VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

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

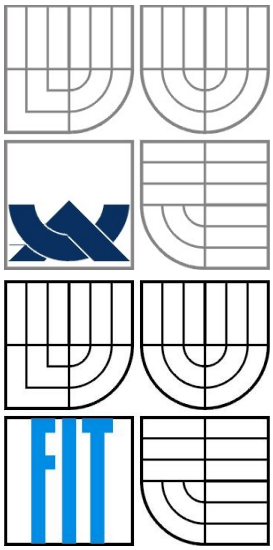
DATABÁZE POHYBUJÍCÍCH SE OBJEKTŮ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAROSLAV VALIŠ

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

DATABÁZE POHYBUJÍCÍCH SE OBJEKTŮ

MOVING OBJECTS DATABASE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. JAROSLAV VALIŠ
AUTHOR

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

BRNO 2008

Abstrakt

Práce pojednává o problematice reprezentace pohybujících se objektů a typických operací nad těmito objekty. Seznamuje s podporou pro časoprostorová data v prostředí databázového serveru Oracle10g a představuje dva návrhy struktury databáze pohybujících se objektů. Na základě těchto návrhů byla, s použitím uživatelsky definovaných datových typů, databáze implementována. Ukázková aplikace poskytuje grafický výstup prostorových dat, uložených v databázi a umožňuje volání implementovaných časoprostorových operací. Na závěr je provedeno zhodnocení dosažených výsledků a jsou uvedeny možnosti dalšího rozvoje projektu.

Klíčová slova

Pohybující se objekty, časoprostorové datové typy, časoprostorové databáze, objektově-relační model dat, Oracle Database 10g, Oracle MapViewer, PL/SQL, J2EE, JSP, EJB, JDBC, OC4J.

Abstract

This work treats the representation of moving objects and operations over these objects. Introduces the support for spatio-temporal data in Oracle Database 10g and presents two designs of moving objects database structure. Upon these designs a database was implemented using the user-defined data types. Sample application provides a graphical output of stored spatial data and allows us to call an implemented spatio-temporal operations. Finally, an evaluation of achieved results is done and possible extensions of project are discussed.

Keywords

Moving objects, spatio-temporal data types, spatio-temporal databases, object-relational data model, Oracle Database 10g, Oracle MapViewer, PL/SQL, J2EE, JSP, EJB, JDBC, OC4J.

Citace

Jaroslav Vališ: Databáze pohybujících se objektů, diplomová práce, Brno, FIT VUT v Brně, 2008

Databáze pohybujících se objektů

Prohlášení

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

.....
Jaroslav Vališ
4.5.2008

Poděkování

Chtěl bych velmi poděkovat vedoucímu práce doc. Ing. Jaroslavu Zendulkovi, Csc. za jeho cennou pomoc a čas, který mi věnoval. Dále bych chtěl poděkovat Ing. Jaroslavu Rábovi za pomoc spojenou s technickým zázemím mé práce.

Obsah

Kapitola 1

Úvod

Každý objekt v reálném světě má svůj tvar a pozici v jistém časovém okamžiku, kromě toho může obsahovat i další vlastnosti, jako např. barvu, texturu, teplotu atd. Informace o současné, minulé a předpokládané budoucí poloze a tvaru objektu nazýváme časoprostorová data. S těmito daty pracuje množství aplikací a vzniká tak potřeba zachytit tyto vlastnosti objektů do databáze, což je předmětem této práce.

Rozlišujeme dva typy časoprostorových objektů, diskrétně pohybující se objekty a spojitě pohybující se objekty [1]. Pro diskrétně pohybující se objekt je poměrně snadné ukládat do databáze informace o změnách jeho tvaru a polohy a to pomocí vkládání nových záznamů do databáze. Stejně tak je snadná tvorba dotazů na disktrétní změny objektu díky možnosti využití relačního modelu dat, kdy jsou použity oddělené sloupce pro časová a prostorová data. Časový interval potom udává platnost prostorových dat.

Objekty, spojitě měnící tvar nebo polohu, jsou na rozdíl od diskrétních objektů komplexnější a také obtížnější pro uložení do databáze. Není možné využít oddělených časových a prostorových dat, neboť není k dispozici funkce s konstatním krokem a také není možné provádět aktualizaci databáze s každou prostorovou změnou objektu.

Tato práce se bude věnovat především problematice spojitě pohybujících se objektů a jejich reprezentaci s využitím objektových rysů databázového serveru Oracle 10g. Přestože jsou spojitě pohybující se objekty bližší realitě a existuje nad nimi větší rozsah operací, není prozatím implementována jejich podpora na úrovni databázového serveru.

V našem případě se bude jednat o ukládání, dotazování a dolování v datech pocházejících ze senzorové sítě (kamerového systému). Naším úkolem bude zvolit vhodnou reprezentaci pohybujících se objektů a implemetovat nad nimi potřebné operace a to na úrovni uložených procedur databázového serveru nebo jako metody databázového objektu.

Práce je strukturována následovně: Kapitola 2 představuje problematiku reprezentace pohybujících se objektů, uvádí rámec abstraktních datových typů a operace nad těmito typy, které budou dále využity. Kapitola 3 pojednává o podpoře časoprostorových dat v prostředí databázového serveru Oracle10g. Kapitola 4 představuje dva návrhy struktury databáze pohybujících se objektů. Kapitola 5 detailně popisuje implementovanou strukturu databáze, námi definované datové typy a vytvořené časoprostorové operace. V kapitole 6 je představena ukázková aplikace MOD. Kapitola 7 zhodnocuje dosažené výsledky a diskutuje možnosti dalšího rozvoje projektu. Kapitola 8 obsahuje závěr.

Kapitola 2

Reprezentace pohybujících se objektů

Tato kapitola seznamuje s problematikou reprezentace pohybujících se objektů, uvádí rámec abstraktních datových typů pro časoprostorová data a typické operace nad těmito typy. Rámec vychází ze základních, prostorových a časových typů a pomocí typových konstruktorů, které jsou aplikovány na tyto výchozí typy, vznikají nové temporální typy.

2.1 Časoprostorové datové typy

V této části definujeme systém datových typů a operací, neboli algebru, vhodnou pro reprezentaci a dotazování se nad časově proměnlivými geometrickými objekty. Definice algebry sestává ze dvou částí. V první části navrhneme systém typů uvedením několika základních typů a jejich konstruktorů. Pro každý typ je dána jeho sémantika definicí *nosné množiny*. Nosnou množinou je např. množina celých čísel pro typ int, množina znaků abecedy pro typ string atd. V druhé části navrhneme soubor operací nad typy tohoto typového systému. Pro každou operaci je definována signatura popisující syntaxi operace, tj. správný argument a výsledný typ. Sémantika operace je dána definicí funkce na nosné množině typů, do níž náleží typ jejího argumentu [1]. Následující systém typů a operace nad typy tohoto systému typů jsou převzaty z [1].

2.2 Systém typů

Systém typů je definován jako signatura, která sestává ze *tříd* a operátorů, kde třídy kontrolují použitelnost operátorů [2]. Třídy popisují určitou podmnožinu typů a v roli operátorů máme *konstruktory typů*. Signatura dále generuje množinu termů, jež přesně popisují typy dostupné v našem typovém systému. Signatury jsou dobře známé z definice abstraktních datových typů, např. pro popis zásobníku máme třídy STACK, INT, BOOL a operátory *push, pop, empty.* Příkladem signatury může být STACK x INT → STACK a poté termem této signatury je např. *push*(*empty*, 8).

V tabulce 2.1 jsou zobrazeny signatury definující náš systém typů. Třídy jsou označeny velkými písmeny a konstruktory typů kurzívou. Např. do třídy BASE náleží všechny základní typy, do třídy SPATIAL prostorové typy atd.

Termy a tedy i typy generované signaturou jsou např. *int*, *region*, *moving*(*point*), *range*(*int*) atd. Konstruktor typu *range* je aplikovatelný na všechny typy třídy BASE a TIME, tedy všechny typy, které je možné vytvořit konstruktorem range, jsou *range(int), range(real), range*(*string*), *range(bool), range*(*instant*). Konstruktor typu bez argumentu, např. *point*, je již typem a je nazýván *konstantní typ.* Pro získání uzavřeného systému operací nad časoprostorovými typy je nutné do systému typů přidat následující základní, prostorové a časové typy.

Konstruktor typu	Signatura	
int, real, string, bool		→ BASE
point, points, line, region		→ SPATIAL
instant		→ TIME
moving, intime	BASE ∪ SPATIAL	→ TEMPORAL
range, periods	BASE ∪ TIME	→ RANGE

Tabulka 2.1 Systému typů [1]

2.2.1 Základní typy

Základními typy jsou *int*, *real*, *string* a *bool,* rozšířené o nedefinovanou hodnotu. Základní typy náleží do jednodimenzionálního prostoru, který jako jediný obsahuje úplné uspořádání.

2.2.2 Prostorové typy

Základními entitami prostorových databází jsou bod, úsečka a region (anglicky *point*, *line*, *region*) [3]. *Bod* spolu s dalšími body může vytvářet konečné množiny bodů. *Křivka* je tvořena konečnou množinou bodů v rovině a *region* je tvořen konečnou množinou disjunktních ploch, které se spolu dotýkají na hranicích, a které mohou obsahovat díry. Prostorové typy náleží do dvojdimenzionálního prostoru.

2.2.3 Časový typ

Typ *okamžik* (anglicky *instant*) představuje bod v čase, jeho nosnou množinou je množina reálných čísel, sjednocená s nedefinovanou hodnotou, náleží tedy do jednodimenzionálního prostoru. Čas je považován za lineární a spojitý.

2.2.4 Temporální typy

Ze základních a prostorových typů, pro jejichž označení použijeme symbol α, jsou odvozeny odpovídající temporální typy. Pro tento účel slouží typový konstruktor *moving*, který pro daný datový typ α mapuje čas do typu α. Nosná množina typu moving(α) je tvořena funkcemi, které popisují časový vývoj hodnot z nosné množiny α. Funkce sestávají z konečného množství spojitých komponent a vytvářejí nové temporální typy *mint*, *mreal*, *mstring*, *mbool*, *mpoint*, *mline* a *mregion*. Temporální typy, vzniklé pomocí konstruktoru moving, reprezentují funkce času nebo konečné množiny párů (okamžik, hodnota).

Typový konstruktor *intime* převádí daný typ α do typu, který sdružuje časový okamžik s hodnotou α, tj. vznikne jeden pár (okamžik, hodnota). Nosnou množinou typu intime(α) je kartézský součin nosné množiny typu α a nosné množiny typu instant.

Temporální typy mohou náležet jak do jedno, tak i dvojdimenzionálního prostoru v závislosti na typu, ze kterého jsou odvozeny.

2.2.5 Rozsahové typy

Pomocí typového konstruktoru *range* získáme pro temporální typy jejich projekci do odpovídající domény a rozsahu. Máme-li základní typ α, jehož doména je jednodimenzionální, poté pro temporální typ odvozený z α, tj. pro moving(α), je projekce vyjádřena jako množina intervalů nad jednodimenzionální doménou. Tedy pomocí konstruktoru range můžeme vytvářet typy reprezentující množiny intervalů nad množinami celých čísel, reálných čísel, atd.

Pro rozsahy nad časovou doménou zavádíme typ *periods*, kde periods = range(instant). Typ periods náleží do jednodimenzionálního prostoru.

2.3 Operace nad nontemporálními typy

V této části definujeme operace nad nontemporálními typy. Později z těchto operací, pomocí procesu *přizpůsobení*, získáme operace nad temporálními typy.

2.3.1 Notace

Pro zápis signatury zavedeme notaci, kde symbol π značí typ, jehož proměnné jsou proměnné typu α, které nabývají pouze jedné hodnoty a symbol σ značí typ, jehož proměnné jsou proměnné typu α, které nabývají množiny hodnot. Proměnná typu π tak může být např. typu bool, real, instant, point apod. Proměnná typu σ může být např. typu range(int), range(string), periods, line apod. Proměnné typu π a σ tedy mohou být jak jedno, tak i dvojdimenzionální proměnné, viz tabulka 2.2.

Pro definici sémantiky zavedeme notaci, kde symboly *k*, *l*, *m* označují jednotlivé hodnoty typu π a symboly *K*, *L*, *M* označují množiny hodnot typu σ.

Pro popis topologie zavedeme označení ∂K pro vyjádření hranice množiny bodů *K* a označení K° pro vyjádření vnitřku množiny bodů *K*. Pro topologii, na níž byla provedena uzávěrová operace, použijeme označení $\rho(K)$.

Některé operace jsou omezeny do určitého prostoru, proto u signatury, popisující syntaxi takovéto operace, je v hranatých závorkách uvedeno omezení. Např. signatura α → β [1D] je platná pro všechny jednodimenzionální prostory.

	Jednodimenzionální prostor					Dvojdimenzionální p.
π	int	bool	string	real	instant	point
σ	range(int)	range(bool)	range(string)	range(real)	periods	points, line, region

Tabulka 2.2 Přehled nontemporálních typů [1]

2.3.2 Predikáty

Predikáty mohou být unární nebo binární. Unární predikát *isempty* vrací hodnotu true, jestliže je bod nedefinovaný, nebo je-li množina bodů prázdná. Binární predikáty slouží pro vyhodnocení možných interakcí mezi dvěma body, bodem a množinou bodů, dvěma množinami bodů a dvěma topologiemi. Přehled predikátových operací se nachází v tabulce 2.3.

Operace	Signatura		Sémantika
$=, \neq$	$\pi \times \pi$	$\rightarrow$ bool	$k = l, k \neq l$
	$\sigma \times \sigma$	$\rightarrow$ bool	$K = L, K \neq L$
intersects	$\sigma \times \sigma$	$\rightarrow$ bool	$K \cap L \neq \emptyset$
inside	$\sigma \times \sigma$	$\rightarrow$ bool	$K \subseteq L$
	$\pi \times \sigma$	$\rightarrow$ bool	$k \in L$
$<, \leq, \geq, >$	$\pi \times \pi$	$\rightarrow$ bool [1D]	$k < l$, atd.
before	$\sigma \times \sigma$	$\rightarrow$ bool [1D]	$\forall k \in K, \forall l \in L : k \leqslant l$
	$\pi \times \sigma$	$\rightarrow$ bool [1D]	$\forall l \in L : k \leqslant l$
	$\sigma \times \pi$	$\rightarrow$ bool [1D]	$\forall k \in K : k \leqslant l$
touches	$\sigma \times \sigma$	$\rightarrow$ bool	$\partial K \cap \partial L \neq \emptyset$
attached	$\sigma \times \sigma$	$\rightarrow$ bool	$\partial K \cap V^o \neq \emptyset$
overlaps	$\sigma \times \sigma$	$\rightarrow$ bool	$K^o \cap L^o \neq \emptyset$
on_border	$\pi \times \sigma$	$\rightarrow$ bool	$k \in \partial K$
in_interior	$\pi \times \sigma$	$\rightarrow$ bool	$k \in K^o$

Tabulka 2.3 Přehled binárních predikátů [1]

2.3.3 Množinové operace

Množinové operace jsou dostupné pro všechny typy definované nad množinami hodnot, tj. pro typy line, range(int) atd. Kde je to přípustné, dovolíme množinové operace i nad jednotlivými body. Jednoprvkové množiny nebo prázdné množiny, které mohou vzniknout jako výsledek množinové operace, interpretujeme jako bodové hodnoty a to díky tomu, že doména zahrnuje i nedefinovanou hodnotu, již ztotožňujeme s prázdnou množinou.

Protože dvojdimenzionální (prostorové) typy jsou uzavřené, je nezbytné, po aplikaci některých množinových operací, aplikovat na výsledek uzávěrovou operaci. Při použití množinových operací nad typy různých dimenzí nás zajímá výsledek v nejvyšší dimenzi, tj. při operacích průniku, sjednocení a rozdílu odstraníme z výsledku všechny části z nižších dimenzí. Přehled množinových operací se nachází v tabulce 2.4.

Operace	Signatura		Sémantika
intersection	π x π	$\rightarrow \pi$	**if** $k = l$ **then** k **else** undef.
	π x σ, σ x π	$\rightarrow \pi$	**if** $k \in L$ **then** k **else** undef.
	σ x σ	$\rightarrow \sigma$	$K \cap L$
minus	π x π	$\rightarrow \pi$	**if** $k = l$ **then** undef. **else** k
	π x σ	$\rightarrow \pi$	**if** $k \in L$ **then** undef. **else** k
	σ x π	$\rightarrow \sigma$	**if** $is2D(K)$ **then** $\rho(K \setminus \{1\})$ **else** $K \setminus \{1\}$
	σ x σ	$\rightarrow \sigma$	**if** $is2D(K, L)$ **then** $\rho(K \setminus L)$ **else** $K \setminus L$
union	π x σ, σ x π	$\rightarrow \sigma$	**if** $is1D(L) \vee type(L) = points$ **then** $L \cup \{k\}$ **else** L
	σ x σ	$\rightarrow \sigma$	$K \cup L$
crossings	line x line	$\rightarrow$ points	viz kapitola 2.3.3
touch_points	region x line	$\rightarrow$ points	
	line x region	$\rightarrow$ points	
	region x region	$\rightarrow$ points	
common_border	region x region	$\rightarrow$ line	

Tabulka 2.4 Přehled množinových operací [1]

Pro definici sémantiky v tabulce 2.4 jsou použity predikáty *is1D* a *is2D* pro kontrolu, zdali je argument jednodimenzionálního resp. dvojdimenzionálního typu.

Tak, jak jsme definovali operaci průniku, která vrací výsledek v nejvyšší možné dimenzi, tj. bere v úvahu i dimenzionální uspořádání point < points < line < region, tak v některých případech je žádoucí brát jako výsledek průniku nižší dimenzi. Pro představu, při průniku regionu a křivky, je možným výsledkem množina bodů (nižší dimenze) nebo množina úseček (vyšší dimenze).

Pro popis sémantiky následujících operací využijeme variantu operace průniku, která vrací výsledek v nižší dimenzi. Operace *crossings(line1, line2)* vrací množinu bodů - průsečíků úseček. Operace *touch_points(line, region)* vrací množinu průsečíků úseček s konturou regionu a operace *touch_points(region1, region2)* vrací množinu průsečíků hraničních kontur dvou regionů. Operace *common_border(region1, region2)* vrací uzávěr průniku dvou regionů.

Některé z těchto operací budou použity v námi navržené databázi pohybujících se objektů.

2.3.4 Agregace

Agregace redukují množinu hodnot do jedné hodnoty. V jednodimenzionálním prostoru, kde existuje relace úplného uspořádání, máme operace *min* a *max*, které vrací minimální a maxi-

mální hodnotu. Operace *avg* vrací průměrnou hodnotu, ve dvojdimenzionálním prostoru je založena na vektorovém součtu a vrací těžiště. K dispozici jsou i další agregační funkce, např. *sum*, *count*.

2.3.5 Operace vzdálenosti a směru

U spojitých typů můžeme měřit vzdálenosti, operace *distance* vrací minimální vzálenost nejbližší dvojice bodů z prvního a druhého argumentu. U časové domény je využit fakt, že její nosnou množinou je množina reálných čísel.

Pro určení směru je využita operace *direction*, která vrací úhel přímky, tvořené dvěma body, s osou x. Jestliže jsou body tvořící přímku totožné, operace direction vrací nedefinovanou hodnotu.

2.4 Operace nad temporálními typy

Hodnotami temporálních typů jsou funkce, které mapují nosnou množinu časových okamžiků do nosné množiny typu α.

2.4.1 Projekce do domény a rozsahu

Pro temporální typy, tj. typy vzniklé pomocí konstruktoru moving, existuje množství operací, které poskytují projekci do jejich domény a rozsahu. Operace *deftime* vrací čas, po který je hodnota typu moving(α) definována, výsledným typem je typ periods. Pro typy z jednodimenzionálního prostoru, operace *rangevalues* vrací hodnoty přiřazené během času jako množinu intervalů, tj. výsledek je typu range(α).

Operace	Signatura	
deftime	moving(α)	→ periods
rangevalues	moving(α)	→ range(α) [1D]
locations	moving(point)	→ points
	moving(points)	→ points
trajectory	moving(point)	→ line
	moving(points)	→ line
traversed	moving(line)	→ region
	moving(region)	→ region
inst	intime(α)	→ instant
val	intime(α)	→ α

Tabulka 2.5 Operace pro projekci do domény a rozsahu [1]

Operace *locations* poskytuje pro typ moving(point) jeho projekci do bodů v rovině a operace *trajectory* jeho projekci do úseček v rovině. Výsledkem operace *traversed* pro pohybující se objekt (typu line nebo region) je uzavřený region.

Pro hodnoty typu intime(α), které jsou tvořeny párem (okamžik, hodnota typu α), operace *inst* poskytuje projekci do domény typu instant a operace *val* projekci do domény typu α. Přehled operací pro projekci temporálních hodnot do domény a rozsahu je zobrazen v tabulce 2.5.

2.4.2 Interakce

Tato podkapitola se zabývá interakcemi pohybujících se objektů v čase. Mezi interakce může patřit případ, kdy bod pohybující se v rovině prošel prostorem daného regionu nebo když proměnná typu moving(int) nabyla určité hodnoty z množiny celých čísel. Dále operace interakcí mohou sloužit, např. pro určení jaké hodnoty nabyla proměnná typu α v čase t nebo v rozmezí dvou zadaných časových okamžiků atd. Přehled operací se nachází v tabulce 2.6.

Operace *atinstant* omezuje pohybující se objekt do zadaného okamžiku, jejím výsledkem je pár (okamžik, hodnota), zatímco operace *atperiods* jej omezuje do zadaného časového intervalu a jejím výsledkem je konečná množina párů (okamžik, hodnota). Operace *initial* a *final* vrací první a poslední pár (okamžik, hodnota). Pomocí operace *present* můžeme zjistit, zda pohybující se objekt existuje v daném časovém okamžiku nebo zdali existoval během daného časového intervalu, výsledek operace je typu bool.

Operace	Signatura	
atinstant	moving(α) x instant	→ intime(α)
atperiods	moving(α) x periods	→ moving(α)
initial	moving(α)	→ intime(α)
final	moving(α)	→ intime(α)
present	moving(α) x instant	→ bool
	moving(α) x periods	→ bool
at	moving(α) x α	→ moving(α) [1D]
	moving(α) x range(α)	→ moving(α) [1D]
	moving(α) x point	→ mpoint [2D]
	moving(α) x β	→ moving(α) [2D]
atmin	moving(α)	→ moving(α) [1D]
atmax	moving(α)	→ moving(α) [1D]
passes	moving(α) x β	→ bool

Tabulka 2.6 Operace interakcí [1]

Operace *at* je obdobou operací atinstant, atperiods s tím rozdílem, že pohybující se objekt omezujeme do zadané hodnoty nebo rozsahu hodnot. Např. můžeme omezit proměnnou typu moving(int) pouze na dobu, kdy nabývala hodnoty mezi 5 a 10. Výsledkem je poté konečná množina párů (okamžik, hodnota z daného rozsahu). Ve dvojdimenzionálním prostoru můžeme operací at omezit pohybující se objekt jiným objektem prostorového typu. Výsledek je typu moving(α), kde α je stejného typu, jako objekt nižší dimenze z těchto dvou objektů podle uspořádání point < points < line < region.

Operace *atmin* a *atmax* omezují pohybující se objekt pouze na dobu, kdy je jeho hodnota minimální nebo maximální s ohledem na relaci úplného uspořádání v jednodimenzionálním prostoru. Operace *passes* umožňuje kontrolu, zdali pohybující se objekt někdy nabyl zadané hodnoty nebo hodnot z daného rozsahu, výsledek je typu bool.

2.4.3 Přizpůsobení operací na časově závislé operace

Tato podkapitola se zabývá přizpůsobením operací nad nontemporálními typy pro jejich aplikaci nad temporálními typy. Smyslem je umožnit, aby argumentem operace byl temporální typ a jejím výsledkem také temporální typ. Tedy každý typ argumentu je možné změnit na časově závislý typ, který dále přemění výsledný typ také do časově závislého typu. Např. pro operaci intersection nad nontemporálními typy, která má signaturu:

- *point* x *region* → *point*

její přizpůsobená verze nad temporálními typy může mít signaturu:

- *mpoint* x *region* → *mpoint*
- *point* x *mregion* → *mpoint*
- *mpoint* x *mregion* → *mpoint*

Principem metody je tedy konverze proměnné typu, která nabývá určité hodnoty nebo rozsahu hodnot do proměnné odpovídajícího temporálního typu, která je tvořena párem (okamžik, hodnota), respektive množinou párů (okamžik, hodnota).

Z operací definovaných nad základními a prostorovými typy jsme tak získali časově závislé operace. To nám umožňuje omezit časově závislou hodnotu do intervalů, kdy tato hodnota splňuje určitou vlastnost danou predikátem. S využitím pouze operací definovaných nad základními a prostorovými typy by takovéto omezení nebylo možné, neboť by bylo nutné vyhodnotit danou operaci nekonečně mnohokrát.

Pro ilustraci vezměme případ, kdy budeme považovat auto za pohybující se objekt pouze v době, kdy jeho rychlost je větší než nula. Se základními operacemi by bylo nutné vyhodnocovat jeho rychlost na nekonečné časové ose, jestliže ale uvažujeme auto jako proměnnou temporálního typu, jejíž hodnota je tvořena konečnou množinou párů (okamžik, hodnota), je takovéto vyhodnocení možné.

2.4.4 Rychlost změny

Jednou z vlastností proměnné časově závislého typu je její rychlost změny, výpočet této rychlosti je založen na diferenciálním počtu. Výpočet rychlosti změny je možný pro typ mreal a to operací *derivative* a pro typ mpoint třemi následujícími operacemi. Operace *speed* je založena na Euklidovské vzdálenosti, operace *turn* na směru mezi dvěma body a operace *velocity* na rozdílu vektorů. Pro typ mpoint můžeme vypočítat zrychlení bodu pomocí operace derivative(speed(mpoint)).

Kapitola 3

Podpora časoprostorových dat v Oracle10g

Databázový server Oracle10g nepodporuje přímo časoprostorová data, ale obsahuje zvlášť podporu pro časová a prostorová data.

3.1 Podpora prostorových dat

Prostorové rozšíření systému řízení báze dat (*SŘBD*) obsahuje množinu funkcí a procedur, které umožňují ukládat, přistupovat a analyzovat prostorová data. Prostorová data reprezentují základní charakteristiky umístění objektů, tj. jak se objekty vztahují k prostoru, ve kterém se nacházejí. Pro reprezentaci prostorových dat je obvykle používán *objektově-relační model dat*.

3.1.1 Model dat

Objektově-relační model dat umožňuje ukládat celou geometrii v prostorovém objektovém datovém typu *SDO_GEOMETRY*, přičemž tabulka může obsahovat jeden nebo více SDO_GEOMETRY sloupců. Prostorový datový model je hierarchická struktura elemetů, geometrií a vrstev, kde geometrie jsou tvořeny elementy a kolekce geometrií vytvářejí vrstvy.

Geometrie je uspořádaná posloupnost vrcholů, které jsou propojeny úsečkami nebo kružnicovými oblouky. Umožňuje nám pracovat s matematickými vlastnostmi objektů, jako jsou rozměry a vzájemné vztahy mezi body, křivkami atd. Geometrií může být bod, křivka, polygon, kruh a další složitější útvary vytvořené ze základních elementů.

SŘBD také podporuje ukládání a indexaci troj a čtyřdimenzionálních geometrických typů, kde souřadnice definují vrcholy prostorových entit. Avšak prostorové funkce, s vyjímkou funkcí *lineárního referenčního systému*, mohou pracovat s nanejvýš dvojdimenzionálními entitami a prostorové operátory, s výjimkou SDO_FILTER, není možné použít, jestliže byl *prostorový index* vytvořen na více než dvou dimenzích (viz dále).

3.1.2 Prostorové datové typy

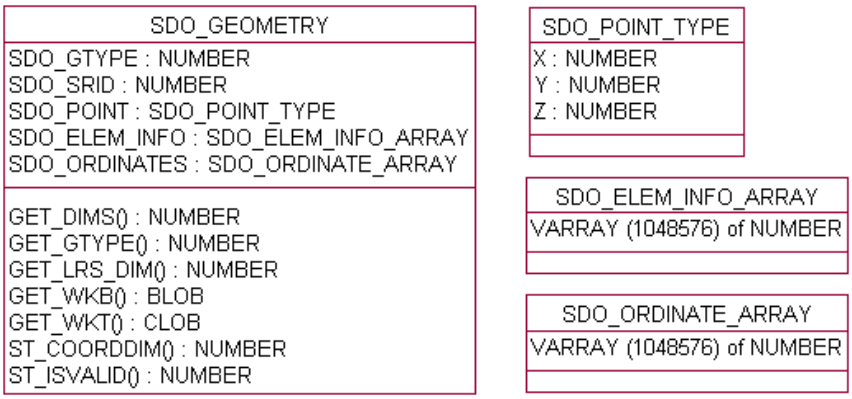
Prostorové rozšíření SŘBD sestává z množiny objektových datových typů, metod typů a operátorů, funkcí a procedur používajících tyto typy. Objektový datový typ SDO_GEOMETRY je kontejner pro ukládání bodů, úseček, polygonů a homogenních nebo heterogenních kolekcí těchto elementů.

Atributy typu SDO_GEOMETRY sestávají z identifikátoru typu geometrie (atribut SDO_GTYPE), identifikátoru souřadnicového systému (atribut SDO_SRID), pole deskriptorů elementů (atribut SDO_ELEM_INFO) a souřadnicového pole (atribut SDO_ORDINATES), jež obsahuje

hodnoty párů nebo trojic souřadnic, které definují vrcholy elementů. Pole deskriptorů elementů určuje, jak by tyto souřadnice měly být přiřazeny k elementům, jež tvoří geometrii. Dále určuje, zda pár nebo trojice vrcholů je propojena přímkou nebo obloukem kružnice.

Nejsou-li atributy SDO_ELEM_INFO a SDO_ORDINATES definovány, tj. obsahují-li hodnotu NULL a současně atribut SDO_POINT je definován, poté je jeho pole hodnot považováno za souřadnice bodu v prostoru. V ostatních případech je atribut SDO_POINT ignorován.

Struktura typu SDO_GEOMETRY a struktura typů jeho atributů je znázorněna na obr. 2.1.



Obrázek 2.1 Typ SDO_GEOMETRY

3.1.3 Souřadnicový systém

Souřadnicový systém je prostředek pro přiřazení souřadnic k polohám a pro vytvoření vazeb mezi množinami souřadnic. Umožňuje interpretovat množinu souřadnic jako reprezentaci polohy v reálném světě. Všechna prostorová data mají k sobě přiřazen souřadnicový systém, který může, ale také nemusí, být *georeferenční*, tj. související se Zemskou polohou [4].

SŘBD poskytuje podporu pro různé souřadnicové systémy a pro převádění dat mezi nimi. Prostorová data mohou být přidružena k následujícím souřadnicovým systémům:

- Kartézské souřadnice slouží pro měření polohy bodu od definovaného počátku, tj. nejedná se o georeferenční souřadnicový systém. Není-li uvedeno jinak, kartézský souřadnicový systém je implicitně přiřazen ke geometrii.
- Geodetické souřadnice jsou úhlové souřadnice pro vyjádření zeměpisné délky a šířky.
- Promítnuté souřadnice jsou rovinné kartézské souřadnice, jež jsou výsledkem matematického mapování bodu na Zemském povrchu do roviny.
- Lokální souřadnice jsou kartézské souřadnice, sloužící pro vyjádření poloh nevztahujících se k Zemi, používají se např. pro CAD aplikace.

K souřadnicovému systému se váže i tolerance, která určuje míru přesnosti prostorových dat, tj. určuje vzdálenost, kdy jsou dva body ještě považovány za totožné. Význam hodnoty tolerance závisí na použitém souřadnicovém systému.

3.1.4 Dotazování

Pro provedení dotazu je použit dvouvrstvý dotazovací model, tj. jsou provedeny dvě rozdílné operace a výsledkem je kombinace výsledků těchto dvou operací. Operace jsou označované jako *primární* a *sekundární* filtrační operace:

- Primární filtrační operace umožňuje rychlý výběr záznamů pro jejich zpracování v sekundárním filtru. Jejím účelem je porovnat přibližnost geometrií pro snížení výpočetní složitosti. Jejím výsledkem je množina možných správných výsledků.
- Sekundární filtrační operace provádí přesný výpočet na množině potenciálně správných výsledků z primárního filtru. Jejím výsledkem je správná odpověd na prostorový dotaz. Kvůli její výpočetní náročnosti je nejdříve aplikován primární filtr, díky kterému již není nutné procházet celou datovou množinu.

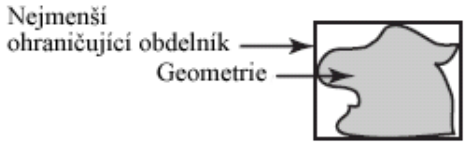
Účelem primárního filtru je rychlé vytvoření podmnožiny dat a tím snížení režie zpracování v sekundárním filtru. Pro implemetaci primárního filtru je použita indexace prostorových dat.

3.1.5 Indexace prostorových dat

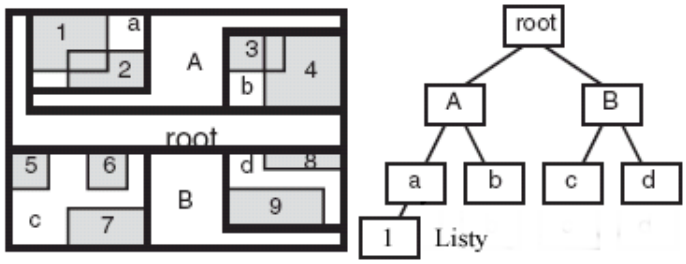
Účelem indexace prostových dat je, jako u jiných indexů, omezení vyhledávání v množině dat. Mechanismus indexace prostorových dat je založen na prostorových kritériích, jako např. průnik a inkluze.

Prostorový index je považován za logický index, který závisí na umístění geometrie v souřadnicovém prostoru. Pro indexaci prostorových dat je nejčastěji používán index v podobě R-stromu, který umožňuje indexaci až čtyřdimenzionálních prosotorových dat.

Index v podobě R-stromu je založen na vytvoření nejmenšího obdelníku, který obklopuje danou geometrii, čímž dojde k jejímu zjednodušení, obrázek 2.2. Pro vrstvu geometrií index v podobě R-stromu sestává z hierarchicky uspořádaného stromu indexů pro jednotlivé geometrie, obrázek 2.3.



Obrázek 2.2 Nejmenší obdelník ohraničující geometrii [4]



Obrázek 2.3 Hierarchický strom indexů v podobě R-stromu [4]

3.1.6 Klasifikace prostoru

Prostor je rámec pro formalizaci specifických vztahů nad množinou prostorových objektů. V závislosti, jaké vztahy nás zajímají, existují různé modely prostoru, jako např. topologický, síťový nebo metrický prostor.

Topologický prostor používá pojem sousedství pro formalizaci vztahů mezi body. Topologie mohou být uzavřené, propojené, překrývající se nebo se jedna může nalézat uvnitř druhé. Síťový prostor pracuje s pojmy jako nejkratší cesta a spojitost. Metrický prostor formalizuje vzdálenostní vztahy.

3.1.7 Prostorové vztahy a filtrování

Pro určení prostorových vztahů mezi entitami v databázi je používán již zmíněný sekundární filtr. Prostorové vztahy jsou založené na umístění geometrií, nejčastěji jsou založené na topologii a vzdálenosti. Např. určení přilehlosti dvou oblastí je založené na společných bodech, tvořících hraniční křivky, které oddělují dané oblasti od zbytku souřadnicového prostoru.

Vzdálenost mezi dvěma prostorovými entitami je minimální vzdálenost mezi dvěma body z prostorových entit. Pro určení prostorových vztahů existuje několik sekundárních filtračních metod:

- Operátor SDO_RELATE vyhodnocuje topologická kritéria.
- Operátor SDO_FILTER určuje, které geometrie mohou být v interakci s danou geometrií.
- Operátor SDO_WITHIN_DISTANCE určuje, zda dvě prostorové entity se nálezají uvnitř dané vzdálenosti.
- Operátor SDO_NN identifikuje nejbližší sousední entity pro danou prostorovou entitu.
- Operátor SDO_NN_DISTANCE vrací vzdálenost sousedních entit vrácených operátorem SDO_NN.

Operátor SDO_RELATE rozlišuje devět možných interakcí mezi body, křivkami a polygony. Dvě prostorové entity mohou být disjunktní, sobě rovné, mohou se dotýkat, překrývat, jedna entita může být uvnitř druhé atd.

V dotazu musí být prostorový operátor použit v klauzuli WHERE. První parametr operátoru určuje sloupec geometrií v tabulce, jež bude prohledáván a druhý parametr určuje *okno dotazu*. Okno dotazu prostorově omezuje prohledávaný prostor, ve kterém se nacházejí námi hledané prostorové entity.

3.1.8 Prostorové procedury a funkce

Prostorové procedury a funkce jsou poskytnuty jako podprogramy v PL/SQL balíčcích, jako např. SDO_GEOM, SDO_CS a SDO_LRS. Tyto podprogramy nevyžadují a ani nepoužívají prostorové indexy. Mohou být použity v klauzuli WHERE nebo v poddotazech. Geometrie, jež jsou vstupními parametry procedury nebo funkce, musí používat stejný souřadnicový systém.

Prostorové agregační funkce vracejí agregovaný výsledek SQL dotazu, jako objekt typu SDO_GEOMETRY. Všechny geometrie používané v agregačních funkcích musí být definovány s použitím čtyř číselných SDO_GTYPE hodnot.

3.1.9 Lineární referenční systém

Lineární referenční systém slouží pro přidružení atributů nebo událostí k polohám nebo k částem lineárních vlastností. Je často používaný v dopravních aplikacích, jeho výhodou je schopnost zachycení atributů a událostí spolu s lineárními vlastnostmi pomocí pouze jednoho atributu, označovaného jako *míra* (anglicky *measure*), namísto dvou atributů, např. zeměpisné délky a šířky. Části lineárních vlastností mohou být tvořeny a odkazovány dynamicky, aniž by bylo nutné je explicitně ukládat, díky uvedení počáteční a koncové polohy spolu s vlastnostmi.

Segmentem geometrie může být křivka nebo polygon a musí obsahovat míru přinejmenším pro počáteční a koncový bod. Míry jsou přiřazené uživatelem nebo odvozené od již existujících částí geometrie. Bod na segmentu geometrie je reprezentován trojicí (x, y, m), kde *x* a *y* označují polohu a *m* značí míru.

Atribut míry u bodu na segmentu geometrie je lineární vzdálenost tohoto bodu od počátečního nebo koncového bodu segmentu, na kterém leží. Informace o míře nemusí nutně být ve stejném měřítku jako vzdálenost.

Směr segmentu geometrie je určen pořadím vrcholů od počátečního ke koncovému bodu segmentu. Míra u bodů je buď stoupající nebo klesající a to podle směru segmentu.

3.2 Podpora časových dat

Pro ukládání časových informací slouží datové typy DATE a TIMESTAMP.

3.2.1 Datový typ DATE

Datový typ DATE ukládá do tabulky hodnotu časového okamžiku, tj. ukládá století, rok, měsíc, den, hodinu, minutu a sekundu. Oracle pro uložení data používá svůj vnitřní formát, který má pevnou délku sedmi bytů. Standardní formát data ve tvaru DD-MON-YY je možné změnit pomocí parametru NLS_DATE_FORMAT nebo pro zadání data, které není ve standardním formátu, lze použít funkci TO_DATE, u které se jako druhý parametr uvede formát zadávaného data.

Oracle ukládá čas ve 24 hodinovém formátu HH:MI:SS. Není-li s datem současně zadán i čas, je implicitně uložena hodnota času 00:00:00. Je-li ukládán pouze čas, jako datum je implicitně uložen první den aktuálního měsíce a roku.

3.2.2 Datový typ TIMESTAMP

Datový typ TIMESTAMP je rozšířením typu DATE a je vhodný pro ukládání přesných hodnot času. Ukládá rok, měsíc a den datového typu DATE plus hodinu, minutu a hodnotu sekund. Umožňuje zvolit přesnost sekundové hodnoty pomocí TIMESTAMP [zlomek_sekundy], kde *zlomek_sekundy* je celé číslo od 0 do 9, udávající počet desetiných míst, implicitně je zlomek_sekundy nastaven na 6.

Datový typ TIMESTAMP WITH TIME ZONE je vatiantou typu TIMESTAMP, který zahrnuje informaci o časovém pásmu. Tento datový typ je vhodný pro ukládání a vyhodnocování času napříč zeměpisnými oblastmi.

Kapitola 4

Návrh struktury databáze

Současné systémy řízení báze dat nejsou příliš vhodné pro práci s časoprostorovými daty, které se průběžně v čase mění. Jejich účelem je uchovávat aktuální konstantní data, dokud nejsou explicitně modifikována. Současně je při modifikaci ztracen předchozí stav databáze. Naproti tomu u pohybujících se objektů je nezbytné velmi často provádět aktualizace a při dotazování nás zajímají, kromě současných poloh objektů i jejich polohy minulé a možné budoucí.

Dnešní SŘBD obsahují podporu pro časová a prostorová data, avšak ne pro integrovaná časoprostorová data. Přesto bude muset být podpora pohybujících se objektů postavena právě na základě existujících SŘBD. Našim úkolem je tedy zvolit vhodnou datovou strukturu, která bude obsahovat podporu pro spojitě měnící se geometrie, tj. pro pohyb objektů a současně bude použitelná ve stávajících systémech řízení báze dat.

4.1 Úvod do problematiky

V našem případě budou data pocházet ze senzorového systému, tvořeného soustavou kamer, který snímá v daném časovém okamžiku polohu pohybujících se objektů v prostoru. Objekty jsou reprezentovány jako body v prostoru, kde každý bod je tvořen trojicí (x, y, t), tj. k informaci o poloze bodu v rovině je přidána časová informace. Pohybující se objekty jsou tedy reprezentovány jako body v trojdimenzionálním prostoru, kde třetí dimenzi tvoří čas.

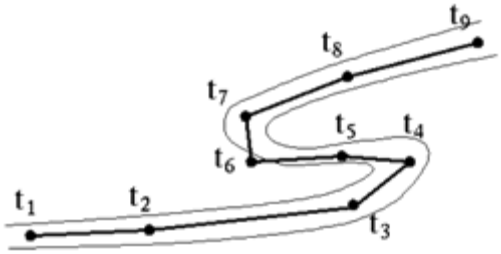
Pohybující se objekt vytváří *trajektorii*, což je globální sekvence poloh daného objektu. Trajektorie může být vyjádřena jako sekvence bodů (x, y, t) nebo jako funkce času. Pomocí funkce času můžeme popisovat současnou a budoucí polohu objektu. V tomto případě nejsou do databáze ukládány polohy objektu, ale informace o pohybu, ze kterých je možné vypočíst polohu v jakémkoliv časovém okamžiku.

Pro ilustraci, jestliže provedeme aktualizaci databáze, je do ní uložena aktuální rychlost a směr pohybujícího se objektu, které se mohou nadále měnit, ale frekvence jejich změn je zcela jistě nižší než frekvence změn u polohy objektu. Z toho vyplývá méně častá potřeba provádět aktualizaci databáze, na druhou stranu ale databáze obsahuje pouze aktuální stav objektů a nejsou tedy možné dotazy týkající se minulosti.

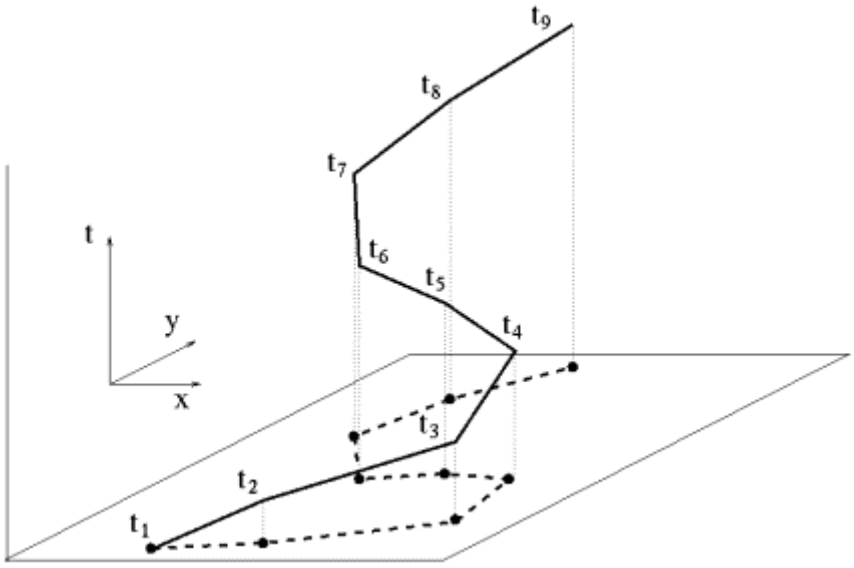
Našim úkolem je nyní zvolit vhodnou strukturu dat pro ukládání trajektorií do databáze. V následujích podkapitolách budeme diskutovat dva přístupy, v prvním je trajektorie reprezentována jako množina bodů v trojdimenzionálním prostoru. Ve druhém přístupu, jež je založen na *síťovém grafu*, je každý bod považován za uzel, který je spojen hranou se sousedními uzly.

4.2 Trojdimenzionální reprezentace

Pohybující se bod je základní abstrakcí fyzického objektu, který mění svoji polohu v čase. Pohyb bodu v rovině je možné popsat trajektorií, jež je zobrazena na obrázku 3.1. Pro zachycení časové informace je možné, do původní reprezentace bodu v rovině, přidat třetí, časovou dimenzi. Bod je poté ukládán v databázi jako prostorová entita tvořená trojicí (x, y, t), jejíž pohyb v prostoru lze také znázornit trajektorií, obrázek 3.2.



Obrázek 3.1 Trajektorie v rovině



Obrázek 3.2 Trajektorie v trojdimenzionálním prostoru

4.2.1 Reprezentace času

Obecně lze na čas nahlížet jako na absolutní (16.12.2007, 23:25:00) nebo relativní (dva dny). Pro reprezentaci času v prostorovém modelu dat je nutná transformace absolutního času na relativní, vzhledem k danému počátku časové osy. Rozpětí času můžeme prezentovat jako interval se známou délkou, ale bez specifikovaného počátku a konce. Protože čas považujeme za spojitý, je nutné jej reprezentovat hodnotou typu real.

Nicméně je však obtížné mapovat spojitý čas do množiny reálných čísel, neboť je nemožné přesně určit hodnotu času. Proto je často s časem nakládáno jako s diskétní doménou, kde jsou časové jednotky mapovány do množiny celých čísel. Je tedy nutné určit granularitu časových jednotek, může se jednat o dny, hodiny, minuty apod. Čím je granularita tvořena menšími časovými úseky, tím blíže je diskrétní čas ke spojitému. Je také možné použít různou granularitu pro různé typy objektů, např. pro trajektorii použít sekundy, zatímco pro události použít minuty, událostí může být např. přelet letadla do vzdušného prostoru jiné země. Při vyhodnocování dotazů nad různými typy objektů je potom nutné granularitu unifikovat [5].

4.2.2 Model dat

Pro ukládání dat v databázi lze použít objektově-relační přístup nebo objektově orientovaný přístup. Např. pohybující se bod můžeme v objektově-relačním SŘBD reprezentovat jako uživatelsky definovaný typ a v objektově orientovaném SŘBD jako třídu.

Klíčovou otázkou je, jak zaznamenávat aktuální pozici pohybujícího se objektu, která se v čase neustále mění. Je výhodné vytvořit nový objekt, reprezentující trajektorii, který zachází s měnící se polohou objektu. Existují tři důvody, proč ukládat informace o trajektorii objektu. Prvním je možnost vyhodnocovat dotazy, týkající se polohy pohybujícího se objektu. Za druhé, informace z trajektorie mohou být použity pro zjištění a predikci určitých vzorů pohybu. A za třetí, informace z trajektorie mohou být užity pro zjištění okolností pohybu, např. doba letu letadla.

Patrně největší výhodou trojdimenzionální reprezentace časoprostorových dat je, že je poměrně obecná a lze jí takto reprezentovat všechny temporální typy popsané v kapitole 2.2.4. Tato skutečnost je zásadní ve srovnání s modelem založeném na síťovém grafu, který je poměrně jasný na pochopení, méně náročný na implementaci, ale který umožňuje reprezentovat pouze pohybující se bod a jeho trajektorii.

4.2.3 Dotazy nad prostorovými entitami

Obecně existují dva typy dotazů nad body, které reprezentují pohybující se objekty. Prvním jsou dotazy, týkající se polohy bodu v daném čase a druhým typem jsou dotazy na čas, kdy se bod nacházel na určité pozici nebo množině pozic.

Další množinu dotazů tvoří dotazy, týkající se vzájemných vztahů mezi prostorovými entitami. Je snažší porovnat dvě proměnné typu integer než dva plygony nebo dvě křivky. Naše databáze pohybujících se objektů přesto bude muset implemetovat některé operace, týkající se takových vztahů. Např. nás může zajímat v kolika různých, senzory snímaných prostorech, se daný objekt vyskytoval. V tomto případě se jedná o relaci prostorových entit typu region a point.

Pro některé dotazy nad prostorovými daty můžeme s výhodou využít prostorového rozšíření SŘBD databázového serveru Oracle10g, které nám poskytuje prostorové oprátory, jako např. SDO_RELATE, SDO_FILTER, SDO_WITHIN_DISTANCE, SDO_NN_DISTANCE a další.

4.2.4 Vizualizace časoprostorových dat

Jedním z problémů trojdimenzionální reprezentace časoprostorových dat je jejich vizualizace. S běžnými prostředky můžeme zobrazovat data jen ve dvou dimenzích. Je tedy nevyhnutelné, že časová dimenze bude zobrazena jiným způsobem než umožňuje trojdimenzionální model. Jedním z možných řešení je textová reprezentace časové informace u pozic bodů v rovině, které společně vytvářejí trajektorii.

4.3 Síťová reprezentace

I v tomto druhém přístupu bude pohyb objektu zaznamenáván pomocí trajektorie. Pohybující se objekt bude reprezentován výhradně bodem, jehož časově proměnlivé pozice tvoří koncové body jednotlivých segmentů trajektorie, která znázorňuje celkový pohyb objektu. Reprezentace pohybu pomocí trajektorií je dostačující pro odvození určitých vlastností a vztahů pohybujících se objektů.

4.3.1 Trajektorie

Trajektorie je charakterizována množinou vlastností, jako např. rychlost a směr pohybu, uražená vzdálenost a doba pohybu. Trajektorie je ve vztahu s okolním prostorem i jinými trajektoriemi, jedná se např. o vzdálenosti mezi trajektoriemi, křížení trajektorií, vstup do určitého prostoru, výstup z prostoru atd.

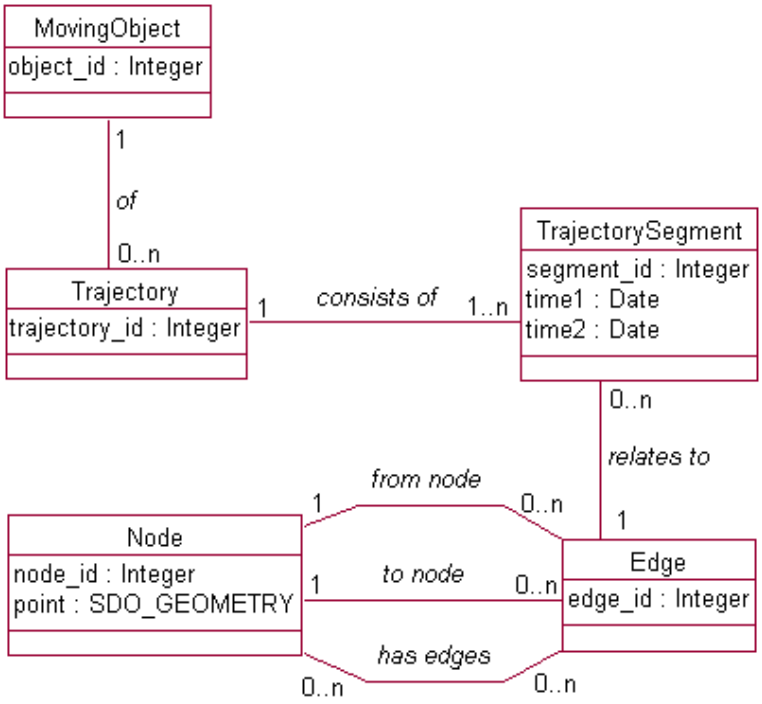
Pro ukládání dat v databázi je vhodné použít objektově-relační přístup, kde uživatelsky definovaný typ Trajektorie bude obsahovat identifikaci pohybujícího se objektu, identifikaci trajektorie a její polohu, tj. dráhu objektu.

4.3.2 Datový model pro trajektorii

Reprezentaci trajektorie můžeme založit na síťovém grafu, pomocí kterého lze znázornit dráhu objektu. Na obrázku 3.3 je znázorněno konceptuální schéma pro návrh a pozdější implementaci trajektorie prostřednictvím sítě. Schéma obsahuje informace o objektu, trajektorii a síti.

Síť modelující trajektorii se skládá z uzlů (NODE) a hran (EDGE), kde uzel představuje bod v rovině. Každá hrana je v relaci se dvěma uzly a navíc uchováváme informaci o vedlejších hranách uzlů (HAS EDGES), např. pro znázornění dopravní sítě. Trajektorie sestává ze segmentů, které jsou ve vztahu (RELATE TO) k hranám sítě. Uzly spolu s hranami definují prostorový rozměr trajektorie. Časový aspekt trajektorie je vyjádřen přiřazením dvou časových značek ke každému segmentu, které udávají počáteční a koncový čas [5].

Výhodou na síti založenému modelu dat je jeho podpora v prostorovém rozšíření SŘBD databázového serveru Oracle10g.



Obrázek 3.3 Schéma pro návrh a implementaci trajektorie

4.3.3 Podpora síťového modelu v Oracle 10g

V databázovém serveru Oracle 10g je zabudována podpora pro topologie a síťové datové modely. SŘBD je schopen spravovat informace o uzlech, hranách a plochách a také poskytuje množinu funkcí pro práci nad topologickými daty. Dále SŘBD implementuje síťový model, který obsahuje vestavěné síťové algoritmy a spravuje geometrické informace, jako např. propojení mezi uzly a hranami, směr propojení, ohodnocení uzlů a hran atd.

4.3.4 Vlastnosti navrženého modelu

Největší nevýhodou navrženého modelu, založeného na síti je, že ze své podstaty, ze všech prostorových entit, podporuje pouze body. Pro náš senzorový systém, kde všechny objekty budou reprezentovány jako pohybující se body, to příliš velkým problémem není. Tato nevýhoda se také dá dále zmírnit, pakliže určitá část systému, kde je to možné, bude implementována s využitím síťového modelu a zbylá část bude založena na trojdimenzionální reprezentaci časoprostorových dat.

Výhodou síťového modelu dat je možnost použití absolutního času při ukládání časoprostorových informací a to narozdíl od trojdimenzionální reprezentace dat pomocí objektů typu SDO_GEOMETRY, u které je možné ukládat pouze relativní čas.

Kapitola 5

Implementace databáze

Návrh struktury databáze, jež bude implementována, vychází ze dvou výše uvedených přístupů, ve kterých je model dat založen na trojdimenzionální resp. síťové reprezentaci trajektorie. Z nich si návrh struktury databáze bere filosofii síťové reprezentace trajektorie, která však není implentována s využitím podpory síťového modelu v prostorovém rozšíření SŘBD a to z důvodu ukládání času, ale pomocí *uživatelsky definovaných datových typů*, které zahrnují trojdimenzionální data a současně představují jednotlivé prvky trajektorie, tj. uzly, hrany atd.

Tento přístup je možný, neboť pohybující se objekt je reprezentován výhradně bodem, jehož časově proměnlivé pozice tvoří koncové body jednotlivých segmentů trajektorie, která znázorňuje celkový pohyb objektu. To vyplývá z domény použití námi vytvářeného systému, kterým je síť senzorů, snímající pohyb objektů, které se jeví pouze jako body s identifikací, mající v daném časovém okamžiku určitou polohu.

Podstatou celého návrhu struktury databáze je tedy otázka uložení a prezentace pohybujícího se bodu. Ta a další jsou podrobněji popsány v následujících kapitolách.

5.1 Model dat

Pro implementaci navržené databáze je použit objektově-relační model dat, kdy jsou jednotlivé entity reprezentovány pomocí uživatelsky definovaných datových typů, které zapouzdřují strukturu dat a také obsahují uživatelem vytvořené procedury a funkce pro práci nad těmito daty. Jedná se o obdobu uložených procedur a funkcí databázového serveru, ke kterým je však přistupováno kvalifikovaným jménem přes název uživatelsky definovaného datového typu, ke kterému náležejí.

V našem případě tyto procedury a funkce zajišťují prostorové a časoprostorové operace nad prostorovými a časoprostorovými daty. Jsou implementovány v jazyce *PL/SQL*, který nám umožňuje využívat pokročilé vlastnosti databázového serveru Oracle 10g a to zejména prostorové datové typy a vestavěné prostorové operace, což je velkou výhodou oproti jiným imperativním jazykům, které nejsou tak těsně svázány s databázovým serverem. Implementace těchto procedur a funkcí by např. v jazyce Java byla mnohem obtížnější, nemluvě o tom, že tento přístup umožňuje opravdu důsledně oddělit databázovou vrstvu od aplikační logiky.

Kromě jednoho jsou u všech námi vytvořených datových typů implementovány i funkce, zajišťující ukládání dat těchto typů do databáze. To je velmi důležité, neboť tyto jednotlivé uživatelsky definované datové typy jsou mezi sebou silně provázány a ukládání jejich dat pouze za pomoci SQL příkazů by bylo velmi komplikované a uživatelsky nepřívětivé.

5.2 Návrh struktury databáze

Struktura databáze je tvořena pěti uživatelsky definovanými datovými typy, které pokrývají požadavky na ukládání pohybujících se bodů a na reprezentaci jejich pohybu pomocí trajektorií. Budou dále popsány v následujících kapitolách. Kromě nich jsou vytvořeny další tři typy, které slouží jako struktury pro předávání výsledků z volaných procedur a funkcí.

První dva typy nazvané *IDARRAY1* a *IDARRAY2* jsou pole hodnot typu NUMBER, které zpravidla obsahují identifikátory prostorových entit (bodů a úseček). Na místo těchto dvou typů by postačil pouze jeden, avšak tento přístup byl vybrán z důvodu snadné rozlišitelnosti, kdy do prostorové operace vstupují dva pohybující se objekty a výsledky operace jsou příslušného typu v závislosti na tom, ze kterého objektu jsou vyvozeny.

Třetí typ, pojmenovaný *GEOMARRAY*, je pole objektů typu SDO_GEOMETRY, obsahující nově vytvořené prostorové entity, např. body vzniklé jako průsečíky úseček tvořících trajektorie.

Protože je pro ukládání dat použit objektově-relační model dat, jsou vazby mezi uživatelsky definovanými datovými typy skutečně ukládány jako reference na jiné datové typy v atributech typu, který se na ně odkazuje. Vazby tedy nevznikají až v okamžiku dotazu, jak je tomu u relačního modelu dat, ale jsou explicitně uloženy.

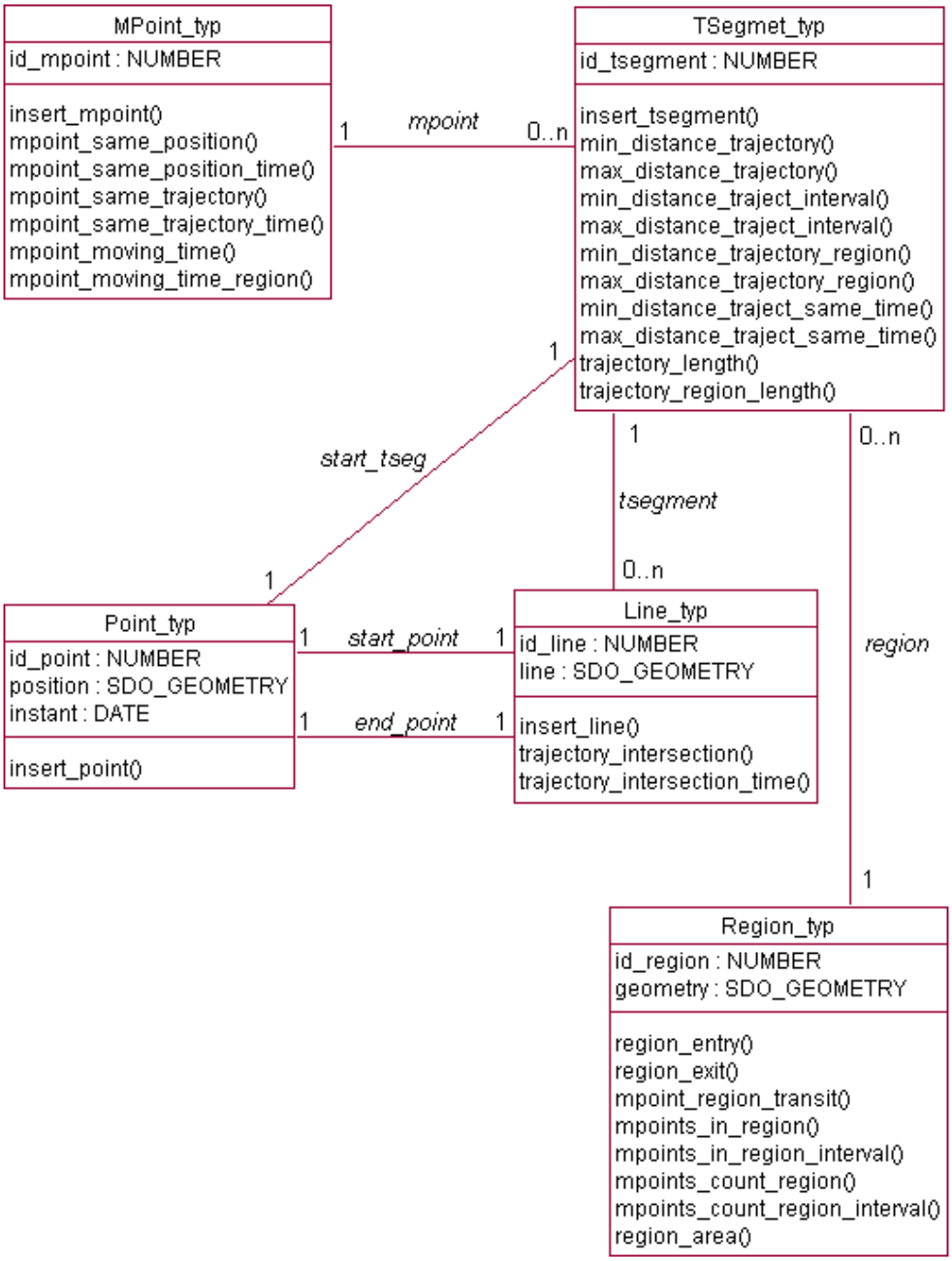
5.2.1 Diagram tříd

Pomocí konceptuálního diagramu tříd je na obrázku 5.1 znázorněna struktura uživatelsky definovaných datových typů včetně pojmenovaných vazeb mezi nimi. U typů nejsou uvedeny signatury operací a to z důvodu přehlednosti.

Z diagramu tříd je vidět velká provázanost mezi jednotlivými uživatelsky definovanými typy, která je dána potřebami dané domény použití, kde každá trajektorie, náležející pohybujícímu se objektu, je tvořena ze segmentů, které se nacházejí v námi snímaných prostorech. Každý segment trajektorie se dále skládá z úseček, které jsou vymezeny koncovými body.

Typ *MPoint_typ* představuje pohybující se objekt, v našem případě bod, jehož trajektorie se skládá z kolekce segmentů, které jsou reprezentovány typem *TSegment_typ*. Každý segment se nachází v právě jednom prostoru, který je reprezentován typem *Region_typ*. Dále každý segment sestává buď z počátečního bodu nebo kromě něj i z kolekce úseček, které jsou reprezentovány typem *Line_typ*. Každá úsečka je spojena se dvěma body, počátečním a koncovým, které jsou reprezentovány typem *Point_typ* a kromě informace o poloze obsahují také časovou informaci.

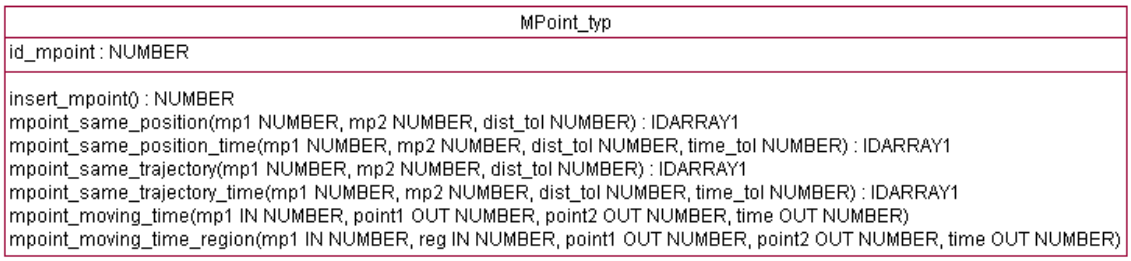
Přes systém vazeb mezi jednotlivými typy lze jednoduše dohledat, které body vytvářejí trajektorii pohybujícího se bodu, ve kterém námi snímaném prostoru se nacházejí, lze zjistit počáteční a koncový čas pohybu objektu a to jak u celé trajektorie, tak i u jejích částí tvořených segmenty atd.



Obrázek 5.1 Konceptuální diagram tříd

5.2.2 Typ MPoint_typ

Tímto uživatelsky definovaným datovým typem, jehož struktura je zobrazena na obrázku 5.2, je reprezentován pohybující se objekt. Jeho jediným atributem je jeho identifikátor. V budoucnu je možné typ obohatit o další vlastnosti podle toho, co bude daným typem reprezentováno, bude-li se např. jednat o auto, dalšími vlastnostmi mohou být SPZ, výrobce, jméno majitele atd.



Obrázek 5.2 Struktura typu MPoint_typ

Uživatelsky definovaný datový typ se vytváří příkazem *CREATE TYPE*, ve kterém se specifikuje jméno typu, jeho atributy a případně signatury procedur a funkcí. V našem případě typ obsahuje pět funkcí a dvě procedury, jejichž tělo se definuje v příkazu *CREATE TYPE BODY*. Procedury a funkce byly implementovány jako statické a to z důvodu jejich snadného volání a použití v jiných procedurách a funkcích. Následuje popis procedur a funkcí:

- FUNCTION insert_mpoint RETURN NUMBER

  Provede vložení nového pohybujícího se bodu do databáze a vrátí jeho identifikátor.
- FUNCTION mpoint_same_position(mp1 NUMBER, mp2 NUMBER, dist_tol NUMBER) RETURN IDARRAY1

  Zjistí, zda se dva pohybující se objekty nacházely na stejném místě. Nastavováním tolerance vzdálenosti lze určovat jemnost/hrubost výsledků, tj. do jaké vzdálenosti jsou dvě pozice považovány ještě za stejné. Funkce vrací identifikátory bodů z trajektorie objektu zadaného v prvním parametru, které určují pozice, na kterých se pohybující se objekty setkaly. Funkce je variací operace locations z tabulky 2.5 a operace distance z kapitoly 2.3.5.
- FUNCTION mpoint_same_position_time(mp1 NUMBER, mp2 NUMBER, dist_tol NUMBER, time_tol NUMBER) RETURN IDARRAY1

  Obdoba předchozí funkce zjistí, zda se dva objekty nacházely na stejném místě ve stejný čas. Tolerance času se zadává v sekundách. Funkce je variací operace atperiods z tabulky 2.6, operace locations z tabulky 2.5 a operace distance z kapitoly 2.3.5.
- FUNCTION mpoint_same_trajectory(mp1 NUMBER, mp2 NUMBER, dist_tol NUMBER) RETURN IDARRAY1

  Funkce pro určení, zda se dva objekty pohybovaly stejnou cestou. Porovnává, zda se počáteční a koncový bod úseček, tvořících trajektorii objektu zadaného v prvním parametru *mp1*, nachází do vzdálenosti, která je určená tolerací *dist_tol,* od úseček, tvořících trajektorii druhého objektu *mp2*. Zvolené řešení je přesnější, nežli by bylo porovnávání vzdálenosti pouze mezi úsečkami navzájem, neboť v námi zvoleném řešení není možné, aby jeden z

konců úsečky nesplňoval danou podmínku vzdálenosti. Funkce vrací identifikátory úseček, tvořící trajektorii objektu, zadaného v druhém parametru. Pro získání společných drah obou objektů je nutné funkci zavolat dvakrát s prohozenými parametry. Protože se porovnává vzdálenost bodů od úseček, může se stát, že funkce vyhodnotí podmínku vzdálenosti v jednom případě jako splněnou, zatímco při druhém volání funkce s prohozenými parametry již podmínka splněna nebude. Jedná se o krajní případ, který by neměl nastat, jestliže se objekty skutečně pohybovaly stejnou trasou, avšak vyplývá z použité metodiky. Funkce je variací operace locations a trajectory z tabulky 2.5 a operace distance z kapitoly 2.3.5.

- FUNCTION mpoint_same_trajectory_time(mp1 NUMBER, mp2 NUMBER, dist_tol NUMBER, time_tol NUMBER) RETURN IDARRAY1

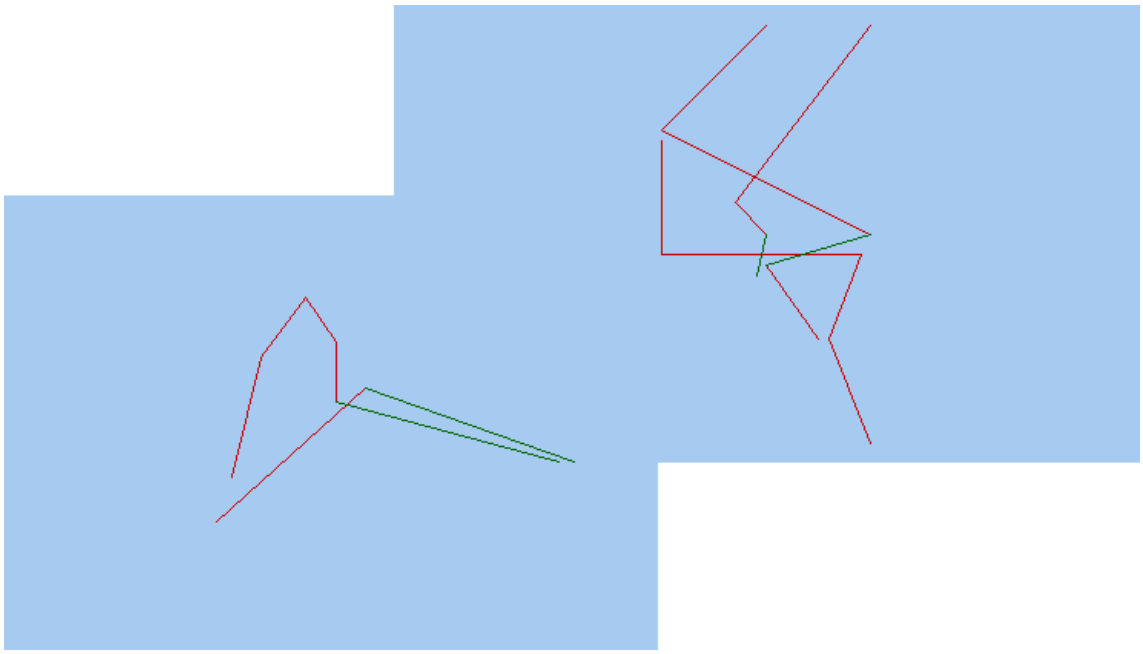
  Obdoba předchozí funkce určuje, zda se dva objekty pohybovaly stejnou cestou společně, tj. ve stejný čas. Jestliže se pohybovaly společně, potom jejich pohyb byl zachycen snímačem v přibližně stejný čas. Z tohoto předpokladu plyne, že počáteční a koncové body úseček obou objektů by se měly nacházet v určitém časovém intervalu, který se zadává v sekundách pomocí parametru *time_tol* a určuje tak toleranci času. Na obrázku 5.3 jsou zeleně znázorněny společné trasy objektů. Funkce je variací operací atperiods, locations, trajectory a distance.

- PROCEDURE mpoint_moving_time(mp1 IN NUMBER, point1 OUT NUMBER, point2 OUT NUMBER, time OUT NUMBER)

  Procedura slouží pro vypočítání celkové doby pohybu daného objektu a kromě ní, ve výstupních parametrech, vrací i identifikátor počátečního a koncového bodu trajektorie. Procedura odpovídá operaci deftime z tabulky 2.5.

- PROCEDURE mpoint_moving_time_region(mp1 IN NUMBER, reg IN NUMBER, point1 OUT NUMBER, point2 OUT NUMBER, time OUT NUMBER)

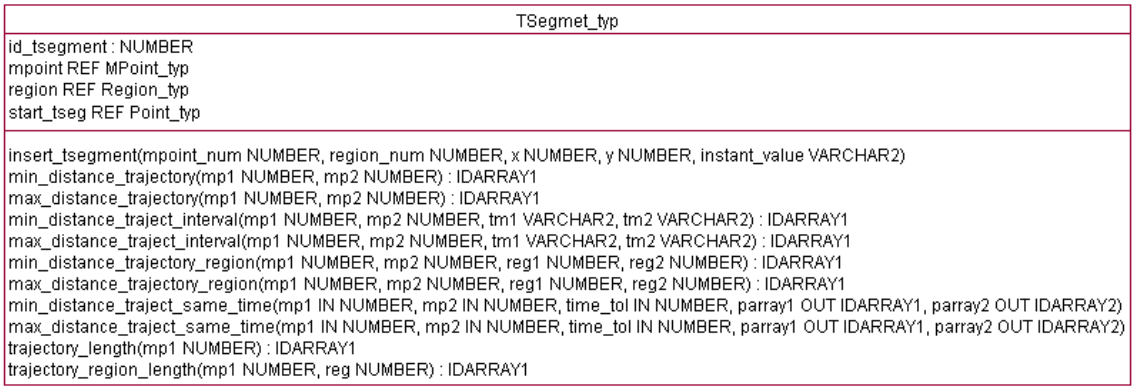
  Obdobná procedura pro výpočet doby pohybu objektu ve snímaném prostoru, který je zadán parametrem *reg*. Procedura kombinuje operaci at z tabulky 2.6 s operací deftime z tab. 2.5.



Obrázek 5.3 Společné dráhy objektů

5.2.3 Typ TSegment_typ

Tento typ reprezentuje části trajektorie, nazývané segmenty. Při každém vstupu objektu do snímaného prostoru se vytváří nový segment, u kterého se zároveň ukládá odkaz na počáteční bod, reprezentovaný typem Point_typ. Segment je tedy vždy vztažený k právě jednomu danému prostoru a vždy přináleží právě jednomu objektu typu MPoint_typ. Uspořádaná kolekce segmentů poté vytváří celkovou trajektorii pohybujícího se objektu.



Obrázek 5.4 Struktura typu TSegment_typ

Na obrázku 5.4 je znázorněna struktura typu se signaturami procedur a funkcí a mezi atributy jsou uvedeny i atributy, obsahující odkazy na objekty jiných typů. Typ obsahuje jedenáct níže uvedených procedur a funkcí:

- PROCEDURE insert_tsegment(mpoint_num NUMBER, region_num NUMBER, x NUMBER, y NUMBER, instant_value VARCHAR2)

  Procedura pro vložení nového segmentu trajektorie do databáze. Současně se ukládá i počáteční bod segmentu, který je typu Point_typ a vytváří se na něj reference, jež je uložena do atributu *start_tseg*.
- FUNCTION min_distance_trajectory(mp1 NUMBER, mp2 NUMBER) RETURN IDARRAY1

  Funkce pro nalezení bodu z trajektorie prvního objektu, který se nachází nejblíže trajektorii druhého objektu. Princip metody spočívá v měření vzdáleností mezi body a úsečkami. Hledá se minimální vzdálenost mezi trajektoriemi, tj. trajektorie se mohou skládat i z více segmentů a její hodnota je posléze vypsána na výstup databáze DBMS_OUTPUT. Funkce kombinuje operace locations, trajectory a distance.
- FUNCTION max_distance_trajectory(mp1 NUMBER, mp2 NUMBER) RETURN IDARRAY1

  Obdoba předchozí funkce, která hledá maximální vzdálenost mezi trajektoriemi. U tohoto druhu funkcí, které vracejí pouze jednu hodnotu, by byl dostačujícím návratovým typem typ NUMBER, avšak je stále používán typ IDARRAY1 a to z důvodu jednotného způsobu práce s funkcemi.

- FUNCTION min_distance_traject_interval(mp1 NUMBER, mp2 NUMBER, tm1 VARCHAR2, tm2 VARCHAR2) RETURN IDARRAY1

  Funkce pro zjištění minimální vzdálenosti trajektorie objektu *mp1* od trajektorie objektu *mp2* v daném časovém intervalu. Interval je zadán počátečním a koncovým časem *tm1* a *tm2*. Funkce vrací identifikátor nejbližšího bodu z trajektorie prvniho objektu, jehož čas leží v zadaném intervalu. Název PL/SQL procedur a funkcí může mít nejvýše třicet znaků, proto byl název této funkce zkrácen. Funkce kombinuje operace atperiods, locations, trajectory a distance.

- FUNCTION max_distance_traject_interval(mp1 NUMBER, mp2 NUMBER, tm1 VARCHAR2, tm2 VARCHAR2) RETURN IDARRAY1

  Obdoba předchozí funkce, která hledá maximální vzdálenost mezi trajektoriemi v zadaném časovém intervalu.

- FUNCTION min_distance_trajectory_region(mp1 NUMBER, mp2 NUMBER, reg1 NUMBER, reg2 NUMBER) RETURN IDARRAY1

  Funkce hledá minimální vzdálenost mezi segmenty trajektorií dvou pohybujících se objektů, které se nacházejí ve snímaných prostorech zadaných parametry *reg1* a *reg2*. Funkce kombinuje operace at, locations, trajectory a distance.

- FUNCTION max_distance_trajectory_region(mp1 NUMBER, mp2 NUMBER, reg1 NUMBER, reg2 NUMBER) RETURN IDARRAY1

  Obdobná funkce hledající maximální vzdálenost mezi segmenty ve snímaných prostorech.

- PROCEDURE min_distance_traject_same_time(mp1 IN NUMBER, mp2 IN NUMBER, time_tol IN NUMBER, parray1 OUT IDARRAY1, parray2 OUT IDARRAY2)

  Tato procedura slouží pro určení, kdy a kde si dva pohybující se objekty byly ve stejný čas vzájemně nejblíže. Oproti předešlým funkcím se liší tím, že časový rozdíl výskytu bodů na trajektoriích, které se porovnávají, musí být v toleranci zadané parametrem *time_tol*. Identifikátory bodů z obou trajektorií, které jsou si nejblíže, jsou vraceny výstupními parametry *parray1* a *parray2* a hodnota vzdálenosti je vypsána na výstup databáze DBMS_OUTPUT. Procedura je variací operací atperiods, locations, trajectory a distance.

- PROCEDURE max_distance_traject_same_time(mp1 IN NUMBER, mp2 IN NUMBER, time_tol IN NUMBER, parray1 OUT IDARRAY1, parray2 OUT IDARRAY2)

  Obdoba předchozí procedury, která hledá kdy a kde si dva pohybující se objekty byly ve stejný čas nejdále.

- FUNCTION trajectory_length(mp1 NUMBER) RETURN IDARRAY1

  Funkce slouží pro vypočítání celkové délky trajektorie. Neřeší problematiku překrývajících se snímaných prostorů, kdy délka segmentů z těchto prostorů by mohla být započítána vícekrát. Záleží na doméně použití, zdali je tato vlastnost problematická či nikoliv, existují prostory, které se ze své podstaty nemohou překrývat (hranice států), ale i takové, u kterých je to zcela běžné (křižovatka silnic).

- FUNCTION trajectory_region_length(mp1 NUMBER, reg NUMBER) RETURN IDARRAY1

  Funkce pro výpočet délky segmentu trajektorie v zadaném snímaném prostoru.

5.2.4 Typ Line_typ

Tímto typem jsou reprezentovány úsečky, vytvářející segmenty trajektorie. Každá úsečka náleží k právě jednomu segmentu a dále se odkazuje na dva body, které ohraničují její začátek a konec. Na obrázku 5.5 je znázorněna struktura typu, kde můžeme vidět, že jeden z jejích atributů je typu SDO_GEOMETRY a slouží pro uchovávání prostorové informace o úsečce. Tato informace by mohla být odvozena i z počátečního a koncového bodu, avšak kvůli efektivitě prostorových operací byla zvolena tato varianta, obsahující mírnou redundanci informací.

Line_typ
id_line : NUMBER line : SDO_GEOMETRY tsegment REF TSegment_typ start_point REF Point_typ end_point REF Point_typ
insert_line(mpoint_num NUMBER, region_num NUMBER, x NUMBER, y NUMBER, instant_value VARCHAR2) trajectory_intersection(mp1 IN NUMBER, mp2 IN NUMBER, larray1 OUT IDARRAY1, larray2 OUT IDARRAY2, iarray OUT GEOMARRAY) trajectory_intersection_time(mp1 IN NUMBER, mp2 IN NUMBER, time_tol IN NUMBER, larray1 OUT IDARRAY1, larray2 OUT IDARRAY2, iarray OUT GEOMARRAY)

Obrázek 5.5 Struktura typu Line_typ

Typ Line_typ obsahuje pouze tři níže popsané procedury:

- PROCEDURE insert_line(mpoint_num NUMBER, region_num NUMBER, x NUMBER, y NUMBER, instant_value VARCHAR2)

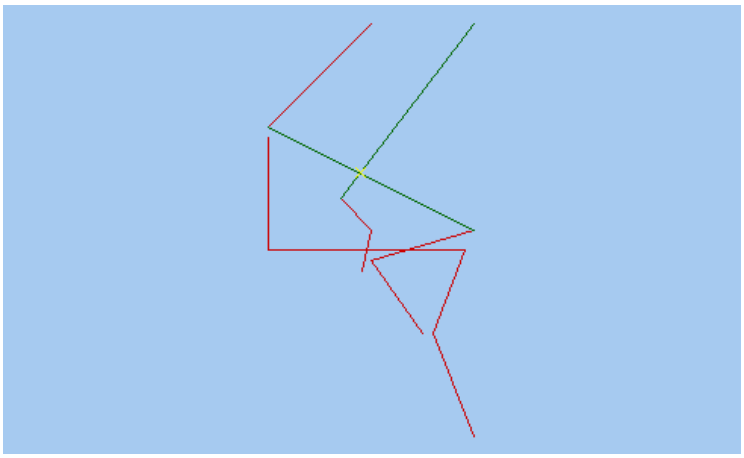
  Procedura pro uložení nové úsečky do databáze. Současně s ukládáním úsečky se ukládá i bod, jehož souřadnice a čas jsou zadané v parametrech procedury, tento bod vždy slouží jako koncový bod úsečky. Vkládá-li se úsečka, která je první úsečkou segmentu trajektorie, poté jako její počáteční bod, je vzat bod, který je odkazován atributem *start_tseg* typu TSegment_typ, tj. jako počáteční bod úsečky se bere počáteční bod segmentu. V ostatních případech se jako počáteční bod úsečky bere koncový bod předešlé úsečky.

- PROCEDURE trajectory_intersection(mp1 IN NUMBER, mp2 IN NUMBER, larray1 OUT IDARRAY1, larray2 OUT IDARRAY2, iarray OUT GEOMARRAY)

  Procedura pro nalezení průsečíků trajektorií dvou pohybujících se objektů. Jestliže se dvě úsečky kříží, poté jsou jejich identifikátory vráceny ve výstupních parametrech *larray1* a *larray2*. Současně je vytvořen nový objekt typu SDO_GEOMETRY, který obsahuje souřadnice průsečíku a pole těchto objektů je vraceno ve výstupním parametru *iarray*. Procedura kombinuje operaci trajectory z tabulky 2.5 s operací crossings z tabulky 2.4.

- PROCEDURE trajectory_intersection_time(mp1 IN NUMBER, mp2 IN NUMBER, time_tol IN NUMBER, larray1 OUT IDARRAY1, larray2 OUT IDARRAY2, iarray OUT GEOMARRAY)

  Procedura pro zjištění, zdali se trajektorie dvou pohybujících se objektů křížily ve stejný čas. Je-li tomu tak, můžeme předpokládat, že jsou objekty snímány stejným senzorem, který v přibližně stejný čas zachycuje polohu všech objektů. Proto pro určení, zdali se dvě úsečky kříží ve stejný čas, můžeme využít rozdílů časů dvou počátečních a dvou koncových bodů úseček, které by měly spadat do tolerance zadané parametrem *time_tol*. Opět se vytváří nový objekt typu SDO_GEOMETRY, který uchovává souřadnice průsečíku. Procedura kombinuje operace atperiods, trajectory a crossings. Na obrázku 5.6 můžeme vidět grafické znázornění křížících se trajektorií, vytvořené na základě výstupu této procedury.

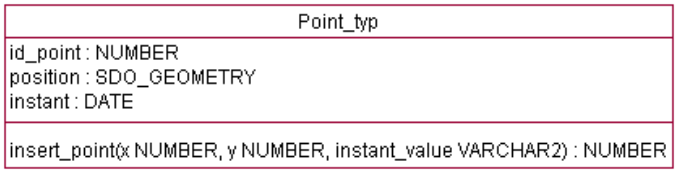


Obrázek 5.6 Průsečík trajektorií

5.2.5 Typ Point_typ

Uživatelsky definovaný datový typ Point_typ reprezentuje jednotlivé body trajektorie, tj. výskyty pohybujícího se objektu v čase. Obsahuje atribut typu SDO_GEOMETRY, který uchovává prostorovou informaci, avšak čas není přidán jako třetí dimenze objektu typu SDO_GEOMETRY, ale jako samostatný atribut typu Point_typ.

Na venek tedy typ vytváří trojdimenzionální reprezentaci časoprostorových dat, uvnitř však obsahuje dvojdimenzionální prostorovou a jednodimenzionální časovou informaci. Tím je umožněno využívat prostorové operace SŘBD, které jsou použitelné na nejvýše dvojdimenzionálních prostorových objektech. Další výhodou je možnost ukládat absolutní čas namísto pouze relativního, jak by tomu muselo být v případě, kdy by čas tvořil třetí dimenzi objektu typu SDO_GEOMETRY.

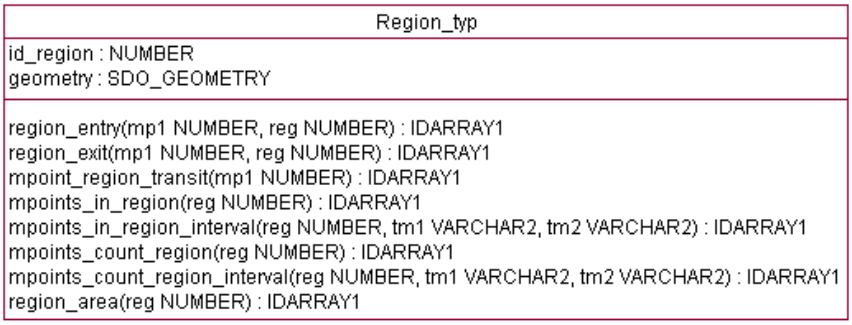


Obrázek 5.7 Struktura typu Point_typ

Na obrázku 5.7 je znázorněna struktura typu Point_typ, který obsahuje pouze jednu funkci `FUNCTION insert_point(x NUMBER, y NUMBER, instant_value VARCHAR2) RETURN NUMBER`, jež slouží pro ukládání dat tohoto typu do databáze a vrací identifikátor právě uloženého objektu.

5.2.6 Typ Region_typ

Objekt typu Region_typ v sobě zahrnuje prostorovou informaci o snímaném prostoru, ve kterém se nacházejí pohybující se objekty. V databázi bude uložen poměrně malý počet objektů toho typu, které se v čase nemění. Z tohoto a také z důvodu nemožnosti předvídat tvary snímaných prostorů, není implementována funkce, která by zajišťovala ukládání dat tohoto typu do databáze. Společně s fyzickou instalací snímače se do databáze uloží informace o tvaru snímaného prostoru pomocí příkazu SQL, který je poměrně jednoduchý, neboť typ neobsahuje žádné reference na objekty jiných typů, jak můžeme vidět z obrázku 5.8.



Obrázek 5.8 Struktura typu Region_typ

Typ Region_typ obsahuje níže uvedené funkce:

- FUNCTION region_entry(mp1 NUMBER, reg NUMBER) RETURN IDARRAY1

  Funkce vrací identifikátor bodu, který je vstupním bodem pohybujícího se objektu do snímaného prostoru. Na výstup databáze DBMS_OUTPUT jsou vypsány souřadnice bodu a čas. Funkce odpovídá operaci initial z tabulky 2.6.
- FUNCTION region_exit(mp1 NUMBER, reg NUMBER) RETURN IDARRAY1

  Obdobná funkce, která vrací výstupní bod objektu ze snímaného prostoru. Odpovídá operaci final z tabulky 2.6.
- FUNCTION mpoint_region_transit(mp1 NUMBER) RETURN IDARRAY1

  Funkce vrací identifikátory prostorů, kterými objekt prošel. Není udělána zvlášť funkce, která by vracela identifikátory prostorů, kterými objekt prošel v zadaném časovém intervalu, neboť tuto informaci lze získat pomocí funkcí, které vrací čas vstupu a výstupu objektů z daného snímaného prostoru. Funkce je variací operace passes z tabulky 2.6.
- FUNCTION mpoints_in_region(reg NUMBER) RETURN IDARRAY1

  Funkce vrací identifikátory objektů, které se vyskytovaly v zadaném prostoru. Je variací operace locations z tabulky 2.5 a operace intersection z tabulky 2.4.
- FUNCTION mpoints_in_region_interval(reg NUMBER, tm1 VARCHAR2, tm2 VARCHAR2) RETURN IDARRAY1

  Funkce vrací identifikátory objektů, které se vyskytovaly v zadaném prostoru v rozmezí časů zadaných parametry *tm1* a *tm2*. Je variací operací atperiods, locations a intersection.

- FUNCTION mpoints_count_region(reg NUMBER) RETURN IDARRAY1

  Funkce vrací počet objektů, které se vyskytovaly v daném prostoru. Kombinuje operaci at z tabulky 2.6 s operací count z kapitoly 2.3.4.

- FUNCTION mpoints_count_region_interval(reg NUMBER, tm1 VARCHAR2, tm2 VARCHAR2) RETURN IDARRAY1

  Funkce vrací počet objektů, které se vyskytovaly v daném prostoru během časového intervalu zadaného parametry *tm1* a *tm2*. Kombinuje operace atperiods, at a count.

- FUNCTION region_area(reg NUMBER) RETURN IDARRAY1

  Funkce pro výpočet velikosti plochy regionu.

Kapitola 6

Ukázková aplikace MOD

Smyslem ukázkové aplikace MOD (Moving Objects Database) je grafické znázornění uložených dat a demonstrace funkcionality implementovaných prostorových operací. Je to v podstatě jednoduchá SQL konzole pro časoprostorová data, která umožňuje provádět dotazy, jejichž výsledky jsou typu SDO_GEOMETRY. Vyžaduje proto znalost SQL a znalost schématu námi navržené databáze. Ukázková aplikace je postavena na technologii *J2EE* a prostředím pro běh je *OC4J* kontejner. Generování grafického výstupu z prostorových dat zajišťuje aplikace *Oracle MapViewer*.

6.1 J2EE

Technologie J2EE je soubor specifikací, který definuje standardy pro tvorbu vícevrstvých aplikací. Aplikace jsou typicky třívrstvé, rozdělené na klientskou, aplikační a databázovou vrstvu. Klientskou vrstvu tvoří webové stránky, aplikační vrstvu tvoří aplikační server, ve kterém běží webové komponenty, vytvořené technologií *JavaServer Pages* a aplikační komponenty, vytvořené technologií *Enterprise JavaBeans*.

J2EE komponenty jsou napsány v jazyce Java a jsou kompilovány stejným způsobem jako běžné Java aplikace. Rozdíl mezi komponentami a běžnými třídami jazyka Java spočívá v tom, že komponenty, tvořící J2EE aplikaci, jsou vytvořeny ve shodě s J2EE specifikací a jsou spouštěny a spravovány aplikačním serverem.

Databázovou vrstvu tvoří databázový server. Námi navrhnutá databáze je použitelná v databázovém serveru Oracle 10g, běžícím na školním serveru pcuifs1, a na serveru Oracle 11g, běžícím na školních serverech berta a minerva2. Databáze je přístupná z aplikací skrze *JDBC* ovladač.

JDBC je aplikační rozhraní Javy, které umožňuje připojení k relační databázi a nezávislost aplikací na použitém databázovém serveru. Pro připojení je nutné vytvořit databázové URL, které identifikuje konkrétní ovladač a databázi. Obecný formát adresy URL je `jdbc:<podprotokol>:<identifikátor_databáze>`, pro naši databázi běžící na serveru minerva2 např. `jdbc:oracle:thin:@minerva2.fit.vutbr.cz:1521:ORACLE`.

6.1.1 OC4J

OC4J kontejner je jádrem aplikačního serveru Oracle, které slouží jako běhové prostředí pro webové a aplikační komponenty. Aplikační server rozšiřuje možnosti samotného kontejneru o podporu transakcí, rozložení zátěže, autorizaci uživatelů, bezpečnost, škálovatelnost atd., avšak pro naši aplikaci postačí samotný OC4J kontejner.

Při volání námi vytvořené databázové procedury, vracející výsledek typu GEOMARRAY, docházelo v Java aplikaci k výjimce `java.sql.SQLException:Bigger type length than Maximum`, která se ukázala být způsobená chybou JDBC ovladače, jež je dodáván společně s OC4J kontejnerem, ve kterém běží aplikace MapViewer a naše ukázková aplikace. Námi použitý OC4J kontejner verze 10.1.2.0.2 obsahuje JDBC ovladače classes12.jar a ojdbc14.jar verze 10.1.0.2.0. Ovladač classes12.jar je používán, běží-li OC4J na JDK verze 1.2 a 1.3, ovladač ojdbc14.jar je používán s JDK verze 1.4 a 1.5. Tyto ovladače je nutné nahradit ovladači verze 10.2.0.3.0 a vyšší, které již pracují bez problému.

6.2 Oracle MapViewer

Oracle Application Server MapViewer [9] je nástroj pro vizualizaci prostorových dat, který běží uvnitř J2EE kontejneru. Přijímá požadavky skrze HTTP protokol, po jejich zpracování vygeneruje mapu, kterou vrací jako odpověď. Požadavky a odpovědi jsou ve formě XML, což umožňuje nezávislost na platformě a použitém programovacím jazyku. Použití XML však může být těžkopádné, proto MapViewer poskytuje Java API, které umožňuje vytvářet mapy na základě požadavků z Java aplikací, servletů nebo JSP.

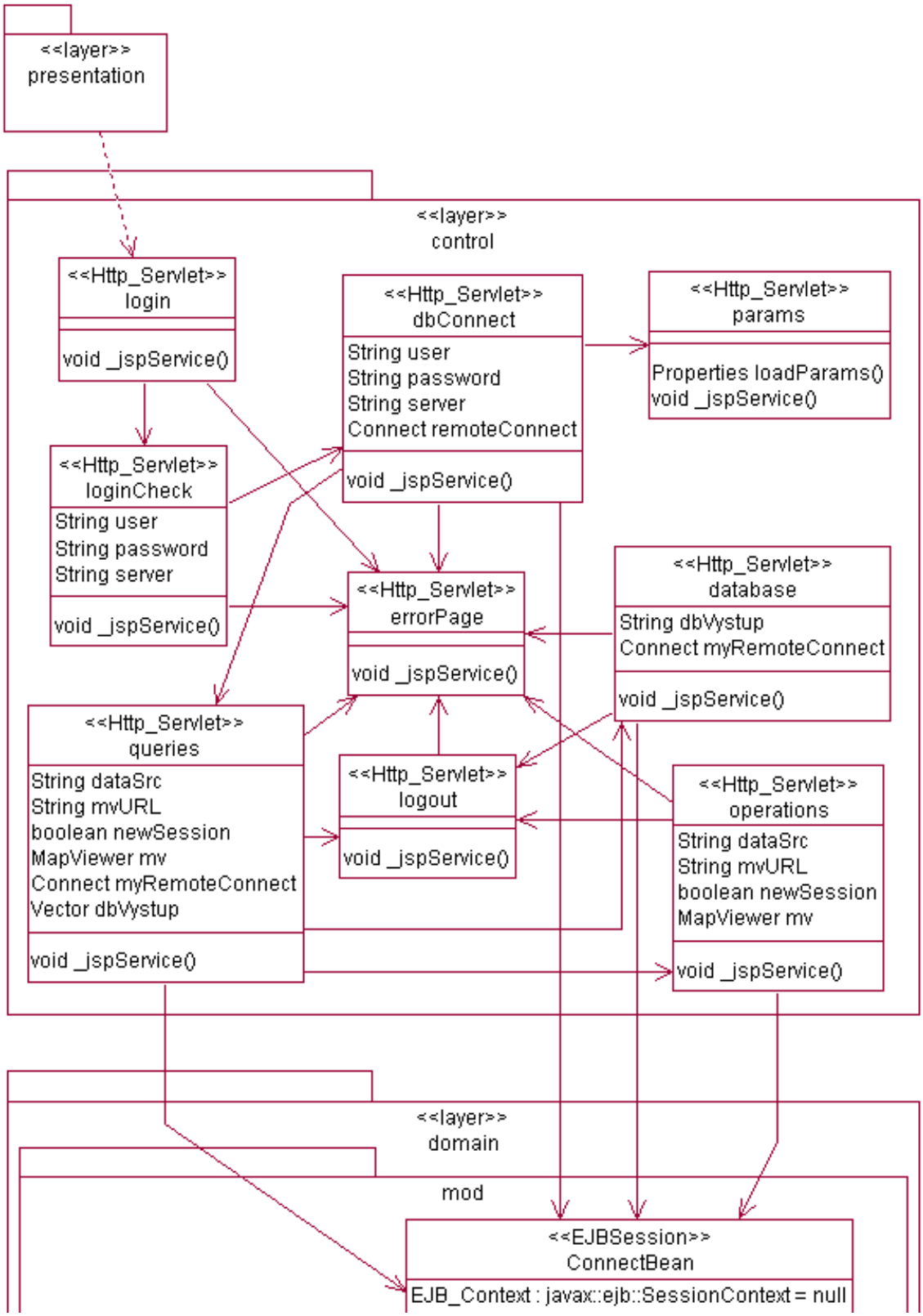
Pro možnost používat metody MapVieweru z Java aplikací je nutné do CLASSPATH přidat umístění souboru `mvclient.jar`, který implementuje API rozhraní. V API aplikace MapViewer je uvedena metoda `addDataSource(String name, String host, int port, String sid, String user, String password, String mode, int nmapper)`, která slouží pro zadání zdroje dat, avšak z bezpečnostních důvodů již není možné ji používat.

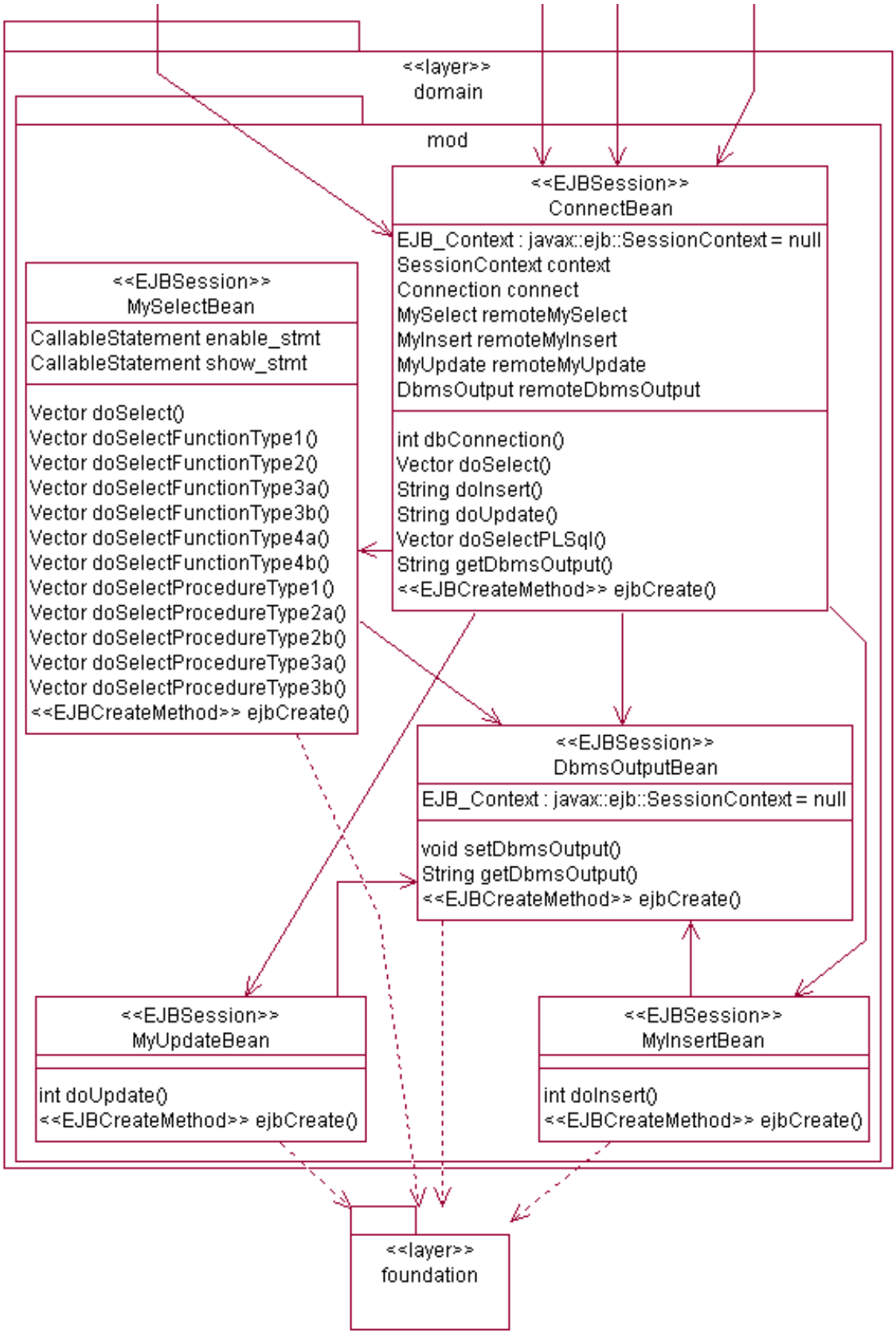
Pro zadání zdroje dat je nutné použít administrátorské rozhraní aplikace MapViewer a posléze podle jména se na zdroj odkazovat. Zdroj dat může být trvale nastaven pomocí konfiguračního XML souboru aplikace MapViewer, který je při startu automaticky načten. Je nutné mít na paměti, že při uvádění hesla v konfiguračním souboru je nutné před něj uvést vykřičník. Bez neuvedeného vykřičníku není možné se k databázi připojit a není lehké tuto chybu odhalit. Následuje příklad nastavení zdroje dat v konfiguračním souboru:

```
<map_data_source name="orcl"
                    jdbc_host="minerva2.fit.vutbr.cz"
                    jdbc_sid="ORACLE"
                    jdbc_port="1521"
                    jdbc_user="jmeno"
                    jdbc_password="!heslo"
                    jdbc_mode="thin"
                    number_of_mappers="3"
         />
```

6.3 Ukázková aplikace

Na následujících dvou stranách je na obrázku 6.1 znázorněn diagram tříd ukázkové aplikace.


<<layer>>
presentation
<<layer>>
control
<<Http_Servlet>>
login
void _jspService()
<<Http_Servlet>>
dbConnect
String user
String password
String server
Connect remoteConnect
void _jspService()
<<Http_Servlet>>
params
Properties loadParams()
void _jspService()
<<Http_Servlet>>
loginCheck
String user
String password
String server
void _jspService()
<<Http_Servlet>>
errorPage
void _jspService()
<<Http_Servlet>>
database
String dbVystup
Connect myRemoteConnect
void _jspService()
<<Http_Servlet>>
queries
String dataSrc
String mvURL
boolean newSession
MapViewer mv
Connect myRemoteConnect
Vector dbVystup
void _jspService()
<<Http_Servlet>>
logout
void _jspService()
<<Http_Servlet>>
operations
String dataSrc
String mvURL
boolean newSession
MapViewer mv
void _jspService()
<<layer>>
domain
mod
<<EJBSession>>
ConnectBean
EJB_Context : javax::ejb::SessionContext = null

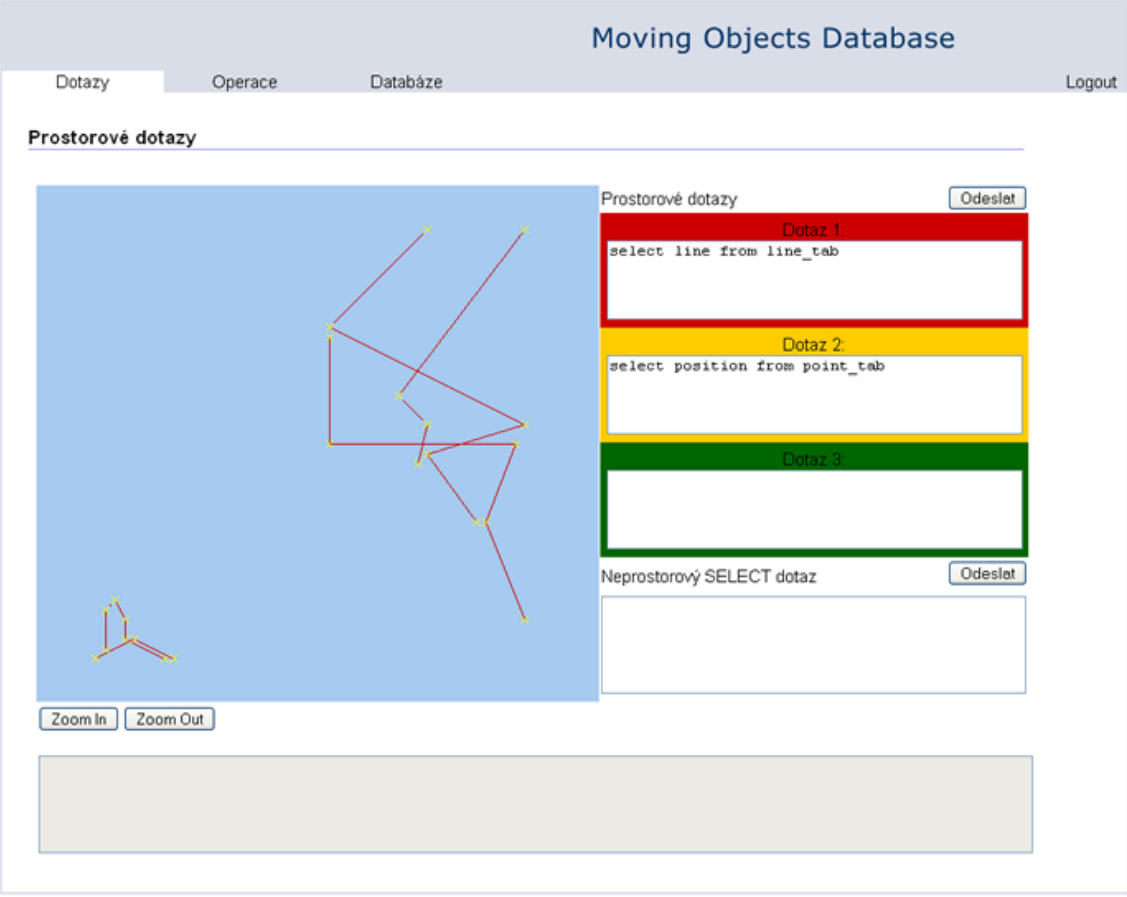


Obrázek 6.1 Diagram tříd

Z diagramu tříd je vidět, že ukázková aplikace je rozdělěna do několika vrstev, kde vrstva control příjímá uživatelský vstup a zajišťuje výstup aplikace, je realizována pomocí JSP stránek. Vrstva domain zajišťuje aplikační logiku, příjímá volání z vrstvy control a komunikuje s databázovou vrstvou foundation. Vrstva domain je realizována pomocí Enterprise JavaBeans komponent.

Při přihlašování k ukázkové aplikaci uživatel vybere databázový server, ke kterému se chce připojit. Je nutné, aby tento databázový server byl nastaven i jako zdroj dat v aplikaci Oracle MapViewer výše popsaným způsobem. Nastavení se provádí v administrátorské části aplikace MapViewer.

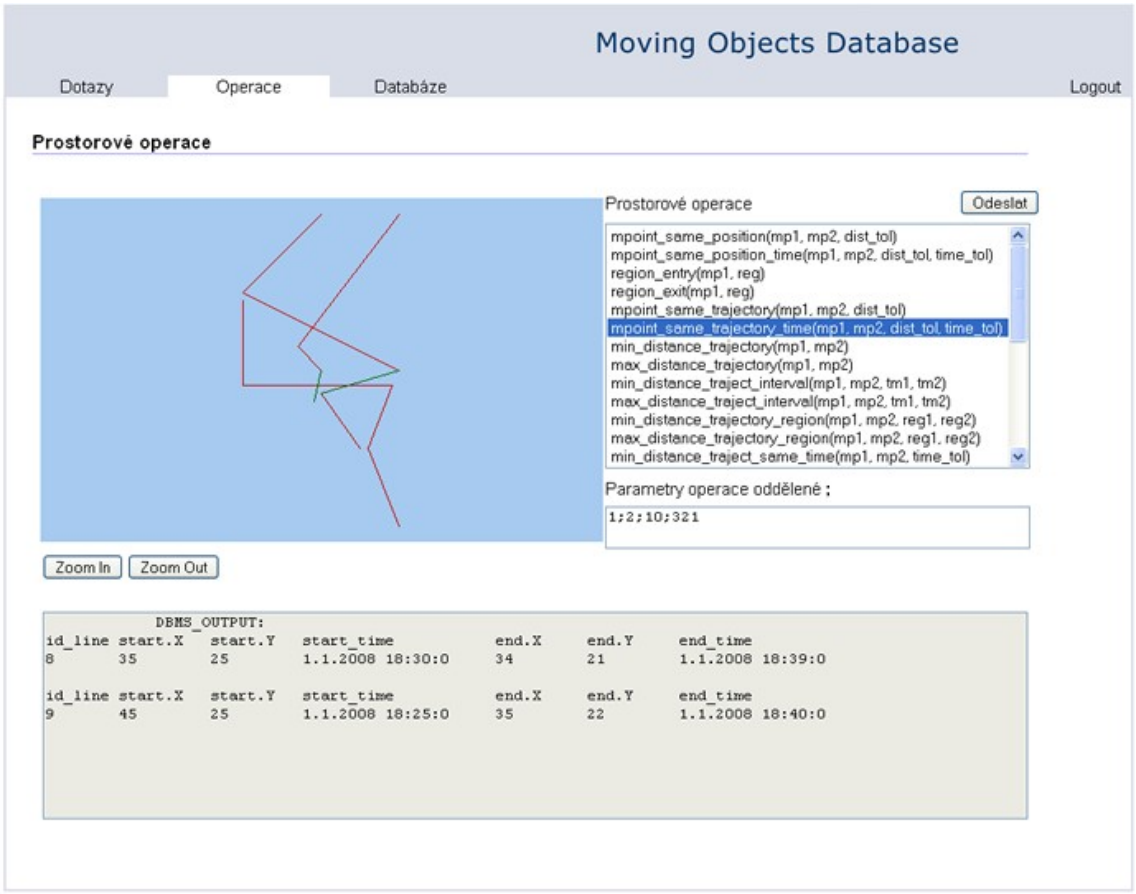
Aby ukázková aplikace mohla komunikovat s aplikací MapViewer a zobrazovat tak grafický výstup, je nutné u aplikace správně nastavit naslouchání na adresu, na které MapViewer zasílá odpovědi ve formě vygenerovaných map. Adresa je ve formátu `http://<server>:<port>/mapviewer/omserver`, např. při spuštění na lokálním počítači je adresa `http://localhost:8888/mapviewer/omserver`.



Obrázek 6.2 Zobrazení prostorových dotazů

Na obrázku 6.2 je znázorněna webová stránka ukázkové aplikace, která umožňuje provádět SQL dotazy. Umožňuje provádět naráz až tři prostorové dotazy, tj. dotazy, jejichž výsledky jsou typu SDO_GEOMETRY, které jsou posléze zpracovány aplikací MapViewer a zobrazeny v příslušných barvách. Dále umožňuje provádět neprostorový SELECT dotaz, jehož výsledek je vypsán ve spodním textovém poli, které však neumožňuje textově vypisovat hodnoty atributů, jež obsahují reference na objekty jiných typů.

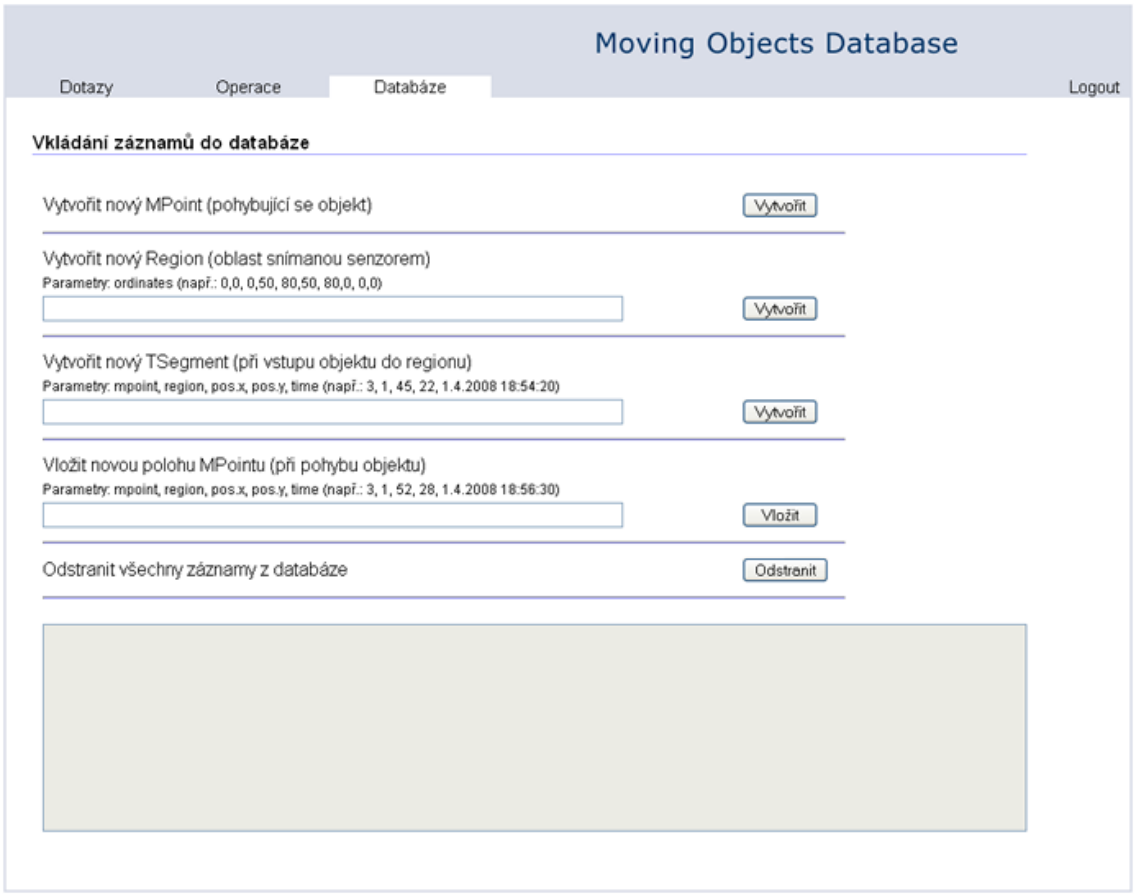
Na obrázku 6.3 je znázorněna webová stránka, umožňující provádět prostorové operace nad uloženými daty. Námi vytvořené databázové procedury a funkce vracejí pole výsledků a současně vypisují text na výstup databáze DBMS_OUTPUT. Pole výsledků typicky obsahuje identifikátory objektů námi vytvořených typů, to umožňuje větší variabilitu než při vracení přímo objetů typu SDO_GEOMETRY. Struktura databáze se může dále rozšiřovat a je možné se z uživatelských aplikací dotazovat na další atributy objektů, identifikovaných pomocí vráceného pole identifikátorů. Současně je také možné používat procedury a funkce v rámci práce s databází, kdy nám poskytují výsledky i v textové formě, doplněné o jejich interpretaci, na výstupu databáze DBMS_OUTPUT.



Obrázek 6.3 Zobrazení prostorových operací

Na obrázku 6.4 je znázorněna webová stránka, umožňující manipulaci s databází a to vkládání nových záznamů a mazání záznamů. Není však možné mazání jednotlivých částí trajektorií a to z důvodu jejich silné provázanosti, také doména použití aplikace nepředpokládá potřebu mazání jednotlivých záznamů, reprezentujících pohybující se objekty.

Vkládání záznamů je implementováno prostřednictvím Enterprise JavaBeans komponent, které poskytují rozhraní pro volání svých veřejných metod i z jiných aplikací. Je tak umožněno, aby i jiné aplikace mohly vkládat záznamy do databáze přes tyto komponenty.



Obrázek 6.4 Stránka pro manipulaci s databází

Na obrázku 6.5 je znázorněn grafický výstup ukázkové aplikace MOD (v různém přiblížení) pro databázi obsahující 400 tisíc záznamů, které byly poskytnuty Jiřím Vetešníkem [10]. Záznamy uložené v databázi představují městskou dopravní síť.

Obrázek 6.5 Městská dopravní síť

Kapitola 7

Zhodnocení výsledků

V některé literatuře bývá námi zvolený přístup k řešení problematiky ukládání a reprezentace pohybujících se objektů označován za naivní a sofistikovaným řešením bývá označováno takové řešení, které pro reprezentaci pohybujících se objektů využívá pouze výchozí pozici a rychlost pohybu objektu. Jedná se ale vždy o pohyb objektů po předem definované množině drah již uložených v databázi, např. městská dopravní síť. V případech, kdy se objekty mohou libovolně pohybovat, není tento přístup k řešení problematiky použitelný.

Námi navržená databáze umožňuje zachytit pohyb objektů na základě ukládání prostorových entit, vytvářejících trajektorii. Každá pozice objektu v čase je reprezentována bodem. Spojením dvou bodů vzniká úsečka, reprezentující dráhu pohybu. Spojováním úseček se vytváří segment trajektorie, který představuje pohyb objektu v daném snímaném prostoru. Spojováním segmentů vzniká trajektorie, reprezentující celkový pohyb objektu.

Ukázková aplikace, vytvořená technologií J2EE, umožňuje základní práci s databází, provádění prostorových operací a grafické zobrazení jejich výsledků. Pro generování map je využita aplikace Oracle MapViewer.

7.1 Pokračování projektu

Možností pokračování tohoto projektu je rozšiřování struktury databáze, tj. obohacování uživatelsky definovaných datových typů. Např. typ Region_typ by kromě snímaných prostorů mohl obsahovat informace o dalších nemovitých objektech v něm se nalézajících, tj. např. o budovách, dopravní síti atd. Stejně tak je možné přidávat atributy do typu, reprezentujícího pohybující se objekt, pro jeho větší specifičnost.

Dále je možné rozšiřovat množinu prostorových operací nad trajektoriemi. Prozatím jsou implementovány operace, které vyhodnocují určitý prostorový vztah mezi trajektoriemi dvou zadaných pohybujících se objektů, např. najdi průsečík drah těchto objektů. Další skupinou operací, které by bylo užitečné vytvořit, jsou operace, které jako parametr mají pouze jeden pohybující se objekt a v prostoru hledají množinu jiných objektů, splňujících zadanou podmínku, tj. např. najdi všechny objekty, jejichž trajektorie leží do určité vzdálenosti od pozice daného objektu v daném čase. K těmto dotazům lze využít již implementované operace, avšak jejich samostatné vytvoření by bylo mnohem efektivnější.

V uživatelské aplikaci by bylo přínosné do vygenerované mapy doplnit další informace v textové podobě nebo ještě lépe, vytvořit interaktivní mapu.

Kapitola 8

Závěr

Tato práce se zabývá problematikou reprezentace pohybujících se objektů a typickými operacemi nad takovými objekty. Seznamuje čtenáře s podporou pro časoprostorová data v prostředí databázového serveru Oracle10g. Jsou představeny dva návrhy struktury databáze pohybujících se objektů. V prvním návrhu je trajektorie reprezentována jako množina bodů v trojdimenzionálním prostoru a ve druhém návrhu je trajektorie reprezentována pomocí síťového grafu, kde každý bod je uzlem, který je spojen hranou se sousedními uzly.

Na základě těchto návrhů je vytvořena struktura databáze, částečně kombinující oba přístupy. Implementovaná databáze je použitelná jak v databázovém serveru Oracle 10g, tak i serveru Oracle 11g. Bylo vytvořeno 26 prostorových operací, které pracují nad daty, uloženými v námi navržené databázi.

Dále byla vytvořena ukázková aplikace, demonstrující funkčnost prostorových operací a tím i navržené databáze. Aplikace je vytvořena jako třívrstvá aplikace pomocí J2EE technologie, jejíž aplikační logika je implementována v J2EE komponentách, běžících v prostředí aplikačního serveru OC4J. V tomto aplikačním serveru společně s ní běží i aplikace Oracle MapViewer, která se stará o generování grafického výstupu z prostorových dat.

Na závěr je provedeno zhodnocení dosažených výsledků a jsou diskutovány tři možné směry dalšího rozvoje projektu.

Literatura

[1] Guting, R. H. et al: *A Foundation for Representing and Querying Moving Objects*. ACM Transactions on Database Systems. Vol. 25, No. 1 March 2000, pp. 1- 42.

[2] Loeckx, J., Ehrich, H.-D., Wolf, M.: *Specification of Abstract Data Types*. Wiley/Teubner computing series. John Wiley and Sons, Inc., New York, NY. 1997.

[3] Guting, R. H.: *An Introduction to Spatial Database Systems*. VLDB J. 3 (Oct.), 357 - 399. 1994.

[4] Oracle Corporation: *Oracle Spatial User's Guide and Reference, 10g Release 2 (10.2)*. Dokument dostupný na URL http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14255.pdf (prosinec 2007)

[5] Brakatsoulas, S., Pfoser, D., Tryfona, N.: *Modeling, Storing and Mining Moving Object Databases*. International Database Engineering and Applications Symposium (IDEAS'04) 1098-8068/04. IEEE Computer Society Press. 2004.

[6] Sharma, J., Herring, J., Ravada, S.: *Oracle Spatial*. Oracle Corporation. May 2001. Dokument dostupný na URL http://www-db.deis.unibo.it/courses/SI2/papers/OracleSpatial.pdf (prosinec 2007)

[7] Li, X., Lin, H.: *A Data Model for Moving Object in Dynamic Road Network*. Dokument dostupný na URL https://umdrive.memphis.edu/xli1/www/index_files/15.pdf (prosinec 2007)

[8] Wolfson, O. et al: *Moving Objects Databases: Issues and Solutions*. Proceedings 10th International Conference on Scientific and Statistical Database Management (SSDBM'98), pp.111-122, Capri, Italy, 1998. Dokument dostupný na URL http://www.cs.berkeley.edu/~franklin/CS286S01/wolfson.pdf (prosinec 2007)

[9] Oracle Corporation: *Oracle Application Server MapViewer User's Guide, 10g Release 2 (10.1.2).* Dokument dostupný na URL http://download-west.oracle.com/docs/cd/B14099_19/web.1012/b14036.pdf (únor 2008)

[10] Vetešník, J.: *Indexování pohybujících se objektů.* Diplomová práce, Brno, FIT VUT v Brně, 2008.

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

[12] Oracle Corporation: *Oracle Database PL/SQL User's Guide and Reference, 10g Release 2 (10.2).* Dokument dostupný na URL http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14261.pdf (únor 2008)

[13] Oracle Corporation: *Oracle Database 10g Developing Spatial Applications Using Oracle Spatial and MapViewer.* Dokument dostupný na URL http://www.oracle.com/technology/products/mapviewer/pdf/mapviewer_spatial_wp.pdf (únor 2008)

[14] Oracle Corporation: *Oracle Spatial Topology and Network Data Models, 10g Release 2 (10.2).* Dokument dostupný na URL http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14256.pdf (únor 2008)

[15] Chaudhri, A. B.: *Succeeding with Object Databases.* John Wiley & Sons, Inc., Canada, 2001.

[16] Groff, J. R., Weinberg, P. N.: *SQL Kompletní průvodce.* CP Books, a.s., Brno, 2005.

[17] Kumar, B. V., Sangeetha, S., Subrahmanya, S. V.: *J2EE Architecture.* Tata McGraw-Hill, New Delhi, 2005.

Seznam použitých zkratek

API	Application Programming Interface
EJB	Enterprise JavaBeans
HTTP	HyperText Transfer Protocol
J2EE	Java 2, Enterprise Edition
JDBC	Java Database Connectivity
JDK	Java Development Kit
JSP	JavaServer Pages
MOD	Moving Objects Database
OC4J	Oracle Application Server Containers for J2EE
SQL	Structured Query Language
SŘBD	Systém Řízení Báze Dat
URL	Uniform Resource Locator
XML	eXtensible Markup Language

Seznam příloh

Příloha 1

Uživatelská příručka

Instalace aplikace

Aplikační server OC4J na přiloženém CD-ROMu má v sobě již umístěny aplikace MOD a Oracle MapViewer, je tedy připraven k okamžitému použití. Spouští se dávkovým souborem `start.bat` nebo `start.sh`, ve kterém je nutné nastavit cestu k adresáři s JDK a cestu k adresáři, ve kterém se nachází OC4J. Pro běh aplikace MOD na serveru je nutné modifikovat soubor `config.properties` v adresáři `/MOD/MOD/classes`, ve kterém je nastavení JNDI pro daný systém. Pro kompilaci zdrojových souborů je nutné do CLASSPATH aplikace MOD přidat knihovnu `mvclient.jar`, která umožňuje volání metod aplikace MapViewer, viz [9].

Přihlášení do systému

Při přihlášení do systému si uživatel zvolí z nabídky databázový server, ke kterému se hodlá připojit. Je nutné aby pro daný databázový server byl uživatel uveden i v konfiguračním souboru aplikace Oracle MapViewer, podrobnosti viz kapitola 6.2. Po přihlášení si uživatel může vybrat z nabídky záložek webových stránek nazvaných Dotazy, Operace a Databáze.

Stránka Dotazy

Obsahuje SQL konzoli, která umožňuje provádět naráz až tři SQL dotazy, jejichž výsledky jsou typu SDO_GEOMETRY a které jsou posléze graficky znázorněny v příslušných barvách v levé části webové stránky. Na stránce se dále nachází textové pole pro provádění běžných SQL dotazů, jejichž výsledek je vypsán ve spodním textovém poli, které má však omezení, že není schopno textově vypisovat obsah atributů typu REF, jež obsahují reference na jiné objekty a atributů typu SDO_GEOMETRY.

Příklady prostorových SQL dotazů:

- Dotaz pro zobrazení všech trajektorií uložených v databázi:
  ```
  select line from line_tab
  ```
- Dotaz pro zobrazení trajektorie pohybujícího se objektu s ID 3:
  ```
  select l.line from line_tab l, tsegment_tab t, mpoint_tab m
  where m.id_mpoint=3 AND t.mpoint=REF(m) AND l.tsegment=REF(t)
  ```
- Dotaz pro zobrazení všech bodů, tvořících trajektorie uložené v databázi:
  ```
  select position from point_tab
  ```

- Dotaz pro zobrazení všech snímaných prostorů uložených v databázi:
  `select geometry from region_tab`

Stránka Operace

Umožňuje volání implementovaných prostorových a časoprostorových operací. Z nabídkového menu se podle jména vybere žádaná operace. Parametry operace se uvádějí v textovém poli, umístěném pod menu s opracemi. Jednotlivé parametry operace je nutné oddělit ";" a pro zadávání parametrů lze použít pouze celá čísla.

Použité parametry operací:

- parametry *mp1* a *mp2* značí ID pohybujících se objektů
- parametr *reg* značí ID snímaného prostoru (regionu)
- parametr *dist_tol* značí toleranci vzdálenosti v jednotkách
- parametr *time_tol* značí časovou toleranci v sekundách
- parametry *tm1* a *tm2* značí časové značky ve formátu DD.MM.YYYY HH:MI:SS

Např. pro operaci `mpoint_same_trajectory_time(mp1, mp2, dist_tol, time_tol)` lze zadat parametry ve tvaru: 1;2;10;321.

Stránka Databáze

Umožňuje vkládání nových záznamů do databáze a vymazání všech záznamů z databáze. Lze vytvořit nový pohybující se objekt (MPoint) a dále nový snímaný prostor (Region), u něhož se zadá pole vrcholů, které definuje daný prostor. Ve spodním textovém poli je posléze vypsáno ID právě vytvořených entit.

Při vstupu objektu do daného prostoru se vytváří nový segment trajektorie (TSegment) a při dalším pohybu objektu v tomto prostoru se ukládají jeho nové pozice. Pro ukládání záznamů se používají parametry oddělené ",". Parametry jsou tvořeny identifikátorem pohybujícího se objektu, identifikátorem regionu, x-ovou a y-ovou pozicí objektu a časem. Příklad zadání parametrů: 3,1,45,22,5.4.2008 18:22:25. Pro zadávání parametrů lze použít pouze celá čísla.

Příloha 2

Obsah CD

K diplomové práci je přiložen CD-ROM nosič s následujícími adresáři a obsahem:

- `/Text`: adresář obsahuje text této práce ve formátu pdf
- `/SQL`: adresář obsahuje skripty pro vytvoření a naplnění databáze
- `/MOD`: adresář se zdrojovými soubory aplikace MOD
- `/MapViewer/mapviewer.ear`: aplikace Oracle MapViewer
- `/MapViewer/mapviewer101202.zip`: aplikační server OC4J s aplikací MapViewer
- `/OC4J`: aplikační server OC4J s již umístěnými aplikacemi MOD a Oracle MapViewer