



## – Objekte mit bekannten Kategorien

- Objekte sind in Kategorien eingeteilt und entsprechend markiert (z.B. „Schrauben“, „Nägel“, „Bolzen“, ...), d.h. Ergebnis der Anfragen ist bekannt

- Übersicht

	erwünscht	unerwünscht
gefunden	richtig positive (rp)	falsch positive (fp)
nicht gefunden	falsch negative (fn)	richtig negative (rn)

- Recall (Sensitivität): Wie viele der erwünschten Objekte wurden gefunden?

$$\frac{rp}{rp + fn} = \frac{\text{gefundene erwünschte Objekte}}{\text{alle erwünschten Objekte}}$$

- Precision: Wie viele der gefundenen Objekte sind erwünscht?

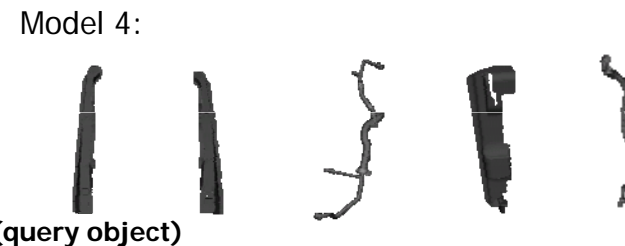
$$\frac{rp}{rp + fp} = \frac{\text{gefundene erwünschte Objekte}}{\text{alle gefundenen Objekte}}$$

- Spezifität: WS, dass Test für unerwünschtes Obj. negativ verläuft

$$\frac{rn}{rn + fp} = \frac{\text{richtig negativ}}{\text{alle unerwünschten Objekte}}$$

## – Objekte mit unbekanntem Kategorien

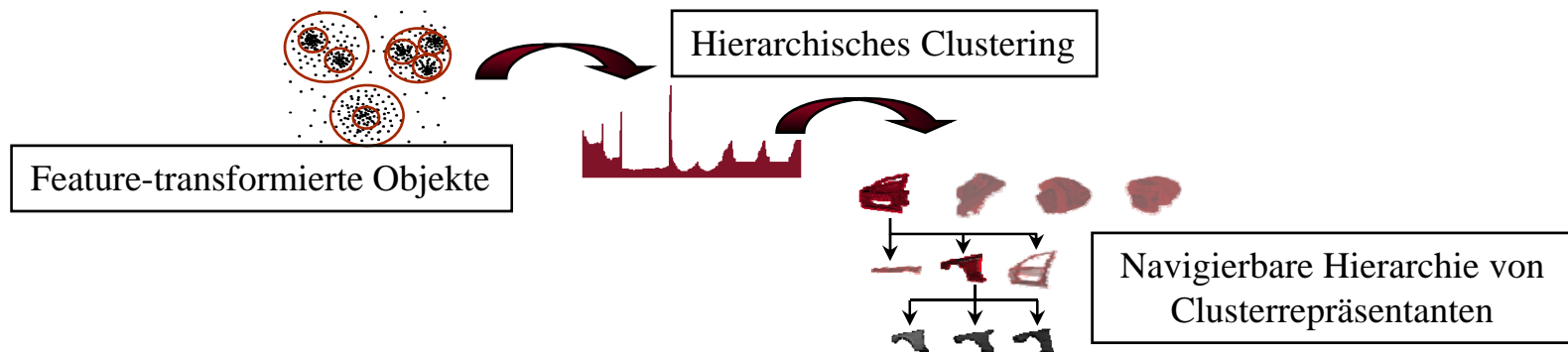
- Ergebnis der Anfragen ist unbekannt
- Manuelle Evaluation weniger zufälligen  $k$ -NN-Queries
- Problem: Qualität des Modells hängt ab von
  - einer geringen Anzahl von Query-Objekten
    - » Besser: möglichst alle Objekte der DB spielen eine Rolle bei der Evaluation
  - der Wahl dieser Query-Objekte
    - » Schlechtes Anfrageergebnis für gegebenes  $q$  bedingt nicht schlechtes Modell
    - » Gutes Anfrageergebnis für gegebenes  $q$  bedingt nicht gutes Modell



- BOSS (Browsing OPTICS-plots for Similarity Search)

[Brecheisen, Kriegel, Kröger, Pfeifle. Proc. SIAM Int. Conf. Data Mining (SDM), 2004]

- Idee: benutze Data Mining Methoden
- Clustering
  - » Fasse Objekte in Gruppen zusammen, sodass die Objekte in einer Gruppe (Cluster) ähnlich, Objekte aus verschiedenen Clustern unähnlich sind
  - » Hierarchisches Clustering: erstelle eine Hierarchie von ähnlichen Objekten
- Clustererkennung und Clusterrepräsentation
  - » Erkenne automatisch geeignete Cluster in der Hierarchie
  - » Stelle jeden Cluster durch einen geeigneten Repräsentanten dar
- Evaluation/Retrieval
  - » Hierarchie von Clusterrepräsentanten ist navigierbar
  - » Evaluation der Cluster um Ähnlichkeitsmodell zu evaluieren
  - » Ähnlichkeits-basierte Suche nach Objekten ohne konkretes Anfrageobjekt angeben zu müssen



---

# Kapitel 3

## Anfragen auf räumlichen Objekten

---

Skript zur Vorlesung: Spatial, Temporal, and Multimedia Databases  
Sommersemester 2013, LMU München

© 2007 Prof. Dr. Hans-Peter Kriegel, Dr. Peer Kröger, Dr. Peter Kunath, Dr. Matthias Renz, Arthur Zimek

# Übersicht

## 3.1 Schnitthanfragen räumlicher Objekte

- Raumpartitionierende Anfrage-Methoden
- Datenpartitionierende Anfrage-Methoden

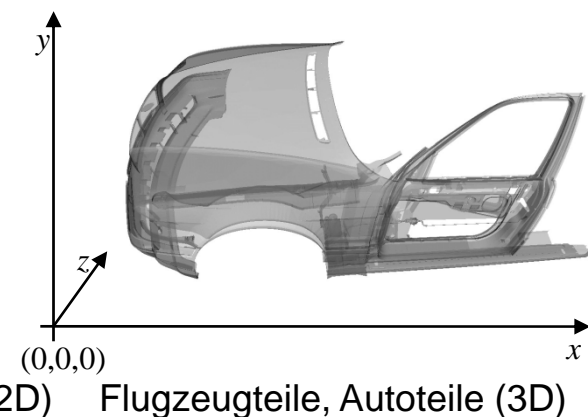
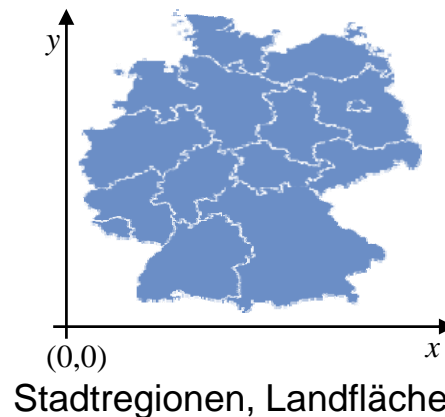
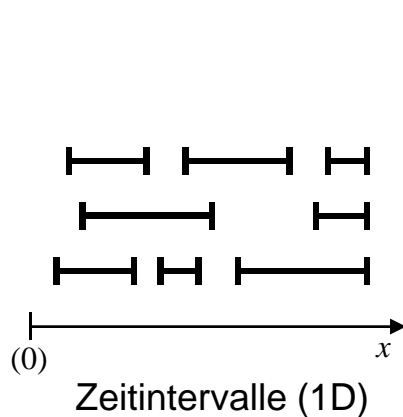
## 3.2 Ähnlichkeitssuche in räumlichen Objekten

- Invarianten
- Räumliche Partitionierungen
- Räumliche Features

## 3.1 Schnitthanfragen räumlicher Objekte

### – Allgemeines

- Unter räumlichen Objekten verstehen wir Objekte mit räumlichem Bezug (Lage im Raum) und räumlicher Ausdehnung.
- Räumliche Objekte können neben den räumlichen Attributen auch nicht-räumliche Attribute enthalten, wie z.B. Objekt-ID, Bezeichnung, Farbe, Material, etc.
- Beispiele von räumlichen Objekten:

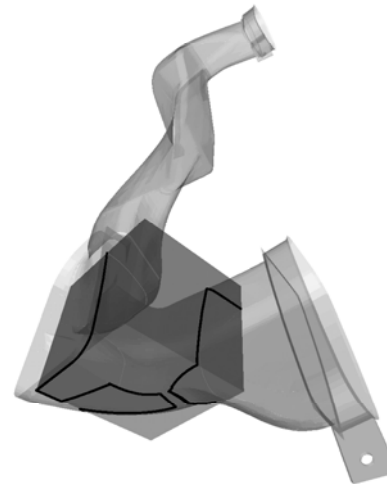


- Anfragen auf räumlich ausgedehnten Objekten (GEO-Objekte, CAD-Objekte)



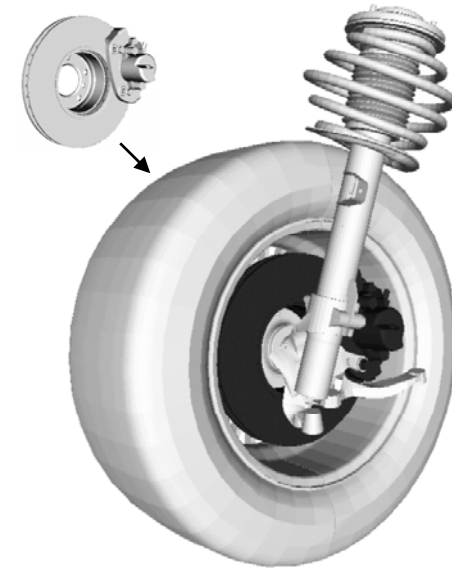
**Bereichsanfragen in  
GEO Datenbanken:**

Welche Strassen durchlaufen das  
ausgewählte Fenster?



**Räumliche Selektion:**

Welche Bauteile befinden sich in der angegebenen Anfragebox?

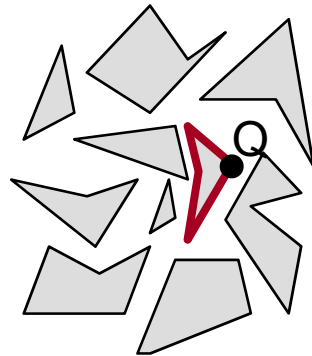
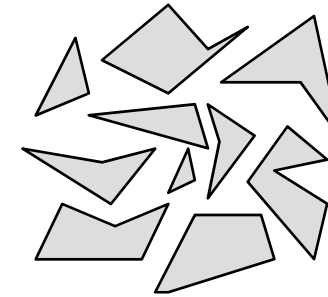


**Schnittanfragen:**

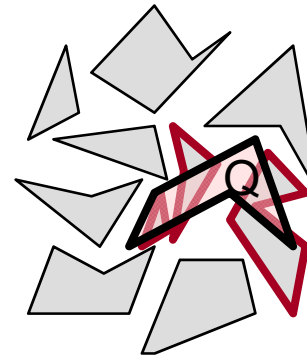
Welche Maschinenteile  
haben direkten Kontakt mit  
der vorderen rechten  
Bremscheibe?



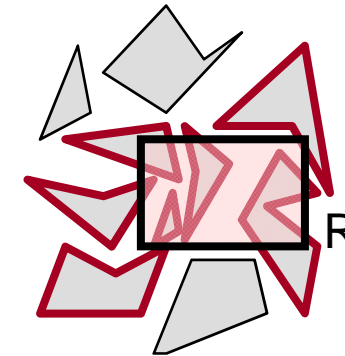
- Gegeben:  
Menge mit räumlich ausgedehnten Objekten
- Zu unterstützende Anfragen:



Punkt-Anfrage



Schnitt-Anfrage



Fenster-Anfrage

Gegeben ein Anfragepunkt  $P$ , ein Anfrageobjekt  $Q$ , bzw. ein Anfragerechteck  $R$  (2D)

- Punkt-Anfrage: Finde die Objekte, die  $P$  enthalten.
- Schnitt-Anfrage: Finde die Objekte, die  $Q$  schneiden.
- Fenster-Anfrage: Finde die Objekte, die  $R$  schneiden.

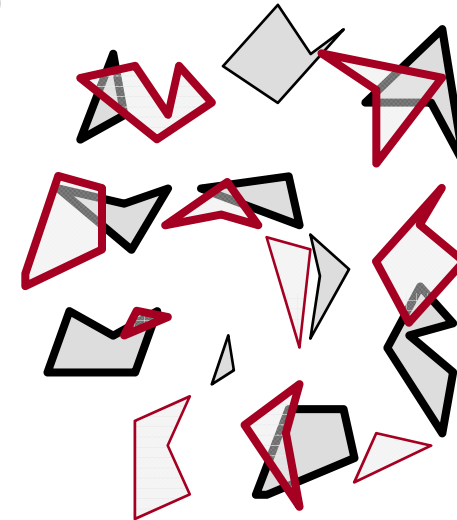
- Zu unterstützende Anfragen (Forts.)
  - Räumliche Verbund-Anfrage (Spatial Join)

Gegeben:

Zwei Mengen  $M = \{Obj_{M,1}, \dots, Obj_{M,m}\}$ ,  
 $N = \{Obj_{N,1}, Obj_{N,n}\}$  räumlich  
ausgedehnter Objekte

Spatial Join:

Finde die Menge von Objekt-Paaren  
 $\{(Obj_1, Obj_2) \mid Obj_1 \in M, Obj_2 \in N \text{ und } Obj_1 \text{ schneidet } Obj_2\}$ .



Auch andere räumliche Prädikate möglich, z.B.

- » andere Topologien wie *enthalten-in*
- » Distanz-basierte Anfragen
- » etc.

## – Ziele:

- **Effektivität**

- Verwaltung von (komplex strukturierten) geometrischen Objekten
- räumliche Anfragebearbeitung

- **Effizienz**

- kurze Antwortzeiten für Anfragen
- schnelles Einfügen, Ändern und Entfernen

- **Skalierbarkeit**

- Verwaltung sehr großer Datenmengen
- Anbindung vieler Benutzer

→ Reduzierung der Beschreibungskomplexität (d.h. einfachere Darstellung der Objekte)

→ Verwendung geeigneter Zugriffsstrukturen

→ Mehrstufige Anfragebearbeitung

## – Repräsentation Räumlicher Objekte

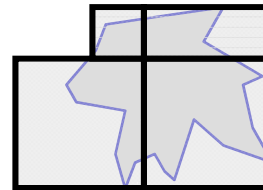
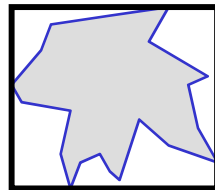
- Typischerweise sind räumlich ausgedehnte Objekte als geschlossener Kantenzug (2D) oder durch geschlossene triangulierte Oberflächen beschrieben.



triangulierte Oberfläche



- Für effiziente Anfragen werden in einem Filterschritt zunächst räumliche Approximationen der Objekte verwendet

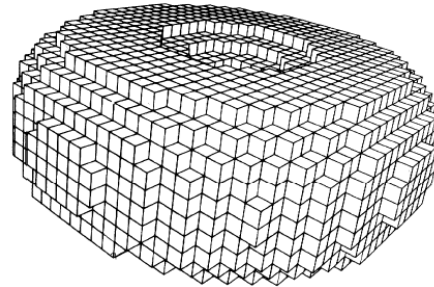


- **Raumpartitionierende Verfahren**
  - Repräsentationen durch feste Raumpartitionsgrenzen definiert
- **Datenpartitionierende Verfahren**
  - Repräsentationen durch die räumliche Ausdehnung der Daten definiert

### 3.1.1 Raumpartitionierende Anfrage-Methoden

- Ausgangslage: Voxel-basierte Objektbeschreibung (Raster-Modell)

- Bei der Voxel-basierten Zerlegung wird ein Körper in eine Menge identischer Zellen (i.d.R. Würfel) zerlegt, die auf einem festen regelmäßigen Gitter angeordnet sind.
- Beispiel (3D):

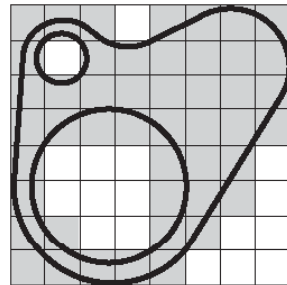


- Beschreibung der Objekte (des Objektraums) durch drei-dimensionales ARRAY [...] OF BOOLEAN (oder seltener eine Liste belegter Zellen)
- Anwendungen:
  - » Computer Tomographie, Virtual Engineering (CAD), etc.

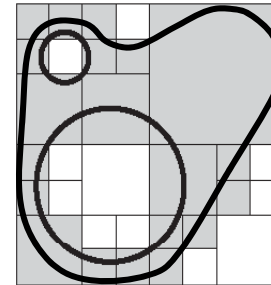
**Nachteil:**

genaue Approximation mit hohem Speicherplatzaufwand verbunden  
(Auflösung:  $n$  Voxel pro Dimension  $\rightarrow$  Speicherplatzaufwand =  $n^3$ )

- Octree-basierte Objektrepräsentation:
  - Hierarchische Variante der Voxel-basierten Zerlegung
  - Idee:
    - » hierarchische binären Unterteilung in Quadranten/Oktanten;
    - » Darstellung durch größte Quadranten die vollständig gefüllt sind.



Voxel-basierte Zerlegung

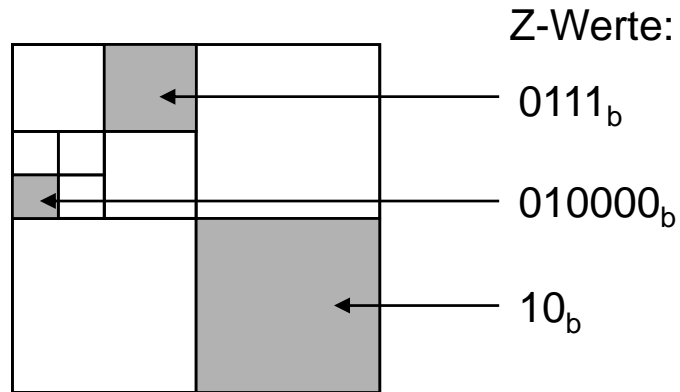


Octree (Quadtree) Zerlegung

- Vorteil:
  - » große einheitliche Flächen/Volumen Anteile werden über wenig große Partitionen (Quadranten/Oktanten) repräsentiert.
  - » Randbereiche, die eine beliebig komplexe Struktur (Form) annehmen können, werden über viele kleine Partitionen repräsentiert, die eine exaktere Darstellung erlauben.

- Indexierung

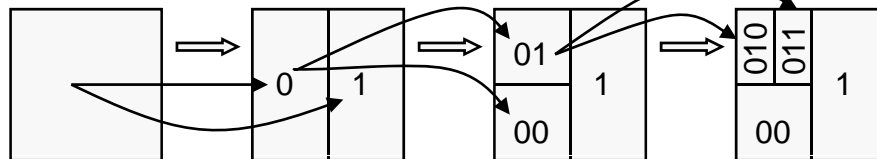
- Quadtree-Zellen durch Z-Werte benennen:



Z-Werte bestehen aus

- » Bitfolge  $\langle b_1, b_2, \dots, b_n \rangle$  durch abwechselnd rechts/links und oben/unten partitionieren (links/unten  $\rightarrow 0$ , rechts/oben  $\rightarrow 1$ )
- » Level = Anzahl der Bits

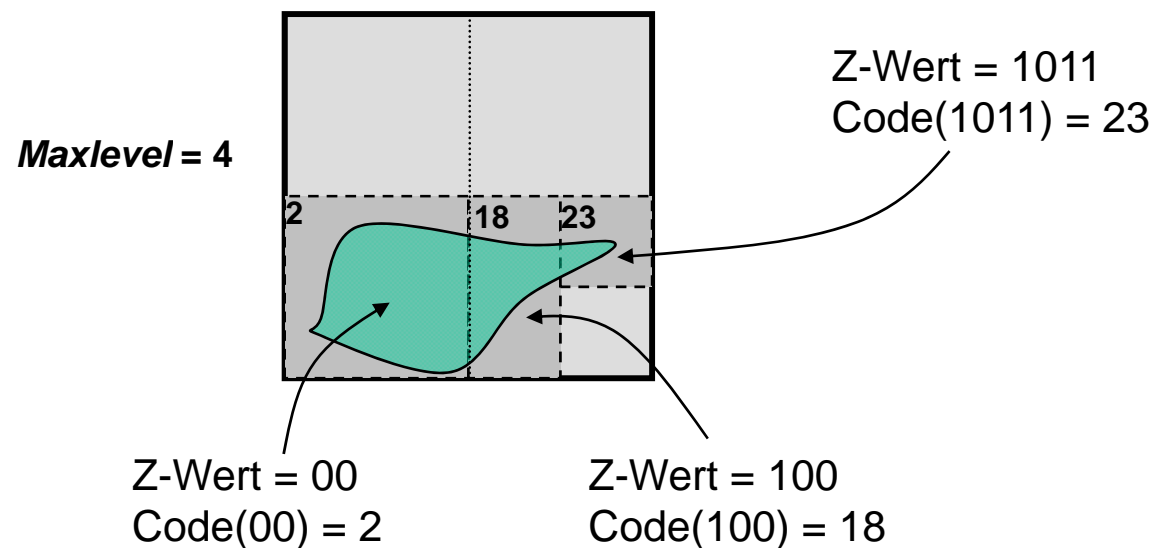
rekursive Schritte für die generierung der Z-Werte:



- Indexierung (Forts.)
  - Kodierung der Z-Werte => lineare Ordnung auf den Z-Werten

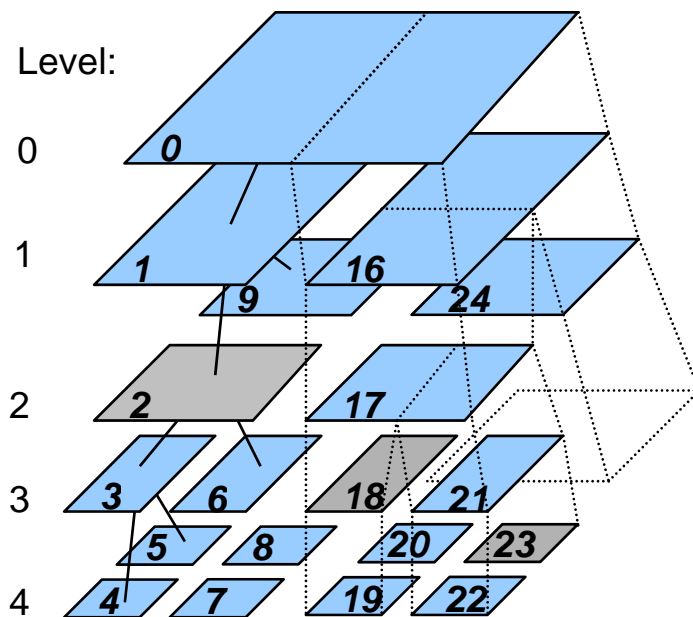
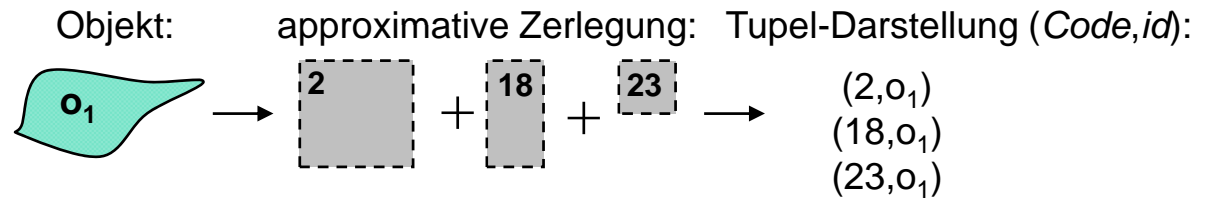
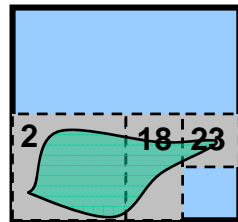
$$\text{Code}(\langle b_0, b_1, \dots, b_n \rangle) = \sum_{i=0..n} \begin{matrix} 1 & \text{falls } b_i = 0 \\ 2^{(\text{MaxLevel}-i)} & \text{falls } b_i = 1 \end{matrix}$$

*MaxLevel* = maximaler Level für die rekursive Partitionierung  
(s.o. Maxlevel = 6)

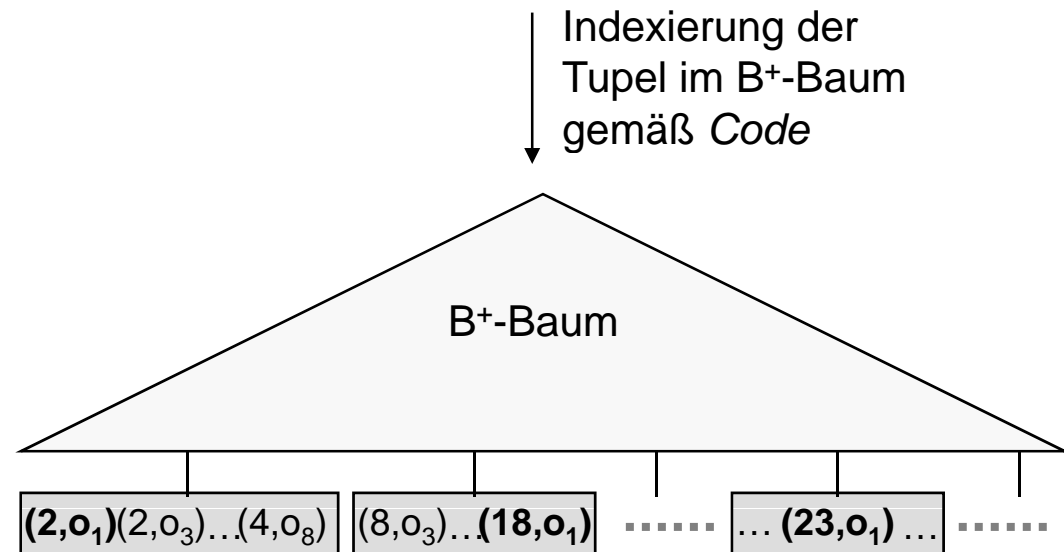




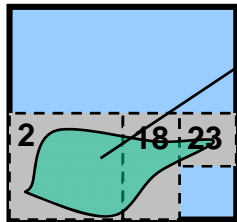
• Verwaltung der Objekte im B<sup>+</sup>-Baum



**Maxlevel = 4**



• Schnitthanfrage

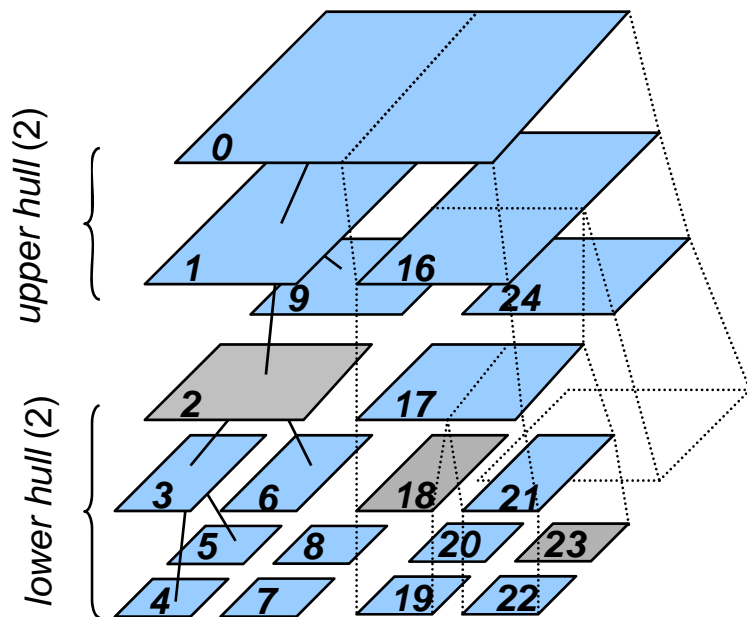


Anfrage-Objekt  $o_1$

**Schnitthanfrage:**

Ermittlung aller Objekte  $o \in DB$  die von  $o_1$  geschnitten werden:

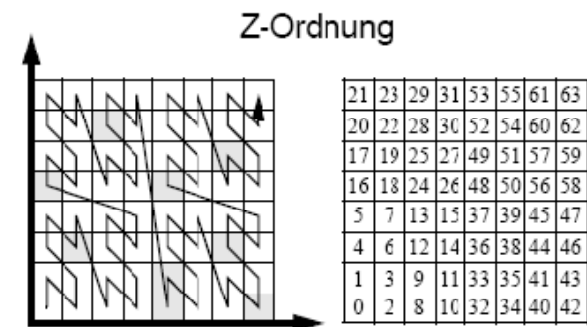
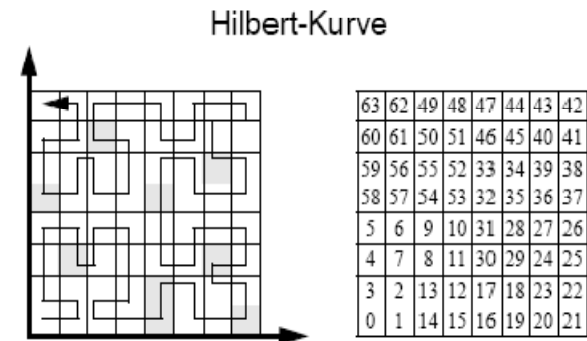
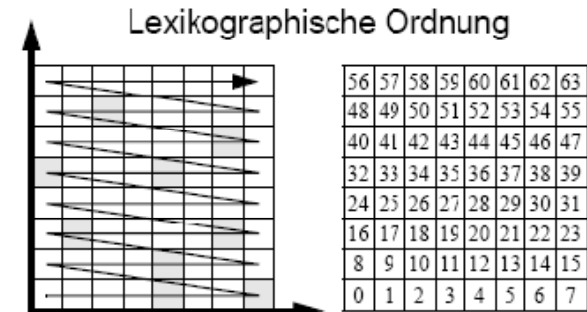
- Anfrage bezieht sich auf die 3 Partitionen: 2, 18 und 23
- Für jede Anfragepartition  $P_i$ : Ermittlung aller Objekte, die  $P_i$  schneiden, d.h.



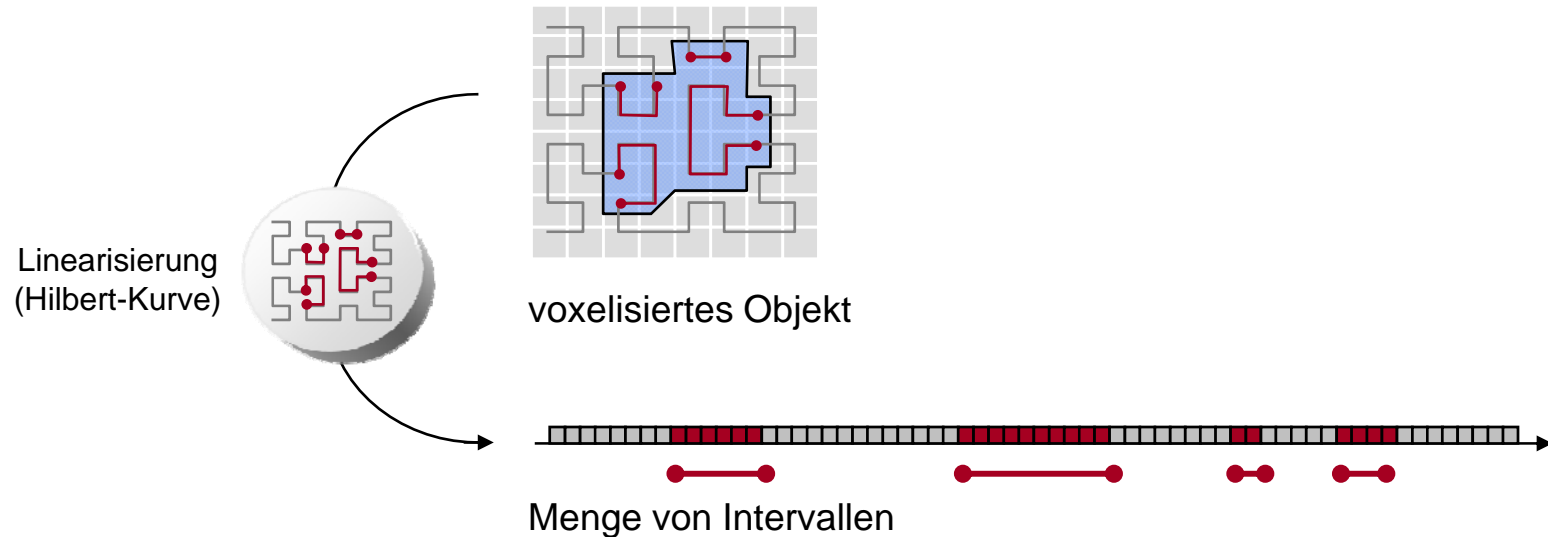
- 1) Ermittlung aller Partitionen  $P_j$  die  $P_i$  schneiden
  - **upper hull** = Partitionen  $P_j$  die  $P_i$  überdecken  
z.B.:  $P_i = 2 \rightarrow upper\ hull = \{0, 1\}$
  - **lower hull** = Partitionen  $P_j$  die von  $P_i$  überdeckt werden  
z.B.:  $P_i = 2 \rightarrow lower\ hull = \{2, 3, 4, 5, 6, 7, 8\}$
- 2) Anfrage im B+-Baum gemäß aller  $P_j = upper\ hull + lower\ hull$   
(kann durch Bereichsanfragen im B+-Baum effizient unterstützt werden)

- Intervall-basierte Objektrepräsentation

- Idee: Einbettung in eindimensionalen Raum
  - » vollständige Zerlegung des Datenraums in gleichförmige disjunkte Zellen (Voxelisierung)
  - » Definition einer linearen Ordnung auf diesen Zellen (mittels raumfüllenden Kurven)
  - » Gruppierung direkt aufeinanderfolgender Zellen zu ein-dimensionalen Intervallen
- Vorteile:
  - » erlaubt auch nicht-rechteckige Repräsentationen
  - » einfache Verwaltung von Intervallen (über konventionelle (ein-dimensionale) Indexstrukturen wie den B<sup>+</sup>-Baum)
- Raumfüllende Kurven:
  - » Lexikographische Ordnung
  - » Hilbert-Kurve
  - » Z-Ordnung
- *Z-Ordnung* erhält die räumliche Nähe relativ gut
- *Z-Ordnung* ist effizient berechenbar



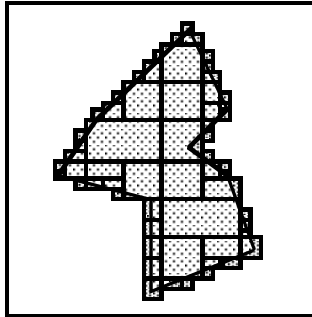
- Intervall-basierte Objektrepräsentation (Fortsetzung):
  - Beispiel:



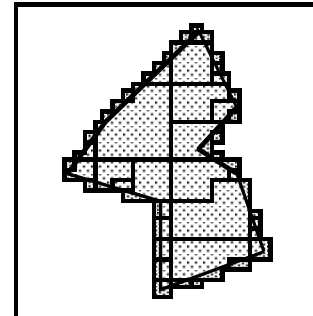
- Indexierung:
  - Codierung von Intervallen
    - » lineare Ordnung entsprechend der Intervallgrenzen (Startpunkt / Endpunkt)  
[Bozkaya and Özsoyoglu: *Indexing Valid Time Intervals*. Int. Conf. on Database and Expert Systems Applications, 1998]
    - » indirekte Codierung der Intervalle, z.B. Relationaler Intervall-Baum  
[H.-P. Kriegel, M. Pötke, T. Seidl: *Managing Intervals Efficiently in Object-Relational Databases*, VLDB 2000]
  - Verwaltung der Intervalle in einem B<sup>+</sup>-Baum

- Vergleich zur Quadtree-basierten Zerlegung:
  - Die Z-Ordnung-basierte Intervall-Zerlegung erzeugt signifikant weniger Redundanz als die Quadtree/Octree-basierte Zerlegung.

Beispiel:



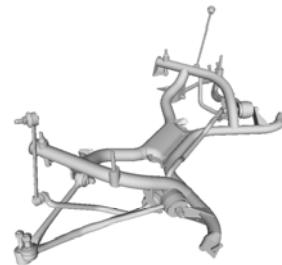
Quadtree-basierte Zerlegung  
(60 Quadranten)



Z-Ordnungsbasierte Zerlegung  
(41 Intervalle)

- Beobachtung:
  - » Anzahl der Partitionen (Quadtree- oder Intervall-basiert) ist proportional zum Umfang (2d) bzw. zur Oberfläche (3d) des Objekts, da eine Raumunterteilung nur zur Randkodierung des Objekts nötig ist.
  - » je kleiner das Verhältnis von Oberfläche zu Volumen, desto höher die Redundanz.
  - » raumpartitionierende Verfahren für komplex strukturierte Objekte wie z.B. Kabelstränge, Leitungen nicht geeignet.

Beispiel für komplexes Objekt:  
Fahrwerk eines Autos



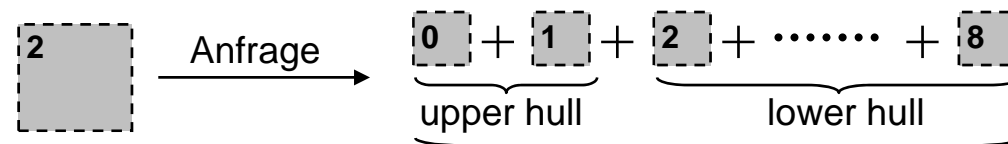
- Räumliche Anfrage:

- Problem:

- » (Anfrage-)Objekte zerfallen in viele Partitionen (Quadtree-basierte Zerlegung)



- » jede Partition des Anfrageobjektes führt zu mehreren Partitions-Schlüsseln, die angefragt werden müssen



Anfrage = Sequenz von Primärschlüssel

⇒ sehr viele (Einzel-)Anfragen notwendig ⇒ Anfrage sehr teuer

- Ziel: Reduktion der Anfragekosten durch

- Vermeidung von Redundanz

- » sammeln aller Anfragepartitionen und Löschen redundanter Partitionen bevor Anfrage startet

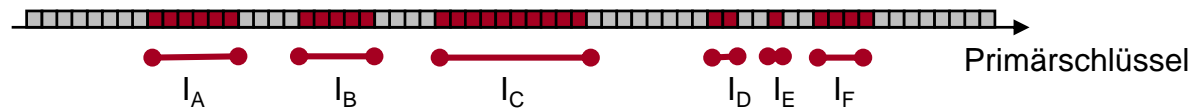
- Geeigneten Indexdurchlauf

- » geeignete Wahl zwischen Scan auf Blattebene und Index-basierter Suche

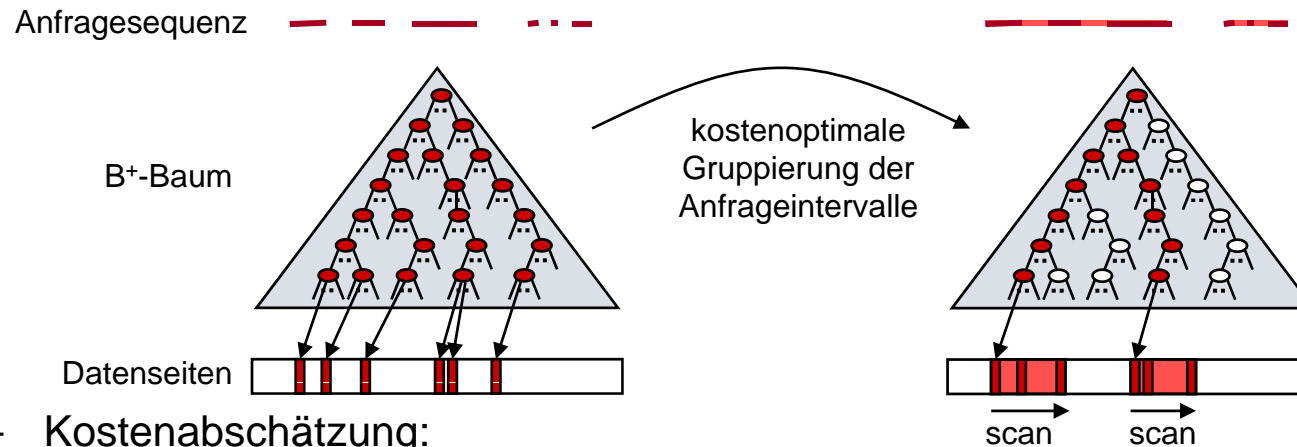
- **Kostenbasierte Suche:**

[Kriegel, Kunath, Pfeifle, Renz: Proc. Int. Conf. on Database Systems for Advanced Applications (DASFAA), 2004]

- Idee: optimale Nutzung des partiellen Scans im B<sup>+</sup>-Baum durch Bildung von Bereichsanfragen
- Gegeben: Anfragesequenz als Menge von Bereichsanfragen

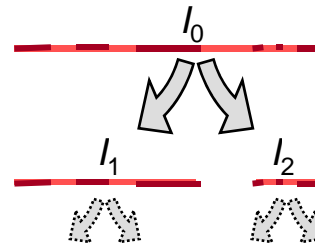


- Prinzip: kostenoptimale Gruppierung von Bereichsanfragen



- **Kostenabschätzung:**
  - » I/O-Kosten für Suche im B<sup>+</sup>-Baum ~ Höhe des B<sup>+</sup>-Baum
  - » I/O-Kosten für Scan auf Blattebene: Abschätzung mittels Statistiken über die Verteilung der Daten auf Blattebene gemäß dem Suchschlüssel

- Kostenbasierte Anfrage (Bildung der finalen Abfragesequenz):
  - » konservative Approximation der Anfrage mit nur einem Anfragebereich  $I_0$
  - » rekursive Zerlegung von  $I_0$  mit dem Ziel die geschätzten Abfragekosten zu minimieren (Zerlege  $I_0$  in  $I_1$  und  $I_2$ , falls  $K(I_1) + K(I_2) < K(I_0)$ )



- » Zerlegungsheuristik: Zerteile approximierten Anfragebereich an der größten Lücke zwischen zwei Anfragebereichen
- » Zerlegungsalgorithmus:

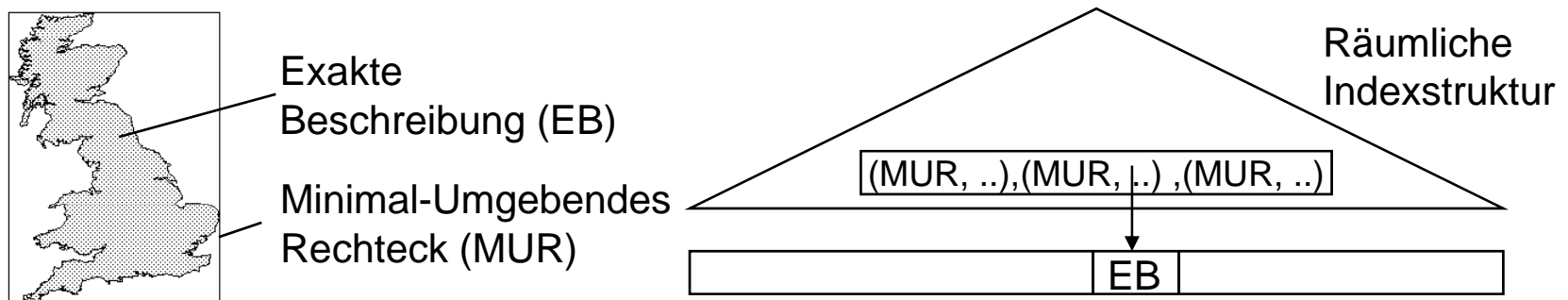
**Decompose( $I_0$ )**

```
{  zerlege  $I_0$  in  $I_1$  und  $I_2$  (z.B. durch Split an der größten Lücke)
  schätze Kosten:  $K(I_0)$ ,  $K(I_1)$ ,  $K(I_2)$ 
  if  $K(I_0) > K(I_1) + K(I_2)$  then
    report {Decompose( $I_1$ ) $\cup$ Decompose( $I_2$ )};
  else
    report  $I_0$ ;
}
```



## 3.1.2 Datenpartitionierende Anfrage-Methoden

- Grundlegende Idee:
  - direkte Verwaltung von räumlichen Objekten anstatt Verwaltung von Raumpartitionen
  - Problem: räumliche Objekte variieren stark in ihrer Größe (Größe = benötigter Speicherplatz), sind aber auf Seiten fester Größe abzuspeichern
  - Organisation von vereinfachten Räumlichen-Objekten (**Approximationen**), z.B. minimal umgebende Rechtecke (MUR)
  - Verweis auf die exakte Beschreibung (EB) der Räumlichen-Objekte



## – Arten von Approximationen

- Konservative Approximationen

- enthalten das zu approximierende Objekt vollständig
- dienen insbesondere zur Bestimmung von Fehltreffern

Beispiel:  $\neg(a.kons\_appr \cap b.kons\_appr) \Rightarrow \neg(a \cap b)$

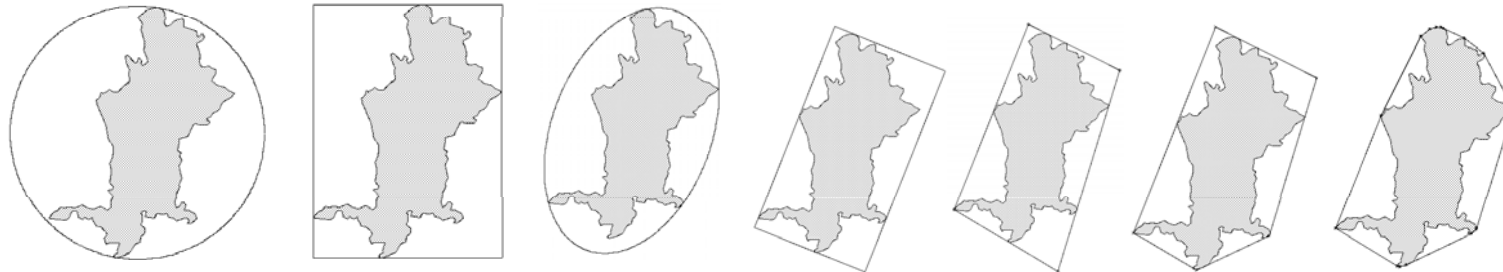
- Progressive Approximationen

- sind vollständig im zu approximierenden Objekt enthalten
- dienen insbesondere zur Bestimmung von Treffern

Beispiel:  $(a.prog\_appr \cap b.prog\_appr) \Rightarrow (a \cap b)$

## – Konservative Approximationen

- Vergleich verschiedener konservativer Approximationen (2D)



<b>Approximation:</b>	Kreis	MUR	Ellipse	gedrehtes MUR	4-Eck	5-Eck	Konvexe Hülle
<b>Güte:</b>	215%	193%	168%	162%	144%	133%	123%
<b># Parameter:</b>	3	4	5	5	8	10	$O(n)$

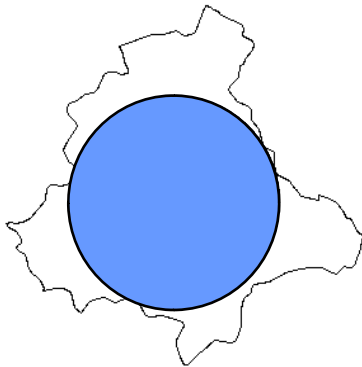
(Güte = durchschnittliche Flächen der Approximationen in Prozent zur exakten Objektfläche (=100%). Gemittelt über verschiedene Landkartendatensätze.)

Quelle: Brinkhoff T., Kriegel H.-P., Schneider R.: 'Comparison of Approximations of Complex Objects used for Approximation-based Query Processing in Spatial Database Systems', Proc. 9th Int. Conf. on Data Engineering, 1993)

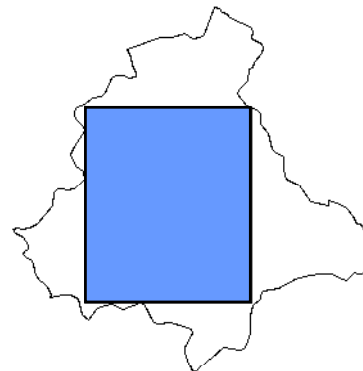
**→ 5-Eck: guter Kompromiss zwischen Genauigkeit und Speicherplatzbedarf**

## – Progressive Approximationen

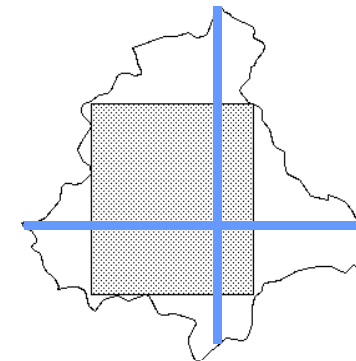
- Berechnung schwierig (insbesondere für maximale progressive Approximationen)



Kreis  
(42%)



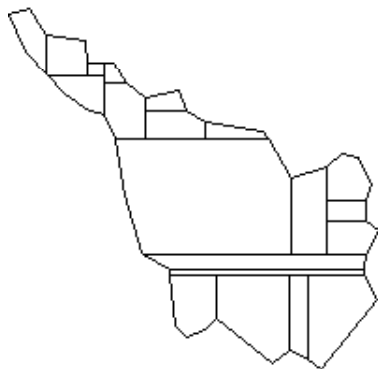
Rechteck  
(45%)



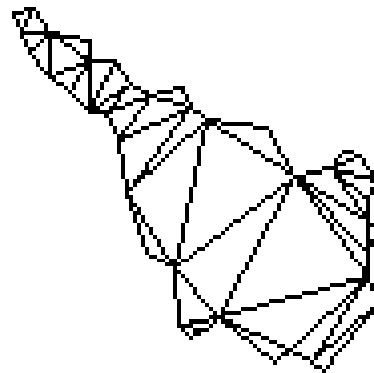
Rechteck + Strecken

## – Strukturelle Zerlegung

- Prinzip:
  - Zerlegung der Objekte in mehrere einfache Basiskomponenten
- Vorteile:
  - Vereinfachung der algorithmischen Komplexität von Anfragen und Operationen (z.B. Flächenberechnung, etc.)
  - Lokalität von Anfragen und Operationen kann ausgenutzt werden (Unterstützung von partiellen Objekt-Zugriff)  
=> effiziente Anfragebearbeitung (bei selektiven Anfragen)
- Zerlegungsvarianten:



konvexe Zerlegung

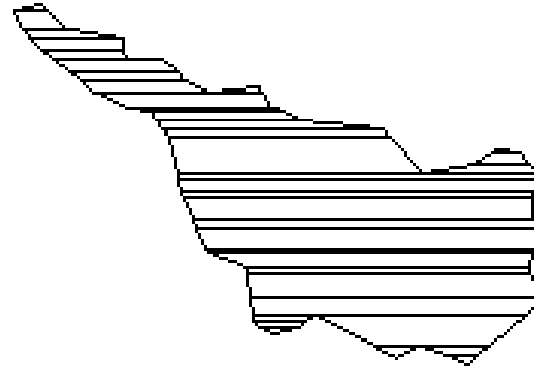


Dreiecks-Zerlegung

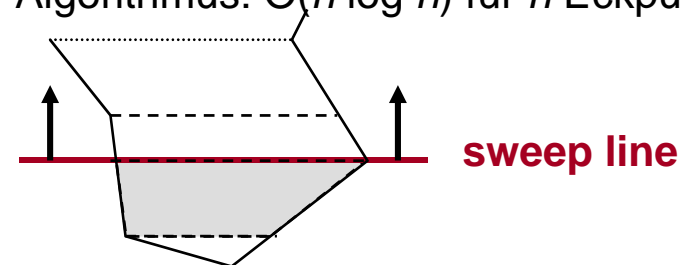


Zerlegung in Trapeze

- Zerlegung eines Polygons in (achsenparallele) Trapeze (Asano+Asano 1983)

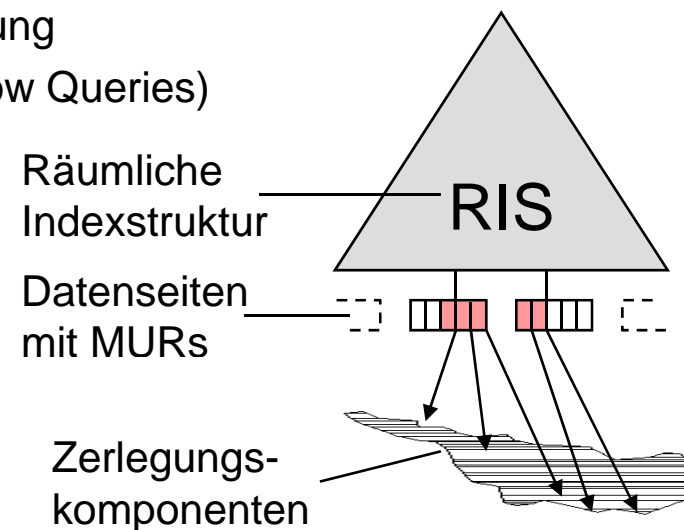


- Berechnung durch Plane-Sweep-Algorithmus:  $O(n \log n)$  für  $n$  Eckpunkte

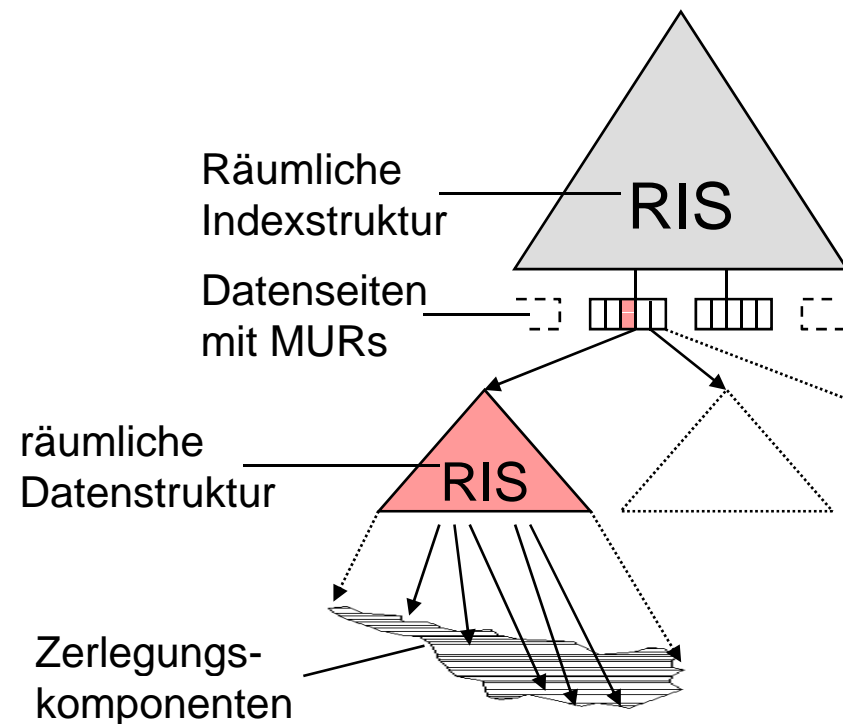


- Speicherplatzaufwand
  - » Anzahl der Komponenten =  $n$  bei  $n$  Eckpunkten
  - » Aber: Vervielfachung des Speicherplatzes da nun Trapeze statt Punkten verwaltet, d.h. abgespeichert werden

- Beobachtung
  - Operationen auf Zerlegungskomponenten sind einfach
  - Aber: Anzahl der Zerlegungskomponenten =  $O(n)$
  - Kein Gewinn, falls alle Komponenten getestet werden
  - Einsatz von Datenstrukturen zur Auswahl “relevanter” Komponenten
  - Einsatz von räumlichen Indexstrukturen (RIS)
- 1. Ansatz
  - Eine RIS verwaltet Zerlegungskomponenten aller Objekte
  - Redundanz bei der Anfragebearbeitung  
(betrifft insbesondere größere Window Queries)



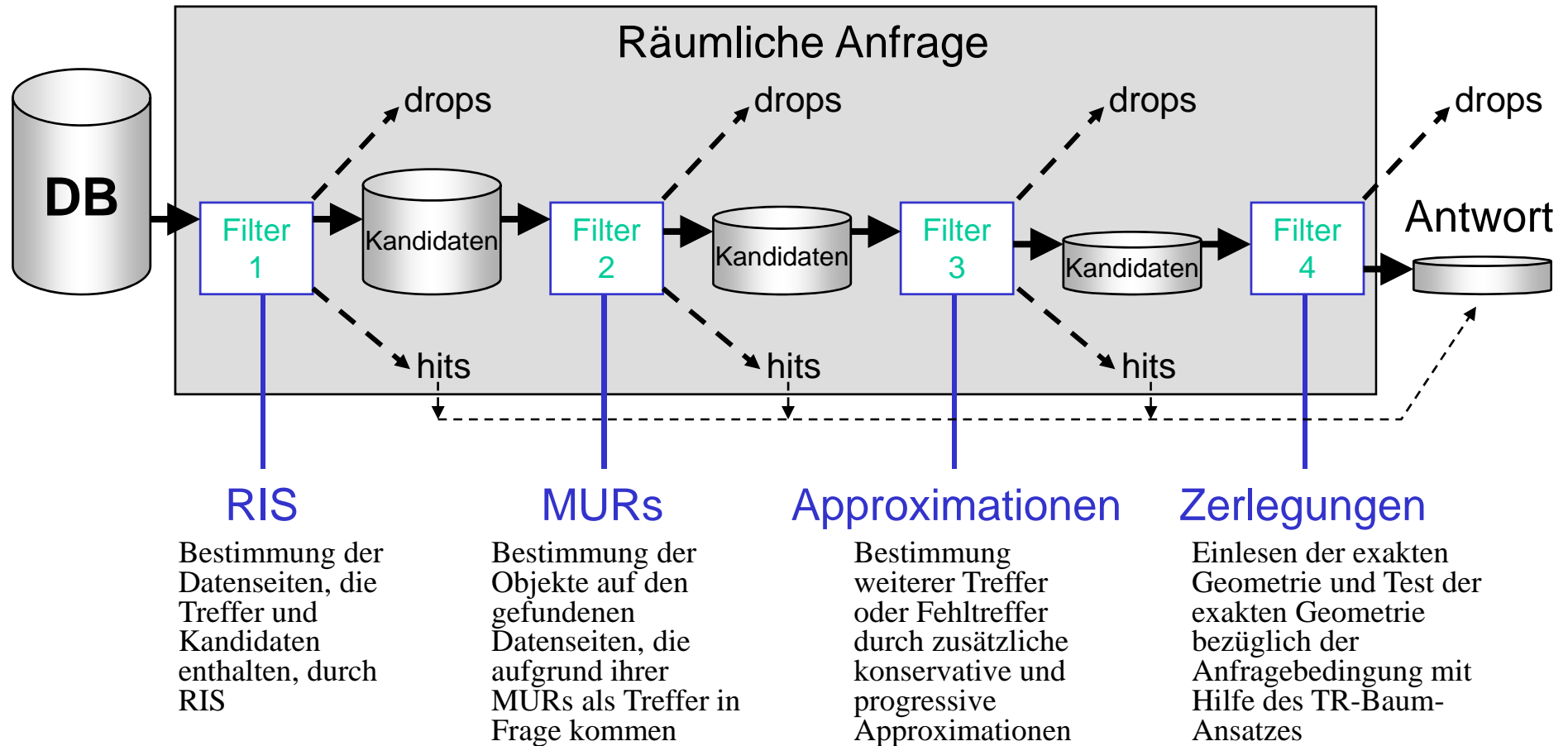
- 2. Ansatz
  - Eine RIS verwaltet die Objektapproximationen (MUR) aller Objekte
  - Für jedes Objekt verwaltet eine separate räumliche Datenstruktur die Zerlegungskomponenten dieses Objektes
  - Wenn die exakte Objektgeometrie untersucht werden muss, werden die Zerlegungskomponenten einschließlich der zugehörigen räumlichen Datenstruktur vom Sekundärspeicher in den Hauptspeicher eingelesen





- TR-Ansatz (Ausprägung des 2. Ansatzes)
  - Verwende R-Baum zur Verwaltung der Zerlegungskomponenten
- ➔ Anpassung des R-Baums an die neuen Anforderungen (TR-Baum):
  - TR-Baum soll für den Hauptspeicher ausgelegt werden
    - » möglichst kleine Knotengröße
  - TR-Baum soll möglichst schnell in den Hauptspeicher eingelesen werden
    - » kompakte Speicherung auf dem Plattenspeicher
    - » kein dynamischer Aufbau, keine Adressneuberechnungen
  - TR-Baum sollte möglichst kompakt gespeichert sein
- Eigenschaften des TR-Baums
  - sehr schnelle Bearbeitung geometrischer Operationen (z.B. Punkt-In-Polygon-Test)
  - erheblich höherer Speicherplatzbedarf
  - (und damit höhere Übertragungskosten beim Einlesen der exakten Geometrie)

– Mehrstufige Anfragebearbeitung für räumliche Objekte



- GEMINI: Prototyp-System, das diese Anfragebearbeitung realisiert