

Idee

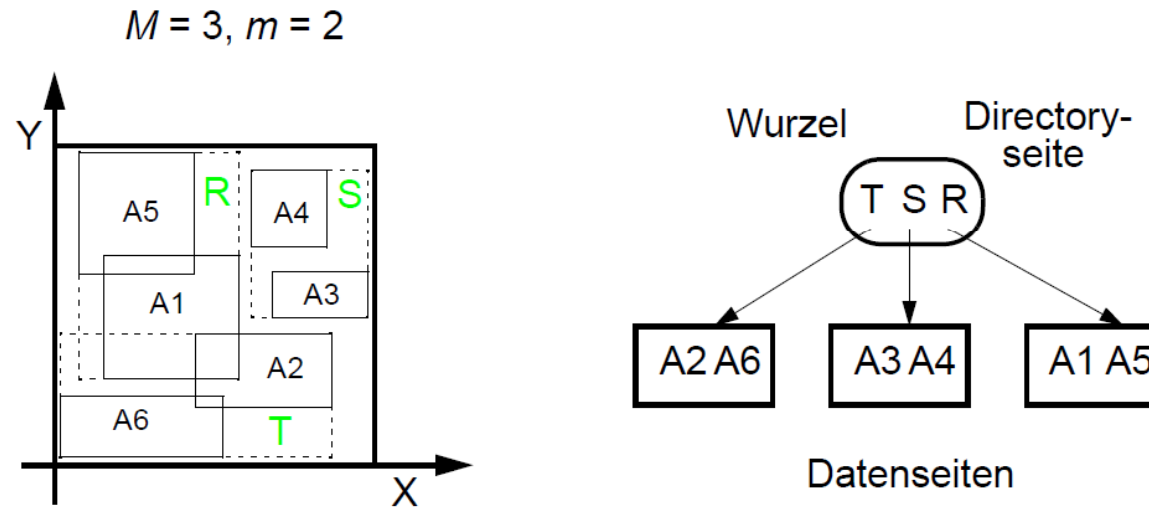
- basiert auf der Technik überlappender Seitenregionen
- verallgemeinert die Idee des B⁺-Baums auf den 2-dimensionalen Raum

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



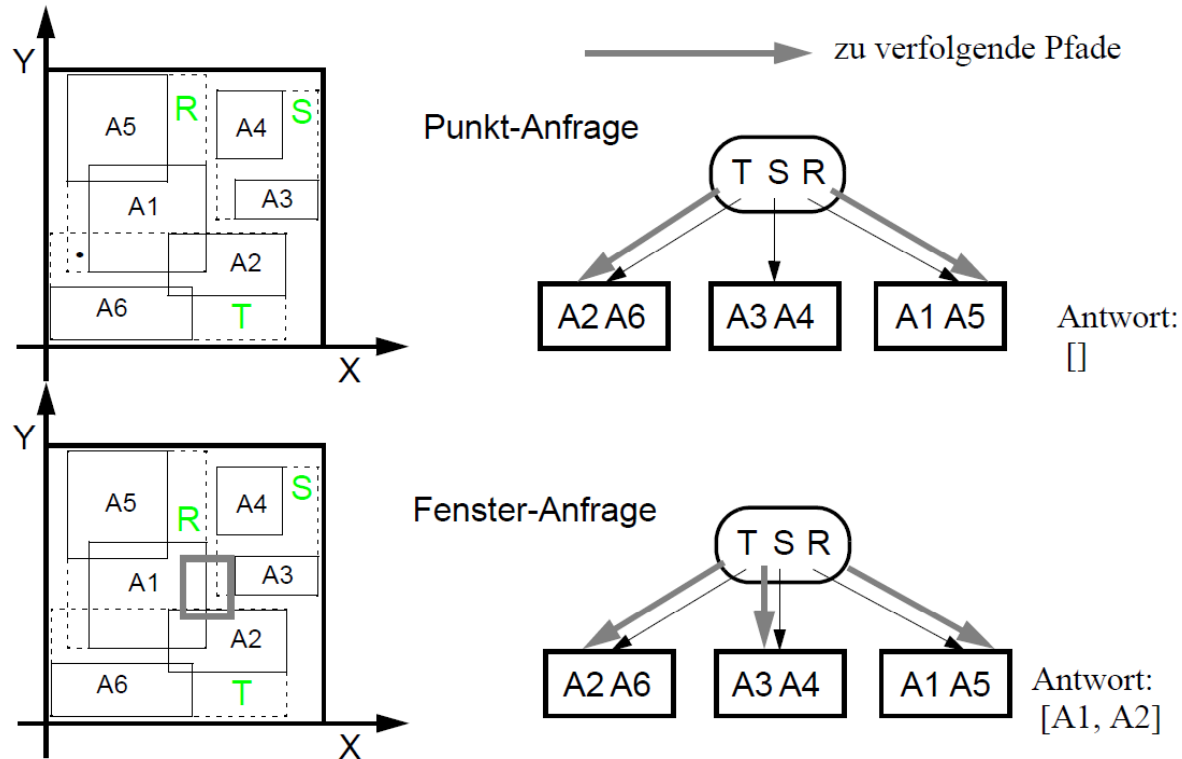
Höhe eines R-Baums

- 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 ∈ 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)

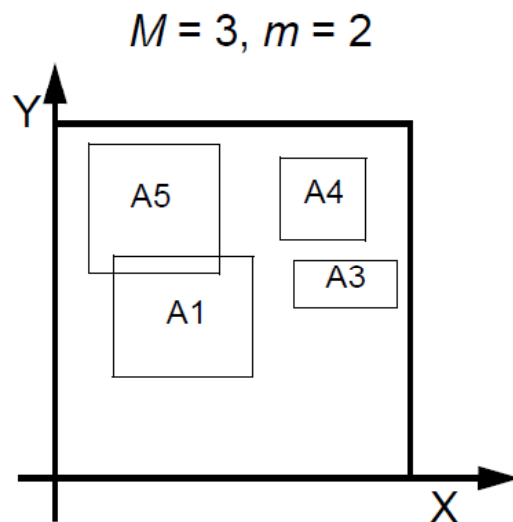
ELSE


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

Optimierungsziele

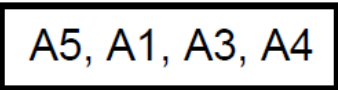
- geringe Überlappung der Seitenregionen
- Seitenregionen mit geringem Flächeninhalt
⇒ geringe Überdeckung von totem Raum
- Seitenregionen mit geringem Umfang

Aufbau

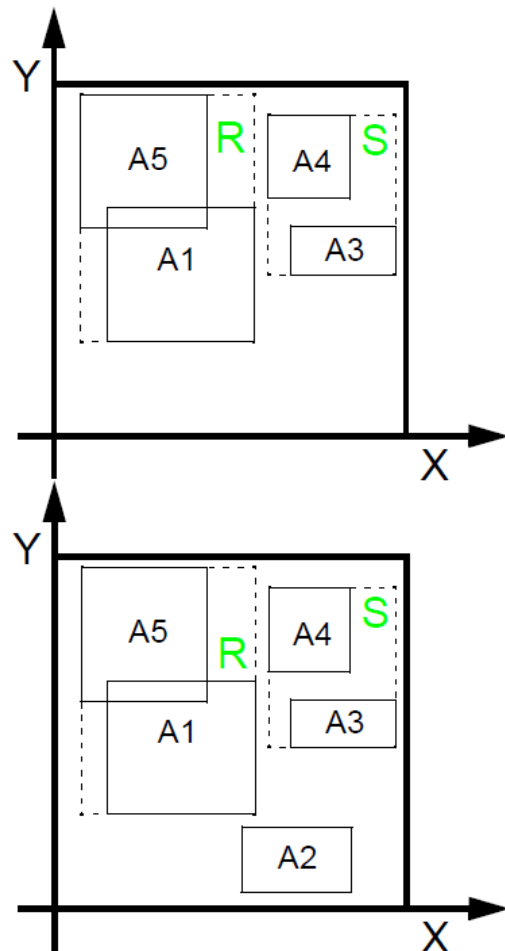


Start:  leere Datenseite
(= Wurzel)

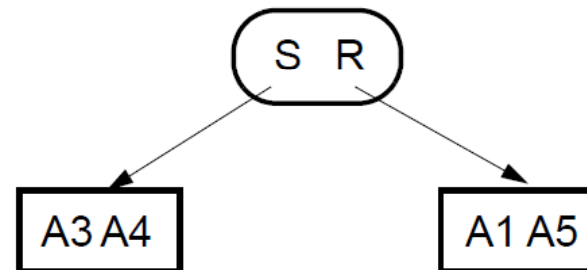
Einfügen von: A5, A1, A3, A4

 * (Überlauf)

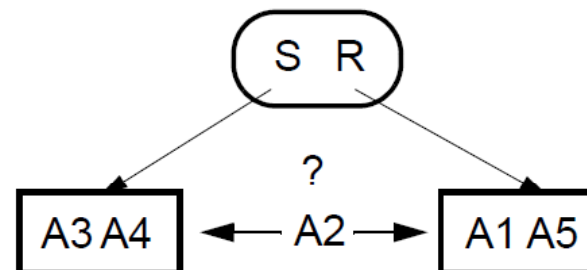
Aufbau (Fortsetzung)



⇒ 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 (Auswahl gemäß kleinster Differenz beider potentiellen Flächenzuwächse von R_1 und R_2) 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

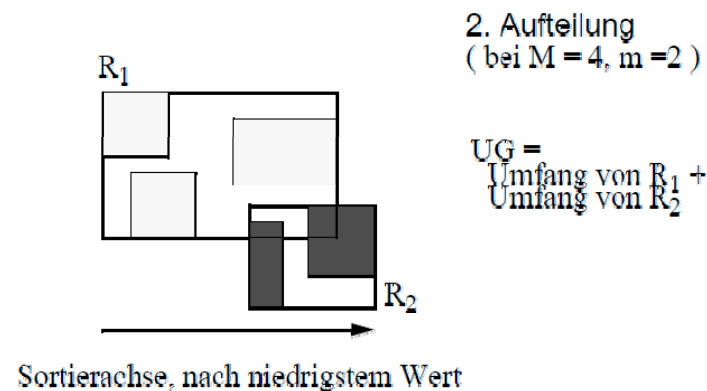
- Der lineare Algorithmus ist identisch mit dem quadratischen Algorithmus bis auf die Auswahl des initialen Paars (R_1, R_2) .
- Wähle das Paar von Rechtecken R_1 und R_2 mit dem "größten Abstand", genauer:
 - Suche für jede Dimension das Rechteck mit dem kleinsten Maximalwert und das Rechteck mit dem größten Minimalwert (*maximaler Abstand*).
 - Normalisiere den *maximalen Abstand* jeder Dimension, indem er durch die Summe der Ausdehnungen aller $R_i \in K$ 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
- Split in 2 Schritten:
 1. Bestimmung der Splitachse
 2. Bestimmung der Splitdimension
- 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.

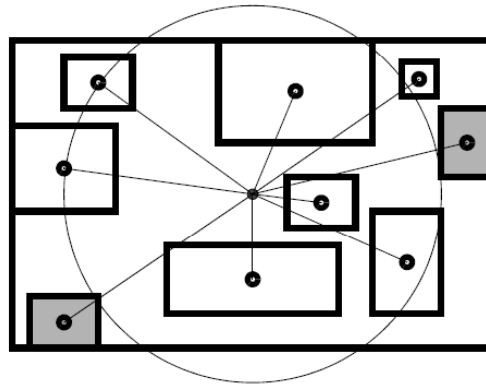
Bestimmung der Aufteilung

- 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.

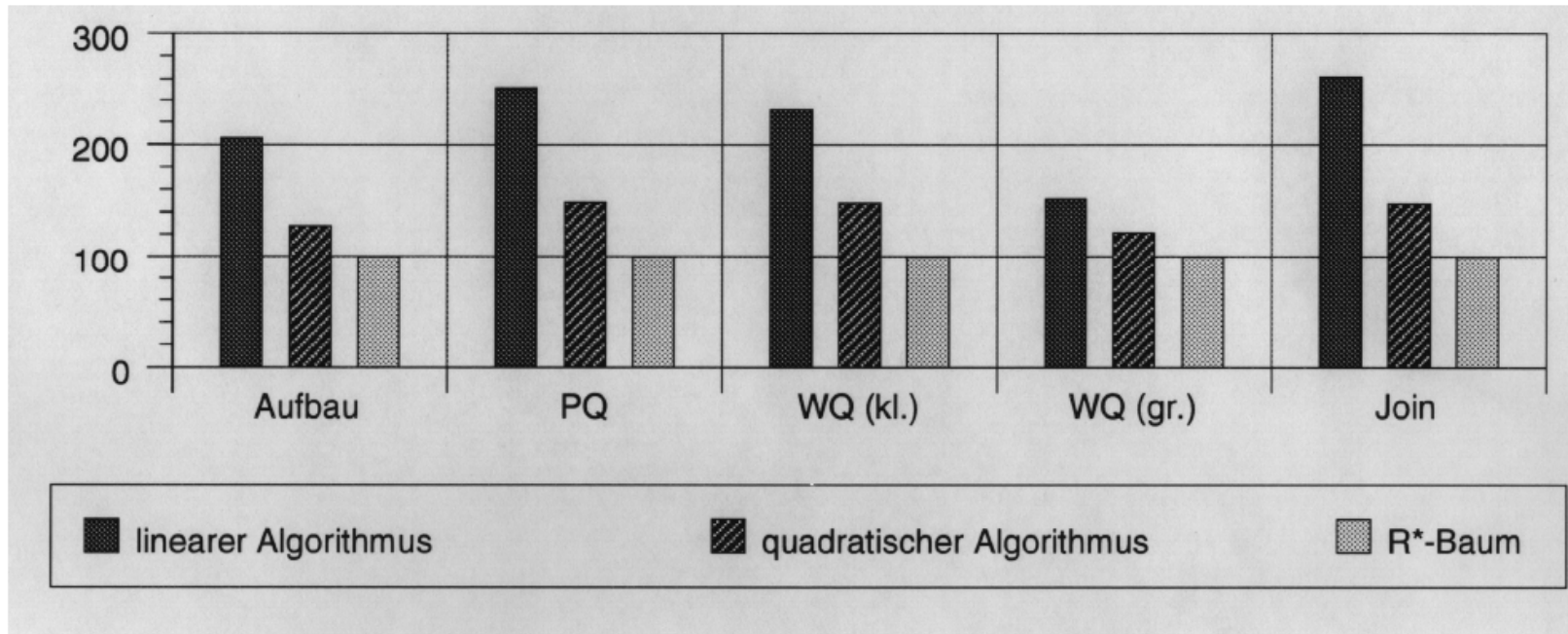
Parameterwahl

Die besten Resultate ergeben sich bei $m = 0,4 * M$
(Experimentell ermittelte Werte)

- 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) \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%



- 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

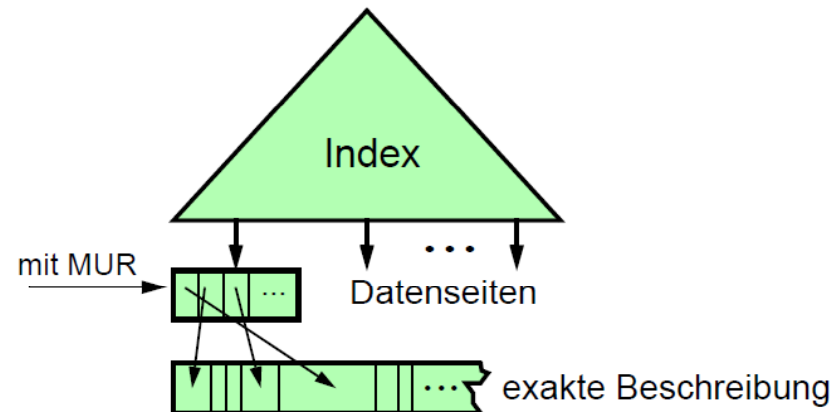
- räumlich benachbarte Objekte werden häufig gemeinsam in einer Anfrage angefordert
- Zugriff auf mehrere physisch benachbarte Seiten ist effizienter als der mehrfache Zugriff auf einzelne (weit voneinander entfernte) Seiten
- einzelne Objekte können sich über mehrere Seiten erstrecken

Ansätze zur Verwaltung der exakten Beschreibungen

- Sekundärorganisation
- Primärorganisation
- Clusterorganisation

Sekundärorganisation

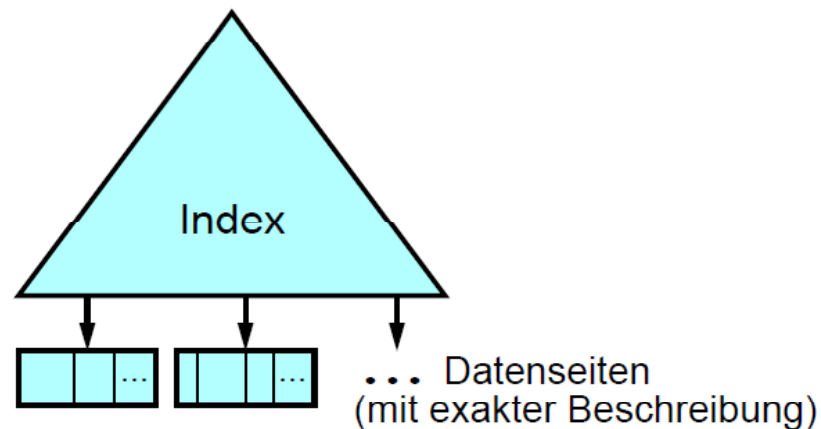
- Räumliche Indexstruktur verwaltet Approximationen (MUR) und Verweise auf exakte Beschreibung (z.B. Polygon)
- Exakte Beschreibung wird unabhängig von räumlicher Indexstruktur verwaltet



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

Primärorganisation

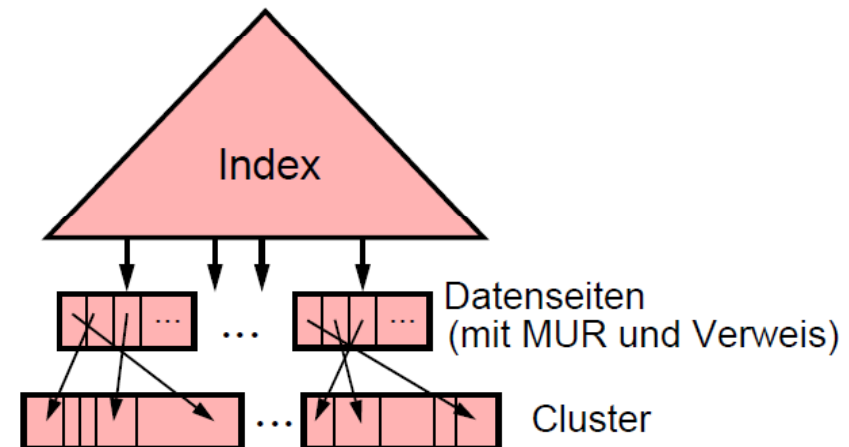
- Räumliche Indexstruktur verwaltet Approximationen *und* exakte Beschreibungen in den Datenseiten



- + 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

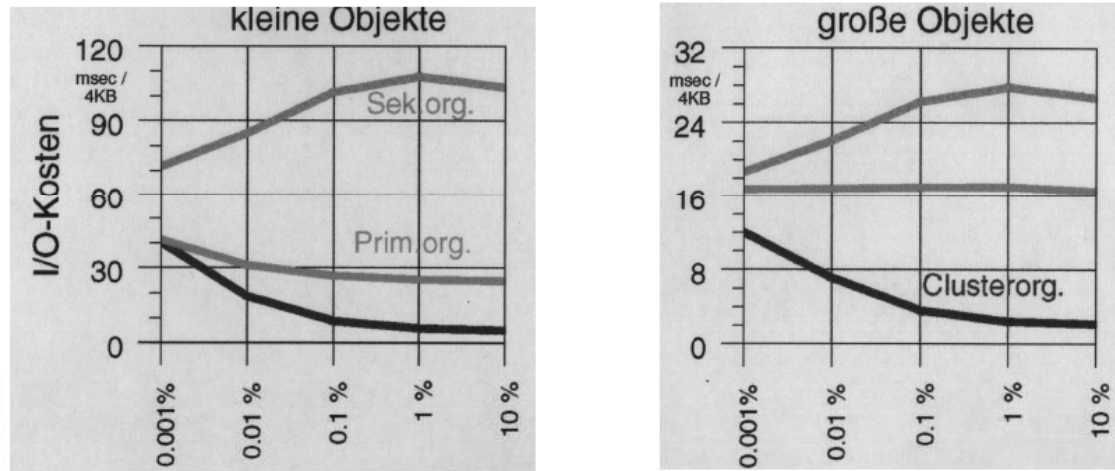
Clusterorganisation

- 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

