# 7. Mehrdimensionale Zugriffsstrukturen

- Anwendungen, Anforderungen u. Grundprobleme
  - Geo-Datenbanken
  - Erhaltung der Topologie (Clusterbildung)
- 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





# Motivation

## 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.





# Effiziente Unterstützung von Anfragen

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

## Anfragen:

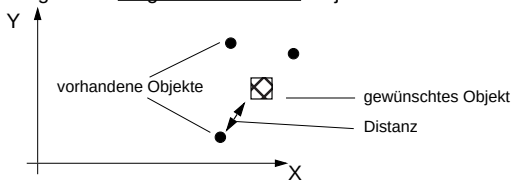
- exact match query:
  Gegeben $(x_1,\ldots,x_k) \in D$. Suche den Datensatz $r \in R$ mit $r.A_1 = x_1, \ldots, r.A_k = x_k$.
- partial match query:
  Gegeben $m \le k$ und Index $(i_1,\ldots,i_m)$ mit $1 \le i_1 \le \ldots \le i_m \le k$. Gegeben $(x_{i1},\ldots,x_{im})$ mit $x_{ij} \in dom(A_{ij})$. Suche alle Datensätze $r \in R$ mit $r.A_{i1} = x_{i1}, \ldots, r.A_{im} = x_{im}$.
- window query (Bereichsanfrage):
  Gegeben $(u_1,\ldots,u_k), (o_1,\ldots,o_k) \in D$ mit $u_i \le o_i$. Suche alle Datensätze $r \in R$ mit $u_1 \le r.A_1 \le x_1, \ldots, u_k \le r.A_k \le x_k$.





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

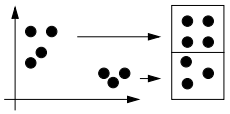
- gewünschtes Objekt nicht vorhanden. Stattdessen:
  Frage nach möglichst ähnlichen Objekten




- "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
  **$(\forall\ r \in B)\ \{r \neq q \ \Rightarrow\ [D(r,p) \geq D(q,p)]\}$**

# 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.

  ==> im Datenraum nah beieinander liegende DS sollen gemeinsam in einer Seite liegen.

  ==> hohe Speicherplatzausnutzung





# 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.
- Klassifizierung der Verfahren kann bzgl. der Eigenschaften der Regionen vorgenommen werden:

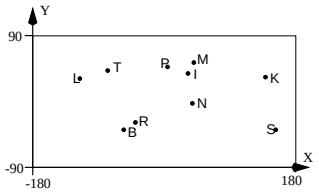
| | homogen | vollständig | disjunkt | Verfahren |
|---|---|---|---|---|
| C1 | X | X | X | PLOP-Hashing, Grid-File |
| C2 | X | X | | Twin Grid-File |
| C3 | X | | | R-Baum |
| C4 | X | | X | R+-Baum, Buddy-Baum |
| C5 | | X | X | BANG-File, ZB+-Baum |





# Beispiels-Datenbank

| | x-Koord. | y-Koord. |
|---|---|---|
| **S**ydney | 150 | - 36 |
| **B**uenos Aires | - 58 | - 36 |
| **R**io de Jainero | - 42 | - 26 |
| **L**os Angeles | -118 | 34 |
| **T**oronto | - 80 | 45 |
| **P**aris | 2 | 50 |
| **M**oskau | 38 | 56 |
| **K**yoto | 136 | 36 |
| **N**airobi | 36 | 0 |
| **I**stanbul | 29 | 41 |

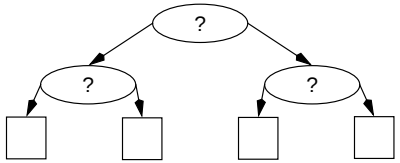







# 7.1. Mehrdimensionale Binäre Bäume

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




interne Knoten

Datenseiten

- binäre Bäume sind primär für den Hauptspeicher entwickelt worden und sind i.a. nicht effizient für den Externspeicher verwendbar.

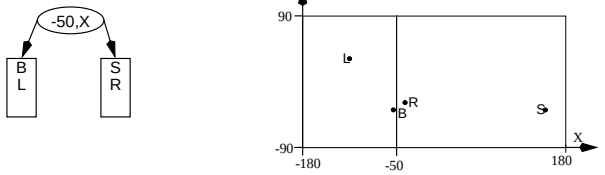
- im Folgenden werden die drei wichtigsten Varianten vorgestellt:
  - adaptiver kd-Baum (Friedman, Bentley & Finkel, 1977)
  - kd-trie (Orenstein, 1982)
  - BD-Baum (Ohsawa & Sauki, 1983)

# 7.1.2 Adaptiver kd-Baum

Idee:

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

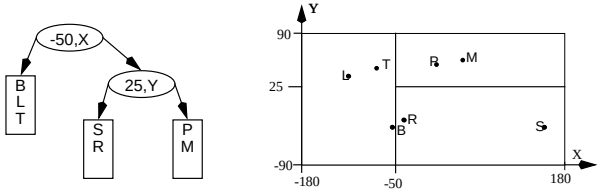
- nach dem Einfügen der ersten vier Datensätze:



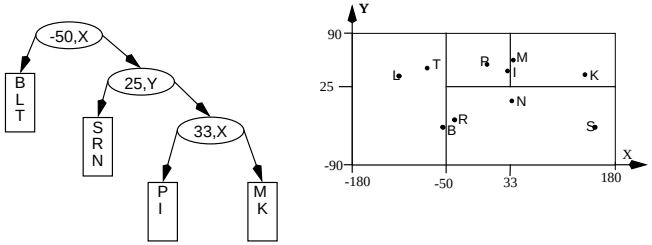





- Einfügen von Toronto, Paris und Moskau




- nachdem alle Datensätze eingefügt sind

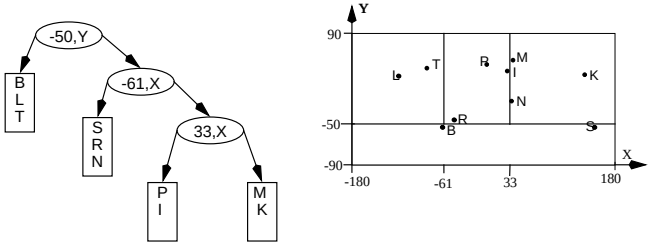







# Eigenschaften

- 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 er Splitachse bestimmt die Unterteilung des Datenraums



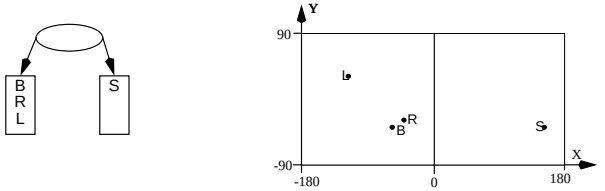




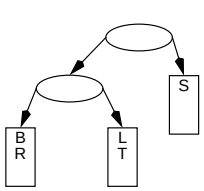

# 7.1.2 kd-trie

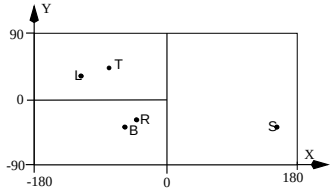
Idee:

- 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.

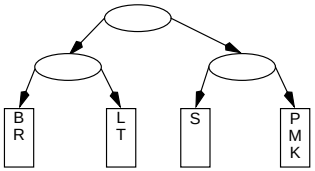


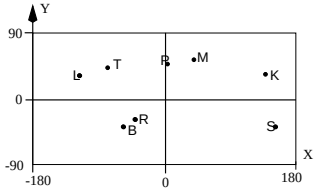
- Einfügen von Toronto führt wieder zu einem Ueberlauf




- Einfügen von Paris, Moskau und Kyoto

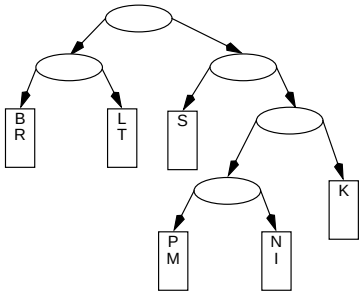


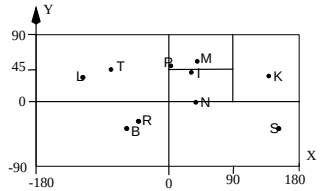








- nachdem alle Datensätze eingefügt sind:




## Eigenschaften:

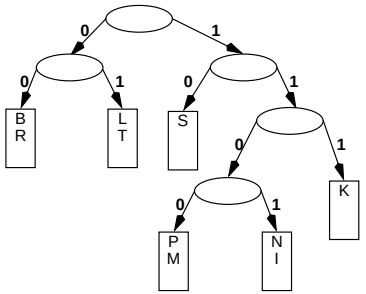
- kd-trie kann auch unbalanciert sein. Jedoch ist die Höhe des Baums durch die Auflösung des Datenraums beschränkt.
- Seiten können im schlechtesten Fall nur einen Datensatz enthalten.
- Unterteilung des Datenraums wird **nicht** durch die Eingabereihenfolge bestimmt!

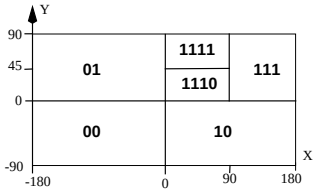




# Z-String

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




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

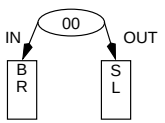


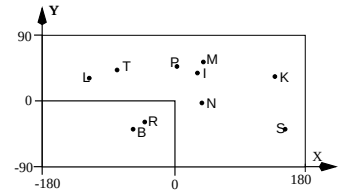


# 7.1.3 BD-Baum

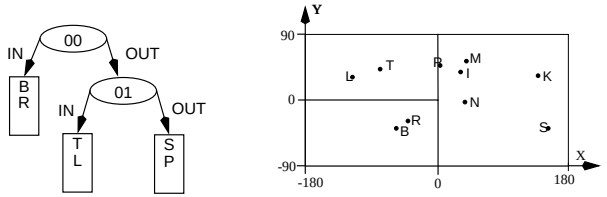
## Idee:

- 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:

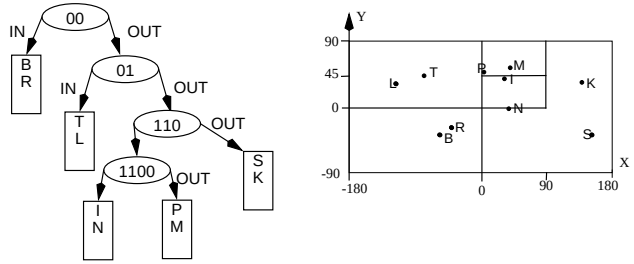


nach zwei weiteren Datensätzen:




BD-Baum nachdem alle Datensätze eingefügt sind:

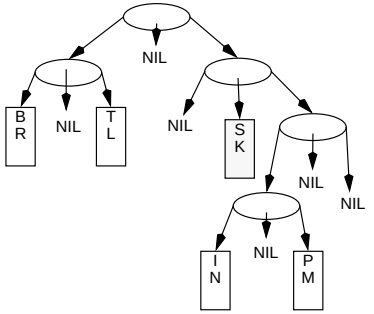


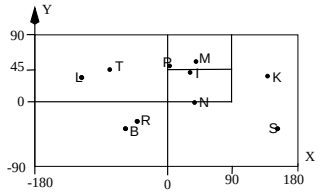





# Alternative Darstellung von BD-Bäume

- ein BD-Baum kann als kd-trie 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.








# 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?





# 7.2 Mehrdimensionale Externspeicher-Zugriffsstrukturen

## Ziel beim Entwurf

- entwerfe ein Verfahren mit den gleichen Eigenschaften eindimensionaler Verfahren (B+-Baum)

## Verfahren

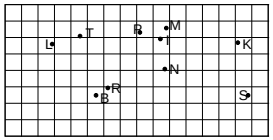
- basierend auf den Prinzipien des $B^+$-Baums
  - KDB-Baum
  - ZB+-Baum
  - BANG-File & Varianten
  - R-Baum
- basierend auf den Prinzipien von Hash-Verfahren
  - Grid-File & PLOP-Hashing

# 7.2.2 ZB+-Baum

Idee:

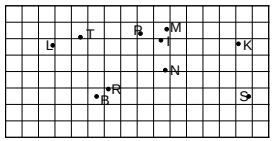
- Bilde einen k-dimensionalen Punkt in einen eindimensionalen Raum ab, so daß die mehrdimensionale Ordnung möglichst bewahrt bleibt.
- Speichere die eindimensionalen Punkte in einem gewöhnlichen B+-Baum

Lexikographische Ordnung




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

Z-Ordnung




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



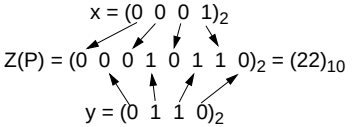


# Berechnung der Z-Werte

- vereinfachende Annahme: k = 2 (Dimensionen), D = {0,…,$2^n$-1} X {0,…,$2^n$-1}
- Sei (x,y) ∈ 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 des Punkts (x,y) (zum Level 2n) ergibt sich dann als

$$Z(x,y) = Z((x,y),2n) = \sum_{i=0}^{n-1} a_i \cdot 2^{2i+1} + \sum_{i=0}^{n-1} b_i \cdot 2^{2i}$$





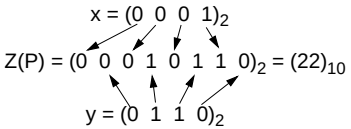



- i. a. wird nicht der komplette Z-Wert zur Identifizierung eines Punkts benötigt, sondern stattdessen nur ein Präfix:

  Der Z-Wert zum Level L, 0 ≤ L < 2n, ist definiert als

$$Z((x,y),L) = Z(x,y) \text{ div } 2^{2n-L}$$

- Beispiel: n = 4, P = (1, 6)




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

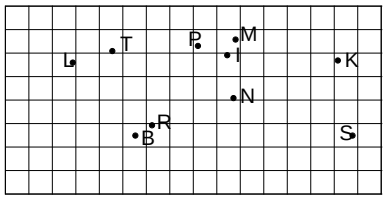
- Bemerkung
  - Ein Z-Wert zum Level L beschreibt eine Zelle des Datenraums, in der höchstens $2^{2n-L}$ verschiedene Punkte liegen können.
  - Ein Z-Wert zum Level L genügt für die Abspeicherung eines Punkts, falls es keinen weiteren Punkt in der zugehörigen Zelle gibt.



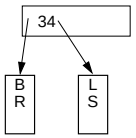


# Beispiel

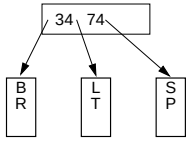
| | x | y | Z((x,y),7) |
|---|---|---|---|
| **S**ydney | 150 | - 36 | 92 |
| **B**uenos Aires | - 58 | - 36 | 25 |
| **R**io de Janero | - 42 | - 26 | 28 |
| **L**os Angeles | -118 | 34 | 38 |
| **T**oronto | - 80 | 45 | 56 |
| **P**aris | 2 | 50 | 104 |
| **M**oskau | 38 | 56 | 105 |
| **K**yoto | 136 | 36 | 118 |
| **N**airobi | 36 | 0 | 97 |
| **I**stanbul | 29 | 41 | 99 |



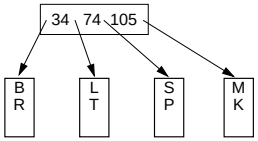

Einfügen von S, B, R, L




Einfügen von T, P




Einfügen von M, K



# 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)
- Datenraumunterteilung: vollständig, inhomogen, disjunkt
- inhomogene Regionen: nicht rechteckig und nicht zusammenhängend
  - negativer Einfluß auf Bereichsanfragen
- spezieller Algorithmus für Bereichsanfrage ist notwendig
  - top-down
  - Zerlegung des Anfrageraums in Regionen, die durch Z-Werte beschrieben werden können.
- Verwendung anderer raumfüllender Kurven
  - Hilbert-Kurve