Implementierung von Datenbanksystemen

Kapitel 6: Mehrdimensionale Zugriffspfade

Kapitel 6: Mehrdimensionale Zugriffspfade

- Ziele
  - Entwurfsprinzipien für Zugriffspfade auf die Sätze einer Tabelle, bei denen mehrere Suchkriterium symmetrisch unterstützt werden
  - Indexierung von Punktobjekten und räumlich ausgedehnten Objekten
- Klassifikation der Anfragen
- Grundprobleme
  - Organisation räumlicher Daten
  - Erhaltung der Topologie (Clusterbildung)
  - Objektdarstellung
- Organisation der Datensätze
  - Quad-Tree
  - Multi-Key-Hashing
- Organisation des umgebenden Datenraums
  - k-d-Baum
  - Grid-File
- Zugriffspfade für ausgedehnte räumliche Objekte
  - R-Baum
  - R^+-Baum

Klassifikation der Anfragetypen

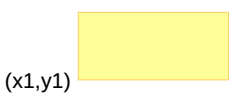
- Definitionen:
  - Eine Datei ist eine Sammlung von N Sätzen des Typs $R = (A_1, ..., A_n)$, wobei jeder Satz ein Punktobjekt durch ein geordnetes n-Tupel $t = (a_1, a_2, ..., a_n)$ von Werten darstellt. Die Attribute $A_1, ..., A_k$ ($k \leq n$) seien Schlüssel.
  - Eine Anfrage Q spezifiziert einige Bedingungen, die von den Schlüsselwerten der Sätze in der Treffermenge erfüllt sein müssen.
  - Schnittbildende Anfragen (intersection queries):
    sich qualifizierende Objekte überlappen mit dem Anfragebereich
  - Enthaltenseins- oder Umschließungsanfragen (containment, enclosure queries):
    sich qualifizierende Objekte sind ganz im Anfragebereich enthalten oder enthalten den Anfragebereich vollständig

Klassifikation der schnittbildenden Anfragen

- Exakte Anfrage (exact match query): spezifiziert für jeden Schlüssel einen Wert
  - $Q = (A_1 = a_1) \wedge (A_2 = a_2) \wedge \ ... \wedge (A_k = a_k)$
- Partielle Anfrage (partial match query): spezifiziert $s < k$ Schlüsselwerte
  - $Q = (A_{i1} = a_{i1}) \wedge (A_{i2} = a_{i2}) \wedge \ ... \wedge (A_{is} = a_{is})$ mit $1 < s < k$ und $1 < i_1 < i_2 < ... < i_s < k$
- Bereichsanfrage (range query):
  spezifiziert einen Bereich $r_i = [l_i < a_i < u_i]$ für jeden Schlüssel A_i
  - $Q = (A_1 = r_1) \wedge \ ... \wedge (A_k = r_k)$
    $\equiv (A_1 > l_1) \wedge \ (A1 < u_1) \wedge \ ... \wedge (A_k > l_k) \wedge \ (A_k < u_k)$
- Partielle Bereichsanfrage (partial range query):
  spezifiziert für $s < k$ Schlüssel einen Bereich
  - $Q = (A_{i1} = r_{i1}) \ \wedge \ ... \wedge (A_{is} = r_{is})$
    mit $1 < s < k$ und $1 < i_1 < ... < i_s < k$ und $r_{ij} = [l_{ij} < a_{ij} < u_{ij}]$, $1 < j < s$
- Allgemeine Bereichsanfrage:
  - genauer Bereich $[l_i = a_i = u_i]$
  - unendlicher Bereich $[-\infty < a_i < \infty]$
  - Alle 4 Fragetypen der schnittbildenden Anfragen lassen sich als allg. Bereichsfrage ausdrücken

Klassifikation der Enthaltenseinsanfragen

- Punktanfrage (point query): Gegeben ist ein Punkt im Datenraum D. Finde alle Objekte, die ihn enthalten.
- Gebietsanfrage (region query): Gegeben ist ein Anfragegebiet. Finde alle Objekte, die es schneiden (es umschließen, in ihm enthalten sind).
- Beispiele: Suche nach Rechtecken
  - Relation RECTANGLE(x1,y1,x2,y2)



  - Bestimmung aller Rechtecke, die den Punkt (2,5) enthalten

```
SELECT    x1, y1, x2, y2
FROM      RECTANGLES
WHERE     x1 <= 2 AND  x2 >= 2 AND  y1 <= 5  AND  y2 >= 5
```

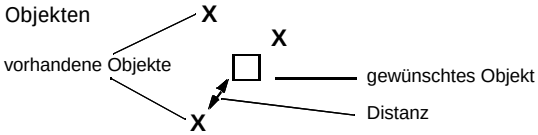
  - Bestimmung der Rechtecke mit Punkt (1,3) als Eckpunkt links unten

```
SELECT    x1, y1, x2, y2
FROM      RECTANGLES
WHERE     x1 = 1 AND y1 = 3
```

  - ➢ Das sind alles harmlose Fragen, die sogar im Relationenmodell mühelos beantwortet werden können

Nächster-Nachbar-Anfragen

- Nächster-Nachbar-Anfragen (best match query, nearest neighbor query)
  - gewünschtes Objekt nicht vorhanden
  - ➢ Frage nach möglichst ähnlichen Objekten



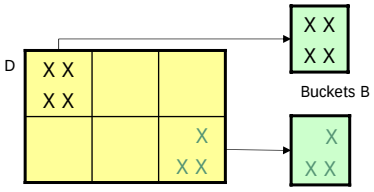
  - "best" wird bestimmt über verschiedene Arten von Distanzfunktionen
- Beispiele:
  - Objekt erfüllt nur 8 von 10 geforderten Eigenschaften
  - Objekt ist durch Synonyme beschrieben
- Bestimmung des nächsten Nachbars:
  - D = Distanzfunktion
    B = Sammlung von Punkten im k-dimensionalen Raum
  - Gesucht: nächster Nachbar von p (in B)
  - Der nächste Nachbar ist q, wenn
    $(\forall\ r \in B)\ \{r \neq q \ \Rightarrow\ [D(r, p) > D(q, p)]\}$

Kapitel 6: Mehrdimensionale Zugriffspfade

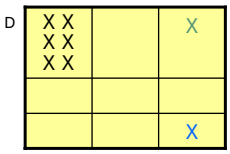
- Ziele
  - Entwurfsprinzipien für Zugriffspfade auf die Sätze einer Tabelle, bei denen mehrere Suchkriterium symmetrisch unterstützt werden
  - Indexierung von Punktobjekten und räumlich ausgedehnten Objekten
- Klassifikation der Anfragen
- Grundprobleme
  - Organisation räumlicher Daten
  - Erhaltung der Topologie (Clusterbildung)
  - Objektdarstellung
- Organisation der Datensätze
  - Quad-Tree
  - Multi-Key-Hashing
- Organisation des umgebenden Datenraums
  - k-d-Baum
  - Grid-File
- Zugriffspfade für ausgedehnte räumliche Objekte
  - R-Baum
  - R⁺-Baum

Grundprobleme

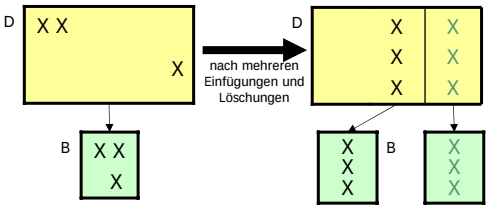
- Erhaltung der topologischen Struktur



- Stark variierende Dichte der Objekte
  - Starke Änderung der räumlichen Belegung über die Zeit
  - Keine regelmäßige Aufteilung von D
  - gleiche Bucketgröße



- Objektdarstellung
  - Punktobjekte
  - Objekte mit Ausdehnung
- Dynamische Reorganisation

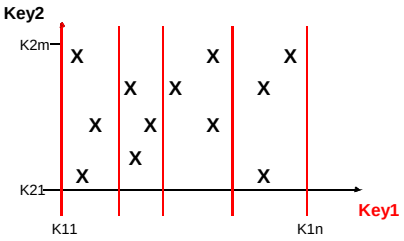


- Balancierte Zugriffsstruktur
  - beliebige Belegungen und Einfüge-/Löschreihenfolgen
  - Garantie eines gleichförmigen Zugriffs
  - 2 oder 3 Externspeicherzugriffe

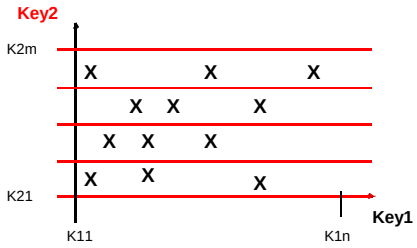
Nutzung eindimensionaler Zugriffspfade

- Bisher:
  - Indexierung (Invertierung) einer Dimension, z.B. B*-Baum
- Zerlegungsprinzip des Schlüsselraumes beim B*-Baum (2-dim.)
- Zusätzlicher B*-Baum (Key2) möglich:

- B*-Baum (Key1)
  - Partitionierung des Raumes nach Werten von Key1

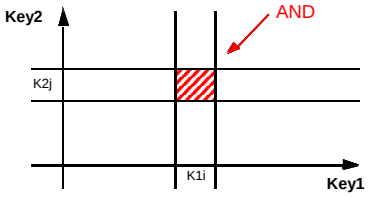


- B*-Baum (Key2)
  - Partitionierung des Raumes nach Werten von Key2



Nutzung eindimensionaler Zugriffspfade

- Mehrattributsuche
  - Zugriff nach (Key1 = K1i) { OR / AND } (Key2 = K2j)
  - Trefferliste für K1i : aus B*-Baum (Key1)
  - Trefferliste für K2j : aus B*-Baum (Key2)
  - Mischen + Zugriff auf Ergebnistupel
- große Trefferlisten und Zwischenergebnisse!



Nutzung eindimensionaler Zugriffspfade

- Simulation des mehrdimensionalen Zugriffs mit einem B*-Baum?
- Idee:
  - Konkatenierte Schlüssel:
  - Konkatenierte Werte:

Key1	Key2
K11	K21
K11	K22
	⋮
K11	K2m
K12	K21
K12	K22
	⋮
K12	K2m
K13	K21
	⋮
K1n	K2m

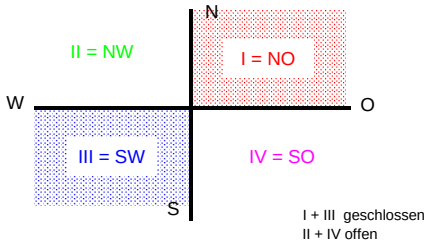
- Unterstützung für Suchoperationen?
  - (Key1 = K1i) AND (Key2 = K2j)
  - Key2 = K2j
  - Key1 = K1i
  - OR-Verknüpfung

Kapitel 6: Mehrdimensionale Zugriffspfade

- Ziele
  - Entwurfsprinzipien für Zugriffspfade auf die Sätze einer Tabelle, bei denen mehrere Suchkriterium symmetrisch unterstützt werden
  - Indexierung von Punktobjekten und räumlich ausgedehnten Objekten
- Klassifikation der Anfragen
- Grundprobleme
  - Organisation räumlicher Daten
  - Erhaltung der Topologie (Clusterbildung)
  - Objektdarstellung
- Organisation der Datensätze
  - Quad-Tree
  - Multi-Key-Hashing
- Organisation des umgebenden Datenraums
  - k-d-Baum
  - Grid-File
- Zugriffspfade für ausgedehnte räumliche Objekte
  - R-Baum
  - R^+-Baum

Quad-Tree (Quadranten-Baum)

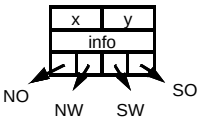
- Speicherungsstruktur:
  - für 2-dimensionalen Mehrattributzugriff
  - Zerlegungsprinzip des Datenraumes D: rekursive Partitionierung durch Quadranten
- Realisierung als Generalisierung des Binärbaumes
  - jeder Knoten enthält einen Satz
  - Außengrad eines Knotens: max. 4
  - Wurzel teilt 2-dimensionalen Raum in 4 Quadranten auf
  - rekursive Aufteilung jedes Quadranten durch Wurzel eines Unterbaumes
  - i-ter Unterbaum eines Knotens enthält Punkte im i-ten Quadranten
- Ziel: Berücksichtigung von Nachbarschaftsbeziehungen
  - Beispiel: geographische Daten mit Koordinaten x und y
- Aufteilung:

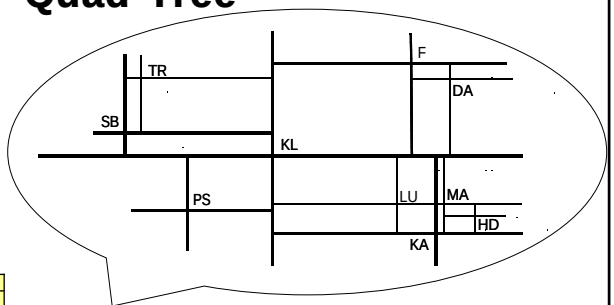


- Verallgemeinerung:
  - k-dimensionaler Schlüssel ⇒ Außengrad jedes Knotens: 2^k
  - k=3: Octtree, k=4: Hextree

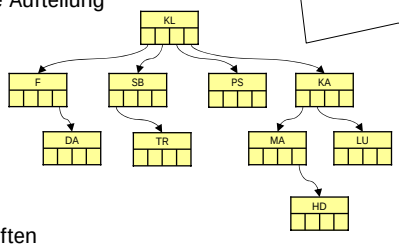
Quad-Tree

- Knotenformat





- Räumliche Aufteilung



- Eigenschaften
  - Baumstruktur abhängig von Einfügereihenfolge (unbalanciert)
  - aufwendiges Löschen (Neueinfügen der Unterbäume)
  - keine Abbildung auf Seiten

Multi-Key-Hashing (partitioned hashing)

- Zerlegungsprinzip von D:
  - Partitionierung durch Hashfunktionen in jeder Dimension
  - wird realisiert durch Aufteilung der Bucketadresse in k Teile (k = Anzahl der Schlüssel)
- Für jeden Schlüssel i:
  - eigene Hash-Funktion h_i, die Teil der Bucketadresse bestimmt
  - Anzahl der Buckets sei 2^B (Adresse = Folge von B Bits)
  - jede Hash-Funktion h_i liefert b_i Bits

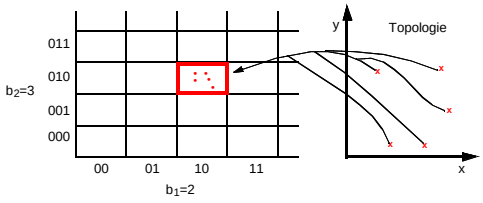
    mit $B = \sum_{i=1}^{k} b_i$

    Satz $t = (a_1, a_2, ..., a_k, ...)$ gespeichert in Bucket mit Adr. = $h_1(a_1) \mid h_2(a_2) \mid ... \mid h_k(a_k)$

- Vorteile:
  - kein Index
  - geringer Speicherplatzbedarf und Änderungsaufwand
  - Exact-Match-Anfragen: Gesamtadresse bekannt, Zugriff auf 1 Bucket
  - Partial-Match-Anfrage: Eingrenzung des Suchraumes ($A_i = a_i$): Anzahl zu durchsuchender Buckets reduziert sich um $2b_i$ $N_B = 2^B / 2^{bi} = 2^{B-bi}$
- Nachteile / Probleme:
  - topologische Struktur der Daten bleibt nicht erhalten
  - keine Unterstützung von Bereichs- und Best-Match-Anfragen
  - Optimale Zuordnung der b_i zu A_i abhängig von Fragehäufigkeiten

Multi-Key-Hashing: Beispiel

- Anwendungsbeispiel:
  Pnr: INT (5) b1 = 4
  Svnr: INT (9) b2 = 3
  Aname: CHAR (10) b3 = 2
  B=9 (512 Buckets)



Bucket mit Adresse '10010' enthält alle Sätze mit
$h_1(a_1)$ = '10' und $h_2(a_2)$ = '010'

h_1(Pnr)	= Pnr mod 16	h_1(58651)	= 11	→ 1011
h_2(Svnr)	= Svnr mod 8	h_2(130326734)	= 6	→ 110
h_3(Aname)	= L(Aname) mod 4	h_3(XYZ55)	= 1	→ 01

B-Adr. = 101111001

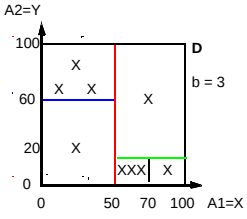
- Anzahl der Zugriffe
  - Exact-Match-Anfragen: Zugriff auf 1 Bucket
  - Partial-Match-Anfragen:
    - Pnr = 58651 $N_B = 2^9/2^4 = 32$
    - (Pnr = 73443) AND (Svnr = 2332) $N_B = 2^9/ 2^{4+3} = 4$

Kapitel 6: Mehrdimensionale Zugriffspfade

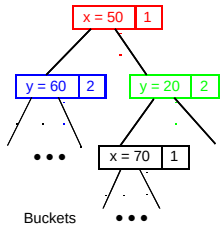
- Ziele
  - Entwurfsprinzipien für Zugriffspfade auf die Sätze einer Tabelle, bei denen mehrere Suchkriterium symmetrisch unterstützt werden
  - Indexierung von Punktobjekten und räumlich ausgedehnten Objekten
- Klassifikation der Anfragen
- Grundprobleme
  - Organisation räumlicher Daten
  - Erhaltung der Topologie (Clusterbildung)
  - Objektdarstellung
- Organisation der Datensätze
  - Quad-Tree
  - Multi-Key-Hashing
- Organisation des umgebenden Datenraums
  - k-d-Baum
  - Grid-File
- Zugriffspfade für ausgedehnte räumliche Objekte
  - R-Baum
  - R⁺-Baum

Organisation des umgebenden Datenraums – Divide and Conquer

- Zerlegungsprinzip von D
  - D wird dynamisch in Zellen aufgeteilt
  - Die Objekte einer Zelle werden als Sätze in Buckets abgelegt
  - bei Bucket-Überlauf: lokale Zellverfeinerung (Divide and Conquer)
  - abschnittsweise Erhaltung der Topologie (Clusterbildung)
  - Baum als Zugriffsstruktur für die Buckets hat nur Wegweiserfunktion
- Beispiel:
  - Heterogener k-d-Baum mit k = 2:



- Eigenschaften des k-d-Baumes
  - Clusterbildung durch Buckets ist Voraussetzung für praktischen Einsatz
  - Eingebauter Balancierungs-mechanismus ist nicht vorhanden
  - Wie werden Aktualisierungsoperationen (Löschen!) durchgeführt?
  - Wie werden die verschiedenen Anfragetypen unterstützt?

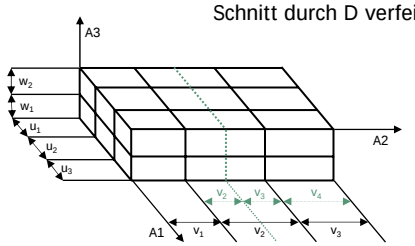


Organisation des umgebenden Datenraums – Dimensionsverfeinerung

- Prinzip:
  - Datenraum D wird dynamisch durch ein orthogonales Raster (grid) partitioniert, so dass k-dimensionale Zellen (Grid-Blöcke) entstehen
  - Die in den Zellen enthaltenen Objekte werden in Buckets gespeichert
- Beispiel:

Datenraum D = A1 x A2 x A3
Zellpartition P = U x V x W
Abschnitte der Partition
$U = (u_1, u_2, ..., u_l)$
$V = (v_1, v_2, ..., v_m)$
$W = (w_1, w_2, ..., w_n)$

  - Eine Zelle ist deshalb eindeutig einem Bucket zuzuordnen
  - Die klassenbildende Eigenschaft dieser Verfahren ist das Prinzip der Dimensionsverfeinerung, bei dem ein Abschnitt in der ausgewählten Dimension durch einen vollständigen Schnitt durch D verfeinert wird



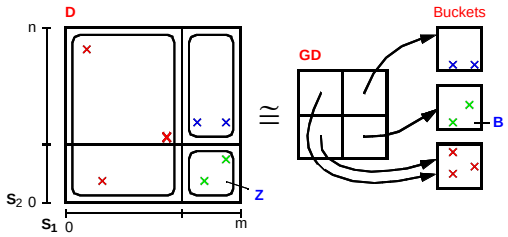
Dreidimensionaler Datenraum D mit Zellpartition P; Veranschaulichung eines Split-Vorganges im Intervall v_2

Organisation des umgebenden Datenraums – Dimensionsverfeinerung

- Probleme der Dimensionsverfeinerung
  - Wie viele neue Zellen entstehen jedes Mal?
  - Was folgt für die Bucketzuordnung?
  - Welche Abbildungsverfahren können gewählt werden?
  - Gibt es Einschränkungen bei der Festlegung der Dimensionsverfeinerung?

Grid-File: Idee

- Zerlegungsprinzip von D: Dimensionsverfeinerung



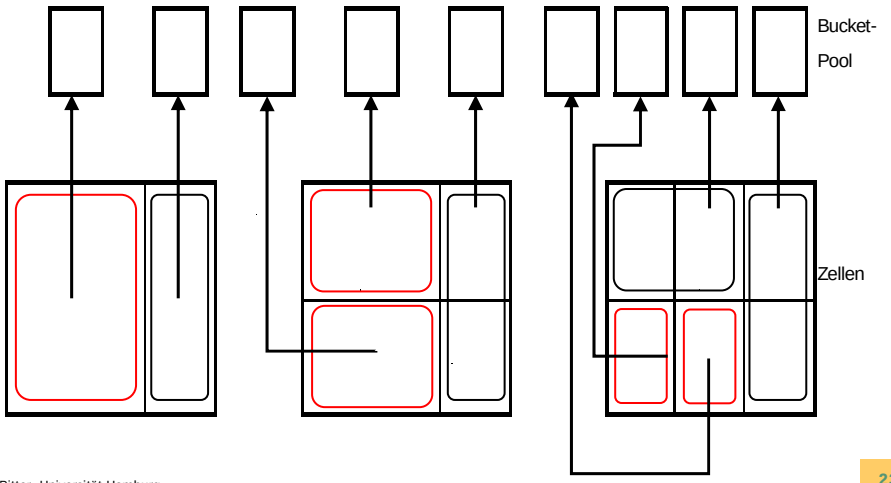
- Komponenten:
  - k Skalierungsvektoren (Scales) definieren die Zellen (Grid) auf k-dim. Datenraum D
  - Zell- oder Grid-Directory GD: dynamische k-dim. Matrix zur Abbildung von D auf die Menge der Buckets
  - Buckets: Speicherung der Objekte einer oder mehrerer Zellen pro Bucket
- Eigenschaften:
  - 1:1-Beziehung zwischen Zelle Z_i und Element von GD
  - Element von GD = Zeiger auf Bucket B
  - n:1-Beziehung zwischen Z_i und B
- Ziele:
  - Erhaltung der Topologie
  - effiziente Unterstützung aller Anfragetypen
  - vernünftige Speicherplatzbelegung

Quelle: [NH+84]

Zentrale Datenstruktur: Grid-Directory

- Anforderungen:
  - Prinzip der zwei Plattenzugriffe unabhängig von Werteverteilungen, Operationshäufigkeiten und Anzahl der gespeicherten Sätze
  - Split- und Mischoperationen jeweils nur auf zwei Buckets
  - Speicherplatzbelegung
    - durchschnittliche Belegung der Buckets 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)
- Operationen auf GD:
  - direkter Zugriff auf einen GD-Eintrag
  - relativer Zugriff (NEXTABOVE, NEXTBELOW)
  - Mischen zweier benachbarter Einträge einer Dimension (mit Umbenennung der betroffenen Einträge)
  - Splitting eines Eintrages einer Dimension (mit Umbenennung)

Schachtelförmige Zuweisung von Zellen zu Buckets

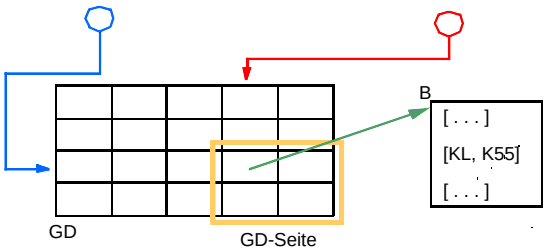


Suchfragen: Exakte Anfrage

- Exakte Anfrage (exact match)

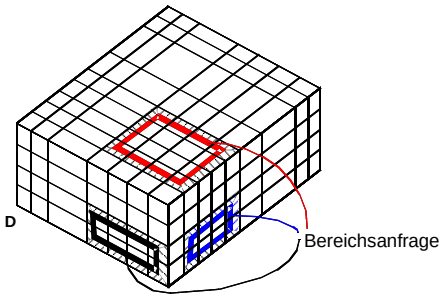
```
SELECT  *
FROM        PERS
WHERE       ORT = 'KL' AND ANR = 'K55'
```

S1: AA, DA, FR, MK, ZZ S2: K00, K17, K39, K52, K89, K99



Suchfrage: Bereichsanfrage

- Bereichsanfrage
  - Bestimmung der Skalierungswerte in jeder Dimension
  - Berechnung der qualifizierten GD-Einträge
  - Zugriff auf die GD-Seite(n) und Holen der referenzierten Buckets



Kapitel 6: Mehrdimensionale Zugriffspfade

- Ziele
  - Entwurfsprinzipien für Zugriffspfade auf die Sätze einer Tabelle, bei denen mehrere Suchkriterium symmetrisch unterstützt werden
  - Indexierung von Punktobjekten und räumlich ausgedehnten Objekten
- Klassifikation der Anfragen
- Grundprobleme
  - Organisation räumlicher Daten
  - Erhaltung der Topologie (Clusterbildung)
  - Objektdarstellung
- Organisation der Datensätze
  - Quad-Tree
  - Multi-Key-Hashing
- Organisation des umgebenden Datenraums
  - k-d-Baum
  - Grid-File
- Zugriffspfade für ausgedehnte räumliche Objekte
  - R-Baum
  - R^+-Baum

Zugriffspfade für ausgedehnte räumliche Objekte

- Ausgedehnte räumliche Objekte besitzen
  - allgemeine Merkmale wie Name, Beschaffenheit, . . .
  - Ort und Geometrie (Kurve, Polygon, ...)
- Indexierung des räumlichen Objekts
  - genaue Darstellung?
  - Objektapproximation durch schachtelförmige Umhüllung – effektiv!
  - ➢ dadurch werden Fehltreffer möglich
- Probleme
  - neben Objektdichte muss Objektausdehnung bei der Abbildung und Verfeinerung berücksichtigt werden
  - Objekte können andere enthalten oder sich gegenseitig überlappen
- Klassifikation der Lösungsansätze
  - sich gegenseitig überlappende Regionen (R-Baum)
  - Clipping (R^+-Baum)
  - Transformationsansatz
  - ➢ bildet ausgedehnte räumliche Objekte funktional auf höherdimensionale Punkte ab
  - ➢ begrenzte Anwendbarkeit und Tauglichkeit!

R-Baum

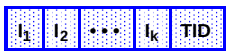
- Ziel:
  - Effiziente Verwaltung räumlicher Objekte (Punkte, Polygone, Quader, ...)
- Anwendungen:
  - Kartographie:
    Speicherung von Landkarten, effiziente Beantwortung “geometrischer” Fragen
  - CAD:
    Handhabung von Flächen, Volumina und Körpern (z.B. Rechtecke beim VLSI-Entwurf)
  - Computer-Vision und Robotics
- Hauptoperationen:
  - Punktanfragen (point queries): Finde alle Objekte, die einen gegebenen Punkt enthalten
  - Gebietsanfragen (region queries): Finde alle Objekte, die mit einem gegebenen Suchfenster überlappen (es umschließen, in ... vollständig enthalten sind)
- Ansatz: Speicherung und Suche von achsenparallelen Rechtecken
  - Objekte werden durch Datenrechtecke repräsentiert und müssen durch kartesische Koordinaten beschrieben werden
  - Repräsentation im R-Baum erfolgt durch minimale begrenzende (k-dimensionale) Rechtecke/Regionen
  - Suchanfragen beziehen sich ebenfalls auf Rechtecke/Regionen

Quelle: [Gut84]

R-Baum

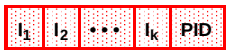
- R-Baum ist höhenbalancierter Mehrwegbaum
  - jeder Knoten entspricht einer Seite
  - pro Knoten maximal M, minimal m (>= M/2) Einträge

Blattknoteneintrag:



kleinstes umschreibendes Rechteck (Datenrechteck) für TID

Zwischenknoteneintrag:



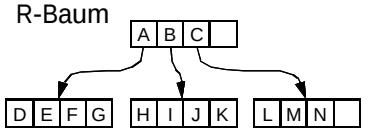
Intervalle beschreiben kleinste umschreibende Datenregion für alle in PID enthaltenen Objekte

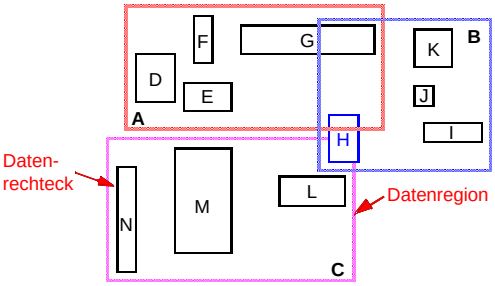
I_j : geschlossenes Intervall bzgl. Dimension j
TID : Verweis auf Objekt
PID : Verweis auf Sohn

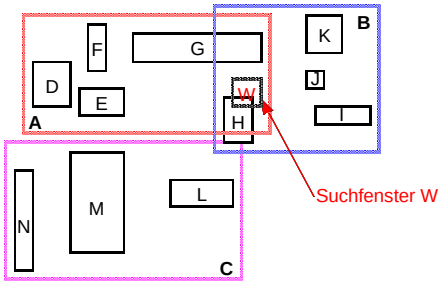
- Eigenschaften
  - starke Überlappung der umschreibenden Rechtecke/Regionen auf allen Baumebenen möglich
  - bei Suche nach Rechtecken/Regionen sind ggf. mehrere Teilbäume zu durchlaufen
  - Änderungsoperationen ähnlich wie bei B-Bäumen

Abbildung beim R-Baum

- Aufteilung des Datenraumes
- R-Baum



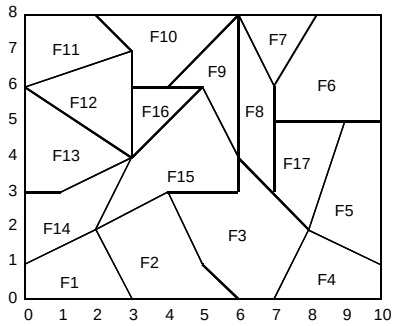




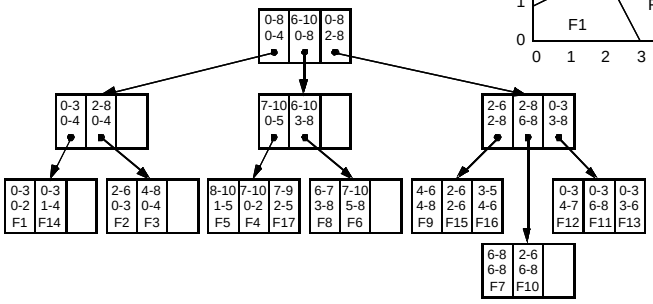
- Beispiel für ein „schlechtes“ Suchfenster

R-Baum: Beispiel

- **Abzuspeichernde Flächenobjekte**

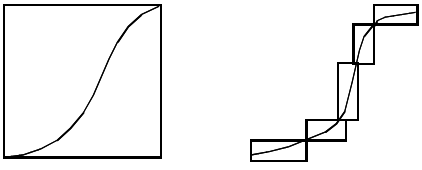


- **Zugehöriger R-Baum**



Suchoptimierung durch den R⁺-Baum

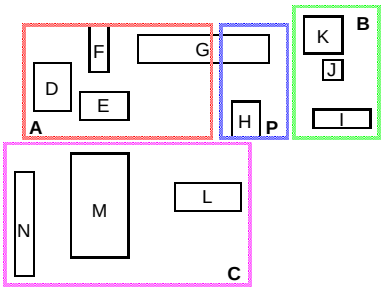
- Überdeckung und Überlappung bei einer Ebene des R-Baumes
  - Überdeckung (coverage) ist der gesamte Bereich, um alle zugehörigen Rechtecke zu überdecken
  - Überlappung (overlap) ist der gesamter Bereich, der in zwei oder mehr Knoten enthalten ist
  - Minimale Überdeckung reduziert die Menge des „toten Raumes“ (leere Bereiche), der von den Knoten des R-Baumes überdeckt wird.
  - Minimale Überlappung reduziert die Menge der Suchpfade zu den Blättern (noch kritischer für die Zugriffszeit als minimale Überdeckung)
  - ➢ Effiziente Suche erfordert minimale Überdeckung und Überlappung
- Idee
  - Es sind Partitionierungen erlaubt, die Datenrechtecke „zerschneiden” (Clipping)
  - Vermeidung von Überlappungen bei Zwischenknoten
- Konsequenz
  - Daten-Rechteck wird ggf. in eine Sammlung von disjunkten Teilrechtecken zerlegt und auf der Blattebene in verschiedenen Knoten gespeichert.



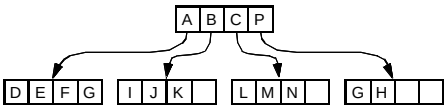
Aufteilungsmöglichkeiten eines langen Linienobjektes im R⁺-Baum

R⁺-Baum

- Aufteilung des Datenraumes
  - Höhere Flexibilität durch Partitionierung von Daten-Rechtecken



- R⁺-Baum



- Eigenschaften
  - Überlappung von Datenregionen wird vermieden
  - Überdeckungsproblematik wird wesentlich entschärft
  - Clusterbildung der in einer Region zusammengefassten Objekte kann durch Clipping verhindert werden
  - komplexere Algorithmen für bestimmte Anfragen (Enthaltensein)
  - schwierigere Wartung, keine Leistungsvorteile gegenüber R-Baum

Quelle: [Sel87]

Aspekte der Erweiterung der DB-Sprache

- Kein Raumbezug und keine räumlichen Operatoren im Relationenmodell
  - Hohe Komplexität bei einfachen Beispielen
  - Beispiel: Darstellung beliebig gelagerter Rechtecke in der Ebene
    R-ECK (RE-NR, X1, Y1, X2, Y2, X3, Y3, X4, Y4)
  - Anfrage: Finde alle Rechtecke, die ein Rechteck mit den Punkten (PA, PB, PC, PD) echt umschließen.

```
SELECT RE-NR FROM R-ECK
WHERE     XA > X1 AND YA < G(P1, P2, XA)
            AND YA > G(P4, P1, XA) AND
          XB < Y2 AND YB < G(P1, P2, XB)
            AND XB < G(P2, P3, XB) AND
          XC < X3 AND YC < G(P2, P3, XC)
            AND YC > G(P3, P4, XC) AND
          YD > Y4 AND YD > G(P3, P4, XD)
            AND YD > G(P1, P4, XD);
```

  mit Abkürzungen: z.B. G(P1, P2, XA) für
  ((Y1 - Y2) * XA) / (X1 - X2) + (Y2 * X1 - Y1 * X2) / (X1 - X2)
- Objekt-relationale Datenmodelle – Schlüsseleigenschaft:
  - Erweiterbarkeit mit benutzerdefinierten Funktionen und Operatoren
  - Anfragen mit Operatoren wie Überlappung, Schnitt, Entfernung, Enthaltensein («) ...

```
SELECT RE-NR
FROM R-ECK X
WHERE [PA, PB, PC, PD] « X;
```

Zusammenfassung

- Wichtige Eigenschaften mehrdimensionaler Zugriffspfade
  - Erhaltung der topologischen Struktur des Datenraumes
  - Adaption an die Objektdichte
  - Dynamische Reorganisation
  - Balancierte Zugriffsstruktur
  - Unterstützung von verschiedenen Anfragetypen (intersection queries, best match queries)
- Darstellung von räumlichen Objekten
  - Abstraktion zu punktförmiger Repräsentation ist Regelfall
  - „Ausgedehnte" Darstellung nur in grober Annäherung: Verarbeitungsprobleme
- Vorteile neuerer Konzepte
  - Organisation des umgebenden Datenraumes – Grid File
  - Darstellung räumlich ausgedehnter Objekte – R- und R+-Baum
    Jedoch: Es existieren sehr viele Vorschläge und Konzepte, die sehr speziell und praktisch nicht erprobt sind
- Notwendigkeit der DBS-Integration von „konventionellen" (eindimensionalen) und mehrdimensionalen Zugriffspfaden
  - Zugriffsmöglichkeiten über räumliche und zeitliche Dimensionen
  - Unterstützung von „Data Warehouse"-Anwendungen, Geographischen Informationssystemen, . . .
- Erweiterung objekt-relationaler DBS durch mehrdimensionale Zugriffspfadstrukturen (vor allem Grid File und R-Baum)

Literatur zu diesem Kapitel

[Gut84] A. Guttman: R-Trees: A Dynamic Index Structure for Spatial Searching. In: Proc. ACM SIGMOD Conf., 1984.

[NH+84] Jürg Nievergelt, Hans Hinterberger, Kenneth C. Sevcik: The Grid File: An Adaptable, Symmetric Multikey File Structure. In: ACM Trans. Database Syst. 9(1) 1984.

[Sel87] Sellis, T. et al.: The R+-Tree: A Dynamic Index for Multi-Dimensional Objects. In: Proc.14th Int. Conf. on Very Large Data Bases, Brighton, 1987.