LMU LUDWIG-MAXIMILIANS-UNIVERSITY MUNICH | DEPARTMENT INSTITUTE FOR INFORMATICS | DATABASE SYSTEMS GROUP

Kapitel 3: Räumliche Indexstrukturen

Skript zur Vorlesung
Geo-Informationssysteme
Wintersemester 2011/12
Ludwig-Maximilians-Universität München

4. Räumliche Indexstrukturen

4.1 Einführung (I)

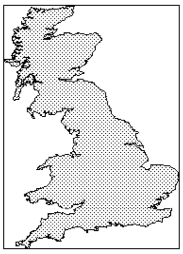
Grundlegende Ideen

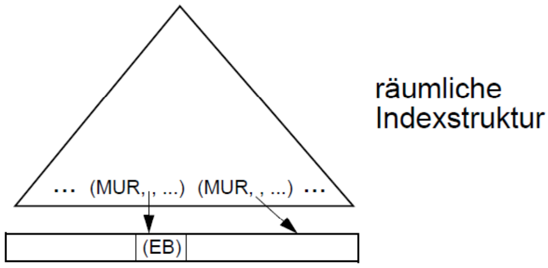
- konventionelle Zugriffsstrukturen sind für die Verwaltung von geometrischen Daten schlecht geeignet

  ⇒ *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

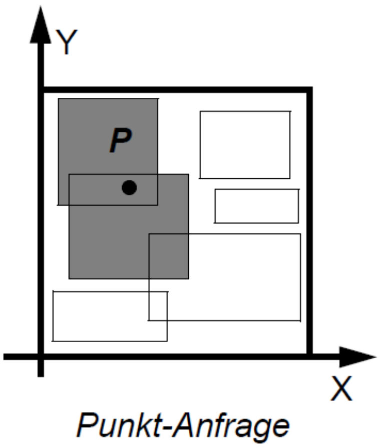


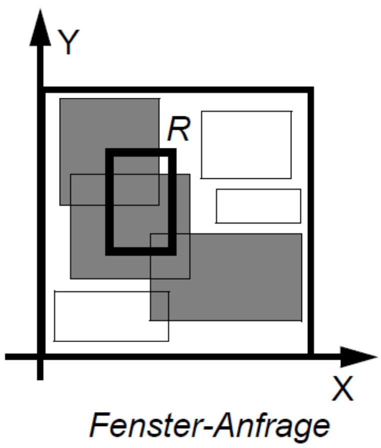


4.1 Einführung (II)

Zu unterstützende Anfragen

- Gegeben ein Anfragepunkt *P* bzw. *ein Anfragerechteck R*
  - *Punkt-Anfrage:* Finde die Geo-Objekte Obj mit P ∈ Obj.MUR.
  - *Fenster-Anfrage:* Finde die Geo-Objekte Obj mit R ∩ Obj.MUR ≠ ∅.

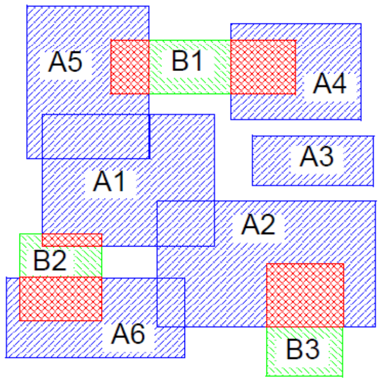




4.1 Einführung (III)

Zu unterstützende Anfragen (Forts.)

- Gegeben zwei Mengen minimal umgebender Rechtecke
  $M1 = \{MUR_{1,1}, MUR_{1,2}, \ldots, MUR_{1,m}\}$
  $M_2 = \{MUR_{2,1}, MUR_{2,2}, \ldots, MUR_{2,n}\}$
- *Spatial-Join: Berechne die Menge*
  $\{ (MUR_1, MUR_2) \mid MUR_1 \in M_1, MUR_2 \in M_2 \text{ und } MUR_1 \cap MUR_2 \neq \emptyset\}$.
  $\Rightarrow$ auch andere räumliche
  Prädikate möglich



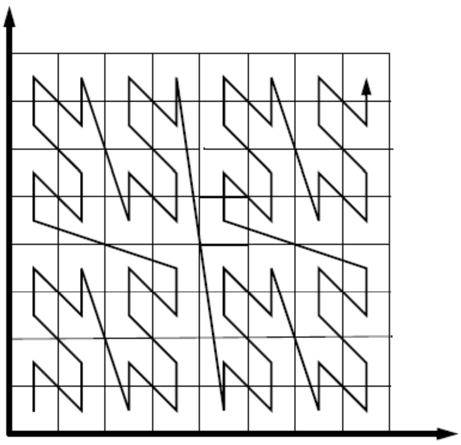
Antwortmenge:
(A5, B1)
(A4, B1)
(A1, B2)
(A6, B2)
(A2, B3)

Spatial-Join

4.1 Prinzipielle Techniken (I)

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)



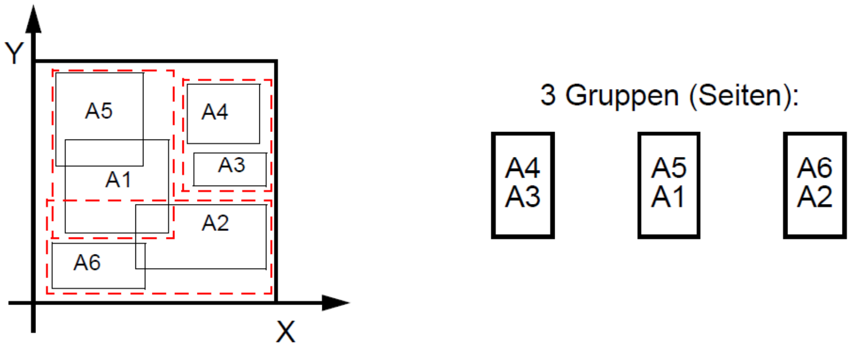
21	23	29	31	53	55	61	63
20	22	28	30	52	54	60	62
17	19	25	27	49	51	57	59
16	18	24	26	48	50	56	58
5	7	13	15	37	39	45	47
4	6	12	14	36	38	44	46
1	3	9	11	33	35	41	43
0	2	8	10	32	34	40	42

Z-Ordnung

4.1 Prinzipielle Techniken (II)

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

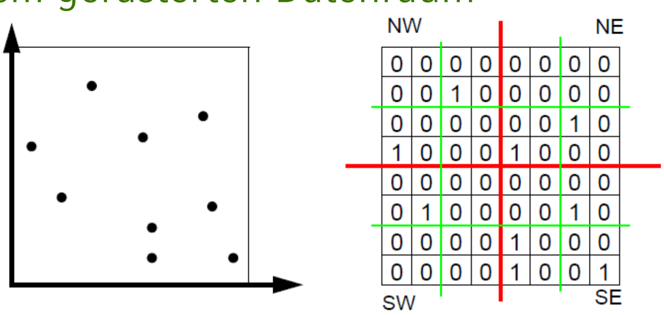


- Im Directory MURs der im Teilbaum enthaltenen MURs abspeichern.
- Split des Datenraums je nach abgespeicherten Daten.
  ⇒ R-Baum

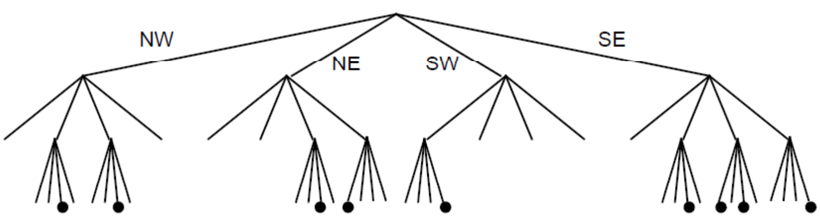
4.1 Prinzipielle Techniken (III)

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

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

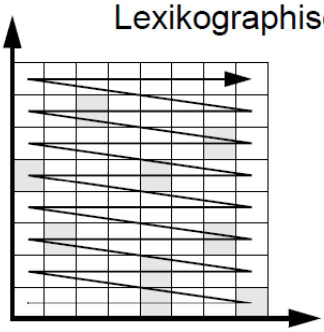


⇒ Quadtrees



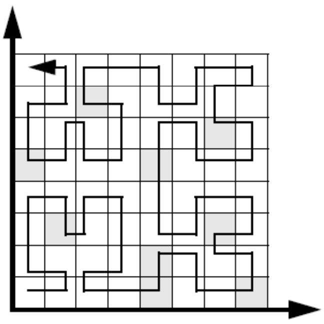
4.2 Raumfüllende Kurven

Lexikographische Ordnung



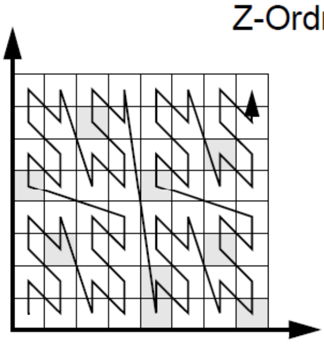
56	57	58	59	60	61	62	63
48	49	50	51	52	53	54	55
40	41	42	43	44	45	46	47
32	33	34	35	36	37	38	39
24	25	26	27	28	29	30	31
16	17	18	19	20	21	22	23
8	9	10	11	12	13	14	15
0	1	2	3	4	5	6	7

Hilbert-Kurve



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

Z-Ordnung

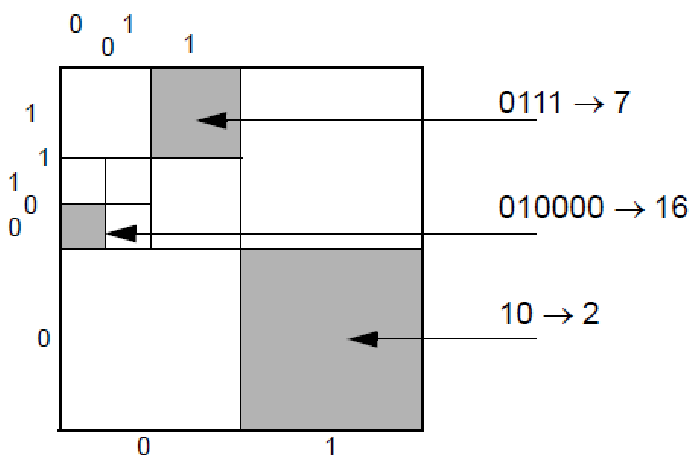


21	23	29	31	53	55	61	63
20	22	28	30	52	54	60	62
17	19	25	27	49	51	57	59
16	18	24	26	48	50	56	58
5	7	13	15	37	39	45	47
4	6	12	14	36	38	44	46
1	3	9	11	33	35	41	43
0	2	8	10	32	34	40	42

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

4.2 Z-Ordnung (I)

Codierung von Zellen



abwechselnd rechts/links und oben/unten partitionieren

Z-Werte

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

DATABASE SYSTEMS GROUP

4.2 Z-Ordnung (II)

LMU

Codierung von Rechtecken (MURs)

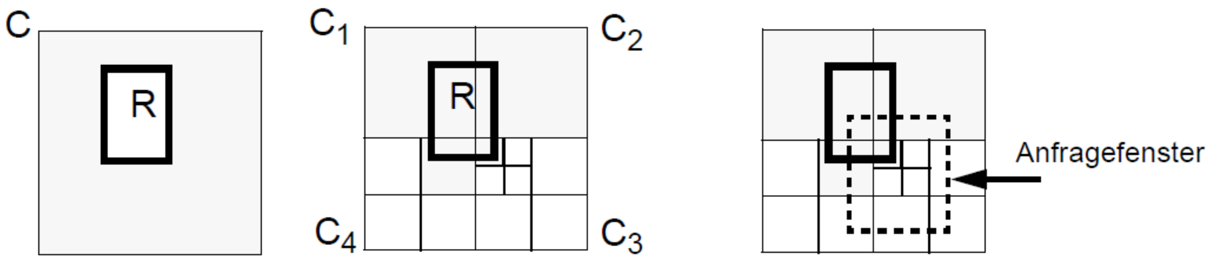
- durch die minimal umgebende Zelle

  ⇒ 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

DATABASE SYSTEMS GROUP

4.2 Z-Ordnung (III)

LMU

Abbildung auf B+-Baum

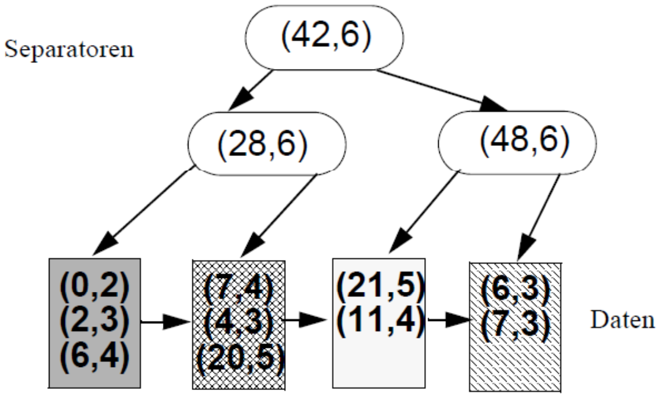
- Lineare Ordnung zur Verwaltung im B+-Baum
  - Seien (c_1, l_1) und (c_2, l_2) zwei Z-Werte und sei $l = \min\{l_1, l_2\}$.
  - Dann ist die Ordnungsrelation $\leq_Z$ wie folgt definiert:

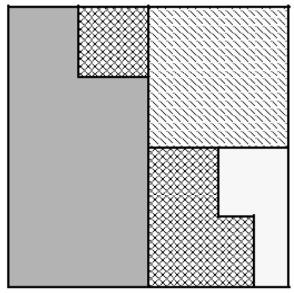
    $(c_1, l_1) \leq (c_2, l_2)$ falls $c_1 \text{div } 2^{l1-l}$ $c_2 \text{div } 2^{l2-l}$
  - Beispiele:

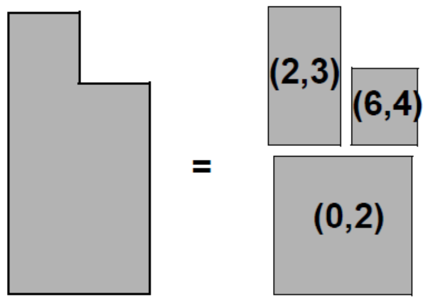
    $(1,2) \leq_Z (3,2)$, $(3,4) \leq_Z (3,2)$, $(1,2) \leq_Z (10,4)$
- Wenn ein Blatt des B+-Baums überläuft, dann Split der Seite in 2 Seiten gemäß B+-Baum Strategie.

4.2 Z-Ordnung (IV)

Beispiel







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

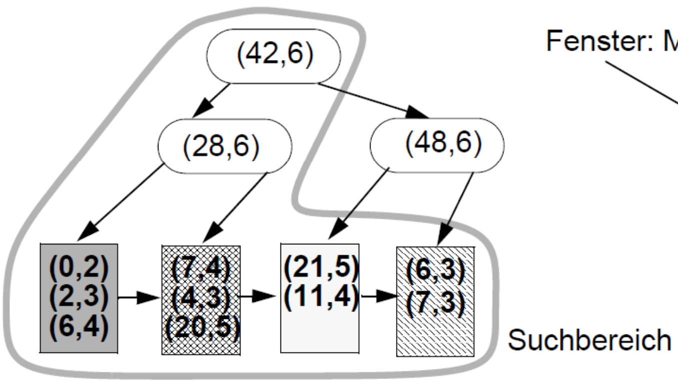
$(2,3) \leq (28,6) \leq (4,3) \leq (42,6)$

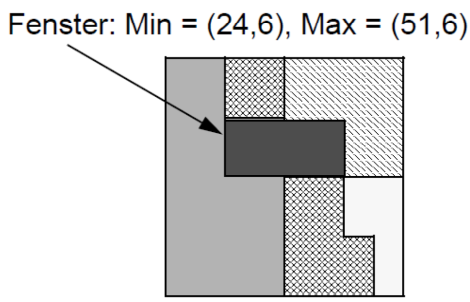
$(6,4) \leq (28,6) \leq (20,5) \leq (42,6)$

4.2 Z-Ordnung (V)

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



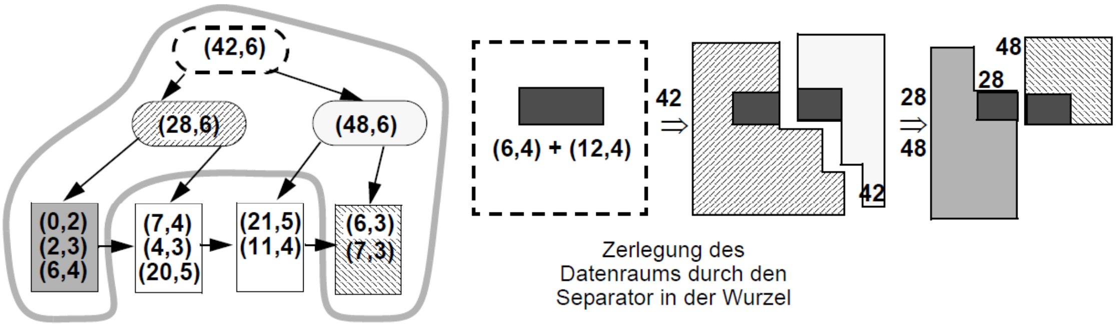


  - ineffizient, da der Suchbereich verglichen mit dem Fenster sehr groß ist

4.2 Z-Ordnung (VI)

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



Zerlegung des Datenraums durch den Separator in der Wurzel

- Mehraufwand für das Durchlaufen der Indexseiten im B^+-Baum (Separatoren)

- Teilbereiche sind nicht notwendigerweise Rechtecke

+ Zugriff nur auf die tatsächlich relevanten Datenseiten

4.3 R-Bäume (I)

Idee

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

4.3 R-Bäume (I)

Definition

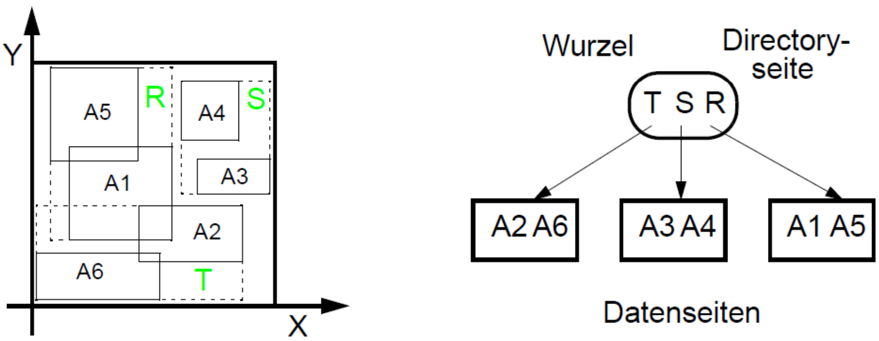
Ein *R-Baum* mit ganzzahligen Parametern *m* und *M*, $2 \leq m \leq \lceil M/2 \rceil$, organisiert eine Menge von Rechtecken in einem Baum mit folgenden Eigenschaften:

- 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.

4.3 R-Bäume (II)

Beispiel

$M = 3$, $m = 2$



Höhe eines R-Baums

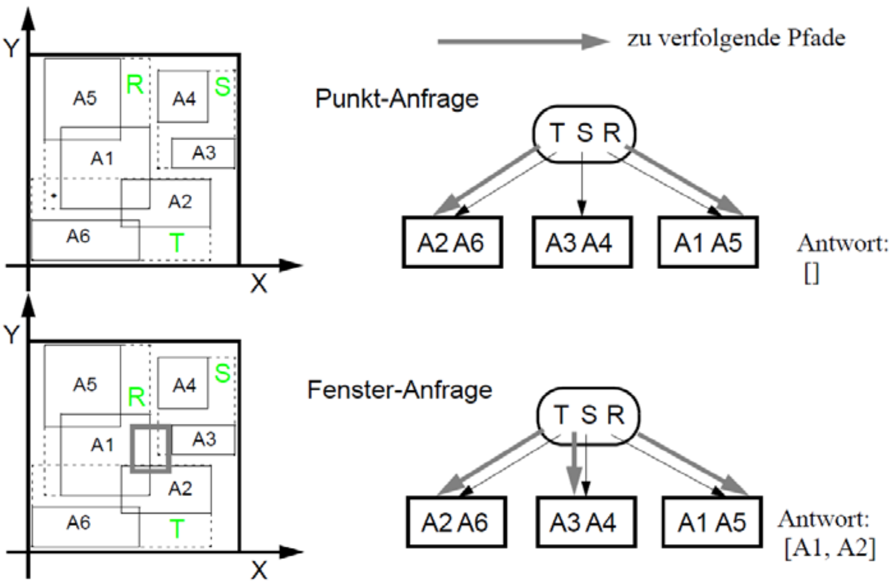
- Ist N die Anzahl der gespeicherten Datensätze, so gilt für die Höhe h des R-Baumes:

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

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

4.3 R-Bäume (III)

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.

4.3 R-Bäume (IV)

Punkt-Anfrage

```
PointQuery (Page, Point);
  FOR ALL Entry ∈ Page DO
    IF Point IN Entry.Rectangle THEN
      IF Page = DataPage THEN
        Write (Entry.Rectangle)
      ELSE
        PointQuery (Entry.Subtree^, Point);
```

4.3 R-Bäume (IV)

Fenster-Anfrage

```
Window Query (Page, Window);
  FOR ALL Entry ∈ Page DO
    IF Window INTERSECTS Entry.Rectangle THEN
      IF Page = DataPage THEN
        Write (Entry.Rectangle)
      ELSE
        WindowQuery (Entry.Subtree^, Window);
```

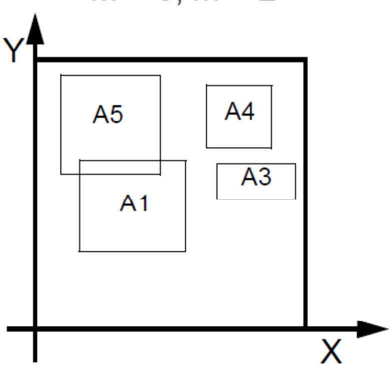

4.3 R-Bäume (V)

Optimierungsziele

- geringe Überlappung der Seitenregionen
- Seitenregionen mit geringem Flächeninhalt
  $\Rightarrow$ geringe Überdeckung von totem Raum
- Seitenregionen mit geringem Umfang

Aufbau

$M = 3$, $m = 2$



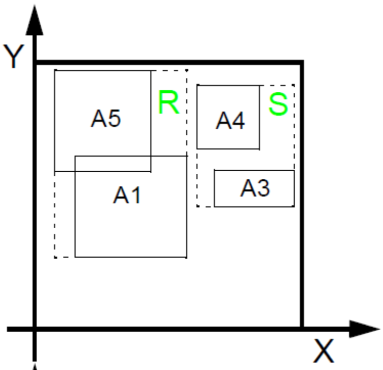
Start: leere Datenseite (= Wurzel)

Einfügen von: A5, A1, A3, A4

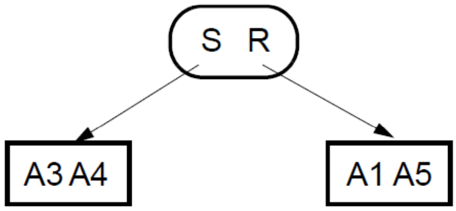
A5, A1, A3, A4 * (Überlauf)

4.3 R-Bäume (VI)

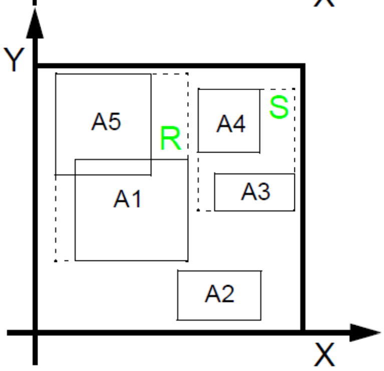
Aufbau (Fortsetzung)

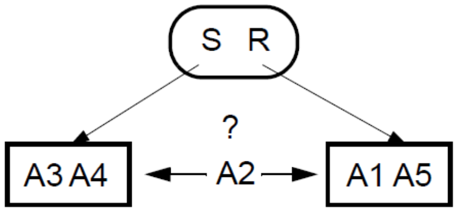


⇒ Split in 2 Seiten



Frage: Wie wird aufgeteilt? (*Splitstrategie*)





Frage: Wo wird eingefügt? (*Einfügestrategie*)

4.3 Einfügestrategie für R-Bäume (I)

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 $D_1, \ldots, D_n$
  ⇒ 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

4.3 Einfügestrategie für R-Bäume (II)

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

4.3 Splitstrategien für R-Bäume (I)

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 ($M \approx 200$)

4.3 Splitstrategien für R-Bäume (II)

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 Ki 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 zufüllen: 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.

4.3 Splitstrategien für R-Bäume (III)

Linearer Algorithmus

- Der lineare Algorithmus ist identisch mit dem quadratischen Algorithmus bis auf die Auswahl des initialen Paars (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 Ri 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.

4.3 Splitstrategien für R-Bäume (IV)

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

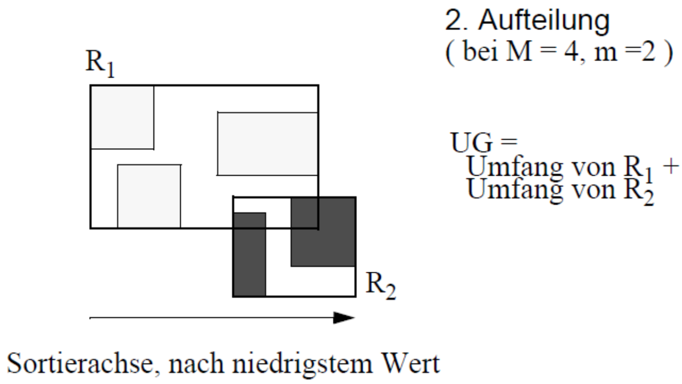
4.3 Splitstrategien für R-Bäume (V)

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.

4.3 Splitstrategien für R-Bäume (VI)



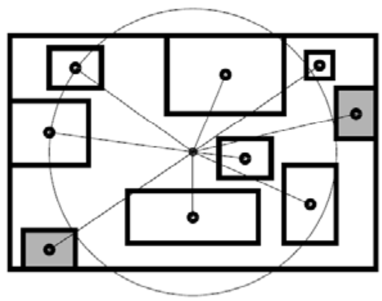
Bestimmung der Aufteilung

- 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*

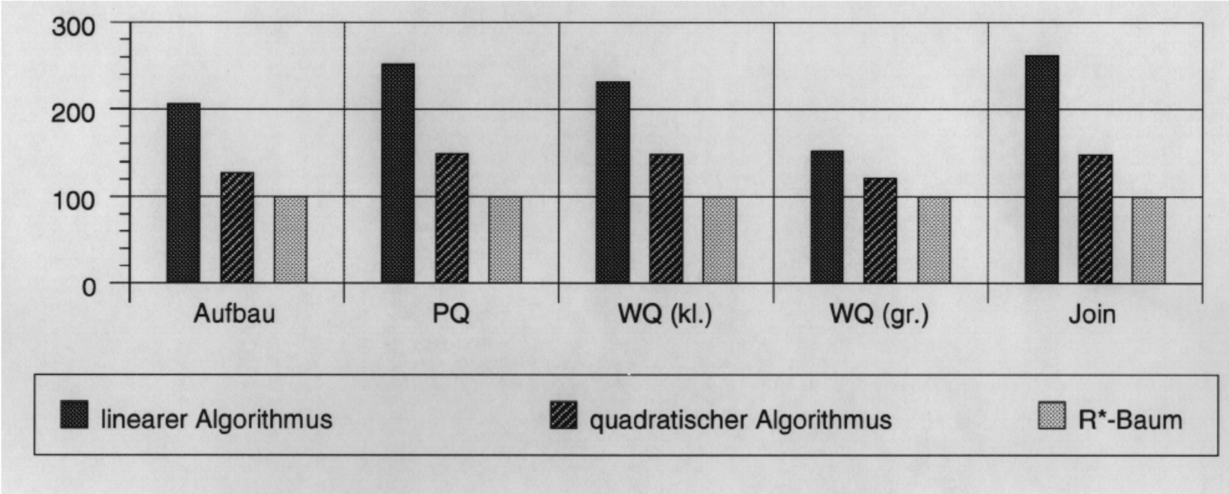
4.3 Vermeidung von Splits (R*-Baum)

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



- Ziele
  - Vermeiden von Splits (nicht immer möglich) ⇒ 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

4.3 R-Bäume: Zusammenfassung

- 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

4.3 Exakte Beschreibungen (I)

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 der
- 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

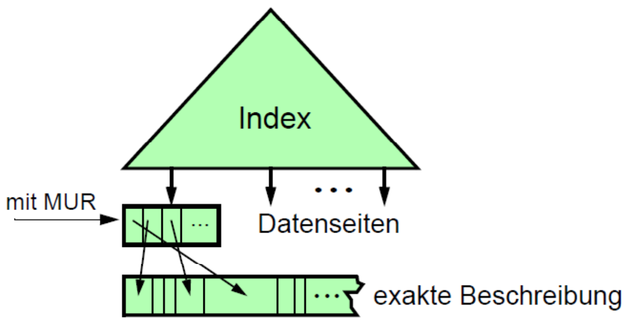
Ansätze zur Verwaltung der exakten Beschreibungen

- Sekundärorganisation
- Primärorganisation
- Clusterorganisation

4.3 Exakte Beschreibungen (II)

Sekundärorganisation

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

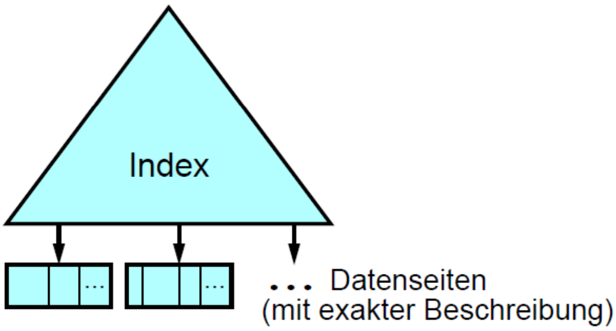


+ einfach
+ Trennung zwischen Approximation und exakter Geometrie
- keine räumliche Clusterbildung der exakten Beschreibung (Einfügezeitpunkt oder andere Aspekte bestimmen Speicherort)

4.3 Exakte Beschreibungen (III)

Primärorganisation

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

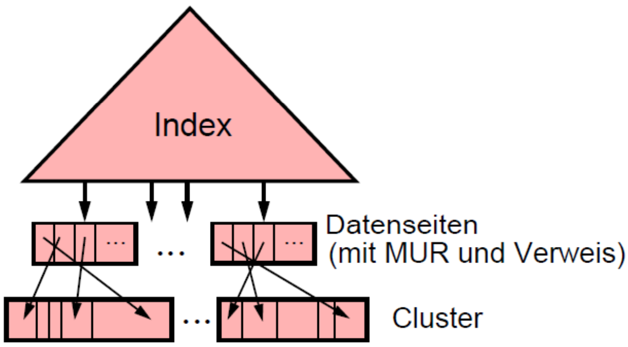


+ räumliche Clusterbildung auf Approximation und exakter Geometrie
- jede Datenseite enthält u.U. deutlich weniger Objekte
- geringer Umfang der Clusterbildung (nur innerhalb einer Seite)
- keine Trennung zwischen Approximation und exakter Geometrie
- Überlaufbehandlung für Objekte, die größer als eine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

4.3 Exakte Beschreibungen (IV)

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.



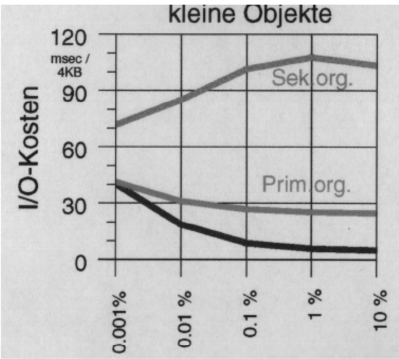
+ räumliche Clusterbildung auf Approximation und exakter Geometrie
+ Trennung zwischen Approximation und exakter Geometrie
+ Clusterbildung für Objekte mehrerer Seiten
- schlechtere Speicherplatzausnutzung als bei Primärorganisation

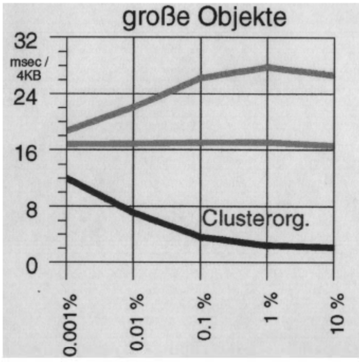
DATABASE SYSTEMS GROUP

4.3 Leistungsvergleich

LMU

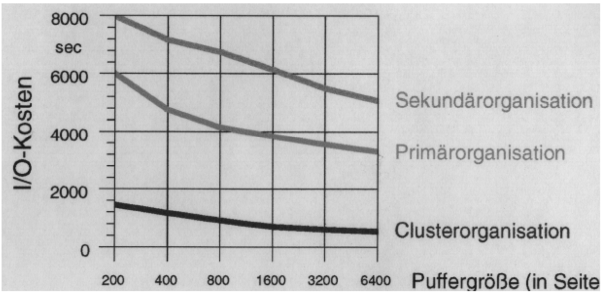
- Fenster-Anfragen





Fläche des Anfragefensters in % der Fläche des Datenraums

- Spatial Join



DATABASE SYSTEMS GROUP

4.4 Quadtrees

LMU

Überblick

- Klasse räumlicher Indexstrukturen, die den Datenraum rekursiv in 4 gleich große
  Zellen unterteilen (*Quadranten NW, NE, SW, SE)*
- Verwaltung von Punkten, Kurven, Flächen usw.
  häufig verwendet in kommerziellen Geo-Informationssystemen
- Weitere Anwendungen: Komprimierung von Rasterbildern, Bildverarbeitung,
  Computergrafik

Literatur

- Samet: *'The Design and Analysis of Spatial Data Structures', Addison-Wesley*, 1990
- Samet: *'Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS', Addison-Wesley, 1990*