Kapitel 2
Prinzipien der Anfragebearbeitung in STMM-DBS

Skript zur Vorlesung: Spatial, Temporal, and Multimedia Databases
Sommersemester 2012, LMU München

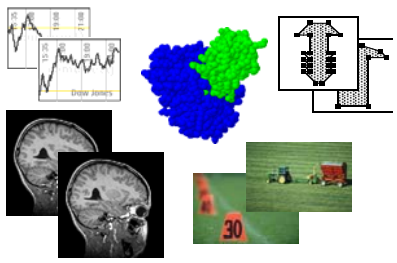
Übersicht

2.1 Feature-Räume

2.1.1 Das Prinzip der feature-basierten Ähnlichkeit

– Grundidee der Feature-Transformation
  - Erwünscht: effiziente Ähnlichkeitssuche in Datenbanken
  - Meist ist Effizienz ohne Einsatz von Indexstrukturen nicht zu verwirklichen
  - Entwickle nicht für jedes der einzelnen Anwendungsgebiete/ Ähnlichkeitsmaße spezielle Indexstrukturen
  - Versuche mit wenigen Arten von Indexstrukturen möglichst viele Anwendungsgebiete abzudecken
  - Indexstrukturen für:
    – Multidimensionale Vektoren
    – Allgemein metrische Daten (beliebige Objekte, auf denen eine metrische Distanzfunktion definiert ist)

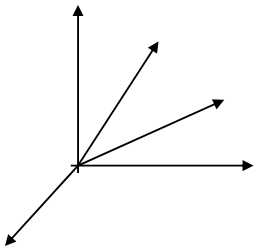
- Extrahiere charakteristische (nummerische) Eigenschaften („Features“) aus den Objekten



Anwendungsgebiete (Objektraum)

Feature-Transformation

Histogramm-Bildung
Moment-Invarianten
Rechteck-Abdeckung
Sektoren-Ermittlung
Fourier-Transformation Kurvatur
usw.



Featurevektor-Raum

- Wichtigste Eigenschaft der Feature-Transformation:
  – Ähnlichkeit der Objekte entspricht geringem Abstand der Feature-Vektoren
  => Ähnlichkeitsanfragen im Objektraum entsprechen Nachbarschaftsanfragen im Feature-Raum
  => Unterstützung durch geeignete multidimensionale Indexstrukturen

– Erweiterungen
  - Oft reichen die Mächtigkeit von Vektoren für die Modellierung nicht aus (vergleiche z.B. Relationales und OO-Modell)
  - Transformation in andere Räume, die die Definition einer Distanzfunktion mit Metrik-Eigenschaften erlauben
    – Graphen
    – Punktmengen
    – …

– Fazit:
  - Das Prinzip der Feature-Transformation ist ein mächtiges Werkzeug zur Modellierung der Ähnlichkeit von komplexen STMM-Objekten
  - Herausforderung: Finden geeigneter Feature-Transformationen

2.1.2 Feature-Räume und Distanzen

– Allgemeiner Feature-Raum

Ein Feature-Raum ist ein Tupel Φ = (Dom, dist) mit
  - Dom ist ein Wertebereich (Domain)
  - dist ist eine Distanzfunktion, d.h. es gilt
    – Reflexivität $\forall x,y \in Dom: dist(x,y) = 0 \Leftrightarrow x=y$
    – Positiv-Definitheit $\forall x,y \in Dom, x \neq y: dist(x,y) > 0$
    – Symmetrie $\forall x,y \in Dom: dist(x,y) = dist(y,x)$

– Metrischer Raum

Ein metrischer Raum ist ein Tupel Φ_M = (Dom, dist) mit
  - (Dom,dist) ist ein allgemeiner Feature-Raum
  - dist erfüllt zusätzlich die
    – Dreiecksungleichung $\forall x,y,z \in Dom: dist(x,z) \leq dist(x,y) + dist(x,z)$

– Vektorraum

Ein (euklidischer) Vektorraum der Dimension *d* (*d*-dimensionaler Vektorraum) ist ein Tupel Φ_E = (Dom, dist) mit

- (Dom,dist) ist ein metrischer Raum
- Dom = $\mathbb{R}^d$

– Feature-Transformation

Eine Feature-Transformation ist eine Abbildung

T: OBJ $\rightarrow$ (Dom,dist)

die jedem Objekt o$\in$OBJ aus dem Objektraum ein Objekt aus dem Wertebereich Dom zuordnet.

Die Distanz im Objektraum wird durch die Distanz im Feature-Raum repräsentiert, d.h.

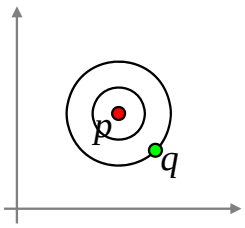
$\forall x,y \in OBJ: dist_{OBJ}(x,y) \equiv dist_{Dom}(T(x),T(y))$

– Distanzmaße in Vektorräumen

- Euklidische Norm (L_2):

  $dist = ((p_1-q_1)^2+(p_2-q_2)^2+...)^{1/2}$

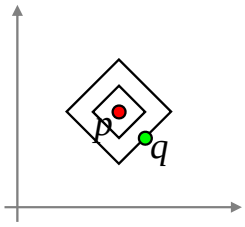
  Natürliches Distanzmaß



- Manhattan-Norm (L_1):

  $dist = |p_1-q_1|+|p_2-q_2|+...$

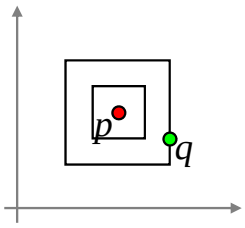
  Die Unähnlichkeiten der einzelnen Merkmale werden direkt addiert



- Maximums-Norm (L_∞):

  $dist = \max\{|p_1-q_1|, |p_2-q_2|,...\}$

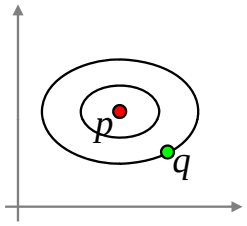
  Die Unähnlichkeit des am wenigsten ähnlichen Merkmals zählt



- Verallgemeinerung L_p-Abstand: $dist_p = (|p_1-q_1|^p + |p_2-q_2|^p + ...)^{1/p}$
- Gewichtete Euklidische Norm:

  $dist = (w_1(p_1-q_1)^2+w_2(p_2-q_2)^2+...)^{1/2}$

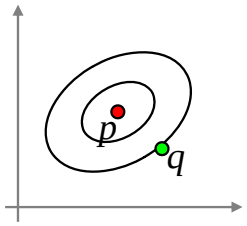
  Häufig sind die Wertebereiche der Merkmale deutliche unterschiedlich
  Beispiel: Merkmal $M_1 \in [0.01 \ldots 0.05]$
  Merkmal $M_2 \in [3.07 \ldots 22.2]$
  Damit M1 überhaupt berücksichtigt wird muss es höher gewichtet werden



- Quadratische Form:

  $dist = ((p - q)\ \mathbf{M}\ (p - q)^T)^{1/2}$

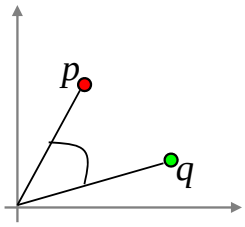
  Bisherige Abstandsmasse gewichten Merkmale nur getrennt
  Besonders bei Farbhistogrammen müssen verschiedene Merkmale gemeinsam gewichtet werden



- Cosinus-Distanz

  $dist$ = cos(winkel(p,q))

  Berechnet den cosinus des Winkels
  Meist für sehr hochdimensionalen Featurevektoren (z.B. Texten)



– Bemerkungen

- Jeder Vektorraum ist ein metrischer Raum, jeder metrische Raum ein allgemeiner Feature-Raum
- Sprechweise meist: „Feature-Raum“ statt (euklidischer) Vektorraum
- Transformation komplexer Objekte meist immer in metrische Räume wegen der Dreiecksungleichung (Performanz!!!)

2.2 Algorithmische Paradigmen zur Anfragebearbeitung

2.2.1 Übersicht

- Typen von Ähnlichkeitsanfragen
  - Bereichsanfragen
  - Nächste Nachbarn Anfragen
  - Reverse Nächste Nachbarn
  - Skyline Anfragen
- Algorithmische Paradigmen
  - Naive (sequentielle) Suche
    - Für alle *n* Objekte der Datenbank wird das Anfrage-Prädikat (d.h. meist eine Distanzberechnung zum Anfrageobjekt) ausgewertet
    - Kosten: O(*n* · Kosten für das Anfrage-Prädikat)
  - Indexbasierte Suche
  - Mehrstufige Anfragebearbeitung

2.2.2 Indexstrukturen

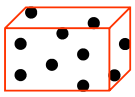
- Prinzip [Böhm, Berchtold, Keim. ACM Computing Surveys, 2001]
  - Organisiere Objekte der Datenbank so, dass während einer Ähnlichkeitsanfrage nur auf „relevante“ Objekte zugegriffen werden muss
  - Baumartige Organisation; jedem Knoten des Baumes ist zugeordnet
    - Seite des Hintergrundspeichers
    - Region des Datenraums
  - Typen von Knoten (= Seiten)
    - Blattknoten sind Datenseiten, speichern Objekte
    - Innere Knoten sind Directoryseiten, speichern Directory-Einträge
      - » Verweis zur Kindseite (Adresse auf dem Hintergrundspeicher)
      - » Beschreibung der Region der Kindseite

- Physische vs. logische Seiten
  - ursprünglich: eine physische Seite des Hintergrundspeichers wird verwendet
    - » kleinste Informationseinheit, die zwischen Plattenspeicher und RAM übertragen werden kann
    - » ABER: physische Seiten meist zu klein
  - daher: logische Seiten fassen aufeinanderfolgende physische Seiten zusammen
  - Meistens: einheitliche Seitengröße für alle Seiten eines Indexes (um komplizierte Freispeicherverwaltung zu vermeiden)
  - Kapazität der Directory-/Datenseiten (max. Anzahl der Einträge)

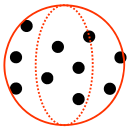
$$c_{\text{Data}} = \left\lfloor \frac{\text{Seitengröße} - \text{Verwaltungsoverhead}}{\text{Größe eines Datensatzes}} \right\rfloor \qquad c_{\text{Directory}} = \left\lfloor \frac{\text{Seitengröße} - \text{Verwaltungsoverhead}}{\text{Größe eines Directoryeintrags}} \right\rfloor$$

  - Meist keine 100% Füllung der Seiten (Platz für neue Datensätze) aber minimale Füllung (z.B. 40%) wegen Speicherauslastung
  - Speicherauslastung ist die durchschnittliche Anzahl besetzter Einträge
- Jedes Objekt wird in genau einer Datenseite gespeichert
- Regionen gewährleisten, dass ähnliche Objekte möglichst auf den selben Datenseiten (oder Teilbäumen) gespeichert werden

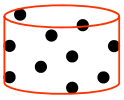
- Gestalt der Seitenregionen
  - Vektordaten:
    - » Achsenparallel Rechtecke, die minimal um die Punktmenge gespannt werden (MUR=minimal umgebendes Rechteck/ MBR=minimum bounding rectangle) (R-Tree, R*-Tree, X-Tree, …)



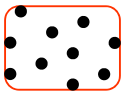
    - » Kugeln (SS-Tree)



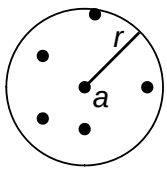
    - » Zylinder (TV-Tree)



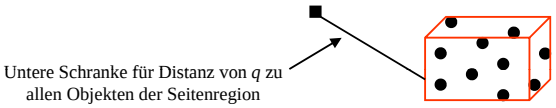
    - » Kombinationskörper (Kugel+MBR, SR-Tree)



  - Algemein metrische Daten
    - » Kein „Raum“ im Euklidischen Sinne, keine Geometrie
    - » Ankerobjekt *a* (anchor object) + Hüllradius *r* (covering radius)
      Für alle Objekte *o* der Seitenregion gilt: dist(*o*,*a*) ≤ *r*



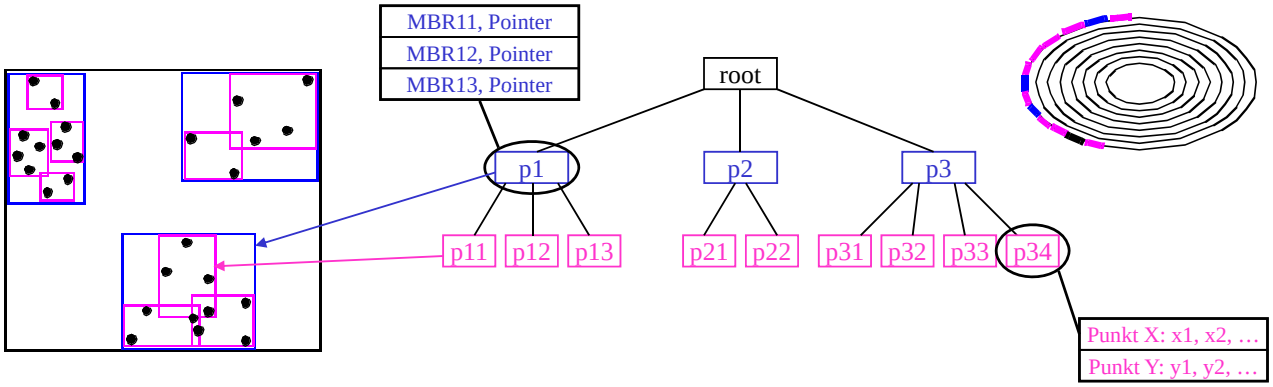
- Seitenregion umfasst immer alle Objekte der Datenseite bzw. des Teilbaums der Directoryseite
  - Konservative Approximation, lower-bounding property: die Distanz von einem Objekt zu einer Seitenregion kann immer mit einer unteren Schranke abgeschätzt werden



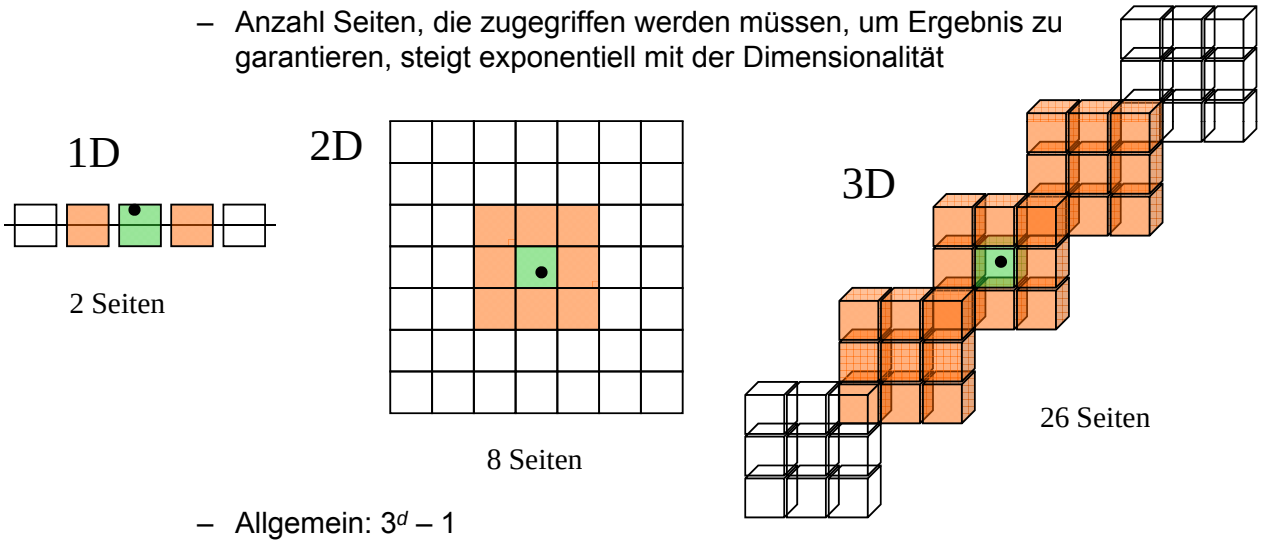
  - Damit kann man Vollständigkeit der Anfrageergebnisse gewährleisten
- Bäume sind meist balanciert (alle Blätter auf selbem Level)
- Indexstrukturen sind dynamisch, d.h. Insert/Delete sind effizient
  - Tiefensuche nach entsprechender Datenseite (auf einen Pfad beschränkt)
  - Einfügen/Löschen des Objektes
  - Überlauf/Unterlauf-Behandlung durch Split/Merge
    - » Verschiedene Kriterien (minimale Überlappung, toter Raum, …)
    - » Verschiedene Algorithmen (linear, quadratisch, …)
    - » Ensprechendes Einfügen/Löschen im Elternknoten (rekursiv, notfalls bis Wurzel)

– Beispiel: R*-tree

- Entworfen für 2D Rechtecksdaten (Punkte sind spezielle Rechtecke), oft verwendet für hochdimensionale Vektordaten
- Design
  - Balancierter Index mit einheitlich großen Daten-/Directoryseiten
  - Seitenregionen: MBRs
  - Insert-Strategie: Overlap, Volumen
  - Überlaufbehandlung: Forced Re-Insert-Konzept
  - Splitkriterien: Umfang/Oberfläche, Überlappungsvolumen, (toter Raum)



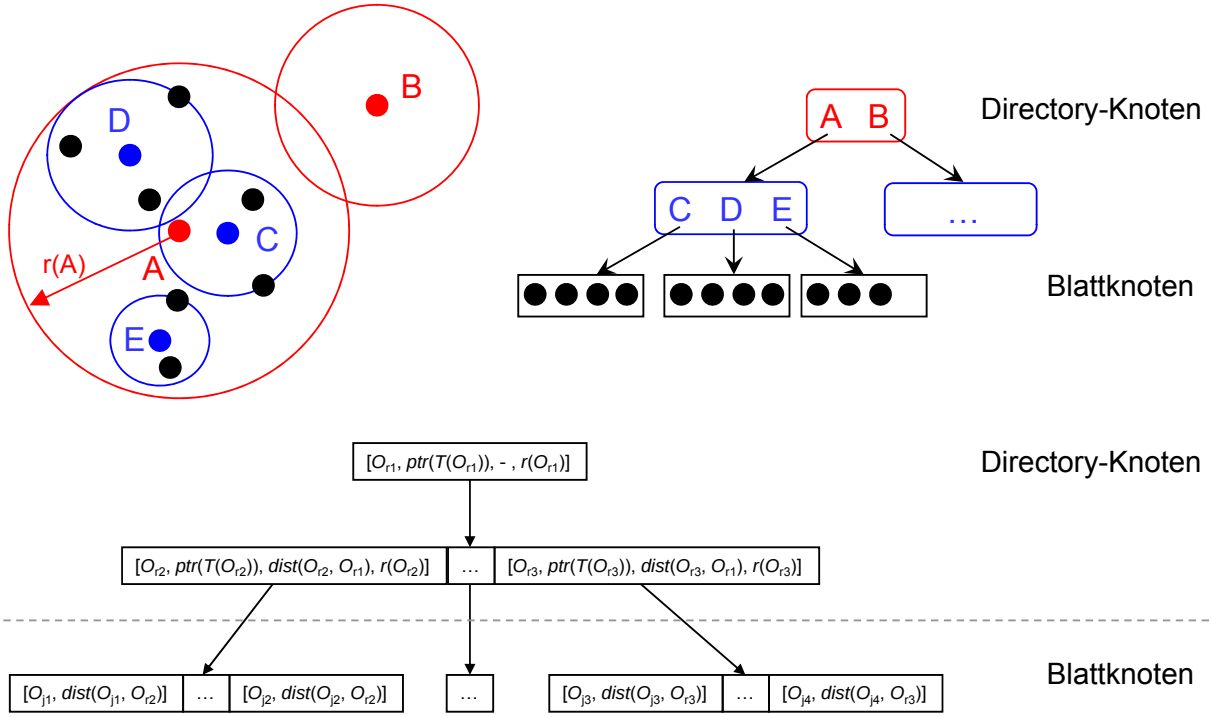
– Curse of Dimensionality (Vektordaten)
  - Anfrageleistung von Indexstrukturen verschlechtern sich mit zunehmender Dimension
    – Anzahl Seiten, die zugegriffen werden müssen, um Ergebnis zu garantieren, steigt exponentiell mit der Dimensionalität



    – Allgemein: $3^d - 1$
    – Seitenregionen überlappen sich stärker
    – Folge: oft muss komplettes Directory gelesen werden

– Beispiel: M-tree
  - Dynamische Indexstruktur für allgemeine metrische Räume
  - Die Distanzfunktion zur Berechung der Ähnlichkeit zweier Objekte muss die Eigenschaften einer Metrik erfüllen
  - Design
    – Balancierter Index mit einheitlich großen Daten-/Directoryseiten
    – Die indexierten Datenbank-Objekte werden in den Blattknoten abgespeichert
    – Die Directory-Knoten enthalten sog. *Routing Objekte*
    – Routing Objekte entsprechen Databank-Objekten, denen eine *Routing Rolle* zugewiesen wurde
    – Wenn ein Knoten überläuft und geplittet werden muß, vergibt der Splitalgoritmus eine Routing Rolle an ein Objekt
    – Zusätzlich zur Objektbeschreibung enthält ein Routing Objekt einen Zeiger auf seinen zugehörigen Unterbaum und den Radius, in dem sich alle Objekte des Unterbaums befinden
    – Wahl der Routing Objekte: die beiden am weitesten voneinander entfernt liegenden Objekte der übergelaufenen Seite

– M-tree Struktur



2.2.3 Mehrstufige Anfragebearbeitung

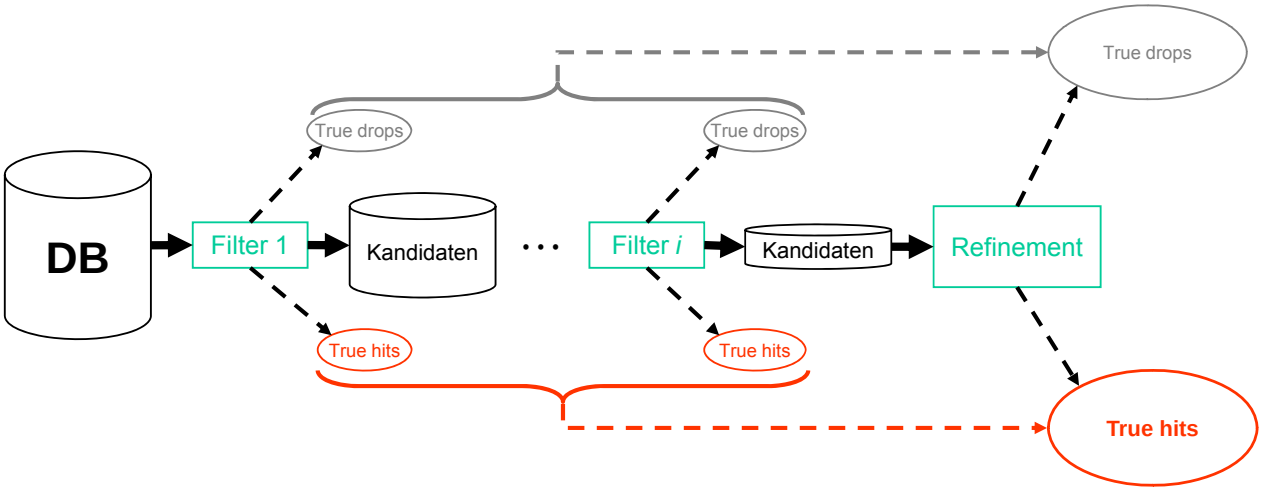
– Idee:

- Verwendetes Distanzmaß ist sehr teuer (z.B. Edit-Distanz) oder nicht feature-basiert (z.B. Überlappungsfläche von Polygonen)
- Vektorraum sehr hochdimensional (Curse of Dimensionality)
- Benutze ein feature-basiertes (meist niedrig-dimensionaler Vektorraum) Distanzmaß als Filterschritt (Filterdistanz)
  - Filterdistanz sollte billiger sein als exakte Distanz (entsprechend niedrig-dimensional => Dimensionsreduktion?)
  - Werte Anfrage-Prädikat (Distanzberechnung) mit Filterdistanz aus
  - Ergebnisse sind noch keine exakten Treffer sondern Kandidaten
  - Kandidatenmenge sollte möglichst klein sein (Filterselektivität)
  - Filterselektivität

$$\sigma_F = \frac{\#\text{Kandidaten}}{n}$$

  - Verfeinerung: für die Kandidaten wird das eigentliche Distanzmaß berechnet, was i.A. teurer dafür selektiver als der Filterschritt ist

– Mehrstufige Anfragebearbeitung:
  - Ein oder mehrere (kaskadierende) Filterschritte schränken die Kandidatenmenge sukzessive ein
  - Verfeinerungschritt testet auf Korrektheit der Kandidaten



– Zusammenpassen von Filter und Refinement
  - Idealfall: Filterdistanz ist obere oder unter Schranke (upper/ lower bound) der exakten Distanz => es kann garantiert werden, dass keine exakten Treffer verloren gehen (no false dismissals/drops)
  - Sonst: Ergebnisse u.U. nicht vollständig!!!
  - Lower Bounding Filter F_{LB}

$$\forall x, y \in DB : dist_{F_{LB}}(F_{LB}(x), F_{LB}(y)) \leq dist(x, y)$$

    – Konservative Approximation (d.h. Obermenge) der Ergebnismenge
    – Objekte können evtl. bereits aufgrund der Filterdistanz als exakte Fehltreffer (true drops) identifiziert werden
  - Upper Bounding Filter F_{UB}

$$\forall x, y \in DB : dist_{F_{UB}}(F_{UB}(x), F_{UB}(y)) \geq dist(x, y)$$

    – Progressive Approximation (d.h. Untermenge) der Ergebnismenge
    – Objekte können evtl. bereits aufgrund der Filterdistanz als exakte Treffer (true hits) identifiziert werden