course:

Searching the Web (NDBI038)

Searching the Web and Multimedia Databases (BI-VWM)

lecture 11:

Similarity queries and multi-modal search

prof. RNDr. Tomáš Skopal, Ph.D.
Department of Software Engineering, Faculty of Mathematics and Physics, Charles University, Prague
Department of Software Engineering, Faculty of Information Technology, Czech Technical University in Prague

Today's lecture outline

- fundamentals
- similarity queries
  - similarity ordering, range, kNN, kRNN queries
  - similarity joins, self-joins, (k) closest pairs
- multi-modal search
  - early fusion
    - multi-metric model
  - late fusion
    - skyline operator
    - top-k operator
    - re-ranking
    - query languages

Fundamentals

- similarity search = a content-based retrieval concept
- formalized model
  - **feature extraction** procedure $\mathbf{e: X \rightarrow U}$
    - transforming a multimedia object from database universe **X** into a descriptor of descriptor universe **U**
      - the original database $\mathbf{D \subset X}$,
      - the descriptor database $\mathbf{S \subset U}$
  - **distance (dissimilarity) function** $\mathbf{\delta: U \times U \rightarrow R}$
    - i.e., close means similar
  - similarity queries: query-by-example paradigm
    - similarity query defined by its **type**, an **example** object **q** and an **extent** of the query
    - returns **ranked query result**, consisting of descriptors based on their closeness to **q**

Similarity queries

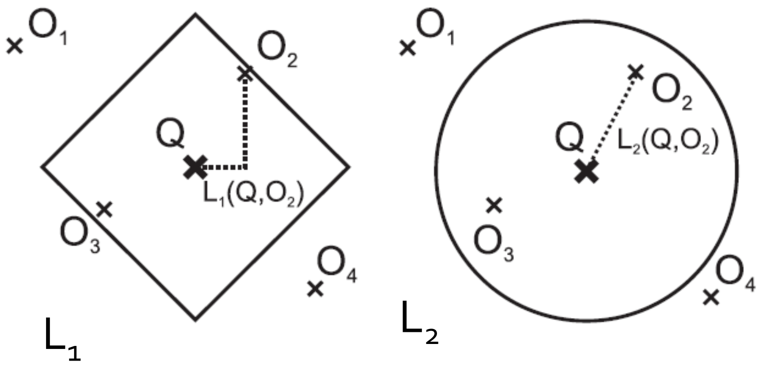
- similarity ordering
  - given a query object $\mathbf{q} \in \mathbf{U}$ and the descriptor universe $\mathbf{U}$,

  - SimOrder: $\mathbf{U} \rightarrow \mathbf{S}^{|S|}$,
    where $\forall \mathbf{i} \in (1, |\mathbf{S}|)$: $\delta(\mathbf{q}, \text{SimOrder}(\mathbf{q})[i]) \leq \delta(\mathbf{q}, \text{SimOrder}(\mathbf{q})[i+1])$
    - informally, SimOrder is the database **S** ordered desc. by distance of their elements to **q**

  - SimOrder is the basic concept when defining a similarity query
  - just an abstraction (i.e., SimOrder is not fully materialized when querying)

Similarity queries

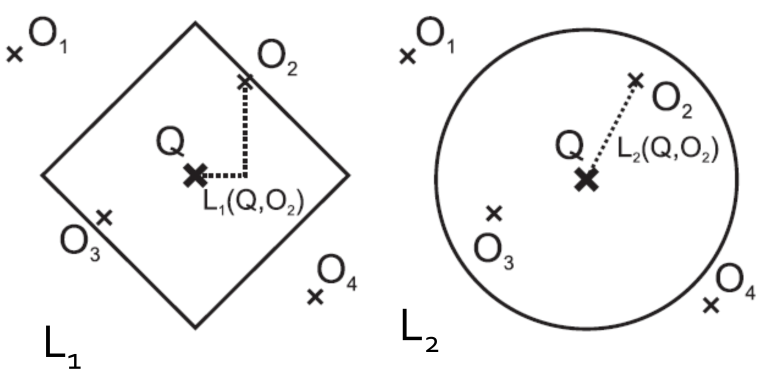
- range query
  - given a distance radius **r** (dissimilarity threshold), a range query returns all database descriptors the distances of which to **q** is no more than **r**
    - i.e., a prefix **P** ⊂ SimOrder(**q**), such that **x**∈**P**, δ(**q**, **x**) ≤ **r**
    - shortly, (**q**, **r**) = {**x**∈**S** | δ(**q**, x) ≤ **r**}
  - a "ball "in the distance space
    - just visualization, as geometry is meaningful only in vector spaces
    - delimits a region (subset) in **S**



$$L_p(v_1, v_2) = (\sum_{i=1}^{D} |v_1[i] - v_2[i]|^p)^{\frac{1}{p}} \qquad (p \geq 1)$$

Similarity queries

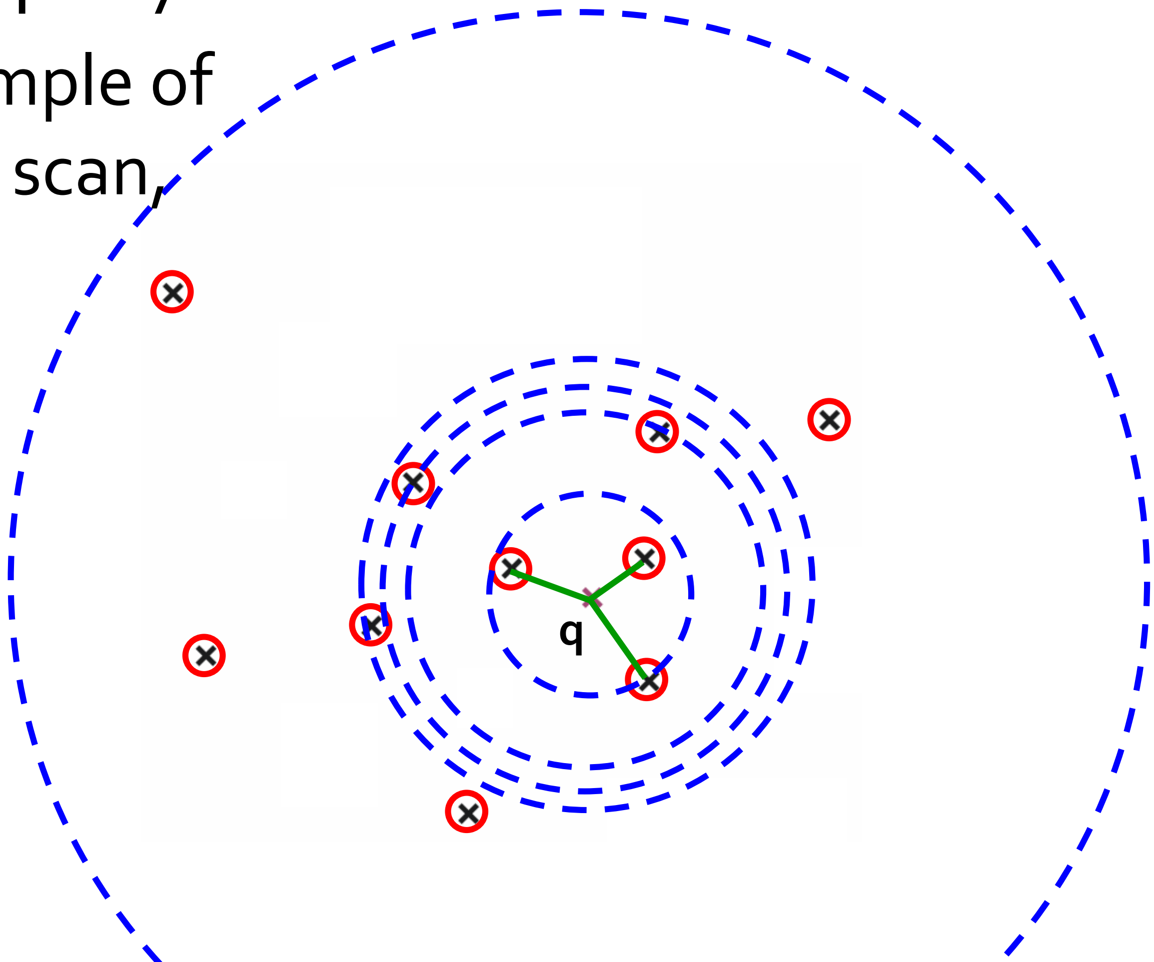
- k nearest neighbor (kNN) query
  - given a number of desired descriptors k,
    a kNN query returns k database descriptors closest to **q**
    - i.e., a prefix **P** $\subset$ SimOrder(**q**), such that **|P| = k**
    - shortly, (**q**, **k**) = {**C** | **C**$\subseteq$**S**, |**C**|=**k**, $\forall$**x**$\in$**C**, **y**$\in$**S**-**C**$\Rightarrow$ δ(**q**,**x**) ≤ δ(**q**,**y**) }
    - ties are solved arbitrarily
      (**k**th and (**k**+1)th NN could be
      the same close to **q**)
  - also "ball "in the distance space
    - but the query radius
      is not known in advance
    - delimits a region (subset) in **S**



$$L_p(v_1, v_2) = (\sum_{i=1}^{D} |v_1[i] - v_2[i]|^p)^{\frac{1}{p}} \qquad (p \geq 1)$$

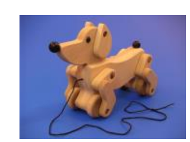
Similarity queries

- kNN query
  - example of seq. scan, k=3



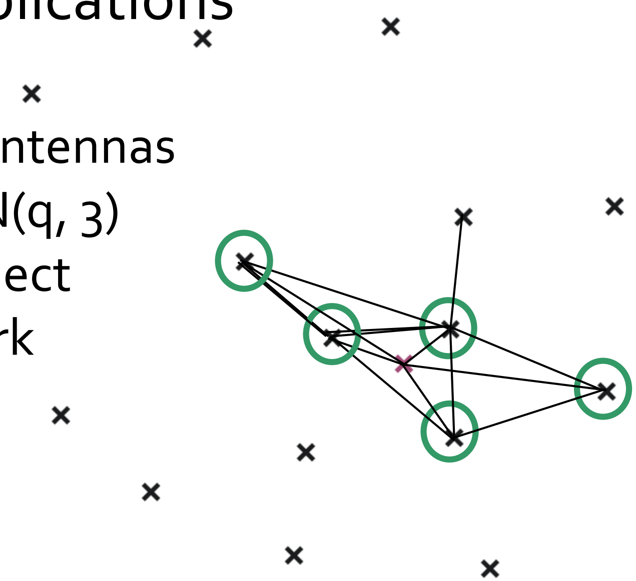
Similarity queries

- range vs. kNN queries
  - **range query** appropriate when
    - end-user is able to specify **r**, i.e., knows the semantics of the model
    - e.g., edit distance on strings,
      counting the smallest number of character edits to transform s_1 into s_2
      range query ('drier', 2) = {driver, d_iver, _river, drive_}
    - 100% recall is guaranteed (because of user's confidence on **r**)
  - **kNN query** appropriate when
    - user cannot specify **r**, i.e., does not know the semantics of the model
    - majority of cases

4NN, q=  ⇨  0.2  0.3  0.5  0.6  0.7

Similarity queries

- *k* reverse nearest neighbors (kRNN)
  - not very frequent, but interesting query type
  - for a query descriptor **q**, *k*RNN returns all descriptors x_i from the database for which $\mathbf{q} \in \text{kNN}(x_i, k)$
  - identifies the closest distinct neighborhood around **q**
- mostly used in spatial databases applications
  - e.g., in a GIS application,
    let the descriptors be positions of GSM antennas
    (**q** is a planned one), if the result of kRNN(q, 3)
    is large enough, **q** is needed to interconnect
    the other antennas into a reliable network
- could be used as similarity query
  - e.g., for identification of redundancy



Similarity operators

- database operator = an operation on the database with result
- query vs. operator
  - query – the query expression/example is a parameter
    - small subset of the database is expected as a result
    - repeated queries of the same type on a static database makes sense (query expression is changing)
  - operator – often defined as non-parameterized operation
    - repeated processing of non-parameterized operator on a static database leads to the same result
    - only dynamic databases expected to run non-parameterized operators repeatedly
    - the answer is often large (more a database transformation than query)

Similarity operators

- similarity join
  - joining descriptors of database A with descriptors of database B, based on a similarity-query predicate
    - range or kNN query predicate
  - if A=B, we talk about similarity self-join
  - self-joins are suitable for near-duplicates detection

Similarity operators

- *k* closest pairs
  - based on the distance function δ, select the *k* pairs $\langle x, y \rangle \in A \times B$, that have the smallest distance $\delta(x, y)$
  - repeated usage makes sense for different combinations of A and B, dynamic or streamed databases, where the closest pairs have to be continuously updated

Multi-modal search

- description of objects/queries by multiple modalities (models, examples, interfaces, etc.)
  - often reduced to multi-modal query-by-example
  - e.g., keywords+image, multiple images, image+audio+relational, etc.
- early fusion (in similarity search)
  - all modalities aggregated in single similarity model
  - single descriptor per object (1:1 problem, one query/index)
- late fusion
  - each modality represented (and queried, indexed) individually
  - hence, 1:N problem, where fusion step is needed

Early fusion

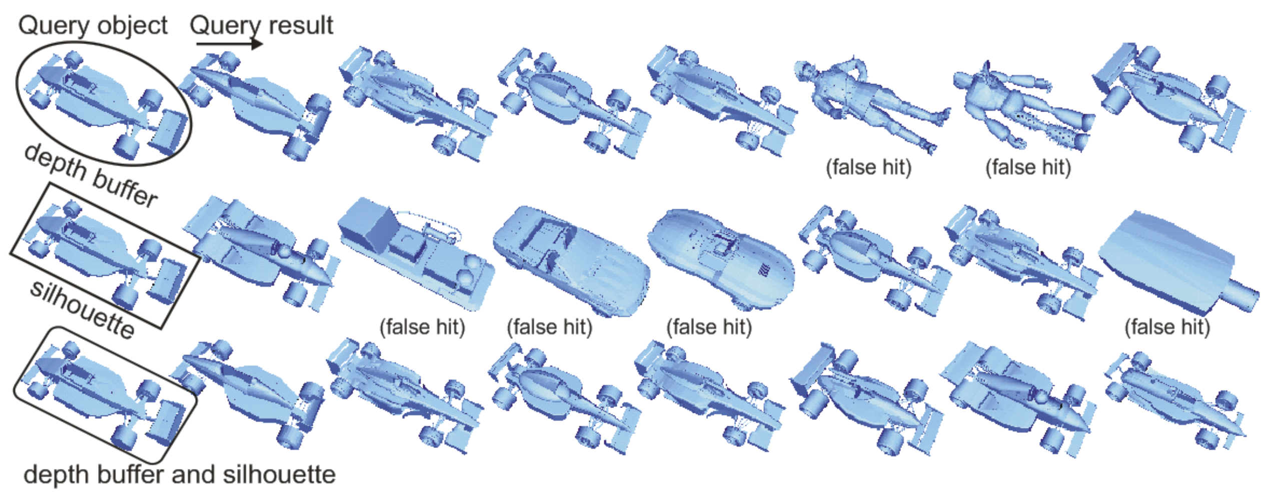
- partial similarity models aggregated
- e.g., multi-metric approach
  - provides flexible (parameterized) similarity in single model
  - descriptor composed from multiple sub-descriptors
  - each sub-descriptor its own similarity model (metric space)
  - for each query object different contribution of particular sub-descriptor needed
  - multi-metric
    - linear combination of distances on sub-descriptors
    - vector of weights W defined at query time

$$\mathbb{W} = \langle w_i \rangle$$

$$\Delta_{\mathbb{W}}(O_1, O_2) = \sum_{i=1}^{m} w_i \cdot \delta_i(O_1, O_2)$$

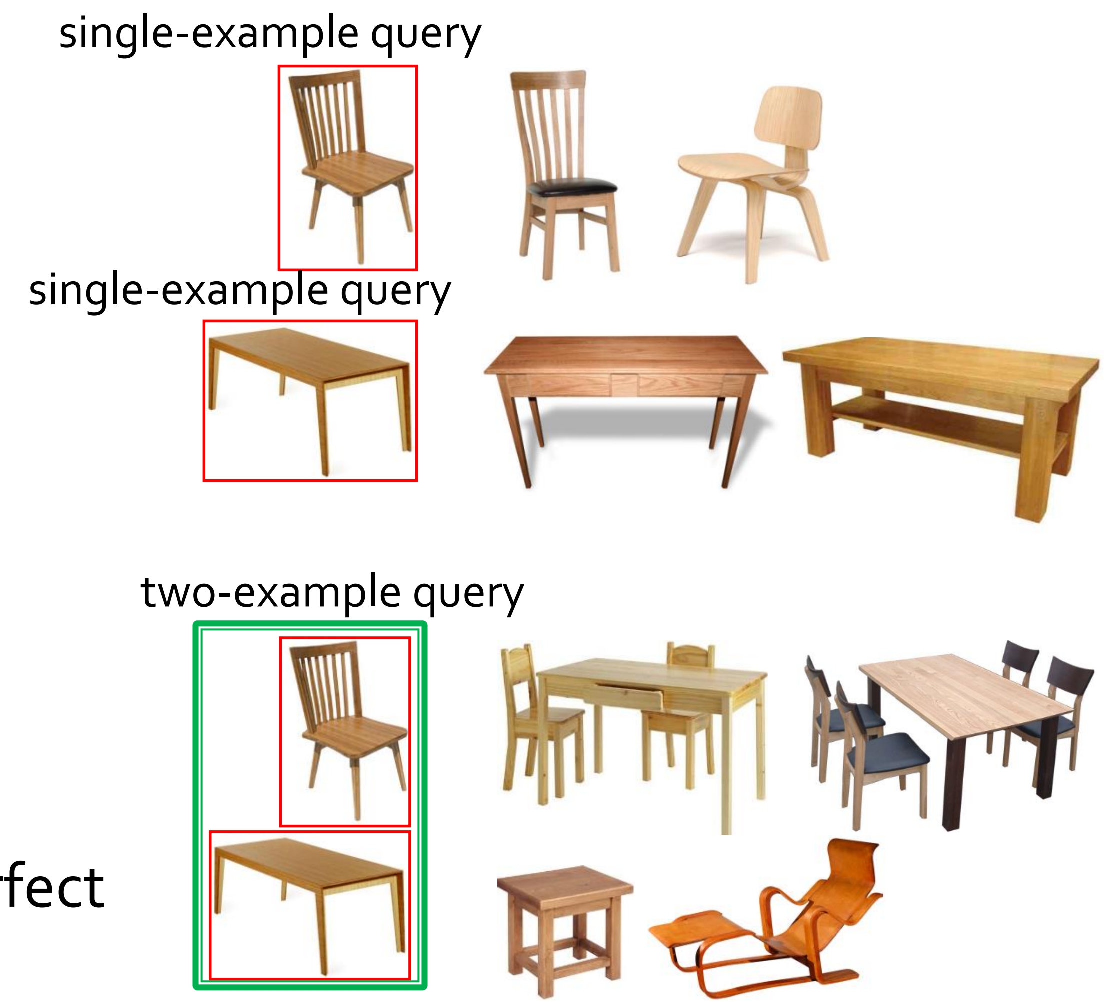
Early fusion – multi-metric model

- multi-metric model consisting of
  - depth buffer feature descriptor + distance δ_1
  - silhouette feature descriptor + distance δ_2
- single multi-metric index
- results for query model "formula"; W set at query time as
  - $w_1 = 1, w_2 = 0$
  - $w_1 = 0, w_2 = 1$
  - $w_1 = 0.5, w_2 = 0.5$



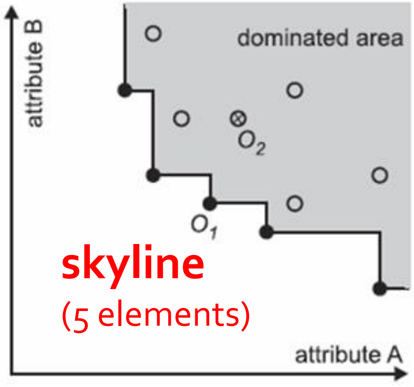
Late fusion – skyline operator

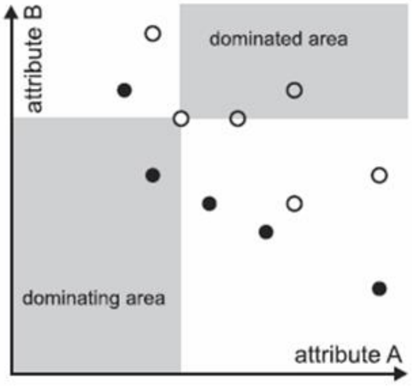
- single-example queries often not sufficient
  - user is not able to find perfect example
- multi-example queries
  - aggregating queries/operators
  - multiple not-so-perfect examples



Late fusion – skyline operator

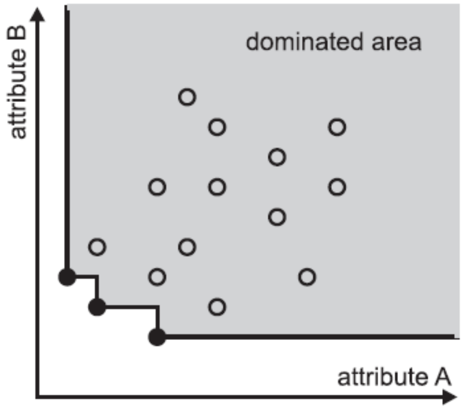
- traditional skyline operator (also Pareto set)
  - database **S** modeled in several ordered domains (attributes)
  - skyline = subset of elements from **S** that are not dominated by other elements
    - element is not dominated if there does not exist another element that is better in all the attributes (let's better = lower value)
  - why the term "skyline"?
    - when connected by vertical and horizontal lines, the set looks like skyline of a city
  - application
    - e.g., market basket,
      consider database of hotels
      with attributes
      price and
      distance to airport



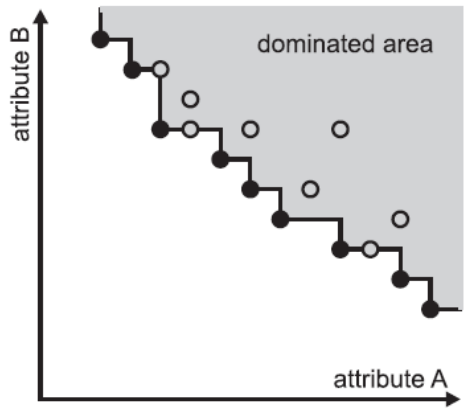


Late fusion – skyline operator

- problems
  - skyline is not limited in size
    - correlated data lead to very small skyline (a)
    - anti-correlated data lead to very large skyline (b)
  - there is not ranking/ordering inside
  - i.e., problems similar to the Boolean model
- often too large skyline
  - manufacturers/distributors create additional unique attributes to put their products on the skyline
    - e.g., my hotel is the closest one to a winery ☺



(a)



(b)

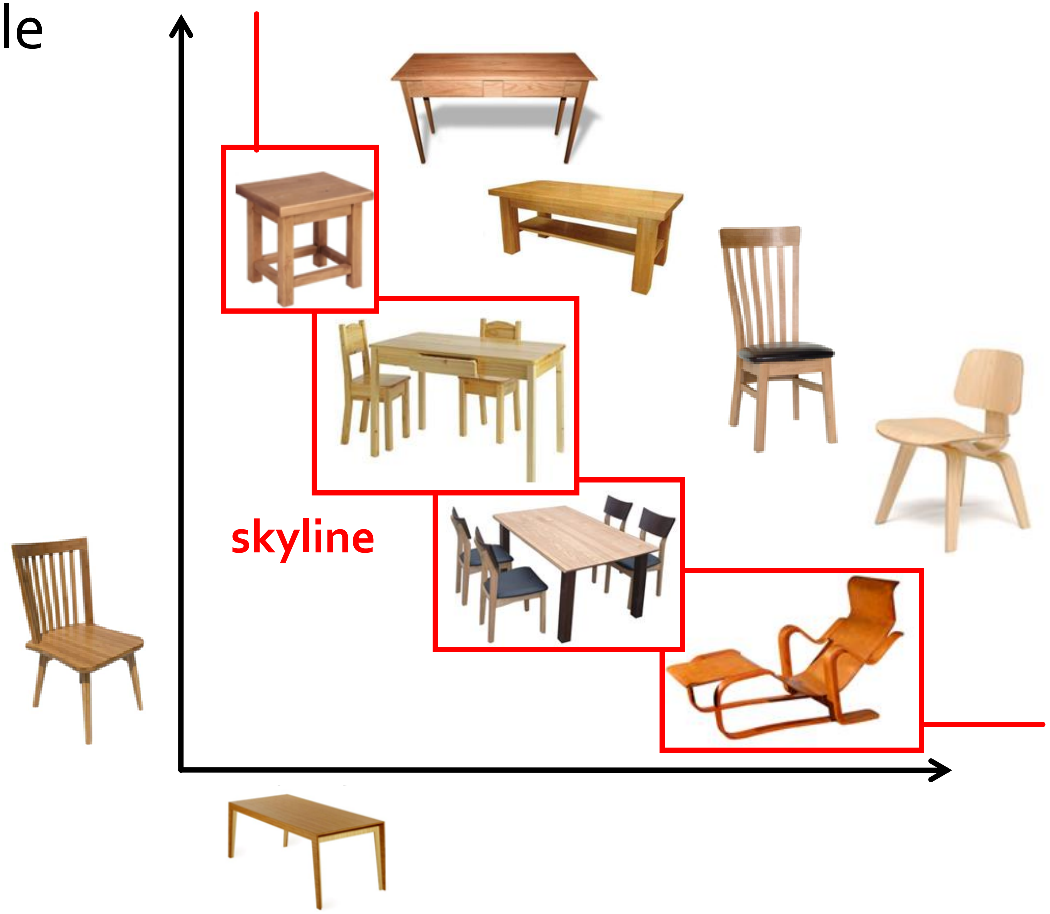
Late fusion – skyline operator

- let the “attributes” be interpreted as similarity orderings with respect to multiple query examples

- the skyline operator then becomes a **multi-example similarity query**
  - a particular skyline is a set of descriptors that are compromises with respect to the query examples
  - dynamic schema (attributes)
    - the coordinate system is established for each query separately
    - cannot be implemented by traditional skyline operator

Late fusion – skyline operator

- example of two-example similarity skyline query



Late fusion – top-k operator

- motivation
  - consider a single database of multimedia objects (or web pages)
  - consider several (similarity) models that can be used to rank the database – search result fusion
    - similarity query, the similarity ordering, respectively
    - other rankings, e.g., PageRank
  - for example, database of web pages including images
    - ranking #1 = vector model of inf. retrieval (cosine sim. of web page text)
    - ranking #2 = similarity of images of the web pages (e.g., MPEG7 and L1)
    - ranking #3 = the PageRank of the web pages
  - thus, we need an aggregation procedure to create one final ranking based on the partial rankings – the **top-k operator**

Late fusion – top-k operator

- top-k operator
  - real-valued ordered attributes $A_1,\ldots,A_m$
    - in our case, let the attributes be **different rankings of the same objects**
    - for now let's better value = higher value (but could be defined inversely)
  - aggregation function $f: A_1 \times \ldots \times A_m \rightarrow R$
    - let f be monotonic, i.e., if $x > y$, then $f(\ldots, x, \ldots) > f(\ldots, y, \ldots)$
    - e.g., Min, Max, Avg
  - the top-k operator evaluates the aggregation function on all the objects' partial ranks and returns *k* objects with the highest aggregate ranks
    - could be done sequentially, but how to do that efficiently?
    - Fagin and Threshold algorithm (already discussed in Lecture 5)

Late fusion – re-ranking

- multi-phase querying (dependent aggregation)
  - applying different similarity model on part of the previous ranking
  - when single similarity is too complex to model

- example
  - first ranking as 4NN – color layout
  - re-ranking as 3NN – shape

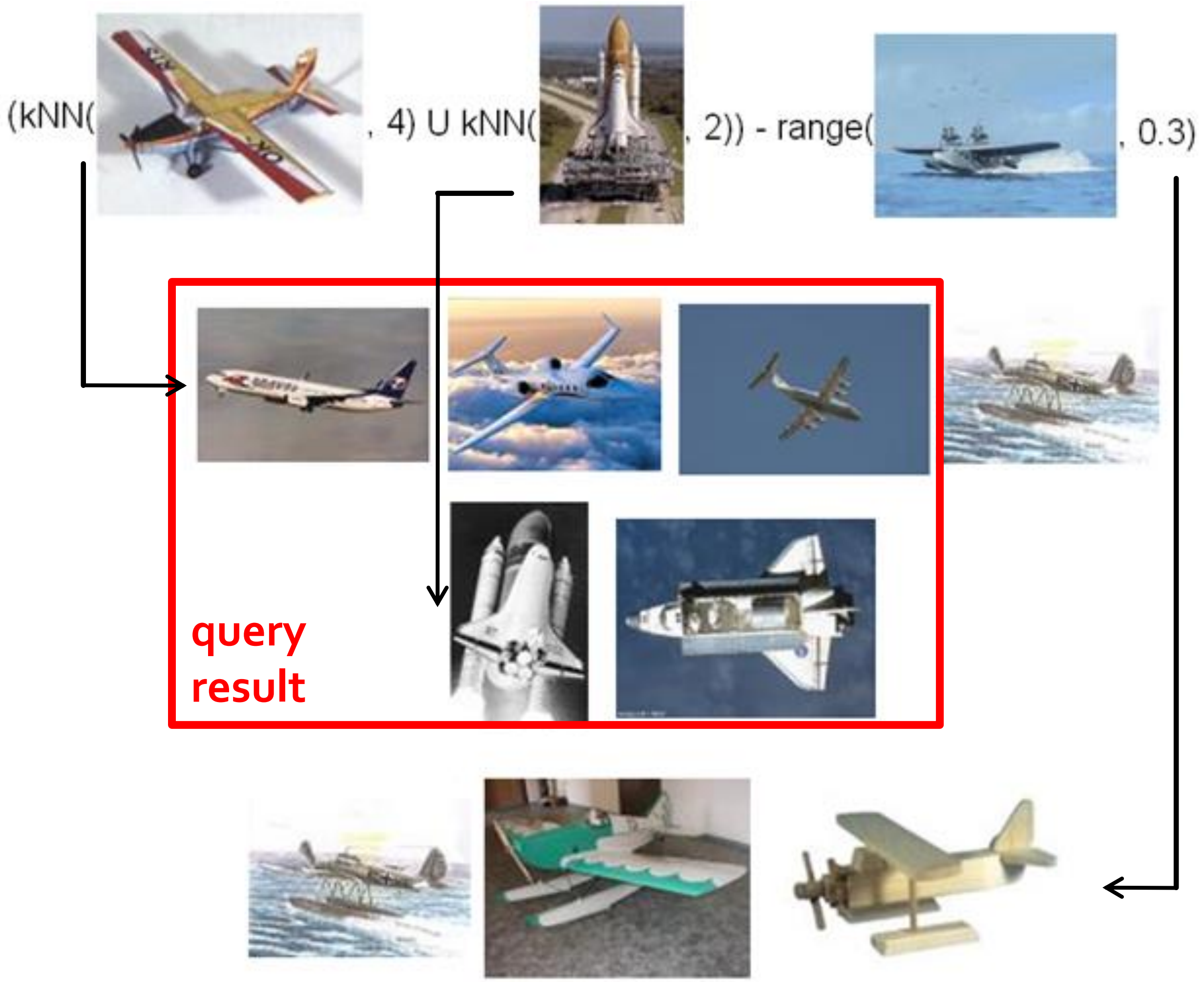
Late fusion – query languages

- similarity queries could serve as a basis for higher-level late-fusion query models

- query languages for similarity search
  - ad-hoc set-based expressions
  - extension of SQL
    - allows combination with structured attributes (relational modality)

Late fusion – query languages

- set operations with similarity queries



Late fusion – integration to SQL

- let a database of descriptors is stored in a BLOB-type attribute in a table of relational database
- new SQL predicates could enable relational databases to execute similarity queries
  - general SQL predicate
    - given an expression, the predicate condition is evaluated for all rows of a given table (or a join)
    - if the row passes the expression, the predicate is true
    - classic SQL predicate is, e.g., LIKE, ANY, IN, etc. in WHERE or HAVING
  - similarity predicates
    - range(example.MMattribute, table.MMattribute, r)
    - kNN(example.MMattribute, table.MMattribute, k)

Late fusion – integration to SQL

- simple queries

```
SELECT Id FROM BioData WHERE
    range(JohnSmith.Fingerprint, BioData.FingerPrint, 0.01)

SELECT Id FROM DNAData WHERE
    kNN(MickeyMouse.DNA, DNAData.DNA, 1)
```

Late fusion – integration to SQL

- join based on range query
  - range(A's descriptor, B's descriptor, query radius)
  - example in SQL

    SELECT Criminal.Id, Citizen.Id FROM Criminal
    SIMILARITY JOIN Citizen ON range(Criminal.FingerPrint,
    Citizen.FingerPrint, 0.01)
- join based on kNN query
  - kNN(A's descriptor, B's descriptor, k)
  - example in SQL

    SELECT Mammal.Id, Insect.Id FROM Mammal
    SIMILARITY JOIN Insect ON kNN(Mammal.DNA, Insect.DNA, 2)