VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

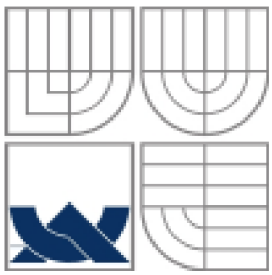
INDEXOVÁNÍ POHYBUJÍCÍCH SE OBJEKTŮ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

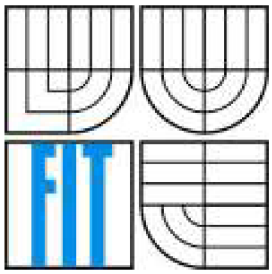
AUTOR PRÁCE
AUTHOR

Bc. JIŘÍ VETEŠNÍK

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INDEXOVÁNÍ POHYBUJÍCÍCH SE OBJEKTŮ

MOVING OBJECTS INDEXING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. JIŘÍ VETEŠNÍK
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. JAROSLAV ZENDULKA, CSc.
SUPERVISOR

BRNO 2008

Abstrakt

Tato práce se zaměřuje na navržení vhodného indexování pohybujících se objektů. S rozšířením mobilních zařízení je potřeba spravovat rozsáhlé množiny časoprostorových dat. Práce zahrnuje problematiku časoprostorových dat a základní obecné přístupy indexování těchto dat. Dále ukazuje podporu prostorových dat systému Oracle. Pohyb je typicky reprezentován trajektorií ve dvou dimenzionálním prostoru se složkou času ve třetí dimenzi. Součástí práce jsou experimenty v databázovém systému Oracle na uměle vygenerovaných datech.

Klíčová slova

Oracle, časoprostorová data, indexování pohybujících se objektů, trajektorie, R-strom, Quad-strom

Abstract

This work is aimed for proposing acceptable indexing of moving objects. With the enlargement of mobile computing it is needed to manage large sets of spatiotemporal data. We introduce the problem of spatiotemporal data and basic general approaches of indexing these data. Further, we show support of spatial data in Oracle. The movement is typically represented as trajectory in two dimensional space with temporal component in third dimension. The thesis contains experiments performed in database Oracle on artificially generate data.

Keywords

Oracle, spatiotemporal data, indexing moving objects, trajectory, R-tree, Quad-tree

Citace

Vetešník Jiří: Indexování pohybujících se objektů, diplomová práce, Brno, FIT VUT v Brně, 2008

Indexování pohybujících se objektů

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením doc. Ing. Jaroslava Zendulky, CSc. Další informace mi poskytl Ing. Jaroslav Ráb.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jméno Příjmení
Datum

Poděkování

Chtěl bych poděkovat vedoucímu diplomové práce doc. Ing. Jaroslavu Zendulkovi, CSc. a Ing. Jaroslavu Rábovi za jejich nápady, podněty a odborné vedení při realizaci práce.

Obsah

1 Úvod

Po dlouhou dobu systémy pro řízení relačních databází (RSŘBD) poskytují uživatelům s pokročilejším datovým řízením užitečné rysy. Například centrální skladiště pro datová úložiště, souběžné řízení, zálohovací a zotavovací mechanismy, podporu pro více uživatelů a transakční zpracování, nástroje pro řízení velmi rozsáhlých datových množin. Databáze se skládají z relačních tabulek a v jejich sloupcích se objevují jednoduché datové typy (např. integer, float, string). RSŘBD má integrovánu podporu indexování pro tyto jednoduché datové typy. Bohužel tyto datové typy a podpora indexování jsou nedostačující pro více složité aplikační domény. Proto se objevuje potřeba spravovat složité datové typy s podporou indexování těchto složitějších datových typů integrovaných do RSŘBD. Specializovaný prostorový objekt a vícedimenzionální index může radikálně zvýšit rychlost nalezení záznamů pro rozsáhlé databáze oproti alternativě mapovat prostorový objekt na jednoduché datové typy a indexové struktury, které by normálně byly dostupné.

S masovým rozšířením mobilních telefonních zařízení, GPS, bezdrátových počítačů a jejich schopností přesně hlásit aktuální pozici a všudypřítomných aplikací (dopravní aplikace, řízení loďstva, lokalizačně informační služby), které tyto informace mohou prospěšně využít, jsou velmi žádané. S daty těchto zařízení se objevuje problém naléhavé potřeby podpory časoprostorových dat uvnitř RSŘBD . Cokoliv mění pozici v čase, tak může být klasifikováno jako časoprostorová data. Například vozidla vybavená GPS přijímačem mohou přenášet data do centrálního počítače pomocí radiové komunikace nebo mobilního telefonu každých několik sekund nebo minut v závislosti na zařízení. Můžeme jednoduše zjistit, kde se monitorované vozidlo nachází a přijímat signály z vozidla jako jsou alarm, hlášení o nehodě. Atributy pohybujícího se objektu můžeme rozdělit do dvou kategorií: *statické* a *dynamické*. Mezi *statické* atributy řadíme rysy, které nesouvisí s umístěním objektu jako jsou id či název objektu. Tato data mohou být zpracovávána obyčejnou relační databází. Naproti tomu *dynamické* atributy odkazují na reálnou časovou informaci určující pozici objektu. Protože pozice objektu se dynamicky nepřetržitě mění v čase, produkuje tento pohyb velké množství dat, které je potřeba vhodným způsobem uložit a poté se na tato data dotazovat.

Jsou dva základní typy časoprostorových databází, ty které obsahují *historické* informace a ty které spravují *současné* informace pro současně/budoucí účely dotazu. Tato zpráva se zaměřuje na první kategorii, tj. předpokládáme databázové úložiště kompletních historických pohybujících se objektů v čase a musí umět odpovídat na dotazy v jakémkoliv čase historie objektů. Například historická data o vozidle mohou být použita pro operační rozhodovací podporu, zlepšení kvality služeb a naplánování dopravy. Každý záznam má atributy $<oid, x, y, t_s, t_e>$ kde:

- $<oid>$ identifikuje objekt
- $<x,y>$ prostorové souřadnice

- $<t_s, t_e>$ označuje časový interval, během kterého objekt zůstal na pozici $<x, y>$

Model odráží aplikace reálného světa, kde dráha neboli trajektorie objektu je nepřerušený pohyb mezi datovými body. Přijmutí objektu sledující lineární dráhu mezi datovými body může vést do nesprávných předpokladů. Například v bezpečnostně/monitorovacích aplikacích by osobě mohlo být přiřazeno mít přístup k vyhrazené části z důvodu přerušení pohybu objektu. Časový interval je implicitně datově komprimován pro případ, kdy se lokace objektu nemění (protože se objekt nepohybuje nebo se nepohybuje mimo přesnost daného zařízení) a pouze jediný řádek je vložen do databáze. Časoprostorový model dotazu Q nabývá formy Q = <R, T>, kde R je prostorová oblast a T je časový rozsah. Q vrací unikátní identifikátory záznamů (OID). Přičemž souřadnice $<x, y>$ jsou uvnitř R a interval $<t_s, t_e>$ protíná T.

Tato práce navazuje na diplomovou práci [13], jejímž obsahem bylo navržení vhodné databázové struktury pohybujících se objektů a ukázkové aplikace ověřující funkčnost navrhnuté struktury databáze. Já jsem převzal navrženou databázovou strukturu a na ní jsem posléze prováděl experimenty, které jsou obsahem této diplomové práce. Ve výsledku to znamená, že je možné aplikovat mé navržené experimenty do funkční aplikace uvedené v [13]. Pro kapitoly 1 až 3 jsem čerpal zejména z [1]. Kapitola 2 rozebírá podrobněji problematiku časoprostorových dat, zahrnuje diskuzi o časoprostorovém generování dat a modelování, typy časoprostorových dotazů. Dále zhodnocuje existující časoprostorové přístupové metody, které mohou být plně integrovány do RSŘBD . Vysvětluje, co jsou to časoprostorová data a jak je lze reprezentovat. Kapitola 3 rozebírá problematiku časoprostorových dat z pohledu podpory současných RSŘBD a možnostech integrování uvnitř RSŘBD . Sekce 4 popisuje současné možnosti ukládání a indexování časoprostorových dat v RSŘBD Oracle 10g. Také rozvádí podporu Oracle pro časoprostorová data, indexování a dotazování. Pátá kapitola popisuje úkony, které bylo třeba vykonat před samotným realizováním experimentů. Zahrnuje vytvoření návrhu databáze, vygenerování časoprostorových dat a rozvádí detailněji problematiku a implementaci R-strom a Quad-strom indexů. V kapitole šest představuji návrhy jednotlivých experimentů. Těsně navazující kapitola sedm zhodnocuje informace získané z experimentů a staví je proti sobě v přehledné formě tabulek a grafů. Závěrečná osmá kapitola shrnuje celou moji práci a nastiňuje možné způsoby navázání na tuto diplomovou práci.

2 Reprezentace a způsoby indexování

2.1 Časoprostorová data

Časoprostorová data pocházejí z různých zdrojů, například pohybu lidí, zvířat, vozidel, letadel, lodí, počasí (bouřky), změny hranic zemí. Seznam je prakticky nekonečný, protože cokoliv, co mění pozici v čase, může být klasifikováno jako časoprostorová data. Důležitým rozdělením pohybujících se objektů je definování omezení pro jejich pohyb, jež mohou nastat. Obecně se dá říci, že existují tři běžné scénáře pohybu, a to neomezený pohyb (např. veslař v moři), omezený pohyb (např. auta, chodci) a posledním případem je pohyb v sítích (např. vlaky, tramvaje).

Důležité využití časoprostorových informací zahrnuje sledování osob, dopravních prostředků, časoprostorového data-mining (hledání vzorů v rozsáhlých datových množinách měnících se dat). Dopravní a směrovací aplikace ve městech pomáhají řešit problémy s dopravními zácpami. Navádění uživatele kudy jít, pokud se ztratí či efektivně doporučit alternativní cestu pro případ již zmíněné zácpy nebo uzavřené cesty.

Pro sběr dat mohou být použity různé technologie. Pokud zařízení zahrnuje GPS modul, uživatelská pozice může být určena s přesností v rozmezí 2–20 metrů (uvnitř budov nedosahuje takové přesnosti). Poloha mobilního telefonu může být zjištěna bez potřeby GPS modulu. A to zjištěním, ve které buňce se telefon nachází nebo měřením vzdáleností mezi překrývajícími se buňkami. Polohu mobilního telefonu je možné určit v okruhu 50 metrů v městské oblasti a několika kilometrech ve venkovském osídlení. Výhodou je, že existující mobilní technologie může být využita pro sledování informací o poloze bez nutnosti budování zvláštní technologické infrastruktury.

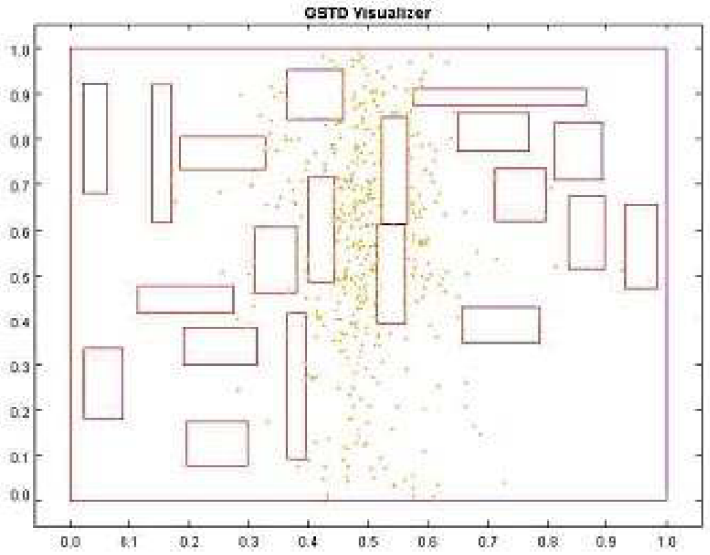
2.1.1 Generování časoprostorových dat

Z důvodu majetnické a soukromé povahy dat generovaných GPS nebo telefonem může být obtížné získat reálné datové množiny časoprostorových dat pro experimentální účely. Množiny o velikostech několika stovek záznamů jsou nedostatečné. Kvůli nedostatku snadno dostupných reálných datových množin a potřebě generovat data v řízeném tvaru, existují umělé časoprostorové generátory. Mezi nejznámější generátory patří „The Generate Spatio-Temporal Data“ (GSTD), „Network-based Generator of Moving Object a v neposlední řadě „City Simulator“ od firmy IBM.

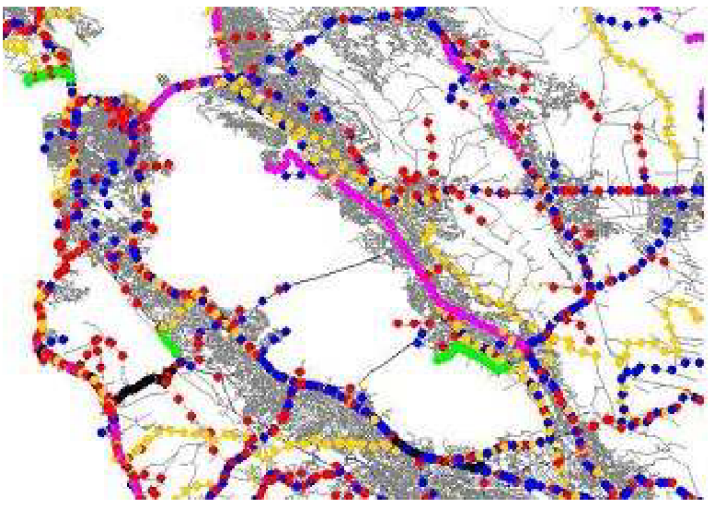
GSTD generuje data podle předepsaných statistických distribucí. Mezi současně podporovanými jsou *Gaussian* a *Skewed* distribuce. Pro bodová data, GSTD požaduje počáteční čas a střed distribuce. Počáteční distribuce určí, kde objekty začínají uvnitř celkového prostoru, který GSTD předpokládá. Čas distribuce řídí časová razítka, kdy bude poloha objektu aktualizována. Střed distribuce řídí pohyb

objektů v prostoru. Pokročilejšími rysy lze definovat překážky, které objekty mohou zaznamenat. Výsledek generování ve vizualizované podobě můžeme vidět na obrázku 1.

„Network-based Generator of Moving Objects“ může generovat datovou množinu, kde objekty pocházejí z dané sítě (např. silnice). Objekty předpokládají startovací polohy a cíle. Klíčovou výhodou je možnost generování umělých dat, která sledují topologii sítě (např. síť silnic). Ukázka reprezentace generovaných dat je k nahlédnutí v obrázku 2. Tečky reprezentují rozdílné typy pohybujících se objektů a čáry infrastrukturu, kterou objekty sledují.



Obr. 1 GSTD (Převzato z [1]).



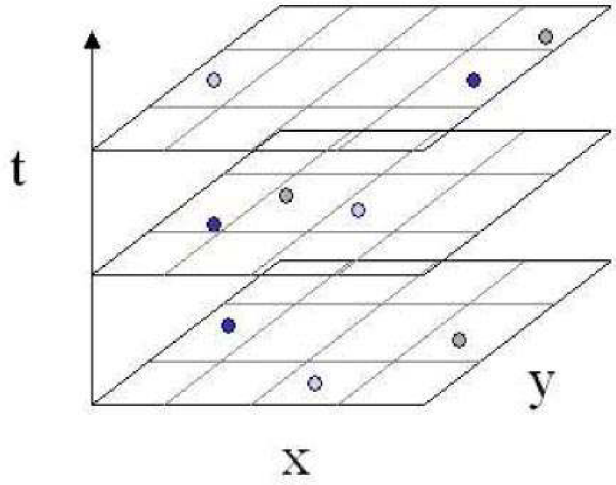
Obr. 2 Network-based Generator of Moving Objects (Převzato z [1]).

„City simulator“ je na Jave založený časoprostorový generátor vytvářející časoprostorová data reprezentující pohyb lidí v plně trojdimenzionálním virtuálním městě. Může generovat datovou množinu až jednoho miliónu lidí pohybujících se skrz infrastrukturu města (například parky,

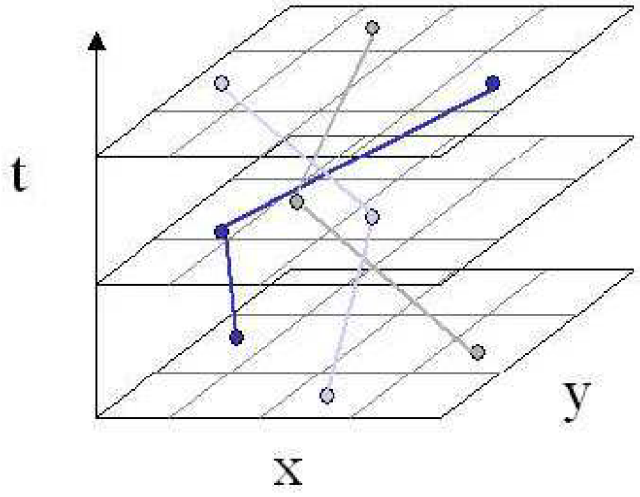
budovami, silnicemi, křižovatkami). Většina generátorů dovoluje nejvýše dvojdimenzionální pohyb v prostoru plochy. „City simulator" je však unikátní v tom, že dovoluje pohyb objektů simulovat i mezi patry budov.

2.1.2 Modelování časoprostorových dat

Obojí reálná i umělá časoprostorová data mohou být modelována různými způsoby v závislosti na sémantice aplikace a typu dotazů. Data mohou být modelována dvojdimenzionálními nebo trojdimenzionálními prostorovými objekty s časem přidaným do další dimenze. Na obrázku 3 můžeme vidět body ve trojdimenzionálním prostoru. Přestože je pohyb objektů v čase souvislý, data se obvykle skládají z diskrétního vzorků bodů udávající pozici objektu v čase. Například vozidlo může hlásit pozici centrálnímu serveru každých 5 minut.



Obr. 3 Modelování časoprostorových dat jako bodů (Převzato z [1]).



Obr. 4 Modelování časoprostorových dat jako trajektorií (Převzato z [1]).

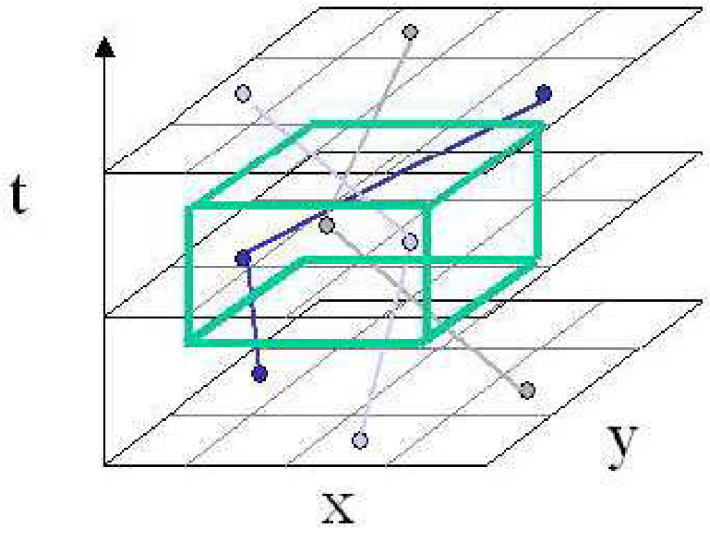
Pro tyto body můžeme použit formu abstrakce, kterou nazýváme trajektorie. V takovém případě je lineární trajektorie dopočítána mezi datovými body. Na obrázku 4 vidíme, že tento model předpokládá pohyb objektu v přímé linii konstantní rychlostí mezi skutečnými zaznamenanými body. Model trajektorie může být obzvláště užitečné použít v oblastech, kde je pozice objektů aktualizována relativně občas a kde se lze předpokládat, že dotazy mohou padnout mezi časové intervaly datových bodů.

Třetí možností modelování dat je velmi často využívána pro současně/budoucí pozici pohybujících se objektů a nazývá se parametrický model. Zaznamenávání pozice aktuálních datových bodů do časoprostorové databáze je nahrazeno uložením parametrizované poměrně rychlé funkce pro každý pohybující se objekt v systému. Na první pohled je zřejmé, že tento model poskytuje výrazné zredukování ukládaných záznamů. Cenou za redukci je ovšem nemožnost zaručit přesnost požadované informace, protože model předpokládá, že objekt sleduje plánovanou dráhu.

2.2 Časoprostorové dotazy

2.2.1 Úvodní rozdělení časoprostorových dotazů

Rozlišení typu časoprostorového dotazu záleží na významu dotazu, pokud nás zajímají informace z minulosti, pak řadíme dotazy do kategorie *historické*. Naproti tomu kategorie *současně/budoucí* odpovídá na dotazy, které vracejí informace o budoucnosti. Efektivní získání co nejpřesnějších datových množin pro oba typy dotazů je velmi důležité pro časoprostorově založené aplikace.



Obr. 5 Časoprostorový rozsahový dotaz (Převzato z [1]).

Současně/budoucí časoprostorové přístupové metody podpory dotazů předpovídají pohyb pozice objektu v daném čase založeném na současné rychlosti objektu. Odpověď na dotaz je

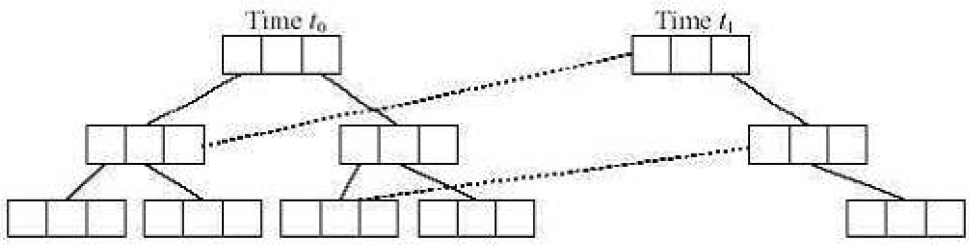
prozatímní. Pohybový vektor objektů se může měnit v jakémkoliv čase a to určuje množinu odpovědí dotazu.

Historické časoprostorové přístupové metody podpory dotazů mohou být klasifikovány jako založené na souřadnicích nebo trajektoriích. Na souřadnicích založené mohou nabývat mnoha forem, zahrnující rozsah, nejbližšího souseda, k-nejbližších sousedů. Pomocí rozsahových dotazů můžeme získat objekty uvnitř předepsané oblasti dané časovou periodou. Kvádr na obrázku 5 reprezentuje příklad rozsahového dotazu. Mějme souřadnice pohybujícího se objektu, pak dotaz na nejbližšího souseda nalezne objekt uvnitř nejbližšího sousedství v daném čase. K-nejbližších sousedů (kde *k* je uživatelem specifikovaná konstanta) nalezne *k*-nejbližších objektů dotazu k souřadnicím. Dotazy založené na trajektoriích jsou zaměřeny na topologické nebo navigační informace. Topologický dotaz se ptá, zda trajektorie vstoupila, opustila, zkřížila, setrvala uvnitř nebo obešla daný časoprostorový rozsah. Navigační dotaz uvažuje odvozené informace jako jsou například rychlost (nejvyšší, průměrná), směr, procestovanou vzdálenost, pokrytou oblast atd. Můžeme využívat i dotazy kombinované.

2.2.2 Dotazy na historii časoprostorových objektů

Data typicky modelujeme jako trajektorii pro účely časoprostorových přístupových metod, příklad takové trajektorie můžeme vidět na obrázku 4. Máme dvojdimenzionální prostorovou oblast, čas uvedeme jako třetí dimenzi a pohyb objektů mezi záznamy je modelován ve smyslu linie segmentů. Mnoho navržených časoprostorových přístupových metod je založeno na R-strom přístupu.

3-D R-strom je jednou z variant R-stromu. S časovou dimenzí zachází jednoduše jako s běžnou prostorovou dimenzí. Časoprostorová data jsou indexována použitím standardní trojdimenzionální R-strom struktury.



Obr. 6 HR-strom (Převzato z [1]).

HR-strom je příkladem překrývající se struktury s více verzemi, které také přizpůsobují R-strom pro *historická* časoprostorová data. Rozšiřuje myšlenku B+stromu ve více verzích na časoprostorové doméně. Je vytvořen „virtuální" prostorový R-strom pro každé časové razítko (jednoduchý bod) v datové množině. Vyhýbáme se tím nadbytečné režii ukládání odděleného R-strom indexu v každém

časovém razítku. Uzly odkazující na uzly, které se nemění, jsou provázány zpět do originální struktury. Pro časový výřez dotazu je tento přístup extrémně účinný, protože R-strom existuje v každém časovém razítku. Naopak pro rozsahové dotazy je přístup neefektivní. Ukázka HR-stromu je zobrazena na obrázku 6.

TB R-strom (trajectory bunde tree) navrhuje trajektorií orientované přístupové metody, které mohou dávat výsledek pro trajektorií orientovaným dotazům rychleji, nežli je tomu u R-stromu. TB R-strom vyžaduje modifikaci R-strom struktury, kde uzly listové úrovně obsahující informace z nějaké trajektorie jsou provázány dohromady. To může vést k redukci času dotazu v případě dotazování se na kompletní trajektorii objektu.

3 Časoprostorová podpora RSŘBD

3.1 Úvodem

Potřeba poskytnutí RSŘBD podpory pro časoprostorová data je přirozeným rozšířením postoje vzatého výhradně pro prostorovou nebo výhradně pro časovou databázovou podporu. V nejběžnějším slova smyslu pracují časoprostorové databáze s geometrickými změnami v čase. RSŘBD organizuje data do tabulkových souborů (tabulek), které mohou být navzájem provázány. Sloupce každé tabulky representují datová pole a mají předurčené datové typy (například integer, float, string). Řádky tabulky, často nazývané n-tice uchovávají skutečná data. RSŘBD poskytuje prostředky pro indexování sloupců tabulek za účelem zvýšení výkonu dotazu. Index urychluje vykonání dotazu poskytnutím oddělené datové struktury s ukazateli do řádků identifikovaným jedinečným identifikátorem řádku.

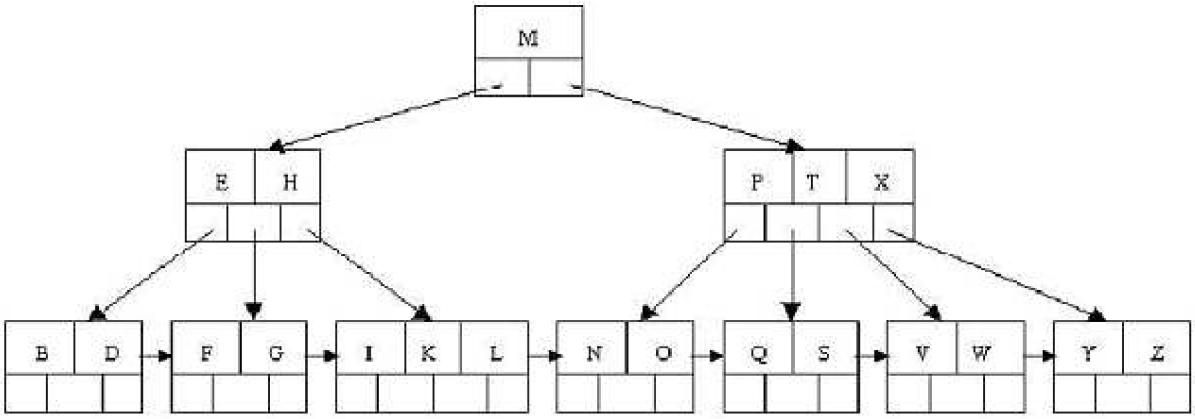
3.2 Struktura indexu

Základní struktura indexu poskytnutá uvnitř RSŘBD je hašování na indexu v podobě stromu. Hašování na indexu v podobě stromu využívá funkce mapování záznamu do stránek na disku. Odpovídá dotazům na specifickou hodnotu efektivně, funkce hašování na indexu v podobě stromu aplikovaná na tuto hodnotu vrací stránku, ve které jsou umístěny všechny záznamy s danou hodnotou. Předpokládejme efektivní funkci, to je často případ kdy je nezbytný pouze jediný přístup na disk k znovuzískání odpovídajících záznamů v čase dotazu.

3.2.1 B+strom

Typický B+strom index je na stromu založená struktura uvnitř RSŘBD. B+strom je budován přidáním záznamu spolu s ukazatelem (unikátní identifikátor řádku tohoto záznamu) do uzlů listové úrovně indexu, kde každý uzel koresponduje se stránkou na disku. Na obrázku 7 vidíme rodičovské uzly vybudovány za účelem obsáhnutí hodnot, které ohraničují minimální a maximální rozsah hodnot, které jejich synovské uzly obsahují. Tento proces je rekurzivně opakován až po vybudování horní úrovně stromu. Minimální číslo m a maximální číslo M udává počet položek na uzel (mimo kořenový), což zabezpečuje vyváženou strukturu stromu. Pro položku dotazu, tj. dotaz proti prosté hodnotě je následována prostá cesta stromem směrem k uzlům listové úronvě. Pro rozsahové dotazy musí být také následována pouze prostá cesta stromem směrem k listům. Pokud je nalezena minimální hodnota, indexový rozsah prohledávání může postupovat následnými ukazateli do uzlů listové úrovně. Pro jednodimenzionální data, tj. sloupec znaků v databázové tabulce nabízí B+strom index extrémně dobrý výkon a rozšiřitelnost. Dokonce i při existenci řádově miliónů databázových řádků požaduje struktura

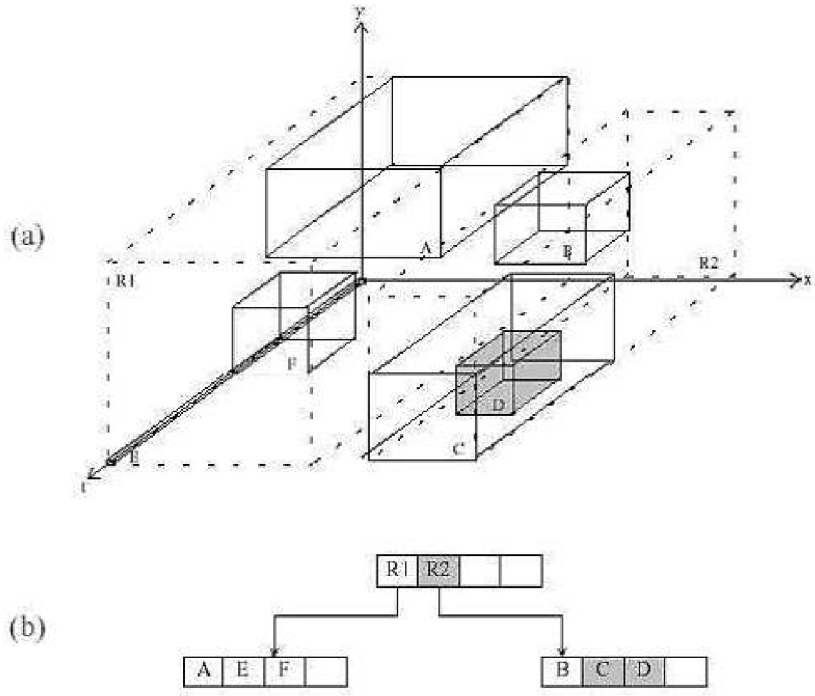
B+stromu typicky pouze tři úrovně. Za předpokladu, že kořen a uzly prostřední úrovně jsou uloženy v „čachr“, je k zodpovězení dotazu nezbytný pouze jeden přístup na disk, protože uzly listové úrovně obsahují skutečnou hodnotu sloupce uloženou v jeho přidruženém tabulkovém záznamu.



Obr. 7 B+strom (Převzato z [1]).

3.2.2 R-strom

Podpora prostorových datových typů a indexů je poskytována ve většině RSŘBD. Převážná většina RSŘBD zakládá podporu indexování na klasické struktuře R-stromu. Aproximací každého prostorového objektu v databázi buduje R-tree minimální ohraničený obdélník (MBR) a poté každý takový MBR vkládá do uzlů listové úrovně R-stromu. Obrázek 8 ilustruje trojdimenzionální R-strom, kde boxy A – F representují MBR skutečného trojdimenzionálního prostorového objektu na disku. Skupiny objektů MBR jsou sdruženy do rodičovských uzlů R1 a R2. Cenou založený algoritmus je v čase vložení použit k rozhodování, který uzel nového objektu bude vložen na základě kritéria limitovaného množství překrytí mezi uzly a množstvím mrtvých míst ve stromu. Například seskupení objektů A, E a F společně do R1 v obrázku 8 vytváří menší MBR, než kdyby A, E a C byly seskupeny namísto toho. Analogicky jako u B+stromu je nutné určit minimální a maximální číslo položek na uzel pro zabezpečení vyrovnanosti stromu. V čase dotazu je strom procházen od kořene procházením každého uzlu, ve kterém okno dotazu protíná MBR. V uzlech listové úrovně jsou pak pouze ty objekty MBR, které protínají dotaz MBR. Vzpomeňme, že u B+stromu je potřeba projít pouze jedinou cestu skrze strom. Nicméně u R-stromu může být v některých případech nezbytné projít i několik cest stromem, protože okno dotazu může protínat několik MBR v každém uzlu.



Obr. 8 Tří dimenzionální MBRs uložený v R-strom struktuře (Převzato z [1]).

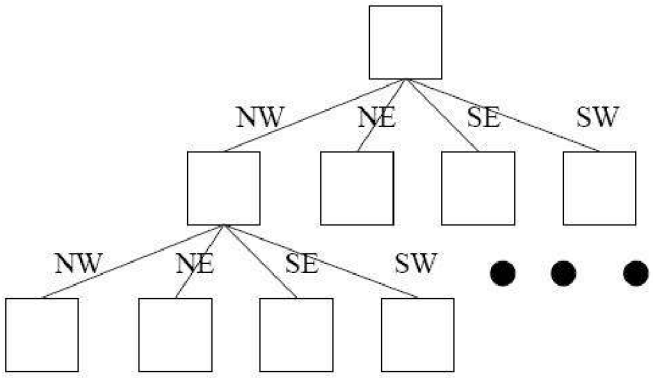
3.2.3 Quad-strom

Dalším typem prostorové indexové struktury podporované v mnoha RSŘBD je Quad-strom. Quad-strom je index založený na rozdělování, kde objekty jsou rekurzivně rozdělovány do kvadrantů. Prostorový objekt je reprezentován kvadrantem, ke kterému náleží. Struktura Quad-strom indexu je zobrazena na obrázku 9. Celý datový prostor je reprezentován kořenovým uzlem. Druhá úroveň representuje rozdělení datového prostoru do kvadrantů a následující úrovně rozdělují každý z kvadrantů do menších kvadrantů. Uzly listové úrovně ukazují na skutečné prostorové objekty uvnitř kvadrantu. V čase dotazu je strom navigován následnými uzly, ve kterých okno dotazu protíná kvadrant, dokud nejsou ukazatelé listové úrovně dosaženy. R-strom má sklon být více účinný než-li Quad-strom pro obecné použití prostorových aplikací.

Toto chování je částečně způsobeno schopností přizpůsobení dat R-stromu k reprezentaci, tj. index je založen na skutečných prostorových objektech uložených v databázi, nikoliv na dělení prostoru. Quad-strom nabízí lepší výkon v oblasti obrázkově založených aplikací z důvodu existence dat po celém prostoru.

Stejně jako u prostorových dat, zde existuje potřeba včlenit podporu pro časové datové typy a indexové struktury uvnitř RSŘBD. Standardní databáze podporuje pohled na data pouze v jediném časovém bodě, tj. může odpovídat dotazům typu „Jaký je krevní tlak pacienta?”. Uložení historické informace nebo požadavek na validitu informace v budoucnu je důležitý v mnoha oblastech. Například dotaz typu „Jaký byl krevní tlak pacienta minulý rok?”. Možným přístupem je Relational-Interval

strom (RI-Strom), který integruje časový index do RSŘBD modifikováním R-stromu za účelem účinného indexování jednodimenzionálních časových intervalů. Do RSŘBD Oracle může být přístup integrován použitím databázové rozšiřovací technologie, která umožňuje mapování mezi API indexem a strukturou dotazu, základními databázovými tabulkami a indexovými strukturami.



Obr. 9 Quad-strom (Převzato z [1]).

3.3 Alternativy RSŘBD pro časoprostorová data

3.3.1 Úvodní rozčlenění alternativ

Je zde několik možností, podle kterých může být problém indexování časoprostorových dat uvnitř RSŘBD klasifikován. V této sekci popíši implementačně orientovanou klasifikaci. Zaměřím se na různé volby integrování interního mechanismu podpory uvnitř RSŘBD. Rozlišujeme tři alternativy pro poskytování RSŘBD podpory pro časoprostorová data:

- Přepracování problému způsobem takovým, že mohou být použity existující RSŘBD prostředky pro časoprostorová data
- Navrhnutí nové časoprostorové přístupové metody a volně navázat s RSŘBD přes obalové funkce, které mohou být volány uvnitř SQL dotazů
- Navrhnutí nové časoprostorové přístupové metody a pevně navázat s RSŘBD použitím přirozených RSŘBD prostředků (tj. relační tabulky a existující indexové podpory)

3.3.2 Přepracování problému

Mnoho RSŘBD (např. Oracle) poskytuje podporu pro prostorová data (např. již zmíněné R-strom a Quad-strom indexy). Je několik možností, jak přepracovat problém časoprostorového indexování dat využitím podpory RSŘBD pro prostorová data. Možná nejvíce přímým východiskem je modelovat čas jako přídavnou prostorovou dimenzi, přičemž může být znovu užito existující podpory pro prostorové

indexy. Kombinací dvojdimenzionální prostorové domény a jednodimenzionální časové domény vznikne trojdimenzionální R-strom (viz. obrázek 8). Tento R-strom může být přizpůsoben k podpoře časoprostorového indexování použitím tohoto přístupu. Typická konstrukce R-stromu vyžaduje, aby každá dimenze byla ohraničena. Důležitým faktem je, že typické prostorové struktury poskytované v RSŘBD jsou laděny pro dvojdimenzionální data. Například trojdimenzionální R-strom v Oracle podporuje pouze malou podmnožinu funkčnosti dotazu a možnosti optimalizace, které jsou běžně poskytované v datech dvojdimenzionálních. Přístup Z-uspořádání s B-stromem a LRS strategie založená na R-stromu jsou příklady technik umožňující indexování časoprostoru uvnitř RSŘBD přepracovávající problém do prostorového datového problému, který může být řešen užitím existující RSŘBD technologie.

3.3.3 Volně vázané

Uvnitř RSŘBD může být integrováno mnoho struktur založených na R-stromu, o kterých jsme již diskutovali užitím přístupu volného vázání, tj. použitím existující implementace časoprostorové přístupové metody a následně zabalením do aplikačního programového rozhraní (API). Bohužel v současné době nejsou v praxi využitelné žádné takové volné vázané přístupy navržené pro časoprostorové oblasti, částečně z důvodu vážných nevýhod. Časoprostorové přístupové metody mají limitovanou schopnost řídit vyrovnávací paměti, pracovat s diskovými stránkami nebo přistupovat do vnitřních databázových struktur kvůli volnému vázání. Při návrhu časoprostorové přístupové metody musíme sami integrovat RSŘBD rysy (například souběžné řízení a zotavovací mechanismus).

3.3.4 Pevně vázané

Pevné vázání časoprostorové přístupové metody uvnitř RSŘBD zahrnuje poskytnutí relačního mapování časoprostorové přístupové metody. Relational Interval strom (RI-strom) a MAP21 jsou techniky využívající takový přístup v časové doméně. Klíčem obou těchto přístupů je přepočet logické časové indexové struktury do implementace spoléhající se na předcházející RSŘBD funkcionalitu. RI-strom využívá databázovou rozšiřovací technologii, zatímco MAP21 používá datovou překladovou funkci k mapování počátečních a koncových bodů časového intervalu do jednodimenzionálních hodnot, které mohou být indexovány B+stromem.

Tento přístup se také používá v prostorové doméně k mapování Quad-stromu a R-stromu do RSŘBD. R-strom v Oracle je čistě logická struktura, jehož základní implementace je založena na relačních tabulkách a B+strom indexech. Podobně je tomu tak i v případě Quad-stromu, kde základní mechanismus spoléhá na Z-uspořádání, relační tabulky a B-strom index.

Největším problémem integrace existující časoprostorové přístupové metody uvnitř RSŘBD je enormní mezera mezi konceptuálním modelem a relačním mapováním. Mapování časoprostorové přístupové metody uvnitř RSŘBD není obecně obvyklé.

4 Databázový systém Oracle

4.1 Úvodem

Oracle, podporující prostorová data vyžaduje použití Enterprise Edition (EE) Oracle 10g, který má s verzí Standart stejné základní rysy. Nicméně několik pokročilých rysů, jako jsou například rozšířené datové typy, jsou dostupné pouze v EE. Například pro rozdělování tabulek musíme mít EE a aktivní volbu rozdělování.

4.2 Oracle podporující prostorová data

Poskytuje SQL schémata a funkce, které usnadňují vykonávání běžných operací jako jsou uložení, získání, aktualizace a dotazování na souhrnné prostorové rysy v Oracle databázi. Sestává z:

- Schéma (MDSYS), které stanovuje úložiště, syntaxi a sémantiku podporovaných geometrických datových typů
- Mechanismus indexování
- Operátory, funkce, procedury pro vykonávání rozsahových dotazů, prostorového spojování dotazů a jiné prostorové analytické operace
- Funkce a procedury pro zlepšení funkčnosti a ladící funkce
- Topologii datového modelu pro práci s daty okolo uzlů, hran a aspektu v topologii

4.3 Objektově relační model

Prostorová komponenta prostorového charakteru je geometrická reprezentace jeho tvaru v nějakých souřadnicích. Takovou komponentu označujeme jako geometrii, jejíž model reprezentujeme využitím podpory objektově relačního modelu reprezentující geometrii. Tento model ukládá veškerou geometrii v přirozeném prostorovém datovém typu Oracle pro vektorová data SDO_GEOMETRY. Oracle tabulka může obsahovat jeden nebo i více SDO_GEOMETRY sloupců. Výhodami objektově relačního modelu jsou:

- Podpora více geometrických typů, zahrnujících oblouky, kruhy, složené polygony, složené přímky řetězců a optimalizované obdélníky
- Ulehčené použití při vytváření a údržbě indexů a také při vykonávání prostorových dotazů
- Modelovaní geometrie v jediném sloupci
- Optimální výkon

4.3.1 Výhody objektově relačního modelování

V zásadě je objektový model Oracle podobný mechanismu definování tříd v jazyku C++ či Java. Objekty umožňují jednoduše modelovat složitosti reálného světa a jeho logiku. Znovupoužitelnost objektů umožňuje vytvářet databázové aplikace rychlejší a mnohem výkonnější. Oracle dovoluje aplikačním vývojářům přistupovat přímo do datových struktur používaných jejich aplikacemi. Žádná mapovací vrstva není potřeba mezi objekty na straně klienta, sloupci relační databáze ani tabulkami, obsahujícími data. Abstrakce objektu a zapouzdření chování objektu dělá aplikace srozumitelné a udržovatelé.

Databáze jako taková obsahuje pouze data. Naproti tomu objekty mohou zahrnovat i operace, které je potřeba vykonávat nad daty. Aplikace může jednoduše volat příslušné metody za účelem získání dat z databáze. Jednou vytvořená metoda objektu je dostupná pro jakoukoliv aplikaci. Vývojáři tedy nemusí znovu vytvářet podobné metody zvlášť v každé aplikaci.

Oracle implementuje objektový typ rozšířením relačního modelu. Rozhraní objektového typu podporuje standardní relační databázovou funkcionalitu (dotazy SELECT…FROM…WHERE, rychlé potvrzování, zálohování, obnovu a mnohé další).

4.4 Prostorová data

Oracle, podporující prostorová data je navržen takovým způsobem, aby učinil prostorové datové řízení jednodušší pro uživatele lokalizačně otevřených aplikací a geografických informačních systémových (GIS) aplikací. Jakmile jsou jednou prostorová data vložena do databáze, poté s nimi může být jednoduše zacházeno.

Například automapa je dvojdimenzionální objekt, který obsahuje body, čáry a polygony reprezentující města, silnice či politické ohraničení jako jsou státy a provincie. Automapa je tedy vizualizace geografických informací. Lokace měst, cest a hranic, které existují na povrchu země, jsou promítnuty na dvojdimenzionální displej nebo arch papíru, zachovávající relativní pozici a vzdálenost poskytnutých objektů.

Prostorová data (jiná než GIS), která mohou být uložena použitím prostorových souřadnic, zahrnují data z computer-aided design (CAD) a computer-aided manufacturing (CAM) systémy. CAD/CAM systémy pracují v menším měřítku (například automobilový motor nebo tištěná deska), namísto operování nad objekty v geografickém měřítku.

Složitost dat je nezávislá na absolutním měřítku, například tištěný obvod má pravděpodobně několik tisíc objektů vyleptaných na jeho povrchu. Informace shromážděné v této malé oblasti mohou být daleko více složitější nežli detailní pohled na povrch silnice.

Všechny tyto aplikace mohou ukládat, získávat, aktualizovat nebo se dotazovat na rysy, které mají prostorové i neprostorové atributy. Příkladem neprostorových atributů jsou jméno, typ.

Prostorové atributy si můžeme vyložit jako souřadnice geometrie nebo vektorově založené reprezentace tvaru vlastnosti.

4.5 Datový model

Prostorový datový model je hierarchická struktura sestávající se z elementů, geometrií a vrstev. Vrstvy se skládají z geometrií, ve kterých se vyskytují elementy.

Element

Element je základní stavební blok geometrie. Podporované prostorové typy elementu jsou body, úsečky a polygony. Každá souřadnice v elementu je uložena jako pár x, y hodnot. Data bodů jsou tvořena jednou souřadnicí. Data úseček se sestávají ze dvou souřadnic reprezentující element. Polygony jsou tvořeny třemi a více souřadnicemi, jeden vrchol pro každý segment úsečky polygonu. Souřadnice jsou definovány okolo polygonu proti směru hodinových ručiček (pro vnější polygonový kruh) nebo po směru hodinových ručiček (pro vnitřní polygonový kruh).

Geometrie

Geometrie (neboli geometrický objekt) reprezentuje prostorový rys a je modelován jako uspořádaná množina primitivních elementů. Geometrie se může skládat z jednoho elementu, který je instancí jednoho z podporovaných primitivních typů nebo stejnorodé či různorodé kolekce elementů. Například vícenásobný polygon použitý k reprezentaci množiny ostrovů je stejnorodá kolekce naproti tomu heterogenní kolekce zobrazuje elementy různých typů (body, polygony).

Vrstva

Vrstva je kolekce geometrií mající stejnou množinu atributů. Jedna vrstva v GIS může zahrnovat topografické rysy, zatímco jiná popisuje hustotu zalidnění a další popisuje síť silnic a mostů v oblasti. Geometrie a přidružený prostorový index jsou uloženy v databázi ve standardních tabulkách pro každou vrstvu.

4.6 Detailní pohled na typ SDO_GEOMETRY

V Oracle, podporujícím prostorová data je popis geometrie prostorového objektu uložen v jednom řádku jednoho sloupce objektového typu SDO_GEOMETRY v uživatelem definované tabulce. Tabulka, která má sloupec typu SDO_GEOMETRY musí mít další sloupec nebo množinu sloupců definující unikátní primární klíč pro tuto tabulku. Tabulky tohoto druhu také někdy nazýváme jako prostorové tabulky nebo tabulky prostorových geometrií. Oracle, podporující prostorová data definuje objektový typ SDO_GEOMETRY následujícím způsobem:

```
CREATE TYPE sdo_geometry AS OBJECT (
SDO_GTYPE     NUMBER,
SDO_SRID      NUMBER,
SDO_POINT     SDO_POINT_TYPE,
SDO_ELEM_INFO SDO_ELEM_INFO_ARRAY,
SDO_ORDINATES SDO_ORDINATE_ARRAY);
```

Dále definuje typy SDO_POINT_TYPE, SDO_ELEM_INFO_ARRAY a SDO_ORDINATE_ARRAY, které jsou použity v definici SDO_GEOMETRY následujícím způsobem:

```
CREATE TYPE sdo_point_type AS OBJECT (
X            NUMBER,
Y            NUMBER,
Z            NUMBER);
CREATE TYPE  sdo_elem_info_array AS VARRAY (1048576) of NUMBER;
CREATE TYPE  sdo_ordinate_array AS VARRAY (1048576) of NUMBER;
```

Maximální počet čísel v SDO_GEOMETRY je 1 048 576, tedy počet vrcholů v objektu SDO_GEOMETRY závisí na počtu dimenzí na vrchol (349 525 pro tři dimenze).

Atribut SDO_GTYPE indikuje typ geometrie. Všechny geometrie jednoho sloupce musí mít stejný počet dimenzí. Nemůžeme kombinovat například dvojdimenzionální a trojdimenzionální data ve stejné vrstvě. Hodnota atributu se skládá ze čtyř číslic ve formátu *dltt,* kde:

- *d* identifikuje počet dimenzí (dvě, tři nebo čtyři)
- *l* identifikuje LRS míru pro tří dimenzionální LRS geometrii, kde dimenze (třetí nebo čtvrtá) obsahuje hodnotu míry. V případě nepoužití LRS nebo přijmutí standardně poslední dimenze jako míry pro LRS geometrii, specifikujeme 0
- *tt* identifikuje typ geometrie (00 až 07, 08 až 99 rezervované pro budoucí využití)

Atribut SDO_SRID může být použit ke specifikování souřadného systému asociovaného s geometrií. V případě hodnoty „null“ není přiřazen žádný souřadný systém. Všechny geometrie ve sloupci musí mít stejnou hodnotu SDO_SRID.

Dalším rozebíraným atributem je SDO_POINT definující používání objektového typu SDO_POINT_TYPE, který má atributy X, Y a Z (všechna jsou čísla). Pokud pole SDO_ELEM_INFO a SDO_ORDINATES jsou nastaveny na null a atribut SDO_POINT není null, pak X a Y hodnoty jsou považovány za souřadnice pro geometrii bodu. Jinak je atribut SDO_POINT ignorován. Pokud budou ve vrstvě pouze geometrie bodu, je doporučováno ukládat geometrie bodu v SDO_POINT (jsou vytvářena optimální úložiště). Tento atribut se nesmí použít v případě definování LRS bodů.

Atribut SDO_ELEM_INFO je definován proměnnou délkou pole čísel. Atribut říká, jak interpretovat souřadnice uložené v SDO_ORDINATES atributu. Každá trojice čísel je interpretována tímto způsobem:

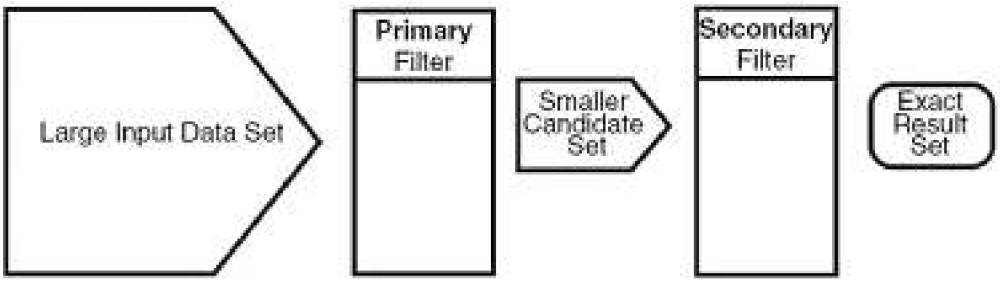
- SDO_STARTING_OFFSET indikuje ofset uvnitř pole SDO_ORDINATES, kde je první souřadnice uložena pro tento element. První hodnota začíná na jednotce
- SDO_ETYPE indikuje typ elementu
- SDO_INTERPRETATION znamená v závislosti zda je nebo není SDO_ETYPE složený element. Pokud se jedná o složený element, pole specifikuje kolik dalších trojic hodnot jsou součástí elementu. V případě, že se nejedná o složený element, pole specifikuje kolik následných souřadnic je interpretováno pro tento element

Atribut SDO_ORDINATES je definován proměnnou velikostí pole čísel, která vytváří ohraničení prostorového objektu. Toto pole musí být vždy použito ve spojení s SDO_ELEM_INFO polem. Hodnoty jsou seřazeny podle dimenze. Počet dimenzí přidružených ke každému bodu je uložen v tabulce USER_SDO_GEOM_METADATA.

4.7 Model dotazu

Oracle, podporující prostorová data používá dvojvrstvý model dotazu za účelem provedení prostorového dotazování a spojení. Výstup těchto dvou kombinovaných operací vynáší přesnou množinu odpovědí. Tyto operace označujeme jako *primární* a *sekundární* filtrovací operace.

- *Primární filtr* dovoluje rychlou selekci kandidátních záznamů před sekundárním filtrem. Primární filtr porovnává blízkosti geometrií za účelem redukování výpočetní složitosti. Je proto tedy považován za výpočetně méně náročnější filtr. Sekundárnímu filtru předává poměrně přesnou množinu výsledků
- *Sekundární filtr* požaduje na vstupu přesné výpočty geometrií, které jsou předány z primárního filtru. Výstupem sekundárního filtru je přesná odpověď na prostorový dotaz. Je výpočetně náročnější, ale tato náročnost je redukována tím, že pracuje pouze s malou množinou (výsledky primárního filtru) namísto celé datové množiny

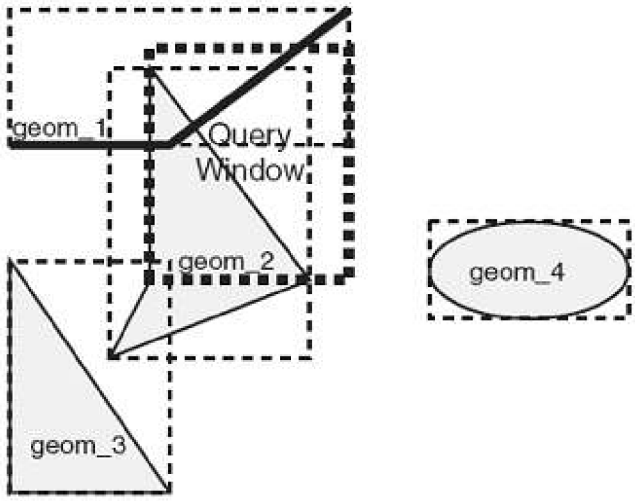


Obr. 10 Model dotazu (Převzato z [2]).

Jak je vidět v obrázku 10, operace primární filtr na velké vstupní datové množině produkuje menší datovou množinu, která obsahuje nejméně přesnou množinu výsledků, avšak může obsahovat daleko více záznamů. Oracle, podporující prostorová data používá prostorový index k implementaci primárního filtru. V některých případech je použití primárního filtru dostačující. Primární filtr velmi rychle vrací množinu výsledků dotazu, na které poté mohou aplikace použít rychlé rutiny k zobrazení cílové oblasti. Účel primárního filtru je rychle vytvořit podmnožinu dat a redukovat zátěž zpracování na sekundárním filtru. Primární filtr by proto měl být tak účinný jak je to jen možné.

V prostorovém R-strom indexu je každá geometrie reprezentována jeho minimálním ohraničením obdélníkem (MBR), uvažujme následující vrstvu obsahující několik objektů v obrázku 11. Každý objekt je popsán jeho geometrickým jménem (geom_1 pro úsečku, geom_2 pro čtyřstranný polygon, geom_3 pro trojúhelník a geom_4 pro elipsu) a přerušovanou čarou je reprezentován MBR okolo každého objektu.

Typický prostorový dotaz se ptá všech objektů, zda leží uvnitř okna dotazu definující ohraničení. Dynamické okno dotazu odkazuje na obdélníkovou oblast, která není definována v databázi. Na obrázku 11 vidíme okno dotazu reprezentované jako oblast vymezenou tučnými tečkami



Obr. 11 Vrstva ukazující okno dotazu (Převzato z [2]).

Při operaci spojení, Oracle porovnává všechny geometrie jedné vrstvy ke všech geometriím jiné vrstvy. To je zásadní rozdíl oproti oknu dotazu, kde je porovnávána jedna geometrie vůči všem geometriím vrstvy. Prostorového spojení může být použita k zodpovězení otázky charakteru „Které dálnice protínají národní parky?“.

Detailněji bude model filtrování a samotné filtry v závislosti na implementovaném indexu popsány v kapitole 5.4.

4.8 Prostorové vztahy

Oracle, podporující prostorová data používá sekundární filtr pro určení prostorových vztahů mezi položkami v databázi. Prostorový vztah je založen na poloze geometrie. Nejběžnější prostorové vztahy jsou založeny na topologii a vzdálenosti. Ohraničení oblasti se sestává z množiny křivek, které oddělují oblast od zbytku souřadného systému. Dále vnitřek oblasti je definován všemi body z oblasti, které nejsou jejím ohraničením. Říkáme, že dvě oblasti jsou přilehlé, pokud sdílejí část ohraničení a zároveň nesdílí žádné body jejich vnitřků.

Vzdálenost mezi dvěma prostorovými objekty je definována jako minimální vzdálenost mezi jakýmikoliv body v nich obsažených. Dva objekty jsou uvnitř dané vzdálenosti, pokud je jejich vzdálenost menší než daná vzdálenost. Oracle podporující prostorová data nabízí několik sekundárních filtrovacích metod pro určení prostorového vztahu:

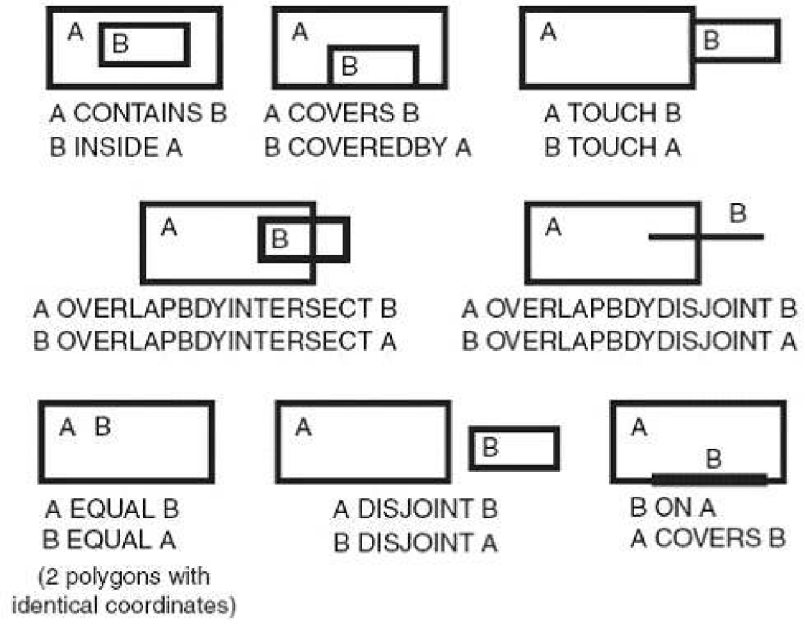
- SDO_RELATE - ohodnotí topologická kritéria
- SDO_WITHIN_DISTANCE - určí, zda dva prostorové objekty jsou uvnitř specifikované vzdálenosti
- SDO_NN - identifikuje nejbližší sousedy prostorového objektu

SDO_RELATE operátor implementuje devíti-průnikový model kategorizující topologické vztahy mezi body, úsečkami a polygony. Každý prostorový objekt má vnitřek, ohraničení a zevnějšek. Ohraničení se sestává z bodů nebo čar, které oddělují vnitřek od zevnějšku. Ohraničení úseček řetězců se sestává z jejich koncových bodů, nicméně jestliže se koncové body překrývají (tzn. stejné body), úsečka řetězce nemá ohraničení. Používají se následující pojmenování pro vztahy:

- disjoint – ohraničení a vnitřky nejsou v průniku
- touch – ohraničení jsou v průniku, ale vnitřky nejsou
- overlapbdydisjoint – vnitřek jednoho objektu protíná ohraničení a vnitřek druhého objektu, ale obě ohraničení nejsou v průniku. Tento vztah se vyskytuje, když čára začíná mimo polygon a končí uvnitř polygonu.
- equal – dva objekty mají totožné ohraničení i vnitřek
- contains – vnitřek a ohraničení jednoho objektu je kompletně obsažen ve vnitřku jiného objektu
- covers – vnitřek jednoho objektu je kompletně obsažen ve vnitřku nebo ohraničení jiného objektu a jejich ohraničení se protíná
- inside – protějšek CONTAINS. A INSIDE B určuje B CONTAINS A
- coveredby – protějšek COVERS. A COVEREDBY B určuje B COVERS A

- on – vnitřek a ohraničení jednoho objektu je na ohraničení jiného objektu (a druhý objekt pokrývá první objekt). Tento vztah nastává, když čára je na ohraničení polygonu.
- anyinteract – objekty jsou disjunktní

SDO_WITHIN_DITANCE operátor zjišťuje, zda dva prostorové objekty A, B jsou uvnitř specifikované vzdálenosti navzájem jeden od druhého. Tento operátor nejprve vybuduje vzdálenostní buffer *Db* okolo referovaného objektu B. Poté kontroluje, zda jsou A a *Db* disjunktní. Vzdálenostní buffer objektu se sestává ze všech bodů uvnitř dané vzdálenosti od objektu.



Obr. 12 Topologické vztahy (Převzato z [2]).

SDO_NN (nearest neighbour) operátor vrací počet objektů, které jsou nejblíženi specifikované geometrii (např. nejbližších 6 hotelů v okolí hlavního vlakového nádraží). K určení specifikované vzdálenosti dvou geometrických objektů jsou vzaty v úvahu nejbližší dva body z povrchu každého ze dvou porovnávaných objektů.

4.9 Indexování prostorových dat

Prostorový index stejně jako i jiné indexy poskytuje mechanismus limitovaného hledání, ale v tomto případě je mechanismus založen na prostorovém kritériu jako jsou průnik a omezení. Prostorový index je potřebný pro:

- Nalezení objektů uvnitř indexovaného datového prostoru, které jsou v interakci s daným bodem nebo oblastí našeho zájmu (okno dotazu)
- Nalezení dvojice objektů uvnitř dvou indexovaných datových prostorů, které jsou navzájem v prostorové interakci (prostorové spojení)

Položky indexu mohou být seřazeny použitím lineárního uspořádání a souřadnice pro geometrii mohou být párem integer, float nebo double čísel. Oracle, podporující prostorová data dovoluje použít R-strom index (výchozí) nebo Quad-strom index. V praxi se převážně využívá mechanismus indexovaní R-stromu.

4.9.1 R-strom index

Po nahrání dat do prostorových tabulek hromadným nebo transakčním importováním by měl být vytvořen prostorový index nad tabulkami pro efektivní přístup k datům. Pokud není vytvořen prostorový index, nelze využívat prostorové operátory. Ukázka příkazu vytvoření R-strom indexu pojmenovaného area_idx:

```
CREATE INDEX area_idx ON areas (area_geom) INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```

Pokud vytvoření indexu není z jakéhokoliv důvodu kompletní, index je označen jako neplatný a musí být následně smazán příkazem:

```
DROP INDEX area_idx;
```

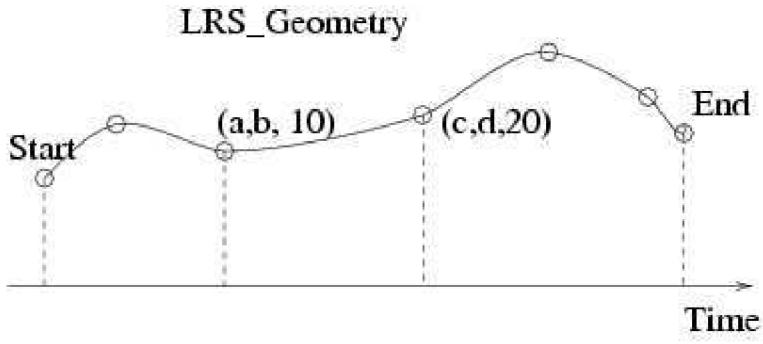
Prostorové indexy mohou být postaveny na dvou, třech nebo čtyřech dimenzích dat. Výchozí počet dimenzí jsou dvě. Pakliže jsou data definována na více než dvou dimenzích, můžeme použít klíčové slovo SDO_INDX_DIMS pro specifikování počtu dimenzí na kterých bude postaven index. Nicméně pokud je prostorový index postaven na více než dvou dimenzích vrstvy, pouze jediný prostorový operátor, který je možné používat na této vrstvě, je SDO_FILTER (primární filtr). Operátory sekundárního filtru SDO_RELATE, SDO_NN a SDO_WITH_DISTANCE nelze v tomto případě využívat.

Značný počet operací vložení (aktualizací) a mazání nepříznivě ovlivňuje R-strom index. Což může vést ke snížení kvality struktury R-stromu a v neposlední řadě negativnímu ovlivnění výkonu celého dotazu. R-strom je hierarchická struktura s uzly v různých výškách stromu. Oblast pokrytá ve stupni 0 reprezentuje oblast obsazenou minimálním ohraničením obdélníků dat geometrií, oblast v úrovni 1 indikuje oblast pokrývající uzly listové úrovně R-stromu.

4.10 Možnosti indexů pro časoprostorová data

4.10.1 LRS založený R-strom

Nebo také jinak označovaný R-strom + časový B+strom přístup. Možným řešením je užít navržený Linear Referencing System (LRS) k indexaci třetí (časové) dimenze časoprostorových objektů. Výhodou užívání LRS je, že časová dimenze může být indexována odděleně od prostorové bez nutnosti vytvářet oddělené skladovací umístění (sloupec) pro uložení časové informace. Obrázek 13 ukazuje LRS_Geometry objekt reprezentující trajektorii pohybujícího se objektu. Bod (a, b, 10) reprezentuje objekt na pozici (a, b) v čase 10. Počáteční a koncový čas trajektorie může být získán a indexován použitím funkcí poskytovaných v LRS balíčku. Prostorová komponenta trajektorie je poté indexována odděleně od časové použitím standardního dvojdimenzionálního R-stromu. Klíčovou výhodou užití LRS k indexaci časoprostorových dat je, že časoprostorové objekty mohou být uloženy jako jediný objekt v databázové tabulce.

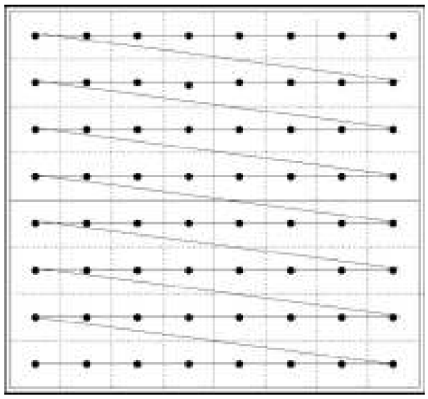


Obr. 13 Příklad časoprostorové trajektorie modelované užitím LRS (Převzato z [1]).

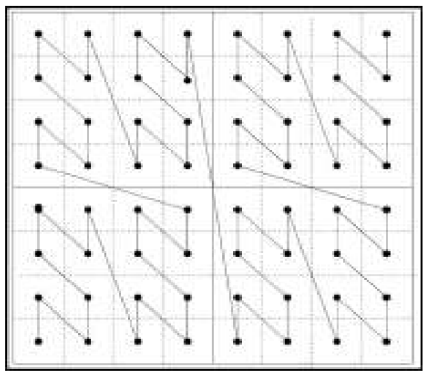
4.10.2 Z-uspořádání a B+strom přístup

Další volbou, které lze využít k indexování časoprostorových dat je užitím vyplňující křivky. Tyto křivky nejsou součástí Oracle, podporující prostorová data a musejí být implementovány odděleně. Daný datový prostor je rozdělován do pravoúhlých buněk a v tomto prostoru může být vyplňující křivka myšlena jako nit, která navštíví každou buňku v mřížce pouze jednou. Jedná se o takové lineární uspořádání na vícedimenzionálním prostoru. Protože B+strom index je neodmyslitelně lineární (jednodimenzionální), vyplňující křivka může být užita k mapování vícedimenzionálních dat do jednodimenzionálního prostoru, na kterém již může být vytvořen B+strom index. Objekty mohou být reprezentovány číslem buňky definované křivky, namísto používání jejich přesných prostorových souřadnic. Obrázek 14 ilustruje nejznámější používané vyplňující křivky mapující dvojdimenzionální prostor do jednodimenzionálního prostoru. Jmenovitě Sweep, Z a Gilbert křivky. Mapování musí

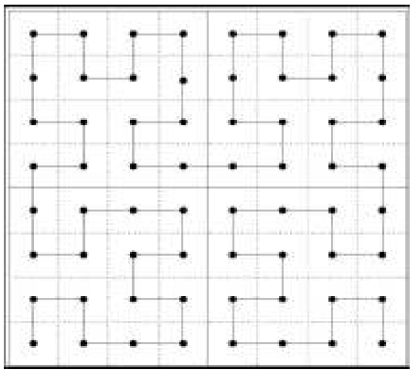
zachovávat prostorové umístění, ve smyslu takovém, že body blízko u sebe ve dvojdimenzionálním prostoru, by měly být ještě blíže v prostoru jednodimenzionálním definovaném křivkou. Každá z uvedených křivek definuje rozdílné lineární uspořádání na prostoru s unikátními vlastnostmi zachování umístění.



Sweep



Z-křivka



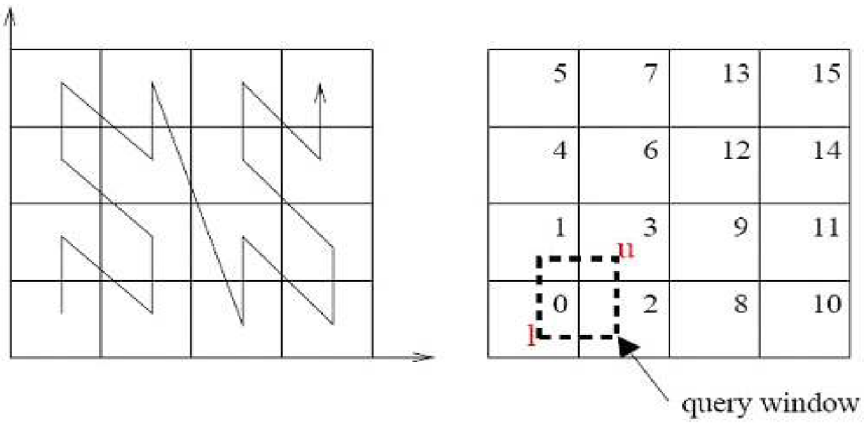
Hilbert

Obr. 14 Vyplňující křivky (Převzato z [1]).

V ideálním provedení chceme dva body, které jsou blízko u sebe v euklidovském prostoru stejně blízko v lineárním uspořádání definovaném vyplňující křivkou. Také čísla buněk by měla být jednoduše spočitatelná, protože vložení, aktualizace dat a dotazy budou požadovat opakované výpočty čísel buněk. Typicky zde nastává kompromis mezi kvalitou zachováním prostorového umístění a

složitostí výpočtu křivky. Sweep křivka nezachovává prostorové umístění obzvláště dobře, ale umožňuje jednoduchý výpočet. Z-křivka je takto označena z důvodu tvaru její cesty „Z“. Výpočet je velmi efektivní použitím bitového prokládacího procesu a poskytuje dobré vlastnosti zachování prostorové lokality. Hilbert křivka nabízí mírně lepší prostorové shlukování, nežli je tomu v případě Z-křivky, ale výpočet je mnohem nákladnější. Vyplňující křivky je možné použít rekurzivně a tím pokrýt i vícedimenzionální prostor.

Použitím Z-uspořádání a B-strom přístupu je prostorová složka záznamů aproximována buňkami definovanými podle lineárního Z-uspořádání a je vytvořen B-strom index nad Z-hodnotami a časem. V čase dotazu je nezbytné mapovat prostorovou složku dotazu do jednodimenzionálního prostoru. Jak ukazuje obrázek 15, okno dotazu se stává rozsahem dotazu na lineárním uspořádání. Nalezení všech položek (Z-hodnot) v rozsahu *[l, u]*, kde *l* = nejmenší Z-hodnota okna (spodní levý roh) a *u* = největší Z-hodnota okna (horní pravý roh). Hledání přes index nejprve zúží hledání záznamů uvnitř rozsahu Z-hodnot *[l, u]* a poté uvnitř časového rozsahu okna dotazu.



Obr. 15 Z-uspořádání vyplňující křivka k číslu buňky (Převzato z [1]).

4.10.3 3D Z-křivka

Zachází s časovou dimenzí jako třetí prostorovou dimenzí. Trojdimenzionální vyplňující křivka může být užita pro časoprostorové účely indexovaní. Zacházet s časem jako prostorovou dimenzí za účelem vytvoření trojdimenzionální Z-křivky ignoruje unikátní vlastnosti časové dimenze, tj. čas je monotónně rostoucí. Mimoto je čas poté neohraničený, ale vyplňující křivky mohou být počítány pouze uvnitř fixní prostorové oblasti. Vytvoření trojdimenzionální křivky zahrnuje dělení času do ohraničených intervalů, ve kterých již může být počítána Z – křivka.

4.10.4 Quad-strom

Oracle podporující prostorová data poskytuje vestavěnou podporu prostorového Quad-strom indexu. Vytvoření Quad-stromu je velmi podobné jako v případě R-stromu. Standardně, kdykoliv uživatel vytvoří prostorový index, tak je implicitně vytvořen R-strom index. Specifikováním úrovně rozdělování prostoru (SDO_LEVEL) je vytvořen Quad-strom index namísto R-strom indexu. Index Quad-strom nad prostorovou komponentou dat a časový B-strom indexe by měli dle předpokladu dosahovat slabšího výkonu ve srovnání s R-strom indexem. Dále dodejme, že Quad-strom v Oracle nepodporuje trojdimenzionální data.

5 Příprava pro experimenty

Před samotným provedením experimentů porovnání různých způsobů indexování časoprostorových dat je třeba nejdříve navrhnout vhodnou strukturu databáze pro uložení těchto dat (trajektorií) v databázovém systému Oracle 10g.

Dalším bezprostředně navazujícím krokem v části příprav bylo získat vhodný vzorek časoprostorových dat (co možná nejvíce odpovídající reálným vlastnostem a situacím), který by se dal využít pro účely experimentů. Porovnání a následné zhodnocení bude probíhat na velké datové množině obsahující stovky tisíc záznamů pohybujících se bodů v čase.

V podkapitole 5.4 bude představen detailní pohled na implementaci indexů Quad-strom a R-strom a jejich způsob zpracování.

5.1 Použité nástroje

Popis použitých nástrojů popíši chronologicky tak, jak jsem se s nástroji postupně seznamoval a používal je při řešení projektu.

SQL Developer je IDE prostředí volně dostupné ke stažení na webových stránkách Oracle. Prostředí po připojení k serveru nabízí příjemné uživatelské rozhraní a přehledné zobrazení všech dat v databázi (samotných dat, procedur, indexů, uživatelem definovaných datový typů, atd.) a mnoho dalších užitečných detailních informací. Do pracovní oblasti lze zadávat příkazy SQL (PL/SQL), které jsou zpracovány, a jejich odezva je zobrazena v samostatném okně.

Generátor časoprostorových dat Network-Based Generator of Moving Objects [6] je nástroj implementovaný v programovacím jazyce Java pro generování rozsáhlých datových množin sestávajících se z pohybujících se objektů v čase. Detailněji je tento nástroj popsán v kapitole 5.3.

SQL*Plus je jednoduchá a výkonná konzole databázového systému Oracle jejímž prostřednictvím lze po připojení k serveru zadávat příkazy do příkazové řádky a efektivně pracovat s databází. Této konzole jsem využil především pro vložení rozsáhlých časoprostorových dat do databáze.

PHP (Hypertext Preprocesor) je oblíbený skriptovací jazyk na straně serveru, který jsem při řešení projektu využil ke konverzi výstupních dat generátoru Network-Based Generator of Moving Objects na formát korespondující s návrhem použité databázové struktury.

5.2 Návrh struktury databáze

Při vytváření návrhu objektově-relační databáze shromažďující údaje o pohybujících se objektech bylo použito uživatelsky definovaných datových typů (objektů Oracle), které jsou jedním ze základních

rysů Oracle 10g. Významnou výhodou tohoto rysu je, že uživatel může formalizovat datové struktury a operace, které se objevují v aplikacích. Důležitým aspektem práce s objektovými typy je vytvoření metod operujících nad těmito objekty. Pohled z objektově orientované perspektivy nám dovoluje přenést více struktur reálného světa do databázového schématu. Tyto nové objektové datové typy jsou zapsány v jazyce PL/SQL, který je integrován do systému Oracle. Ukázka kódu definující všechny výchozí uživatelsky definované datové typy jsou k nahlédnutí v příloze 1. Definování pouze objektových typů ještě neznamená, že jsme vytvořili databázové tabulky. Vytvoření typu pouze definuje logickou strukturu, v této chvíli nejsou vytvářena fyzická úložiště dat. Fyzické úložiště vznikne až po vytvoření databázové tabulky založené na definovaném objektovém datovém typu popisující strukturu tabulky.

5.2.1 Výchozí struktura databáze

Slovo výchozí je zvoleno záměrně. Protože z důvodu implementování některých indexů bylo třeba tuto strukturu přeorganizovat. Logický návrh databáze pro uložení pohybujících se objektů (trajektorií) je rozveden v následujícím odstavci. V příloze 2 je zobrazen namodelovaný diagram tříd navržené databáze a vazeb mezi nimi. Model je vytvořen v notaci jazyka UML 2.0. Pro lepší pochopení problematiky následujícího výkladu stručně popíši objektové typy vyskytující se v návrhu databáze:

- Region
- MPoint
- TSegment
- Point
- Line

Region si můžeme vyložit jako část nebo celek sledovaného prostoru (např. kamery, které snímají disjunktní prostory místnosti). Sledovaný prostor může být pokryt jedním či několika regiony. Trajektorie je chápána jako dráha pohybujícího se objektu, která může procházet několika regiony, v takovém případě úsek každého regionu nazýváme segment. Přičemž spojením jednotlivých segmentů dostaneme celou žádanou trajektorii. Při přechodu pohybujícího se objektu do jiného regionu je pro trajektorii vytvořen nový segment ihned navazující na předchozí.

V databázi je segment zaveden jako další objektový typ pojmenovaný TSegment. Jak jsme si mohli všimnout, tak ve výčtu objektových typů není uvedena trajektorie. Ta je zkomponována z příslušejících segmentů a vypočítána v čase dotazu. Počátek trajektorie je uchován v atributu start_tseg (reference na bod do objektu Point) objektového typu TSegment.

Objektový typ MPoint jednoznačně identifikuje pohybující se objekt v čase (např. vozidlo, chodec). Tato hodnota by mohla být součástí entity TSegment, ale v takovém případě by docházelo

k nepříjemnému důsledku, tzv. redundanci záznamů (stejně tak je tomu v případě objektu Region). Z tohoto důvodu je tu zvolena forma agregace.

Význam objektového typu Point je zřejmý z názvu. Typ Point uchovává informace o všech bodech v prostoru a čase.

Poslední velmi důležitým typem je Line schraňující informace o úsečkách jednoho segmentu (pozn. segment se sestává z úseček, analogicky k případu trajektorie a segmentů). Pro vytvoření úsečky je zapotřebí dvou bodů. Nastávají dva scénáře tvoření úsečky. V prvém případě jsme ve stavu, kdy k segmentu budeme vytvářet první úsečku. Přečteme ze segmentu počáteční bod (již zmíněný atribut start_tseg) a jako koncový bod nově vytvářené úsečky je použit aktuálně vkládaný. Druhý případ nastává, když jsou k segmentu přiřazeny úsečky a chceme vložit novou. Přečteme poslední bod segmentu (jinými slovy se jedná o koncový bod naposledy vložné úsečky) a použijeme jej jako počáteční bod nové úsečky. Koncovým bodem nově vkládané úsečky je opět použit aktuálně vkládaný a zároveň se tento bod stává i koncovým bodem segmentu (případně trajektorie).

Ke všem definovaným objektovým datovým typům vyjma typu Region (jeho tvar není dopředu znám) jsou definovány procedury nebo funkce v jazyce PL/SQL pro vložení nového záznamu. Operaci vložení pro typ Region nelze zobecnit z důvodu libovolnosti tvaru regionu. Tyto procedury a funkce nám usnadňují práci při vkládání a poskytují nám jednoduché rozhraní (výší formu abstrakce).

5.3 Generování a vložení datové množiny

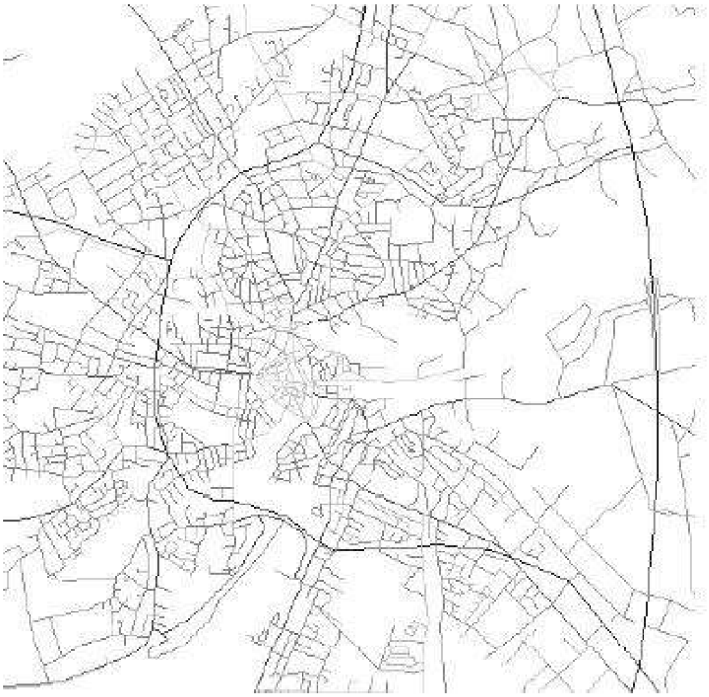
V dnešní době stále není jednoduché obstarat reálné množiny časoprostorových dat ať už z důvodu majetnické či soukromé povahy. Po marné snaze získat reálná data jsem se rozhodl jít cestou vygenerování umělé datové množiny. Prostudoval jsem možnosti syntetických generátorů. Ovšem v oblasti generování takových specifických dat je počet publikací značně omezen. Existují některé zajímavé volně dostupné generátory, které jsou popsány v kapitole 2.1.1, dále jsem objevil a prostudoval generátor Oporto. Po zvážení a přihlédnutí k požadavkům pro účely této práce jsem rozhodl využít generátoru Network-based Generator of Moving Objects [6].

Tento generátor je vytvořen v programovacím jazyce Java a je volně dostupný ke stažení z webových stránek [6]. Jak píše sám autor tohoto generátoru, projekt vznikl zejména z důvodu zvyšující se potřeby rozsáhlých množin časoprostorových dat pro testovací účely a doposud nezaplněné mezery na poli software k této tématice. Program kombinuje reálná data (síť) s umělými daty, která jsou definována specifickými vlastnostmi přímo uživatelem. Generování probíhá ve čtyřech krocích:

- Příprava sítě
- Definování požadovaných parametrů
- Výpočet pohybujících se objektů
- Vygenerování zprávy popisující vygenerované pohybující se objekty

Program načte parametry sítě ze souborů. Pro tento projekt jsem zvolil soubory popisující síť města Oldenburg (Německo), jehož model můžeme vidět na obrázku 16 (opět volně ke stažení včetně dokumentace). Program je možné spustit jako applet v okně prohlížeče nebo standardní aplikaci. Definování parametrů programu je načteno z textového souboru. Aplikace má grafické uživatelské rozhranní, ve kterém lze definovat specifické atributy (počet objektů, rozsah časového razítka, a další). Výsledná data jsou zapsána do běžného textového souboru. Tento soubor jsem automaticky zpracoval skriptem v jazyce PHP za účelem transformování vygenerovaných dat do podoby korespondující s modelem databáze a vytvoření kódu skriptu SQL*Plus vkládajícího celou rozsáhlou množinu záznamů do databáze.

Vygeneroval jsem datovou množinu popisující pohyb 4000 objektů (sestávající se z téměř 450 000 časoprostorových bodů). Jsou vzorkovány dvěma tisíci časovými razítky. To ovšem neznamená, že každý sledovaný objekt je složen ze dvou tisíc diskrétních zaznamenaných bodů v čase. Objekty se chovají reálně a sledují nadefinovanou síť města Oldenburg. Z toho lze odvodit, že objekty jsou tvořeny různými počty bodů tvořící trajektorii objektu. Transformoval jsem časová razítka způsobem takovým, že ve výsledku jsou objekty slevovány v pětiminutových intervalech (od 1.1.2008 12:00:00 do 8.1.2008 10:40:00).



Obr. 16 Město Oldenburg, Německo (převzato z [6])

5.3.1 Problémy s vložením dat do databáze

Při plánování harmonogramu průběhu prací na diplomové práci jsem neuvažoval s rizikem zpoždění ve fázi vkládání dat. Toto zpoždění činilo několikatýdenní průtah (celkem 6 týdnů). Tuto nepříjemnou komplikaci zapříčinila skutečnost, že databáze je navržena objektově (viz. kapitola 5.2). V relačních databázích je vložení záznamů otázkou několika minut. Ovšem již samotné prostorové databáze jsou výpočetně velmi náročné a v kombinaci s navrženou objektovou databází náročnost dále značně vzrostla. V následujících odstavcích popíši nejzávažnější komplikace, které se začali vynořovat na povrch a kterým jsem musel čelit v průběhu těchto 6 týdnů. Řešení jsem hledal zejména na Internetu a ze zkušeností lidí v podobných situacích jako já se snažil poučit. V neposlední řadě jsem konzultoval postup řešení vzniklých problémů s administrátorem školního databázového serveru Oracle.

Problém se zpracováním anonymního bloku

Data jsem do databáze vkládal použitím jednoho rozsáhlého anonymního bloku neboli PL/SQL bloku v SQL*Plus, který ve svém těle obsahoval příkazy volání funkcí a procedur objektů definujících provedení vložení. Těchto příkazů zajišťující vložení množiny časoprostorových dat bylo téměř 450 000 a velikost souboru s tímto blokem dosahovala necelých 33 MB.

Ukázka kódu:

```
BEGIN
     MPoint_typ.insert_mpoint(); COMMIT;
     TSegment_typ.insert_tsegment(1, 1, 6502, 22603, '1.1.2008 12:05:00'); COMMIT;
     MPoint_typ.insert_mpoint(); COMMIT;
     TSegment_typ.insert_tsegment(2, 1, 5657, 16009, '1.1.2008 12:05:00'); COMMIT;
     Line_typ.insert_line(2, 1, 5604, 15982, '1.1.2008 12:10:00'); COMMIT;
     Line_typ.insert_line(1, 1, 6527, 22601, '1.1.2008 12:10:00'); COMMIT;
     MPoint_typ.insert_mpoint(); COMMIT;
     TSegment_typ.insert_tsegment(3, 1, 10860, 6834, '1.1.2008 12:10:00'); COMMIT;
     MPoint_typ.insert_mpoint(); COMMIT;
     TSegment_typ.insert_tsegment(4, 1, 15806, 23086, '1.1.2008 12:10:00'); COMMIT;
     Line_typ.insert_line(4, 1, 15841, 23103, '1.1.2008 12:15:00'); COMMIT;
     Line_typ.insert_line(3, 1, 10819, 6876, '1.1.2008 12:15:00'); COMMIT;
     Line_typ.insert_line(2, 1, 5551, 15956, '1.1.2008 12:15:00'); COMMIT;
     Line_typ.insert_line(1, 1, 6553, 22600, '1.1.2008 12:15:00'); COMMIT;

     -- Následuje několik stovek tisíc příkazů

END
```

Popis problému:

Na tento postup odpovídal server Oracle chybovým hlášením „PLS-00123: program too large (Diana nodes)“. Po vyhledání významu tohoto hlášení jsem zjistil, že konzole SQL*Plus má mnoho striktních omezení, mezi nimiž je i omezení na počet znaků v anonymním bloku, a proto skript nebyl schopen zpracovat blok tak rozsáhlého objemu příkazů.

Problém se zpracováním menších bloků

Dalším přístupem řešení vzniklého problému bylo rozložení značně objemného anonymního bloku do několika stovek menších bloků. Každý blok byl v samostatném souboru a tyto soubory jsem zpracoval způsobem nazývaným vnořené skripty (nesting scripts). To znamená, že jsem z jednoho hlavního (centrálního) skriptu „run.sql“, který byl spuštěn z konzole, volal jednotlivé vnořené skripty.

Ukázka kódu:

Run.sql

```
START insert1.sql
START insert2.sql

--následuje několik stovek takových volání
```

Insert1.sql

```
BEGIN
 MPoint_typ.insert_mpoint(); COMMIT;
 TSegment_typ.insert_tsegment(1, 1, 6502, 22603, '1.1.2008 12:05:00'); COMMIT;
 MPoint_typ.insert_mpoint(); COMMIT;
 TSegment_typ.insert_tsegment(2, 1, 5657, 16009, '1.1.2008 12:05:00'); COMMIT;
 Line_typ.insert_line(2, 1, 5604, 15982, '1.1.2008 12:10:00'); COMMIT;
 Line_typ.insert_line(1, 1, 6527, 22601, '1.1.2008 12:10:00'); COMMIT;

 --následuje několik tisíc příkazů
END
```

Popis problému:

Problém reportovaný databázovým serverem Oracle se objevil po vložení několika tisíc časoprostorových bodů (vykonání několika tisíc příkazů). Chybové hlášení tvaru „Oracle Error :: ORA-02392 exceeded session limit on CPU usage, you are being logged off“ mi sdělilo, že jsem byl odpojen od serveru z důvodu vyčerpání času procesoru vyhrazeného pro jedno sezení (session).

Přístup vedoucí ke správnému řešení

Ukázalo se, že předchozí řešení bylo téměř správné. Pouze bylo třeba ve skriptu „run.sql“ před každým spuštěním vnořeného skriptu navázat spojení se serverem a po vykonání vnořeného skriptu odpojit od serveru a anonymní blok byl nahrazen příkazy „execute“. Tím bylo zajištěno, že nedocházelo k vyčerpání limitu času procesoru na jedno sezení.

Ukázka kódu:

Run.sql

```
connect user/password@dbname
@insert1.sql
disconnect
connect user/password@dbname
@insert2.sql
```

```
disconnect

--následuje několik stovek takových volání
```

Insert1.sql

```
execute MPoint_typ.insert_mpoint();
execute TSegment_typ.insert_tsegment(1, 1, 6502, 22603, '1.1.2008 12:05:00');
execute MPoint_typ.insert_mpoint();
execute TSegment_typ.insert_tsegment(2, 1, 5657, 16009, '1.1.2008 12:05:00');
execute Line_typ.insert_line(2, 1, 5604, 15982, '1.1.2008 12:10:00');
execute Line_typ.insert_line(1, 1, 6527, 22601, '1.1.2008 12:10:00');
execute MPoint_typ.insert_mpoint();
execute TSegment_typ.insert_tsegment(3, 1, 10860, 6834, '1.1.2008 12:10:00');
execute MPoint_typ.insert_mpoint();
execute TSegment_typ.insert_tsegment(4, 1, 15806, 23086, '1.1.2008 12:10:00');
execute Line_typ.insert_line(4, 1, 15841, 23103, '1.1.2008 12:15:00');
execute Line_typ.insert_line(3, 1, 10819, 6876, '1.1.2008 12:15:00');
execute Line_typ.insert_line(2, 1, 5551, 15956, '1.1.2008 12:15:00');
execute Line_typ.insert_line(1, 1, 6553, 22600, '1.1.2008 12:15:00');

--následuje několik tisíc příkazů
```

Zhodnocení přístupu:

Po úspěšném vložení několika desetitisíc časoprostorových bodů skript znovu zhavaroval a server Oracle reportoval hlášení, že došlo k překročení prostorové kvóty v tabulce USERS. Tento problém jsem odstranil po kontaktování administrátora serveru Oracle a požádání o zvýšení dotyčné kvóty. Poté již vkládání probíhalo bez komplikací, co se logiky týče. Bohužel nastaly nové již neřešitelné komplikace, které průběh testů opět ztížily. Jednalo se o časovou náročnost, vkládání dat pro jeden experiment trvalo v některých případech až jedenáct a půl hodiny. Dále pokud během vkládání došlo k neočekávané situaci, pak jsem musel celou množinu doposud vložených dat smazat a celý proces vložení provést znovu jiný den. Mezi neočekávanými situacemi, které nastaly, byly například odpojení síťového disku, porucha připojení k Internetu v CVT.

5.4 Rozbor Quad-stromu a R-stromu

V kapitole 6 jsou navrženy různé způsoby indexování prostorových dat navrženého databázového schématu. Tyto navržené způsoby vycházejí ze dvou prostorových indexů implementovaných uvnitř Oracle, jmenovitě Quad-stromu a R-stromu. Tato kapitola má za cíl detailněji popsat implementaci struktur Quad-strom a R-strom indexů a princip jejich zpracování.

Tyto indexy jsou implementovány rozšiřitelným rámcem indexování Oracle, včleňuje a zvyšuje některé doposud nejlepší návrhy z existujících výzkumů prostorových indexů. Quad-strom počítá aproximace rozčleňujícího prostoru (tile) pro geometrie a užívá existujících B-strom indexů pro provedení prostorového hledání a dalších DML (Data Manipulation Language) operací. Tento přístup má za následek jednoduší vytvoření indexu, rychlejší aktualizaci a zdědění vestavěného souběžného řízení zápisu B-stromu. R-strom je implementován na logické úrovni jako strom a na fyzické použitím tabulek uvnitř databáze Vyhledávání zahrnuje rekurzivní SQL procházející strom od kořene k

příslušným uzlům listové úrovně. Tento přístup bude nejspíše více efektivní v oblasti dotazování z důvodu lepšího zachování prostorové sousednosti, ale jeho výkon bude slabší v operacích aktualizace a vytvoření indexu. R-strom implementuje své vlastní mechanismy souběžného zápisu. V následujících podkapitolách popíši tyto implementace v hlubších detailech a porovnám relativní chování zejména s ohledem na výkon dotazu.

5.4.1 Rámec indexu

Jak již bylo zmíněno, Quad-strom a R-strom indexy jsou implementovány použitím rozšiřitelného rámce. Tento rámec nám dovoluje vytvořit nové specifické indexy, asociované operátory dotazování a zabezpečuje integraci uživatelsky specifikovaných dotazů uvnitř Oracle. Oracle, podporující prostorová data podporuje typ indexu „prostorový_index“ za účelem indexování prostorových dat. Protože jsou tyto indexy implementovány jako část rozšiřitelného rámce indexování, tak mohou být jednoduše vytvořeny na sloupci SDO_GEOMETRY databázové tabulky zapsáním příkazu v rozšířené SQL syntaxi. Součástí takového vytvoření indexu je provedení odpovídající rutiny pro vytvoření prostorového indexu, poté je zbudovaný prostorový index uložen v databázi jako tabulka „prostorový_index“. Tabulka indexů shromažďuje informace o indexu, mezi které řadíme uzly R-stromu a úroveň rozčlenění plochy (tiles) v případě Quad-strom. Metadata pro kompletní index jsou uložena v jednom řádku v oddělené tabulce metadat. Tato metadata obsahují název tabulky „prostorový_index“ uchovávající indexy, počet rozměrů, ukazatel na kořen pro R-strom a stupeň rozčlenění plochy (tiling level) pro Quad-strom.

Operátory dotazování použité v klauzuli „WHERE“ SQL příkazu vybírají data, jež vyhovují specifikovanému kritériu dotazu s ohledem na definované okno dotazu. Operátory mají identické použití pro oba typy indexů. Popíši obecnou metodologii zpracování dotazu v Oracle podporující prostorová data.

5.4.2 Zpracování dotazu v Oracle podporující prostorová data

Při zpracovávání dotazů na složitých prostorových datech uplatňuje Oracle vícestupňový model dotazu. Model dotazu byl již popsán v kapitole 4.7. Při detailnějším pohledu na model dotazu se mezi primárním a sekundárním filtrem vyskytuje prostřední filtr (jedná se o část primárního filtru, v dalších kapitolách je budu uvažovat odděleně). Ve stručnosti lze říci, že v prostředním filtru modelu jsou kandidátní geometrie porovnávány s dotazem použitím aproximace vnitřku dotazu a kandidátních geometrií. Ve výsledku je kandidátní geometrie buď akceptována, nebo eliminována na základě kritéria dotazu. Zbytek geometrií, jejíž interakce (viz. Kapitola 4.8) není určena v prostředním filtru je předán na vstup sekundárnímu filtru modelu, který již vynese přesnou množinu výsledků.

Následuje stručný popis zpracování primárního a prostředního filtru pro R-strom index (podobně pracuje i Quad-strom index). Uvažujme geometrii dotazu Q a data geometrie G.

Předpokládejme, že dotaz hledá všechny geometrie, které protínají geometrii dotazu (tj. vztah nejsou disjunktní). Primární filtr určí, zda dotaz Q může být v interakci s geometrií G použitím její aproximace zevnějšku (MBR), který udává vzájemné protnutí. Prostřední filtr rozhodne, zda zevnějšek geometrie G je uvnitř vnitřku dotazu Q. Tímto způsobem je určeno, zda dotaz splnil kritérium dotazu (nejsou disjunktní) a zahrne geometrii přímo do výsledné množiny předanou sekundárnímu filtru. Porovnání dvou geometrií (geometrie dotazu a geometrie kandidátních dat) sekundárního filtru je náročné z důvodu načtení obou dat geometrií a drahého výpočtu geometrie požadovaného k určení vztahu mezi dvěmi geometriemi. Eliminace nebo akceptace prostředního filtru značně redukuje čas zpracování dotazu.

5.4.3 Filtrování založené na indexu

Primární filtr implementován Quad-strom indexem

Uživatel specifikuje stupeň rozčlenění (tiling level) a všechna data geometrie jsou aproximována minimální množinou pokrývajících rozčleněných ploch (tiles). Při dláždění (tessellation) jsou rozčleněné plochy (tiles) pro geometrii děleny do vnitřku a ohraničení založeném na tom, zda jsou nebo nejsou kompletně vnitřkem pro geometrii. Všechny pokrývající rozčleněné plochy (tiles) pro geometrii jsou poté vloženy do tabulky „prostorový_index“ společně s číslem řádku geometrie. Následkem tohoto přístupu může mít geometrie vícenásobné řádky v tabulce „prostorový_index“, kde každý řádek ukládá (rozdílné) kódy rozčleněné plochy (tile-code), vnitřek nebo ohraničení (stav) a číslo řádku geometrie. V čase dotazu je geometrie dotazu dlážděna (tessellated) do množiny rozčleněných ploch (tiles). Použitím každé rozčleněné plochy (tile) geometrie dotazu, čísla řádek všech geometrií, jejíž rozčleněné plochy (tiles) odpovídají rozčleněným plochám (tiles) dotazu jsou identifikovány kódem rozčleněné plochy (tile-code) spojením používající B-strom index na kódy rozčleněných ploch (tile-codes).

Výše uvedený přístup činí zacházení s aktualizacemi mnohem jednoduší. Po vložení nové geometrie do datové množiny je provedeno dláždění (tessellated) do uživatelem specifikovaného stupně rozčlenění ploch (tiles level) a poté jsou tyto plochy (tiles) vloženy do tabulky „prostorový_index“. B-strom index je automaticky aktualizován, aby zahrnul plochy (tiles) i pro novou geometrii. Stejně je tomu v případě mazání geometrie, kdy jsou korespondující řádky smazány z tabulky „prostorový_index“. Navzdory těmto značným výhodám má Quad-strom jednu stinnou stránku, tou je potřeba zvolit vhodný stupeň rozčlenění prostoru (tiling level) pro dláždění (tessellation) dat a geometrie dotazu. Tento stupeň je nutné zjistit několika experimenty na dané množině dat za účelem zjištění optimálního výkonu. Stupeň je ovlivněn i charakterem datové množiny, tudíž nelze použít obecně stejný stupeň pro odlišné množiny.

Primární filtr implementován R-strom indexem

R-strom udržuje logickou strukturu stromu a je implementován tabulkou, ve které každý uzel R-stromu koresponduje s řádkem v tabulce a ukazatel na potomka R-stromu koresponduje s *id* řádku synovského řádku ve stejné tabulce. Číslo řádku kořene (ukazatel) R-stromu je uloženo v metadatech pro index a umožňuje navigaci od kořene R-stromu až do uzlů listové úrovně. Uzly listové úrovně R-stromu uchovávají jeden MBR pro každá data geometrie spolu s číslem řádku geometrie. Operace dotaz a aktualizace obdrží kořen R-stromu a pohybují se stromem dolů směrem k uzlům listové úrovně. Zpracování dotazu a aktualizace v R-stromu dovoluje zpracování více rekurzivních SQL příkazů, nežli v případě Quad-stromu. Následkem toho některé DML operace zejména aktualizace budou více náročné.

Prostřední filtr v Oracle podporující prostorová data

Prostorový index pracuje v primárním filtru s aproximacemi zevnějšku dotazu a daty kandidátních geometrií za účelem rozhodnutí, zda jsou nebo nejsou v interakci. Pokud jsou v interakci, potom dotaz a geometrie jsou spolu s některými aproximacemi vnitřků předány prostřednímu filtru. Prostřední filtr porovná dotaz *q* a kandidátní geometrii *g* a vrací „true“ (v případě, že geometrie je zahrnuta do výsledku), „false“ (pokud geometrie není zahrnuta do výsledku) nebo „unknow“ (pokud nelze s jistotou určit vztah, v takovém případě je také zahrnuta do výsledku pro sekundární filtr, kde se určí přesný vztah). Prostřední filtr porovnává vnitřky dotazů a kandidátních geometrií (předané z primárního filtru) k určení vztahů. V případě Quad-stromu jsou porovnány vnitřek a ohraničení rozčleněných ploch (tiles) kandidátních geometrií s vnitřkem a ohraničením ploch (tiles) geometrie dotazu. U R-stromu je použito pouze vnitřku geometrie dotazu a vnitřky dat kandidátních geometrií nejsou počítány.

Předpoklady porovnání R-stromu a Quad-stromu

Kritickým problémem u Quad-stromu je určení stupně rozčlenění plochy (tiling level), který se významně podílí na výkonu Quad-stromu. Pro malé stupně rozčlenění plochy (tiling level) od 2 do 8 padne každá geometrie kompletně uvnitř jedné plochy (tile) a čas potřebný pro vytvoření indexu je malý. Ovšem s narůstajícím stupněm (tiling level) od 12 do 16 počet ploch (tiles) na geometrii razantně vzrůstá. Výhodou rostoucí hodnoty stupně (tiling level) je získání jemnějších a lepších aproximací ploch (tile) pro geometrie. V důsledku toho bude méně dat geometrií (aproximací tile) v interakci s daty geometrie a bude se zlepšovat výkon dotazu Quad-strom indexu. Nicméně vrůstající stupeň (tiling level) mimo jisté úrovně má za následek vytvoření příliš mnoho jemných aproximací ploch (tiles). Primární filtr se vzrůstajícím stupněm (tiling level) vrací menší množiny výsledků z důvodu lepší aproximace dat geometrií a geometrií dotazu. Z důvodu snížení počtu vrácených geometrií výrazně klesá výkon zpracování dotazu. Náklady na vyhledání ve velkém počtu ploch (tiles) převáží jakékoliv další výnosy z lepší aproximace.

Časová náročnost vytvoření Quad-strom indexu by měl být značně rychlejší oproti R-strom indexu. Protože čas pro seskupování na velké množině dat je v R-stromu relativně nákladný. Pro vložení dat bodů by Quad-strom měl dosahovat lepších výsledků než-li R-strom. Protože dláždění (tessellation) bodů je velmi rychlé, všechny Quad-stromy musejí provádět vložení plochy (tile) do tabulky „prostorový_index“ a vykonat přidružené aktualizace B-stromu. Pro toto vkládání dědí Quad-strom relativní rychlost B-strom aktualizací. Při vkládání složitých rozsáhlých objektů se situace změní a Quad-strom bude několikanásobně pomalejší než R-strom, kde značnou část času spotřebuje dláždění (tessellation). Časová náročnost vkládání pro R-strom bude růst téměř lineárně, nezávisle na velikosti a složitosti objektů.

Velikost úložiště (MB) bude u Quad-stromu několikanásobně vyšší než u R-stromu, protože stupeň rozčlenění (tiling level) optimalizuje výkon dotazu a tím produkuje mnoho ploch (tiles) pro každá data geometrie, což spotřebovává příliš mnoho fyzického prostoru.

6 Návrh experimentů

V této sekci představuji a popisuji jednotlivé návrhy experimentálních přístupů indexování časoprostorových dat. Výchozí model byl představen v kapitole 5.2, kód definující výchozí objektové datové typy jsou k nahlédnutí v příloze 1 a návrh diagramu tříd tohoto modelu je dostupný k nahlédnutí v příloze 2. Pro jednotlivé experimenty jsem strukturu databáze upravoval za účelem zjištění nejlepší struktury uložení trajektorií a po naplnění této databázové struktury příslušnými daty (diskutovanými v kapitole 5.3) vytvořil různé varianty indexů. Výsledky těchto experimentů jsou vůči sobě porovnány v kapitole 7.

Primárním záměrem této práce je vyšetření chování indexů a jejich výkonů na objektových typech Point a Line. V experimentech uvažuji indexovaní pouze těchto dvou typů. Objektový typ Region jsem definoval pouze jediný pokrývající celý prostor experimentů (město Oldenburg).

6.1 Index na Point

Úprava struktury databáze

V tomto balíčku experimentů budu indexovat pouze bod, přesněji atribut *position* tabulky Point_tab. Tento atribut je prostorového datového typu SDO_GEOMETRY. Záměrem je vyšetřit jakých výkonů bude dosahovat daná databázová struktura indexující pouze jednotlivé body, nikoliv úsečky.

```
CREATE OR REPLACE TYPE Point_typ AS OBJECT (
      id_point     NUMBER,
      position     SDO_GEOMETRY,
      instant      DATE,
      STATIC FUNCTION insert_point(x NUMBER, y NUMBER, instant_value VARCHAR2)
      RETURN NUMBER);
```

Struktura tabulky Line_tab má následující strukturu. Pouze se v tomto experimentu nebude indexovat atribut line prostorového datového typu SDO_GEOMETRY.

```
CREATE OR REPLACE TYPE Line_typ AS OBJECT (
      id_line      NUMBER,
      start_point  REF Point_typ,
      end_point    REF Point_typ,
      tsegment     REF TSegment_typ,
      line         SDO_GEOMETRY,
      STATIC PROCEDURE insert_line(mpoint_num NUMBER, region_num NUMBER, x NUMBER,
      y NUMBER, instant_value VARCHAR2));
```

Vytvoření indexů

V prvním testu tohoto experimentu vytvořím R-strom index na atributu *position* tabulky Point_tab:

```
CREATE INDEX point_idx ON Point_tab(position) INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```

Ve druhém testu vytvořím Quad-strom index na stejném atributu tabulky Point_tab:

```
CREATE INDEX Point_idx_quadstrom ON Point_tab(position) INDEXTYPE IS
MDSYS.SPATIAL_INDEX PARAMETERS('SDO_LEVEL=10');
```

Testy provedu v prvé iteraci bez indexováni časové složky časoprostorových dat. Ve druhé iteraci budu zkoumat zda vytvoření indexu na časové složce uložené v atributu *instant* tabulky Point_tab bude mít vliv na výkonnost zpracování dotazů a tím i celé databáze. Pro tento účel vytvořím B-strom index:

```
CREATE INDEX time_idx ON Point_tab(instant);
```

6.2 Index na Line

Úprava struktury databáze

V tomto balíčku experimentů budu indexovat pouze úsečku (ze kterých je sestavena celý segment, respektive trajektorie), přesněji atribut *line* tabulky Line_tab. Tento atribut je prostorového datového typu SDO_GEOMETRY. Ve výsledku bych měl zjistit, zda je výhodnější indexovat bod nebo úsečku z pohledu nejvhodnějšího způsobu indexace dat.

```
CREATE OR REPLACE TYPE Line_typ AS OBJECT (
      id_line      NUMBER,
      start_point  REF Point_typ,
      end_point    REF Point_typ,
      tsegment     REF TSegment_typ,
      line         SDO_GEOMETRY,
      STATIC PROCEDURE insert_line(mpoint_num NUMBER, region_num NUMBER, x NUMBER,
      y NUMBER, instant_value VARCHAR2));
```

Tabulka Point_tab zůstává nezměněna. Obsahuje atribut *position* prostorového datového typu SDO_GEOMETRY, který zůstane neindexován.

```
CREATE OR REPLACE TYPE Point_typ AS OBJECT (
      id_point     NUMBER,
      position     SDO_GEOMETRY,
      instant      DATE,
      STATIC FUNCTION insert_point(x NUMBER, y NUMBER, instant_value VARCHAR2)
      RETURN NUMBER);
```

Vytvoření indexů

Vytváření indexů bude identické s předchozím případem. Pouze se změní cílový atribut, na kterém se bude vytvářet index. V prvním testu vytvořím R-strom index na atributu *line* tabulky Line_tab:

```
CREATE INDEX Line_idx ON Line_tab(line) INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```

Ve druhém testu opět vytvořím index typu Quad-strom na atributu *line* tabulky Line_tab:

```
CREATE INDEX Line_idx_quadstrom ON Linet_tab(line) INDEXTYPE IS MDSYS.SPATIAL_INDEX
PARAMETERS('SDO_LEVEL=10');
```

Tyto testy také provedu v prvé iteraci bez indexu na časové složce časoprostorových dat. A ve druhé iteraci budu zkoumat, zda indexování časové složky uložené v atributu *instant* tabulky Point_tab bude mít vliv na výkonnost zpracování dotazů a celé databáze. Pro tento účel vytvořím B-strom index:

```
CREATE INDEX time_idx ON Point_tab(instant);
```

6.3 Index na Point a Line

Úprava struktury databáze

V tomto experimentu budu zároveň indexovat bod i úsečku, přesněji atributy *position* tabulky Point_tab a *line* tabulky Line_tab. Oba atributy jsou prostorového datového typu SDO_GEOMETRY. A budu zkoumat, zda indexováním obou atributů bude databáze dosahovat vyšších výkonů oproti alternativě indexování pouze jednoho z nich.

```
CREATE OR REPLACE TYPE Line_typ AS OBJECT (
      id_line      NUMBER,
      start_point  REF Point_typ,
      end_point    REF Point_typ,
      tsegment     REF TSegment_typ,
      line         SDO_GEOMETRY,
      STATIC PROCEDURE insert_line(mpoint_num NUMBER, region_num NUMBER, x NUMBER,
      y NUMBER, instant_value VARCHAR2));

CREATE OR REPLACE TYPE Point_typ AS OBJECT (
      id_point     NUMBER,
      position     SDO_GEOMETRY,
      instant      DATE,
      STATIC FUNCTION insert_point(x NUMBER, y NUMBER, instant_value VARCHAR2)
      RETURN NUMBER);
```

Vytvoření indexů

Vytváření indexů bude opět identické s předchozími experimenty. Pouze s tím rozdílem, že budou vytvořeny dva indexy na atributech. Pro první test se jedná o R-strom indexy na atributech *position* (tabulka Point_tab) a *line* (tabulka Line_tab):

```
CREATE INDEX point_idx ON Point_tab(position) INDEXTYPE IS MDSYS.SPATIAL_INDEX;
CREATE INDEX Line_idx ON Line_tab(line) INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```

Pro druhý test vytvořím indexy typu Quad-strom na stejné atributy jako v případě pro R-strom:

```
CREATE INDEX Point_idx_quadstrom ON Point_tab(position) INDEXTYPE IS
MDSYS.SPATIAL_INDEX PARAMETERS('SDO_LEVEL=10');
CREATE INDEX Line_idx_quadstrom ON Linet_tab(line) INDEXTYPE IS MDSYS.SPATIAL_INDEX
PARAMETERS('SDO_LEVEL=10');
```

Tyto testy budu také provádět s indexováním časové složky dat a bez indexování časové složky. Příkaz pro vytvoření B-strom indexu:

```
CREATE INDEX time_idx ON Point_tab(instant);
```

6.4 Sekvenční prohledávání (full scan)

Úprava struktury databáze

V tomto experimentu nebudu využívat podpory prostorového datového typu SDO_GEOMETRY a nebudu vytvářet žádné indexy na dimenze prostoru. Ze struktury tabulky Line_tab je odebrán atribut *line* typu SDO_GEOMETRY. Výsledná struktura tabulky Line_tab má následující podobu:

```
CREATE OR REPLACE TYPE Line_typ AS OBJECT (
      id_line      NUMBER,
      start_point  REF Point_typ,
      end_point    REF Point_typ,
      tsegment     REF TSegment_typ,
      -- line      SDO_GEOMETRY, tento atribut bude odstraněn
      STATIC PROCEDURE insert_line(mpoint_num NUMBER, region_num NUMBER, x NUMBER,
      y NUMBER, instant_value VARCHAR2));
```

V tabulce Point_tab byl odstraněn atribut *position* typu SDO_GEOMETRY a byly přidány dva atributy pos_x, pos_y typu number zastupující odebraný atribut. Tyto dva atributy, jak je zřejmé z jejich názvů, ukládají pozici x a pozici y.

```
CREATE OR REPLACE TYPE Point_typ AS OBJECT (
```

```
      id_point      NUMBER,
      pos_x         NUMBER,
      pos_y         NUMBER,
      instant       DATE,
      STATIC FUNCTION insert_point(x NUMBER, y NUMBER, instant_value VARCHAR2)
      RETURN NUMBER);
```

Vytvoření indexů

V přístupu sekvenčního prohledávání nebude použit žádný prostorový index. Za účelem porovnání s ostatními navrženými indexovými strukturami. Testy provedu v prvé iteraci bez indexu na časové složce časoprostorových dat. Ve druhé iteraci budu zkoumat, zda indexování časové složky uložené v atributu *instant* tabulky Point_tab bude mít vliv na výkonnost zpracování dotazů a celé databáze. Pro tento účel vytvořím B-strom index:

```
CREATE INDEX time_idx ON Point_tab(instant);
```

6.5 Z-uspořádání a B-strom přístup

Úprava struktury databáze

V tomto posledním navrženém experimentu také nebudu využívat podpory prostorového datového typu SDO_GEOMETRY. Databázová struktura je zcela shodná jako u předešlého experimentu a má následující podobu:

```
CREATE OR REPLACE TYPE Line_typ AS OBJECT (
      id_line       NUMBER,
      start_point   REF Point_typ,
      end_point     REF Point_typ,
      tsegment      REF TSegment_typ,
      -- line       SDO_GEOMETRY, tento atribut bude odstraněn
      STATIC PROCEDURE insert_line(mpoint_num NUMBER, region_num NUMBER, x NUMBER,
      y NUMBER, instant_value VARCHAR2));

CREATE OR REPLACE TYPE Point_typ AS OBJECT (
     id_point       NUMBER,
     pos_x          NUMBER,
     pos_y          NUMBER,
     instant        DATE,
     STATIC FUNCTION insert_point(x NUMBER, y NUMBER, instant_value VARCHAR2)
     RETURN NUMBER);
```

Vytvoření indexů

V tomto experimentu jsem implementoval funkčně založený index. Jedná se o funkci *ZValue*, která implementuje diskutované Z-uspořádání v kapitole 4.10.2. Ve stručnosti, funkce transformuje dvojdimenzionální prostor na jednodimenzionální s ohledem na co nejlepší poměr náročnosti výpočetního času a zachování sousednosti. Není třeba vytvářet další sloupec v tabulce pro uložení Z-hodnoty, protože ta je využita pouze při vytváření indexu. Ukázka kódu funkce:

```
CREATE OR REPLACE FUNCTION zvalue (x NUMBER, y NUMBER) RETURN NUMBER DETERMINISTIC
     IS
       x_grid    NUMBER;
       y_grid    NUMBER;
       num_bits  NUMBER;
       shift     NUMBER;
       mask      NUMBER;
       zvalue    NUMBER := 0;
       i         NUMBER;
       Np        NUMBER := 13; -- počet rozdělení použitých v 1D

     BEGIN
       x_grid := x * Np;
       y_grid := y * Np;

       IF x = 24000 THEN  -- max. hranice x- osy
         x_grid := x_grid - 24000;
       END IF;

       IF y = 31000 THEN  -- max. hranice y- osy
         y_grid := y_grid - 31000;
       END IF;

       num_bits := LOG(2, Np); -- počet bitů použitých k zakódovaní z-hodnoty
       shift := num_bits;

       FOR i IN 1..num_bits LOOP
         mask := POWER(2, num_bits-1);
         zvalue := zvalue + BITAND(x_grid, mask) * POWER(2, shift);
         shift := shift - 1;
         zvalue := zvalue + BITAND(y_grid, mask) * POWER(2, shift);
       END LOOP;

       RETURN ROUND(zvalue);

     END;
```

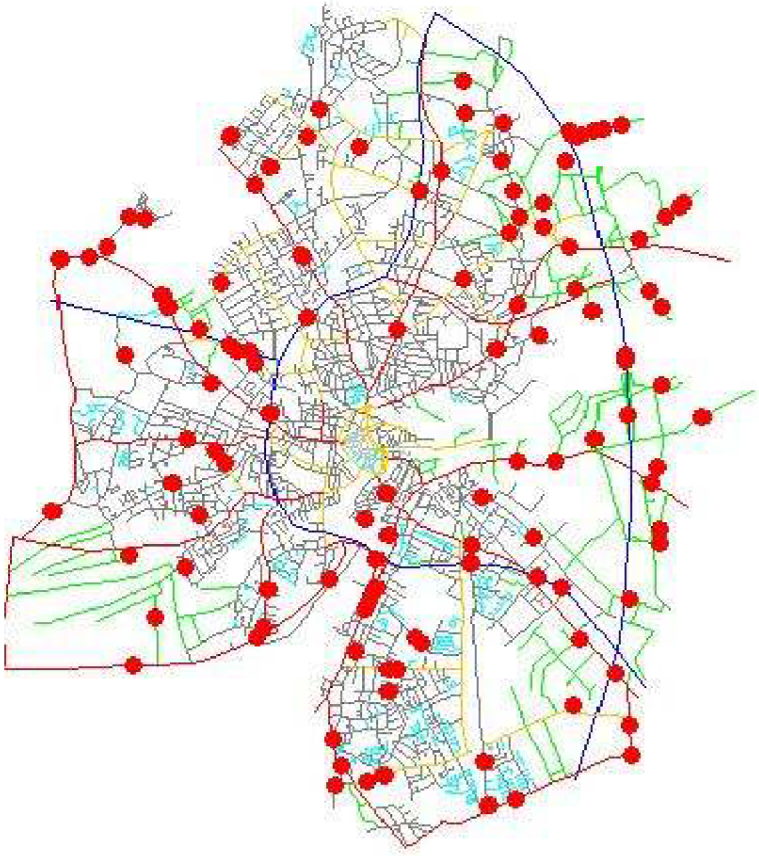
Samotný B-strom index je vytvořen na základě této funkce provedením následujícího příkazu:

```
CREATE index idx_zorder ON Point_tab(zvalue(pos_x, pos_y));
```

7 Zhodnocení experimentů

7.1 Úvodem

Pro všechny provedené experimenty bylo použito jediné množiny vygenerovaných časoprostorových dat (popsaných v kapitole 5.3) generátorem Network-based Generator of Moving Objects. Množina se sestává z téměř 450 000 časoprostorových bodů popisujících 4000 trajektorií v síti města Oldenburg. Obrázek 17 ilustruje několik desítek pohybujících se objektů v síti města, červeně jsou vyznačeny pohybující se objekty (například chodec, vozidlo). Pro měření doby zpracování příkazů a dotazů bylo použito vestavěné funkce Oracle nazývané časování (timing) v SQL*Plus. Tato funkce vrací přesný časový údaj v milisekundách (ms).



Obr. 17 Ilustrace několika pohybujících se objektů v síti města

Všechny experimenty a dílčí testy byly vykonány na školních serverech fakulty FIT na adresách pcuifs1.fit.vutbr.cz:1550 a berta.fit.vutbr.cz:1550 na kterých běží databázový server Oracle 10g.

7.2 Experimenty bez časové dimenze

Pro experimenty 6.1 až 6.3 bylo použito stejné struktury databáze a vložení dat do této databáze trvalo 690 minut (11hod 30min). V případě experimentů 6.4 a 6.5 nebylo použito datových typů SDO_GEOMETRY a z tohoto důvodu byla náročnost vkládání dat snížena na 394 minut. Jednalo se přesně o 442 184 časoprostorových bodů. Vložení bylo provedeno užitím SQL*Plus konzole systému Oracle. V tomto kroku se ještě nevytvářely žádné indexy, pouze byla databáze naplněna daty.

Tabulky 1 až 4 prezentují informace o spotřebovaném času operací vytvoření indexu (v případě experimentu index na Point a Line, hovoříme o vytvoření indexů) Každý experiment byl rozdělen na dva testy, v prvém jsem zkoumal vlastnosti R-strom indexu a v druhém Quad-strom indexu.

Experiment	Index	Čas vytvoření v sekundách
Index na Point	R-strom	48 s
	Quad-strom	18 s
	B-strom	1.5 s

Tab. 1 Časy vytvoření indexů experimentu index na Point

Experiment	Index	Čas vytvoření v sekundách
Index na Line	R-strom	53 s
	Quad-strom	45 s
	B-strom	1.7 s

Tab. 2 Časy vytvoření indexů experimentu index na Line

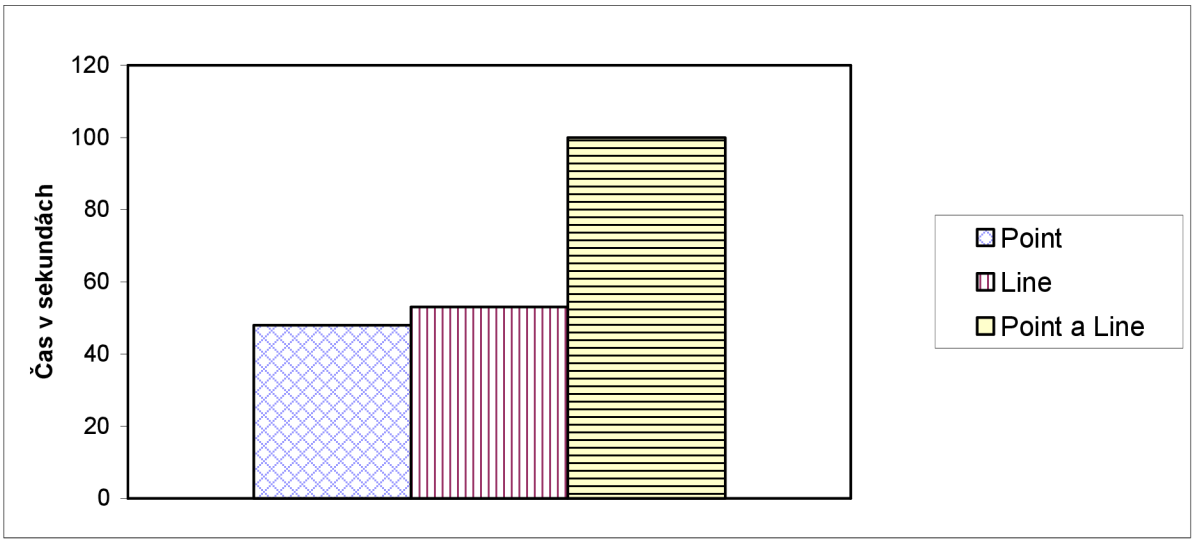
Experiment	Index	Čas vytvoření v sekundách	
Index na Point a Line	R-strom	Vytvoření indexu na Point:	49 s
		Vytvoření indexu na Line:	51 s
		Celkový:	100 s
	Quad-strom	Vytvoření indexu na Point:	18 s
		Vytvoření indexu Line:	52 s
		Celkový:	70 s
	B-strom	Vytvoření indexu:	1.6 s

Tab. 3 Časy vytvoření indexů experimentu index na Point a Line

Experiment	Čas vytvoření v sekundách
Z-uspořádání	216 s

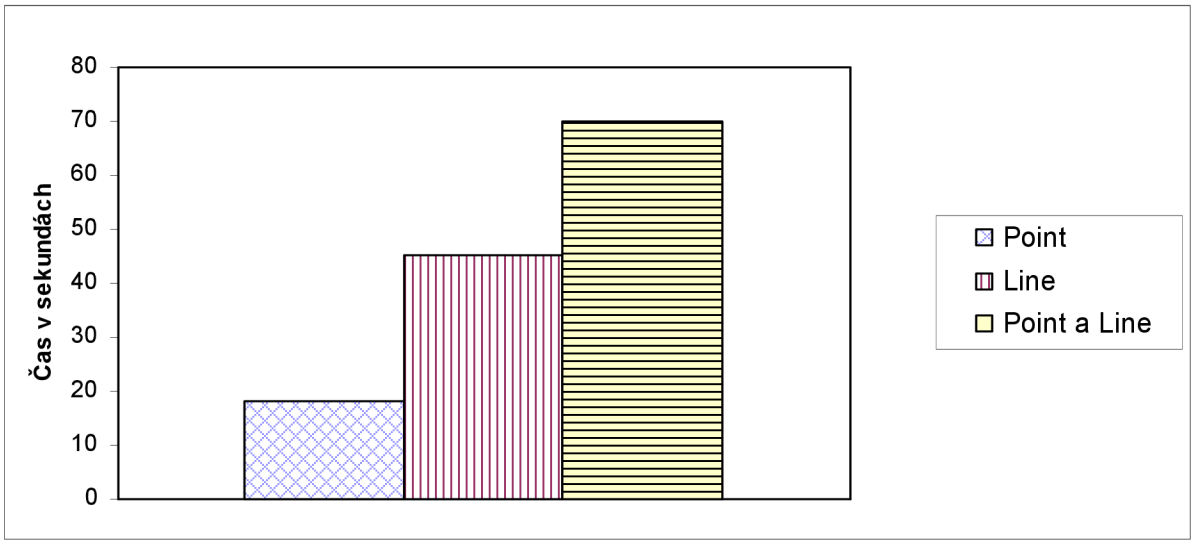
Tab. 4 Čas vytvoření funkčně založeného indexu experimentu Z-uspořádání

Následující grafy zohledňují zaznamenané výsledky z tabulek 1 až 3. Přesněji v grafu 1 vidíme čas potřebný k vytvoření R-strom indexů pro jednotlivé experimenty. Naproti tomu v grafu 2 jsou vyneseny časové záznamy pro vytvoření Quad-strom indexů.



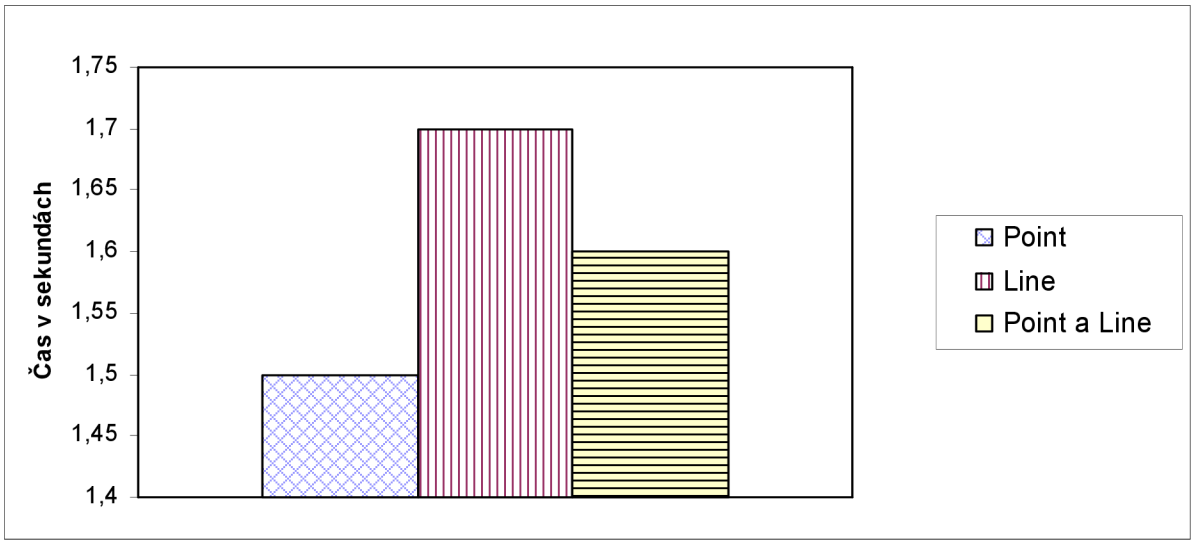
Graf 1 Časy vytvoření R-strom indexů pro jednotlivé experimenty

Z grafu 1 můžeme vyčíst, že doby vytvoření R-strom indexů pro všechny experimenty jsou srovnatelné. Rozdíl mezi nejrychlejším a nejpomalejším provedením činí pouze 5 sekund (viz. Tabulka 3, R-strom na Point a Line je dán součtem dílčích hodnot 49 a 51). S narůstající velikostí databáze by měl čas pro vytvoření R-stromu narůstat, protože v takovém případě bude strom rozsáhlejší a tím tedy výčet složitější. Blíže je implementace vytváření indexů popsána v kapitole 5.4.



Graf 2 Časy vytvoření Quad-strom indexů pro jednotlivé experimenty

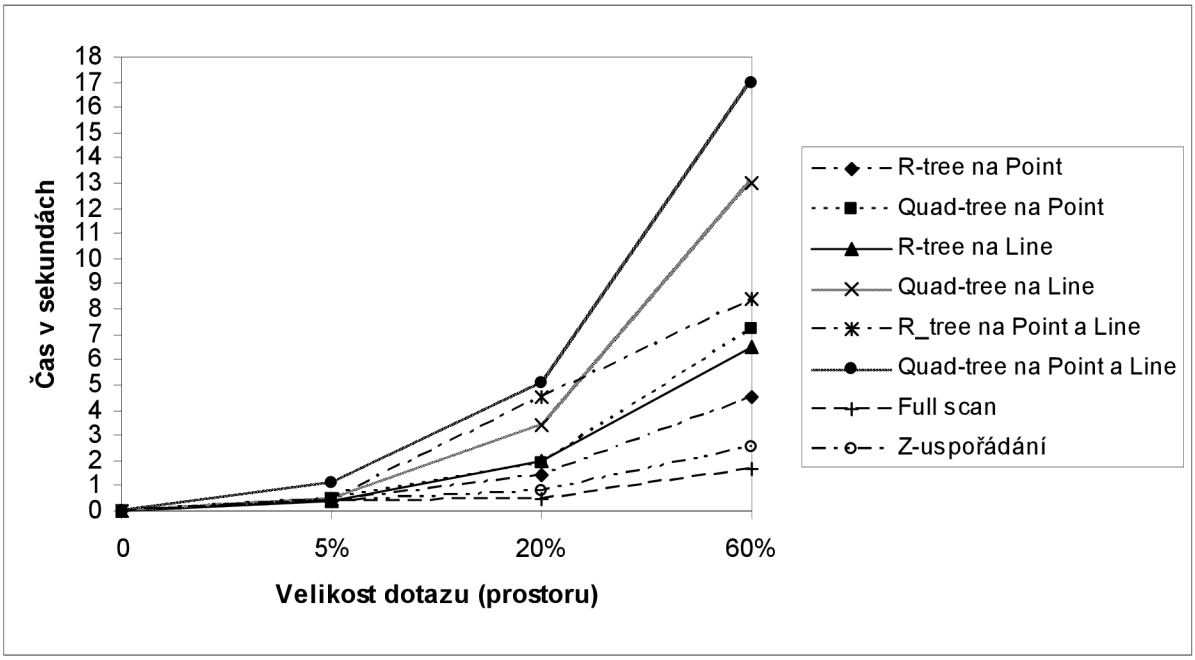
Z grafu 2 již vidíme, že vytvoření Quad-strom indexu na Line je časové náročnější. Což potvrdili téměř identické hodnoty při vytváření indexů v případech Line (45s), Point a Line (70s, což je součet dílčích hodnot 18s a 52s). Ve srovnání s grafem 1 vidíme, že vytváření indexů R-strom je výrazněji náročnější. Třetí graf vynáší hodnoty vytvoření časového B-strom indexu. Lze vysledovat, že rozdíly v hodnotách vytvoření tohoto indexu jsou zanedbatelné i při počtu téměř 450 000 časoprostorových bodů v databázi.



Graf 3 Časy vytvoření B-strom indexu na časovou složku experimentů

Další srovnání navržených experimentů spočívalo v porovnání rychlosti zpracování prostorových dotazů nad daty databáze. Implementoval jsem několik dotazů využívající prostorový operátor sekundárního filtru SDO_RELATE (podrobněji příloha 3). Tyto dotazy jsou navržené tak, aby pokrývaly různé rozsahy prostoru. Dotazy implementují okno dotazu a výsledkem je množina

trajektorií, které se v definovaném prostoru vyskytují bez ohledu na časovou dimenzi. Jinými slovy, zda mají porovnávané trajektorie nějaký prostorový vztah s oknem dotazu (prostorové vztahy viz. kapitola 4.8). Označení dotazů 5% koresponduje se selektivitou 5% prostoru na každé dimenzi, 20% koresponduje se selektivitou 20% prostoru na každé dimenzi a 60% zcela analogicky s 60% prostoru na každé dimenzi. Je potřeba uvést, že tato okna dotazu jsou soustředěna v obdélníkovém tvaru okolo centra (středu) sítě města Oldenburg. To znamená, že oblastí zájmu 5% prostorového dotazu je pouze pohyb objektů a tím jejich trajektorií v bezprostředním centru města. Dotaz 20% rozšiřuje 5% okno dotazu o přilehlé části k centru a poslední 60% dotaz (pokrývající 5% i 20% dotazy) již pokrývá mimo centra a přilehlého okolí i okrajové části města Oldenburg. Dotazy jsem provedl odděleně pro každý test (druh indexu) v experimentu a opakoval jej šestkrát. Z těchto šesti naměřených hodnot jsem spočetl aritmetický průměr a ten zanesl jako konečný výsledek do tabulky 5.



Graf 4 Zobrazuje časy přístupů indexování v závislosti na velikosti okna dotazu bez časové složky

Graf 4 zohledňuje informace uložené v tabulce 5. Z grafu je jednoznačně vidět, že sekvenční prohledávání (full scan) a z-uspořádání významně překonávají ostatní přístupy. Ovšem při nepoužití prostorového datového typu SDO_GEOMTRY (sekvenční prohledávání, Z-uspořádání) není možné vyžívat prostorových operátorů poskytovaných Oracle (např. SDO_RELATE) a tudíž zdánlivou výhodu rychlosti zpracování dotazu jednoznačně převyšuje tato nevýhoda. Nejhorších výsledků dosahuje přístup index Quad-strom na Line (kapitola 6.2). Naopak nejlépe z navrhnutých přístupů si vede R-strom na Point (kapitola 6.1). Z tabulky lze vyčíst, že obecně indexování na Line dosahuje značně slabších výkonů oproti alternativě indexování na Point.

Index na		Dotaz rozsahu 5%	Dotaz rozsahu 20%	Dotaz rozsahu 60%
Point	R-strom	0.4	1.4	4.5
	Quad-strom	0.5	1.9	7.2
Line	R-strom	0.4	2.0	6.5
	Quad-strom	0.5	3.4	14.0
Point a Line	R-strom	0.7	4.5	8.4
	Quad-strom	1.1	5.1	17.0
Sekvenční prohledávání (full scan)		0.4	0.5	1.7
Z-uspořádání		0.4	0.8	2.5

Tab. 5 Experimenty s různými rozsahy prostorových dotazů bez časové dimenze

7.3 Experimenty s časovou dimenzí

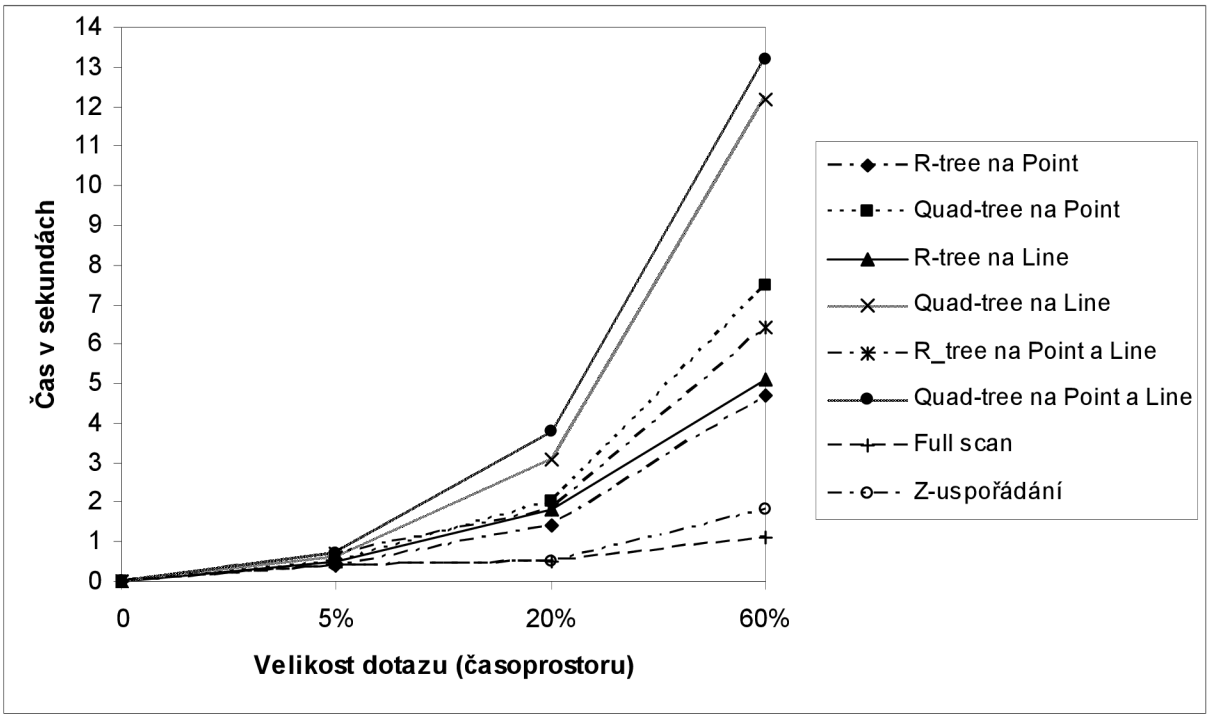
V této kapitole jsem provedl experimenty nad stejnými prostorovými dotazy, popsanými v předchozí kapitole, které jsem rozšířil o časovou dimenzi. Výsledkem dotazů je množina trajektorií, které byly v jakékoliv interakci s daným prostorem okna dotazu v definovaný časový interval. Pro testování jsem použil konstantní výsek časového intervalu, který odpovídá 25% časové dimenze. Tyto testy byly obsáhlejší z důvodu zkoumání, zda indexování časové dimenze přinese zlepšení výkonu zpracování prostorového dotazu. Každý experiment byl popsán dvěma tabulkami, kde jedna korespondovala s testem R-strom indexu a druhá zobrazovala výsledky testu s Quad-strom indexem. Po provedení všech testů jsem ovšem zjistil, že indexování časové dimenze nemělo žádný vliv na rychlost zpracování dotazu. Výsledky byly identické. Po prozkoumání plánu provádění dotazu jsem zjistil, že optimalizátor časový index vynechával. Pokoušel jsem se optimalizátor přinutit pomocí zápisu příkazu v SQL použít časového indexu. Kód příkazu:

```
/*+ index(point_tab point_idx) index(point_tab time_idx) */
```

Bohužel optimalizátor i přes tuto snahu časový index stále vynechával. Což můžeme vidět na obrázku 18, na kterém je sejmuta obrazovka programu SQL Developer zobrazující žádaný dotaz (i s příkazem udávající optimalizátoru použití časového indexu) a plán provádění. Z důvodu redundance informací jsem výše popsané tabulky nahradil pouze jednou tabulkou 6, která zobrazuje pouze užitečné informace. Kvůli nemožnosti přinutit optimalizátor Oracle použít žádaný časový index, mohu konstatovat, že se nevyplatí indexovat časovou dimenzi.

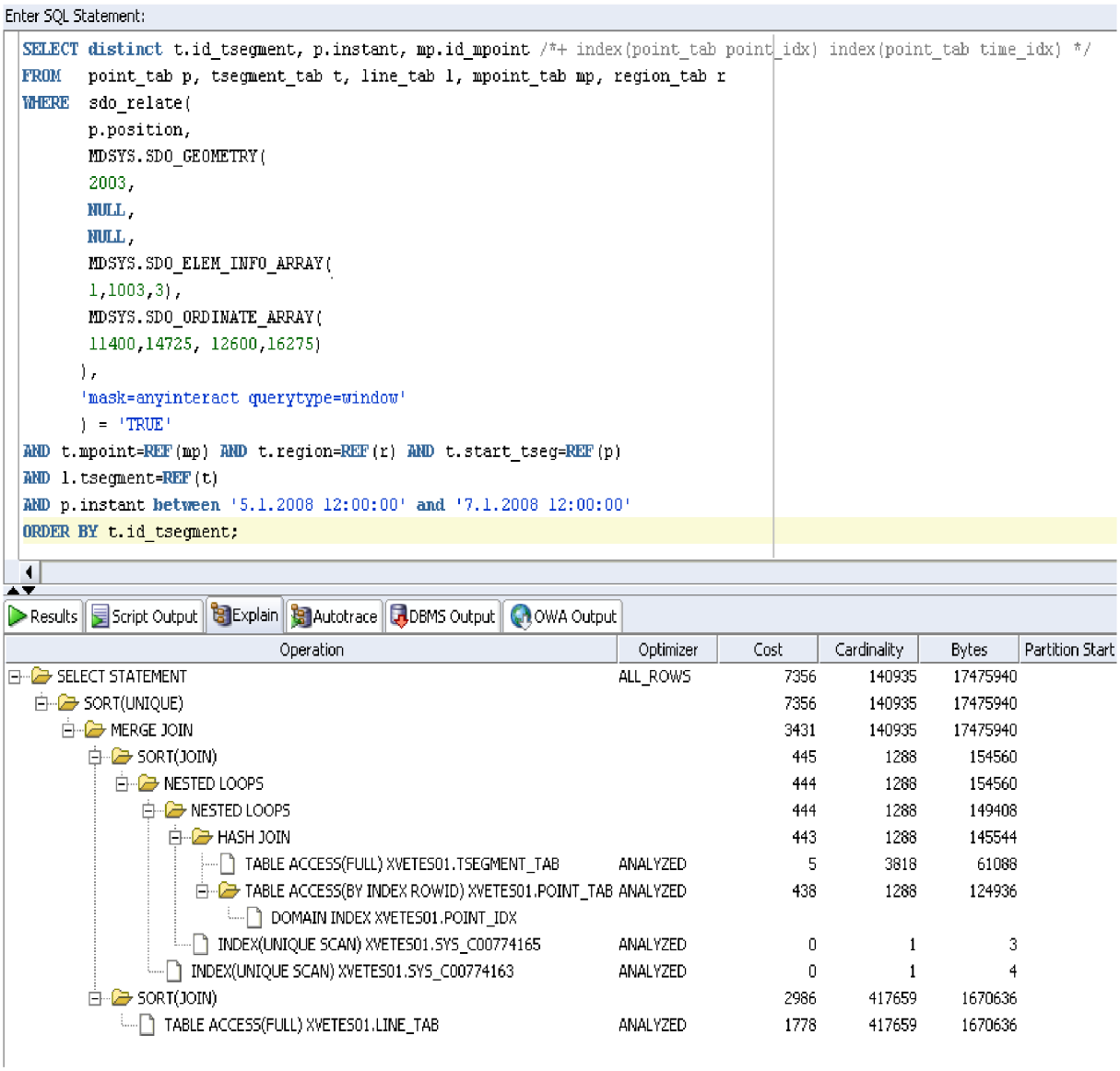
Index na		Dotaz rozsahu 5%	Dotaz rozsahu 20%	Dotaz rozsahu 60%
Point_typ	R-strom	0.4	1.4	4.7
	Quad-strom	0.5	2.0	7.5
Line_typ	R-strom	0.5	1.8	5.1
	Quad-strom	0.6	3.1	12.2
Point_typ a Line_typ	R-strom	0.7	1.8	6.4
	Quad-strom	0.7	3.8	13.2
Sekvenční prohledávání (full scan)		0.4	0.5	1.1
Z-uspořádání		0.4	0.5	1.8

Tab. 6 Experimenty s různými rozsahy prostorových dotazů s časovou dimenzí



Graf 5 Zobrazuje časy přístupů indexování v závislosti na velikosti okna dotazu a časové složky

Graf 5 vynáší do přehledné podoby data z tabulky 6. Tato tabulka, jak již bylo zmíněno, shromažďuje zaznamenané časové údaje o spotřebovaném čase časoprostorovými dotazy. Jedná se o identické dotazy, které jsou prezentovány v grafu 4 (potažmo tabulce 5). S tím rozdílem, že v tomto grafu a tabulce je do dotazu zahrnuta i časová dimenze. To znamená, že dotaz je omezen nejen prostorovými dimenzemi ale i časovou dimenzí. Výsledky z grafu nám opět ukazují, že sekvenční prohledávání a z-uspořádání významně překonávají ostatní navržené přístupy (ovšem jak bylo řečeno, tuto rychlost převyšuje nevýhoda nemožnosti použití prostorových operátorů). Opět si nejlépe v tomto srovnání vede přístup index R-strom na Point (kapitola 6.1). Naopak nejhoršího výsledku dosahuje přístup index Quad-strom na Point a Line (6.3).



Obr. 18 Obrazovka SQL Developer zobrazující časoprostorový dotaz a plán provádění

8 Závěr

Na základě požadavků zadání jsem z mnoha zdrojů prostudoval reprezentaci pohybujících se objektů a problematiku indexování časoprostorových dat těchto objektů. Součástí této práce jsou kapitoly zabývající se obecným vysvětlením prostorových dat a možnostmi struktur indexování v podobě vhodné snad i pro čtenáře, jenž nemá základní znalosti v dané oblasti. Problém, který zkoumáme, je právě definování praktického úložiště a modelu indexování plně integrovaného do RSŘBD, který je schopen uložit velmi rozsáhlou databázi záznamů (o velikosti i několika miliónů záznamů), provádět pravidelné aktualizace v databázi a efektivním způsobem odpovídat na časoprostorové dotazy. V našem případě v databázovém systému Oracle 10g. Čtvrtá kapitola pokrývá a představuje možnosti tohoto konkrétního databázového systému. Počínaje představením samotné prostorové podpory až po zamýšlené časoprostorové možnosti tohoto systému. Přínosem této části projektu pro mne bylo, že jsem nahlédl do dané problematiky a seznámil se s implementací a principy indexování prostorových dat. Nastudoval jsem podmětné informace pro následující praktickou část práce, ve které jsem již navrhl konkrétní experimenty pro ověření vlastností různých způsobů uložení časoprostorových dat v databázi a jejich indexace.

Před samotným návrhem experimentů bylo třeba vyřešit jeden zásadní úkol nezbytný pro úspěšné provedení experimentů. A to, kde získat vhodná časoprostorová data, která by co nejvíce odpovídala podmínkám reálného světa. Nakonec jsem zvolil cestu vygenerování své vlastní datové množiny dat založené na reálné síti města Oldenburg generátorem Network-based Generator of Moving Objects. Detailní popis tohoto generátoru a samotného procesu generování datové množiny je vysvětlen v kapitole 5.3. V průběhu řešení nastaly problémy s vkládáním datové množiny časoprostorových dat do databáze. Tato fáze se protáhla až na neočekávaných šest týdnů. Vzniklé problémy a jejich řešení byly tak rozsáhlé, že jsem se rozhodl tomuto tématu věnovat samostatnou kapitolu 5.3.1. V některých případech trvalo vložení dat až 690 minut. A pokud během výpočtu nastala jakákoliv neočekávatelná chyba (omezení SQL*Plus, kvóta dostupného prostoru, porucha Internetu v CVT, odpojení síťového disku, a další), tak jsem musel doposud vloženou část vkládaných dat smazat a proces vkládání spustit od počátku další den. Během těchto několika týdnů jsem byl v neustále komunikaci s administrátorem školního databázového serveru Oracle.

Návrh výchozí databázové struktury je popsán v kapitole 5.2, popisující uložení trajektorie v databázi a diagram návrhu tříd této struktury je k nahlédnutí v příloze 2. Experimenty popsané v kapitole 6 byly navrhnuty nad různými variantami této databázové struktury použitím zejména indexů v podobě R-stromu a Quad-stromu.

V sedmé kapitole tohoto diplomového projektu jsem zhodnotil a přehledně zobrazil formou tabulek a grafů získané informace z realizací navrhnutých experimentů z pohledu nejvhodnějšího způsobu uložení a indexace dat reprezentujících pohybující se objekty. Experimentální výsledky

potvrzují předpoklady, které jsem prezentoval v kapitole 5.4. Quad-strom index bych doporučoval pro případy častých aktualizací používající jednoduché polygony geometrií. Nicméně pro Quad-strom index musí být určena úroveň rozčlenění plochy (tiling level) k získání nejlepšího výkonu. R-strom index nepotřebuje specifikovat kromě počtu dimenzí žádné další parametry. R-strom dosahoval ve všech provedených experimentech lepších nebo ekvivalentních výkonů oproti Quad-strom indexu. Je to způsobeno tím, že databáze používá jednoduché polygony geometrií (bod, úsečka) a objem dat není tak rozsáhlý. Úroveň rozčlenění plochy (tiling level) jsem po několika pokusech určil na hodnotu 10. Tato hodnota prokazovala nejlepší výkon Quad-strom indexu pro danou množinu dat. Jak jsem předpokládal, výkon sekvenčního prohledávání (full scan) dat zůstává relativně konstantní pro všechny situace. Část důvodu obtížného překonání sekvenčního prohledávání tabulky, obzvláště při zvětšující se velikosti dotazu je, že sekvenční prohledávání (full scan) tabulky vyžaduje velmi malé vyhledávací operace na disku. Data jsou prohledána a filtrována v souvisejících blocích. Účelem této práce bylo především zjištění, zda je výhodnější indexovat Point či Line a zda indexování časové složky má vliv na výkon zpracování dotazu. Dle výsledků můžu říci, že je výhodnější z hlediska výkonu indexovat Point pomocí R-stromu indexu. Experimentování s B-strom indexem na časové dimenzi ukázalo, že nelze optimalizátor Oracle donutit využít časový index a proto tedy nemá vliv na výkon dotazu. Tudíž mohu konstatovat, že index na časové složce nemá význam vzhledem k výkonu dotazu.

Na tuto práci lze volně navázat. Jednou z možností je nalezení lepšího zpracování vložení rozsáhlých objektových prostorových časoprostorových dat do databáze. Po vyřešení tohoto problému již bude možné vkládat rozsáhlejší množiny dat o velikosti několika miliónů záznamů. Kde se již výrazněji projeví výkon R-strom indexu.

Literatura

[1] Mallet, D., J.: *Relational Database Support for Spatio-Temporal Data*, Department of computing Science University of Alberta Edmonton, Alberta, Canada, 2004.

[2] *Oracle Spatial User's Guide and Reference*, 10g, Release 2 (10.2), Oracle Corporation, 2006. URL: http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14255/toc.htm (únor 2008)

[3] Pfoser D., Jensen Ch.: *Indexing of Network Constrained Moving Objects*, Proceedings of the 11th ACM international symposium on Advances in geographic information systems, New Orleans, Louisiana, USA, 2003.

[4] Pfoser D., Brakatsoulas S., Tryfona N.: *Modeling, Storing and Mining Moving Objects Databases*, ideas, pp.68-77, International Database Engineering and Applications Symposium (IDEAS'04), 2004.

[5] Weng J., Wang W., Huang J.: *Design and Implementation of Spatial-Temporal Data Model in Vehicle Monitor System*, College of Software, Beihang University, Beijing China, 2005.

[6] Brinkhoff Thomas, *Network-based Generator of Moving Objects*, 2005. URL: http://www.fh-oow.de/institute/iapg/personen/brinkhoff/generator/ (únor 2008)

[7] *Oracle Database Application Developer's Guide - Object-Relational Features*, 10g, Release 2 (10.2), Oracle Corporation, 2005. URL: http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14260/toc.htm (únor 2008)

[8] *SQL*Plus® User's Guide and Reference*, 10g, Release 2 (10.2), Oracle Corporation, 2005. URL: http://download.oracle.com/docs/cd/B19306_01/server.102/b14357/toc.htm (únor 2008)

[9] Sharma J., Herring J., Ravada S.: *Oracle Spatial*, Oracle Corporation, 2001.

[10] *Oracle Database Utilities*, 10g, Release 2 (10.2), Oracle Corporation, 2005. URL: http://download.oracle.com/docs/cd/B19306_01/server.102/b14215/toc.htm (únor 2008)

[11] *Oracle® Database PL/SQL User's Guide and Reference*, 10g, Release 2 (10.2), Oracle Corporation, 2005. URL: http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14261/toc.htm (únor 2008)

[12] *Oracle® Database SQL Reference*, 10g, Release 2 (10.2), Oracle Corporation, 2005. URL: http://download.oracle.com/docs/cd/B19306_01/server.102/b14200/toc.htm (únor 2008)

[13] Vališ J.: *Databáze pohybujících se objektů*, diplomová práce, Brno, FIT VUT v Brně, 2008.

Seznam příloh

Příloha 1 Uživatelsky definované datové typy v jazyce PL/SQL

Příloha 2 Diagram tříd návrhu databáze

Příloha 3 Kód experimentálních dotazů

Příloha 4 CD-ROM nosič s následujícím obsahem:

- /generovani_dat:
  - generátor dat,
  - skript PHP
  - testovací datová množina
- /kody_experimentu:
  - kódy pro vytvoření databázové struktury experimentů
  - kódy experimentálních dotazů
- /zprava:
  - obsahuje text diplomové práce ve formátech .doc a .pdf

Příloha 1 Uživ. def. datové typy v PL/SQL

Definování datového typu MPoint_typ reprezentující pohybující se objekt a metody operující nad tímto objektem, která provádí vložení nového záznamu.

```
CREATE OR REPLACE TYPE MPoint_typ AS OBJECT (
     id_mpoint      NUMBER,
      STATIC PROCEDURE insert_mpoint);

CREATE OR REPLACE TYPE BODY MPoint_typ
AS
     STATIC PROCEDURE insert_mpoint
     IS
             seq_num NUMBER;
             sql_stmt VARCHAR2(200);
     BEGIN
             SELECT MPoint_seq.nextval INTO seq_num FROM dual;

             sql_stmt := 'INSERT INTO MPoint_tab VALUES (:1)';
             EXECUTE IMMEDIATE sql_stmt USING seq_num;
     END;
END;
```

Vytvoření databázové tabulky MPoint_tab na základě datového typu MPoint_typ.

```
CREATE TABLE MPoint_tab OF MPoint_typ (id_mpoint PRIMARY KEY)
       OBJECT IDENTIFIER IS PRIMARY KEY;
```

Definování datového typu Point_typ reprezentující časoprostorový bod a metody operující nad tímto objektem, která provádí vložení nového bodu do databáze.

```
CREATE OR REPLACE TYPE Point_typ AS OBJECT (
       id_point      NUMBER,
       position      SDO_GEOMETRY,
       instant       DATE,
       STATIC FUNCTION insert_point(x NUMBER, y NUMBER, instant_value
                                    VARCHAR2) RETURN NUMBER);

CREATE OR REPLACE TYPE BODY Point_typ
AS
       STATIC FUNCTION insert_point(x NUMBER, y NUMBER, instant_value
                            VARCHAR2) RETURN NUMBER
       IS
              seq_num  NUMBER;
              sql_stmt VARCHAR2(200);
       BEGIN
              SELECT Point_seq.nextval INTO seq_num FROM dual;

              sql_stmt := 'INSERT INTO Point_tab VALUES
       (:1, SDO_GEOMETRY(2001,NULL,SDO_POINT_TYPE(:2,:3,NULL),NULL,NULL), :4)';
              EXECUTE IMMEDIATE sql_stmt USING seq_num, x, y, instant_value;

              RETURN seq_num;
       END;
END;
```

Vytvoření databázové tabulky Point_tab na základě datového typu Point_typ.

```
CREATE TABLE Point_tab OF Point_typ (id_point PRIMARY KEY)
      OBJECT IDENTIFIER IS PRIMARY KEY;
```

Definování datového typu Region_typ reprezentující oblast sledování.

```
CREATE OR REPLACE TYPE Region_typ AS OBJECT (
      id_region    NUMBER);
```

Vytvoření databázové tabulky Region_tab na základě datového typu Region_typ.

```
CREATE TABLE Region_tab OF Region_typ (id_region PRIMARY KEY)
      OBJECT IDENTIFIER IS PRIMARY KEY;
```

Definování datového typu TSegment_typ reprezentující segment trajektorie a metody operující nad objektem, která provádí vložení nového segmentu do databáze.

```
CREATE OR REPLACE TYPE TSegment_typ AS OBJECT (
      id_tsegment  NUMBER,
      mpoint       REF MPoint_typ,
      region       REF Region_typ,
      start_tseg   REF Point_typ,
      STATIC PROCEDURE insert_tsegment(mpoint_num NUMBER, region_num NUMBER,
            x NUMBER, y NUMBER, instant_value VARCHAR2));

CREATE OR REPLACE TYPE BODY TSegment_typ
AS
      STATIC PROCEDURE insert_tsegment(mpoint_num NUMBER, region_num NUMBER,
            x NUMBER, y NUMBER, instant_value VARCHAR2)
      IS
            seq_num   NUMBER;
            point_num NUMBER;
            sql_stmt  VARCHAR2(200);
      BEGIN
            point_num := Point_typ.insert_point(x, y, instant_value);

            SELECT TSegment_seq.nextval INTO seq_num FROM dual;

            sql_stmt := 'INSERT INTO TSegment_tab
                        SELECT :1, REF(m), REF(r), REF(p)
                        FROM MPoint_tab m, Region_tab r, Point_tab p
                        WHERE m.id_mpoint=:2 AND r.id_region=:3 AND
                        p.id_point=:4';
            EXECUTE IMMEDIATE sql_stmt USING seq_num, mpoint_num, region_num,
            point_num;
      END;
END;
```

Vytvoření databázové tabulky TSegment_tab na základě datového typu TSegment_typ.

```
CREATE TABLE TSegment_tab OF TSegment_typ (
     PRIMARY KEY (id_tsegment),
     FOREIGN KEY (mpoint) REFERENCES MPoint_tab,
     FOREIGN KEY (region) REFERENCES Region_tab,
     FOREIGN KEY (start_tseg)    REFERENCES Point_tab)
     OBJECT IDENTIFIER IS PRIMARY KEY;
```

Definování datového typu Line_typ popisující úsečku segmentu a metody operující nad objektem, která provádí vložení nové úsečky do databáze.

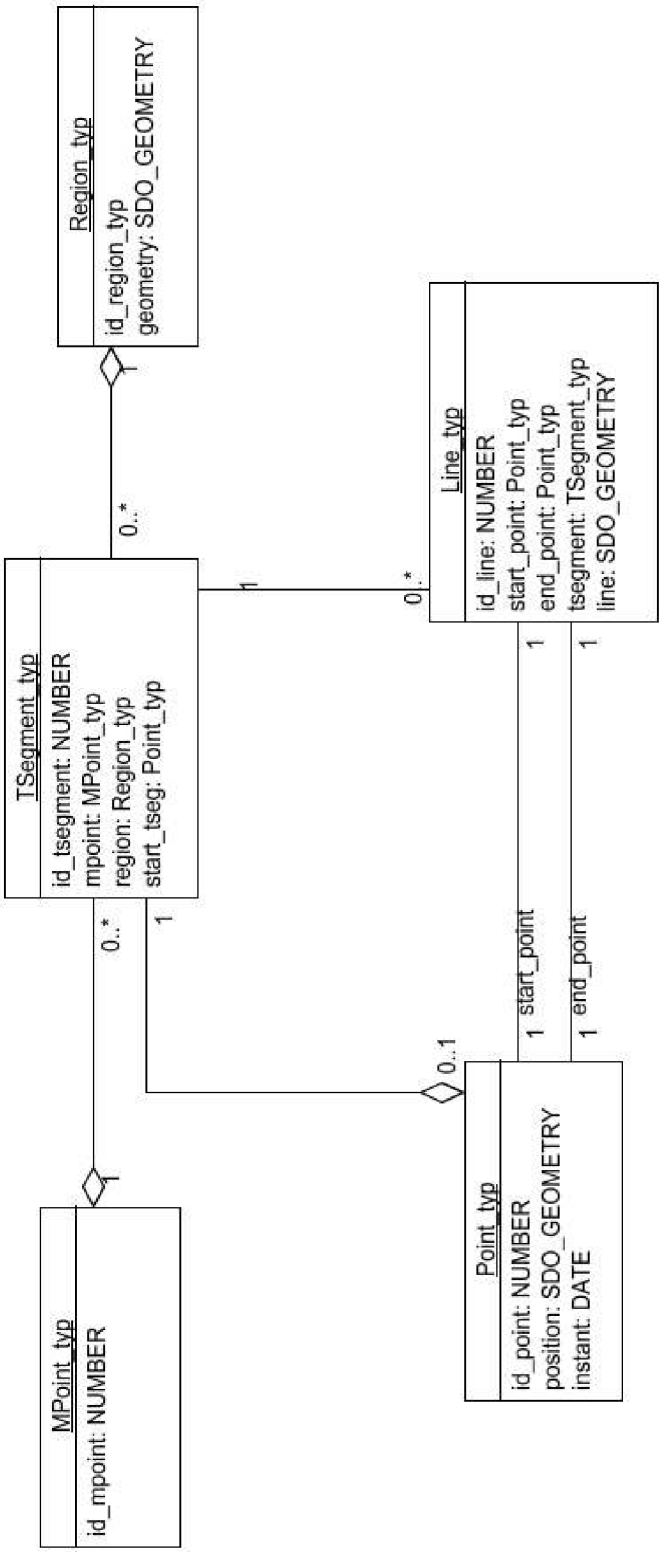
```
CREATE OR REPLACE TYPE Line_typ AS OBJECT (
       id_line      NUMBER,
       start_point  REF Point_typ,
       end_point    REF Point_typ,
       tsegment     REF TSegment_typ,
       line         SDO_GEOMETRY,
       STATIC PROCEDURE insert_line(mpoint_num NUMBER, region_num NUMBER, x NUMBER,
              y NUMBER, instant_value VARCHAR2));

CREATE OR REPLACE TYPE BODY Line_typ
AS
       STATIC PROCEDURE insert_line(mpoint_num NUMBER, region_num NUMBER, x NUMBER,
              y NUMBER, instant_value VARCHAR2)
       IS
              seq_num             NUMBER;
              count_of_lines      NUMBER;
              start_point_num     NUMBER;
              end_point_num       NUMBER;
              tsegment_num        NUMBER;
              point_x             NUMBER;
              point_y             NUMBER;
              max_line            NUMBER;
              sql_stmt            VARCHAR2(1000);
      BEGIN
      end_point_num := Point_typ.insert_point(x, y, instant_value);

      SELECT Line_seq.nextval INTO seq_num FROM dual;

      EXECUTE IMMEDIATE 'SELECT MAX(t.id_tsegment) FROM TSegment_tab t, MPoint_tab m
       WHERE t.mpoint=REF(m) AND m.id_mpoint=:1' INTO tsegment_num USING mpoint_num;

      EXECUTE IMMEDIATE 'SELECT COUNT(l.id_line) FROM Line_tab l, TSegment_tab t
              WHERE l.tsegment=REF(t) AND t.id_tsegment=:1' INTO count_of_lines
              USING tsegment_num;

      IF count_of_lines > 0 THEN

        EXECUTE IMMEDIATE 'SELECT MAX(l.id_line) FROM Line_tab l, TSegment_tab t
              WHERE l.tsegment=REF(t) AND t.id_tsegment=:1'INTO max_line
              USING tsegment_num;

        EXECUTE IMMEDIATE 'SELECT p.id_point FROM Point_tab p, Line_tab l
              WHERE l.id_line=:1 AND l.end_point=REF(p)' INTO start_point_num
              USING max_line;

        EXECUTE IMMEDIATE 'SELECT pos.X FROM Point_tab p,
            TABLE(SDO_UTIL.GETVERTICES(p.position)) pos
            WHERE p.id_point=:1' INTO point_x USING start_point_num;

        EXECUTE IMMEDIATE 'SELECT pos.Y FROM Point_tab p,
              TABLE(SDO_UTIL.GETVERTICES(p.position)) pos
              WHERE p.id_point=:1' INTO point_y USING start_point_num;

        sql_stmt := 'INSERT INTO Line_tab
              SELECT :1, REF(s), REF (e), REF(t),
               SDO_GEOMETRY(2002,NULL,NULL,
               SDO_ELEM_INFO_ARRAY(1,2,1),SDO_ORDINATE_ARRAY(:2,:3, :4,:5))
               FROM Point_tab s, Point_tab e, TSegment_tab t
               WHERE s.id_point=:6 AND e.id_point=:7 AND t.id_tsegment=:8';

        EXECUTE IMMEDIATE sql_stmt USING seq_num, point_x, point_y, x, y,
              start_point_num, end_point_num, tsegment_num;
```

```
     ELSE

      EXECUTE IMMEDIATE 'SELECT pos.X FROM Point_tab p, Point_tab pp,
            TSegment_tab t, TABLE(SDO_UTIL.GETVERTICES(p.position)) pos
            WHERE t.id_tsegment=:1 AND t.start_tseg=REF(pp) AND
            pp.id_point=p.id_point' INTO point_x USING tsegment_num;

      EXECUTE IMMEDIATE 'SELECT pos.Y FROM Point_tab p, Point_tab pp, TSegment_tab
            t, TABLE(SDO_UTIL.GETVERTICES(p.position)) pos
            WHERE t.id_tsegment=:1 AND t.start_tseg=REF(pp) AND
            pp.id_point=p.id_point' INTO point_y USING tsegment_num;

      EXECUTE IMMEDIATE 'SELECT p.id_point FROM TSegment_tab tr, Point_tab p
            WHERE tr.id_tsegment=:1 AND tr.start_tseg=REF(p)' INTO start_point_num
            USING tsegment_num;

      sql_stmt := 'INSERT INTO Line_tab
            SELECT :1, REF(s), REF (e), REF(t),
              SDO_GEOMETRY(2002,NULL,NULL,
              SDO_ELEM_INFO_ARRAY(1,2,1),SDO_ORDINATE_ARRAY(:2,:3, :4,:5))
            FROM Point_tab s, Point_tab e, TSegment_tab t
            WHERE s.id_point=:6 AND e.id_point=:7 AND t.id_tsegment=:8';

      EXECUTE IMMEDIATE sql_stmt USING seq_num, point_x, point_y, x, y,
            start_point_num, end_point_num, tsegment_num;
     END IF;
    END;
END;
```

Vytvoření databázové tabulky Line_tab na základě datového typu Line_typ.

```
CREATE TABLE Line_tab OF Line_typ (
      PRIMARY KEY (id_line),
      FOREIGN KEY (start_point) REFERENCES Point_tab,
      FOREIGN KEY (end_point)   REFERENCES Point_tab,
      FOREIGN KEY (tsegment)    REFERENCES TSegment_tab)
      OBJECT IDENTIFIER IS PRIMARY KEY;
```

Příloha 2 Diagram tříd návrhu databáze



Příloha 3 Kód experimentálních dotazů

Dotazy pro experimenty 6.1 až 6.3 používající SDO_GEOMETRY

Kód je ukázkou dotazu (okna dotazu) pro prostorové dimenze použité v kapitole 7.2. Tvar okna dotazu je dvoudimenzionální obdelník. V ukázce se jedná o 5% dotaz (20%, 60% liší pouze v odlišných hodnotách souřadnic, z důvodu duplicity je zde již neuvádím). Do výsledné sady jsou zahrnuty všechny trajektorie, které jsou v jakékoliv interakci s oknem dotazu.

```
SELECT distinct t.id_tsegment, p.instant, mp.id_mpoint
FROM   point_tab p, tsegment_tab t, line_tab l, mpoint_tab mp, region_tab r
WHERE  sdo_relate(
       p.position,
       MDSYS.SDO_GEOMETRY(
       2003,
       NULL,
       NULL,
       MDSYS.SDO_ELEM_INFO_ARRAY(
       1,1003,3),
       MDSYS.SDO_ORDINATE_ARRAY(
       11400,14725, 12600,16275)              -- zde urcena velikost dotazu
      ),
      'mask=anyinteract querytype=window'
      ) = 'TRUE'
AND t.mpoint=REF(mp) AND t.region=REF(r) AND t.start_tseg=REF(p)
AND l.tsegment=REF(t)
ORDER BY t.id_tsegment;
```

Následující kód prezentuje tentýž 5% dotaz rozšířený o časovou dimenzi použitý v kapitole 7.3. Do výsledné sady jsou zahrnuty pouze trajektorie, které jsou v jakékoliv interakci s oknem dotazu v definovaném časovém intervalu. Pro účely testování jsem zvolil konstantní výsek časového intervalu, který odpovídá 25% časové dimenze.

```
SELECT distinct t.id_tsegment, p.instant, mp.id_mpoint
FROM   point_tab p, tsegment_tab t, line_tab l, mpoint_tab mp, region_tab r
WHERE  sdo_relate(
       p.position,
       MDSYS.SDO_GEOMETRY(
       2003,
       NULL,
       NULL,
       MDSYS.SDO_ELEM_INFO_ARRAY(
       1,1003,3),
       MDSYS.SDO_ORDINATE_ARRAY(
       11400,14725, 12600,16275)
      ),
      'mask=anyinteract querytype=window'
      ) = 'TRUE'
AND t.mpoint=REF(mp) AND t.region=REF(r) AND t.start_tseg=REF(p)
AND l.tsegment=REF(t)
AND p.instant between '5.1.2008 12:00:00' and '7.1.2008 12:00:00'
ORDER BY t.id_tsegment;
```

Dotazy pro experimenty 6.4 a 6.5 nepoužívající SDO_GEOMETRY

Kód je ukázkou dotazu (okna dotazu) pro prostorové dimenze použité v kapitole 7.2. Tvar okna dotazu je dvoudimenzionální obdelník. V ukázce se jedná o 5% dotaz (20%, 60% liší pouze v odlišných hodnotách souřadnic, z důvodu duplicity je zde již neuvádím). Do výsledné sady jsou zahrnuty všechny trajektorie, které jsou v jakékoliv interakci s oknem dotazu.

```
SELECT distinct t.id_tsegment, p.instant, mp.id_mpoint
FROM   point_tab p, tsegment_tab t, line_tab l, mpoint_tab mp, region_tab r
WHERE  p.pos_x > 11400 AND p.pos_x < 12600
AND p.pos_y > 14725 AND p.pos_y < 16275
AND t.mpoint=REF(mp) AND t.region=REF(r) AND t.start_tseg=REF(p)
AND l.tsegment=REF(t)
ORDER BY t.id_tsegment;
```

Následující kód prezentuje tentýž 5% dotaz rozšířený o časovou dimenzi použitý v kapitole 7.3. Do výsledné sady jsou zahrnuty pouze trajektorie, které jsou v jakékoliv interakci s oknem dotazu v definovaném časovém intervalu. Pro účely testování jsem zvolil konstantní výsek časového intervalu, který odpovídá 25% časové dimenze.

```
SELECT distinct t.id_tsegment, p.instant, mp.id_mpoint
FROM   point_tab p, tsegment_tab t, line_tab l, mpoint_tab mp, region_tab r
WHERE  p.pos_x > 11400 AND p.pos_x < 12600
AND p.pos_y > 14725 AND p.pos_y < 16275
AND t.mpoint=REF(mp) AND t.region=REF(r) AND t.start_tseg=REF(p)
AND l.tsegment=REF(t)
AND p.instant between '5.1.2008 12:00:00' and '7.1.2008 12:00:00'
ORDER BY t.id_tsegment;
```