Porovnání některých metod pro vyhledávání a indexování multimediálních dat

Michal Krátký, Tomáš Skopal, Václav Snášel*

Abstrakt

Předmětem tohoto článku je diskuze dvou přístupů vyhledávání a indexování multimediálních dat. Obě tyto metody se opírají o již existující matematický aparát – v jednom případě se využívá vlastností vektorových prostorů a ve druhém případě vlastností metrických prostorů. Abstraktní datová struktura UB-strom je potom realizací prvního přístupu, M-strom využívá přístup druhý.

1 Úvod

Vyhledáváním a indexováním dat se zabývají *Dokumentografické informační systémy (DIS)*[1]. Tyto systémy umožňují organizovat velkou skupinu dokumentů tak, aby v ní bylo možno efektivně vyhledávat. Význam onoho pojmu *dokument* specializuje jednotlivé typy DIS pro specifické třídy dat. S klasickými textovými dokumenty (např. HTML stránkami) pracují zejména fulltextové systémy či faktografické systémy.

Pojem *multimediální dokument* chápeme obecně jako blok dat s nějakou vnitřní strukturou a sémantikou. Typickými představiteli mohou být vedle již zmíněných textů také obrázky, zvuky, animace, atd ...

Indexování a vyhledávání dat přináší základní charakteristiku systémů DIS a tou je *neurčitost*. Výsledky dotazů nevracejí exaktní odpovědi, ale odpovědi více či méně *relevantní*. V této souvislosti se často hovoří o významové *podobnosti* dokumentu-dotazu, či *vzdálenosti* dokumentu-dotazu. Zde bychom odkázali na literaturu [PSH98, PAT99].

*VŠB-Technická Univerzita Ostrava, Katedra informatiky

[1] angl. Information Storage and Retrieval (ISR)

1.1 Extrakce vlastností

Organizování/indexování dokumentu v DISu probíhá podle určitých vlastností dokumentu. Nalezení takové skupiny vlastností pro daný typ dokumentu se nazývá **extrakce vlastností**. U rastrového obrázku velikosti $x \times y$ si můžeme představit jako jednoduchou skupinu vlastností *velikost*, všechny obrázky o stejné velikosti pak z tohoto pohledu splývají. Opačným extrémem je extrahování barev *všech bodů* rastru jako $x \times y$ vlastností (vlastně celý obrázek). Oba uvedené extrémy jsou pro praxi nevhodné. Použitelnou vlastností obrázku je histogram jeho barev, což je posloupnost čísel, ve které každé číslo reprezentuje počet pixelů stejné barvy v obrázku. Například dvoubarevný obrázek velikosti 16×16 s modrým pozadím a jedním červeným pixelem uprostřed může mít histogram tvaru *(255,1)*.

1.2 Vektorové prostory

Vektorový prostor $\Omega = D_1 \times D_2 \times \ldots \times D_n$ má dimenzi n. Objekt (resp. vektor, resp. bod) $\mathcal{O} = [a_1, a_2, \ldots, a_n]$ patřící do vektorového prostoru je jednoznačně určen svými souřadnicemi $a_i \in D_i, 1 \leq i \leq n$, kterých je právě n. Každá jednotlivá dimenze má svoji doménu D_i – množinu hodnot (resp. vlastností), kterých může příslušná vektorová souřadnice nabývat.

Příklad: Zmiňovaný obrázek (resp. jeho histogram) je bod ve vektorovém prostoru na souřadnicích [255,1]. Jiným příkladem může být prostor klíčových slov *"kočka"*, *"myš"*, *"pes"*, kde každé slovo postupně představuje jeden rozměr s doménou $\{ne, ano\}$. Dokument *"Kočka honí myš"* bude v tomto prostoru bod na souřadnicích $[ano, ano, ne]$.

Věta: Pro konečný vektorový prostor (takový, že domény jsou konečné množiny) existuje (mnoho) bijektivní(ch) zobrazení $\phi : \Omega \rightarrow C$, kde $C \subset N$ (N – přirozená čísla)

Jinými slovy – prostor Ω lze úplně vyplnit křivkou (křivkami). Potom číslo $p \in C$ je parametr křivky (nebo také adresa na křivce). Důsledkem je fakt, že vektorový prostor můžeme **lineárně uspořádat**. Především této vlastnosti pak využívá struktura UB-strom.

Poznámka: *Vektorový prostor vykazuje "globálně-topologické" charakteristiky tím, že objekty jsou vztaženy k souřadnicovému systému.*

1.3 Metrické prostory

Metrický prostor je definován metrikou. Metrika je funkce měřící vzdálenost[2] mezi dvěma objekty. Metrický prostor obecně nemá definovanou dimenzi a tudíž v něm nemusíme nutně pracovat s vektorovým počtem.

Objekt v_i patřící do metrického prostoru musí splňovat vůči ostatním objektům v metrickém prostoru následující axiomy metriky d:

$$d(v_x, v_x) = 0 \tag{1}$$

$$d(v_x, v_y) > 0 \qquad (v_x \neq v_y) \tag{2}$$

$$d(v_x, v_y) = d(v_y, v_x) \tag{3}$$

$$d(v_x, v_z) + d(v_z, v_y) \geq d(v_x, v_y) \tag{4}$$

Příklad: Asi nejznámější je Minskowského metrika pro metrické vektorové prostory[3]

$$L_p(v_x, v_y) = (\sum_{j=1}^{n} |v_x[j] - v_y[j]|^p)^{\frac{1}{p}} \qquad (p \geq 1) \tag{5}$$

Volbou $p = 2$ dostaneme Euklidovu (L_2) metriku – tj. měření vzdáleností v běžném životě.

Poznámka: *Metrický prostor vykazuje "lokálně-topologické" charakteristiky tím, že objekty jsou vztaženy vzájemně jeden ke druhému.*

2 Univerzální B-stromy (UB-stromy)

Univerzální B-strom – UB-strom – je jedna z přístupových metod SAM[4] pro indexování vícerozměrných dat ve vektorových prostorech. Tuto přístupovou metodu navrhl Rudolf Bayer (autor B-stromu) a poprvé ji zveřejnil v roce 1996 v [BAY96]. Další poměrně rozsáhlou publikací o UB-stromech a indexování vícerozměrných dat obecně je disertační práce jeho doktoranda Volkera Markla [MAR99].

Myšlenka UB-stromu je založena na *Z-uspořádání* a *B^+-stromech*. *B^+-strom* je vyvážený, a poskytuje tak logaritmické časové složitosti pro základní operace a minimální režii při vkládání a rušení indexovaných objektů. Jestliže budeme chtít použít tak efektivní datovou strukturu jakou je B^+-strom, je nutné, aby na zaindexovaných hodnotách atributů (vlastností) bylo zavedeno uspořádání. Jak ovšem uspořádat body v n-rozměrném prostoru? Odpověď najdeme ve větě z kapitoly 1.2. Konkrétně pro UB-stromy pak definujeme tzv. *Z-křivku*, která je konstruována pomocí *Z-uspořádání*, resp. *Z-adres*.

[2] pojem *vzdálenost* je zde velmi relativní a můžeme ho chápat v nepřímé úměře s pojmem *podobnost*
[3] vektorový prostor, ve kterém platí navíc i axiomy metriky
[4] Spatial Access Methods

2.1 Z-adresa

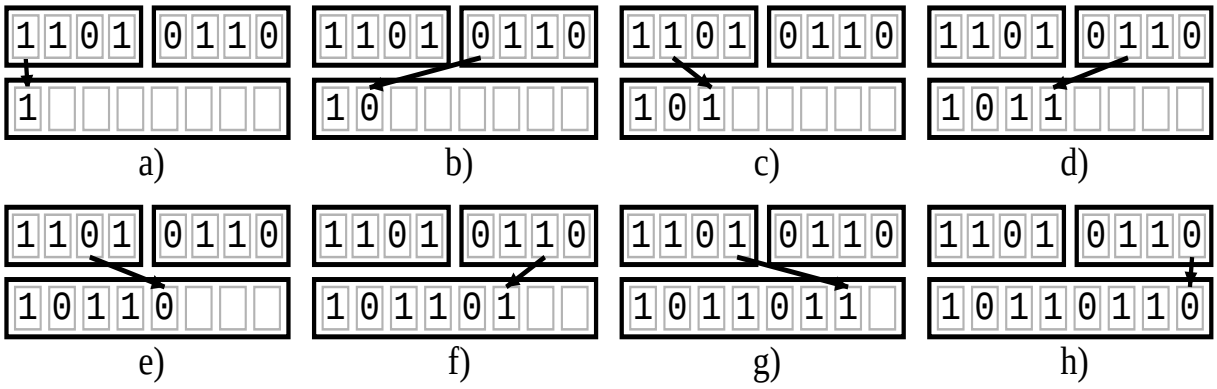
Definice: Z-adresa
Pro objekt $\mathcal{O} \in \Omega$ a binární reprezentaci hodnoty každého jeho atributu $A_i = A_{i,s-1}A_{i,s-2}\ldots A_{i,0}$ definujeme bijektivní funkci $Z(\mathcal{O})$ takto:

$$Z(\mathcal{O}) = \sum_{j=0}^{s-1}\sum_{i=1}^{n} A_{i,j}2^{jn+i-1}$$

Tato funkce spočítá k danému bodu v prostoru jeho Z-adresu. Pro výpočet Z-adresy existuje *algoritmus bitového prokládání* s lineární časovou složitostí. Existence takového algoritmu je velmi důležitá, protože výpočet Z-adresy je nejčastěji používaná operace při indexování vícerozměrných dat pomocí UB-stromu.

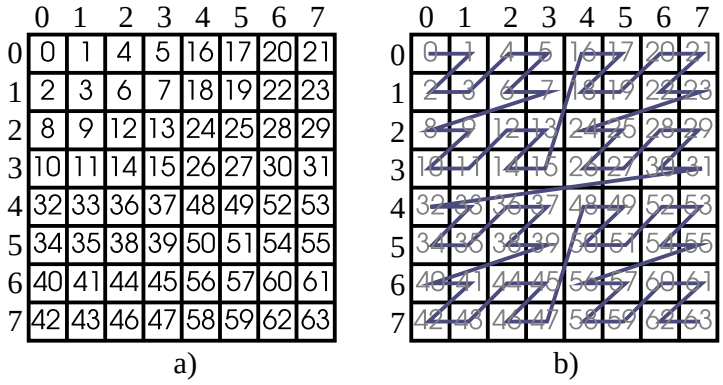
Příklad výpočtu Z-adresy:

Na obrázku 1 vidíme výpočet Z-adresy dle algoritmu bitového prokládání pro objekt s hodnotami atributů [6,13], který můžeme chápat jako bod prostoru. Pokud převedeme čísla 6 a 13 do dvojkové soustavy, získáme binární čísla 0110 resp. 1101. Získáme tedy hodnoty atributů jako 4-místné bitové řetězce. Maximální počet hodnot, které lze potom vyjádřit pomocí 4-bitového binárního čísla je 16. Z tohoto důvodu je doména obou atributů množina celých čísel {0,1, ..., 14,15}. A jelikož má objekt dva atributy, můžeme si jej představit jako bod ve 2-rozměrném prostoru 16×16. Výsledná Z-adresa je potom 10110110, čemuž odpovídá desítkové číslo 182.



Obrázek 1: Výpočet Z-adresy dle algoritmu bitového prokládání objektu s hodnotami atributů $[6, 13]$ ve 2-rozměrném prostoru 16×16.

Pro každý prvek prostoru Ω můžeme spočítat různou Z-adresu. Na obrázku 2.1a) vidíme Z-adresy pro všechny prvky 2-rozměrného prostoru 8×8. Z-adresa nám tedy zobrazuje $n-$rozměrný prostor do rozměru jediného a definuje uspořádání na množině všech bodů prostoru Ω.



Obrázek 2: a) Z-adresy pro každý prvek 2-rozměrného prostoru 8×8. b) Z-křivka pokrývající celý 2-rozměrný prostor 8×8.

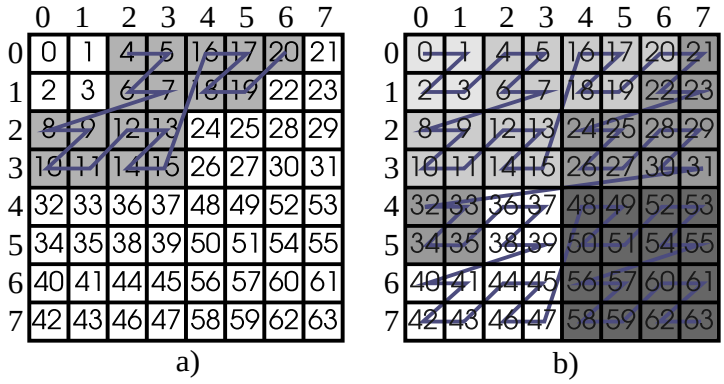
Musíme si uvědomit, že z definice Z-adresy plyne následující:

- délka bitových řetězců hodnot atributů musí být stejná
- tzn. prostor Ω je konečný a jeho velikost se nemění
- to že jsou hodnoty atributů vyjádřeny jako bitové řetězce znamená, že tyto hodnoty mohou být pouze celá čísla (je možné si ovšem představit, jak konečný interval reálných čísel vyjádříme pomocí čísel celých – ovšem s určitou nepřesností)
- pokud vlastnosti/atributy objektů nemají číselnou povahu, pro zaindexování do UB-stromu je potřeba převést hodnoty těchto atributů na celá čísla
- délka bitového řetězce Z-adresy je rovna $n\times$délce bitového řetězce hodnot atributů

Jestliže spočítáme Z-adresy pro všechny prvky prostoru Ω získáme *Z-křivku*, která pokrývá celý prostor. Na obrázku 2.1b) vidíme Z-křivku pokrývající celý 2-rozměrný prostor 8×8.

2.2 Z-region

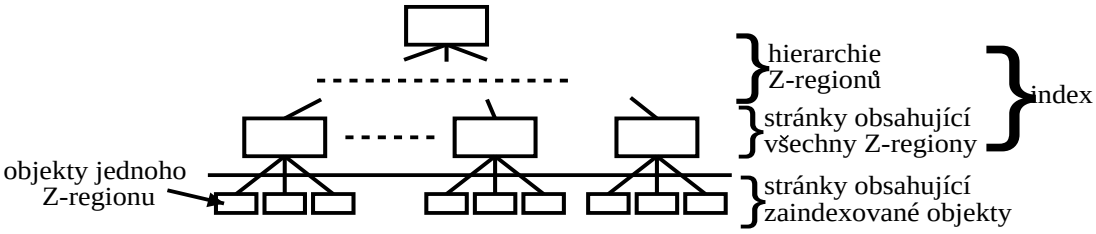
Dále si musíme uvědomit, že pokud bychom zaindexovali Z-adresy objektů přímo do B^*-stromu, resp. B^+-stromu, znamenalo by to např. neefektivní rozsahové dotazy. Z tohoto důvodu jdou UB-stromy ještě dál a zavádí tzv. *Z-regiony* pro shlukování blízko ležících bodů na diskové stránky. Účelem tohoto shlukování je minimalizovat počet přístupů na disk při operacích prováděných nad těmito daty, zejména při rozsahových dotazech.



Obrázek 3: a) Z-region [4:20] ve 2-rozměrném prostoru 8×8. b) Dělení 2-rozměrného prostoru 8×8 definované Z-regiony [0:3],[4:20],[21:35],[36:47],[48,63].

Z-region [α:β] je definovaný jako prostor pokrytý intervalem $< \alpha, \beta >$ na Z-křivce. Na obrázku 3a) je prezentován Z-region [4:20] ve 2-rozměrném prostoru 8×8. Celý prostor lze potom rozdělit do Z-regionů (příklad na obrázku 3b), které jsou uspořádány (protože se nepřekrývají) podle hodnot α a β. Každý z těchto Z-regionů bude obsahovat ty body, které budou uloženy v jedné stránce B^+-stromu, a ta je fyzicky uložena na jedné nebo více diskových stránkách. Tak je zajištěn minimální počet přístupů na disk při získávání bodů, které v prostoru leží blízko u sebe.

Jak již bylo řečeno, UB-strom využívá k uložení dat B^+-strom. Na obrázku 4 vidíme jakým způsobem jsou v UB-stromu uložena data a Z-regiony.



Obrázek 4: Uložení zaindexovaných objektů a Z-regionů v UB-stromu.

2.3 Algoritmy

2.3.1 Nalezení bodu (bodový dotaz)

Vstup: $\mathcal{O}_I$ – vzorový objekt, zadaný indexovanými atributy
Výstup: $\mathcal{O}_O$ – vrácený objekt se všemi atributy

$\xi = Z(\mathcal{O}_I)$
vyhledat v UB-stromu Z-region $[\alpha{:}\beta]$ takový, že $\alpha \leq \xi \leq \beta$
nahrát stránku $(\alpha{:}\beta)$ do paměti
prohledat stránku $(\alpha{:}\beta)$ zda neobsahuje $\mathcal{O}_I$

2.3.2 Vložení bodu

Vstup: $\mathcal{O}_I$ – objekt, který má být vložen do UB-stromu
Výstup: žádný

$\xi = Z(\mathcal{O}_I)$
vyhledat v UB-stromu Z-region $[\alpha{:}\beta]$ takový, že $\alpha \leq \xi \leq \beta$
nahrát stránku $(\alpha{:}\beta)$ do paměti
vložit $\mathcal{O}_I$ do stránky $(\alpha{:}\beta)$
if (count$(\alpha{:}\beta) >$ C) {
  vybrat $\gamma \in [\alpha{:}\beta]$ takové, že $\frac{1}{2}C - \epsilon \leq \text{count}(\alpha{:}\gamma) \leq \frac{1}{2}C + \epsilon$
  rozdělit stránku $(\alpha{:}\beta)$ do stránek $(\alpha{:}\gamma)$ a $(\gamma + 1{:}\beta)$
}

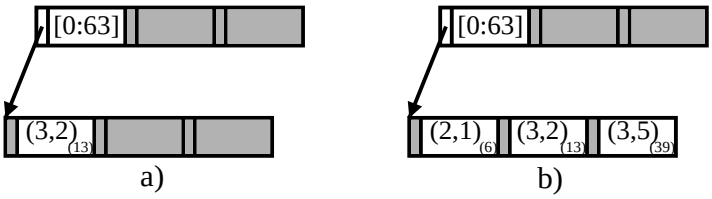
C je kapacita stránky UB-stromu a značí jakou *aritu* má použitý strom – jinak řečeno na maximálně kolik dalších stránek se může z jedné stránky odkazovat. Pokud má dojít k roštěpení stránky (operace SPLIT), pak se může stát (pokud bude překročena kapacita jednotlivých stránek), že dojde ke změnám od listu až ke kořeni stromu.

Příklad na vkládání do UB-stromu

Mějme 2-rozměrný prostor Ω rozměru 8×8. Na obrázku 5a) do 3-árního UB-stromu přidáme bod [3,2] jehož Z-adresa je 13. Před i po vložení tohoto bodu existuje v tomto 2-rozměrném prostoru jediný Z-region [0:63], který pokrývá celý prostor.

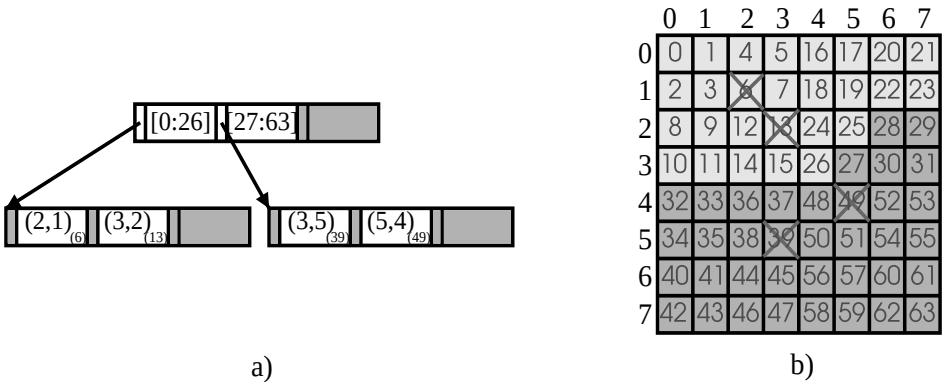
Na obrázku 5b) vidíme UB-strom po vložení bodů [2,1] a [3,5] se Z-adresami 6, resp. 39. Jelikož nebyla překročena kapacita stránky (která je v tomto případě 3), nedošlo ke štěpení žádné stránky.

Na obrázku 6 vidíme štěpení stránky a Z-regionu [0:63] po vložení bodu [5,4] se Z-adresou 49. V UB-stromu dochází při větším počtu indexovaných objektů k hierarchizaci Z-regionů,



Obrázek 5: a) UB-strom po vložení bodu [3,2]. b) UB-strom po vložení dalších bodů [2,1] a [3,5].

ale takovým způsobem, že na předposlední úrovni (v listech jsou objekty obsažené v Z-regionech) jsou stránky obsahující všechny Z-regiony. Na vyšších úrovních stromu jsou potom jejich *nad-Z-regiony* (viz. obrázek 4) tak, aby bylo možné algoritmem s logaritmickou časovou složitostí získat stránku se zaindexovanými objekty v Z-regionu.



Obrázek 6: a) Štěpení stránky a Z-regionu [0:63] po vložení bodu [5,4]. b) Prostor se dvěma nově vzniklými Z-regiony [0:26] a [27:63] po vložení tohoto bodu.

2.3.3 Vymazání bodu

Vstup: $\mathcal{O}_I$ – bod, který má být vymazán z UB-stromu
Výstup: žádný

$\xi = Z(\mathcal{O}_I)$
vyhledat v UB-stromu Z-region[α:β] takový, že $\alpha \leq \xi \leq \beta$
nahrát stránku (α:β) do paměti
smazat $\mathcal{O}_I$ ze stránky (α:β)
if (count(α:β) $< \frac{1}{2}C - \epsilon$)
{
  sloučit stránku (α:β) se sousední stránkou ($\beta + 1$:γ) do stránky (α:γ)
  if (count(α:γ) > C) {
    vybrat $\delta \in$[α:γ] takové, že count(α:δ) $\leq \frac{1}{2}C - \epsilon$
      a count($\delta + 1$:γ) $\leq \frac{1}{2}C + \epsilon$
    rozdělit stránku (α:γ) do stránek (α:δ) a ($\delta + 1$:γ)
  }
}

2.3.4 Dotaz na objekty v n-rozměrném kvádru (rozsahový dotaz)

Pomocí SQL bychom takový dotaz mohli vyjádřit takto:

SELECT * FROM <table>
  WHERE (A_1 BETWEEN a_1 AND b_1) AND
        (A_2 BETWEEN a_2 AND b_2) AND
        ...
        (A_n BETWEEN a_n AND b_n)

Vstup: y, z – body definující dotazovací okno[5], $Z(y) < Z(z)$
Výstup: X – výsledná množina objektů

$\xi = Z(y); \omega = Z(z); X = \emptyset$
do
{
  vyhledat v UB-stromu Z-region $[\alpha : \beta]$ takový, že $\alpha \leq \xi \leq \beta$
  $X = X \bigcup \{(x_1, \ldots, x_n) \in [\alpha : \beta] | (x_1, \ldots, x_n) \in [[y, z]]\}$
  ξ = Z-adresa prvního bodu dotazovacího okna taková, že $\xi > \beta$
} while ($\xi < \omega$)

Rozsahový dotaz vypadá tak, že se dotazovací okno (n-rozměrný kvádr) proloží do

[5] lépe řečeno *hyper-kvádr*, neboť se pohybujeme ve vícerozměrném prostoru

prostoru Ω. Dotazovací okno protíná několik Z-regionů a v těch se pak hledají ty body, které skutečně patří do zadaného dotazovací okna. Z toho plyne, že časová složitost rozsahového dotazu je počet regionů protínající dotazovací okno $\times$ časová složitost bodového dotazu.

2.3.5 Složitosti algoritmů

Pro bodový dotaz, vložení a zrušení bodu poskytuje UB-strom časovou složitost $O(\log_k N)$, kde N je počet zaindexovaných bodů, $k = \frac{1}{2}C$, C je kapacita stránky (tedy arita stromu). Pro rozsahový dotaz UB-strom poskytuje časovou složitost $r \times O(\log_k N)$, kde r je počet regionů protínající dotazovací okno.

2.4 Shrnutí

UB-strom poskytuje několik výhod. Velmi podstatné je to, že je vyvážený a tím dosahuje logaritmické časové složitosti pro základní operace. Dále řeší shlukování dat do Z-regionů a tak i na diskové stránky, čímž se snižuje počet přístupů na disk. Ovšem jen experimenty mohou prokázat nakolik je tato SAM přístupová metoda životaschopná i v praxi.

3 M-stromy

Datová struktura M-strom využívá pro indexování objektů vlastností metrických prostorů. Vnitřní strukturou je to opět perzistentní stránkovaný strom, jako je např. i UB-strom. Rozdíl spočívá v obsahu uzlů a listů stromu. Samotné objekty (*ground objects*) jsou uloženy v listech stromu, nelistové uzly obsahují tzv. *routing objects* (směrovací objekty).

Záznam směrovacího objektu O_r v uzlu obsahuje:

1. samotný objekt O_r[6]
2. odkaz $ptr(T(O_r))$ na svůj podstrom $T(O_r)$ – tzv. *covering tree* (pokrývající strom)
3. hodnotu $r(O_r)$ – tzv. *covering radius* (pokrývací poloměr)
4. hodnotu $d(O_r, P(O_r))$ – vzdálenost od rodičovského (parent) směrovacího objektu $P(O_r)$

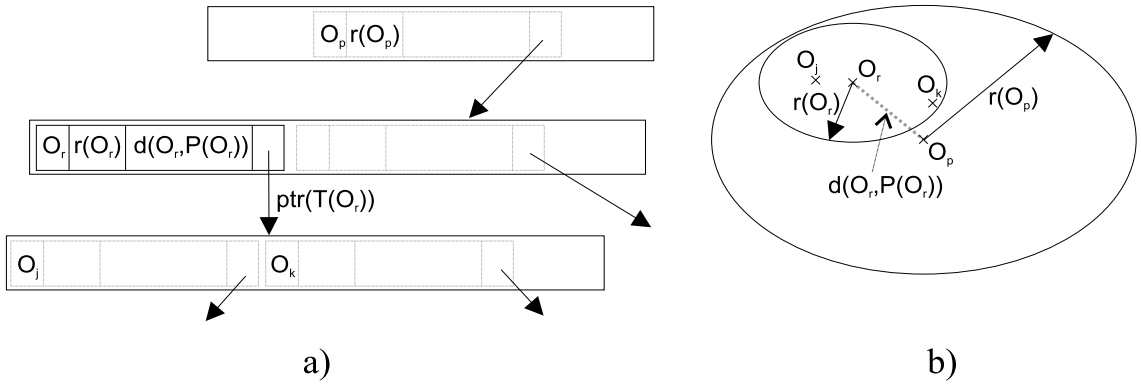
Záznam *ground* objektu vypadá podobně, ovšem místo odkazu na pokrývající strom a poloměru obsahuje $oid(O_j)$ – identifikátor celého objektu (dokumentu) uloženého mimo M-strom v datovém souboru.

[6] *virtuální* – vypočítaný pro účely M-stromu, nebo skutečný – obsažený v některém listu $T(O_r)$

Princip hierarchie M-stromu spočívá stejně jako u UB-stromu v rozdělení prostoru do regionů (ty ale nemusejí být nutně disjunktní jako je tomu u UB-stromu). K tomu slouží směrovací objekty a to tak, že všechny *ground* objekty, které obsahuje pokrývající strom $T(O_r)$ směrovacího objektu O_r jsou ve vzdálenosti maximálně $r(O_r)$ od O_r.

Formálně: $\forall O_i \in T(O_r), \qquad d(O_r, O_i) \leq r(O_r)$

Předpočítaná hodnota vzdálenosti k rodičovskému objektu $d(O_r, P(O_r))$ spolu s pokrývacím poloměrem $r(O_r)$ umožňují při vyhledávání v M-stromu eliminovat *nedotčené* regiony (viz. kapitola 3.1). Na obrázku 7 je schématicky znázorněna struktura stromu a vztahy mezi směrovacími objekty (regiony).



Obrázek 7:
a) uzly (stránky) M-stromu obsahují záznamy objektů
b) regiony směrovacích objektů v metrickém prostoru

3.1 Vyhledávání v M-stromu

Při operacích na M-stromu počítáme se dvěma faktory složitosti. Prvním je **počet přístupů** na diskové stránky – resp. počet prohledávaných regionů, druhým (specifickou novinkou M-stromů) je **počet výpočtů vzdáleností** mezi objekty (což může být výpočetně náročná operace). Snahou je oba tyto faktory minimalizovat. Strukturální minimalizací prvního faktoru se zabývá kapitola 3.4. K minimalizaci druhého faktoru slouží předpočítané hodnoty $r(O_i)$ a $d(O_i, P(O_i))$ spolu se čtvrtým axiomem metriky (viz. kapitoly 1.3 a 3.1.1).

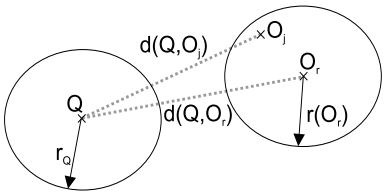
V metrických stromech se setkáváme se dvěma typy dotazů. *Rozsahové dotazy* vyhledají všechny objekty, které leží do určité vzdálenosti od dotazovaného objektu. Dotazy na *k-nejbližších sousedů* zase vyhledají k dotazovanému objektu k nejbližších objektů. V obou případech vidíme snahu o zavedení *částečného uspořádání* na objektech v metrickém

prostoru – vzhledem k dotazovanému objektu. Toto částečné uspořádání nám ilustruje onu *"lokální topologii"* metrického prostoru (viz. kap. 1.3).

3.1.1 Rozsahové dotazy

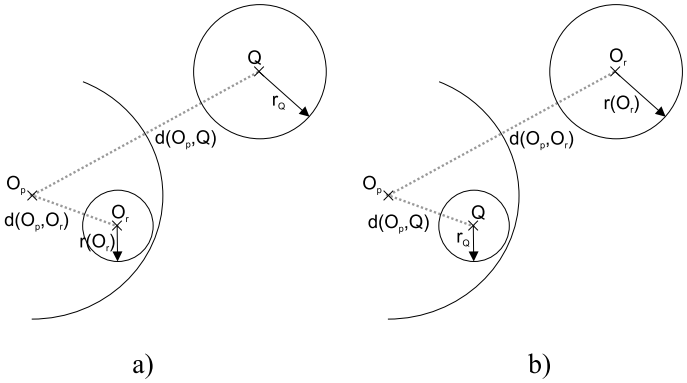
Rozsahový dotaz `range(`Q`,`r_Q`,`$\mathcal{C}$`)` provádí vyhledávání v kolekci objektů $\mathcal{C}$ (v M-stromu), kde Q je vzorový objekt a r_Q je práh maximální vzdálenosti (poloměr), do které se nacházejí hledané objekty od objektu Q, tzn. $d(O_j, Q) \leq r_Q$
Algoritmus `RS` prochází všechny cesty v M-stromu, které nemohou být vyloučeny užitím výše uvedené nerovnosti. Než přistoupíme k samotnému algoritmu `RS`, poukážeme na následující dvě okolnosti:

Lemma 1: Jestliže $d(Q, O_r) > r_Q + r(O_r)$ (tzn. pokud se neprotínají regiony Q a O_r), potom pro každý objekt O_j v podstromu $T(O_r)$ platí $d(Q, O_j) > r_Q$.



Obrázek 8: Důkaz lemma 1

Lemma 2: Jestliže $|d(O_p, Q) - d(O_r, O_p)| > r_Q + r(O_r)$, potom $d(O_r, Q) > r_Q + r(O_r)$.



Obrázek 9: Některé případy lemma 2

První věta zaručuje vyloučení regionu O_r pokud neprotíná region Q, druhá věta pomáhá minimalizovat použití první věty (kde se musí počítat vzdálenost $d(Q, O_r)$) pomocí známých hodnot $d(O_p, Q)$ a $d(O_r, O_p)$. Algoritmus `RS` (viz. obr. 10) obou těchto vlastností využívá.

```
RS(N: node, range(Q,r_Q): query)
{ let O_p be the parent object of node N;
  if N is not a leaf
  { ∀O_r in N do:
    if |d(O_p,Q) − d(O_r,O_p) ≤ r_Q + r(O_r)
    { Compute d(O_r,Q);
      if d(O_r,Q) ≤ r_Q + r(O_r)
         RS(*ptr(T(O_r)), range(Q,r_Q)); } }
  else
  { ∀O_j in N do:
    if |d(O_p,Q) − d(O_j,O_p)| ≤ r_Q
    { Compute d(O_j,Q);
      if d(O_j,Q) ≤ r_Q
         add oid(O_j) to the result; } } }
```

Obrázek 10: Algoritmus RS

3.1.2 Dotazy na k nejbližších sousedů

Algoritmus k-NN-Search je podobný algoritmu RS, avšak s některými modifikacemi. Úplný popis algoritmu ovšem přesahuje rámec tohoto článku. Ukažme si jej tedy alespoň v kostce.

Nejdůležitější je prioritní fronta PR, ze které se postupně vybírají požadavky na hledání v podstromu $T(O_r)$. Položka ve frontě má tvar: $(ptr(T(O_r)), d_{min}(T(O_r)))$. První člen je známý ukazatel na pokrývající strom, druhý člen je spodní hranicí vzdálenosti mezi Q a každým objektem O_j z $T(O_r)$, a platí:

$$d_{min}(T(O_r)) \leq d(O_j, Q) \qquad \forall O_j \in T(O_r)$$

$d_{min}(T(O_r))$ můžeme jednoduše spočítat:

$$d_{min}(T(O_r)) = max\{d(O_r, Q) - r(O_r), 0\}$$

Tyto hranice využívá funkce ChooseNode, která na jejich základě vybere z PR položku. Do pole NN se průběžně ukládá výsledek dotazu.

```
node ChooseNode(PR: priority_queue)
{ let d_min(T(O*_r)) = min{d_min(T(O_r))}, considering all the entries in PR;
  Remove entry [ptr(T(O*_r)), d_min(T(O*_r))] from PR;
  return *ptr(T(O*_r)); }
```

Obrázek 11: Funkce ChooseNode

```
k-NN-Search(T: root_node, NN(Q,k): query)
{ PR = [ptr(T),_];
  for i=1 to k
    NN[i] = [_,∞];
  while PR ≠ 0 do
  { Next_Node = ChooseNode(PR);
    k-NN-NodeSearch(Next_Node, NN(Q,k)); } }
```

Obrázek 12: Algoritmus `k-NN-Search`

Nejvíce "vyhledávací logiky" implementuje funkce `k-NN-NodeSearch`, kterou zde neuvádíme a odkazujeme na literaturu [PAT99].

3.2 Vytvoření M-stromu

Invarianty M-stromu zmíněné na začátku kapitoly definují velmi slabá kritéria pro samotnou organizaci objektů v M-stromu. Stejná kolekce objektů může být v M-stromu indexována mnoha různými způsoby, aniž by byly porušeny axiomy metriky nebo vlastnosti pokrývajících stromů. Nicméně každý takový způsob nemusí být ideální z hlediska efektivity operací prováděných nad M-stromem. Obecně lze říci, že se hledají takové metody konstrukce směrovacích objektů, aby se snížila složitost při dotazování na minimum.

3.3 Vložení objektu

Při vkládání objektu se rekurzivně prochází stromem a hledá se nejvýhodnější list, do kterého se objekt vloží. Nejvhodnější list je takový, při němž nedojde k rozšíření pokrývacího poloměru jeho rodičovských směrovacích objektů. Při postupu stromem "dolů" je vybrán takový směrovací objekt, jehož vzdálenost od vkládaného objektu je minimální. Algoritmus je popsán na obrázku 13. Funkce `Split` štěpí stránku/uzel M-stromu v případě překročení její kapacity. Zejména na pojetí této funkce závisí efektivita M-stromu tak, jak byla popsána v předchozí kapitole. Soubor problémů, které jsou řešeny v rámci štěpení stránek se označuje jako *politiky štěpení* (split policies) (viz. kapitola 3.4).

Popsaný algoritmus vložení objektu a zmíněné politiky štěpení stránek jsou vhodné pro **postupný** růst stromu objekt po objektu. Pro **dávkové** vytvoření M-stromu z kolekce existujících objektů se využívá lepších postupů založených na počátečním shlukování objektů v kolekci a teprve potom vložení do M-stromu. V [PAT99] je popsán původní algoritmus `BulkLoading`, který pracuje právě tímto dávkovým způsobem.

```
Insert(N: node, entry(O_n): M-tree_entry)
  { let 𝒩 be the set of entries in node N;
    if N is not a leaf then
    { let 𝒩_in = entries such that d(O_r, O_n) ≤ r(O_r);
      if 𝒩_in ≠ ∅ then
        let entry(O*_r) ∈ 𝒩_in: d(O*_r, O_n) is minimum;
      else { let entry(O*_r) ∈ 𝒩 : d(O*_r, O_n) − r(O*_r) is minimum;
             let r(O*_r) = d(O*_r, O_n); }
      Insert(*ptr(T(O*_r)),entry(O_n)); }
    else /* N is a leaf */
   { if N is not full then
      store entry(O_n) in N
     else Split(N,entry(O_n)); }}
```

Obrázek 13: Algoritmus vložení objektu

3.4 Politiky štěpení stránky

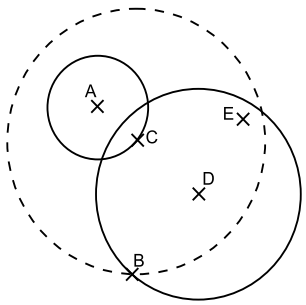
Jako každý vyvážený stránkovaný strom (např. i UB-strom), roste i M-strom směrem zdola-nahoru. Překročení kapacity (přetečení) uzlu N se vyřeší alokování nového uzlu N' na stejné úrovni, rozdělením původního obsahu (*partitioning*) uzlu N do těchto dvou uzlů a delegování dvou nových směrovacích objektů (*promoting*) do rodičovského uzlu N_p (ten může být opět přetečen). Pokud přeteče kořen stromu, rozdělí se, a místo delegování změny je alokován nový kořen, čímž strom o jednu úroveň vyroste. Specifické implementace fází *partitioning* a *promoting* definují politiky štěpení.

Z hlediska udržení maximální efektivity vyhledávání by měly být respektovány následující priority:

- **minimální "objem"**
  Udržování co nejmenších pokrývacích poloměrů umožňuje častěji splnit podmínky lemmat 1 a 2 a tím redukovat množství prohledávaných regionů.

- **minimální "překrývání"**
  Ačkoliv překrývání regionů směrovacích objektů ve stejném uzlu nelze kvůli obecné definici metriky kvantifikovat, je zřejmé, že separované regiony jsou pro vyhledávání výhodnější (ze stejných důvodů jako minimální objem). Vysoká míra překrývání vede k logické degeneraci struktury stromu a není potom zaručena logaritmická složitost při vyhledávání.

- **vyvážený strom**
  Známý požadavek pro ukládání dat ve stromech.



Obrázek 14: Příklad rozštěpení stránky – vidíme rozdělení uzlu (kapacity 4) směrovacího objektu C na dva uzly se směrovacími objekty A (obsahuje A,C) a D (obsahuje B,D,E). Bylo upřednostněno kritérium vyváženosti zaplnění uzlů, ačkoliv by bylo možno zvolit rozdělení A(A), D(B,C,D,E) a minimalizovat tak objemy i překrývání regionů.

3.4.1 Volba směrovacích objektů – promoting

Metoda `Promote` (uvnitř metody `Split`) vybírá dva objekty ze štěpené stránky, které deleguje do rodiče jako nové směrovací objekty pro dva nové regiony. Původní směrovací objekt v uzlu rodiče je vymazán. Uvádíme zde dvě alternativy (z mnoha) výběru nových směrovacích objektů.

- `RANDOM` – Tato metoda vybere náhodně dva z objektů. Ačkoliv se zdá jako "hloupá", je rychlá a může být srovnáním pro ostatní metody.
- `SAMPLING` – Vylepšená `RANDOM` metoda. Náhodně se zvolí vzorek objektů (sample) o velikosti s. Potom se pro každý z $s(s-1)/2$ párů ve vzorku spočítá rozdělení objektů a pokrývací poloměry. Objekty toho páru, který má nejmenší součet poloměrů se vyberou jako směrovací.

3.4.2 Rozdělování objektů – partitioning

Po výběru směrovacích objektů je třeba rozdělit prvky štěpené stránky do dvou stránek nových směrovacích objektů. Používají se dva způsoby rozdělení:

- **obecná nadrovina** – Každý objekt se přiřadí k nejbližšímu směrovacímu objektu.
- **vyvážené rozdělení** – Střídá se přidělování objektů k prvnímu a druhému směrovacímu objektu. V každém kroku se přidává nejbližší objekt k danému směr. objektu. Nové regiony se v součtu liší maximálně o 1 objekt.

3.5 Shrnutí

M-strom je vyvážená, vysoce parametrizovatelná datová struktura umožňující indexovat objekty metrického prostoru. Poskytuje zhruba logaritmické časové složitosti, ovšem narozdíl od UB-stromu nereprezentuje lineární uspořádání, z čehož vyplývá řada komplikací, zejména výpočetně náročné počítání vzdáleností. Na druhou stranu je obecnější než metody SAM.

4 Použití

- **UB-stromy** (a metody SAM obecně) jsou ideální pro indexování vektorových prostorů do dimenze přibližně 20. U vyšších dimenzí se začne projevovat tzv. *prokletí dimenzionality* (curse of dimensionality), které přináší značně neefektivní rozsahové dotazy.
- **M-stromy** Přicházejí ve chvíli, kdy jsou vektory vlastností dokumentů příliš dlouhé (dimenze > 20) nebo pokud dokumenty nelze indexovat ve vektorových prostorech. Definice speciálních metrik umožňují různě interpretovat pojem vzdálenosti (resp. podobnosti) a indexovat data podle jejich povahy.

Reference

[PSH98] J. Pokorný, V. Snášel, D. Húsek: *Dokumentografické informační systémy.* Karolinum, Praha 1998

[BAY96] R. Bayer: *The Universal B-Tree for multidimensional indexing.* `http://mistral.in.tum.de/results/publications/TUM-I9637.pdf`

[MAR99] V. Markl: *Processing Relational Queries using a Multidimensional Access Technique.* `http://mistral.in.tum.de/results/publications/Mar99.pdf`

[PAT99] M. Patella: *Similarity Search in Multimedia Databases.* Dipartmento di Elettronica Informatica e Sistemistica, Bologna 1999 `http://www-db.deis.unibo.it/~patella/MMindex.html`

[BBK] Ch. Böhm, S. Berchtold, D.A. Keim: *Searching in High-dimensional Spaces – Index Structures for Improving the Preformance of Multimedia Databases.* ACM