
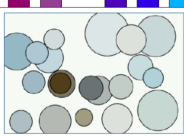
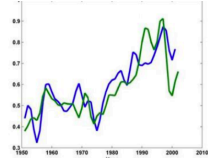


10. Vyhledávání v multimediálních databázích, podobnostní vyhledávání podle obsahu, podobnostní dotazování, agregační operátory, indexování metrické podobnosti, aproximativní vyhledávání.

- multimédia = jakýkoliv druh nestrukturovaného média (fotky, hudba, data ze senzorů...)
- kontext **médií** (context)
 - o popis, anotace, sousední prvky, klíčová slova, GPS souřadnice, kontext ze sítě (fb)
- obsah **médií** (context) (pixely, zvuková vlna...)
- **vyhledávání v multimédiích na základě textové anotace**
 - o příklad obrázku na sociální síti: vyhledávání na základě popisku, tagů, komentů, geolokaci, uživateli (a jeho kontextu), albu (sousední prvky), převažující barvy, EXIF
 - o **výhody:**
 - implementace stejná jako u textového vyhledávání (Boolean/vector model)
 - se vznikem sociálních sítí přichází široká informace o kontextu médií
 - o **nevýhody:**
 - vyžaduje manuální práci
 - subjektivní, nekompletní
- databáze dle struktury (struktura, sémantika = struktura dotazů)
 - o **relační** (pevná struktura, jasná sémantika – víme, co jaký sloupec znamená)
 - o **full-text db** (volná struktura, jasná sémantika – dotazujeme se obsahem)
 - o **multimediální databáze**
 - **volná struktura** (například pixely obrázků)
 - **volná sémantika** (problém s dotazováním => jak se ptát podle pixelů?)

Podobnostní vyhledávání podle obsahu, podobnostní dotazování

- multimédia mají variabilní strukturu a složitou sémantiku
- ➔ potřebujeme vyhledávací model založený na kontextu (zpracovávající raw obsah do zpracovatelné informace vyšší úrovně)
- model podobnostního vyhledávání řeší
 - o **extrakci příznaků z multimediálních dat** (feature extraction ➔ deskriptory)
 - multimédia ➔ deskriptory (tyto jsou ale na rozdíl od RDB skryté)
 - výsledkem extrakce se strukturovaná (relační) databáze
 - **δ: $U \times U \rightarrow R$**
 - způsoby získání deskriptorů
 - **analýza low-level features**
 - o computer vision apod. – nad rámec tohoto předmětu
 - **kombinace low-level features**
 - o k získání výrazného, ale výstižného popisu (wut)
 - o v kompetenci „data exploration“
 - MPEG7 standard definuje vizuální, audio a pohybové deskriptory
 - **forma deskriptorů**
 - **vector** (for feature histograms)
e.g.,
 $x = \langle 0.1, 0.4, 0.3, 0.4, \dots, 0.6, 0.7, 0.5, 0.3 \rangle$ 
 - **set** (for feature signatures)
e.g.,
 $x = \langle \langle c_{1f}, w_{1f} \rangle, \langle c_{2f}, w_{2f} \rangle, \dots, \langle c_{kf}, w_{kf} \rangle \rangle$ 
 - **ordered set** (for time series, sequences, strings)
e.g.,
 $x = \langle t_{1f}, t_{2f}, \dots, t_{kf} \rangle$ 
 - o **podobnostní funkci** (deskriptor \leq deskriptor)
 - slouží k porovnání deskriptorů

- **e: X → U**

- **výběr pravděpodobnostní funkce závisí na struktuře dat**

- **vektorové vzdálenosti** (pro obecné vektory, histogramy)

- Minkovského vzdálenosti (Manhattan, Eukleid...) **O(n)**

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

- kosínové vzdálenosti (záleží na úhlu) **O(n)**

$$\text{SIM}_{\cos}(v_1, v_2) = \frac{\sum_{i=1}^D v_1[i]v_2[i]}{\sqrt{\sum_{i=1}^D v_1[i]^2 \cdot \sum_{i=1}^D v_2[i]^2}}$$

- kvadratické vzdálenosti (histogramy) **O(n²)**

$$d_{QF}(v_1, v_2) = \sqrt{(v_1 - v_2)M(v_1 - v_2)^T}$$

- **adaptivní vzdálenosti** (podpisy, složité množiny)

- signature quadratic form distance

- earth mover's distance **O(2ⁿ)**

- vzdálenost mezi dvěma pravděpodobnostními distribucemi

- Hausdorff distance **O(n²)**

- vzdálenost mezi dvěma podmnožinami ve stejném prostoru

- **vzdálenosti sekvencí** (texty, časová data)

- editační vzdálenost

- dynamic time warping distance

- **dotazování**

- na rozdíl od SQL se nemůžeme dotazovat deskriptory

- můžeme používat pouze prvky modelu samotného (tzn. extrakci příznaků a podobnostní funkci)

- **dotazování se příkladem** (query by example concept)

- multimédium → deskriptor → užití podobnostní funkce pro porovnání se všemi deskriptory v databázi

- výsledek

- **výsledky z rozsahu ?(range query)**

- nalezené objekty nad nějakým thresholdem (práh)

$$(q, r) = \{x \in S \mid \delta(q, x) \leq r\}$$

- **k nejblížešších sousedů (kNN)**

$$(q, k) = \{C \mid C \subseteq S, |C|=k, \forall x \in C, y \in S-C \Rightarrow \delta(q, x) \leq \delta(q, y)\}$$

- příklady **extrakce příznaků** z multimédií

- z obrázků pomocí globálních vlastností

- **Scalable color** (MPEG7) – tvorba histogramu

- **Color structure** (MPEG7) – v podstatě se vytvoří histogram a podobně jako CNN se přesouvá přes obrázek, čímž poskytuje lokální informace o barvách

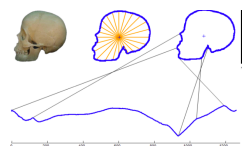
- z obrázků pomocí lokálních vlastností

- **SIFT / SURF** (hledá bloby a zajímavé body; nezáleží na otočení obrázku)

- z časových řad

- **Dynamic time warping distance (DTW) O(n²)**

- **Longest common subsequence (LCSS) O(n²)**



Indexování metrické podobnosti

- počítání podobnostní funkce prvků velmi drahé → pro velké databáze nepoužitelné

- používá se indexování pomocí metrických vzdáleností mezi některými prvky – tím se prostor rozpadne na několik definovaných částí, potom víme, ve které leží hledaný prvek; na konec se menší část projde lineárně

$$\delta(x, y) = 0 \quad \Leftrightarrow x = y \quad \text{reflexivity}$$

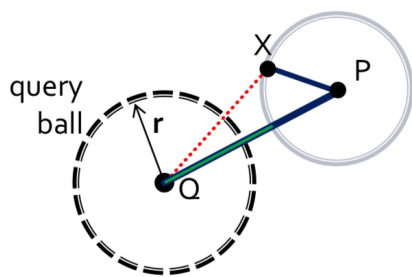
$$\delta(x, y) > 0 \quad \Leftrightarrow x \neq y \quad \text{non-negativity}$$

$$\delta(x, y) = \delta(y, x) \quad \text{symmetry}$$

- $\delta(x, y) + \delta(y, z) \geq \delta(x, z)$ triangle inequality

- rozdělením prostoru vznikne **metrický index**

- **lower-bound distances** mechanismus pro filtrování prvků v databázi (využívá trojúhelníkové nerovnosti)



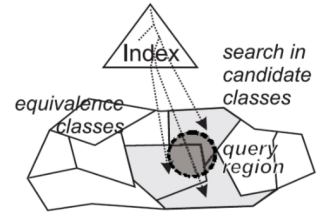
The task: check if **X** is inside query ball

• we know $\delta(Q, P)$

• we know $\delta(P, X)$

• we do not know $\delta(Q, X)$

• we do not have to compute $\delta(Q, X)$, because its lower bound $\delta(Q, P) - \delta(P, X)$ is larger than r , so **X** surely cannot be in the query ball, so **X** is ignored

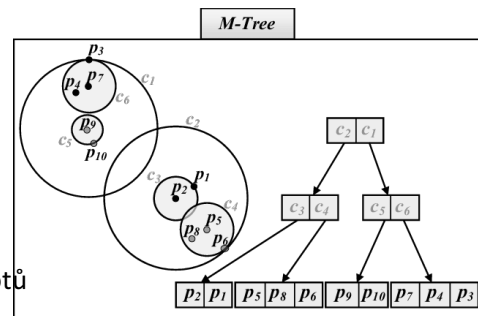


Metric access methods (MAMs)

- indexy pro efektivní vyhledávání v metrických prostorech
- index používá pouze vzdálenosti mezi prvky, o struktuře prostoru nemusí nic vědět
- třídy ekvivalence
- dotaz „spadne“ do nějaké přibližné oblasti a odpovídající třídy ekvivalence jsou pak prohledány a cheap determination of **lower-bound distance**
- $\delta_{LB}(*, *) \leq \delta(*, *)$
- architektury MAM indexů

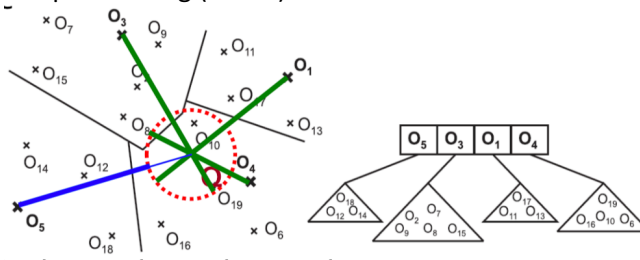
flat pivot tables (LAESA/AESA)

- každý pivot je vektor
- AESA
 - každý prvek je pivot
 - $O(|S|^2)$ matice
- LAESA (linear AESA)
 - pouze vektory vybraných pivotů



trees

- ball-partitioning (**M-tree**)
- hyperplane-partitioning (**GNAT**)



hashed indexes (**D-index**)

- prozradí, zda daný prvek leží v/vně nějaké kružnice
- kombinace předešlých
- v 10. přednášce je podstatně **více detailů** – ale vynechal jsem je, snad to takto bude k SZZ stačit

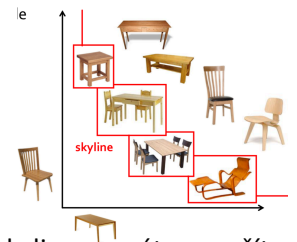
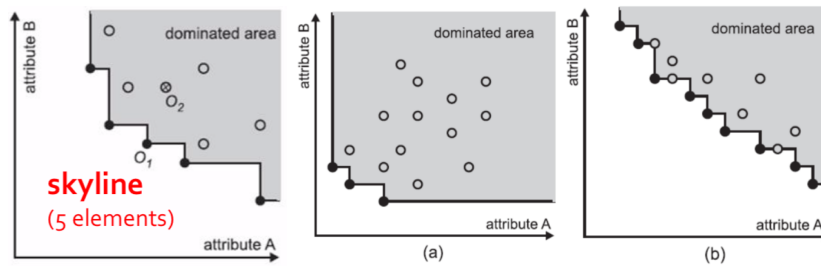
Agregační operátory

- databázové operátory
- **similarity join**
 - spojuje deskriptory db A s deskriptory db B na základě range nebo kNN predikátu
 - pokud $A=B \rightarrow$ self-join
 - užitečný například pro detekci „téměř totožných prvků“
 - join na základě **range query**: $range(A's\ descriptor, B's\ descriptor, query\ radius)$

```
SELECT Criminal.Id, Citizen.Id FROM Criminal
SIMILARITY JOIN Citizen ON range(Criminal.FingerPrint,
Citizen.FingerPrint, 0.01)
```
 - join na základě **kNN query**: $kNN(A's\ descriptor, B's\ descriptor, k)$

```
SELECT Mammal.Id, Insect.Id FROM Mammal
SIMILARITY JOIN Insect ON kNN(Mammal.DNA, Insect.DNA, 2)
```
- **Skyline operator**
 - dotazovat se jedním příkladem nemusí být vhodné – můžeme se chtít dotazovat vícero příklady
 - Jak ale najít nejlepší prvky, když máme více atributů (důležitost atributů)?
 - skyline = podmnožina elementů takových, které nejsou dominovány žádným jiným elementem

- element není dominován, pokud neexistuje žádný další element, který je lepší ve všech atributech



- a) korelovaná data → „malý“ skyline
 - b) nekorelovaná data → „velký“ skyline
- pokud se na atributy můžeme dívat jako na míry podobnosti, pak se dá skyline operátor použít jako **multiple-example similarity query**

- Top-k operátor

o myšlenka

- máme nějaké atributy (reálná čísla) – například hodnocení nějakého objektu
- máme agregační funkci (avg/min/max...)
- vybere podle každého atributu nejlepších k objektů (dle agregační fce)
- toto řešení je však výpočetně moc náročné**

o Fagin algorithm

- cílem je vrátit k nejlepších objektů (podle agregační funkce) na základě vícero atributů
- myšlenka
 - seřad' atributy
 - paralelně procházej jednotlivé atributy a nalezené hodnoty si zapisuj do tabulky T
 - jakmile najdeš k objektů, ke kterým sis již zapsal všechny hodnoty atributů, stůj
 - doplň chybějící atributy T (pomocí náhodného přístupu)
 - najdi minimum ze všech atributů pro každý nalezený objekt (průnik?)

STEP 1
loop(Read attributes from every sorted list)
stop when k objects have been seen in common from all lists (let's k = 2)

List ₁	List ₂
(a, 0.9)	(d, 0.9)
(b, 0.8)	(a, 0.85)
(c, 0.72)	(b, 0.7)
...	...
(d, 0.6)	(c, 0.2)

STEP 3
Compute the aggregate ranks of all objects already accessed
Return the k highest aggregate-ranked objects (in our example k = 2)

ID	A ₁	A ₂	Min(A ₁ , A ₂)
a	0.9	0.85	0.85
d	0.6	0.9	0.6
b	0.8	0.7	0.7
c	0.72	0.2	0.2

o Threshold algorithm

- u Faginu se některé atributy některých položek procházely úplně zbytečně
- místo toho zavedeme threshold hodnotu (v příkladu zde t = 0.7)
- procházíme paralelně listy atributů a pro každou nalezenou hodnotu rovnou nalezneme i zbývající hodnoty z okolních atributů; okamžitě spočítáme minimum z nalezených atributů a pokud je toto minimum větší než threshold, vracíme do výsledku
- při nalezení k objektů končíme

- Read k objects in parallel, such that all their ranks are determined immediately (either by parallel search or random access)
- Determine the aggregate ranks for the read objects
- Keep in buffer the top-k candidate objects (and their ranks)

List ₁	List ₂
(a, 0.9)	(d, 0.9)
(b, 0.8)	(a, 0.85)
(c, 0.72)	(b, 0.7)
...	...
(d, 0.6)	(c, 0.2)

ID	A ₁	A ₂	Min(A ₁ , A ₂)
a	0.9	0.85	0.85
d	0.6	0.9	0.6
b	0.8	0.7	0.7

- oba dva uvedené algoritmy jsou korektní (díky monotonii agregační funkce)
- Threshold algorithm čte méně objektů, dělá však více random accessů do paměti, omezená paměť
- Fagin algoritmus nemá omezené paměťové požadavky, běží v průměru déle než Threshold

Aproximativní vyhledávání

- přesné vyhledávání může být výpočetně velmi náročné → oželíme trochu přesnosti na úkor velkého zrychlení
- lidé neumí formulovat přesné dotazy → ve výsledky tak přesný výsledek nemusí být očekávaný
- indikátor indexovatelnosti dat (strukturovatelnosti) – **intrinsic dimensionality**

intrinsic dimensionality $p(S, d) = \mu^2 / 2\sigma^2$

(μ is **mean** and σ^2 is **variance** of distance distribution in S)

low p (e.g., below 10) means the dataset is **well-structured**

– i.e. there exist tight clusters of objects

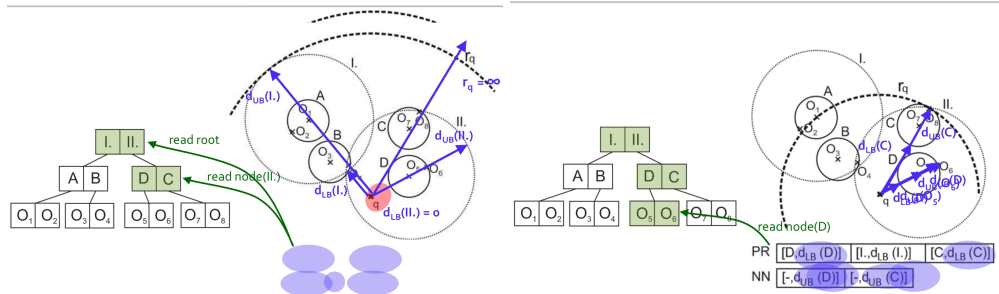
high p means the dataset is **poorly structured** – i.e. objects are

- almost equally distant

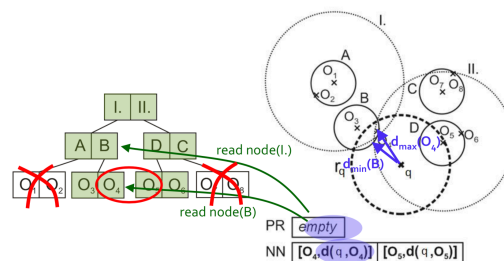
M-tree

o přesné kNN vyhledávání

- query ve formátu (q, r_q) (query, query radius – dynamický)
- základem je „NN“ pole kNN kandidátů a prioritní fronta „PR“, která udržuje nody, které nemohou být vyřazeny z vyhledávání
- myšlenka
 - na začátku je query radius nekonečný, později se snižuje
 - o PR obsahuje pouze M-tree kořen
 - o NN pole je prázdné
 - procházíme M-tree a vkládáme do PR vrcholy, které mají být zpracovány (například když query kružnice se překrývá s některou z kružnic kolem vrcholů stromu)
 - v rámci kNN vybíráme vrcholy z PR a aktualizujeme pole sousedů NN
 - snižujeme radius
 - **jakmile je PR prázdná, došli jsme k výsledku**



•



•

o aproximativní kNN vyhledávání

- různé druhy heuristik
 - **slowdown of improvements**
 - o pokud se pole NN mění pouze málo, zastavíme
 - o měříme derivace změn a pokud dosáhneme nějaké uživatelem nastavené hodnoty, končíme
 - **approximately correct search**
 - o nastavíme nějaký threshold ϵ a do PR potom přidáváme prvky o radiusu $r = r / (1 + \epsilon)$
 - o rychleji tak zastavíme a výsledné sousedy může být až $(1 + \epsilon)$ krát vzdálenější než reálný
 - **good fraction approximation**
 - o řídíme se vzdáleností sousedů
 - o pokud je vzdálenost větší než nějaký threshold, zastavíme