

6. Nichtstandard-Zugriffsstrukturen

Übersicht

- ☐ Einführung
- ☐ Mehrdimensionale binäre Bäume
- ☐ Mehrdimensionale externe Zugriffsstrukturen
- ☐ Indexstrukturen für räumliche Daten
- ☐ Indexstrukturen für Zeit

201.

Anwendungen

Verwaltung von geometrischen Daten

- ☐ Relation Cities(Name, Land, Längengrad, Breitengrad)
Anfrage:

```
SELECT *
FROM Cities
WHERE (Längengrad > -11) AND (Längengrad < 3) AND
      (Breitengrad > 35) AND (Breitengrad < 45)
```

 Antworten:
 (Madrid, E, 40, -4)
 (Barcelona, E, 42, 2)
 ...

- ☐ Daten werden als Punkte in einem 2-dimensionalen Datenraum betrachtet.
- ☐ Anfragebedingung ist auf beiden Dimensionen spezifiziert.

203.

6.1 Einführung

- ☐ Anwendungen, Anforderungen u. Grundprobleme
 - Geo-Datenbanken, Versionsdaten, Verwaltung einfacher Regeln
 - Erhaltung der Topologie (Clusterbildung)
- ☐ Klassischer Ansatz: Invertierte Listen
- ☐ Binäre Bäume
 - kd-Baum, kd-trie, BD-Baum
- ☐ Vier Klassen von Verfahren:
 - mit einer vollständigen, nicht-überlappenden, homogenen Partitionierung
 - Erweiterung von Hash-Verfahren: Grid-File & PLOP-Hashing
 - KDB-Baum
 - mit einer unvollständigen Partitionierung: Buddy-Baum
 - mit einer inhomogenen Partitionierung
 - ZB+-Baum (raumfüllende Kurven)
 - Erweiterung des BD-Baums: BANG-File u. hB-Baum
 - mit einer unvollständigen und überlappenden Partitionierung
 - R-Baum und Varianten

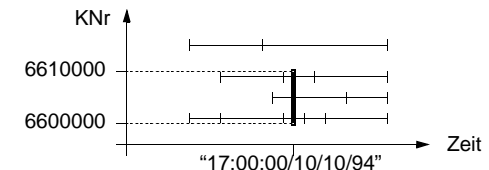
202.

Verwaltung von Versionsdaten

- ☐ Relation Konto(KNr, Name, Stand, Von, Bis)
Anfrage:

```
SELECT *
FROM Konto
WHERE (Von < "16:59:59/10/10/94") AND ("17:00:00/10/10/94" < Bis) AND
      (KNr like "660_____");
```

Antworten:
 (6602222, Schmidt, +20.000, "13:45:32/10/10/94", "10:03:31/14/10/94")
 (6603456, Müller, -33.000, "11:20:44/08/10/94", "09:10:28/11/10/94")
 ...



204.

Verwaltung von einfachen Regeln:

- Produktionsregeln:
 - IF *condition* THEN *action*
 - werden nach dem Einfügen eines Datensatzes überprüft

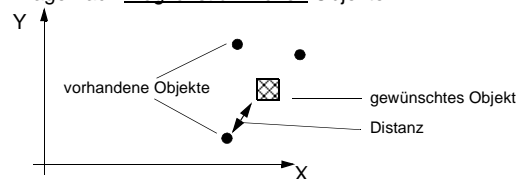
- Beispiel:
Relation Emp(Name, Alter, Gehalt, Abteilung)

IF *IsOdd*(Emp.Alter) THEN Emp.Gehalt = Emp.Gehalt*1.02;
 IF (Emp.Gehalt < 20.000 AND Emp.Alter > 45) THEN *ErrorMsg*("...");
 IF (20.000 < Emp.Gehalt < 30.000) THEN Emp.Gehalt = Emp.Gehalt + 1000;

205.

Nächster-Nachbar-Anfragen (similarity, nearest neighbor query)

- gewünschtes Objekt nicht vorhanden. Stattdessen:
Frage nach möglichst ähnlichen Objekten



- "best" wird bestimmt über verschiedene Arten von Distanzfunktionen, z. B.
 - Objekt erfüllt nur 8 von 10 geforderten Eigenschaften
 - Objekt ist durch Synonyme beschrieben
- nearest neighbor:
 D = Distanzfunktion
 B = Sammlung von Punkten im k -dim. Raum
 Gesucht: nächster Nachbar von p (in B). Der nächste Nachbar ist q , wenn
 $(\forall r \in B) (r \neq q \Rightarrow D(r,p) \geq D(q,p))$

207.

Effiziente Unterstützung von Anfragen

- Relation R mit k (ausgewählten) Attributen A_1, \dots, A_k .
- $$D = \prod_{i=1}^k \text{dom}(A_i)$$

Anfragen:

- exact match query:
Gegeben $(x_1, \dots, x_k) \in D$. Suche den Datensatz $r \in R$ mit $r.A_1 = x_1, \dots, r.A_k = x_k$.
- partial match query:
Gegeben $m \leq k$ und Index (i_1, \dots, i_m) mit $1 \leq i_1 \leq \dots \leq i_m \leq k$. Gegeben $(x_{i_1}, \dots, x_{i_m})$ mit $x_{i_j} \in \text{dom}(A_{i_j})$. Suche alle Datensätze $r \in R$ mit $r.A_{i_1} = x_{i_1}, \dots, r.A_{i_m} = x_{i_m}$.
- window query (Bereichsanfrage):
Gegeben $(u_1, \dots, u_k), (o_1, \dots, o_k) \in D$ mit $u_i \leq o_i$. Suche alle Datensätze $r \in R$ mit $u_1 \leq r.A_1 \leq x_1, \dots, u_k \leq r.A_k \leq x_k$.

206.

Anforderungen

- effiziente Unterstützung der obengenannten Anfragen
- effizientes Einfügen und Löschen
 - dynamische Reorganisation der Datensätze
- alle Dimensionen des Datenraums sollen **gleich** behandelt werden
- Leistung soll unabhängig von der Daten- und Anfrageverteilung sein

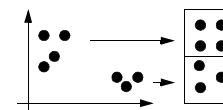


Abbildung der Datensätze auf Seiten

- Datensätze, die oft gemeinsam die gleiche Anfrage erfüllen, sollen möglichst gemeinsam in einer Seite abgespeichert werden.
 \Rightarrow im Datenraum nah beieinander liegende DS sollen gemeinsam in einer Seite liegen.
 \Rightarrow hohe Speicherplatzausnutzung

208.

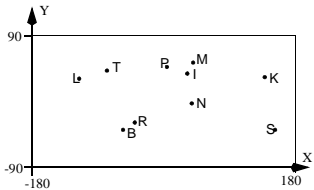
Prinzipielle Vorgehensweise

- ❑ Zerlege den mehrdimensionalen Datenraum D in Regionen. Jede Region R gehört zu einer Datenseite P, so daß alle Datensätze aus P in der Region R liegen müssen.
- ❑ Klassifizierung der Verfahren kann bzgl. der Eigenschaften der Regionen vorgenommen werden:

	homogen	vollständig	disjunkt	Verfahren
C1	X	X	X	PLOP-Hashing, Grid-File
C2	X	X		Twin Grid-File
C3	X			R-Baum
C4	X		X	R+-Baum, Buddy-Baum
C5		X	X	BANG-File, ZB+-Baum

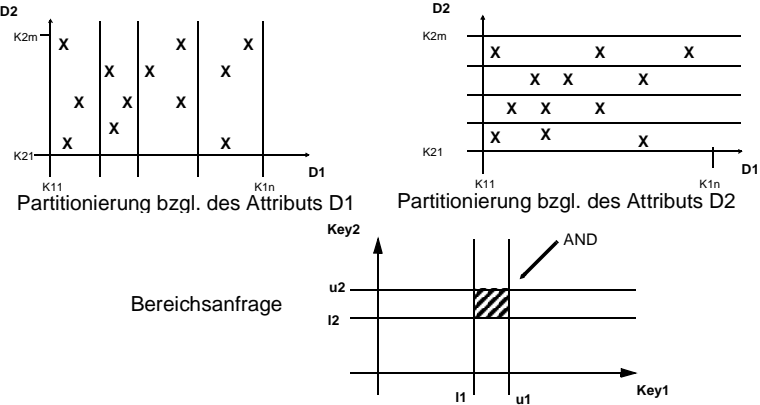
Beispiels-Datenbank

	x-Koord.	y-Koord.
Sydney	150	- 36
Buenos Aires	- 58	- 36
Rio de Jaineiro	- 42	- 26
Los Angeles	-118	34
Toronto	- 80	45
Paris	2	50
Moskau	38	56
Kyoto	136	36
Nairobi	36	0
Istanbul	29	41



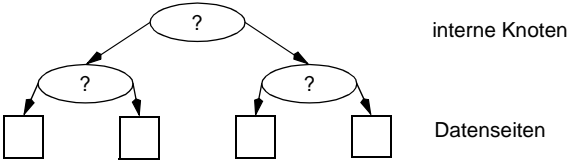
Invertierte Listen

- ❑ merdimensionale Zugriffsstruktur realisiert durch k eindimensionale Strukturen
- ❑ Beispiel (k=2):



6.2 Mehrdimensionale Binäre Bäume

- ❑ jeder interne Knoten enthält höchstens zwei Verweise auf Teilbäume.
- ❑ Blattknoten entsprechen Datenseiten mit einer Kapazität von b Datensätzen.

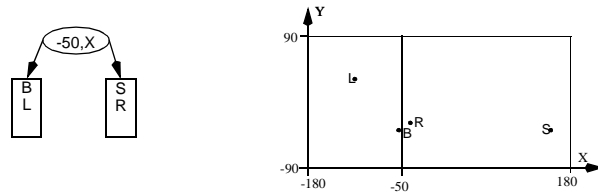


- ❑ binäre Bäume sind primär für den Hauptspeicher entwickelt worden und sind i.a. nicht effizient für den Externspeicher verwendbar.
- ❑ im Folgenden werden die drei wichtigsten Varianten vorgestellt:
 - adaptiver kd-Baum (Frieman, Bentley & Finkel, 1977)
 - kd-trie (Orenstein, 1982)
 - BD-Baum (Ohsawa & Sauki, 1983)

6.2.1 Adaptiver kd-Baum

Idee:

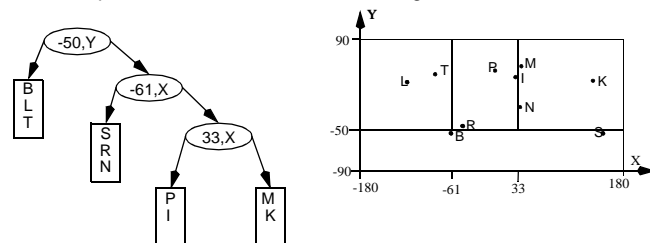
- Baum besteht zunächst aus einer Seite.
- Überläufer werden folgendermaßen eliminiert:
 - Zunächst wird eine Splitachse und ein Splitwert bestimmt.
 - Datensätze links des Splitwerts verbleiben in der Seite, alle anderen werden in eine neu reservierte Seite kopiert.
- nach dem Einfügen der ersten vier Datensätze:



213.

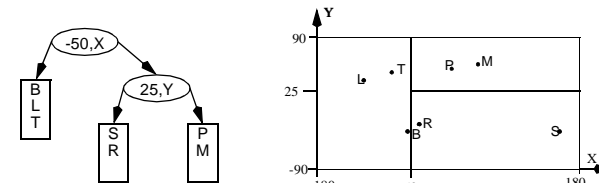
Eigenschaften

- kd-Baum ist nicht balanciert (worst case: $O(n)$ Baumhöhe bei n Datensätzen)
- es gibt kein Splitwert, der die Datensätze gleichmäßig in zwei Gruppen aufteilt
- die Eingabenreihenfolge beeinflusst die Unterteilung des Datenraums
- die Wahl der Splitachse bestimmt die Unterteilung des Datenraums

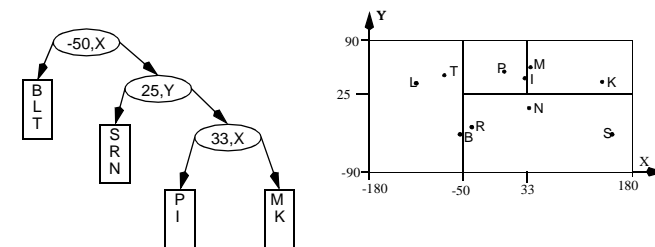


215.

- Einfügen von Toronto, Paris und Moskau



- nachdem alle Datensätze eingefügt sind

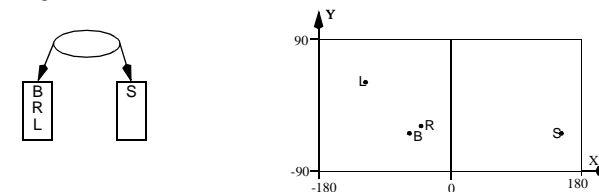


214.

6.2.2 kd-trie

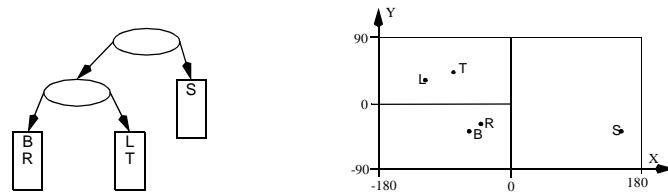
Idee:

- ähnlich der des kd-Baums, jedoch ist die Splitachse und der Splitwert vorbestimmt. Die Splitachse durchläuft zyklisch die Dimensionen des Datenraums; der Splitwert halbiert die Regionen.

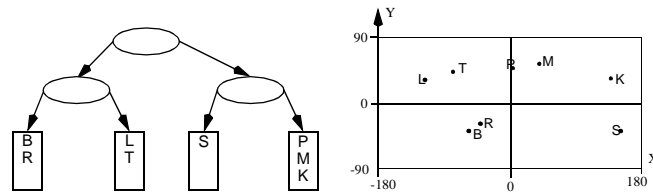


216.

- Einfügen von Toronto führt wieder zu einem Überlauf



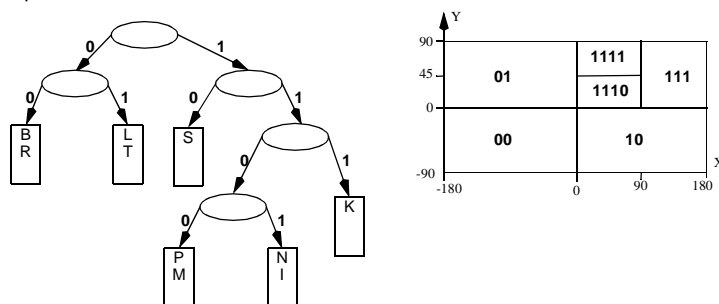
- Einfügen von Paris, Moskau und Kyoto



217.

Z-String

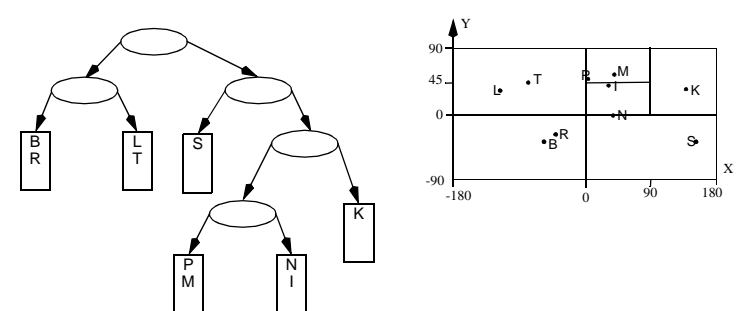
- jede durch den kd-trie erzeugte Region kann durch eine Folge von Bits eindeutig repräsentiert werden.



- Z-String (b_0, b_1, \dots, b_{L-1}) kann durch zwei Parameter eindeutig beschrieben werden:
- L = Länge des Strings (**Level**)
 - ganzzahlige Interpretation des Strings: $b_0 \cdot 2^{L-1} + b_1 \cdot 2^{L-2} + \dots + b_{L-1} \cdot 2^0$ (**Z-Wert**)

219.

- nachdem alle Datensätze eingefügt sind:



Eigenschaften:

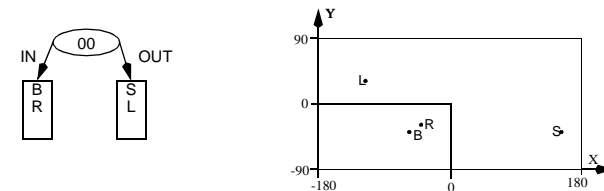
- kd-trie kann auch unbalanciert sein. Jedoch ist die Höhe des Baums durch die Auflösung des Datenraums beschränkt.
- Seiten können im schlechtesten Fall nur einen Datensatz enthalten.
- Unterteilung des Datenraums wird **nicht** durch die Eingabereihenfolge bestimmt!

218.

6.2.3 BD-Baum

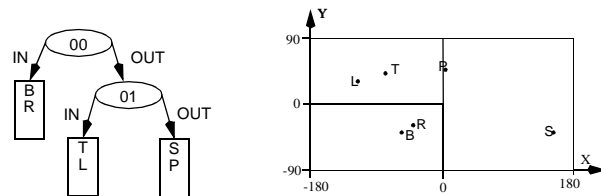
Idee:

- BD-Baum kann als Erweiterung des kd-trie angesehen werden. Der Split einer Seite erzeugt (durch zyklisches Halbieren) eine Region, die zwischen $1/3$ und $2/3$ der ursprünglichen Datensätze enthält.
 1. Führe zunächst den Split nach dem Verfahren des kd-trie durch.
 2. Erfüllt eine dieser Regionen die gewünschte Eigenschaft, STOP.
 3. Andernfalls (d.h. es gibt eine Region mit mehr als $2/3$ der Datensätze), halbiere diese (zu mehr als $2/3$ gefüllte) Region in zwei Regionen.
- BD-Baum und Partitionierung des Datenraums nach den ersten 4 Datensätzen:

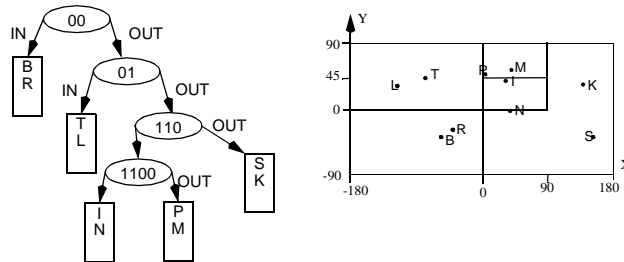


220.

nach zwei weiteren Datensätzen:



BD-Baum nachdem alle Datensätze eingefügt sind:



221.

Eigenschaften binärer Suchbäume

- so lange der Index klein ist, eignen sich binäre Bäume für die Verwaltung der Daten
 - mit einem Index von 16 KB und einem Speicherbedarf von 10 Bytes pro Eintrag, lassen sich etwa 1600 Seiten verwalten.
- mehrdimensionale binäre Bäume geben keine Garantie für ein logarithmisches Wachstum des Index.
- in mehreren Experimenten wurde aber beobachtet, daß die Tiefe des BD-Baums am niedrigsten ist.

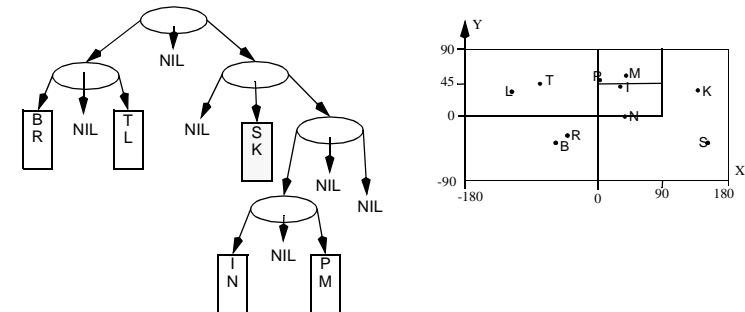
Zentrale Frage:

Wie können die Strategien der binären Bäume auf Externspeicherstrukturen übertragen werden?

223.

Alternative Darstellung eines BD-Baums

- ein BD-Baum kann als kd-tree dargestellt werden, wobei zusätzlich in jedem internen Knoten eine "Restseite" installiert ist.
- Die Restseite enthält alle Datensätze aus der durch den Knoten repräsentierten Region, die nicht in den zugehörigen Teilbäumen abgelegt sind.



222.

6.3 Mehrdimensionale Zugriffsstrukturen für den Externspeicher

- b = Kapazität einer Datenseite
- Finde geeignete Abbildung des Directory auf Externspeicherseiten

Ziel beim Entwurf

- entwerfe ein Verfahren mit den gleichen Eigenschaften

Verfahren

- basierend auf den Prinzipien des B^+ -Baums
 - KDB-Baum
 - ZB+-Baum
 - BANG-File & Varianten
- basierend auf den Prinzipien von Hash-Verfahren
 - Grid-File & PLOP-Hashing

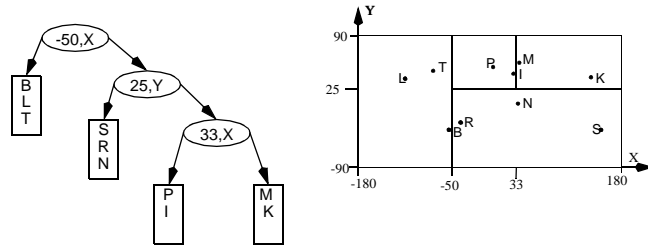
224.

6.3.1 KDB-Baum

- basiert auf der Strategie des kd-Baums
- bildet die Zwischenknoten des kd-Baums auf Externspeicherseiten ab.

Beispiel:

- Annahme:
höchstens 3 Referenzen können in einem Zwischenknoten des KDB-Baums sein



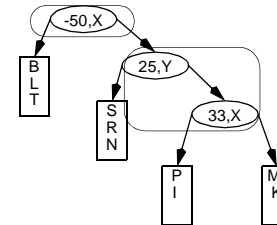
- Frage: Wie werden die 4 Seitenreferenzen auf zwei Seiten aufgeteilt?

225.

Lösung 1:

Benutze die Wurzel des kd-Baums für das Aufteilen der Referenzen:

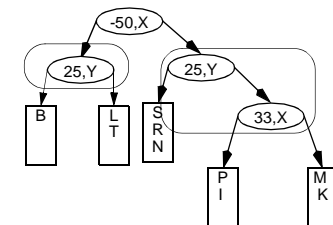
- alle Referenzen im linken Teilbaum verbleiben in der alten Seite
- alle Referenzen im rechten Teilbaum werden in eine neue Seite umgespeichert
- Indexeintrag in der Wurzel des kd-Baums kommt in den Elternknoten.



- Nachteil: schiefe Aufteilung

Lösung 2:

Erzeuge neue Knoten (und Referenzen), so daß eine gleichmäßige Aufteilung möglich ist.



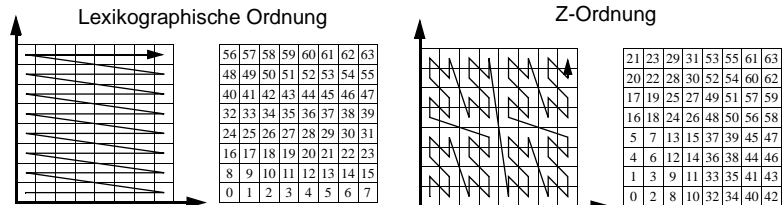
- Nachteil: viele unterfüllte Seiten

226.

6.3.2 ZB+-Baum

Idee:

- Bilde einen k-dimensionalen Punkt in einen eindimensionalen Raum ab, so daß die mehrdimensionale Ordnung möglichst bewahrt bleibt.
- Möglichkeiten bei der Abbildung



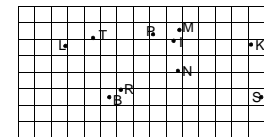
- Verwalte die mehrdimensionalen Daten in einem B+-Baum unter Verwendung der eindimensionalen Ordnung.

227.

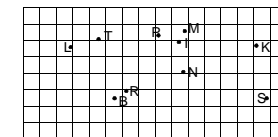
- Eindimensionale Ordnung der Datensätze unseres Standardbeispiels

Lexikographische Ordnung

Z-Ordnung



B-R-S-N-L-I-K-T-P-M



B-R-L-T-S-N-I-P-M-K

228.

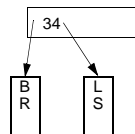
Berechnung der Z-Werte

- Vereinfachende Annahme: $k = 2$ (Dimensionen), $D = \{0, \dots, 2^n - 1\} \times \{0, \dots, 2^n - 1\}$
- Sei $(x, y) \in D$. Dann gibt es $a_i, b_i \in \{0, 1\}$, so daß $x = \sum_{i=0}^{n-1} a_i \cdot 2^i$ und $y = \sum_{i=0}^{n-1} b_i \cdot 2^i$.
- Der **Z-Wert** (zum Level $2n$) ist dann durch $Z(x, y) = \sum_{i=0}^{n-1} a_i \cdot 2^{2i+1} + \sum_{i=0}^{n-1} b_i \cdot 2^{2i}$ definiert.
- Der **Z-Wert zum Level L** , $0 \leq L < 2n$, ist definiert als $Z((x, y), L) = Z(x, y) \div 2^{2n-L}$
- Der Z-Wert zum Level L läßt sich rekursiv berechnen. Es gilt: $Z((x, y), L) = 2 \times Z((x, y), L-1) + c_{L-j}$ wobei c_{L-j} das $(2n-L)$ -te Bit des Z-Werts ist.

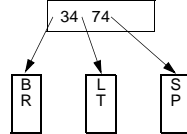
229.

- Verwaltung der Daten durch ihre Z-Werte in einem B+-Baum
 - Die mehrdimensionalen Daten werden wie gewöhnlich in den Blättern des B+-Baums verwaltet.
 - Als Trennschlüssel in den inneren Knoten werden nun Z-Werte verwendet.

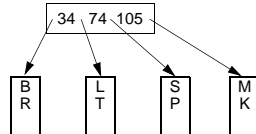
Einfügen von S, B, R, L



Einfügen von T, P



Einfügen von M, K



- Statt eines kompletten Z-Werts können in den inneren Knoten auch nur die Z-Werte eines Levels j abgespeichert werden.

231.

Beispiel

- Sei $n = 4$, $P = (1, 6)$. Dann gilt

$$x = (0 \ 0 \ 0 \ 1)_2$$

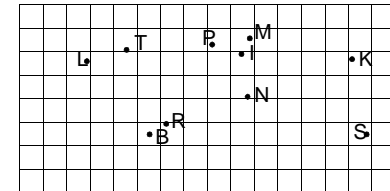
$$y = (0 \ 1 \ 1 \ 0)_2$$

$$Z(P) = (0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0)_2 = (22)_{10}$$

$$Z(P, 6) = (0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0)_2 = (5)_{10}$$

- Betrachten wir nun unser Standardbeispiel:

	x	y	Z(x,y,7)
Sydney	150	-36	92
Buenos Aires	-58	-36	25
Rio de Janeiro	-42	-26	28
Los Angeles	-118	34	38
Toronto	-80	45	56
Paris	2	50	104
Moskau	38	56	105
Kyoto	136	36	118
Nairobi	36	0	97
Istanbul	29	41	99

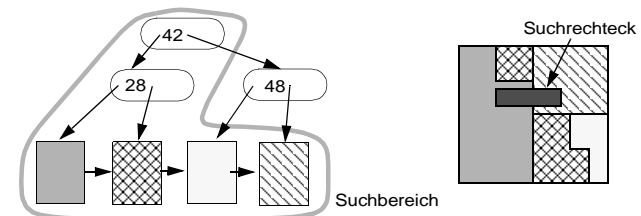


230.

Fensteranfragen

Naiver 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

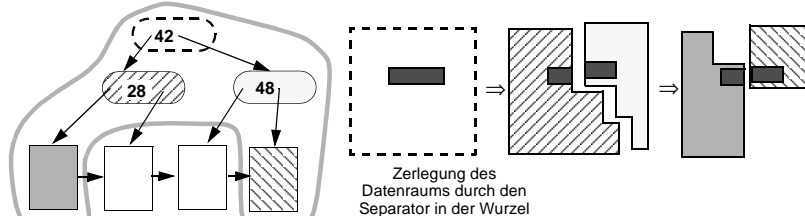


- ineffizient, da der Suchbereich verglichen mit dem Suchrechteck sehr groß ist.

232.

Verfeinerter 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



- ❑ Beurteilung
 - Mehraufwand für das Durchlaufen der Indexseiten im B⁺-Baum
 - + Zugriff nur auf die tatsächlich relevanten Daten- und Directoryseiten.

233.

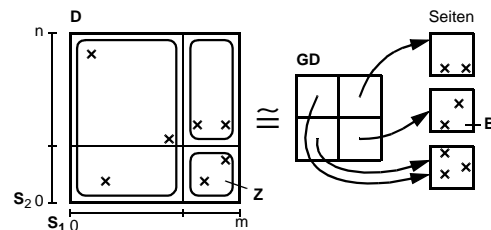
Eigenschaften des ZB⁺-Baums

- ❑ besitzt viele gute Eigenschaften des B⁺-Baums:
 - logarithmische Tiefe des Baums
 - niedrige Kosten für Einfügen u. Löschen (ist auf einen Pfad beschränkt)
 - hohe Speicherplatzausnutzung
- ❑ einfache Integration in bestehende Systeme (z. B. ORACLE, Spatial Data Option)
- ❑ Datenraumunterteilung: vollständig, inhomogen, disjunkt
- ❑ inhomogene Regionen: nicht rechteckig und nicht zusammenhängend
 - negativer Einfluß auf Bereichsanfragen
 - eine Region besitzt maximal eine "Sprungstelle"
- ❑ spezieller Algorithmus für Bereichsanfrage ist effizienter
 - top-down
 - Zerlegung des Anfrageraums in Regionen, die durch Z-Werte beschrieben werden können.
- ❑ Verwendung anderer raumfüllender Kurven
 - Hilbert-Kurve

234.

6.3.3 Grid File

- ❑ verallgemeinert das Prinzip von EH für mehrdimensionale Daten
- ❑ Idee:



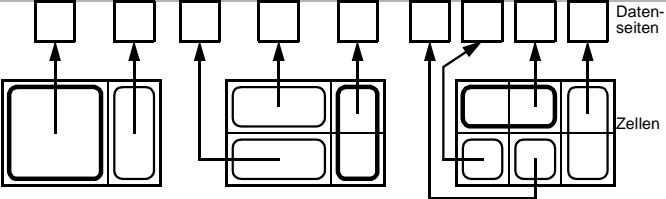
- ❑ Komponenten
 - k Skalierungsvektoren (Scales) definieren die Zellen (Grid) auf k-dim. Datenraum D
 - Zell- oder Grid-Directory GD: dynamische k-dim. Matrix zur Abbildung von D auf die Menge der Seiten
 - Seiten: Speicherung der Objekte einer oder mehrerer Zellen (Seitenbereich SB)

235.

- ❑ Eigenschaften
 - 1:1-Beziehung zwischen Zelle Z_i und Element von GD
 - Element von GD = Zelle zu Seite B
 - n:1-Beziehung zwischen Z_i und B
- ❑ Ziele
 - Erhaltung der Topologie
 - effiziente Unterstützung aller Fragetypen
 - vernünftige Speicherplatzbelegung
- ❑ Anforderungen
 - Prinzip der zwei Plattenzugriffe: unabhängig von Werteverteilungen, Operationshäufigkeiten und Anzahl der gespeicherten Sätze
 - Split- und Mischoperationen jeweils nur auf zwei Seiten
 - Speicherplatzbelegung:
 - durchschnittliche Belegung der Seiten nicht beliebig klein
 - schiefe Verteilungen vergrößern nur GD
- ❑ Entwurf einer Directory-Struktur
 - dynamische k-dim. Matrix GD (auf Externspeicher)
 - k eindim. Vektoren S_i (im Hauptspeicher)

236.

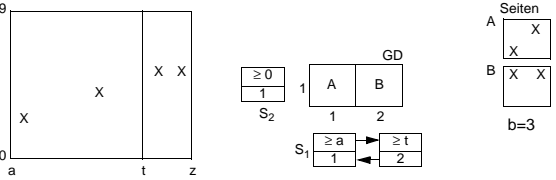
Zuweisung von Zellen zu Seiten



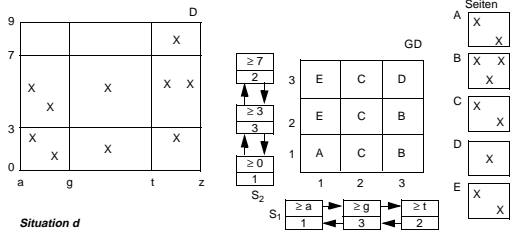
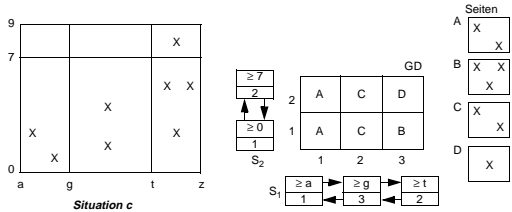
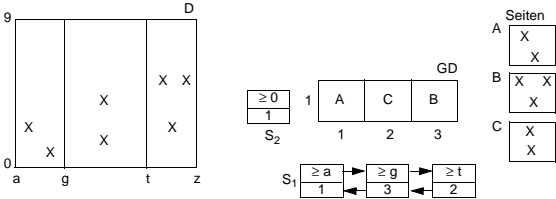
- Anforderung
 - Splitting überfüllter Seiten
 - Verschmelzen unterfüllter Seiten
- Seitenregionen (= Vereinigung aller Zellen einer Seite) sind rechteckig
 - diese Bedingung allein kann zu einer Partitionierung führen, die ein Verschmelzen von Seitenregionen nicht mehr unterstützt (Deadlock)
- Projektionen der Seitenregionen auf die Achsen sind binär konstruierbar
 - keine Deadlocks für $k = 2$

Grid File (Beispiel)

□ nachdem 4 Datensätze eingefügt sind:

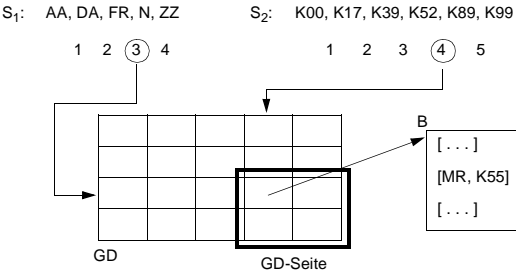


□ nachdem 7 Datensätze eingefügt sind:



Exakte Anfrage

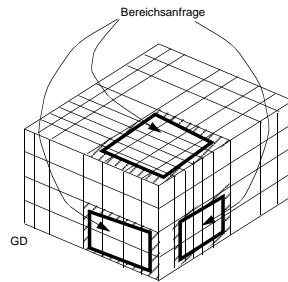
SELECT *
FROM PERS
WHERE ORT = 'MR' AND ANR = 'K55'



- gegeben zwei Indizeinträge der Skalen
 - Wie berechnet man die Adresse der zugehörigen Zelle?

Bereichsanfrage

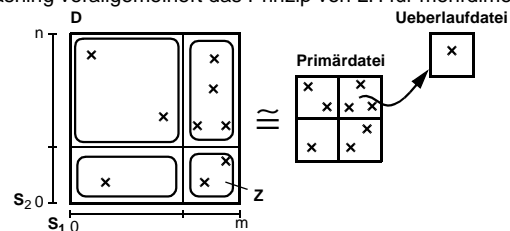
- ❑ Bestimmung der Skalierungswerte in jeder Dimension
- ❑ Berechnung der qualifizierten GD-Einträge
- ❑ Zugriff auf die GD-Zelle(n) und Holen der referenzierten Seiten



241.

PLOP-Hashing

- ❑ PLOP = Piecewise Linear Order Preserving (Kriegel & Seeger 1988)
- ❑ PLOP-Hashing verallgemeinert das Prinzip von LH für mehrdimensionale Daten

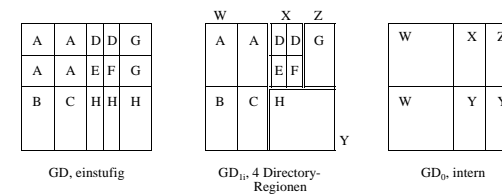


- ❑ Erweitern der Datei
 - Kontrollfunktion: Falls Belegungsfaktor in einer Scheibe größer 100 %
 - Lineares Aufspalten der Seiten in dieser Scheibe
- ❑ Verschmelzen von Seiten
 - Kontrollfkt.: Falls Belegung in zwei benachbarten Scheiben kleiner 40%

243.

Leistungsbetrachtung

- ❑ Wachstum von GD ist superlinear
 - bei gleichförmigen Objektverteilungen: $O(N^{1+(k-1)/k*b})$
 - bei schiefen Objektverteilungen bis zu : $O(N^k)$
- ❑ Verfeinerung in Dimension k
 - Anzahl der hinzukommenden GD-Einträge: $n = (m_1 * m_2 * \dots * m_{k-1})$
- ❑ Speicherplatzausnutzung für Seiten
- ❑ Organisation von GD als Grid-File
 - Regelfall: zweistufige Realisierung



242.

Leistungsbeurteilung

- ❑ im schlechtesten Fall:
 - extrem lange Ketten
 - schlechte SPAN
 - ...
- ❑ Forderung für effiziente Verarbeitung des PLOP-Hashing (und Grid File):
 - Verteilung der Attribute müssen unabhängig voneinander sein.
 Unter diesen Voraussetzungen ist PLOP-Hashing sehr effizient:
 - hohe SPAN (76%)
 - exakte Anfrage (1.01 Zugriffen im Durchschnitt)
 - effiziente Unterstützung von Bereichsanfragen
 - geringer Aufwand in einem Reorganisationsschritt

244.

6.4 Zugriffsstrukturen für ausgedehnte räumliche Objekte

- Ausgedehnte räumliche Objekte besitzen
 - allgemeine Merkmale wie Name, Beschaffenheit, ...
 - Ort und Geometrie (Kurve, Polygon, ...)
- Indexierung des räumlichen Objektes
 - genaue Darstellung?
 - Objektapproximation durch schachtelförmige Umhüllung - effektiv!
==> dadurch werden aber Fehltreffer möglich
- Probleme
 - neben Position muß auch die Ausdehnung bei der Abbildung der Objekte auf Seiten berücksichtigt werden.
 - Objekte können andere enthalten oder sich gegenseitig überlappen
- Klassifikation der Lösungsansätze
 - sich gegenseitig überlappende Regionen (R-Baum)
 - Clipping (R^+ -Baum)
 - Transformationsansatz

245.

- R-Baum ist höhenbalancierter Mehrwegbaum
 - jeder Knoten entspricht einer Seite
 - pro Knoten maximal M , minimal m ($\leq M/2$) Einträge

Blattknoteneintrag:



kleinstes umschreibendes Rechteck
(Datenrechteck) für TID

Zwischenknoteneintrag:



Intervalle beschreiben kleinste
umschreibende Datenregion für
alle in PID enthaltenen Objekte

I_j = geschlossenes Intervall bzgl. Dimension j

TID: Verweis auf Objekt

PID: Verweis auf Sohn

- Eigenschaften
 - starke Überlappung der umschreibenden Rechtecke/Regionen auf allen Baumebenen möglich
 - bei Suche nach Rechtecken/Regionen sind ggf. mehrere Teilbäume zu durchlaufen
 - Änderungsoperationen ähnlich wie bei B-Bäumen

247.

6.4.1 R-Baum

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

Definition

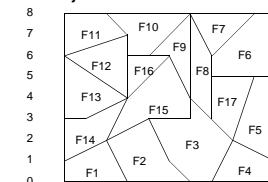
Ein *R-Baum* mit ganzzahligen Parameter m und M , $1 \leq m \leq \lceil M/2 \rceil$, organisiert eine Menge von Rechtecken in einem Baum mit folgenden Eigenschaften:

- Der Baum besteht aus Daten- und Directoryseiten. In einer *Datenseite* werden Rechtecke (plus TID) 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.

246.

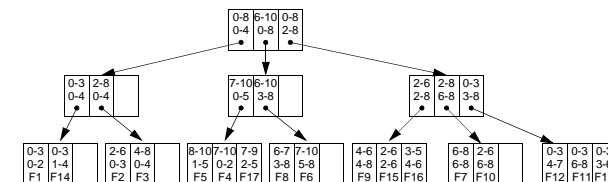
Beispiel (Flächen)

- Abzuspeichernde Flächenobjekte



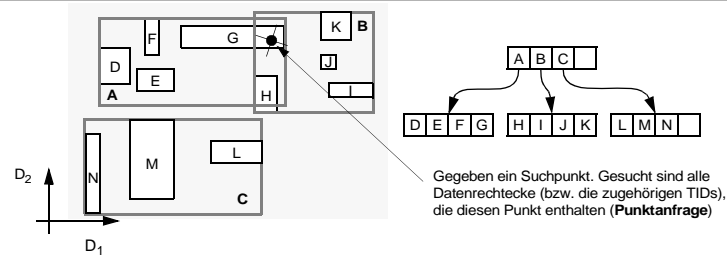
0 1 2 3 4 5 6 7 8 9 10

- Flächenobjektezugehöriger R-Baum



248.

Beispiel (Anfragen)

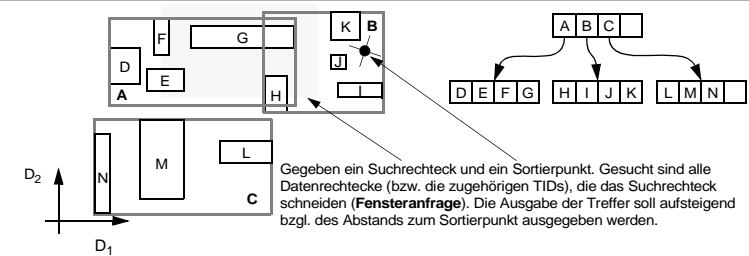


Aufbau einer Anfrage (Tiefendurchlauf)

- ❑ Füge die Referenz des Wurzelknotens in einen Stapel ein.
- ❑ Solange der Stapel nicht leer ist:
 - a) Nimm das oberste Element aus dem Stapel und lies den zugehörigen Knoten.
 - b) Falls der Knoten ein Directoryknoten ist:
Füge die Referenzen der **qualifizierenden** Einträge in den Stapel ein.
 - c) Andernfalls:
Gib die TIDs der qualifizierenden Datensätze aus.

249.

Prioritätsgesteuerte Bereichsanfragen

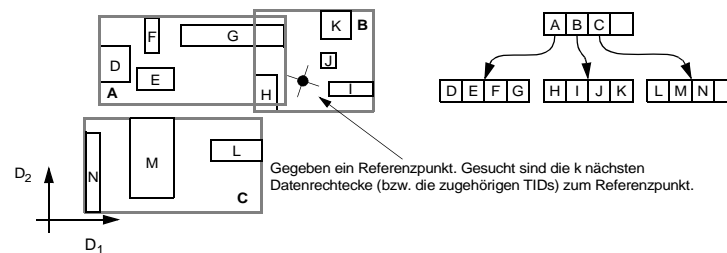


Aufbau der Anfrage

- ❑ Im Unterschied zu der zuvor beschriebenen Anfrage wird nun statt eines Stapels eine Prioritätswarteschlange benutzt.
- ❑ Priorität eines Rechtecks ist der minimale Euklidische Abstand zum Sortierpunkt.
- ❑ Datenrechtecke werden nun ebenfalls in die Prioritätswarteschlange eingefügt.

250.

k-nächste-Nachbaranfrage



Aufbau der Anfrage

- ❑ Die Anfrage entspricht einer prioritätsgesteuerten Bereichsanfrage, wobei der Suchbereich dem gesamten Datenraum entspricht.
- ❑ Abbruch des Algorithmus nachdem k Ausgaben produziert wurden.
- ❑ Beispiel ($k = 3$): H, J, I

251.

Kriterien für effiziente Anfragebearbeitung

- ❑ **Überdeckung** (coverage) einer Ebene eines R-Baums
 - gesamter Bereich, um alle zugehörigen Rechtecke zu überdecken
 - ❑ **Überlappung** (overlap) einer Ebene eines R-Baums
 - gesamter Bereich, der in zwei oder mehr Knoten enthalten ist
 - ❑ **Umfang** einer Ebene eines R-Baums
 - Summe des Umfangs aller Rechtecke in einem Knoten
- ⇒ effiziente Suche erfordert minimale Überdeckung, Überlappung und Umfang
- ❑ minimale Überdeckung reduziert die Menge des "toten Raumes" (leere Bereiche), der von den Knoten des R-Baums überdeckt wird.
 - ❑ minimale Überlappung reduziert die Menge der Suchpfade zu den Blättern (noch kritischer für die Zugriffszeit als minimale Überdeckung)
 - ❑ minimaler Umfang reduziert die Trefferquote bei einer Fensteranfrage
 - ⇒ wird beim Splitting der Seiten versucht zu erreichen.

252.

Einfügestrategie des R-Baums

- Wie beim B+-Baum soll das Einfügen auf genau einen Pfad beschränkt sein.
- Gegeben ein innerer Knoten. An welchen Zweig des Knotens soll die Einfügeoperation weitergeleitet werden?

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: ..., das die geringste Fläche hat)
 - D muß entsprechend vergrößert werden

Variationsmöglichkeiten

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

253.

Quadratischer Algorithmus (Verfeinertes Greedy-Verfahren)

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

255.

Splitstrategien für R-Bäume

- 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 \Rightarrow sehr aufwendig ($M=200$)
- Gibt es eine niedrigere obere Schranke?

Linearer Algorithmus (Greedy-Verfahren)

- 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.

254.

Split des R*-Baums (Plane-Sweep-Verfahren)

- 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 Dimension 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 haben sich bei Experimenten für $m = 0,4 \cdot M$ ergeben)

256.

Vergleich der verschiedenen Strategien

- ❑ Resultate aus Experimenten mit verschiedenen Datenverteilungen (10000 Datensätze) und verschiedenen Anfragetypen.
- ❑ Keine Pufferung (mit Ausnahme eines Pfads)
- ❑ Experimente mit zweidimensionalen Punkten

	linearer Split	quadratischer Split	R*-Baum Split
rel. Anfrageleistung	233.1	175.9	100.0
Speicherplatzausnutzung	64 %	68%	71%
Einfügekosten	7.3	4.5	3.4

- ❑ Experimente mit zweidimensionalen Rechteckmengen

	linearer Split	quadratischer Split	R*-Baum Split
rel. Kosten einer Fensteranfrage (0.1 %)	189.8	126.4	100.0
Speicherplatzausnutzung	63 %	68%	73%
Einfügekosten	12.6	7.7	6.1

Zusammenfassung

- ❑ Wichtige Eigenschaften mehrdimensionaler Zugriffspfade
 - Erhaltung der topologischen Struktur des Datenraumes
 - Adaption an die Objektdichte
 - Dynamische Reorganisation
 - Balancierte Zugriffsstruktur
 - Unterstützung von verschiedenen Anfragetypen: Fenster- u. Nachbaranfrage
- ❑ Darstellung von räumlichen Objekten
 - Abstraktion zu punktförmiger Repräsentation ist Regelfall
 - "Ausgedehnte" Darstellung nur in grober Annäherung
- ❑ Notwendigkeit der Integration von "konventionellen" und mehrdimensionalen Zugriffspfaden in Nichtstandard-DBS
 - Anfrageoptimierung
 - Mehrbenutzerbetrieb
 - Ansätze existieren für den ZB+-Baum und den R-Baum
- ❑ Erprobung der praktischen Tauglichkeit von mehrdimensionalen Zugriffsstrukturen (R-Baum) steht noch aus.