Vícerozměrný přístup pro netriviální vyhledávání termů

Michal Krátký, Tomáš Skopal, Václav Snášel

Katedra informatiky, FEI, VŠB–Technická univerzita Ostrava,
17. listopadu 15, 708 33, Ostrava-Poruba

{michal.kratky,tomas.skopal,vaclav.snasel}@vsb.cz

Abstrakt. V dokumentografických informačních systémech jsou textové dokumenty charakterizovány termy. Pro efektivní vyhledávání těchto termů je nutné zvolit vhodnou datovou strukturu. Přístupy pro uložení termů vyvinuté v minulosti (jako např. invertovaný seznam) umožňují pouze jednoduché dotazování nad množinou termů. Často jsme schopni pouze rozhodnout, jestli term charakterizuje nějaký dokument anebo zda je vůbec uložen v databázi. Složitější vyhledávání, jako např. vyhledat termy dle daného regulárního výrazu nebo dle Hammingovy či Levenshteinovy vzdálenosti, je velmi obtížné či dokonce nemožné. V našem příspěvku uvádíme vícerozměrný přístup pro netriviální vyhledávání termů.

Klíčová slova: Information Retrieval, indexování, datové struktury

1 Úvod

Oblast, která se zabývá zpracováním a vyhledávání v multimediálních dokumentech, je anglicky označována jako *Information Retrieval* [12, 3, 10]. Informační systémy pracující nad množinou multimediálních dokumentů jsou potom nazývány *dokumentografické informační systémy*. V dokumentografických systémech jsou textové dokumenty charakterizovány *termy* (obecně to může být slovo, více slov či slovní spojení přirozeného jazyka), nezávisle na použitém modelu (např. *Boolovském* či *Vektorovém* [10]). Pro efektivní vyhledávání těchto termů je nutné zvolit vhodnou datovou strukturu. Přístupy pro uložení termů vyvinuté v minulosti (jako např. invertovaný seznam) umožňují pouze jednoduché dotazování nad množinou termů. Často jsme schopni pouze rozhodnout zda term charakterizuje nebo necharakterizuje nějaký dokument. Složitější vyhledávání jako např. vyhledávání termů definovaných regulárním výrazem nebo získávání termů v Hammingově či Levenshteinově vzdálenosti (nebo libovolné jiné) od zadaného termu je velmi obtížné či dokonce nemožné. V našem příspěvku uvádíme vícerozměrný přístup pro netriviální vyhledávání termů.

Jeden z přístupů pro uložení termů charakterizujících dokumenty je invertovaný seznam. Invertovaný seznam obsahuje setříděné termy a ke každému z nich potom seznam dokumentů, které jsou daným termem charakterizovány. Pro

efektivní vyhledávání těchto termů, které musí být vykonáno při dotazu uživatele, je možné využít datové struktury jakou je např. B-strom [16]. Tato datová struktura nám umožní vyhledat term s logaritmickou časovou složitostí (s počtem termů). Invertovaný seznam tedy řeší úlohu zjišťující dokumenty, které jsou daným termem charakterizovány.

Tento přístup ale neumožňuje získat dokumenty, které jsou charakterizovány termem zadaným regulárním výrazem nebo termy "podobnými" k zadanému. Např. dotazem (`petr*`) budeme chtít získat všechny dokumenty, které jsou charakterizovány termy začínající prefixem `petr`, např. `petr` nebo `petrklíč`. V [10] je popsán algoritmus umožňující realizovat takovéto dotazy nad klasickým invertovaným seznamem uloženým v B-stromu.

Tento přístup ovšem není možné použít, pokud budeme chtít vyhledat termy "podobné" k zadanému. Pro podobnost dvou řetězců můžeme využít známých metrik jako je např. Hammingova nebo Levenshteinova.

V následující kapitole bude představen vícerozměrný přístup pro netriviální vyhledávání termů. Hlavní důraz bude kladen na konstrukci dotazů na základě dotazu uživatele. Vícerozměrný přístup využívá struktury pro indexování vícerozměrných vektorových prostorů. Jedna z těchto struktur – UB-strom – bude popsána v kapitole 3.

2 Vícerozměrný přístup pro netriviální vyhledávání termů

Ve vícerozměrném přístupu pro netriviální vyhledávání termů využíváme datových struktur pro indexováni *diskrétních vektorových prostorů* [2]. Tyto struktury nám umožňují pohlížet na termy jako na body v n-rozměrném prostoru. Takovými strukturami jsou např. *UB-stromy* [1] nebo *R-stromy* [5]. Získávání požadovaných termů (nebo dokumentů, které jsou těmito termy charakterizovány) se v tomto případě převádí na geometrické úlohy hledání bodů obsažených v definovaném n-rozměrném kvádru. Takovéto dotazy nazýváme *vícerozměrné rozsahové dotazy*. Term je v našem přístupu nutné transformovat do n-rozměrného bodu pro uložení a efektivní dotazování ve struktuře pro indexování vektorových prostorů. Tato transformace se často nazývá *extrakce vlastností* [2, 14], tedy zachycení relevantní informace z indexované entity se kterou můžeme nadále pracovat (tedy použít pro indexování, vyhledávání, atd.).

Definice 1 (n-rozměrný bod reprezentující term).
Vezměme term $t = c_{n-1}, c_{n-2}, \ldots, c_1, c_0$ délky n, kde c_i je znak, $0 \leq i < n$. n-rozměrný bod je definován $p_t = (code(c_{n-1}), code(c_{n-1}), ..., code(c_0))$, kde code je funkce kódující znak do binárního čísla délky m.

Je nutné zvolit maximální délku termu n a termy, které mají délku $l < n$, mají souřadnice p_{term_i}, $l \leq i < n$ rovné nule. Dimenze prostoru Ω je tedy rovna n a a kardinalita domén $|D_i| = 2^m$, $1 \leq i \leq n$. Prakticky může být maximální počet znaků (tedy dimenze prostoru) 32 a kardinalita domén (při použití ASCII

kódování) 256. Volba dimenze 32 plyne z maximálního počtu znaků slov vyskytujících se v přirozených jazycích, ale v případě nutnosti je možné dimenzi prostoru změnit. Počet dimenzí je tedy značný a efektivita může být ovlivněna tzv. *prokletím dimensionality* [2], algoritmy pracující s vícerozměrnými datovými strukturami jsou totiž často exponenciální s dimenzí prostoru. Na druhou stranu je kardinalita domén velmi malá.

Označme maximální hodnotu domény i (tedy hodnotu vyjádřenou m bity) jako $max_d = |D_i| - 1 = 2^m - 1$, v případě ASCII kódování je $max_d = 2^8 - 1 = 255$. Po transformaci termu do bodu n-rozměrného vektorového prostoru jej vložíme do vícerozměrné datové struktury jakou je např. UB-strom, nad kterou jsou prováděny rozsahové dotazy konstruované na základě dotazů uživatele.

2.1 Konstrukce dotazů

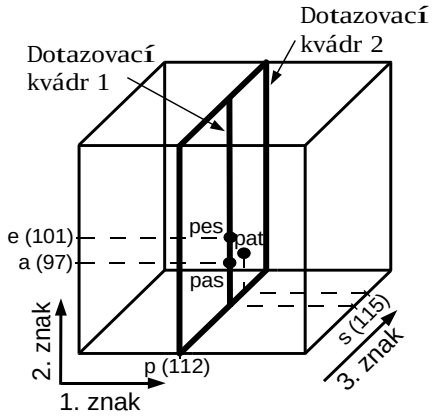
Ve vícerozměrném přístupu se úloha vyhledávání termů převádí na geometrickou úlohu hledání bodů (které v tomto případě charakterizují termy) ve vícerozměrném kvádru. Tento n-rozměrný kvádr nazveme *dotazovací kvádr* a můžeme jej definovat pomocí dvou n-rozměrných bodů. Při dotazu uživatele musíme být schopni vytvořit příslušný vícerozměrný kvádr (musíme tedy definovat dva body určující kvádr) a v datové struktuře vyhledat body, které náleží do tohoto kvádru. Body získané rozsahovým dotazem musíme transformovat do termů (aplikací inverzní funkce code^{-1}). Takto získané termy resp. dokumenty, které jsou těmito termy charakterizovány, vrátíme uživateli.

Příklad 1 (Konstrukce dotazovacího kvádru pro dotaz zadaný regulárním výrazem).
Vezměme termy `pes`, `pas` a `pat`. Po převodu do bodů n-rozměrného prostoru získáme body p_{pas} = (code('p'),code('a'),code('s')) = $(112, 97, 115)$, p_{pes} = (112, $101, 115$) a p_{pat} =$(112, 97, 116)$. Funkce code v tomto případě realizuje ASCII kódování ($m = 8$), kardinalita domén $|D_i| = 256$, $1 \leq i \leq 3$. Dimenzi prostoru n zvolíme 3 z důvodu vizualizace takovéhoto prostoru (viz obrázek 1).

Vezměme dva dotazy zadané regulárními výrazy `(p*s)` a `(p*)`. Pro zadané výrazy musíme nyní zkonstruovat rozsahové kvádry. Pro první regulární výraz definujeme 3-rozměrný dotazovací kvádr (code('p'),0,code('s')) $\times$ (code('p'), max_d, code('s')), výsledný dotazovací kvádr 1 vidíme na obrázku 1. Vidíme, že všechny body tohoto kvádru mají 1. a 3. souřadnice konstantní, pouze druhé souřadnice nabývají hodnot z intervalu $\langle 0, max_d \rangle$. Výsledkem rozsahového dotazu vykonávajícím dotazovací kvádr 1 jsou body reprezentující termy `pas` a `pes`.

Pro druhý regulární výraz definujeme 3-rozměrný dotazovací kvádr (code('p'), 0,0) $\times$ (code('p'),max_d,max_d), výsledný dotazovací kvádr 2 vidíme na obrázku 1. Výsledkem rozsahového dotazu vykonávajícím dotazovací kvádr 2 jsou body reprezentující termy `pes`, `pas` a `pat`.

Dalším typem dotazu může být vyhledání termů, které jsou "podobné" k zadanému termu. Podobnost termů můžeme měřit pomocí *Hammingovy* či *Levenshteinovy metriky* [7, 6].



Obr. 1. 3-rozměrný prostor obsahující body reprezentující termy a dotazovací kvádry pro získání termů daných regulárními výrazy (`p*s`) a (`p*`).

Definice 2 (Hammingova vzdálenost).
Hammingova vzdálenost $d_H(v, w)$ *mezi dvěma řetězci* $v, w \in \Sigma^*$, $|v| = |w|$ *je minimální počet editačních operací* výměna (replace) *ke konverzi* v *na* w.

Definice 3 (Levenshteinova vzdálenost).
Levenshteinova vzdálenost $d_L(v, w)$ *mezi dvěma řetězci* $v, w \in \Sigma^*$, *je minimální počet editačních operací* výměna (replace), vložení (insert) *a* zrušení (delete) *ke konverzi* v *na* w.

Příklad 2 (Příklady Hammingových a Levenshteinových vzdáleností).

$$\begin{array}{ll} d_H(\text{"}pes\text{"}, \text{"}pas\text{"}) = 1 & d_L(\text{"}pes\text{"}, \text{"}past\text{"}) = 2 \\ d_H(\text{"}pes\text{"}, \text{"}pat\text{"}) = 2 & d_L(\text{"}\sqcup pas\text{"}, \text{"}pas\text{"}) = 1 \\ d_H(\text{"}pes\text{"}, \text{"}pes\text{"}) = 0 & d_L(\text{"}pes\text{"}, \text{"}pes\text{"}) = 0 \end{array}$$

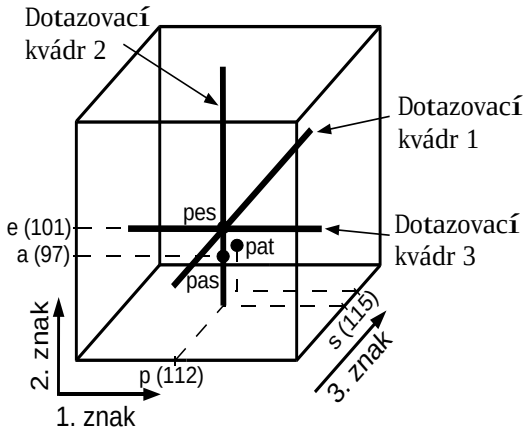
V našem přístupu jsou dotazy na podobnost prováděny sérií rozsahových dotazů realizovaných pomocí zkonstruovaných vícerozměrných kvádrů. Rovněž pro získávání termů zadaných regulárním výrazem nemusí být vždy dostatečný jeden rozsahový dotaz (např. (`*stav*`) pro $n = 12$).

Příklad 3 (Konstrukce dotazovacího kvádru pro Hammingovu vzdálenost).
Chtějme k termu `pes` nalézt všechny termy s Hammingovou vzdáleností 1 a 2, tedy chceme nalézt všechny řetězce w, pro něž platí $d_H(\text{"}pes\text{"}, w) = 1$ resp. $d_H(\text{"}pes\text{"}, w) = 2$. Opět musíme pro takovýto dotaz uživatele zkonstruovat dotazovací kvádr (nebo několik) pro rozsahový dotaz resp. rozsahové dotazy, který bude získávat požadované termy.

Termy s Hammingovou vzdáleností 1 od termu `pes` budou mít v tomto případě ($n = 3$) dva znaky totožné a třetí různý (při délce termu n potom $n - 1$

znaků totožných a jeden různý). Pro získání termů v Hammingově vzdálenosti 1 od zadaného termu musíme provést n rozsahových dotazů. V tomto konkrétním případě musíme realizovat tři rozsahové dotazy s kvádry, které vidíme na obrázku 2 a jsou popsány v následující tabulce:

dotazovací kvádr		získané termy
(code('p'),code('e'),0) × (code('p'),code('e'),max_d)	1	`pes`
(code('p'),0,code('s')) × (code('p'),max_d,code('s'))	2	`pes`, `pas`
(0,code('e'),code('s')) × (max_d,code('e'),code('s'))	3	`pes`



Obr. 2. 3-rozměrný prostor s dotazovacími kvádry pro získání termů v Hammingově vzdálenosti 1 od termu `pes`.

Jestliže budeme chtít získat termy v Hammingově vzdálenosti 2 od zadaného termu, je situace komplikovanější. Obecně bude $n-2$ znaků stejných a 2 odlišné. V tomto případě bude 1 znak stejný a 2 odlišné. Počet kvádrů bude opět n, ale tentokrát se nebude jednat o kvádry se dvěma souřadnicemi konstantními (jako v předchozím případě), ale pouze s jednou. V tomto konkrétním případě musíme realizovat tři rozsahové dotazy s kvádry:

dotazovací kvádr	získané termy
(code('p'),0,0) × (code('p'),max_d,max_d)	`pes`,`pas`,`pat`
(0,code('e'),0) × (max_d,code('e'),max_d)	`pes`
(0,0,code('s')) × (max_d,max_d,code('s'))	`pes`,`pas`

Objem dotazovacího kvádru je důležitý pro efektivitu rozsahového dotazu, doba vykonávání (počet přístupů na disk) je přímo úměrný objemu dotazovacího kvádru (např. v případě UB-stromů pokrývá objemnější kvádr více regionů

– stránek). Pokud by kvádr pokrýval celý prostor, potom bychom museli sekvenčně projít celou datovou strukturu (ve skutečnosti by se vyhledávání změnilo na sekvenční prohledávání všech listů stromu). Rozsahové kvádry pro získání termů délky n s Hammingovou vzdáleností d budou mít d souřadnic nekonstantních a jejich objem se bude zvětšovat. Jedním z úkolů naší budoucí práce bude pokusit se optimalizovat celkový objem série rozsahových dotazů (např. průnikem dotazovacích kvádrů), konstruovaných pro realizaci jednoho dotazu uživatele. Podobným způsobem bychom konstruovali dotazy na základě dalších požadavků uživatele. Dotaz by byl vždy převeden na dotaz rozsahový, který by byl vykonáván v datové struktuře pro indexování vektorových prostorů. Jednou z takovýchto datových struktur je například UB-strom (viz kapitola 3).

Problémem není jen převod dotazu uživatele na dotaz rozsahový dle zvolené metriky, ale především výběr samotné metriky. Jsou Hammingova nebo Levenshteinova vzdálenost skutečně použitelné pro měření podobnosti termů přirozených jazyků? Zodpovězení této otázky popř. vývoj jiné vhodné vzdálenosti by měl být dalším z bodů naší budoucí práce.

3 Univerzální B-stromy (UB-stromy)

Univerzální B-strom – UB-strom – je datová struktura pro indexování vícerozměrných vektorových prostorů. Tuto přístupovou metodu navrhl Rudolf Bayer (spoluautor B-stromu) a poprvé ji zveřejnil v roce 1997 v [1]. Další poměrně rozsáhlou publikací o UB-stromech a indexování vícerozměrných dat obecně je disertační práce [8]. UB-stromy byly dále vyvíjeny v rámci projektu Mistral [9].

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 a garantuje 50% využití uzlů stromu. Jestliže budeme chtít použít tak efektivní datovou strukturu jakou je B^+-strom, je nutné, aby na zaindexovaných hodnotách bylo zavedeno uspořádání. Jak ovšem uspořádat body v n-rozměrném prostoru? Řešením je použití tzv. *prostor plnících křivek*. Konkrétně pro UB-stromy definujeme tzv. *Z-křivku*, která je konstruována pomocí *Z-uspořádání*, resp. *Z-adres*.

3.1 Z-adresa a Z-region

Definice 4 (Z-adresa).
Pro n-tici $\mathcal{O} \in \Omega$ 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:

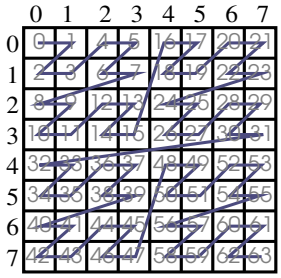
$$Z(\mathcal{O}) = \sum_{j=0}^{s-1}\sum_{i=1}^{n} A_{i,j}2^{jn+i-1}$$

Tato funkce spočítá k danému bodu v prostoru jeho Z-adresu. Pro výpočet Z-adresy existuje *algoritmus bitového prokládání* s lineární časovou složitostí (s délkou adresy). Existence takového algoritmu je velmi důležitá, protože výpočet

Z-adresy je nejčastěji používaná operace při indexování vícerozměrných dat pomocí UB-stromu. Pro každý prvek diskrétního prostoru Ω můžeme spočítat různou Z-adresu. Na obrázku 3a) 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.

	0	1	2	3	4	5	6	7
0	0	1	4	5	16	17	20	21
1	2	3	6	7	18	19	22	23
2	8	9	12	13	24	25	28	29
3	10	11	14	15	26	27	30	31
4	32	33	36	37	48	49	52	53
5	34	35	38	39	50	51	54	55
6	40	41	44	45	56	57	60	61
7	42	43	46	47	58	59	62	63

a)



b)

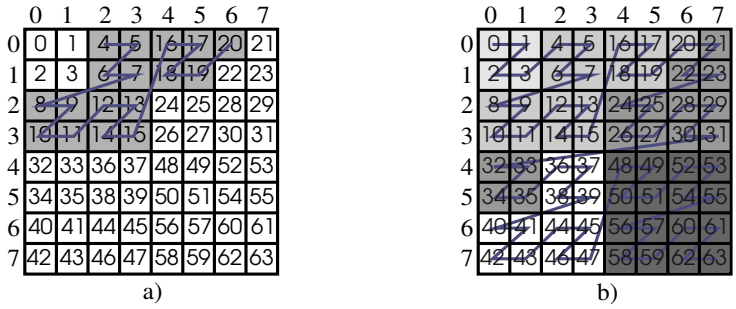
Obr. 3. a) Z-adresy pro každý prvek 2-rozměrného prostoru 8×8. b) Z-křivka pokrývající celý 2-rozměrný prostor 8×8.

Z definice Z-adresy plyne následující:

- délka bitových řetězců hodnot atributů musí být stejná (existují ovšem přístupy výpočtu adresy pro nestejně velké domény atributů, např. [8, 13]),
- tzn. prostor Ω je konečný a jeho velikost se nemění,
- pokud vlastnosti/atributy objektů nemají číselnou povahu, pro zaindexování do UB-stromu je potřeba převést hodnoty těchto atributů na celá čísla (to stejné platí pro čísla reálná),
- 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 3b) vidíme Z-křivku pokrývající celý 2-rozměrný prostor 8×8.

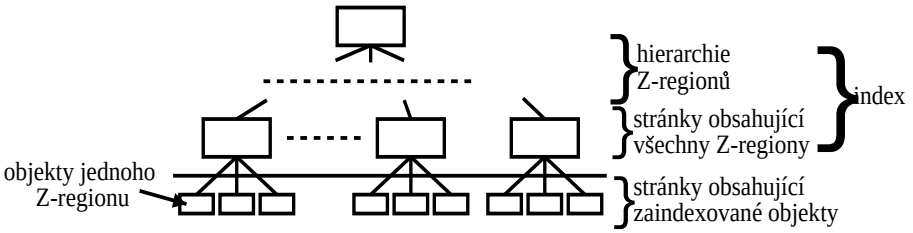
Pokud bychom zaindexovali Z-adresy objektů přímo do B^* resp. B^+-stromu, znamenalo by to neefektivní vykonávání vícerozměrných rozsahových dotazů. Přístupy pro indexování vícerozměrných dat přímo pomocí B-stromů existují, ale nejsou, v porovnání s datovými strukturami pro indexování vícerozměrných dat, příliš efektivní. V [8] autor uvádí porovnání B-stromů a UB-stromů pro indexování vícerozměrných dat. Z důvodu efektivního vykonávání vícerozměrných rozsahových dotazů jsou v UB-stromech zavedeny tzv. *Z-regiony*. Z-regiony shlukují blízko ležící body na diskové stránku. Účelem tohoto shlukování je minimalizovat počet přístupů na disk při operacích prováděných nad těmito daty. Pro vykonávání rozsahových dotazů je důležité, že Z-regiony dělí prostor na podprostory protínající dotazovací kvádr.



Obr. 4. 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 $\langle \alpha, \beta \rangle$ na Z-křivce. Na obrázku 4a) je prezentován Z-region [4:20] ve 2-rozměrném prostoru 8×8. Celý prostor lze potom rozdělit do Z-regionů (příklad na obrázku 4b), 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 5 vidíme jakým způsobem jsou v UB-stromu uložena data a Z-regiony.



Obr. 5. Uložení zaindexovaných objektů a Z-regionů v UB-stromu.

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). Úplný popis algoritmu rozsahového dotazu je poměrně složitý a je popsán např. v [1, 8]. Pro rozsahový dotaz uvádí autoři časovou složitost $r \times O(\log_k N)$, kde r je počet Z-regionů protínajících dotazovací kvádr.

3.2 Implementace UB-stromů

I přesto, že je UB-strom založen na B^+-stromu není implementace triviální (i samotná implementace B-stromu si vyžaduje některé speciální přístupy) a zejména je nutné řešit:

- možné velmi dlouhé bitové retězce. Při dimenzi 15 a kardinalitě domén 2^{32} (pro hodnotu atributu je nutné 32 bitů) musíme pracovat s bitovými řetězci o délce 480 bitů. Při aritě např. 20 bude velikost stránky 1.5 kB. Klíče (což jsou instance typu bitový řetězec) sice nemusí být uloženy v samotných stránkách stromu, ale např. v externím souboru, v každém případě veškeré algoritmy pracují s tímto datovým typem a musí být proto velmi dobře optimalizovány.
- algoritmus rozsahových dotazů. V [1] je uveden exponenciální (s dimenzí) algoritmus, v [8, 11] jsou pak pouze naznačeny algoritmy s lineární časovou složitostí. Tyto nejasnosti souvisí zřejmě s patentováním UB-stromů autory. Samotné UB-stromy byly integrovány do komerčního vícerozměrného SŘBD TransBase HyperCube [15]. Důsledkem je potom nyní nejasný a nedostupný algoritmus rozsahových dotazů. Proto autoři toho příspěvku v současné době pracují na vlastním algoritmu rozsahových dotazů.

Velmi důležitou vlastností UB-stromů je jejich vyváženost díky níž poskytují logaritmické časové složitosti pro základní operace. Dále řeší shlukování dat na diskové stránky pomocí Z-regionů. Hlavním problémem zůstává algoritmus rozsahových dotazů (viz sekce 3.2). Rovněž není zřejmé, a je tedy záležitostí dalších testů, porovnání s dalšími datovými strukturami pro indexování vícerozměrných vektorových prostorů jakou jsou např. dnes velmi dobře známé *R-stromy* [5].

4 Závěr

V tomto příspěvku byl prezentován přístup pro netriviální vyhledávání termů. Byla rozšířena funkcionalita původní struktury invertovaný seznam o možnost dotazů na termy odpovídající zadanému regulárnímu výrazu nebo vyhledávání podobných termů na základě Hammingovy nebo Levenshteinovy vzdálenosti. Ve vícerozměrném přístupu jsou dotazy transformovány na geometrické úlohy hledání bodů reprezentujících termy ve vícerozměrném kvádru. Vykonávání některých dotazů není úplně efektivní, ale dotazy jsou možné na rozdíl od stávajících přístupů.

V budoucí práci bychom chtěli provést testy porovnávající stavající přístupy s vícerozměrným prováděnými nad reálnými daty s využitím různých datových struktur pro indexování vícerozměrných vektorových prostorů, ať již zde popisovaných UB-stromů nebo zmiňovaných R-stromů. Dále například použít struktury pro indexování metrických prostorů (jakou je např. *M-strom* [4]), jejichž efektivitu tolik neomezuje prokletí dimensionality. Naproti tomu při využití těchto struktur je možné zaindexovat termy pouze dle pevně zvolené metriky. Dalším úkolem by mělo být identifikování vhodné vzdálenosti pro měření podobnosti termů přirozených jazyků. Jelikož v současné době není jasné zda jsou Hammingova nebo Levenshteinova vzdálenost skutečně použitelné pro měření takovéto podobnosti.

Reference

1. Bayer R.: *The Universal B-Tree for multidimensional indexing: General Concepts.* In Proceedings of World-Wide Computing and its Applications'97. Tsukuba,Japan.
2. Böhm C., Berchtold S., Keim D.A.: *Searching in High-dimensional Spaces – Index Structures for Improving the Performance Of Multimedia Databases.* ACM, 2002.
3. Baeza-Yates R., Ribeiro-Neto B.: *Modern Information Retrieval.* Addison Wesley, New York, 1999.
4. Ciaccia P, Pattela M., Zezula P.: M-tree: *An Efficient Access Method for Similarity Search in Metric Spaces.* In Proceedings of 23rd International Conference on VLDB, Athens, Greece, 1997, 426–435.
5. Guttman A., *R-Trees: A Dynamic Index Structure for Spatial Searching.* In Proceedings of ACM SIGMOD 1984, Annual Meeting, Boston, USA, ACM Press, 1984.
6. Holub J.: *Simulation of Nondeterministic Finite Automata in Pattern Matching.* Ph.D. thesis, Czech Technical University, Prague, 2000.
7. Levenshtein I.V.: *Binary codes capable of correcting deletions, insertions and reversals.* Soviet Physics-Doklady 10, 1966, 707–710.
8. Markl, V.: *Mistral: Processing Relational Queries using a Multidimensional Access Technique.* `http://mistral.in.tum.de/results/ publications/Mar99.pdf`, 1999.
9. Mistral: *The project MISTRAL (Multidimensional Indexes for Storage and for the Relational Algebra).* `http://mistral.in.tum.de`, 1999.
10. Pokorný J., Snášel V., Húsek D.: *Dokumentografické Informační Systémy.* Nakladatelství Karolinum, Praha, ISBN 382-146-98, 1998.
11. Ramsak F., Markl V., Fenk R., Zirkel M., Elhardt K., Bayer R.: *Integrating the UB-Tree into a Database System Kernel.* In Proceedings of the 26th International Conference on VLDB, Cairo, Egypt, 2000.
12. Salton G., McGill M.J.: *Introduction to Modern Information Retrieval.* McGraw Hill Publications, 1st edition, 1983.
13. Skopal T., Krátký M., Snášel V.: *Properties Of Space Filling Curves and Usage with UB-trees.* ITAT 2002, Brdo, High Fatra, Slovakia, accepted.
14. Skopal T., Krátký M., Snášel V.: *Geometrické dotazování a indexování multimediálních dat.* In Proceedings of DATAKON 2002, Brno, Czech Republic, 2002.
15. TransAction Software GmbH. *TransBase HyperCube - Module for multidimensional indexing of RDBMS TransBase.* `http://www.transaction.de/ en/products/hypercube.html`, 1998.
16. Wirth N.: *Algorithms and Data Structures.* Prentice Hall, 1984.

Annotation.

Multidimensional approach for non-trivial terms searching

The area of computer science dealing with searching/processing collections of multimedial documents is known as *Information Retrieval.* In documentographic systems the text documents are characterized with terms. For effective retrieval the terms must be stored in appropriate data structures. Many term searching approaches (e.g. inverted list) are able to process only the simple queries. We can use these approaches for finding out whether the term is or is not stored in the term database. But non-trivial term searching (according regular expressions, Hamming or Levenshtein distance, etc.) is hardly feasible or even impossible. In this paper, we introduce a multidimensional approach for non-trivial term searching.