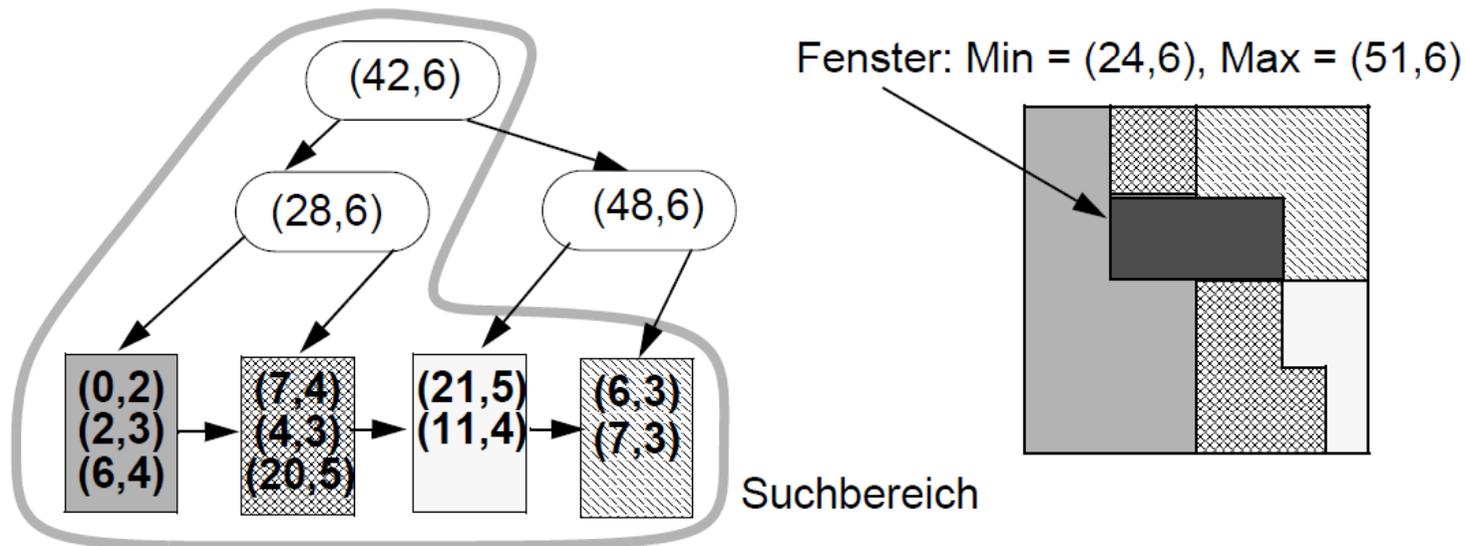
$$(2,3) \leq (28,6) \leq (4,3) \leq (42,6)$$

$$(6,4) \leq (28,6) \leq (20,5) \leq (42,6)$$



Fenster-Anfrage (1. Ansatz)

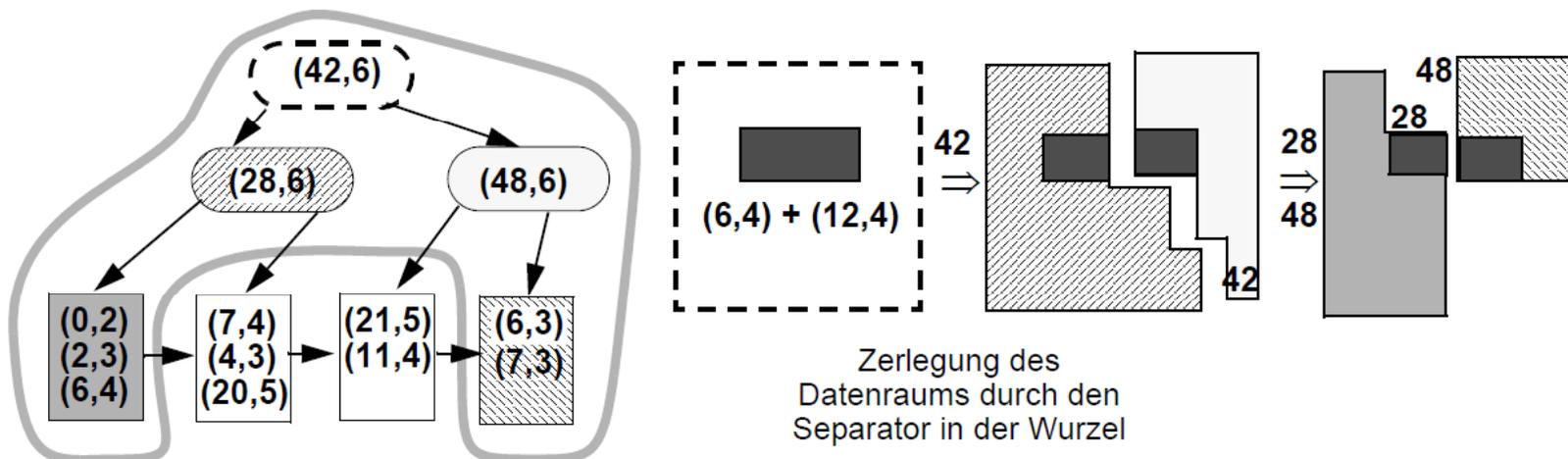
- Benutze den "gewöhnlichen" Algorithmus für Bereichsanfragen im B+-Baum:
 - Suche für den kleinsten Z-Wert des Windows (entspricht dem linken unteren Eckpunkt) das zugehörige Blatt im B+-Baum
 - Durchlaufe sequentiell die Blätter bis ein Z-Wert größer als der größte Z Wert im Suchrechteck gefunden wurde



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

Fenster-Anfrage (2. Ansatz)

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



- Mehraufwand für das Durchlaufen der Indexseiten im B⁺-Baum (Separatoren)
- Teilbereiche sind nicht notwendigerweise Rechtecke
- + Zugriff nur auf die tatsächlich relevanten Datenseiten

Idee

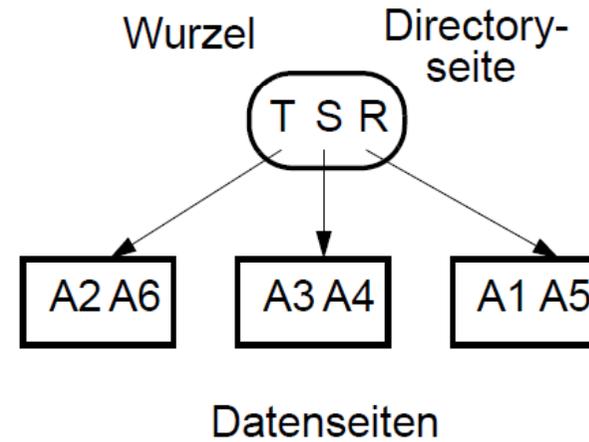
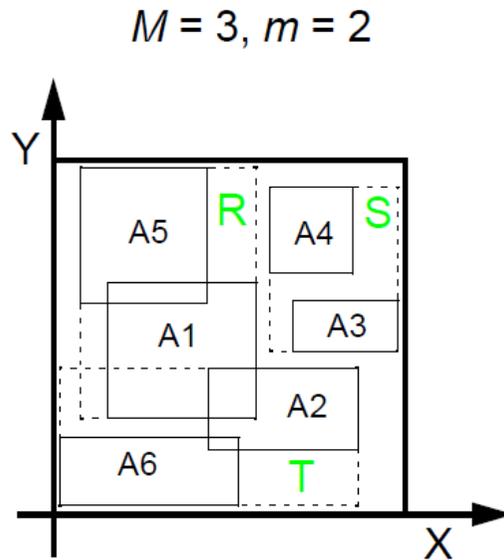
- basiert auf der Technik überlappender Seitenregionen
- verallgemeinert die Idee des B⁺-Baums auf den 2-dimensionalen Raum

Definition

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

- In einer *Datenseite* werden Einträge der Form (Rectangle, Verweis auf die exakte Beschreibung, weitere Attribute) verwaltet.
- In einer *Directoryseite* werden Indexeinträge der Form (Rectangle, Subtree[^]) gehalten. Hier bezeichnet Rectangle ein MUR und Subtree[^] eine Referenz auf einen Teilbaum.
- Jedes Rechteck eines Indexeintrags überdeckt die Datenrechtecke (MURs) des zugehörigen Teilbaums.
- 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



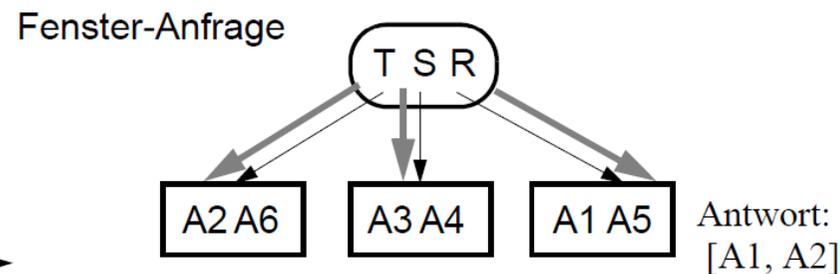
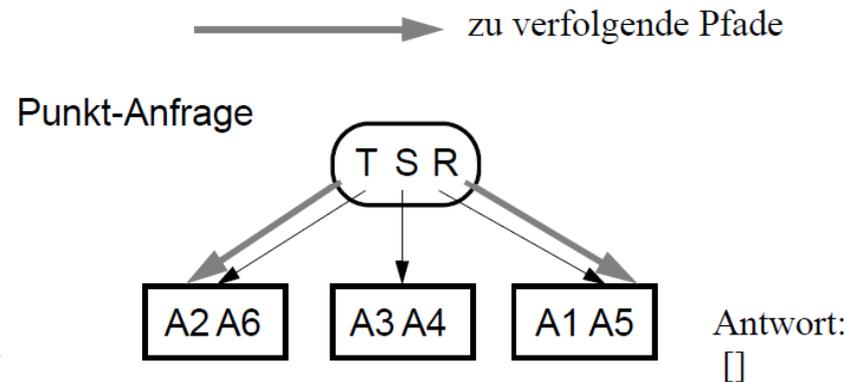
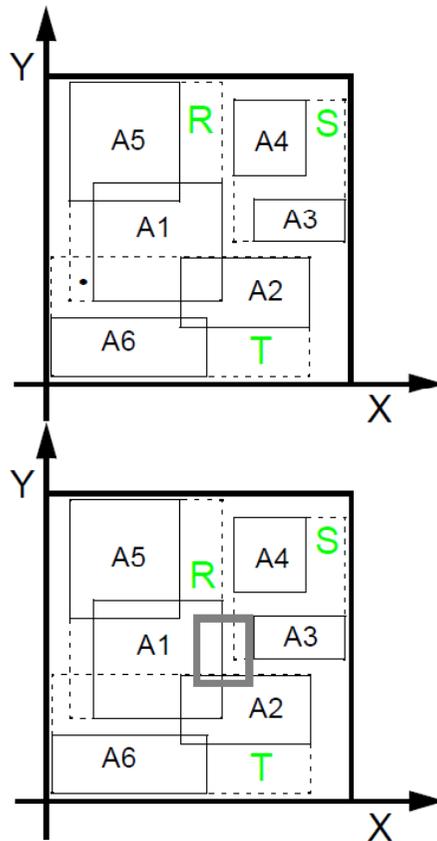
Höhe eines R-Baums

- Ist N die Anzahl der gespeicherten Datensätze, so gilt für die Höhe h des R-Baumes:

$$h \leq \lceil \log_m N \rceil + 1$$

- Die Höhe eines R-Baums ist also $O(\log N)$.

Anfragen



- Erster Aufruf jeweils mit Page = Seite der Wurzel
- Gibt es eine Überlappung der Directory-Rechtecke im Bereich der Anfrage, verzweigt die Suche in mehrere Pfade.

Punkt-Anfrage

PointQuery (Page, Point);

FOR ALL Entry \in Page DO

IF Point IN Entry.Rectangle THEN

IF Page = DataPage THEN

Write (Entry.Rectangle)

ELSE

PointQuery (Entry.Subtree[^], Point);

Fenster-Anfrage

Window Query (Page, Window);

FOR ALL Entry \in Page DO

IF Window INTERSECTS Entry.Rectangle THEN

IF Page = DataPage THEN

Write (Entry.Rectangle)

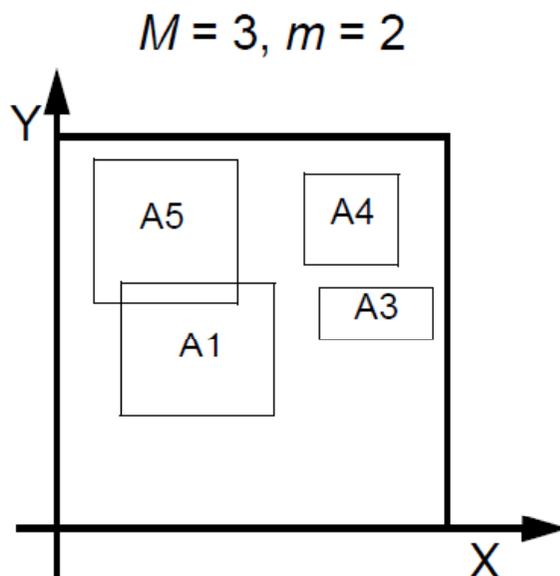
ELSE

WindowQuery (Entry.Subtree[^], Window);

Optimierungsziele

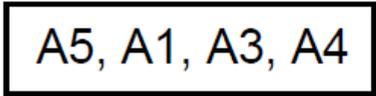
- geringe Überlappung der Seitenregionen
- Seitenregionen mit geringem Flächeninhalt
⇒ geringe Überdeckung von totem Raum
- Seitenregionen mit geringem Umfang

Aufbau

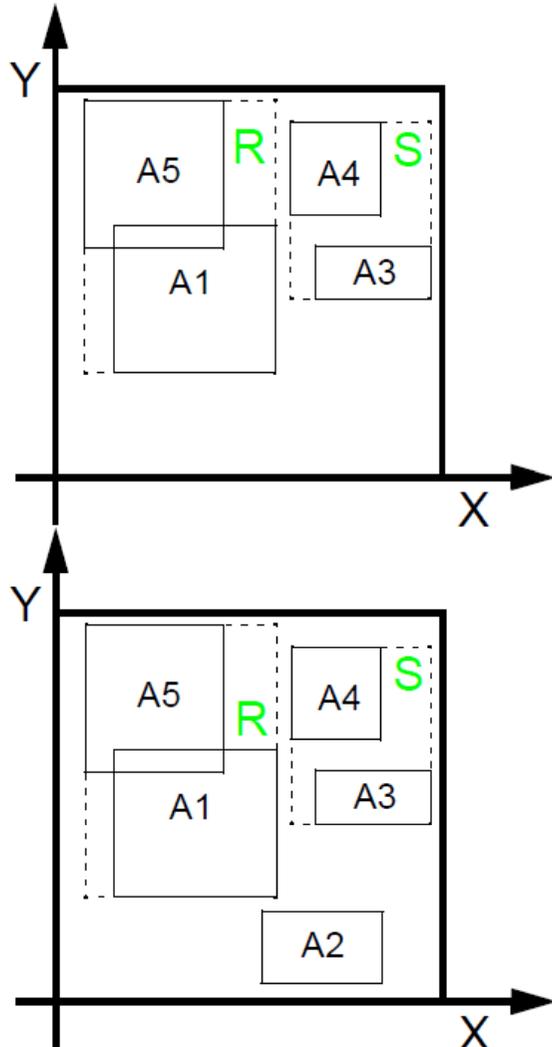


Start:  leere Datenseite
(= Wurzel)

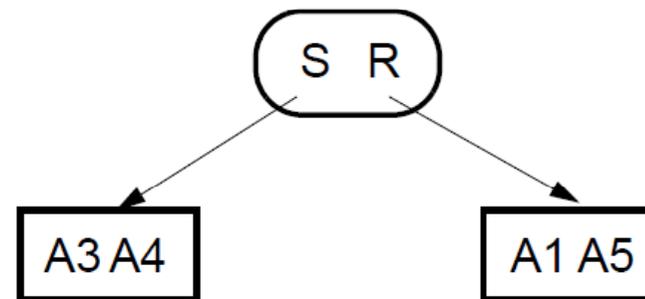
Einfügen von: A5, A1, A3, A4

 * (Überlauf)

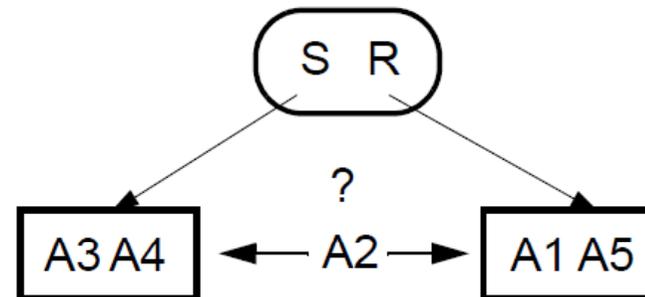
Aufbau (Fortsetzung)



⇒ Split in 2 Seiten



Frage: Wie wird aufgeteilt? (*Splitstrategie*)



Frage: Wo wird eingefügt? (*Einfügestrategie*)

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, \dots, 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: \dots , das die geringste Fläche hat)
⇒ D muß entsprechend vergrößert werden

Das Rechteck R ist in einen R-Baum einzufügen

Strategie des R*-Baums

- Fall 3.a: Die Directoryseite D verweist auf Directoryseiten
⇒ Einfügen in Teilbaum des D, das den geringsten Flächenzuwachs erfährt
- Fall 3.b: Die Directoryseite D verweist auf Datenseiten
⇒ Einfügen in Teilbaum des D, das den kleinsten Zuwachs an Überlappung bringt

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

⇒ Aufteilung auf zwei Knoten K_1 und K_2 , sodaß $|K_1| \geq m$ und $|K_2| \geq m$

Erschöpfender Algorithmus

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

Quadratischer Algorithmus

- Wähle das Paar von Rechtecken R_1 und R_2 mit dem größten Wert für den "toten Raum" im MUR, falls R_1 und R_2 in denselben Knoten K_i kämen.

$$d(R_1, R_2) := \text{Fläche}(\text{MUR}(R_1 \cup R_2)) - \text{Fläche}(R_1) - \text{Fläche}(R_2)$$

Setze $K_1 := \{R_1\}$ und $K_2 := \{R_2\}$.

- Wiederhole den folgenden Schritt bis zu STOP:
 - wenn alle R_i zugeteilt sind: STOP
 - wenn alle restlichen R_i benötigt werden, um den kleineren Knoten minimal zu füllen: teile sie alle zu und STOP
 - sonst: wähle das nächste R_i und teile es dem Knoten zu, dessen MUR den kleineren Flächenzuwachs erfährt. Im Zweifelsfall bevorzuge den K_i mit kleinerer Fläche des MUR bzw. mit weniger Einträgen.

Linearer Algorithmus

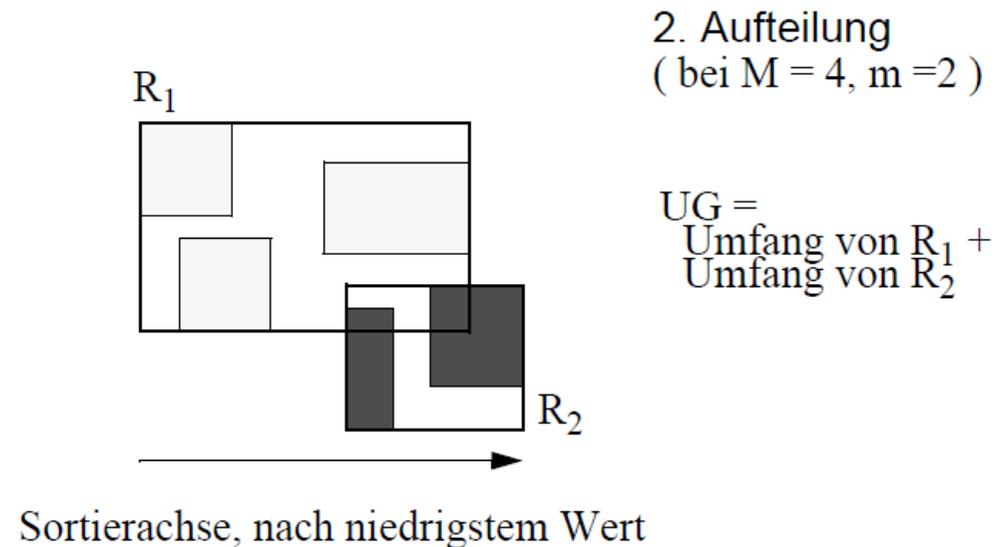
- Der lineare Algorithmus ist identisch mit dem quadratischen Algorithmus bis auf die Auswahl des initialen Paares (R_1, R_2) .
- Wähle das Paar von Rechtecken R_1 und R_2 mit dem "größten Abstand", genauer:
 - Suche für jede Dimension das Rechteck mit dem kleinsten Maximalwert und das Rechteck mit dem grössten Minimalwert (*maximaler Abstand*).
 - Normalisiere den *maximalen Abstand* jeder Dimension, indem er durch die Summe der Ausdehnungen der R_i in der Dimension dividiert wird (*setze den maximalen Abstand der Rechtecke ins Verhältnis zur ihrer Ausdehnung*).
 - Wähle das Paar von Rechtecken mit dem größten normalisierten Abstand bzgl. aller Dimensionen. Setze $K_1 := \{R_1\}$ und $K_2 := \{R_2\}$.
- Dieser Algorithmus ist linear in der Zahl der Rechtecke M und in der Zahl der Dimensionen d .

Idee der R*-Baum Splitstrategie

- sortiere die Rechtecke in jeder Dimension nach beiden Eckpunkten und betrachte nur Teilmengen nach dieser Ordnung benachbarter Rechtecke
- Laufzeitkomplexität ist $O(d * M * \log M)$ für d Dimensionen und M Rechtecke

Bestimmung der Splitdimension

- Sortiere für jede Dimension die Rechtecke gemäß beider 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 die ersten $m-1+j$ 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 Dimension mit dem geringsten US als Splitdimension gewählt.

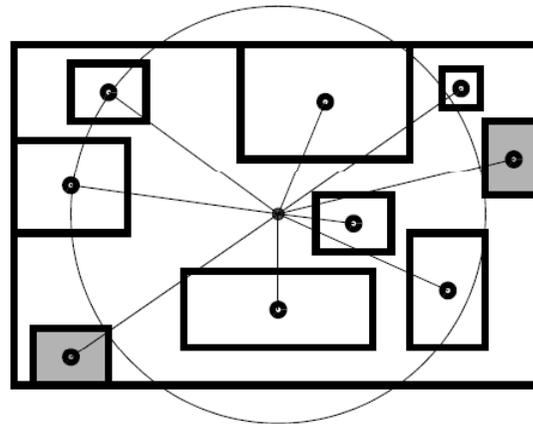


Bestimmung der Aufteilung

- Es wird die Aufteilung der gewählten Splittedimension 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

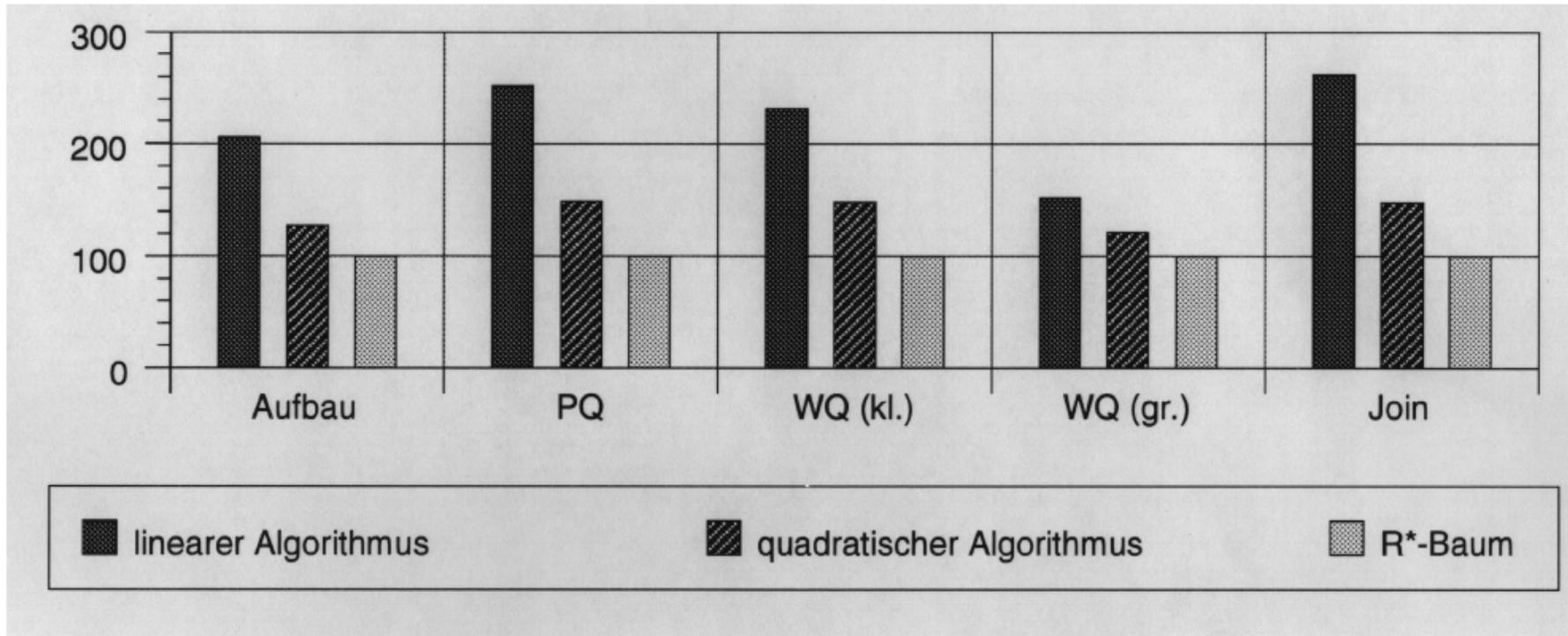
- Bevor eine Seite einem Split unterzogen wird, werden die am weitesten vom Zentrum der Seitenregion entfernten Einträge gelöscht und noch einmal in den R*-Baum eingefügt (*Forced Reinsert*)



- Ziele
 - Vermeiden von Splits (nicht immer möglich) \Rightarrow bessere Speicherplatzausnutzung
 - Anpassung des R*-Baums an die aktuelle Datenverteilung (mehr Unabhängigkeit von der Reihenfolge der Einfügungen)
- Erfahrung: Anteil der zu löschenden und wieder einzufügenden Rechtecke = 30%

4.3 Leistungsvergleich von R-Bäumen

[Beckmann, Kriegel, Schneider, Seeger 1990]



- Messung der Anzahl der Seitenzugriffe für Aufbau, Point Queries (PQ), kleine und grosse Window Queries (WQ) und Spatial Joins
 - R*-Baum auf 100 normalisiert
- ⇒ R*-Baum ist immer am besten in Bezug auf Anzahl der Seitenzugriffe

- Technik der überlappenden Seitenregionen
 - Rechtecke im Directory können sich überlappen
 - Punkt-Anfrage nicht auf einen Pfad beschränkt
- Rechtecke, die Objekte approximieren (MURs), 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
- R*-Bäume sind die Variante mit dem besten Leistungsverhalten

Aufgabe

- effiziente Verwaltung und Manipulation der exakten Beschreibungen
- exakte Beschreibung eines Geo-Objekts durch Linienzug oder Polygon

Umfeld

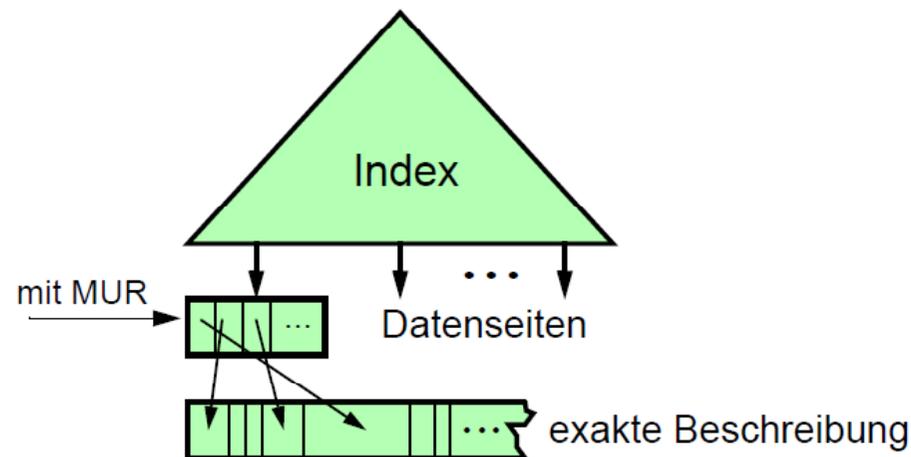
- räumlich benachbarte Objekte werden häufig gemeinsam in einer Anfrage angefordert
- Zugriff auf mehrere physisch benachbarte Seiten ist effizienter als der mehrfache Zugriff auf einzelne (weit voneinander entfernte) Seiten
- einzelne Objekte können sich über mehrere Seiten erstrecken

Ansätze zur Verwaltung der exakten Beschreibungen

- Sekundärorganisation
- Primärorganisation
- Clusterorganisation

Sekundärorganisation

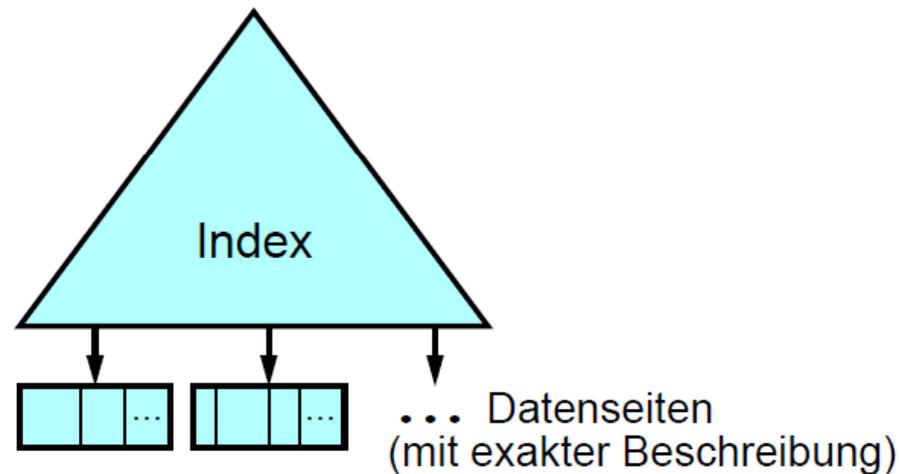
- Räumliche Indexstruktur verwaltet Approximationen (MUR) und Verweise auf exakte Beschreibung (z.B. Polygon)
- Exakte Beschreibung wird unabhängig von räumlicher Indexstruktur verwaltet



- + einfach
- + Trennung zwischen Approximation und exakter Geometrie
- keine räumliche Clusterbildung der exakten Beschreibung
(Einfügezeitpunkt oder andere Aspekte bestimmen Speicherort)

Primärorganisation

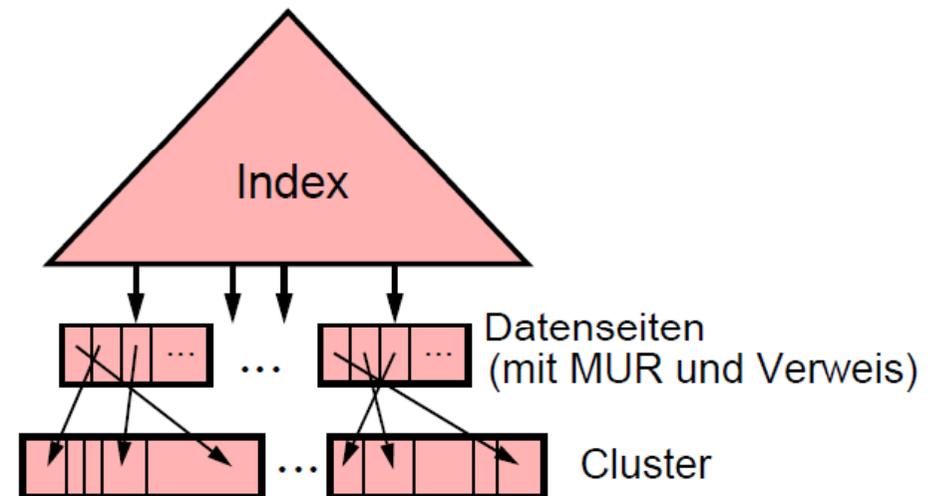
- Räumliche Indexstruktur verwaltet Approximationen *und* exakte Beschreibungen in den Datenseiten



- + räumliche Clusterbildung auf Approximation und exakter Geometrie
- jede Datenseite enthält u.U. deutlich weniger Objekte
- geringer Umfang der Clusterbildung (nur innerhalb einer Seite)
- keine Trennung zwischen Approximation und exakter Geometrie
- Überlaufbehandlung für Objekte, die größer als eine Datenseite sind

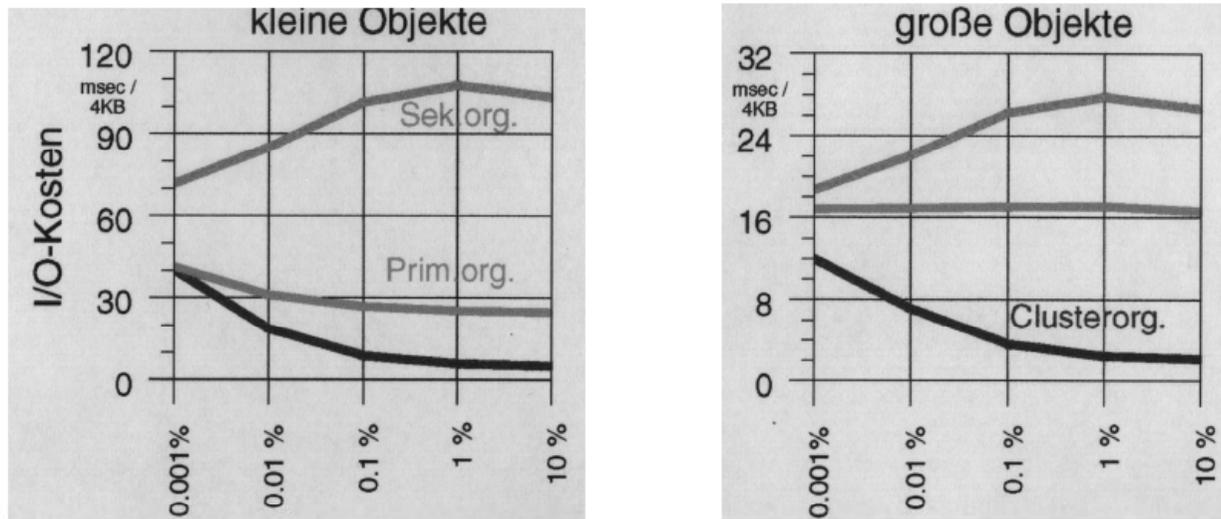
Clusterorganisation

- Die exakte Beschreibung der Objekte, deren MUR in einer Datenseite gespeichert sind, werden auf physisch benachbarten Seiten abgelegt (*Cluster*).
- Die Seiten eines Cluster werden vollständig oder in relevanten Teilmengen eingelesen.



- + räumliche Clusterbildung auf Approximation und exakter Geometrie
- + Trennung zwischen Approximation und exakter Geometrie
- + Clusterbildung für Objekte mehrerer Seiten
- schlechtere Speicherplatzausnutzung als bei Primärorganisation

- Fenster-Anfragen



Fläche des Anfragefensters in % der Fläche des Datenraums

- Spatial Join

