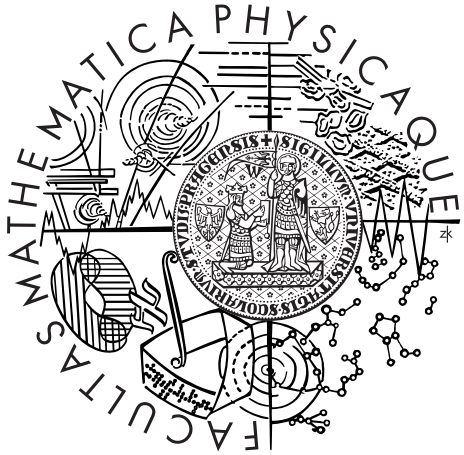
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCA



Bc. Juraj Moško

Modifikace metody Pivot Tables pro perzistentní metrické indexování

Katedra softwarového inženýrství

Vedúci diplomovej práce: doc. RNDr. Tomáš Skopal, Ph.D.

Štúdijný program: Informatika

Štúdijný obor: Softwarové systémy

Praha 2011

Chcel by som poďakovať doc. RNDr. Tomášovi Skopalovi, Ph.D. za vedenie diplomovej práce a za jeho čas strávený pri konzultáciách. Taktiež ďakujem RNDr. Jakubovi Lokočovi, Ph.D. za odborné rady a usmernenie počas neprítomnosti vedúceho diplomovej práce. V neposlednom rade ďakujem mojej rodine a všetkým, ktorí ma podporovali počas štúdia.

Prehlasujem, že som túto diplomovú prácu vypracoval(a) samostatne a výhradne s použitím citovaných prameňov, literatúry a ďalších odborných zdrojov.

Beriem na vedomie, že se na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona v platnom znení, zvlášť skutočnosť, že Univerzita Karlova v Praze má právo na uzatvorení licenčnej zmluvy o použití tejto práce ako školného diela podľa §60 odst. 1 autorského zákona.

V ..... dňa ..... Podpis autora

Názov práce: Modifikace metody Pivot Tables pro perzistentní metrické indexování

Autor: Bc. Juraj Moško

Katedra: Katedra softwarového inženýrství

Vedúci diplomovej práce: doc. RNDr. Tomáš Skopal, Ph.D.

Abstrakt: Metrická prístupová metóda pivot tables je jedna z najefektívnejších metód podobnostného vyhľadávania optimalizovaná na počet výpočtov vzdialeností medzi indexovanými objektmi. V tejto práci bola navrhnutá modifikácia metódy pivot tables, ktorá je navyše optimalizovaná aj na počet I/O operácií. Navrhnutá metóda Clustered pivot tables indexuje zhluky podobných objektov, ktoré boli vytvorené pomocou ďalšej metrickej prístupovej metódy - M-stromu. Indexovanie zhlukov podobných objektov prináša pozitívny efekt do vyhľadávania v indexovanej databáze, keď objekty z jedného zhluku sú uložené v stránke v sekundárnej pamäti, ktorá nemusí byť vôbec načítaná, ak daný zhluk dát neodpovedá príslušnému dotazu. Nerelevantné objekty tak nie sú zbytočne načítané zo sekundárnej pamäti, čím sa znižuje počet I/O operácií a celkový objem prenesených dát. Korektnosť daného prístupu bola experimentálne dokázaná a výsledky navrhovanej metódy bola porovnané s vybranými metrickými prístupovými metódami.

Kľúčové slová: podobnostné vyhľadávanie, metrické prístupové metódy, indexovanie, M-strom, pivot tables

Title: Modification of Pivot Tables method for persistent metric indexing

Author: Bc. Juraj Moško

Department: Department of Software Engineering

Supervisor: doc. RNDr. Tomáš Skopal, Ph.D.

Abstract: The pivot tables is one of the most effective metric access method optimized for a number of distance computations in similarity search. In this work the new modification of the pivot tables method was proposed that is besides distance computations optimized also for a number of I/O operations. Proposed Clustered pivot tables method is indexing clusters of similar objects that were created by another metric access method - the M-tree. The indexing of clustered objects has a positive effect for searching within indexed database. Whereas the clusters are paged in second memory, page containing such cluster, which do not satisfy particular query, is not accessed in second memory at all. Non-relevant objects, that are out of the query range, are not loaded into memory, what has the effect of decreasing number of I/O operations and total volume of transferred data. The correctness of proposed approach was experimentally proved and experimental results of proposed method was compared to selected metric access methods.

Keywords: similarity search, metric access methods, indexing, M-tree, pivot tables

Obsah

Úvod

V posledných rokoch sa charakter spracovávaných a ukladaných dát vyvinul natoľko, že dnes vo svete informačných technológií zohrávajú hlavnú úlohu multimedálne dáta. Obrazové dáta rôzneho charakteru, záznamy z bezpečnostných kamier, časové rady, reťazce DNA, geografické dáta, XML dokumenty to všetko je potreba nejako ukladať a následne s uloženými dátami ďalej pracovať. Navzdory tomu, že najviac používané relačné databázy sa neustále vyvíjajú a ponúkajú nové možnosti na ukladanie multimediálnych dát, ich koncepcia, ktorá vznikla v 60. - 70. rokoch minulého storočia, nepočítala s multimediálnymi dátami. Aj preto vznikol nový typ databáz - multimediálne databázy.

Multimediálna databáza väčšinou nepracuje so samotnými multimediálnymi dátami, ale s vlastnosťami, ktoré sú z týchto dát vyextrahované. Tieto vlastnosti, reprezentujúce daný multimediálny objekt, sú štruktúrované, aby s nimi dokázali procesy multimediálnej databázy pracovať. Okrem samotných indexačných a vyhľadávacích metód, určujúcim parametrom multimediálnej databázy je miera podobnosti objektov, zvyčajne definovaná ako binárna funkcia, ktorá dvojici objektov určí ich vzájomnú *vzdialenosť* . Táto funkcia je využitá pri podobnostnom vyhľadávaní, objekty sú porovnávané s dotazovým objektom a dotaz splňujú tie, ktoré sú dotazovému objektu najpodobnejšie, resp. najbližšie, rozsah výsledku ako aj dotazový objekt sú stanovené parametrami dotazu.

Narastajúci objem spracovávaných multimediálnych dát vedie stále k vzniku novým indexačným metódam, ktoré multimediálne databázy používajú. Vhodné meradlo pre porovnanie týchto metód sa môže s databázou od databázy líšiť. Databáza, do ktorej sú dáta vkladané priebežne, potrebuje dynamickú indexačnú metódu, dôraz je potom kladený aj na rýchlosť indexácie dát. Naproti tomu u kolekcií, ktoré sú nemenné, postačí statická indexačná metóda. U statických metód je akceptovaná aj nižšia rýchlosť indexácie kolekcie. Všeobecne sa však predpokladá, že aj u dynamických indexačných metód sa častejšie v dátach vyhľadáva, ako sa do nich dáta vkladajú. Parametrami pre porovnanie rôznych indexačných metód sú reálny čas vyhľadania, resp. čas indexácie konkrétnych, prípadne všetkých dát, počet výpočtov funkcie vzdialenosti (DC [1]), u perzistentných metód taktiež počet I/O operácií a v neposlednom rade sú metódy porovnávané aj na základe interných výpočtov.

Ciele práce

Cieľom tejto diplomovej práce je navrhnúť a implementovať modifikáciu metrickej prístupovej metódy *pivot tables* (PT), optimalizovanú pre perzistentné indexovanie. Štruktúra indexu metódy PT pozostáva z dvoch častí, matice vzdialeností objektov databázy k referenčným objektom, pomocou ktorej sa filtruje, a samotných databázových objektov. Navrhovaná metóda bude implementovať perzistentný index metódy PT, kde databázová časť bude oddelená od matice vzdialeností. Keďže priama implementácia perzistencie u metódy PT by viedla k vysokému počtu I/O operácií, pokúsime sa tento počet znížiť pred-spracovaním

[1] z anglického distance computation

indexovaných dát ďalšou metrickou prístupovou metódou, *M-stromom*. Súčasťou práce je experimentálne porovnanie priamej (bez pred-spracovania dát) a modifikovanej varianty perzistentnej metódy PT. Keďže hlavný rozdiel oproti bežnej implementácii metódy PT spočíva v predzhlukovaní vstupných dát, navrhnutú modifikáciu sme nazvali *Clustered pivot tables* (CPT). Metrické prístupové metódy predpokladajú existenciu extrahovaných vlastností, preto sa v tejto práci nebudeme venovať metódam extrakcie vlastností z multimediálnych dát.

Štruktúra dokumentu

V nasledujúcej kapitole sú zhrnuté základné princípy podobnostného vyhľadávania a metrických prístupových metód. Kapitola 2 ponúka popis použitých indexačných metód - pivot tables a M-strom. Kapitola 3 obsahuje motiváciu a popis navrhnutej modifikácie metódy CPT. V kapitole 4 nahliadneme na implementačné detaily metódy CPT. V kapitole 5 sú prezentované vykonané experimenty a práca je zakončená zhrnutím dosiahnutých výsledkov, výhod a nevýhod navrhovanej metódy v diskusii (5.4) a v závere.

1. Podobnostné vyhľadávanie

Myšlienka podobnostného vyhľadávania vyvstala ako odpoveď na potrebu vyhľadávať vo veľkých multimediálnych kolekciách. Multimediálne dáta sú všeobecne neštrukturované, preto je náročné v multimediálnych kolekciách vyhľadávať štandardným spôsobom, aký používajú relačné databázy. Z multimediálnych dát sú najprv vyextrahované vlastnosti - štrukturované dáta, ktoré charakterizujú samotné dáta. Vyhľadávanie v kolekcii vlastností prebieha spôsobom *query-by-example*, kde je vyhľadávaciemu algoritmu predaný dotazový objekt (vyextrahované vlastnosti skutočného multimediálneho objektu) a ďalším parametrom je definovaný rozsah dotazu. Vyhľadávacia metóda následne prehľadáva kolekciu vlastností, porovnáva dotazový objekt s objektami v databázy a objekty, ktoré sa nachádzajú v rozsahu dotazového objektu, sú algoritmom vrátené ako výsledok dotazu. Keďže multimediálne kolekcie môžu narásť do obrovských rozmerov, vyhľadávanie sekvenčným priechodom kolekcie by sa mohlo stať neúnosným, preto začali vznikať rôzne indexačné metódy. Najviac uznávané sa stali tzv. *metrické prístupové metódy* [22, 16, 4], zahŕňajúce databázové štruktúry a indexačné metódy, zamerané na zníženie počtu DC a I/O operácií.

1.1 Metrické prístupové metódy

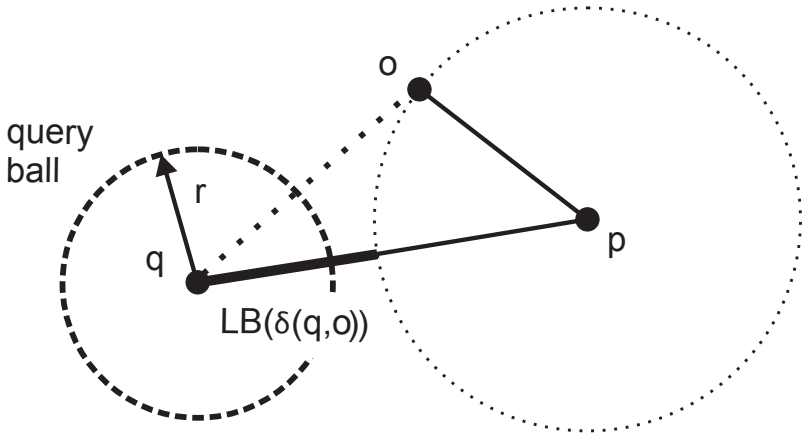
Metrická prístupová metóda (MAM [1]) operuje nad metrickým priestorom $(\mathbb{U}, \delta)$, kde $\mathbb{U}$, označované ako *univerzum*, je priestor extrahovaných vlastností multimediálnych dát a δ - funkcia vzdialenosti je metrika, teda musí splňovať metrické axiómy:

- reflexivitu $\delta(O_i, O_j) = 0$
- pozitivitu $\delta(O_i, O_j) > 0 \Leftrightarrow O_i \neq O_j$
- symetriu $\delta(O_i, O_j) = \delta(O_j, O_i)$
- trojuholníkovú nerovnosť $\delta(O_i, O_j) + \delta(O_j, O_k) \geq \delta(O_i, O_k)$

Úlohou MAM je indexovať kolekciu extrahovaných vlastností $\mathbb{S} \subset \mathbb{U}$ z dôvodu efektívneho vyhľadávania v databázovej kolekcii. MAM zoskupuje objekty kolekcie $o \in \mathbb{S}$ na základe znalosti ich vzájomných vzdialeností, ktorú definuje funkcia vzdialenosti (metrika) $\delta : U \times U \rightarrow \mathbb{R}$. Keďže vyhodnotenie funkcie vzdialenosti medzi dvoma objektami môže byť výpočetne drahé, MAM sa okrem štandardnej databázovéj snahy minimalizovať počet I/O operácií zameriava aj na znižovanie počtu DC.

Väčšina metrických prístupových metód sa pokúša znížiť počet výpočtov drahej funkcie vzdialenosti pomocou výpočtu lacnejšieho dolného odhadu skutočnej vzdialenosti. Tento odhad je určený na základe pred-vypočítaných vzdialeností medzi objektami databázy a nejakými referenčnými objektami, nazývanými pivoti. Práve vzdialenosti k referenčným objektom slúžia na zoskupenie objektov v metrickom priestore a vďaka týmto pred-vypočítaným vzdialenostiam celé

[1] z anglického Metric Access Method



Obrázok 1.1: Určenie dolného odhadu vzdialenosti $\delta(q, o)$.

skupiny objektov môžu byť odfiltrované bez počítania skutočných vzdialeností k dotazovému objektu, čo vedie k zvýšeniu efektivity vyhodnodcovania dotazu.

Princíp výpočtu dolného odhadu skutočnej vzdialenosti znázorňuje obrázok 1.1. Nech $q \in \mathbb{U}$ je dotazový objekt, $p \in \mathbb{S}$ je pivot a $o \in \mathbb{S}$ je objekt, pre ktorý chceme vypočítať vzdialenosť k dotazovému objektu q. Daný je polomer dotazu r, potom dotaz by mal vrátiť všetky objekty o, ktoré sú vzdialené od dotazového objektu nanajvýš o polomer dotazu, teda $\delta(q, o) \leq r$. Odhad skutočnej vzdialenosti porovnávaného objektu od dotazového objektu $\delta(q, o)$ sa dá určiť pomocou znalosti vzdialeností medzi pivotom a porovnávaným objektom, $\delta(p, o)$, a medzi pivotom a dotazovým objektom, $\delta(p, q)$. Vzdialenosť $\delta(p, o)$ je pred-vypočítaná, uložená v štruktúre indexu a vzdialenosť $\delta(p, q)$ je vypočítaná raz behom vyhodnocovania dotazu. Vďaka trojuholníkovej nerovnosti je možné bezpečne odfiltrovať objekt o, ak dolný odhad skutočnej vzdialenosti 1.1,

$$LB(\delta(q, o)) = |\delta(q, p) - \delta(o, p)| \tag{1.1}$$

je väčší než polomer dotazu r.

1.2 Typy dotazov

Vyhľadávania na základe podobnosti objektov so sebou prináša aj odlišný spôsob kladenia dotazov. V relačných databázach sú uložené dáta samo-vysvetľujúce, jednorozmerné a obor hodnôt známy, takže je možné dotazovať sa na konkrétne dáta priamo, bodovým, resp. intervalovým dotazom. Extrahované vlastnosti multimediálnych dát vo väčšine prípadov sú práve naopak viacrozmerné a nie sú samovysvetľujúce, je preto náročné vyjadriť štandardným dotazovacím jazykom podmienku pre vyhľadávanie. Preto bolo zavedené podobnostné vyhľadávanie a dotazy sú kladené formou *query-by-example*.

U podobnostného vyhľadávania rozlišujeme 2 základné typy dotazov:

- rozsahový dotaz
- dotaz na najbližších k susedov

Rozsahový dotaz, ako už názov napovedá, je okrem dotazového objektu určený aj rozsahom dotazu, ktorý definuje, nakoľko môžu byť vrátené objekty vzdialené

od dotazového objektu. Metrické prístupové metódy chápu rozsahový dotaz ako hyper-guľový región metrického priestoru s polomerom odpovedajúcim rozsahu dotazu a so stredom v dotazovom objekte. Vyhodnotenie rozsahového dotazu potom znamená vrátiť tie objekty, ktoré spadajú do tohoto hyper-guľového *dotazového regiónu*.

Dotaz na najbližších k susedov (k-NN [2] dotaz) definuje rozsah dotazu v inom zmysle. Namiesto polomeru dotazu je definovaný parameter k, ktorý stanovuje počet vrátených objektov, teda selektivitu dotazu. Pre k-NN dotaz sa taktiež zavádza pojem polomer dotazu ako vzdialenosť k najvzdialenejšiemu vrátenému objektu. V tomto prípade je polomer dotazu počítaný dynamicky v rámci procesu vyhodnocovania dotazu. Ak je $k = 1$ tak hovoríme o dotaze na najbližšieho suseda (NN dotaz).

Ďalšie typy dotazov sú z týchto dvoch typov odvodené. Bodovým dotazom sa rozumie rozsahový dotaz s polomerom dotazu rovným 0. Dotaz na reverzných k najbližších susedov [9] vráti tie objekty, pre ktoré dotazový objekt patrí medzi ich k najbližších susedov.

1.3 Ďalšie prístupové metódy

Metrické prístupové metódy nie sú jediné indexačné metódy využívané v multimediálnych databázach. Ďalšiu kategóriu tvoria Multidimenzionálne prístupové metódy (SAM [3]), ktoré operujú nad vektorovým priestorom. SAM na dotazovanie využíva tzv. *window query* - obdĺžnikový dotaz, ktorý vo vektorovom priestore vymedzuje región dotazu. Medzi najznámejšie SAM patrí R-strom [7], hierarchická štruktúra ktorý zoskupuje dáta do obdĺžnikových regiónov.

Výhody MAM oproti SAM:

- MAM pracuje nad všeobecným metrickým priestorom, neobmedzuje sa len na vektorový priestor
- MAM využíva metódy vyhovujúce podobnostnému vyhľadávaniu, ako sú dotazy založené na podobnosti objektov (k-NN a rozsahový dotaz), použitie metriky a s tým spojená možnosť filtrovať dáta s využitím metrických axiómov
- SAM je väčšinou optimalizovaná len na minimalizáciu počtu I/O operácií, MAM zavádza novú mieru efektivity indexačnej metódy a to počet DC

[2] z anglického k-nearest neighbors
[3] z anglického Spatial Access Method

2. Súvisiace práce

V tejto kapitole popíšeme 2 metrické prístupové metódy, ktoré sme použili pre implementáciu navrhovanej metódy Clustered pivot tables. Ide o základné princípy metód pivot tables a M-stromu.

2.1 Metóda pivot tables

Metóda pivot tables [13] je jedna z najefektívnejší metrických indexov, pričom ostáva zároveň aj jednou z najjednoduchších. Idea metódy PT bola prvýkrát predstavená s metódou LAESA [12], ktorá sa stala hlavným predstaviteľom princípov PT. Štruktúra indexu PT pozostáva z jednoduchej dvojrozmernej matice, ktorá obsahuje vzdialenosti medzi objektmi databázy a referenčnými objektmi (pivotmi). Hlavný princíp vyhľadávania spočíva v sekvenčnom prechode maticou pred-vypočítaných vzdialeností a filtrovaní objektov, ktoré sa nachádzajú mimo dotazový región. Filtruje sa na základe vypočítaného dolného odhadu skutočnej vzdialenosti medzi objektom databázy a dotazovým objektom, výpočet dolného odhadu popisuje vzťah 1.1.

2.1.1 Výber pivotov

Pre výpočet matice vzdialeností je potrebné zvoliť kolekciu pivotov, ku ktorým sa vzdialenosti budú počítať. Za pivoty sú volené objekty z indexovanej databázy. Existuje niekoľko rôznych metód na výber pivotov, najjednoduchšia metóda navrhuje voliť pivoty náhodne. Metóda výberu pivotov, ktorú ponúkajú autori metódy LAESA [12], volí kolekciu maximálne separovaných pivotov. V práci [14], navrhnutá metóda vyberá pivoty dobre distribuované v priestore, ktoré ale nie sú veľmi vzdialené od seba a zároveň nie sú vzdialené od ostatných objektov databázy. Tento navrhovaný algoritmus výberu dokonca nepotrebuje počet pivotov ako vstupný parameter, vybraná kolekcia pivotov je navyše dynamická, po vložení nových objektov do databázy sa môže kolekcia rozrásť. Na túto prácu nadväzuje ďalšia dynamická metóda [2], ktorá po pridaní nových prvkov skúma aktuálnu kolekciu pivotov a nahradzuje tie pivoty, ktoré sa správajú redundantne.

Vplyv pivotov na efektívne filtrovanie objektov pri dotazovaní určuje jednak ich počet a taktiež aj distribúcia v priestore. Všeobecne prevláda názor, že kolekcia pivotov, ktoré sú ďaleko od seba je efektívnejšia ako kolekcia pivotov, ktoré sú blízko seba, pretože pivoti, ktorí splývajú, sa správajú redundantne pri filtrovaní objektov.

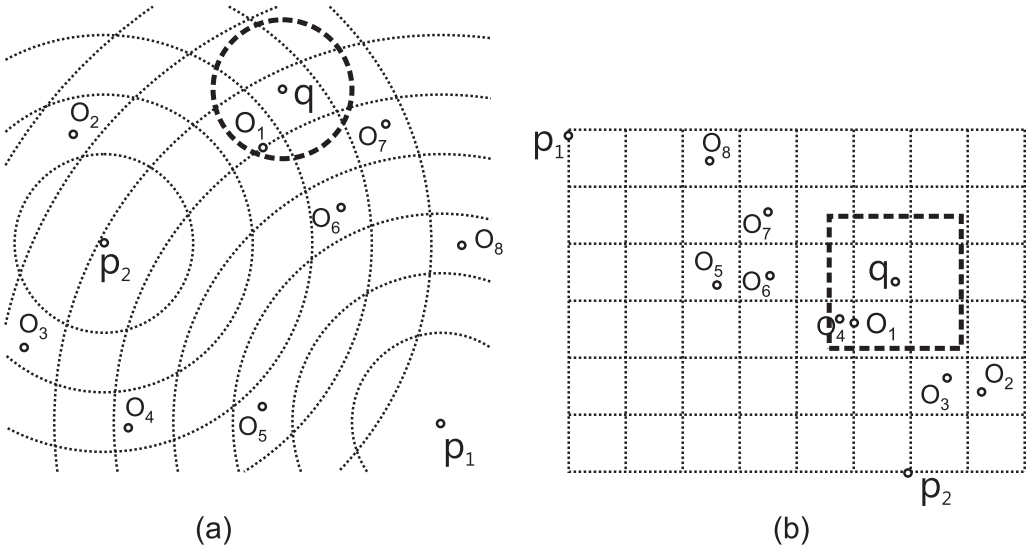
2.1.2 Štruktúra indexu

Po výbere pivotov metóda PT vypočíta a uloží vzdialenosti databázových objektov k pivotom, niektoré metódy [12] si vzdialenosti ukladajú už počas výberu pivotov. Tieto vzdialenosti sú uložené do dvojrozmernej matice a tvoria prvú časť indexu PT. Druhú časť indexu tvoria samotné indexované dáta. Spočítaním vzdialeností k pivotom sú objekty z metrického priestoru databázy namapované

do vektorovového priestoru. Namapované objekty sú v novom priestore pivotov reprezentované m-dimenzionálnymi vektormi, m je počet pivotov.

2.1.3 Vyhľadávanie

Vyhľadávanie v databáze objektov indexovanej metódou PT začína spočítaním vektoru vzdialeností dotazového objektu q k pivotom $p_1, p_2, ..., p_m$. Týmto mapovaním dotazového objektu do priestoru pivotov dostaneme nový stred dotazového regiónu $(\delta(q, p_1), \delta(q, p_2), ..., \delta(q, p_m))$ v namapovanom priestore. Mapovanie pôvodného priestoru do priestoru pivotov je znázornené na obrázku 2.1. Jedna z najdôležitejších vlastností tohoto mapovania je, že funkcia vzdialenosti L_∞ v mapovanom priestorom pivotov je dolným odhadom funkcie vzdialenosti δ v pôvodnom priestore, t.j. ide o neexpanzívne mapovanie. Vďaka tomu objekty, ktoré sa v priestore pivotov nachádzajú mimo namapovaný dotazový región, môžu byť odfiltrované. Pre ostatné objekty, ktoré vyhovujú namapovanému dotazu, sa dopočíta skutočná vzdialenosť k dotazu v pôvodnom priestore a tak sa odfiltrujú ďalšie objekty, ktoré ležia mimo dotazový región v pôvodnom priestore. Konečná kolekcia neodfiltrovaných objektov tvorí výsledok dotazu.



Obrázok 2.1: (a) Pôvodný priestor (b) Priestor pivotov

Pri vyhodnocovaní k-NN dotazu neexistuje polomer dotazu, tým pádom, ani dotazový región v namapovanom priestore, preto sa región dotazu počíta dynamicky v priebehu vyhodnocovania dotazu. Pre prvých k objektov sa spočíta skutočná vzdialenosť k dotazovému objektu, tieto objektu sa stanú kandidátmi na k najbližších susedov a vzdialenosť k najvzdialenejšiemu z nich je použitá ako dynamický polomer dotazu. Pomocou takto vzniknutého polomeru sa filtrujú objekty v mapovanom priestore, pričom neodfiltrované objekty a ich skutočná vzdialenosť k dotazu priebežne aktualizujú dynamický polomer a kolekciu najbližších susedov.

Algoritmus vyhodnocovania NN dotazu, ktorý ponúka metóda LAESA [12], najprv spočíta dolné odhady skutočných vzdialeností objektov databázy k dotazovému objektu, tieto dolné odhady následne zoradí od najmenšieho k najväčšiemu, v tomto poradí dopočítava skutočnú vzdialenosť (aktualizuje dynamický polomer

dotazu) a zároveň aktualizuje kandidáta na najbližšieho suseda. V momente, keď dolný odhad vzdialenosti je väčší ako dynamický polomer dotazu, vyhľadávanie končí a aktuálny kandidát je najbližším susedom dotazového objektu.

2.1.4 Odvodené metódy

Základné princípy pred-vypočítaných vzdialeností metódy PT sa objavili už v metóde AESA [15], z ktorej LAESA vychádza. AESA neuvažuje kolekciu pivotov ako takú, namiesto toho má uložené pred-vypočítané vzdialenosti medzi každými dvoma objektmi databázy. Dotazový objekt je pri vyhľadávaní mapovaný do priestoru vzdialeností po častiach, zároveň prebieha filtrovanie tých objektov, ktoré sa dostanú mimo dotazový región (ten je určený len súradnicami, ktoré už boli namapované).

Ďalšie metódy, ktoré vychádzajú z princípov metódy PT, sú napríklad, TLAESA [11] - matica vzdialeností indexovaná ďalšou stromovou štruktúrou, Spaghettis [3] - matica vzdialeností vo forme usporiadaných polí, OMNI family [21] - matica vzdialeností indexovaná R-stromom a PM-strom [18] - hybridná metóda, ktorá používa princípy metódy PT v M-strome.

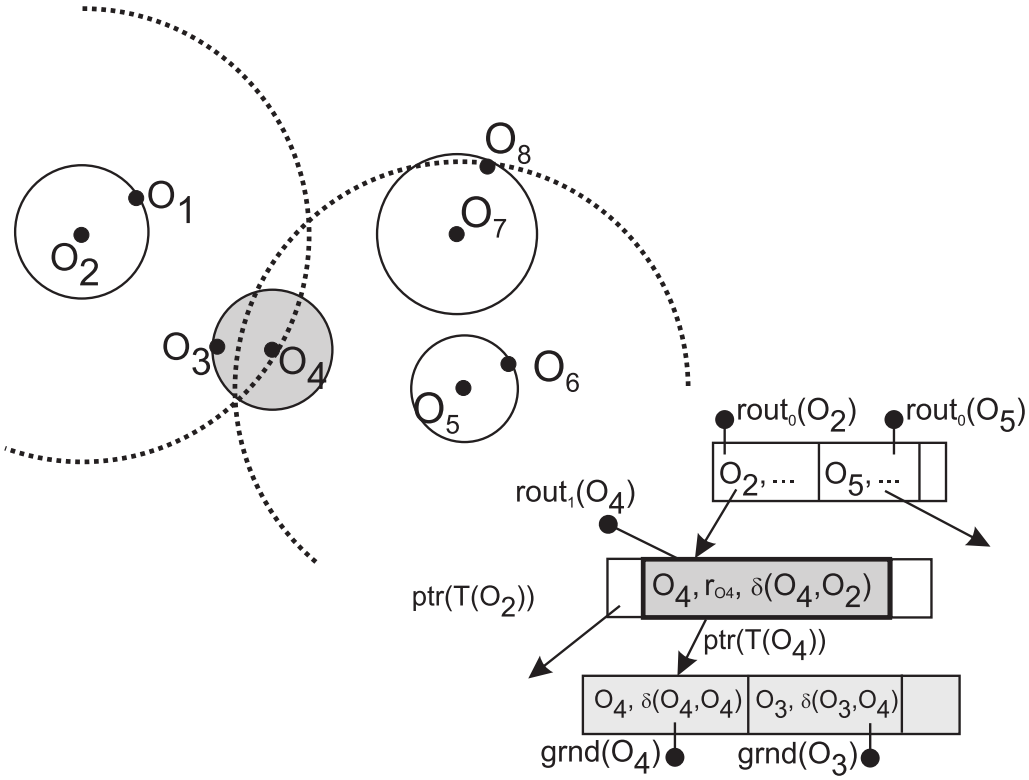
2.2 M-strom

M-strom [6] je dynamická MAM, ktorá vykazuje dobré výsledky pri perzistentnom metrickom indexovaní. Základná myšlienka indexu vychádza zo štruktúry B-stromu a z idey hierarchického rozdelenia priestoru na regióny v R-strome. Index M-stromu tvorí hierarchickú stránkovanú vyváženú štruktúru s databázovými objektmi uloženými v listoch stromu.

2.2.1 Štruktúra indexu

Index M-stromu pozostáva z hyper-guľových regiónov, ktoré zoskupujú podobné objekty v metrickom priestore. Niektoré objekty z databázy sa stávajú stredmi týchto regiónov, polomer regiónu je určený vzdialenosťou k najvzdialenejšiemu objektu v príslušnom regióne. Regióny M-stromu sú hierarchicky rekurzívne usporiadané, rodičovský región obsahuje všetky objekty detských regiónov, atď. Z toho vyplýva, že každý objekt z databázy patrí do nejakého regiónu v rámci celej hierarchie stromu (t.j. na každej hladine). Regióny na jednej hladine sa môžu prekrývať a taktiež rodičovský región nemusí pokrývať celý región potomka. Štruktúra indexu je znázornená na obrázku 2.2.

Vnútorné uzly pozostávajú zo smerovacích záznamov reprezentujúcich regióny v priestore $rout(O_i) = [O_i, r_{O_i}, \delta(O_i, Par(O_i)), ptr(T(O_i))]$, kde $O_i \in \mathbb{S}$ je objekt z databázy reprezentujúci stred regiónu, r_{O_i} je polomer regiónu, $\delta(O_i, Par(O_i))$ je vzdialenosť stredu regiónu O_i k rodičovskému smerovému záznamu (koreňový záznam túto hodnotu neobsahuje) a konečne $ptr(T(O_i))$ je odkaz na podstrom prislúchajúci tomuto smerovému záznamu. Všetky indexované dáta sú uložené v listoch. Listové uzly obsahujú takzvané dátové záznamy $grnd(O_i) = [O_i, \delta(O_i, Par(O_i))]$, kde $O_i \in \mathbb{S}$ je indexovaný databázový objekt a $\delta(O_i, Par(O_i))$ je opäť vzdialenosť objektu O_i k rodičovskému smerovému záznamu. Databázové



Obrázok 2.2: M-strom, hierarchická štruktúra indexu

objekty uložené v listoch reprezentujú región so stredom v príslušnom objekte a s nulovým polomerom.

Uzly M-stromu majú pevne stanovenú kapacitu, listy reprezentujú diskové stránky o pevnej veľkosti. Tu je vhodné si uvedomiť, že nie všetky uzly sú rovnako zaplnené, teda aj skutočný objem dát je v jednotlivých stránkach rozdielny. Kvôli udržaniu vyváženosti štruktúry indexu stanovuje M-strom parameter minimálnej utilizácie pre uzly v strome.

2.2.2 Konštrukcia indexu

Index M-stromu je vytváraný, podobne ako B-strom, štýlom zdola nahor. Proces vkladania do M-stromu začína od koreňa, postupuje k listom a snaží sa nájsť najvhodnejší list, do ktorého by nový objekt mohol byť vložený, s použitím metódy na výber listu. Môže sa stať, že vybraný list je preplnený, v tom prípade je list rozdelený. To môže viesť k reťazovému deleniu rodičovských uzlov (rozdelením listu vzniká u rodičovského uzlu nový smerovací záznam), až po delenie koreňového uzlu, čím sa zvýši celková výška stromu. Pri delení sa najprv vytvorí nový uzol (ako druhý nový uzol sa použije delený uzol), z objektov v delenom uzle sa vyberú 2 objekty, z ktorých sa vytvoria smerové záznamy v rodičovskom uzle - výber vhodných objektov má za úlohu tzv. *promoting policy* a nakoniec sa ostatné objekty deleného uzlu rozdelia do nových uzlov - to má za úlohu tzv. *partitioning policy*. Viac o metódach *partitioning policy* a *promoting policy* informuje článok [6].

Okrem dynamickej metódy konštrukcie indexu vkladaním jednotlivých objektov postupne, existuje aj statická metóda hromadného načítania dát [5, 17].

Snaha hromadne načítať dáta vznikla z dôvodu urýchlenia procesu konštrukcie M-stromu. Takto vytvorený strom je síce rýchlejšie vytvorený, ale oproti dynamicky vytvorenému stromu má menej kompaktnú hierarchiu indexu, regióny sa viac prekrývajú a má väčšie objemy regiónov.

2.2.3 Jednocestný výber listu

Prvotne navrhnutá metóda pre algoritmus vkladania bola metóda jednocestného výber listu [1] [6]. Metóda sleduje jednu cestu od koreňa až do listu. Keďže polomer smerového záznamu je určený vzdialenosťou od stredu príslušného regiónu k najvzdialenejšiemu objektu, každým vložením nového objektu sa tento polomer môže zmeniť. Jednocestný výber listu sa snaží oddialiť zväčšovanie regiónov, preto pri ceste od koreňa do listu je na každej hladine pre pokračovanie zvolený taký smerový záznam, pre ktorý sa vložením nového objektu nezväčší jeho prislúchajúci región. Navyše, ak je takýchto smerových záznamov viac, zvolí sa ten, ktorý má stred regiónu najbližšie k vkladanému objektu.

2.2.4 Viaccestný výber listu

Metóda viaccestného výberu listu [2] [20] vychádza z idey použitej u jednocestnej metódy a to z princípu oddialenia zväčšovania regiónov. Problém jednocestnej metódy je, že snahu minimalizovať zväčšovanie regiónov uvažuje len pre jednu cestu od koreňa do listu. Viaccestný výber listu uvažuje viac ciest, najprv sú zvolený kandidáti pre vloženie nového objektu a to tie listy, ktoré vyhovujú bodovému dotazu, kde za dotazový objekt je dosadený vkladaný objekt. Z kandidátov je vybraný ten list, ktorého rodičovský smerový záznam (stred prislúchajúceho regiónu) je najbližšie k vkladanému objektu. Ak viaccestný výber listu nenašiel žiadneho kandidáta, je použitý klasický jednocestný výber listu. Použitie metódy viaccestného výberu listu síce vedie ku kompaktnejším, menej prekrývajúcim sa regiónom, avšak nároky na tvorbu indexu sú mnohonásobne vyššie ako u jednocestnej metódy.

2.2.5 Metóda vynúteného znovu-vloženia objektov

Pre oddialenie delenia preplnených uzlov bola navrhnutá metóda vynúteného znovu-vloženia objektov [3] [10, 19], metóda vychádza z podobného princípu v R-strome. Ak má dôjsť k deleniu uzlu, metóda vyberie objekty z deleného uzlu a pokúša o ich znovu-vloženie s nádejou, že budú vložené do iného uzlu. Podľa [10] vhodní kandidáti na znovu-vloženie sú hraničné objekty príslušného regiónu. Pri znovu-vložení však môže opäť dôjsť k deleniu uzlu, preto metóda zavádza parameter *recursion depth*, ktorý stanovuje limit rekurzívnych delení. Po dosiahnutí limitu sú zvyšné objekty vložené štandardne s použitím metódy delenia uzlov. Použitie metódy vynúteného znovu-vloženia objektov vedie ku kompaktnejším, menej sa prekrývajúcim regiónom, nároky na tvorbu indexu však sú pomerne nižšie ako u metódy viaccestného výberu listu.

[1] z anglického Single-way leaf selection
[2] z anglického Multi-way leaf selection
[3] z anglického Forced reinserting

2.2.6 Vyhľadávanie v M-strome

Algoritmus vyhľadávania v M-strome začína od koreňa a vstupuje do tých podstromov, ktorých regióny pretínajú dotazový región. Prechod stromom je u rozsahového dotazu implementovaný spôsobom *LIFO* s tým, že sa vstupuje len do relevantných vetví (regiónov). Prekryv dvoch hyper-guľových regiónov je rozhodnutý nasledujúcou vlastnosťou metrík:

Nech $(\mathbb{U}, \delta)$ *je metrický priestor a* $o_i, q \in \mathbb{U}$ *sú body v tomto priestore, potom platí: 2 hyper-guľe* (o_i, r_{oi}) *a* (q, r_q) *sa pretínajú práve vtedy, ak* $\delta(q, o_i) \leq r_q + r_{oi}$.

K-NN dotaz nemá definovaný polomer dotazového regiónu, ten je počítaný dynamicky behom vyhodnocovania dotazu. M-strom pre k-NN dotaz zavádza prioritnú frontu, v ktorej sa pri prechode stromom nachádzajú neodfiltrované uzly. Uzly v prioritnej fronte sú zoradené vzostupne podľa dolného odhadu vzdialenosti dotazového objektu ku všetkým objektom regiónu daného uzlu.

$$d_{min}(T(o_i)) = max\left\{\delta(o_i, q) - r_{oi}, 0\right\} \tag{2.1}$$

Algoritmus si zároveň uchováva kolekciu kandidátov na k najbližších susedov v k-rozmernom poli. V priebehu vyhodnocovania dotazu sa v tejto kolekcii kandidátov objavujú aj celé regióny, nie len dátové záznamy. Pre tieto regióny je zavedený horný odhad vzdialenosti dotazového objektu ku všetkým objektom regiónu daného uzlu.

$$d_{max}(T(o_i)) = \delta(o_i, q) + r_{oi} \tag{2.2}$$

Uzol je odfiltrovaný, ak jeho hodnota $d_{min}(T(o_i))$ je väčšia ako vzdialenosť dotazového objektu k najvzdialenejšiemu kandidátovi na k najbližších susedov, táto hodnota je dynamický polomer dotazového regiónu. U kandidátov - regiónov je za vzdialenosť k dotazovému objektu považovaná hodnota $d_{max}(T(o_i))$ príslušného kandidáta. Po vyhodnotení dotazu je prioritná fronta spracovávaných uzlov prázdna a kolekcia kandidátov obsahuje výsledok dotazu.

3. Metóda Clustered pivot tables

V tejto kapitole popíšeme princípy a implementáciu navrhovanej metódy CPT.

3.1 Motivácia

Na začiatku implementácie metódy CPT bola myšlienka implementovať efektívnu perzistentnú variantu metrickej prístupovej metódy PT. Keďže index metódy PT je zložený z 2 častí (matica vzdialeností a samotné indexované dáta), ponúkajú sa 2 prístupy na uloženie indexu. Prvá možnosť je uložiť tieto časti oddelene do dvoch samostatných súborov v sekundárnej pamäti, druhá alternatíva je mať dáta uložené hneď vedľa pred-vypočítaných vzdialeností a celý index viesť v jednom súbore na disku. U metódy CPT sme sa rozhodli pre prvú možnosť, pretože pri oddelenom uložení je možné načítať do primárnej pamäti len maticu vzdialeností, ktorá je všeobecne menšia, následne pomocou týchto vzdialeností sú odfiltrované nerelevantné objekty a skutočné dáta budú načítané len pre objekty neodfiltrované pomocou prvej časť indexu. Pri druhej alternatíve by do pamäti boli postupne spolu s pred-vypočítanými vzdialenosťami načítané všetky objekty databázy, čím by sa oproti prvej možnosti zvýšil celkový objem načítaných dát.

Metóda PT je jedna z najefektívnejších metrických prístupových metód optimalizovaná na počet DC, v našom návrhu sme chceli túto efektivitu zachovať a navyše optimalizovať perzistentnú metódu PT aj na počet I/O operácií. Tu sme však narazili na problém, pretože usporiadanie dát v indexe metódy PT je náhodné a sekvenčný, alebo dokonca aj aproximatívny prechod maticou, ktorý navrhuje metóda LAESA, by viedol k neprimerane vysokému počtu I/O operácií. To viedlo k myšlienke zapojiť nejakú metódu, ktorá je optimalizovaná na počet I/O operácií. Takouto metódou je M-strom, ktorý vytvára zhluky podobných objektov, tieto zhluky sú stránkované, teda stránka v sekundárnej pamäti obsahuje objekty, ktoré sa nachádzajú v priestore blízko seba. So zapojením M-stromu sú dáta metódou PT indexované v takom poradí, aby sa zachovali zhluky vytvorené pomocou M-stromu. Následne pri vyhľadávaní pomocou takto vytvoreného indexu metódy PT môžu byť odfiltrované celé zhluky a stránka reprezentujúca daný zhluk nemusí byť vôbec načítaná zo sekundárnej pamäte. Metóda CPT tak ponúka implementáciu perzistentnej metódy PT, ktorá nie je optimalizovaná len na počet DC, ale aj na počet I/O operácií.

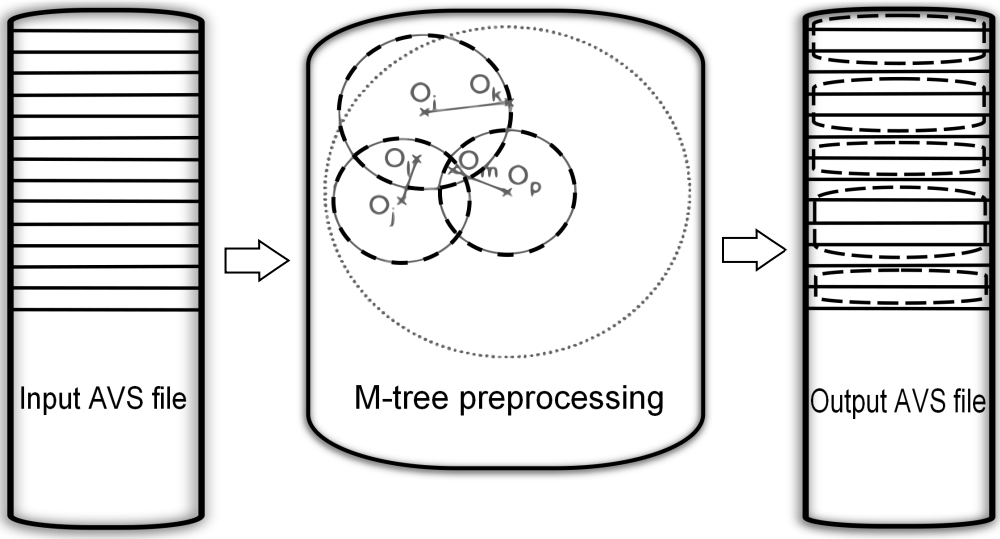
3.2 Pred-spracovanie vstupných dát

Pred-spracovanie dát pomocou M-stromu má esenciálny význam pre optimalizáciu metódy na počet I/O operácií. Dáta, ktoré sa nachádzajú v priestore blízko seba, sú pomocou M-stromu zoskupené a tieto zhluky sú následne indexované metódou PT.

Dáta sú do M-stromu načítané zo vstupného súboru vo formáte AVS 4.1.1 a z nich je následne vytváraný index M-stromu. Keďže implementácia CPT nijako neobmedzuje veľkosť indexovanej databázy, dáta sú zo vstupného súboru načítané a do stromu vkladané po častiach z toho dôvodu, aby sa dáta indexovaných

objektov vošli do primárnej pamäte. Výstupom z procesu pred-spracovania dát je buď výstupný AVS súbor, alebo samotný M-strom.

Jedným z dôvodov prečo bol M-strom zvolený ako štruktúra pre zníženie počtu I/O operácií, bola existencia množstva dynamických techník organizovania dát do kompaktných zhlukov v M-strome. Konštrukcia M-stromu v rámci implementácie navrhovanej metódy CPT je parametrizovaná, je preto možné použiť, tak metódu jednocestného výberu listu 2.2.3, ako aj metódu viaccestného výberu listu 2.2.4. Taktiež pre oddialenie delenia uzlov stromu je možné použiť metódu vynúteného znovu-vloženia objektov 2.2.5. Kvôli vylepšeniu kompaktnosti vytvorených zhlukov by sa v budúcnosti mohli použiť aj nové, prípadne už existujúce techniky konštrukcie M-stromu, ako napríklad slim-down algoritmus. [20].



Obrázok 3.1: Proces predzhlukovania dát pomocou M-stromu

3.3 Výber pivotov

Implementácia metódy CPT ponúka na výber 2 metódy výberu pivotov. Prvou je metóda náhodného výberu pivotov, výber pivotov týmto spôsobom je najrýchlejší a najlacnejší (počet DC je nulový) a výsledky dosahované s takouto kolekciou pivotov sú väčšinou dostatočné. Pre zbežné porovnanie vplyvu rôznych kolekcií pivotov bola implementovaná aj omnoho drahšia (čo do počtu DC) metóda, ktorá bola navrhnutá spolu s metódou LAESA. Metóda sa snaží maximalizovať vzdialeností medzi jednotlivými pivotmi, viac informácií ponúka [12].

3.4 Konštrukcia indexu

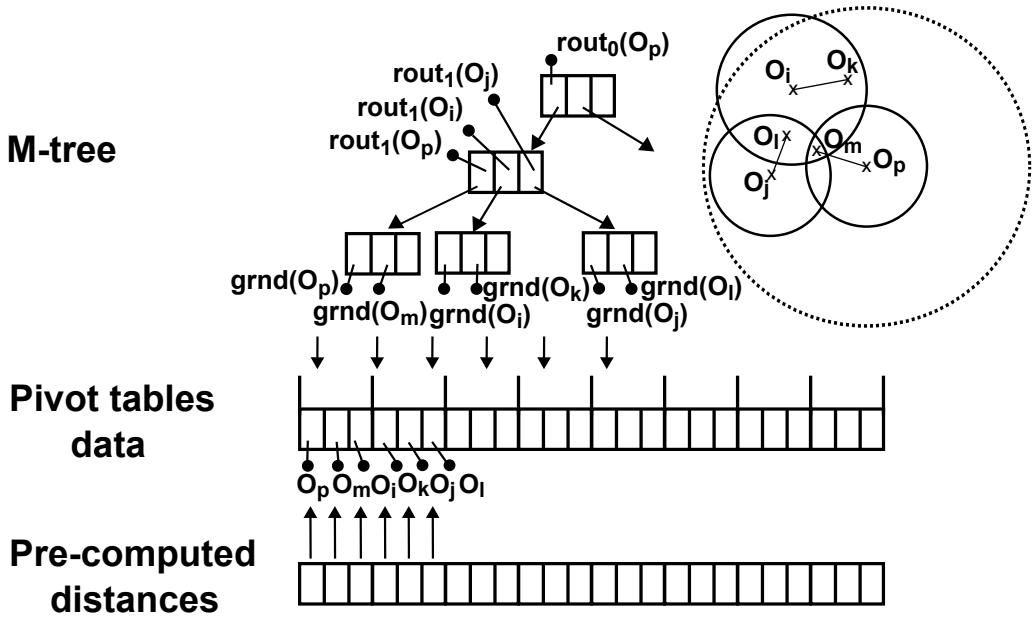
Index metódy CPT pozostáva z dvoch častí, podobne ako je to u štandardnej implementácie metódy PT. Prvú časť tvorí matica vzdialeností medzi pivotmi a objektmi databázy a druhú časť tvoria samotné dáta. Po pred-spracovaní dát sú dáta zaindexované v štruktúre M-stromu, kde sú podobné objekty zoskupené do zhlukov. Keďže metóda CPT reprezentuje perzistentnú implementáciu metódy

PT, pre obe časti indexu je pri indexovaní vytvorený dátový súbor v sekundárnej pamäti. Podľa spôsobu ako sú ďalej implementované obe časti indexu rozlišujeme 2 varianty metódy CPT, statickú a dynamickú.

3.4.1 Statická varianta

Prvotná implementácia metódy CPT predpokladá použitie zhlukovacej metódy len na pred-spracovanie vstupných dát. Po vytvorení M-stromu sú zhlukované dáta, ešte v rámci fázy pred-spracovania dát, uložené do výstupného súboru opäť vo formáte AVS. Fáza pred-spracovania dát v tomto prípade slúži len na preusporiadanie vstupného súboru, štruktúra M-stromu po tejto fáze už nie je potrebná. Proces pred-spracovania dát tak môže byť plne oddelený od ďalších procesov.

Konštrukcia indexu u statickej varianty metódy CPT (obrázok 3.2) postupuje podobne ako konštrukcia indexu klasickej PT, dáta sú ale načítané z preusporiadaného súboru. Pre každý objekt sú spočítané vzdialenosti k jednotlivým pivotom a tie sú uložené do matice vzdialeností, zároveň dáta zo vstupného AVS súboru sú ukladané do sekvenčného dátového súboru, ten je stránkovaný a veľkosť stránky je pevne daná. Vektory vzdialeností jednotlivých databázových objektov sú v matici vzdialeností uložené v rovnakom poradí, v akom sú uložené databázové objekty v dátovej časti indexu CPT.



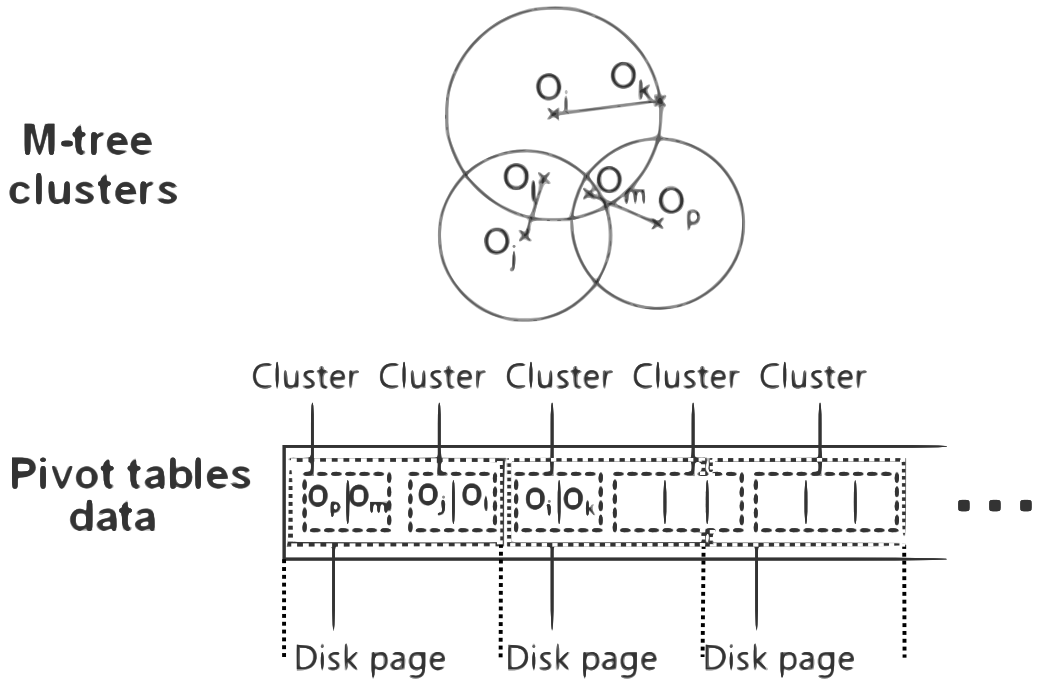
Obrázok 3.2: Statická varianta metódy CPT

Napriek tomu, že ide o statický index, táto varianta metódy CPT bola navrhnutá tak, aby sa implementácii dynamických operácií nekládli žiadne prekážky. Preto v matici vzdialeností je, okrem pred-vypočítaných vzdialeností k pivotom, ku každému objektu uložený aj identifikátor objektu a poradové číslo stránky, v ktorej sa príslušný objekt v dátovej časti indexu CPT nachádza. Proti implementácii dynamických operácií hovorí fakt, že akékoľvek ďalšie vkladanie dát do indexu naruší koncepciu zhlukovaných dát a s tým spojené výhody navrhovanej metódy. Informácia o vytvorených zhlukoch je v indexe uchovaná len vo forme

usporiadania indexovaných dát a len na základe tejto informácie nie je možné novo-vložený objekt zaradiť do správneho zhluku.

Problém stránkovania regiónov

Veľkosť stránky sekvenčného dátového súboru, v ktorom sú uložené dáta indexované metódou CPT, sa zhoduje s veľkosťou stránky (listu) M-stromu. Dáta v listoch M-stromu sú síce stránkované a veľkosť tejto listovej stránky je pevne určená pri konštrukcii indexu M-stromu, napriek tomu konkrétny počet objektov sa v jednotlivých listoch líši, vďaka rôznej zaplnenosti listových regiónov. Tým pádom, hranice stránky dátového súboru indexu CPT nekorešpondujú s hranicami zhlukovaných dát, všeobecne jedna stránka v dátovom súbore indexu CPT môže obsahovať viac zhlukov dát, a dokonca 2 susediace stránky môžu obsahovať dáta z jedného regiónu. Problém potom nastáva pri vyhľadávaní, keď stránka obsahujúca dáta z 2 regiónov nemôže byť odfiltrovaná, hoci jeden z regiónov neprekrýva dotazový región, musí sa ešte rozhodnúť prekryv dotazu s druhým z regiónov. Prekryv stránok a zhlukov je znázornený na obrázku 3.3. Tento problém je vyriešený v dynamickej variante metódy CPT.



Obrázok 3.3: Stránka (Disk page) dátového súboru indexu CPT nekorešponduje s vytvoreným zhlukom (Cluster) dát. Tri stránky môžu napríklad obsahovať päť zhlukov.

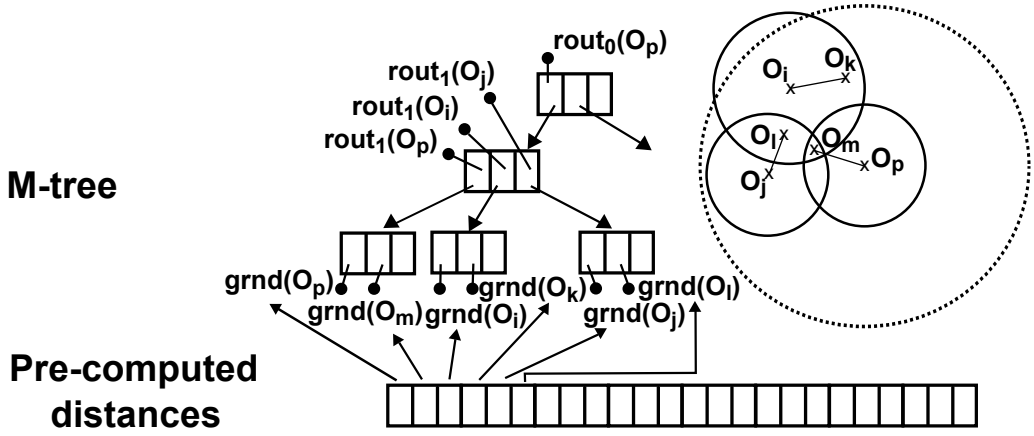
3.4.2 Dynamická varianta

Po narazení na problém stránkovania regiónov pri implementácii indexu metódy CPT sme začali uvažovať o takej implementácii dátovej časti indexu CPT, ktorá by rešpektovala premenlivú veľkosť stránky. Ak by mali stránky premenlivú veľkosť, pre vykonanie operácie *seek* a pre určenie pozície konkrétnej stránky

v databázovom súbore by bolo potrebné uchovávať pozíciu každej stránky v ďalšej štruktúre. Vhodnejšia nám prišla alternatíva využiť už existujúcu štruktúru M-stromu a to tak, že dátová časť indexu CPT bude reprezentovaná priamo listovými stránkami M-stromu. Poradie vektorov vzdialeností jednotlivých databázových objektov v matici vzdialeností taktiež korešponduje s poradím databázových objektov v dátovej časti indexu CPT, podobne ako u statickej varianty.

Je dôležité si uvedomiť, že i keď statická aj dynamická varianta budú indexovať rovnaký počet databázových objektov, počet stránok u dynamickej varianty bude vyšší, čo sa môže prejaviť na výslednom počte I/O operácií. Avšak stránka u dynamickej varianty priamo reprezentuje jeden zhluk dát, takže je predpoklad, že viac stránok bude ešte pred potrebou načítania zo sekundárnej pamäte odfiltrovaných a tým sa zase počet I/O operácií zníži.

Pri dynamickej variante metódy CPT (obrázok 3.4), fáza pred-spracovania dát neukladá dáta M-stromu do výstupného AVS súboru, namiesto toho výstupom tejto fázy je samotný index M-stromu. Proces konštrukcie indexu pokračuje spočítaním a uložením vzdialeností medzi objektmi databázy a pivotmi. Tak ako u statickej varianty aj tu je ku každému databázovému objektu, spolu s vektorom vzdialeností k pivotom, uložený aj identifikátor daného objektu a číslo stránky (teda listu v M-strome), v ktorej sa príslušný objekt nachádza.



Obrázok 3.4: Dynamická varianta metódy CPT

Priamym napojením CPT na M-strom sa nielenže vyriešil problém nekorešpondujúcich databázových stránok s vytvorenými zhlukmi dát, ale pri zachovaní celého indexu M-stromu sa zo statickej metrickej prístupovej metódy v tomto momente stala dynamická metrická prístupová metóda. I keď v súčasnom návrhu metódy CPT nie sú implementované dynamické operácie vkladania a mazania objektov z databázy, implementácii týchto operácií v budúcnosti nič nebráni. Operácia vkladania dát by vložila príslušný objekt do M-stromu, následne by sa spočítali vzdialenosti tohoto objektu k jednotlivým pivotom a tie by sa uložili do prvej časti indexu CPT (pred-vypočítané vzdialenosti) spolu s číslom databázovej stránky (listu v M-strome), v ktorej sa daný objekt nachádza. Vloženie objektu takýmto spôsobom nenaruší koncepciu zhlukovaných dát a preto index ani po niekoľkých operáciách vloženia, prípadne zmazania dát nestráca svoje výhody, ktoré vznikli predzhlukovaním dát.

3.4.3 Jednoúrovňový index

Pre indexovanie menších kolekcií metóda CPT ponúka jednoúrovňový index ako ďalšiu optimalizáciu počtu I/O operácií. Ako už názov napovedá ide o spojenie oboch častí indexu, matice vzdialeností a samotných dát do jednej dátovej štruktúry. Po pred-spracovaní dát pomocou M-stromu a následnom vytvorení zhlukovaného AVS súboru sú dáta z tohto súboru načítané a zaindexované pomocou metódy CPT. Keďže sa dáta a pred-vypočítané vzdialenosti nachádzajú v jednom súbore, teda M-strom už nie je po fáze pred-spracovania dát potrebný, je jasné, že ide o modifikáciu statickej varianty metódy CPT. Každý objekt v indexe je reprezentovaný identifikátorom objektu, vektorom vzdialeností k pivotom a dátovou reprezentáciou príslušného objektu.

Stránkovanie jednoúrovňového indexu vychádza z iných princípov ako stránkovanie štandardného dvojúrovňového indexu. U dvojúrovňového indexu je prvá časť indexu (pred-vypočítané vzdialenosti) nahratá celá, alebo aspoň jej veľká časť do primárnej pamäti, aby sa minimalizoval počet I/O operácií pri filtrovaní objektov. Databázové stránky s neodfiltrovanými objektmi sú následne načítané samostatne podľa potreby. Jednoúrovňový index spája obe časti do jednej, kvôli efektivite filtrovania je preto do primárnej pamäti načítaný veľký blok podobne ako pre pred-vypočítané vzdialenosti u dvojúrovňového indexu.

Keďže vo fáze filtrovania sa spolu s pred-vypočítanými vzdialenosťami musia načítať aj samotné objekty, počet objektov (resp. ich vzdialeností k pivotom) načítaných pri filtrovaní je menší ako tento počet v bloku rovnakej veľkosti vo fáze filtrovania u dvojúrovňového indexu. Pri predpoklade, že sa celý index nevojde do primárnej pamäti, to vedie k zvýšeniu počtu I/O operácií. U jednoúrovňového indexu sa však už v priebehu filtrovania uchovávajú samotné dáta neodfiltrovaných objektov a tie sa potom nemusia načítať v konečnej fáze vyhľadávania. Počet priebežne neodfiltrovaných objektov sa nedá odhadnúť, preto jednoúrovňový index stanovuje limit pre počet týchto uchovaných objektov. Po naplnení limitu sa tieto objekty musia dodatočne načítať v konečnej fáze dofiltrovania na základe skutočnej vzdialenosti.

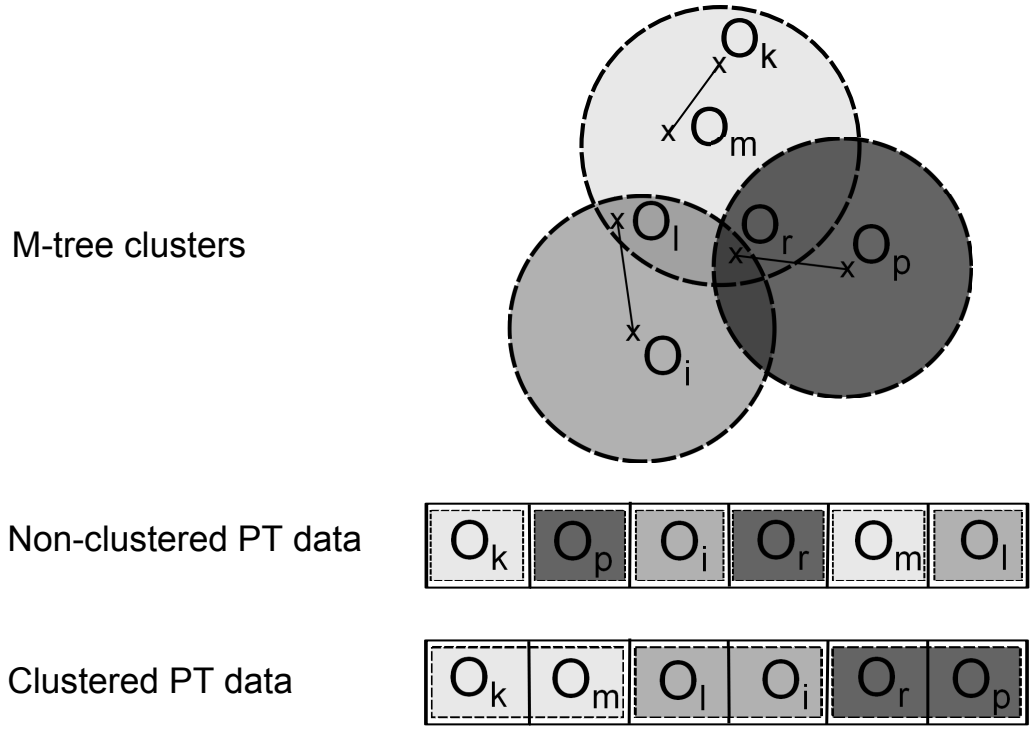
3.5 Vyhľadávanie

Po skonštruovaní indexu je metóda CPT pripravená na vyhľadávanie dát. Vyhľadávanie vychádza zo základných princípov metódy PT, na základe dolného odhadu skutočnej vzdialenosti k dotazovému objektu sú odfiltrované databázové objekty, ktoré sa nachádzajú mimo región vymedzený dotazovým objektom a polomerom dotazu. Z neodfiltrovaných objektov sú následne vyradené tie databázové objekty, ktorých skutočná vzdialenosť k dotazovému objektu nie je väčšia ako polomer dotazu.

Prvotná filtrácia nevyžaduje znalosť konkrétnych dát, lacno vypočítateľný dolný odhad skutočnej vzdialenosti (vzťah 1.1) je odvodený z uložených pred-vypočítaných vzdialeností databázových objektov k pivotom a zo vzdialeností dotazového objektu k pivotom. Skutočná draho vypočítateľná vzdialenosť je následne počítaná len pre neodfiltrované objekty. Až v tejto fáze sú načítané samotné databázové objekty, prvotná filtrácia má okrem významu zníženia počtu DC ďalší význam pri implementácií perzistencie u CPT a to zníženie počtu načítaných

stránok zo sekundárnej pamäti.

Hlavná výhoda navrhovanej metódy CPT sa prejaví práve pri počte stránok načítaných z databázového súboru. Stránka databázového súboru v sekundárnej pamäti reprezentuje jeden, prípadne viac zhlukov podobných objektov. Ak (žiaden) zhluk z tejto stránky neprekrýva región dotazu, všetky objekty v danej stránke sú vo fáze filtrácie odfiltrované a stránka obsahujúce tieto objekty nebude následne načítaná zo sekundárnej pamäti vo fáze dofiltrovania objektov na základe skutočnej vzdialenosti. Ak by boli dáta v databázovom súbore zoradené náhodne (obrázok 3.5), pravdepodobnosť, že všetky objekty nejakej stránky budú odfiltrované v prvej fáze, by bola oveľa nižšia.



Obrázok 3.5: Porovnanie usporiadania databázových objektov v indexoch klasickej PT a navrhovanej CPT

3.5.1 Rozsahový dotaz

Metóda CPT definuje pre rozsahový dotaz okrem štandardných parametrov (dotazový objekt a polomer dotazu) ďalší parameter, a to počet pivotov pomocou ktorých sa filtruje v jednom filtrovacom cykle. Proces vyhľadávania začne s výberom podmnožiny pivotov a spočítaním súradníc vektoru namapovaného dotazového objektu do priestoru pivotov, spočítané sú len tie súradnice, ktoré prislúchajú vybranej podmnožine pivotov. Následne sa filtruje, pre vyhodnotenie dolného odhadu vzdialenosti sa uvažujú opäť len spočítané súradnice vektoru namapovaného dotazového objektu.

Proces filtrovania je tak rozdelení na niekoľko filtrovacích cyklov, v prvom cykle sa zo všetkých databázových objektov vyberie kolekcia relevantných objektov s použitím prvej časti pivotov, druhý cyklus z tejto kolekcie vyradí ďalšie

objekty a aktualizuje ich dolné odhady vzdialeností k dotazovému objektu s použitím ďalšej časti pivotov, atď. až kým nie sú použité všetky pivoty. Tento prístup optimalizuje výpočty potrebné k spočítaniu dolného odhadu vzdialenosti objektu k dotazovému objektu, pre objekt odfiltrovaný v i-tom cykle sú pre výpočet dolného odhadu potrebné len súradnice vektorov vzdialeností odpovedajúce pivotom z prvého až i-teho cyklu, ostatné súradnice pre výpočet neboli použité. Takéto rozdelenie filtrovacej fázy má však zmysel až pri veľkom počte pivotov, kedy vektory vzdialeností k pivotom sú netriviálne veľké.

Objekty sú nakoniec dofiltrované pomocou skutočnej vzdialenosti databázového objektu k dotazovému objektu.

3.5.2 Dotaz na najbližších k susedov

U k-NN dotazu nie je polomer dotazu vopred známy, preto sa v priebehu filtrovania polomer dotazu dynamicky dopočítava. V prvej fáze vyhodnocovania k-NN dotazu sa spočítajú dolné odhady vzdialeností k dotazovému objektu pre všetky objekty databázy. Druhá fáza v sebe zahŕňa počítanie dynamického polomeru, filtrovanie na základe dolného odhadu vzdialenosti a zároveň aj samotné dofiltrovanie na základe skutočnej vzdialenosti.

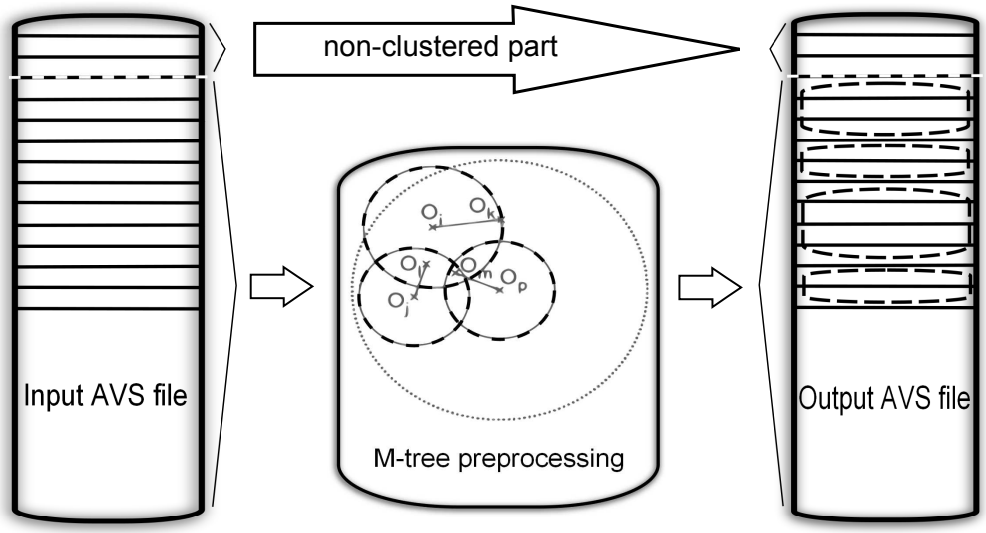
Po spočítaní dolných odhadov vzdialeností k dotazovému objektu môžu byť tieto vzdialeností pred samotným filtrovaním ešte preusporiadané. Ponúkajú sa 2 možnosti ďalšieho spracovania týchto dolných odhadov, prvá možnosť je spracovávať spočítané dolné odhady vzdialeností bez ďalšieho preusporiadania, teda v rovnakom poradí ako sú usporiadané objekty v databázovom súbore. Druhý, štandardný prístup je preusporiadať tieto vzdialeností vzostupne, podobne ako navrhuje metóda LAESA [12] (pozri kapitolu 2.1.3). Metóda CPT sa snaží optimalizovať počet diskových prístupov a pri štandardnom preusporiadanom spracovávaní dolných odhadov vzdialeností by sa každá databázová stránka pravdepodobne musela načítať z disku (bez absencie *cache-e*) viackrát. Dôvodom je, že po preusporiadaní objektov spôsobom aký navrhuje metóda LAESA, metóda CPT príde o jedinú informáciu o vytvorených zhlukoch dát, ktorá je uložená práve v poradí databázových objektov. Keďže tým by sa z metódy CPT stala štandardná metóda PT (neoptimalizovaná na počet I/O operácií), dolné odhady musia byť v ďalšej fáze spracovávané v rovnakom poradí, v akom boli objekty metódou CPT zaindexované. Napriek tomu, metóda CPT ponúka aj implementáciu k-NN dotazu s preusporiadaním dolných odhadov, výhody tejto metódy (zníženie počtu DC) sa môžu prejaviť so zavedením *cache-e* pre stránky databázového súboru.

Druhá fáza začína inicializáciou kolekcie najbližších susedov - spočítaním skutočných vzdialeností k dotazovému objektu pre prvých k objektov databázy. Dynamický polomer dotazu je nastavený na vzdialenosť najvzdialenejšieho objektu z kolekcie najbližších susedov. Každý ďalší objekt je buď odfiltrovaný na základe dolného odhadu vzdialenosti k dotazovému objektu, alebo, v opačnom prípade, je pre neho spočítaná skutočná vzdialenosť k dotazu. Ak je skutočná vzdialenosť menšia ako dynamický polomer, daný objekt nahradí najvzdialenejší objekt v kolekcii najbližších susedov a dynamický polomer dotazu je opäť aktualizovaný na vzdialenosť k najvzdialenejšiemu susedovi. V opačnom prípade je daný objekt odfiltrovaný na základe skutočnej vzdialenosti k dotazu. Algoritmus končí po spracovaní všetkých databázových objektov.

3.5.3 Heuristika pre k-NN dotaz

Efektivita filtrovania u k-NN dotazu závisí na tom, ako rýchlo sa dynamický polomer dotazu priblíži tomu skutočnému polomeru. Keďže dynamický polomer je odvodený od najvzdialenejšieho z kandidátov na najbližšieho suseda, inicializácia a následná aktualizácia kolekcie kandidátov na najbližšieho suseda môže ovplyvniť efektivitu celého algoritmu. Skutočnosť, že sú dáta spracovávané v zhlukovanom poradí, vedie k tomu, že dynamicky polomer aproximuje ten skutočný oveľa neskôr v prípade, že regióny zo začiatku indexu sa nachádzajú v priestore ďaleko od dotazového regiónu. Inicializácia a následné aktualizácie kandidátov na najbližších susedov tak nevedú k rýchlemu zmenšeniu dynamického polomeru a dotazový región tak pokrýva veľa, v skutočnosti nerelevantných, objektov.

Metóda CPT preto zavádza heuristiku pre rýchlu aproximáciu skutočného polomeru. Rýchlu aproximáciu ovplyvňujú vytvorené zhluky podobných objektov na začiatku databázového súboru, preto táto heuristika rozdeľuje index (predvypočítané vzdialenosti aj samotné dáta) metódy CPT na 2 logické časti (obrázok 3.6). Objekty v druhej logickej časti sú indexované už popísaným spôsobom s pred-zhlukovaním dát pomocou M-stromu. Prvá logická časť indexu je však vytvorená z náhodne usporiadaných objektov, v takom poradí v akom boli vo vstupnom súbore (pred pred-spracovaním pomocou M-stromu). Veľkosť tejto nezhlukovanej časti je určená parametrom metódy CPT.



Obrázok 3.6: Proces vytvorenia indexu s použitím heuristiky pre optimalizovanie k-NN dotazu

Algoritmus 3.1 Rozsahový dotaz

Vstup: $q \in \mathbb{S}$, $radius \in \mathbb{R}$
Výstup: $objectsWithinRange$

```
1: isFirstCycle ← true
2: while exists not used pivot do
3:    {VONKAJŠÍ CYKLUS}
4:    currPivots ← selectNextPivots() {vektor vzdialeností dotazového objektu
      je dopočítaný v tých súradniciach príslušných aktuálnej podmnožine pivo-
      tov}
5:    computeQueryPivotsDistances(q, currPivots, qDist output)
6:    {Prvý vonkajší cyklus reprezentuje sekvenčný prechod maticou vzdialenos-
      tí, v ďalšom cykle sa prechádzajú už len relevantné objekty}
7:    while isFirstCycle or exists not processed object within range do
8:       {VNÚTORNÝ CYKLUS}
9:       {výpočet dolného odhadu skutočnej vzdialenosti sa počíta len pre už
         spočítané súradnice dotazového vektoru vzdialeností}
10:      Lmax ← computeLowerBound(currObjDist, qDist)
11:      if Lmax ≤ radius then
12:         if isFirstCycle then
13:            {v prvom vonkajšom cykle sa každý relevantný objekt pridá do vý-
               sledku}
14:            objectsWithinRange.add(currObjDist, Lmax)
15:         else
16:            {v ďalšom vonkajšom cykle sa aktualizuje dolný odhad vzdialenosti
               relevantného objektu}
17:            objectsWithinRange.update(currObjDist, Lmax)
18:         end if
19:      end if
20:      if Lmax > radius and not(isFirstCycle) then
21:         {objekt mimo dotazový región je z výsledku odobraný}
22:         objectsWithinRange.remove(currObjDist)
23:      end if
24:   end while
25:   isFirstCycle ← false
26: end while
27: {dofiltrovanie podľa skutočnej vzdialenosti}
28: for all currObjDist in objectsWithinRange do
29:   {získanie dát samotného objektu zo stránky v sekundárnej pamäti}
30:   currObj ← getObjectFromDatabase(objId, pageId)
31:   {spočítanie skutočnej vzdialeností medzi objektom z databázy a dotazovým
      objektom}
32:   realDist ← computeRealDistance(currObj, q)
33:   if realDist > radius then
34:      {objekt mimo dotazový región je z výsledku odobraný}
35:      objectsWithinRange.remove(currObjDist)
36:   end if
37: end for
```

Algoritmus 3.2 k-NN dotaz

Vstup: $q \in \mathbb{S}$, $k \in \mathbb{N}$, $orderType \in \{NotOrder, OrderByLowerBound\}$
Výstup: *closestObjects*

```
1: {spočítanie vektoru vzdialeností dotazového objektu ku všetkým pivotom}
2: computeQueryPivotsDistances(q, globalPivots, qDist output)
3: {Sekvenčný prechod maticou vzdialeností}
4: for all currObjDist in distanceMatrix do
5:    {výpočet dolného odhadu skutočnej vzdialenosti databázového objektu a
      dotazového objektu}
6:    Lmax ← computeLowerBound(currObjDist, qDist)
7:    {dolný odhad vzdialenosti je uložený do kolekcie relevantných objektov}
8:    objectsWithinRange.add(currObjDist, Lmax)
9: end for
10: if orderType = OrderByLowerBound then
11:    {preusporiadanie kolekcie relevantných objektov zostupne podľa dolného
       odhadu vzdialenosti}
12:    objectsWithinRange.sortByLowerBound()
13: end if
14: {inicializácia dynamického polomeru dotazu}
15: d_kMax ← ∞
16: {dofiltrovanie podľa skutočnej vzdialenosti}
17: for all currObjDist in objectsWithinRange do
18:    if closestObjects.count = k andcurrObjDist.Lmax > d_kMax then
19:       if orderType = OrderByLowerBound then
20:          {najbližších k susedov bolo nájdených}
21:          return
22:       else
23:          {aktuálny objekt je mimo dynamický dotazový región}
24:          continue
25:       end if
26:    end if
27:    {získanie dát samotného objektu zo stránky v sekundárnej pamäti}
28:    currObj ← getObjectFromDatabase(objId, pageId)
29:    {spočítanie skutočnej vzdialeností medzi objektom z databázy a dotazovým
       objektom}
30:    realDist ← computeRealDistance(currObj, q)
31:    if closestObjects.count < k orrealDist < d_kMax then
32:       {aktualizácia kolekcie najbližších k susedov}
33:       closestObjects.update(currObj, realDist)
34:       {aktualizácia dynamického polomeru dotazu}
35:       d_kMax ← closestObjects.distanceOfFurthestObject
36:    end if
37: end for
```

4. Implementácia

Pre implementáciu navrhovanej metódy CPT bol zvolený jazyk C++ z dôvodu vysokej rýchlosti pri spracovávaní časovo náročných operácií. Ďalším dôvodom bolo jednoduché napojenie na použitú implementáciu M-stromu, pre ktorú bol taktiež použitý jazyk C++.

4.1 Použité implementácie

Metóda CPT bola implementovaná s použitím *framework-u* ATOM (Amphora Tree Object Model) [1]. Nad týmto *framework-om* je naimplementovaný aj použitý M-strom [2], ako súčasť implementácie PM-stromu [18].

Datové štruktúry použité z *framework-u* ATOM a z použitej implementácie M-stromu:

- `cPMTree` - Implementácia PM-stromu vo *framework-u* ATOM. Bez použitia globálnych pivotov ide o implementáciu M-stromu, ktorá bola použitá pri implementácii navrhnutej metódy CPT.
- `cDistance` - Reprezentuje funkciu vzdialenosti. Ide o abstraktnú štruktúru, implementácie konkrétnych použitých metrík vychádzajú z tohto dátového typu.
- `cDataObject` - Reprezentuje indexovaný databázový objekt, použitý v M-strome aj v novej metóde CPT.
- `cArray<T>` - Implementácia dynamického poľa. Ide o generický typ, použitý na viacerých miestach implementácie metódy CPT, najčastejšie ako kolekcia databázových objektov - `cArray<cDataObject>`.
- `cMetricScanner` - Implementuje metódu na načítanie dát zo vstupného AVS súboru do pamäťovej štruktúry.

Ako základ pre implementáciu dátovej štruktúry usporiadaného spojového zoznamu `cSortedLinkedList` bol použitý algoritmus *Mergesort For Linked List* a jeho implementácia v jazyku C [3].

4.1.1 Formát vstupných dát

Navrhnutá implementácia CPT prijíma vstupné dáta vo formáte textového AVS (ATOM Vector Set) súboru, ktorý obsahuje hlavičku a samotné vektorové dáta. Formát AVS súboru:

- Hlavička
  - $Count: *hodnota* - Počet objektov v dátovej sade.

[1] vyvinutý výskumnou skupinou Amphora Research Group, http://arg.vsb.cz
[2] vyvinutý výskumnou skupinou Siret Research Group, http://siret.ms.mff.cuni.cz
[3] vyvinul Simon Tatham,
http://www.chiark.greenend.org.uk/ sgtatham/algorithms/listsort.html

  - $Dimension: *hodnota* - Veľkosť dimenzie jedného objektu dátovej sady.
  - $ValueType: *hodnota* - Typ súradnice (float, int, string) jedného dátového objektu v dátovej sade.
  - DATA - Označenie konca hlavičky a začiatku vektorových dát.

- Vektorové dáta - Jednotlivé súradnice vektorov sú oddelené medzerou, alebo čiarkou, vektory sú oddelené novým riadkom.

4.2 Základné dátové štruktúry

- `cPivotTables` - Reprezentuje abstraktné rozhranie pre implementáciu perzistentnej metódy PT.
- `cLAESA` - Reprezentuje navrhovanú metódu CPT, ponúka implementáciu abstraktného rozhrania `cPivotTables`.
- `cPivotSelector` - Reprezentuje abstraktné rozhranie pre implementáciu metódy výberu pivotov. Navrhované riešenie ponúka 2 implementácie tejto abstrakcie, metóda náhodného výberu pivotov - `cRandomPivotSelector` a metóda výberu maximálne oddelených pivotov - `cLAESAPivotSelector`.
- `cPivotIndex` - Reprezentuje prvú časť (pred-vypočítané vzdialenosti k pivotom) indexu metódy CPT, súčasťou je štruktúra `cPivotIndexHeader`, ktorá obsahuje informácie o tejto časti indexu metódy CPT.
- `cDatabase` - Reprezentuje dátovú časť indexu metódy CPT, súčasťou je štruktúra `cDatabaseHeader`, ktorá obsahuje informácie o indexovaných databázových objektoch.
- `eIndexArchitecture` - Vymenovaný typ, určuje typ indexu metódy CPT, nadobúda hodnoty: jednoúrovňový index - `eSingleLayer`, dvojúrovňový index statickej varianty CPT - `eTwoLayer`, alebo index dynamickej varianty CPT - `eTwoLayerOverPMTree`.
- `cPMTreeSort` - Zabezpečuje fázu pred-spracovania dát pomocou M-stromu.
- `cLinkedList` - Implementácia obojsmerného spojového zoznamu. Pomocná dátová štruktúra, používa sa hlavne pri vyhľadávaní na uchovanie aktuálnych objektov v dosahu dotazového regiónu. Jej usporiadaná varianta `cSortedLinkedList` sa používa na uloženie kolekcie kandidátov na najbližších k susedov.
- `cStoredDistance` - Dátová štruktúra použitá pri vyhľadávaní na uloženie informácií o neodfiltrovanom objekte. Okrem identifikátora objektu je tu uložené aj číslo databázovej stránky v ktorej sa objekt nachádza, dolný odhad skutočnej vzdialenosti k dotazovému objektu, prípadne aj samotná skutočná vzdialenosť.
- `cPTQueryResult` - Reprezentuje výsledok dotazu, uchováva samotnú kolekciu relevantných objektov a taktiež aj štatistiky príslušného dotazu. Jeden relevantný objekt z kolekcie reprezentuje `cPTQueryResultObject`.

4.2.1 Implementácia perzistencie

Keďže aktuálny návrh metódy CPT nezahŕňa implementáciu *cache-e*, perzistencia v CPT bola navrhnutá tak, aby sa v budúcnosti mohla metóda rozšíriť o použitie `cache-e` pre stránky zo sekundárnej pamäti. Základnou štruktúrou pre prácu so sekundárnou pamäťou je generická abstraktná trieda `cPTCachePage<T>`, ktorá reprezentuje jednu pamäťovú stránku. Táto trieda dokáže načítať a zapisovať stránky na disk priamo, avšak práca so sekundárnou pamäťou je možná aj pomocou akejsi *pseudocache-e* - triedy `cPTCache<T>`. Trieda `cPTCache<T>` ponúka základnú implementáciu *cache-e*, kde v prvku triedy `lastReadPage` je vždy uchovaná posledná stránka (typu `cPTCachePage<T>`), načítaná skrz túto triedu.

Metóda CPT ponúka štyri reprezentácie stránky sekundárnej pamäti - špecializácie generickej triedy `cPTCachePage<T>`:

- `cDatabasePage` - Reprezentuje stránku z databázového súboru u statickej varianty metódy CPT. Jeden objekt stránky je v primárnej pamäti reprezentovaný objektom typu `cDataObject`. Každá stránka obsahuje hlavičku typu `cDatabasePageHeader` s informáciami o danej stránke.

- `cTreePage` - Reprezentuje stránku z databázového súboru u dynamickej varianty metódy CPT. Keďže u dynamickej varianty je stránka už obsiahnutá v M-strome, ide o *wrapper* listu z M-stromu. Jeden objekt stránky je v primárnej pamäti reprezentovaný objektom typu `cDataObject`.

- `cPivotIndexPage` - Reprezentuje stránku zo súboru s pred-vypočítanými vzdialenosťami u dvojúrovňového indexu. Jeden objekt stránky je v primárnej pamäti reprezentovaný objektom typu `cPivotIndexObject`, ktorý okrem samotného vektoru vzdialeností k pivotom obsahuje aj identifikátor objektu a číslo databázovej stránky v ktorej sa daný objekt nachádza. Každá stránka obsahuje hlavičku typu `cPivotIndexPageHeader` s informáciami o danej stránke.

- `cPivotIndexAndDataPage` - Reprezentuje stránku sekundárnej pamäti zo súboru s pred-vypočítanými vzdialenosťami u jednoúrovňového indexu, teda obsahuje aj samotné databázové objekty. Implementácia stránky indexu vychádza z typu `cPivotIndexPage`. Jeden objekt stránky je v primárnej pamäti reprezentovaný objektom typu `cPivotIndexAndDatObject`, ktorý v sebe zaobaľuje objekty `cDataObject` a `cPivotIndexObject`.

5. Experimenty

Otestovanie navrhovanej metódy CPT sme vykonali na štyroch rôznych metódach konštrukcie M-stromu. Pre takto vytvorený M-strom bol zrealizovaný test vyhľadávania pre dve rôzne implementácie indexu - statickú a dynamickú variantu metódy CPT. Podľa výsledkov prvej fázy testovania sme zo štyroch testovaných vybrali jednu metódu konštrukcie M-stromu a vykonali sme sériu porovnávacích testov navrhovanej metódy CPT (pred-zhlukovanou M-stromom, ktorý bol skonštruovaný vybranou metódou) a priamej (bez fázy pred-zhlukovania dát) implementácie perzistencie u metódy PT. Ak opomenieme absenciu fázy pred-zhlukovania dát pomocou M-stromu, tak priama implementácia metódy PT, s ktorou bola metóda CPT porovnávaná, je v podstate identická so statickou variantou metódy CPT, ktorá má databázu uloženú v sekvenčnom dátovom súbore s pevnou veľkosťou stránky.

Tak ako už bolo niekoľkokrát spomenuté, navrhovaná metóda CPT je zameraná na minimalizáciu počtu I/O operácií, preto jediný testovaný parameter prezentovaný vo výsledkoch je práve počet načítaných diskových stránok. Počet DC pri rozsahovom dotazu je vždy rovnaký pre všetky testované varianty, zanedbateľné navýšenie počtu DC je u k-NN dotazu zapríčinené absenciou fázy zotriedenia dolných odhadov vzdialeností. Použitie zotriedenia dolných odhadov vzdialeností síce vykazovalo približne dvakrát menší počet DC, ale až päťkrát väčší počet I/O operácií.

5.1 Dátové sady

Testy boli vykonávané nad dvoma reálnymi dátovými sadami (Cophir, ColorHistogram) a dvoma syntetickými dátovými sadami (PolygonSet a Cloud). Pre oba typy, reálnu aj syntetickú sadu sme zvolili dáta s vyššou aj s nižšou dimenziou.

Za prvú reálnu sadu bola zvolená podmnožina databázy *CoPhIR* [1], ktorá pozostáva z 1,000,000 celočíselných vektorov vlastností, reprezentujúcich 64-dimenzionálnu štruktúru farieb z objektov uložených vo formáte MPEG-7. Ako funkcia vzdialenosti bola pre túto dátovú sadu použitá Eukleidovská (L2) metrika.

Druhá reálna dátová sada, ColorHistogram, je podmnožinou databázy *Corel* [8] a obsahuje 68,040 32-dimenzionálnych vektorov, ktoré reprezentujú histogramy farieb obrazových dát. Taktiež pre túto dátovú sadu bola zvolená Eukleidovská (L2) metrika, súradnice vektorov sú narozdiel od prvej dátovej sady reálne čísla.

PolygonSet je syntetická databáza pozostávajúca z 250,000 dvojrozmerných polygónov, každý z polygónov obsahuje 5-15 vrcholov. Druhá syntetická databáza, Cloud, obsahuje 110,000 oblakov 60 šesť-dimenzionálnych bodov, kde všetky body sú z priestoru vymedzenom šesť-dimenzionálnou kockou. Prvý bod, stred oblaku bol vygenerovaný náhodne a zvyšok bodov v oblaku bol vygenerovaný okolo tohto stredu so zachovaním normálneho rozdelenia. Tieto syntetické dátové sady boli zvolené ako alternatíva k vektorovým sadám s normálnym rozdelením. Obe tieto sady používajú Hausdorfovu metriku ako funkciu vzdialenosti.

Vo vykonaných experimentoch sú v prípade dátových sád ColorHistogram a Cloud použité všetky dáta a náhodná podmnožina o veľkosti 150,000 objektov

z dátových sád PolygonSet a Cophir. Každý jeden test bol vykonaný s použitím 100 dotazových objektov a výsledky testu boli spriemerované. Dotazové objekty sú distribuované v priestore príslušnej dátovej sady (podľa jej pravdepodobnostného rozdelenia), avšak dotazové objekty sa nenachádzajú v testovanej kolekcii objektov.

5.2 Parametre testov

V prvej fáze testovania boli porovnávané výsledky metód štyroch variant konštrukcie M-stromu. Skúmali sme metódy jednocestného výberu listu, jednocestného výberu listu s použitím vynúteného znovu-vloženia dát, viaccestného výberu listu a ako posledná alternatíva bola uvažovaná metóda viaccestného výberu listu s použitím vynúteného znovu-vloženia dát. Za metódy, použité pri delení uzlov v M-strome, boli zvolené jedny zo základných metód navrhnutých spolu s návrhom M-stromu [6], ako *promoting policy* bola zvolená metóda `mM_RAD` a za *partitioning policy* bola vybraná metóda `Generalized Hyperplane`. Metóda vynúteného znovu-vloženia zavádza ďalšie parametre pre M-strom, maximálna hĺbka rekurzie bola nastavená na hodnotu 10 a maximálny počet znovu-vkladaných objektov pri jednom znovu-vložení bol stanovený na 5. Minimálna utilizácia objektov v uzle M-stromu bola u všetkých testov nastavená na 20%.

Keďže sme chceli porovnať navrhovanú metódu CPT oproti priamej perzistentnej metóde PT, všetky metódy sme testovali pre rovnako veľkú databázovú stránku. Veľkosť databázovej stránky bola pre každú dátovú sadu prispôsobená tak, aby pri úplnom zaplnený obsahovala približne 32 dátových objektov, tzn. pre ColorHistogram a PolygonSet mala stránka veľkosť 4 kB, pre Cophir 8 kB a pre Cloud až 44kB. Testovali sme však aj správanie sa metód na rôznych veľkostiach stránky, v tomto teste počet objektov v stránke varioval v rozmedzí 16 - 64. Pokiaľ nie je naznačené inak, pre testy bola použitá náhodne vybraná kolekcia pivotov. Počet pivotov bol stanovený na 30 pivotov, pri teste s variabilným počtom pivotov, sa počet pivotov menil v rozsahu 10 - 90. Počet pivotov pre jeden filtrovací cyklus rozsahového dotazu (viac v sekcii 3.5.1) bol pri testoch stanovený na 30. Selektivita k-NN dotazu bola nastavená na 1, prípade 10 objektov, podľa výsledného dynamického polomeru z k-NN dotazu bol nastavený polomer pre rozsahový dotaz, aby bol rozsahový dotaz testovaný s približne rovnakou selektivitou (rozsahový dotaz môže pri rovnakom konečnom polomere dotazu vrátiť viac objektov ako k-NN dotaz). V teste, kde sa skúma správanie metód pri variabilnom počte vrátených objektov, hodnota selektivity dotazu variuje v rozsahu 1 - 256. Heuristika pre rýchle určenie dynamického polomeru u k-NN dotazu stanovuje parameter odpovedajúci veľkosti nezhlukovanej časti databázy, tento parameter bol vo vetkých testoch nastavený na 2% z celkového počtu databázových objektov.

Pre jednoduchšiu orientáciu v parametroch testovaných metód boli vytvorená označenia použitých metód a ich parametrov. Prehľad použitých označení je možné nájsť v tabuľke 5.1.

Označenie	**Popis**
PT(n, c)	priama metóda PT (bez fázy pred-zhlukovania dát)
CPT-SF(n, c, *Tree c.*)	statická varianta metódy CPT (s dátami uloženými v sekvenčnom súbore)
CPT-MT(n, c, *Tree c.*)	dynamická varianta metódy CPT (s dátami uloženými v listoch M-stromu)
Mtree(c, *Tree c.*)	metóda M-strom n je počet globálnych pivotov c je kapacita (počet objektov) databázovej stránky
Tree c.	určuje typ konštrukcie M-stromu, nadobúda hodnoty (MW‖SW‖MW RI‖SW RI) MW označuje metódu viaccestného výberu listu SW označuje metódu jednocestného výberu listu RI označuje metódu vynúteného znovu-vloženia dát
range query(k)	rozsahový dotaz so selektivitou rovnou počtu k objektov
sorted by LB	použitie zotriedenia dolných odhadov skutočných vzdialeností v k-NN dotaze (LAESA prístup)
Random pivots	použitie metódy pre náhodný výber pivotov
LAESA pivots	použitie metódy pre výber maximálne oddelených pivotov (metóda navrhnutá s metódou LAESA)

Tabuľka 5.1: Označenia metód použité v experimentoch

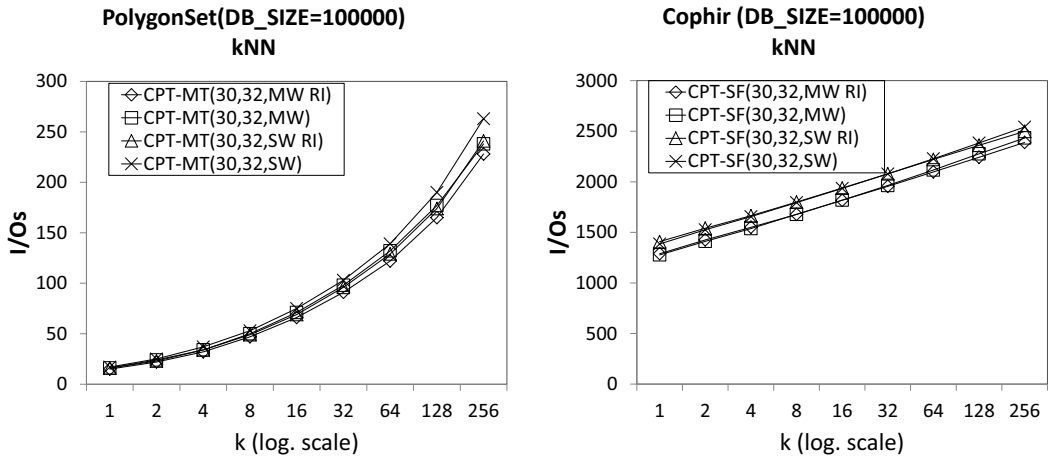
Dátová sada	SW	SW RI	MW	MW RI
ColorHist.	4,115,903	6,249,718	49,557,479	86,196,298
PolygonSet	6,164,869	9,051,290	17,572,966	29,340,106
Cloud	7,829,020	11,313,893	103,194,717	164,596,502
Cophir	6,937,617	10,163,521	191,559,936	304,081,707

Tabuľka 5.2: Počet DC potrebných na konštrukciu M-stromov z obrázku 5.1 a obrázku 5.2

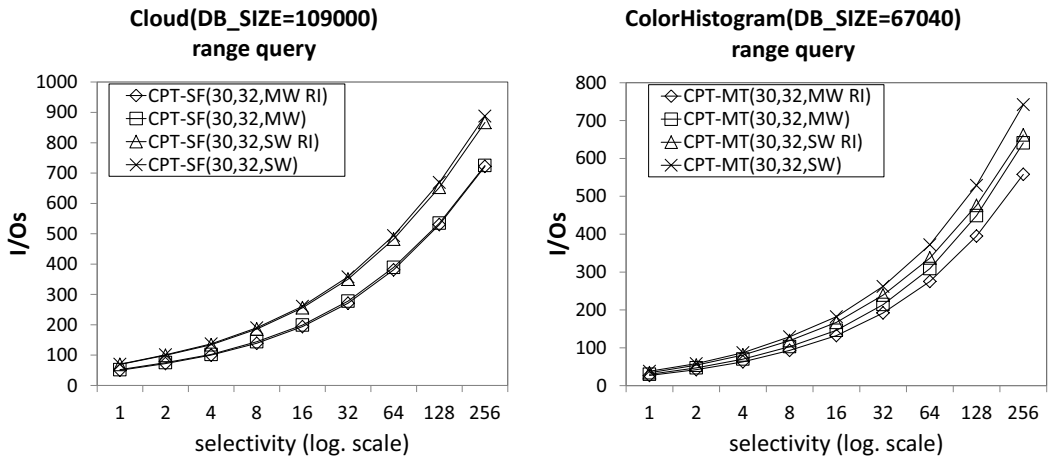
5.3 Výsledky

5.3.1 Varianty M-stromu

V prvej sade experimentov (obrázok 5.1 a obrázok 5.2) sme testovali rozličné spôsoby konštrukcie M-stromu a ich vplyv na kompaktnosť vytvorených regiónov (zhlukov) dát. Metóda viaccestného výberu listu porazila jednocestný výber listu v prípade všetkých testovaných dátových sád, naviac u statickej varianty metódy CPT je rozdiel značný. Pre dynamickú variantu metódy CPT sa metóda CPT-MT(30,32,SW RI) správa podobne ako CPT-MT(30,32,MW), najlepšie (čo do počtu I/O operácií potrebných pri vyhľadávaní) obstála v tomto prípade metóda CPT-MT(30,32,MWRI) a najhoršie metóda CPT-MT(30,32,SW). Ako môžeme pozorovať v Tabuľke 5.2, počet DC potrebných na konštrukciu M-stromu pomocou metódy viaccestného výberu listu je niekoľkonásobne vyšší ako počet DC potrebných pri jednocestnom výbere listu. Metóda vynúteného znovu-vloženia dát zvyšuje počet konštrukčných DC len nepatrne, preto pre ďalšie testovanie sme zvolil použitie metódy jednocestného výberu listu spolu s vynúteným znovu-vložením dát ako metódu konštrukcie zhlukovacieho M-stromu.



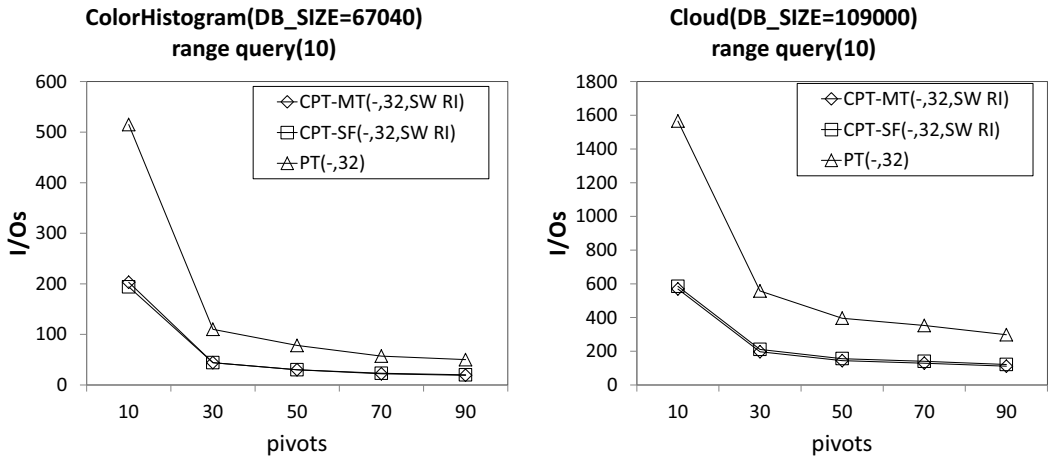
Obrázok 5.1: k-NN dotaz, variabilná selektivita: (a) dynamická varianta CPT (b) statická varianta CPT



Obrázok 5.2: rozsahový dotaz, variabilná selektivita: (a) statická varianta CPT (b) dynamická varianta CPT

5.3.2 Variabilný počet pivotov a veľkosť stránky

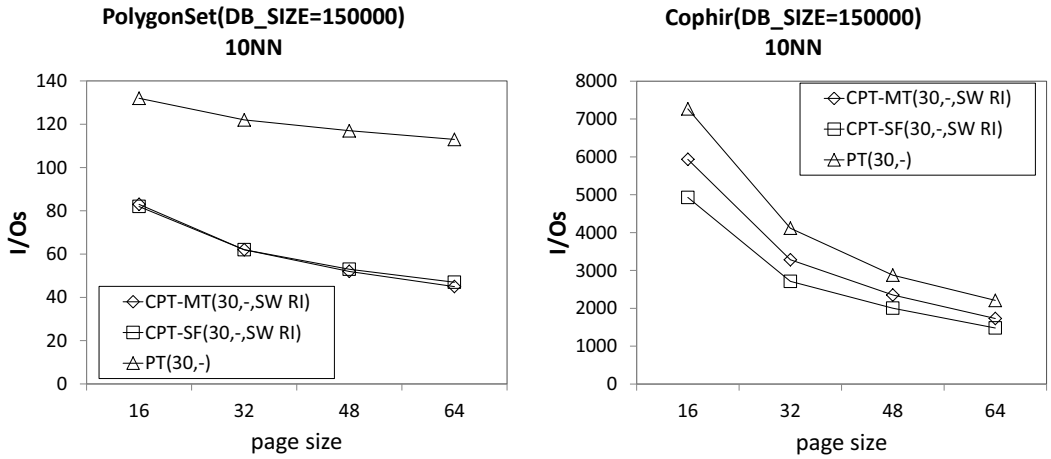
V druhej kolekcii testov sme testovali správanie sa testovaných metód s použitím rôzneho počtu globálnych pivotov (obrázok 5.3). Pri zvyšovaní počtu pivotov sa metódy správajú na reálnej aj syntetickej dátovej sade dosť podobne. Optimálny počet pivotov je niekde okolo 30 pivotov, pri nižších počtoch pivotov počet I/O operácií potrebných pri vyhľadávaní prudko narastá a naopak pri vyššom počte pivotov je efekt zníženia počtu I/O operácií zanedbateľný. Z výsledkov môžme vypozorovať, že pre malý počet pivotov a rozsahový dotaz s nízkou hodnotou selektivity je prínos zhlukovania dát u metódy CPT výrazný.



Obrázok 5.3: variabilný počet pivotov: (a) reálna dátová sada (b) syntetická dátová sada

V ďalšom teste sme sa zamerali na vplyv veľkosti databázovej stránky. Vyšší počet databázových objektov v jednej stránke môže pozitívne ovplyvniť zhlukovanie dát tým, že sa viac podobných objektov zoskupí na jednom mieste. Na druhú stranu to môže viesť k negatívnemu vytváraniu regiónov, ktoré pokrývajú veľký objem nepotrebného *mŕtveho* priestoru. Z vykonaných testov (obrázok 5.4) môže-

me vypozorovať, že zatiaľ čo u syntetickej dátovej sady sa prejavil pozitívny efekt zväčšovania databázovej stránky, u reálnej dátovej sady pozorujem práve opačný negatívny efekt, pretože dátová sada Cophir má vysokú vnútornú dimenziu.



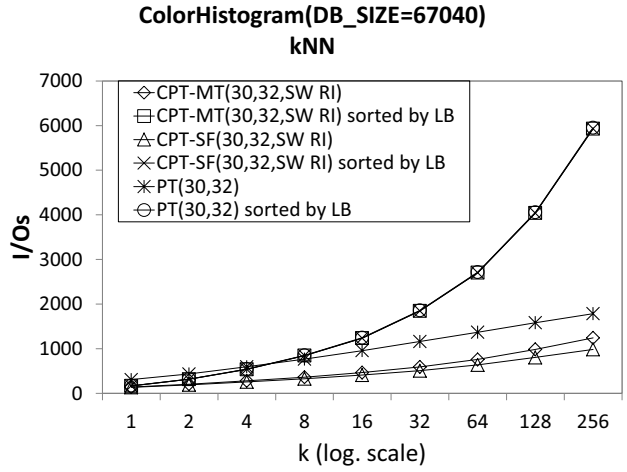
Obrázok 5.4: variabilný veľkosť databázovej stránky: (a) syntetická dátová sada (b) reálna dátová sada

5.3.3 Variabilná selektivita dotazu a veľkosť databázy

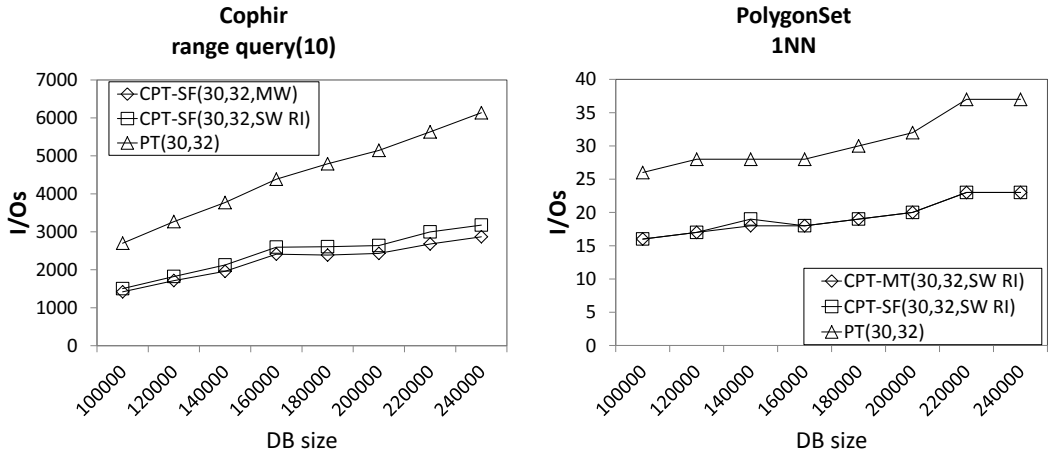
V tretej sade testov sme najprv skúmali štandardný LAESA prístup pre k-NN dotaz, čo znamenalo vzostupne zotriediť dolné odhady skutočných vzdialeností objektov databázy k dotazovému objektu a následne filtrovať objekty v tomto poradí. Metóda CPT dolné odhady nezotrieďuje, keďže preusporiadaním sa zničia vytvorené zhluky dát a tým sa zvýši počet I/O operácií, pretože objekty z jednej databázovej stránky nie sú spracovávané naraz, čo vedie k viacnásobnému načítaniu databázových stránok. Ako je možné vidieť na obrázku 5.5 zotriedením dolných odhadov sa úplne eliminuje pozitívny efekt zhlukovania dát a metóda CPT sa potom správa rovnako ako priama (bez fázy pred-zhlukovania dát) perzistentná metóda PT.

V ďalšom experimente sme skúmali vplyv rastúcej databázy, súčasťou tohto testu bolo tiež porovnanie dvoch metód na konštrukciu M-stromu. Ako môžeme vypozorovať z obrázku 5.6, vplyv zhlukovania dát na odfiltrovanie nepotrebných databázových stránok sa pre rozsahový dotaz zvyšuje spolu s rastúcou databázou. Bohužiaľ tento efekt sme nevysledovali u k-NN dotazu, metóda CPT aj v tomto prípade značne znižuje počet IO operácií, avšak s rastúcou databázou je tento efekt stagnujúci. Na ľavej časti obrázku 5.6 môžeme vidieť, že pre tento test je metóda viaccestného výberu listu len nepatrne lepšia ako metóda jednocestného výberu list s vynúteným znovu-vložením dát, ktorú sme zvolil pre porovnanie metódy CPT s priamou metódou PT.

V ďalšom teste sme sledovali variabilnú selektivitu dotazu. Podľa výsledkov testu (obrázok 5.7) môžeme usúdiť, že spolu s rastúcou selektivitou rozsahového dotazu rastie aj pozitívny efekt filtrovania nepotrebných databázových stránok, minimálne pre dátovú sadu s nízkou vnútornou dimenziou akou je aj skúmaná sada Cloud. Pre k-NN dotaz je filtrovanie najefektívnejšie pre nižšie hodnoty

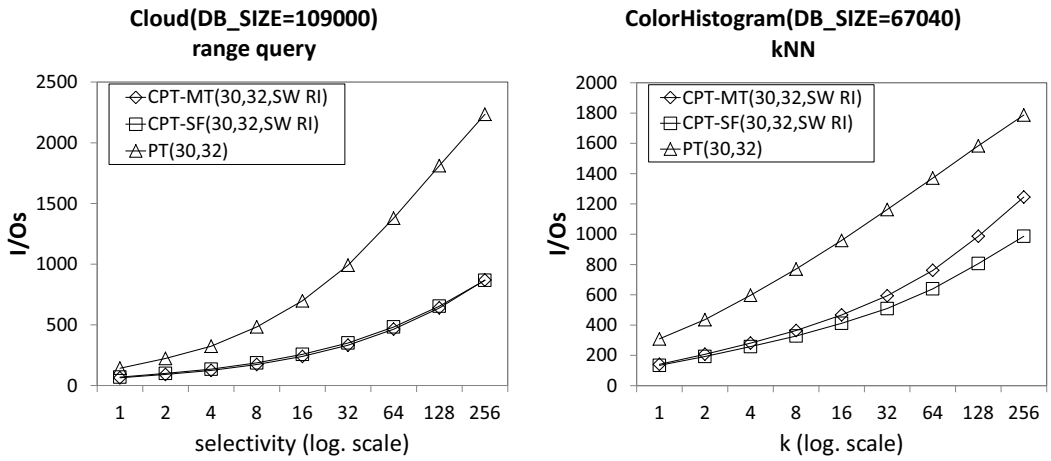


Obrázok 5.5: variabilná selektivita dotazu, porovnanie LAESA prístupu a štandardného CPT prístupu ku k-NN dotazu



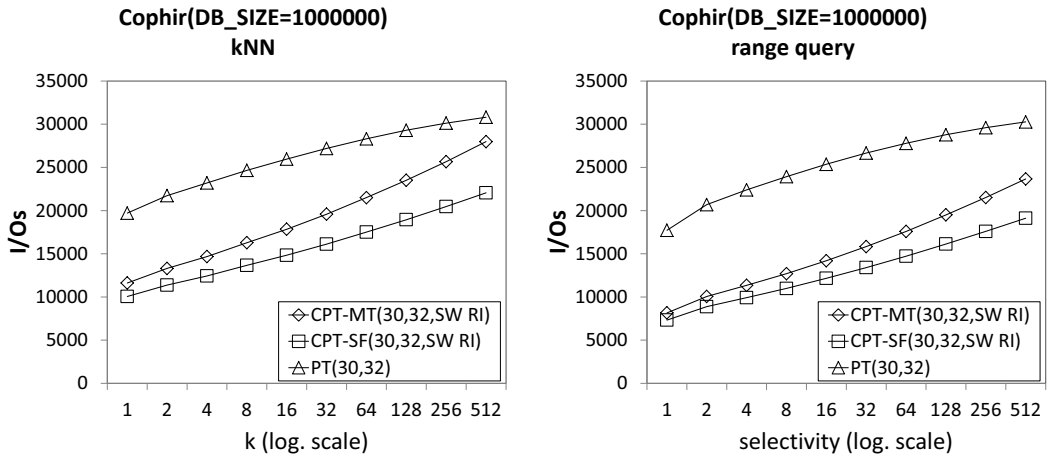
Obrázok 5.6: rastúca veľkosť databázy: (a) rozsahový dotaz (b) k-NN dotaz

selektivity, pri vyššie hodnoty k efekt filtrovania klesá. Podobne sa na reálnej dátovej sade bude správať aj rozsahový dotaz, ako môžeme vypozorovať s posledného testu v rámci tejto sady testov.



Obrázok 5.7: variabilná selektivita dotazu: (a) rozsahový dotaz (b) k-NN dotaz

V poslednom teste sme chceli preskúmať správanie sa metódy CPT na veľkej dátovej sade. Pre tento test sme použili 1,000,000 64-dimenzionálnych vektorov z databázy *CoPhIR*. Z výsledkov testu 5.8 na reálnej dátovej sade môžeme odvodiť, že navrhovaná metóda CPT je výkonnejšia pri vyhľadávaní pomocou rozsahového dotazu, ako pre k-NN dotazovanie. Taktiež môžme vypozorovať, že navrhovaná dynamická varianta metódy CPT so zvyšovaním selektivity dotazu rýchlejšie stráca svoj pozitívny efekt na zníženie počtu I/O operácií ako statická varianta metódy CPT.



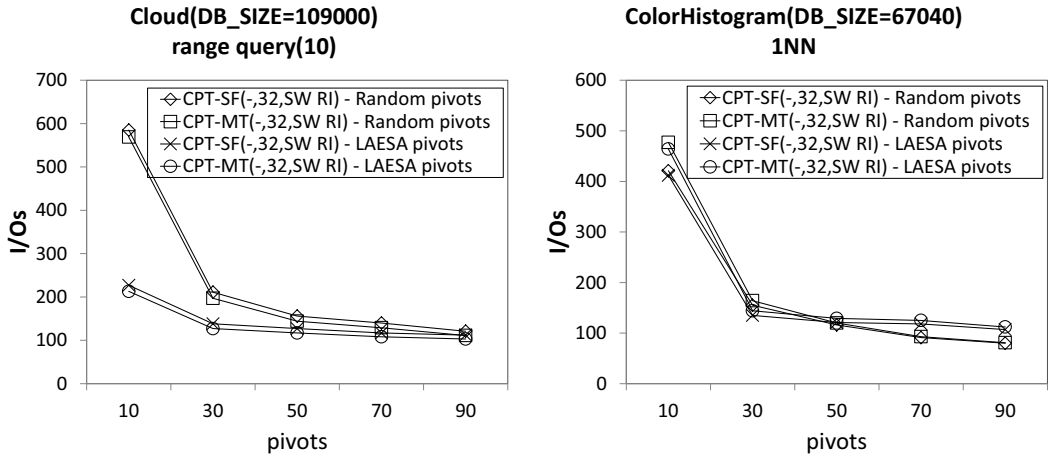
Obrázok 5.8: variabilná selektivita dotazu na veľkej databázy: (a) rozsahový dotaz (b) k-NN dotaz

5.3.4 Metódy výberu pivotov

V tomto teste sme sa zamerali na porovnanie rôznych metód výberu pivotov. Všetky ostatné testy používali náhodne zvolenú kolekciu pivotov, naša implementácia

navrhnutej CPT však ponúka aj metódu maximálne oddelených pivotov, ktorá bola navrhnutá spolu s metódou LAESA [12]. Metóda maximálne oddelených pivotov je neporovnateľne drahšia ako metóda náhodne zvolených pivotov, preto jej použitie musí mať signifikantný vplyv na vyhľadávanie, inak je táto metóda nepoužiteľná.

Ako môžeme pozorovať na výsledkoch testu (obrázok 5.9) použitie maximálne oddelených pivotov u rozsahového dotazu vykazuje značné zníženie počtu I/O operácií, obzvlášť pri menšom počte pivotov. U k-NN dotazu sledujeme naopak len minimálne zlepšenie pre menšie počty pivotov a pri použití väčšieho počtu pivotov kolekcia náhodne zvolených pivotov dokonca dosahuje lepšie výsledky ako maximálne oddelené pivoty. Z týchto výsledkov môžeme vyvodiť záver, že použitie sofistikovanejšej metódy pre výber pivotov môže vylepšiť vlastnosti navrhovanej metódy CPT, takže má zmysel sa v budúcnosti zaoberať skúmaním rôznych metód pre výber pivotov.

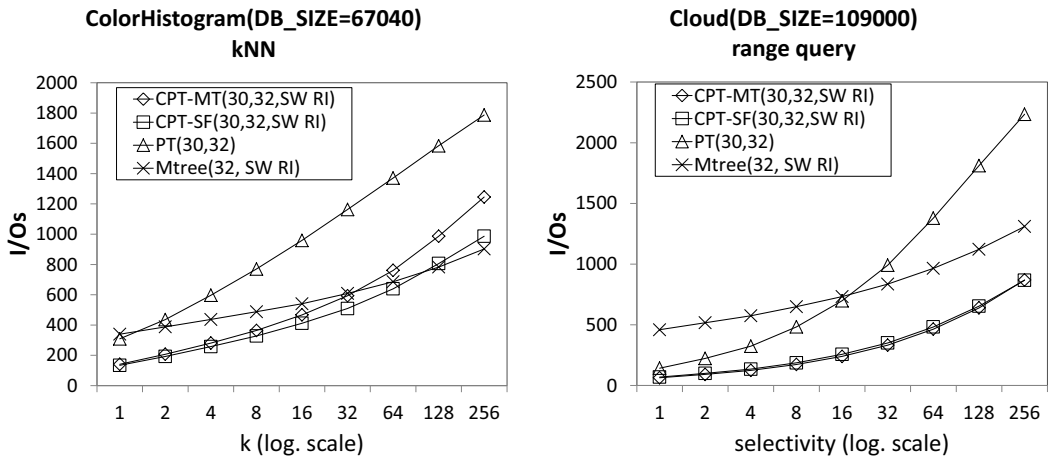


Obrázok 5.9: rôzne metódy výberu pivotov: (a) rozsahový dotaz (b) k-NN dotaz

5.3.5 Porovnanie s M-stromom

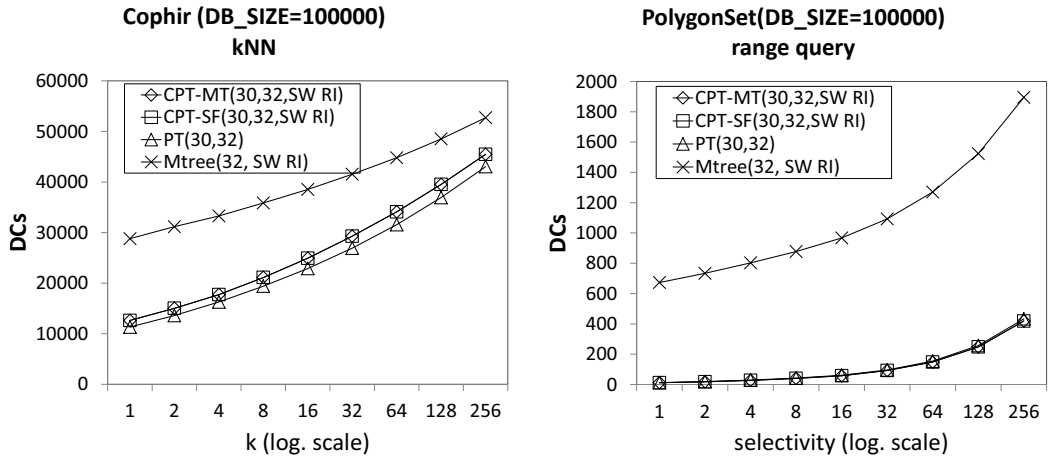
V tomto teste ponúkame porovnanie testovaných metód s implementáciou metódy M-stromu, ktorá bola použitá vo fáze pred-spracovania dát. Prvým parametrom pre porovnanie metód bol, tak ako v ostatných testoch, počet I/O operácií. Pri porovnávacom teste nebola v M-strome použitá *cache* na databázové stránky, keďže metóda CPT zatiaľ *cache* neimplementuje. Ako môžeme pozorovať na výsledkoch testu (obrázok 5.10), pre rozsahový dotaz dosahuje navrhovaná metóda lepšie výsledky v celom rozsahu testovanej selektivity dotazu. U k-NN dotazu to však neplatí, i keď je metóda pre nižšie hodnoty k lepšia ako M-strom, pre vyššie k obe varianty navrhovanej metódy CPT zaostávajú za výsledkami M-stromu.

Druhým testovaným parametrom bol počet výpočtov vzdialeností. Výsledky testu (obrázok 5.11) opäť potvrdili, že navrhovaná metóda dosahuje lepšie výsledky pre rozsahový dotaz v porovnaní s k-NN dotazom. Pri vyššej selektivite k-NN dotazu výsledky metódy CPT nie sú omnoho lepšie ako výsledky M-stromu, dá sa predpokladať, že pri ešte vyššej selektivite dotazu bude M-strom v tomto smere lepšou alternatívou. Taktiež môžeme vidieť, že navrhovaná metóda CPT



Obrázok 5.10: porovnanie s M-stromom, počet I/O operácií: (a) k-NN dotaz (b) rozsahový dotaz

dosahuje v počte DC pre k-NN dotaz mierne horšie výsledky ako priama metóda PT (bez fázy pred-spracovania dát).



Obrázok 5.11: porovnanie s M-stromom, počet výpočtov vzdialeností: (a) k-NN dotaz (b) rozsahový dotaz

5.4 Diskusia

Výsledky experimentov dokázali, že navrhovaná perzistentná implementácia metódy PT môže významne znížiť počet I/O operácií tým, že sa použije M-strom na pred-zhlukovanie indexovaných dát. Vo všetkých testovaných parametroch navrhovaná metóda CPT porazila priamu (bez fáze pred-spracovania dát) implementáciu metódy PT. Navrhovaná statická varianta metódy CPT porazila taktiež aj ďalšiu navrhovanú, dynamickú, variantu. Použitie dynamickej varianty má však zmysel v prípade, že v indexovanej dátovej sade sa počíta s ďalším pridávaním objektov po prvotnom vytvorení indexu, toto pridávanie nedegeneruje vytvorené zhluky a zachováva tak optimalizačný charakter navrhovanej metódy. Najlepší

zhlukovací M-strom bol postavený s použitím metódy viaccestného výberu listu avšak s vysokými konštrukčnými nárokmi. Ak sú tieto vysoké nároky na konštrukciu problém, vhodnou alternatívou je použitie metódy jednocestného výberu listu s vynúteným znovu-vložením dát, kde M-strom postavený pomocou tejto metódy vykazoval rozumné výsledky s oveľa nižšími konštrukčnými nárokmi. Navrhovaná metóda ukázala svoje prednosti aj v porovnaní s ďalšou MAM, s M-stromom.

Metóda CPT je však limitovaná niektorými faktormi a má aj svoje nevýhody:

- Konštrukcia indexu si vyžaduje použitie dvoch MAM, navyše dynamická varianta metódy CPT si vyžaduje udržiavanie dvoch MAM aj počas vyhľadávania.
- Pri použití veľmi drahej funkcie vzdialenosti, optimalizácia na počet DC je oveľa viac vyžadovaná ako optimalizácia na počet I/O operácií. Použitie klasického prístupu (ktorý navrhuje LAESA) pre k-NN dotaz je potom efektívnejšie.
- Ak má indexovaný priestor vysokú vnútornú dimenziu, vytvorenie kompaktných zhlukov dát je skoro nemožné.

Záver

V tejto práci sme navrhli novú perzistentnú implementáciu metódy PT, ktorá využíva pred-zhlukovanie dát pomocou M-stromu. Okrem navrhovanej implementácie dvoch variant metódy CPT, statickej a dynamickej, sme ponúkli aj nový prístup pre k-NN dotaz bez použitia preusporiadania dolných odhadov vzdialeností. Ako ďalšiu heuristiku pre k-NN dotaz sme navrhli rozdelenie indexovanej databázy na zhlukovanú a nezhlukovanú časť. Použitie myšlienky predzhlukovania dát na zníženie počtu I/O operácií sa ukázala ako správna, čo dokazujú výsledky vykonaných experimentov.

Ako ďalšia výhoda navrhovanej metódy CPT je fakt, že oddelenie štruktúry indexu od skutočných dát vedie k zníženiu celkového objem prenesených dát zo sekundárnej pamäti. Toto zníženie celkového objemu prenesených dát, by mohlo byť inšpiráciou aj pre ďalšie MAM (M-strom, PM-strom), kde by sa tiež mohla použiť myšlienka rozdelenia indexu na 2 časti. Celkový prenesený objem dát potom bude mať hlavný vplyv na rýchlosť I/O operácií pri sekundárnych pamätiach s minimálnou dobou prístupu (*seek time*), napr. pri použití technológie *Solid State Disk (SSD)*.

Ďalšia implementácia metódy CPT by sa mohla zamerať na použitie *cache-e* pre databázové stránky, čím by sa skutočný počet prenesených stránok (a teda aj I/O operácií) mohol ešte viac znížiť. Taktiež má zmysel skúmať použitie rôznych metód výberu globálnych pivotov a optimalizovať tak aj kolekciu pivotov na počet I/O operácií. Doimplementovaním operácií vloženia a mazania dát pre dynamickú variantu by sa z metódy CPT mohla stať plnohodnotná dynamická MAM. Zaujímavé môže byť porovnanie metódy CPT s ďalšími MAM na obrovských dátových sadách, kde počet I/O prístupov a celkový objem prenesených dát môže hrať veľkú úlohu. V neposlednom rade by mohla byť metóda CPT otestovaná na sekundárnych pamätiach s použitím technológie *SSD*.

Zoznam použitej literatúry

[1] Bolettieri, P.; Esuli, A.; Falchi, F.; aj.: CoPhIR: a Test Collection for Content-Based Image Retrieval. *CoRR*, ročník abs/0905.4627v2, 2009.
URL http://cophir.isti.cnr.it

[2] Bustos, B.; Pedreira, O.; Brisaboa, N.: A Dynamic Pivot Selection Technique for Similarity Search. *Similarity Search and Applications, International Workshop on*, ročník 0, 2008: s. 105–112.

[3] Chávez, E.; Marroquín, J. L.; Baeza-Yates, R.: Spaghettis: An Array Based Algorithm for Similarity Queries in Metric Spaces. In *Proceedings of the String Processing and Information Retrieval Symposium & International Workshop on Groupware*, SPIRE '99, Washington, DC, USA: IEEE Computer Society, 1999, ISBN 0-7695-0268-7, s. 38–46.
URL http://portal.acm.org/citation.cfm?id=519452.830765

[4] Chávez, E.; Navarro, G.; Baeza-Yates, R.; aj.: Searching in metric spaces. *ACM Computing Surveys*, ročník 33, č. 3, 2001: s. 273–321, ISSN 0360-0300.

[5] Ciaccia, P.; Patella, M.: Bulk Loading the M-tree. In *In Proceedings of the 9th Australasian Database Conference (ADC'98*, 1998, s. 15–26.

[6] Ciaccia, P.; Patella, M.; Zezula, P.: M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, VLDB '97, Morgan Kaufmann Publishers Inc., 1997, ISBN 1-55860-470-7, s. 426–435.

[7] Guttman, A.: R-trees: a dynamic index structure for spatial searching. *SIGMOD Rec.*, ročník 14, June 1984: s. 47–57, ISSN 0163-5808.
URL http://doi.acm.org/10.1145/971697.602266

[8] Hettich, S.; Bay, S. D.: The UCI KDD Archive. http://kdd.ics.uci.edu, 1999.

[9] Korn, F.; Muthukrishnan, S.: Influence sets based on reverse nearest neighbor queries. *SIGMOD Rec.*, ročník 29, May 2000: s. 201–212, ISSN 0163-5808.
URL http://doi.acm.org/10.1145/335191.335415

[10] Lokoc, J.; Skopal, T.: On Reinsertions in M-tree. In *Proceedings of the First International Workshop on Similarity Search and Applications (sisap 2008)*, Washington, DC, USA: IEEE Computer Society, 2008, ISBN 978-0-7695-3101-4, s. 121–128.
URL http://portal.acm.org/citation.cfm?id=1345535.1345858

[11] Micó, L.; Oncina, J.; Carrasco, R. C.: A Fast Branch & Bound Nnearest Neighbour Classifier in Metric Spaces. *Pattern Recogn. Lett.*, ročník 17, č. 7, 1996: s. 731–739, ISSN 0167-8655.

[12] Mico, M. L.; Oncina, J.; Vidal, E.: A new version of the nearest-neighbour approximating and eliminating search algorithm (AESA) with linear preprocessing time and memory requirements. *Pattern Recogn. Lett.*, ročník 15, č. 1, 1994: s. 9–17, ISSN 0167-8655.

[13] Navarro, G.: Analyzing Metric Space Indexes: What For? In *IEEE SISAP 2009*, 2009, ISBN 978-0-7695-3765-8, s. 3–10.

[14] Pedreira, O.; Brisaboa, N. R.: Spatial Selection of Sparse Pivots for Similarity Search in Metric Spaces. In *Proceedings of the 33rd conference on Current Trends in Theory and Practice of Computer Science*, SOFSEM '07, Berlin, Heidelberg: Springer-Verlag, 2007, ISBN 978-3-540-69506-6, s. 434–445.
URL http://dx.doi.org/10.1007/978-3-540-69507-3_37

[15] Ruiz, E. V.: An algorithm for finding nearest neighbours in (approximately) constant average time. *Pattern Recogn. Lett.*, ročník 4, July 1986: s. 145–157, ISSN 0167-8655.
URL http://portal.acm.org/citation.cfm?id=13010.13011

[16] Samet, H.: *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann, 2006, ISBN 0123694469.

[17] Sexton, A. P.; Swinbank, R.: Bulk Loading the M-Tree to Enhance Query Performance. In *Key Technologies for Data Management*, *Lecture Notes in Computer Science*, ročník 3112, editace H. Williams; L. MacKinnon, Springer Berlin / Heidelberg, 2004, ISBN 3-540-22382-7, s. 190–202.
URL http://dx.doi.org/10.1007/978-3-540-27811-5_18

[18] Skopal, T.: Pivoting M-tree: A Metric Access Method for Efficient Similarity Search. In *Proceedings of the 4th annual workshop DATESO, Desná, Czech Republic, ISBN 80-248-0457-3, also available at CEUR, Volume 98, ISSN 1613-0073,* http://www.ceur-ws.org/Vol-98, 2004, s. 21–31.

[19] Skopal, T.; Lokoč, J.: New dynamic construction techniques for M-tree. *J. of Discrete Algorithms*, ročník 7, č. 1, 2009: s. 62–77, ISSN 1570-8667.

[20] Skopal, T.; Pokorný, J.; Krátký, M.; aj.: Revisiting M-Tree Building Principles. In *Advances in Databases and Information Systems*, *Lecture Notes in Computer Science*, ročník 2798, Springer Berlin / Heidelberg, 2003, s. 148–162.
URL http://dx.doi.org/10.1007/978-3-540-39403-7_13

[21] Traina, C., Jr.; Filho, R. F.; Traina, A. J.; aj.: The Omni-family of all-purpose access methods: a simple and effective way to make similarity search more efficient. *The VLDB Journal*, ročník 16, October 2007: s. 483–505, ISSN 1066-8888.
URL http://dx.doi.org/10.1007/s00778-005-0178-0

[22] Zezula, P.; Amato, G.; Dohnal, V.; aj.: *Similarity Search: The Metric Space Approach*. Springer, 2005, ISBN 0387291466.

Zoznam použitých skratiek

DC	distance computation
PT	pivot tables
CPT	Clustered pivot tables
MAM	Metric Access Method
SAM	Spatial Access Method
k-NN	k-nearest neighbor
NN	nearest neighbor
ATOM	Amphora Tree Object Model
AVS	ATOM Vector Set
SSD	Solid State Disk

Zoznam obrázkov

Príloha A
Obsah priloženého média

A.1 Zdrojový kód

Implementácia metódy CPT bola vyvíjaná v prostredí Microsoft Visual Studio 2008, preto sú zdrojové kódy priložené vo formáte *Visual Studio C++ 2008 solution* v adresári `/source`. Metóda Clustered pivot tables je v tejto *solution* súčasťou projektu `PersistentPivotTables`. Keďže pri implementácii novej metódy boli nutne vykonané aj drobné úpravy v knižnici ATOM, prikladáme zdrojové kódy celej knižnice ATOM, v stave v akom bola použitá pre implementáciu metódy CPT, knižnica ATOM je v priloženej *solution* súčasťou projektu `Atom`.

Samotná aplikácia, ktorá bola použitá na otestovanie navrhnutej metódy sa nachádza v adresári `/build`. Cieľom práce nebolo implementovať plnohodnotné rozhranie multimediálnej databázy, priložená aplikácia je spustiteľná z príkazového riadku operačného systému Windows a poskytuje len rozhranie na otestovanie navrhovanej metódy, k tomu boli uspôsobené aj vstupné parametre aplikácie.

A.2 Dátové sady

Testované dátové sady sú priložené v adresári `/datasets`. Dátové sady vo formáte AVS 4.1.1 boli pre úsporu miesta skomprimované (.zip). V podadresári `/datasets/queries` sa nachádzajú dotazové objekty, na ktorých boli vykonané experimenty. Ide o posledných 1000 objektov z každej testovanej sady, preto pri testovaní dátových sád je vhodné znížiť počet testovaných objektov o týchto 1000 objektov. Súbor `queries.avs` obsahuje dotazové objekty pre dátovú sadu ColorHistogram, `queriesINT.avs` pre sadu Cophir, `queriesCLOUD.avs` pre sadu Cloud a `queriesPS.avs` pre sadu PolygonSet.

A.3 Výsledky testov

V adresári `/results` sa nachádzajú výsledky vykonaných testov. Výsledky sú vo formáte Microsoft Excel (.xls), kde sa nachádzajú namerané dáta aj vygenerované grafy. Spolu s výsledkami sú priložené aj dávkové súbory (.bat) s nastavenými parametrami príslušného testu.

A.4 Dokumentácia

V súbore `/documentation/userManual.pdf` sa nachádza popis aplikácie a jej parametrov, ktorá bola použitá na testovanie navrhnutej metódy CPT. Adresár `/documentation/reference` obsahuje referenčnú dokumentáciu vygenerovanú nástrojom Doxygen [1] vo formáte HTML a LATEX.

[1] k dispozícii na adrese http://www.doxygen.org

A.5 Dokumenty

V adresári `/documents` je možné nájsť text diplomovej práce vo formáte PDF.