Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra matematiky

Diplomová práce

ON-THE-FLY GENERALIZACE NAD DATY KATASTRU NEMOVITOSTÍ

Autor: Bc. Radan Šuba

Vedoucí práce: Ing. Karel Janečka, Ph.D.

Plzeň, 2012

Zadání práce

Prohlášení

Prohlašuji, že svou diplomovou práci na téma on-the-fly generalizace nad daty katastru nemovitostí jsem vypracoval samostatně pouze s použitím uvedené literatury.

V Plzni dne 25. 5. 2012 .....

(podpis autora)

Poděkování

Děkuji vedoucímu diplomové práce Ing. Karlu Janečkovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce. Dále děkuji Ing. Petru Jarošovi za odbornou pomoc. V neposlední řadě děkuji svým rodičům za nekonečnou trpělivost a podporu.

V Plzni dne 25. 5. 2012 .....

(podpis autora)

Abstrakt

Práce se zabývá zjednodušováním map v reálném čase. Řeší volbu libovolného měřítka mapy, redundanci dat a strukturu pro ukládání výsledků generalizace. V úvodu popisuje problematiku zjednodušování map on-line, následně hledá možnou datovou strukturu pro generalizaci v reálném čase, tedy on-the-fly. Pracuje na topologickém strukturování dat na uzly, hrany, plochy a vytváří další mapová měřítka zjednodušováním měřítek základních. Těžištěm práce je implementace vybrané datové struktury s názvem tGAP (topological Generalized Area Partition) v databázovém systému Oracle.

Klíčová slova

On-the-fly, generalizace, tGAP

Abstract

The thesis is focused on map simplification in real time. It solves vario-scale, data redundancy and a structure for generalization data. The first, the thesis outlines the problem of on-line map simplification. The secondly, it is searching data structure for generalization in real time (on-the-fly). The structure works with topological data model, which is based on nodes, edges and faces. The structure also solves a creation of new maps scales by simplifying reference scale. The main content of thesis is implantation tGAP (topological Generalized Area Partition) structure in Oracle database.

Keywords

On-the-fly, generalization, tGAP

Seznam zkratek

BLG	Binary Line Generalization
ČÚZK	Český úřad zeměměřický a katastrální
ESRI	Environmental Systems Research Institute
GAP	Generalization Area Partioning
LOD	Level of Detail
MRDB	Multi-Representation Database
OGC	Open Geospatial Consortium
PL/SQL	(Procedural Language/Structured Query Language)
SQL	Structured Query Language
SŘDB	Systém řízení báze dat
tGAP	topological Generalized Area Partition

Obsah

1 Úvod 1

2 Generalizace On-the-fly 2

2.1 Vývoj 2

2.2 Srovnání on-the-fly generalizace oproti víceměřítkové databázi 3

2.3 On-the-fly generalizace pomocí generalizačních algoritmů 4

2.4 On-the-fly generalizace pomocí hierarchických struktur prostorových dat 5

3 Topologické datové modely 7

3.1 Left right topologie bez odkazů na hrany 8

3.2 Left right topologie s ukládáním odkazů na hrany 9

3.3 Okřídlená hrana 10

4 Datová struktura tGAP 11

4.1 Plochy v Topological GAP Tree 12

4.2 Zjednodušení hran: Binary Line Generalization tree 13

4.3 Dougles-Peucker algoritmus 13

4.4 Edge forest 15

4.5 Výhody struktury 15

4.5.1 Větší možnost využití 15

4.5.2 Generalizace v reálném čase 16

4.5.3 Souvislé měřítko 16

4.5.4 Eliminující nadbytečné ukládání dat 16

5 Databázový systém pro zpracování diplomové práce 16

5.1 Požadavky 17

5.2 Volba databázového systému pro zpracování diplomové práce 18

6 Implementace tGAP 18

6.1 Postup implementace 19

6.2 Výběr topologie 19

6.3 Testovací data 21

6.4 Relační datový model 21

6.5 Fyzický datový model 22

6.6 Datový typ Binary Line Generalization (BLG) strom 24
6.6.1 Použití struktury 26
6.7 GAP face tree 26
6.8 GAP edge forest 29
6.9 Princip slučování 32
6.9.1 Špatný přístup 32
6.9.2 Správný přístup 33
6.10 Slučování BLG stromů 33
7 Reálná data 35
7.1 Předzpracování a import dat do Oracle 36
7.2 Z topologického modelu do pre-tGAP 39
7.2.1 Pre-tGAP 40
7.2.2 Fyzický datový model pre-tGAP 41
7.3 Z tGAP do SDO_GEOMETRY 42
7.3.1 Stěžejní procedury při převodu 43
7.4 Výsledky práce s reálnými daty 46
8 SWOT analýza 47
9 Závěr 49
10 Citovaná literatura 51
Příloha A 54
Příloha B 55

1 Úvod

V dnešní době si už jen stěží umíme představit plánování trasy na výlet nebo obchodní cesty pouze s papírovou mapou. Takřka každý při plánování takové cesty využívá internetový mapový portál, kde je možné si prohlédnout mapu v digitální podobě s aktuálními informacemi. Avšak jen málokdo si při této činnosti uvědomí, že mapa, kterou si prohlíží, je složena z vrstev závislých na měřítku. Takto rozdělená data na jednotlivé mapové vrstvy různých měřítek jsou ukládána do několika databází propojených do sebe, jednotlivé vrstvy jsou uloženy zvlášť. Z tohoto důvodu se většina objektů v mapě uloží několikrát a kdykoliv je provedena změna v jednom z měřítek, je velmi problematické přizpůsobit i ostatní objekty v jiných měřítcích tak, aby byla mapa stále aktuální ve všech měřítcích.

Jednotlivé výsledné mapy vztažené k počátečnímu měřítku, ze kterého vycházejí, jsou vytvářeny pomocí procesu generalizace, tedy takové procedury výběru zjednodušení a zevšeobecnění obsahu mapy (objektů, jevů a jejich vztahů), při které se zdůrazní významnější na úkor podružného. Proces generalizace je známý už řadu let a vyskytuje se v kartografii takřka od počátku. Již několik desetiletí je snaha, aby generalizace probíhala automaticky. Stejně tak by bylo velmi užitečné, kdyby se veškeré „odvozené“ mapy vytvářely v reálném čase tzv. *on-the-fly*, kdykoliv se změní původní mapa. Generalizační techniky, které byly navrženy již dříve a které by bylo možné použít, je velmi problematické implementovat v počítačovém prostředí.

S narůstajícím využíváním mobilních zařízení a rozvojem internetu jsou tyto potřeby ještě naléhavější, dokonce zde vzniká další požadavek, který je odvozen od přenosu dat přes internet - nutnost přenosu pouze přiměřeného a pouze nutného množství dat. Řešením je využití generalizace, která pracuje on-line neboli *on-the-fly*.

V práci se budeme zabývat těmito otázkami:

- Jak lze realizovat generalizaci *on-the-fly*?
- Jaké jsou možné přístupy a datové struktury, které lze využít?
- Výhody a nedostatky vhodných struktur?
- Problematika implementace vybraných datových struktur?
- Možnosti použití vybraných struktur na reálná data např. katastru nemovitostí?

Práce navazuje na publikace z Technické Univerzity Delft v Nizozemí. Tato univerzita je uznávaným špičkovým pracovištěm ve světovém měřítku. Zkušenosti s generalizací *on-the-fly* v České republice nebyly dosud publikovány. Cílem práce je ověření publikovaných teoretických a praktických poznatků a vytvoření algoritmů, které by byly prakticky využitelné v našich podmínkách.

2 Generalizace On-the-fly

Tvůrce map musí mít jistotu, že informace, kterou chce pomocí mapy sdělit, je dostatečně jasná. Právě z tohoto důvodu mapa nezobrazuje skutečný svět, ale pouze jeho část, která je pro tvůrce důležitá. V této části je použita právě generalizace. Generalizaci lze tedy definovat jako zvolenou a zjednodušenou reprezentaci skutečného světa odpovídající měřítku a účelu mapy [1].

Též můžeme mapovou generalizaci definovat, a pro nás vhodněji, jako proces vytváření mapy s menší úrovní detailu (Level of Detail (dále LOD)), zatímco zachováváme základní geografickou informaci. *On-the-fly* generalizace, někdy též označovaná jako Real-time generalization, Dynamic generalization nebo On-line generalization, je pak označována jako generalizační proces v reálném čase. Podle [2] tento proces vytváří dočasné generalizované datové soubory výhradně pro vizualizaci, nikoliv pro ukládání či jiné účely. *On-the-fly* generalizace je úzce spjata s vysoce interaktivními aplikacemi kartografie jako jsou mapové služby (web mapping a mobile mapping) nebo krizové systémy, kde jsou zapojena vícenásobná měřítka. Takovéto aplikace jsou z hlediska kartografické kvality benevolentnější, naproti tomu u tradičních papírových map je na kartografickou kvalitu kladen velký důraz. Avšak požadavky na kartografickou kvalitu není možné zcela zanedbat.

Dle [2] lze dosáhnout optimální kartografické kvality dvěma způsoby:

- pomocí rychlých generalizačních algoritmů, které generalizují mapu do vhodného měřítka v reálném čase.
- využitím hierarchických struktur prostorových dat.

V obou případech jsou implementované kartografické operace poměrně jednoduché. Naproti tomu pro vytvoření kvalitní papírové mapy jsou využity sofistikované, výpočetně náročné algoritmy.

2.1 Vývoj

Kartografie byla po staletí výhradně používána pro tvorbu papírových map. V době, kdy vstoupily počítače do kartografie (60. léta [2]), a následující roky, kdy byl též nárůst počítačové techniky, se situace příliš nezměnila. Papír byl stále dominantní médium pro uchování mapy. Zatímco obrazovka sloužila pouze pro editaci a korektury zpracovaných map místo toto, aby byla též koncovým médiem. Následkem toho se výzkum v oblasti automatických generalizačních metod zaměřil na dosažení vysoké kartografické kvality, zatímco významně zanedbával výpočetní efektivitu. Přestože upřednostnění grafické kvality před efektivitou může znít počítačovým expertům nelogicky, z kartografického pohledu je velmi smysluplné. S ohledem na upřednostnění kvality se stává mapová generalizace velmi špatně definovatelnou a je těžké ji

zautomatizovat. Navíc s ohledem na to, že výsledný produkt je pouze statický, můžeme nadbytečnou výpočetní náročnost naprosto zanedbat [2].

S nárůstem osobních počítačů v 80. letech a především rozmachem internetu na začátku 90. let se začaly objevovat nové požadavky na kartografickou a mapovou generalizaci. Přestože používání a práce s mapovými službami jsou časově velmi kritické operace, změny měřítek jsou uskutečňovány v reálném čase, tedy v podstatě se jedná o náročný úkon *on-the-fly* generalizace. Nicméně přes dřívější výzkumy a práce zabývající se *on-the-fly* generalizací (např. zjednodušování linií či výběr objektů užívající reaktivní datové struktury) se vývoj v oblasti automatizace mapové generalizace takřka zastavil v 90. letech, výjimkou může být např. [3]. Známé mapové služby, jako je například mapy.cz, či maps.google.com spoléhají na pragmatičtější řešení, které zahrnuje offline víceměřítkové databáze[1] obsahující mnoho LOD. Tím je *on-the-fly* generalizace snížena na úroveň vyhledávání a zobrazování LOD, které nejvíce vyhovuje požadavkům v reálném čase. Příkladem může být umísťování popisku a symbolů v mapě pro body zájmu.

V posledních letech se objevují potřeby předefinovat kartografickou a mapovou generalizaci. Nové formy mapových aplikací jako jsou mobilní mapové služby, krizové systémy nebo systémy pro podporu rozhodování v reálném čase přesahují web mapping tak, jak jej známe a byl zmíněn dříve. Vzniká zde nová potřeba na získávání dat v reálném čase. Tyto nové mapové aplikace požadují přizpůsobivost, individuální nastavení a zařazení do kontextu obsahu tematických map společně s vyhledávacím dotazem od uživatele. Proto předzpracování a ukládání potencionálních „klasických pohledů“, které známe z běžných mapových služeb, již neposkytují vhodné řešení. Těmito problémy se zabývaly výzkumy v posledních letech. Mnohem vhodnější se ukazuje právě použití *on-the-fly* generalizace.

2.2 Srovnání on-the-fly generalizace oproti víceměřítkové databázi

Jak již bylo zmíněno v předchozí části, u *on-the-fly* generalizace se nejedná o ukládání a práci či načítání s předzpracovanými LOD z víceměřítkové databáze (MRDB). Z tohoto důvodu vyplývá, že je zbytečné jakkoliv lpět na MRDB. Navzdory tomu je víceměřítková databáze stále důležitou oblastí výzkumu. Například některé organizace uchovávají spoustu digitalizovaných map v různých měřítcích. V zájmu těchto organizací by mělo být propojení jednotlivých map či automatická aktualizace od podrobných po méně podrobná měřítka mapy.

Hlavní vlastnosti *on-the-fly* generalizace jsou [2]:

- dočasné, zmenšené měřítko (generalizované). Dataset/mapa je generován za účelem vizualizace z prostorové databáze.

[1] víceměřítkové databáze (Multi-Representation Database) - Větší počet databází propojených dohromady [23].

- mapa musí splňovat uživatelská nastavení (např. individuální obsah) a technické specifikace displeje. Například nízké rozlišení obrazovky a malá velikost.
- měřítko výsledné mapy se může lišit (především vlivem přibližování/oddalování) a není předdefinované.
- generalizace musí proběhnout automaticky, není možná interakce s uživatelem, např. schválení výsledků před publikováním.
- výsledná mapa se musí objevit na displeji během několika sekund. Uživatel nechce čekat příliš dlouho.
- u mobilních zařízení a webových služeb jsou další problémy spojené s omezením přenosu dat.

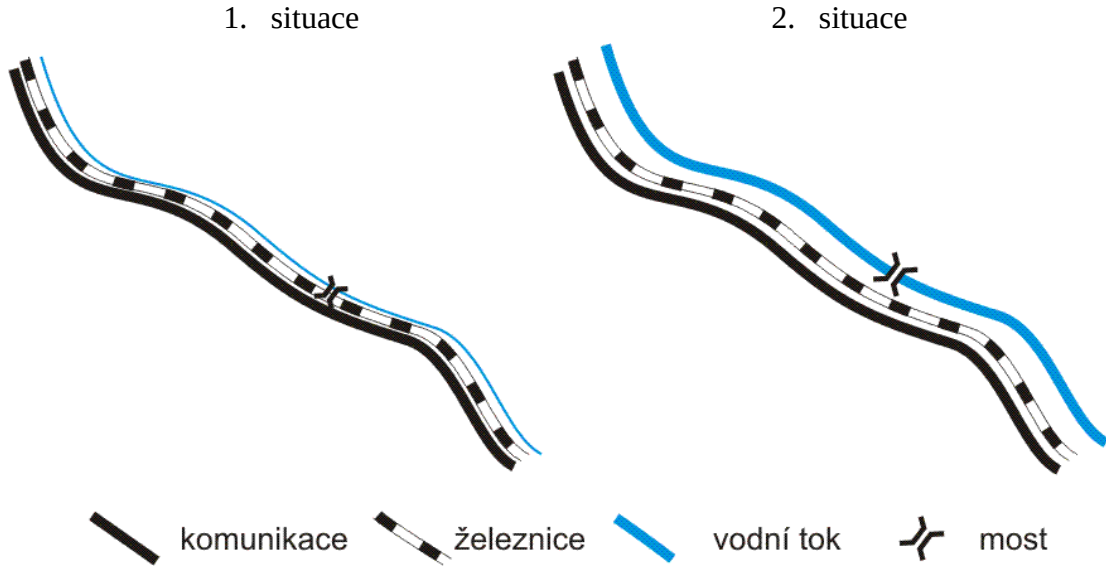
Aplikace, které používají *on-the-fly* generalizaci, využívají dva hlavní přístupy, jak získat jiný LOD v reálném čase. První z přístupů je založen na využití generalizačních algoritmů, zatímco druhý přístup využívá hierarchické datové struktury.

2.3 On-the-fly generalizace pomocí generalizačních algoritmů

Jelikož *on-the-fly* generalizace je náročná operace jak časově, tak paměťově, generalizační algoritmy musí být rychlé a/nebo musí využívat předzpracovaná data. V zásadě jakýkoliv algoritmus běžící v lineárním nebo logaritmickém čase se stává vhodným kandidátem pro *on-the-fly* generalizaci [2]. Jednoduchým příkladem může být algoritmus, který vybírá jednotlivé objekty na základě jejich předpřipraveného atributu. Navíc je snadné takovýto algoritmus implementovat. K rozšíření takovéhoto jednoduchého algoritmu se může použít některý ze známých algoritmů: výběr objektu na základě jeho třídy, selekce plochy podle nimimálního/maximálního kritéra, výběr linie podle její minimální/maximální délky, volba vrstevnice na základě rozsahu výšky, zjednodušení linie využívající Douglas-Peuckerův algoritmus [4] nebo zjednodušení obrysu budovy. Kombinace několika algoritmů je také možná.

Algoritmy, které jsme zmínili doposud, nebyly původně navrženy přímo pro generalizaci *on-the-fly*, avšak jsou pro tyto účely využívány převážně z důvodu jejich jednoduchosti a výpočetní nenáročnosti. Algoritmy, které jsou speciálně navrženy pro dynamickou generalizaci, jsou zmíněny v [2]. Tyto algoritmy provádějí zjednodušení linií velkých datových sad využívajících nového přístupu ke grafickému hardwaru (buffer snímků, baffer barev, baffer šablon, atd.) pomocí vectorového/rastrového přístupu. Pro interaktivní zobrazování jsou předzjednodušené mapy s rozdílným LOD organizovány v hierarchické struktuře, datovém stromu. Tyto metody tedy přímo spoléhají na hierarchické datové skupiny, které budou vysvětleny v další podkapitole. Tím se stává toto řešení hybridem mezi algoritmickým přístupem k řešení generalizace *on-the-fly* a *on-the-fly* generalizací pomocí hierarchických struktur, které budou probrány v další části [2].

Algoritmy, které jsou zmíněné výše, jsou použity pro jednoduché generalizační operace, avšak jsou to operace bez ohledu na kontext. Prvky jsou tedy zjednodušovány bez návaznosti na ostatní mapové prvky. Takovými operaci mohou být zjednodušení linie nebo vyhlazení linie. Komplexní generalizační operace, které generalizují objekty v souvislosti s ostatními prvky, jsou například zvětšení, přesun nebo přemístění [5].



Obr. 2.1: Příklad generalizace s ohledem na kontext. V 1. situaci je zobrazen stav před úpravou. V 2. situaci je zachycena generalizace s ohledem na okolní prvky. Liniový prvek reprezentující vodní tok byl zvětšen a posunut. Prvek reprezentující most byl zvětšen a přemístěn.

Pomocí těchto složitějších operací dosáhneme větší kartografické kvality. Tato kvalita je však vykoupena tím, že se nehodí pro algoritmy pracující v reálném čase, tedy pro *on-the-fly* generalizaci. Důvod, proč nejsou přímo vhodné, je prostý, většinou je nutná interakce s tvůrcem mapy. Možné řešení se naskýtá při použití dvou nebo více vrstev s rozdílným LOD, tedy počáteční a koncovou vrstvou. Víme, z čeho se vychází a kam máme dojít. V takovém případě je však nutné, aby alespoň dvě LOD obsahovaly body, které si korespondují a jsou propojené přes MRDB [2].

Vhodnější přístup jak dosáhnout komplexnějšího generalizačního „chování“, je využití hierarchické datové struktury, které jsou obsahem další sekce.

2.4 On-the-fly generalizace pomocí hierarchických struktur prostorových dat

Výsledkem mapové generalizace jsou v tomto přístupu hierarchicky řazené mapy s „hrubším“ detailem. Z toho je zřejmé, že pro generalizaci je vhodné použít hierarchické datové struktury, které dokonce zaujímají významné místo pro urychlení *on-the-fly* generalizací [2]. V této sekci se budeme zabývat některými příklady, které řeší hierarchickou reprezentaci prostorových dat ve

stromové struktuře. Tyto příklady se snaží vytvořit datovou strukturu s proměnným měřítkem (*variable-scale data structure*), která by eliminovala přebytečné ukládání dat, které je typické pro víceměřítkové databáze s množstvím LOD.

Při generalizaci bodů, běžné u tématických map (např. výskyt zvěře, místa dopravních nehod, místa zájmu), lze též použití hierarchické datové struktury. Jsou navrženy dvě metody. První používá čtyřstromy (*QuadTree*) [6] pro indexaci původních bodů a pustupné ukládání do stromu o několika úrovních. V programu jsou původní body nahrazovány centroidy QuadTree buněk, které odpovídají příslušnému rozlišení (např. měřítko cílové mapy [2]). Nevýhoda tohoto přístupu spočívá v tom, že výsledný bod bude zarovnán do pravidelného rozložení v „mřížce“ čtyřstromu. Druhá metoda využívá hierarchickou *teselaci*[2] mapového prostoru, který souvisí s významem bodů v mapě. Příklad zmíněný v [7] popisuje mapování výskytů zvěře do hierarchické sítě povodí, které často tvoří fyzické překážky pohybu zvířat.

Jedny z prvních návrhů stromové struktury vhodné pro *on-the-fly* generalizaci linií byly zmíněny v publikaci [3]. Jedná se o Binary Line Generalization (BLG) Tree využívající známý Dougles-Peuckerův algoritmus pro běžné zjednodušení linie. Algoritmus projde jednotlivé vrcholy linie a na základě jejich vlastností postupně uloží do binárního stromu[3]. BLG strom je omezen tím, že je vždy použit pouze pro jeden liniový prvek a nemůže být použit pro prostorové operace více mapových objektů (chybí indexace [6]). Toto omezení je však překonáno díky reaktivnímu stromu[4] (*Reactive Tree,* [3]) a jeho rozšíření R-tree [8], který ukládá úrovně důležitosti mapových objektů (důležitější objekty jsou uloženy výše ve stromové struktuře). Reaktivní stromy mají tu výhodu, že je možné přidávat i ubírat prvky ve stromové struktruře.

Reaktivní strom s BLG stále zůstává nevhodný pro generalizaci polygonových map, protože nezachovává vlastnosti objektů takovéto mapy, například vlastnosti ploch: která plocha je vlevo či vpravo. Tento nedostatek vedl k rozšíření s názvem Generalization Area Partioning (GAP) Tree [9], který definuje postupné agregace sousedních polygonů ve stromové struktuře. V [10] byla představena topologická verze GAP Tree -tGAP, která kombinuje BLG Tree a Reaktivní strom. Tato topologická verze se vyhýbá nadbytečnému ukládání dat a eliminuje vznik třísek[5] podél hranic sousedících polygonů. Řeší tak problémy, které často vznikaly v původní struktuře GAP Tree [2].

Přístup, který by řešil oba způsoby najednou, hierarchickou strukturu pro body a strukturu pro plochy, dosud nebyl navrhnut. Zatím se jedná o dva rozdílné směry, kterými se výzkum ubírá. Z hlediska katastru nemovitostí jsou ukládána data, která charakterizují jednotlivé plochy

[2] Teselace je proces, pomocí kterého se obecný polygon převádí na nepravidelnou trojúhelníkovou síť [7].

[3] Binární strom - datová stromová struktura. Každý vrchol stromu může mít maximálně dva následovníky [24].

[4] Reaktivní strom - datová stromová struktura. Důležité objekty jsou být uchovány ve vyšších úrovních stromu [26].

[5] Tříska - třísky a mezery (*Sliver and gaps*) – tento problém nastává, když se dvě hranice polygonů překrývají/nedotýkají, ačkoli mají představovat identickou hranici [25].

(parcely) v podobě linií (chápeme jako úsečku) či polylinií (chápeme jako objekt skládající se z více úseček). Převážně z tohoto důvodu se v dalších částech diplomové práce budeme zabývat pouze *on-the-fly* generalizací pro plochy/linie využívající právě reaktivní datové struktury. Topologická verze struktury GAP Tree jako jediná splňuje potřebné požadavky, které budou zmíněny v dalších částech textu.

3 Topologické datové modely

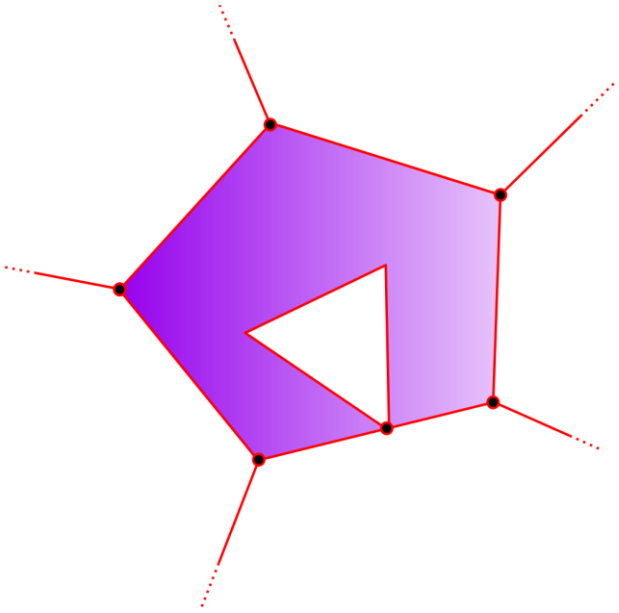
Prostorová data je nutné ukládat, způsob takovéhoto ukládání však není úplně jednoduchý. Dodavatelé databázových systémů mají ve svých produktech již integrované prostorové datové typy více či méně odpovídající specifikacím OGC (*Open Geospatial Consortium*). Podle těchto specifikací může být prostorový objekt reprezentován v systému řízení báze dat[6] (dále jen SŘBD) dvěma rozdílnými logickými vektorovými modely [11]:

- geometrickým datovým modelem.
- topologickým datovým modelem.

Každý z těchto přístupů má jisté výhody stejně tak i stinné stránky. Geometrický model poskytuje přímý přístup k souřadnicím objektu, zatímco topologický model nám dává informace o prostorových vztazích (např. plocha, která leží vlevo). Jelikož geometrický model přímo odkazuje na souřadnice, budou společné hranice ploch uloženy dvakrát, což způsobuje redundanci dat. Topologický přístup se vyhýbá redundanci použitím odkazů na jedinečné identifikované hrany a body [12].

Plošný objekt v topologickém datovém modelu je reprezentován topologickou plochou, která se skládá z hranového řetězce (*ring*) se směrem. Každý hranový řetězec se skládá z jedné nebo více hran (*edge*) a jednoho nebo více uzlů (*node*). Každá plocha má aspoň jeden vnější hranový řetězec (orientovaný po směru hodinových ručiček) a žádný nebo více vnitřní hranových řetězců (orientovaný proti směru hodinových ručiček). Vnější řetězec definuje hranice plochy. Vnitřní řetězec může reprezentovat díry v objektu. Pro topologické plochy platné podle OGC specifikace SQL žádný bod na stejné hraně nesmí být totožný s bodem, který také definuje řetězec. Kromě toho, vnitřní hranový řetězec se může dotýkat vnějšího řetězce nanejvýše v jednom bodě [12].

[6] Systém řízení báze dat (*database management system*) – Zkracováno na databázový systém, s akronymem SŘBD či DBMS

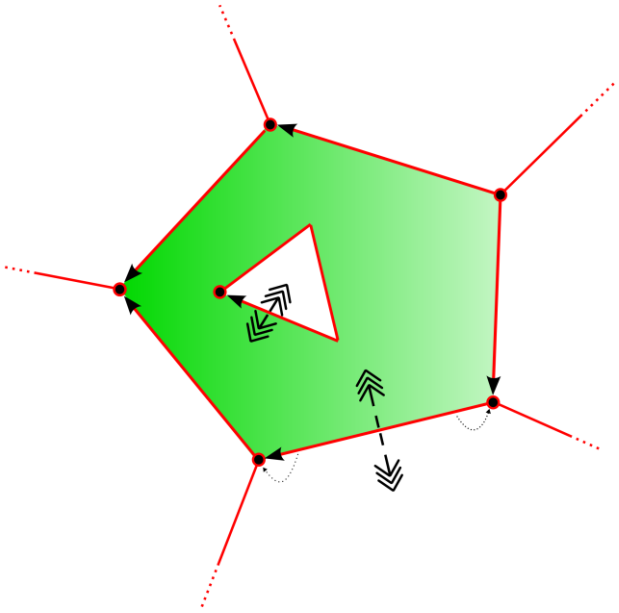


Obr. 3.1: Správně definovaná plocha. Vnitřní prstenec se dotýká vnějšího prstence pouze v jednom bodě [12].

Pro ukládání bodu, hran a ploch, které nám definují topologický model v SŘDB, je nutné vytvořit některé tabulky. Zapotřebí jsou nejméně tři: pro plochy, hrany a body. Větší podrobnosti o tabulkách, stejně tak o funkcích a procedurách, které s nimi pracují, nám dá přesně zvolený typ ukládání topologického modelu v databázi. V [11] jsou zmíněny tři možnosti ukládání, každý má jak své výhody, tak i své nedostatky. Jednotlivé přístupy jsou ilustrovány v kapitolách 3.1, 3.2 a 3.3.

3.1 Left right topologie bez odkazů na hrany

Left right topologie bez odkazů na hrany je založena na ukládání informací o stěně vlevo a vpravo pro každou orientovanou hranu. Na základě této informace můžeme získat kolekci hran náležejících jedné stěně a z nich stěnu zrekonstruovat. Všechny hrany jsou orientované a každá hrana má počáteční a koncový uzel. Mezi vybranými hranami musí proběhnout operace vyhledání hran, které lze dočasně vzájemně „navázat“, tzn. mající společný uzel.



Obr. 3.2: Left right topologie bez odkazů na hrany [12].

Ve chvíli, kdy jsou vytvořeny dočasné hranové řetězce, musí být souřadnice uzlů orientovány příslušným způsobem, tzn. hranový řetězec utvářející vnitřní hranici v protisměru chodu hodinových ručiček, hranové řetězce utvářející vnitřní hranice ve směru chodu hodinových ručiček. Tato orientace může být odvozena z výpočtu tzv. signed area pro každý dočasný hranový řetězec. Každý hranový řetězec obsahuje n bodů $(x_i; y_i)$; i = 0, … , n, $x_0 = x_n$, $y_0 = y_n$. Signed area je pak vypočítána podle následujícího vztahu

$$A_{signed} = \frac{1}{2}\sum_{i=0}^{n-1} a_i,$$

kde $a_i = x_i y_{i+1} - x_{i+1} y_1$

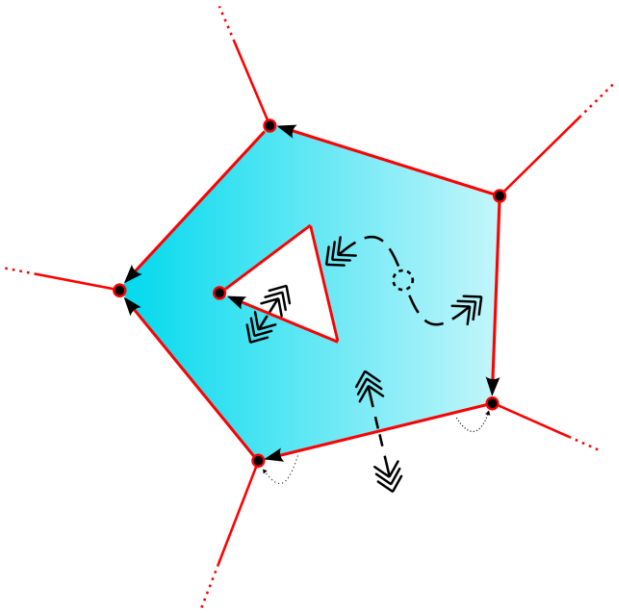
Rozhodnutí, který dočasný hranový řetězec bude označen jako vnitřní a které dočasné hranové řetězce jako vnější, závisí na velikosti absolutní hodnoty A_{signed}. Hranový řetězec s nejvyšší hodnotou musí být řetězcem vnějším.

Výhodou tohoto přístupu je, že se do databáze neukládá příliš mnoho odkazů. Nevýhodou je naproti tomu nemožnost rozhodnout, které dočasné hranové řetězce jsou vnitřní a které vnější bez nutnosti provedení geometrických výpočtů. Rovněž je nutná operace vyhledání sousedních (navazujících) hran [11].

3.2 Left right topologie s ukládáním odkazů na hrany

Tento model je opět založen na ukládání informací o stěně vlevo a stěně vpravo pro každou orientovanou hranu. Na základě této informace může být vybrána množina hran, které náleží

jedné stěně. Každá hrana obsahuje dva uzly - počáteční a koncový. Pomocí těchto informací o uzlech lze zrekonstruovat dočasný hranový řetězec.

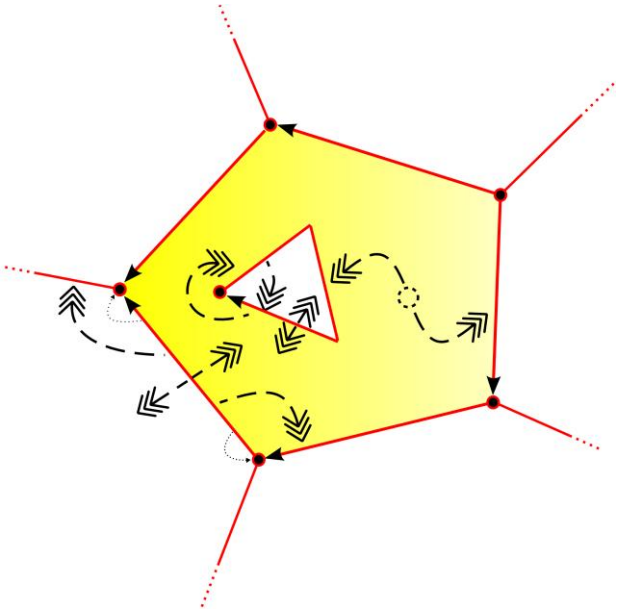


Obr. 3.3: Left right topologie s ukládáním odkazů na hrany [12].

Za další, pro každou stěnu jsou explicitně uloženy odkazy na její vnější hranový řetězec a všechny vnitřní hranové řetězce. Odkazy na počáteční hrany těchto řetězců mají přiřazeny znaménko, aby bylo zřejmé, jak jsou počáteční hrany vůči stěně orientovány. Dočasné hranové řetězce pak mohou být díky této informaci přímo korektně orientovány bez nutnosti dalších výpočtů, což je hlavní výhoda tohoto konceptu. Nevýhodou je režie nutná pro ukládání odkazů na počáteční hrany a znamének u každé stěny [11].

3.3 Okřídlená hrana

Koncept okřídlené hrany je založen na ukládání nejméně dvou odkazů u každé orientované hrany. Odkazů na next left hranu a next right hranu. Pomocí těchto odkazů je možné snadno utvářet hranové řetězce. Rovněž by u každé orientované hrany měly být uloženy odkazy na stěnu vlevo a stěnu vpravo od hrany. Pro každou stěnu je uložen odkaz na počáteční hranu vnějšího hranového řetězce a odkazy na počáteční hrany vnitřních hranových řetězců. Odkazy na počáteční hrany řetězců jsou ukládány společně s orientacemi počátečních hran vůči stěně, což je důležité pro správnou orientaci vnější hranice, resp. vnitřních hranic stěny [11].



Obr. 3.4: Koncept okřídlené hrany [12].

Při sestavování hranového řetězce začneme u počáteční hrany a pokračujeme podle znaménka odkazu na tuto hranu buď hranou pod odkazem next left, případně next right. Proces končí ve chvíli, kdy bychom do řetězce přidali počáteční hranu.

Výhodou této struktury je jednoduchost konstrukce hranového řetězce, pouze pomocí odkazů. Velmi rychle by tak byl například zodpovězen požadavek na vybrání všech (hranic) parcel, pokud by tyto byly uloženy pomocí konceptu okřídlené hrany. Nevýhodou je udržování velkého množství odkazů, což může při značném objemu dat (miliony parcel) vést ke zvýšenému nároku na diskový prostor [11].

Pro provádění generalizace *on-the-fly* s daty katastru nemovitosti se zdá jako vhodná volba využití hierarchických datových struktur, například abychom mohli pracovat s daty v širším kontextu, což bylo zmíněno dříve. Generalizaci jednotlivých objektů (parcel) by bylo vhodné provádět v závislosti na sousedních objektech. Například sloučit dohromady podobné plochy. K tomu, abychom takové podmínky mohly využít, je nezbytné pracovat s daty uloženými v topologickém modelu, proto zde nebyl a ani nebude geometrický model ukládání dat uvažován, v tomto případě je geometricky model irelevantní. Jestliže jsou data katastru nemovitostí uložena v topologickém modelu, vyhneme se tím redundanci dat a zároveň se hodí pro použití struktury topological Generalized Area Partition (dále jen tGAP).

4 Datová struktura tGAP

Tato kapitola bude popisovat datovou strukturu topological Generalized Area Partition, která byla popsána v [10] a následně v dalších pracích, které na základě tohoto článku vznikly, např. [12].

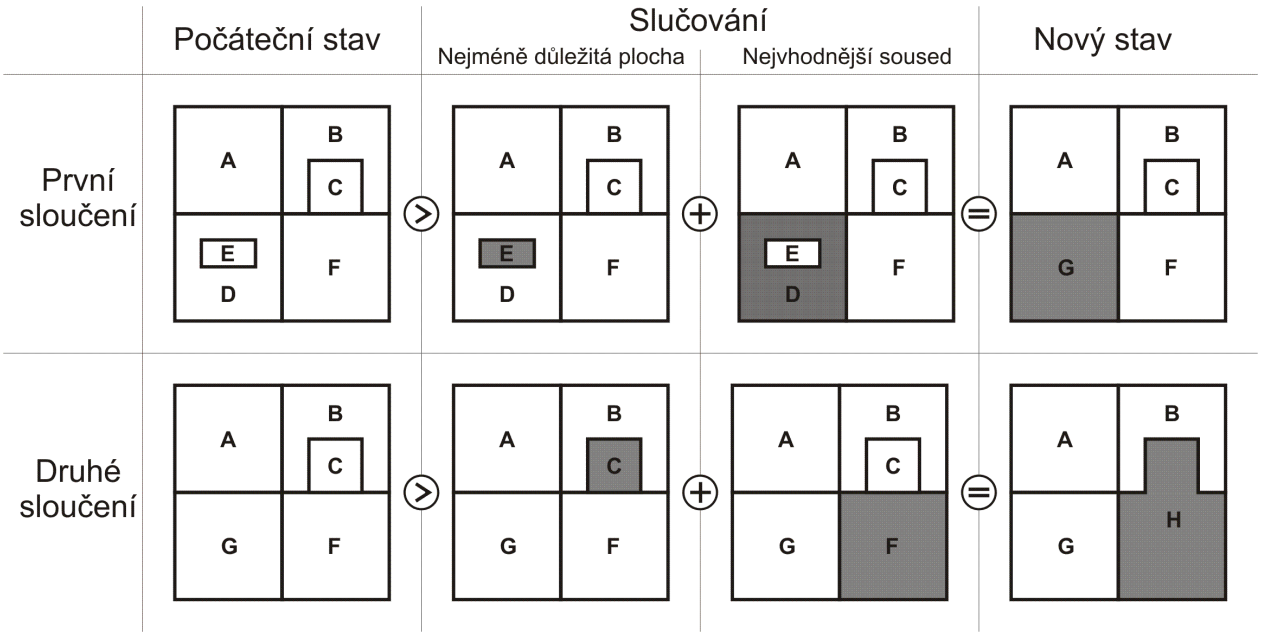
Popis struktury, její části a základní principy, na kterých funguje, budou uvedeny v této kapitole. Jak struktura pracuje s daty společně s ukázkou použití a popisem jednotlivých kroků bude uvedeno v kapitolách 6.7, 6.8, 6.9 a 6.10.

Pomocí této topologické reaktivní struktury je možné uskutečnit slučování ploch. V principu se jedná o vybrání nejmenší plochy a ta se sloučí s největší sousední plochou.

4.1 Plochy v Topological GAP Tree

Topological Generalized Area Partition je rozšířenou verzí staršího konceptu s názvem The GAP tree. Původní GAP tree je stromová struktura, která ukládá informace o generalizovaných topologických plochách. Např. umožňuje výběr plochy vhodné pro jistou úroveň detailu. Největším rozdílem oproti starší verzi je to, že nová struktura využívá topologického modelu ukládání a tím snižuje redundanci dat [10].

Jelikož se využívá dělení rovinného popisu ploch (žádné mezery ani přesahy nejsou možné), není možné jen tak odstranit plochu pro nižší úroveň detailu, protože by to právě vedlo k výskytu děr (*holes*). Pro vyšší úrovně detailu je vždy nejméně důležitá plocha sloučena s jejím nejvhodnějším sousedem. Tento postup pokračuje tak dlouho, až zůstane pouze jediná plocha. Pro takovýto postup jsou potřeba vždy dvě metody: Jedna, která rozhoduje, která plocha je nejméně důležitá a bude tedy sloučena. Druhá, která vrací nejvhodnějšího souseda, se kterým by se mohla vybraná plocha sloučit. Výsledkem těchto metod, pak budou dvě plochy, které se budou slučovat a vytvoří tak novou plochu [12].



Obr. 4.1: Slučování pro konstrukci GAP face tree. Vytvořeno dle vzoru [12]

S použitím struktury GAP face tree, která ukládá plochy do stromové struktury, má každá plocha přiřazen rozsah důležitosti (*importance range*). Tento rozsah se skládá z dolní hranice důležitosti (low importance) a horní hranice důležitosti (high importance). Na základě průniku dvou požadavků: hodnoty důležitosti (importance level) a rozsahu důležitosti jednotlivých ploch, bude výsledem množina ploch, které jsou vhodné pro zobrazení. Prostřednictvím tohoto výběru se bude počet ploch snižovat, protože výše ve stromu bude méně ploch.

4.2 Zjednodušení hran: Binary Line Generalization tree

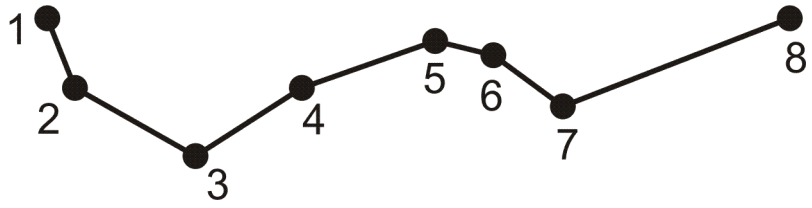
Zatím však bylo ukázáno pouze to, jak jsou zjednodušovány plochy a do jaké struktury se uloží. Co se stane s hranami? Jak budou zjednodušovány? Tyto a další otázky budou zodpovězeny v této části. Pro zjednodušování a ukládání hran je potřeba další struktura.

Každá plocha se skládá z množiny hran a množiny uzlů. Pokud zjednodušíme plochu, musíme zjednodušit i množinu hran. Společně s tímto zjednodušováním se nám redukuje i počet bodů, které linii reprezentují. Využití topologie se zde tím pádem přímo nabízí, protože každá hrana je uložena pouze jednou, a proto je zjednodušena pouze jednou. Přestože dochází v topologickém modelu ke změnám, informace o sousednosti jsou stále zachovány, stejně tak je stále zachována vlastnost, která eliminuje výskyt děr a překrytí. Algoritmus, který je použit v tGAP struktuře pro zjednodušování linií, má název Dougles-Peuckerův algoritmus.

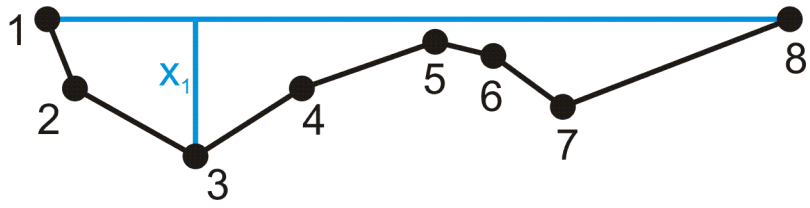
4.3 Dougles-Peucker algoritmus

Dougles-Peucker algoritmus se dá klasifikovat jako iterační, globální algoritmus [5]. Pojem iterační, jak zmiňuje [5], můžeme chápat jako: „*Rekurzivní dělení polyline na menší segmenty na základě geometrické podmínky. V každém rekurzivním poddělení je nalezen bod.*“ Pojem globální algoritmus je pak vysvětlen jako algoritmus, který zohledňuje tvar linie jako celku. Posuzuje geometrické parametry vybraného bodu vzhledem ke všem ostatním bodům lomené čáry [5]. Další výhoda algoritmu může být jeho univerzálnost (lze použít na libovolnou linii). Podrobnější popis lze získat přímo ze zdroje [4].

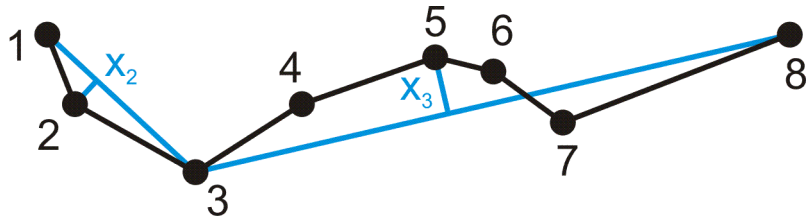
Obrázky obr. 4.2, Obr. 4.3 a Obr. 4.4 nám dávají představu, jak algoritmus pracuje. Jsou v něm nastíněny první dva kroky, ve kterých je naznačeno, jak celý algoritmus pracuje.



Obr. 4.2: Princip Dougles-Peuckerův algoritmus: Linie, počáteční fáze.



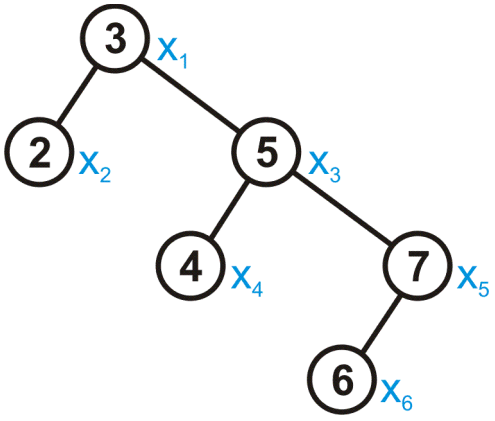
Obr. 4.3:Princip Dougles-Peuckerův algortimus: První krok, při kterém je nalezen první (nejvzdálenější) uzel stromu - kořen stromu.



Obr. 4.4: Princip Dougles-Peuckerův algortimus: Druhý krok, nalezení dalších dvou následovníků.

Algoritmus začne spojením počátečního a koncového budu linie. Pro každý mezilehlý bod je vždy dopočtena vzdálenost od linie. Tato vzdálenost též může sloužit jako jakýsi práh, pomocí kterého je možné například vybrat body vhodné pro zobrazení linie v jisté úrovni detailu. Nejvzdálenější bod pro danou iteraci algoritmu je vybrán a uložen do sktruktury (Např. v obr. 4.3 to je bod 3). V dalším kroku jsou pak procházeny liniové segmenty: část od počátku k bodu, který byl vybrán v předchozím kroku (bod 3) a od vybraného bodu (bod 3) do konce. Takto algoritmus rekurzivně[7] pokračuje, dokud nejsou zaznamenány všechny body linie.

Výsledky zjednodušování linií pomocí Douglas-Peuckerova algoritmu lze ukládat do stromové struktury. Tato binární stromová struktura se nazývá Binary Line Generalization (BLG) Tree. Příklad takovéhoto stromu je zachycen v obrázku Obr. 4.5.



Obr. 4.5: BLG strom pro linii uvedené na Obr. 4.2. Bod tři je uložen ve stromové struktuře nejvýše (byl nalezen pří prvním kroku Dougles-Peuckerovo algoritmu).

[7] Rekurze z latinského slovesa recurso (vrátit se), znamená, že program volá přímo sám sebe.

Pokud hledáme bod linie, pak stačí projít stromovou strukturu. Hodnota vzdálenosti od linie je uložena pro každý bod, tím se vyhneme opětovnému dopočítávání vzdáleností. Nejvzdálenější a většinou nejcharakterističtější body celé linie jsou ukládány v nejvyšších úrovních stromu. Zatímco velmi podrobné a málo se odchylující body od „osy“ linie jsou uloženy hlouběji ve stromové struktuře. Pokud chceme zobrazit linii se všemi detaily, stačí projít cely strom a zobrazit všechny body.

4.4 Edge forest

Pojem Generalized Area Partition edge forest (častěji GAP edge forest) je užíván při procesu zjednodušování hran jednotlivých ploch. Když zjednodušujeme hrany, využíváme k tomu BLG stromy, ale jakmile dochází ke spojování ploch, není úplně zřejmé, jak se máme vypořádat se slučováním hran, které tyto plochy definují. Při sloučení dvou ploch mohou nastat případy, které je nutné řešit: Hrana leží uprostřed slučovaných ploch. V takovém případě je správné takovou hranu odstranit. Toto odstranění může však vést k situaci, že ohraničující hrany slučovaných ploch mají společný bod a měly by se spojit, aby nebyl topologický model narušen. Jakmile však spojíme dvě hrany, měl by vzniknout nový záznam, který by hranu, spojenou ze dvou původních, správně popsal. V tomto okamžiku však dochází k redundanci dat. Lze však tento problém vyřešit tím, že záznam nové hrany bude pouze odkazovat na dva předchozí BLG stromy, nikoliv vytvářen nový.

Spojování hran rozšíří původní BLG základní stromy do větších celků, opět reprezentovaných pomocí stromových struktur. Z toho důvodu jsou tyto hranové datové struktury nazývány GAP edge forest (ze stromů se stanou *lesy* - *forest*) [12].

4.5 Výhody struktury

4.5.1 Větší možnost využití

Získávání a ukládání údajů ve velkém měřítku, jak je obecně známé, je náročný a drahý proces. Možnost použití, takto pracně získaných dat by mohla být navýšena, jestliže by datové sady menších měřítek byly generovány z datových sad měřítek velkých automaticky. K dosazení takového cíle je zapotřebí, aby operace s takovými objemy dat byly rychlé a vhodné ke generalizaci. V posledních letech se některé projekty zabývající se touto problematikou objevují, bohužel pracují pouze s malými soubory dat bez řešení celého problému jako celku. Jeden z fundamentálních problémů je absence struktury, která by ukládala právě taková data a jejich postupné generalizované výsledky. Tato práce, která navazuje na předchozí výzkumy, se snaží právě takovouto vhodnou strukturu popsat a použít.

4.5.2 Generalizace v reálném čase

Generalizace v reálném čase je velmi náročná operace, především z hlediska množství dat, která je potřeba uživateli poskytnout za velmi krátký čas. Uživatel očekává rychlou reakci například na svém mobilním zařízení. Pomocí generalizace je možné tento problém řešit. Proces zjednodušování totiž snižuje i množství dat, s kterými se manipuluje. Tímto způsobem by se snížil objem dat posílaných uživateli, což má za následek i snížení času. Zároveň by to vedlo k rychlejší odezvě, kterou uživatel očekává.

Navzdory tomu, že s použitím *on-the-fly* generalizace snížíme objem posílaných dat, stále nemůže proces zjednodušování fungovat v reálném čase. Generalizační procesy jsou i nadále časově i paměťově náročné. Východiskem z této situace může být ukládání průběžných výsledků generalizačního procesu, díky němuž získáme připravené výsledky ještě předtím, než o ně uživatel zažádá. Následné zpracování uživatelského dotazu bude spočívat pouze v odeslání výsledných dat uživateli, nové zjednodušování neproběhne.

4.5.3 Souvislé měřítko

Měřítka, která jsou požadována při zobrazování geografických dat, jsou velmi závislá na uživatelské aplikaci. Při přechodu mezi jednotlivými aplikacemi či rozhraními může zobrazování dat působit problém. Například jsou objekty zobrazeny příliš blízko nebo naopak příliš „vzdáleně“. V takové situaci by bylo nejvhodnější si zvolit měřítko individuálně. Upotřebit tyto požadavky je možné díky souvislému měřítku (*continuous range of LODs*). Struktura tGAP takovéto souvislé měřítko využívá namísto několika základních měřítek.

4.5.4 Eliminující nadbytečné ukládání dat

Topologický model ukládání dat má tu výhodu, že dokáže identifikovat vztahy, které mezi sebou prostorové objekty navzájem mají. Např. Víme, která plocha leží nalevo či napravo od linie. Díky těmto vlastnostem můžeme eliminovat prvky, které jsou duplicitní. Struktura tGAP využívá topologický model dat a díky tomu je možné ukládání geodat ve struktuře takovým způsobem, že nedochází k jejich redundanci.

5 Databázový systém pro zpracování diplomové práce

Datovou strukturu je možné implementovat do SŘDB, ale aby pracovala struktura správně, je nutné splnit některé požadavky, které musí SŘDB splňovat. V této kapitole budou zmíněny jednotlivé požadavky na vhodný SŘDB a v závěru bude zdůvodněna volba databázového systému použitého v diplomové práci.

5.1 Požadavky

Jednu z nejdůležitějších vlastností, které musí zvolená SŘDB splňovat, je možnost využití nějakého programovacího jazyka. Například pro vytvoření algoritmu, který zrekonstruuje geometrie ze struktury do původního stavu. Tento programovací jazyk by měl být přímo součástí databázového systému. Většina databázových systémů podporuje dotazovací jazyk SQL, ale pro práci se strukturou tGAP to není dostatečné, je potřeba takový jazyk, který umožňuje konstrukce procedurálního programování [13].

Stejně tak je nutné, aby vybraný databázový systém podporoval ukládání prostorových dat využívající topologický model. To znamená, že je nezbytné, aby databázový systém pracoval se základními objekty, jako jsou body, linie a plochy společně s jejich odkazy na jednotlivé atributy.

Je též nezbytné, aby databáze podporovala práci s těmito prostorovými objekty, abychom dokázali vybrat hranu, která pomocí jednoznačného identifikátoru odkazuje na plochu. Potřebujeme též výběr objektů na základě jejich geometrie nebo na základě jejich vztahů v prostoru. Např. Chceme-li vybrat linii, na které leží vybraný bod. Výběry tohoto typu se nazývají prostorové dotazy a je nezbytné, aby byly v SŘDB podporovány.

Dále je nezbytné, aby bylo možné jednotlivé objekty a jejich související záznamy v databázi rychle nalézt. Plochy odkazující na hrany musí být snadno dohledatelné. Toho lze dosáhnout použitím tzv. indexů.

K tomu, abychom mohli používat databázi jako zdroj dat, je nutné zajistit, aby se do ní dostala jen data, která tam patří a neztratila se data, která nemají. K zajištění je potřeba mít určitý mechanismus. Takový mechanismus je integritní omezení. Databáze, respektive data jsou konzistentní, pokud jsou ve stavu, vyhovující integritním omezením. To znamená, že se žádnou úpravou (například vložením, smazáním) neztratila nebo nepoškodila data, nebo že v databázi nejsou data, která tam nemají co dělat, například hrany smazané plochy. Z těchto důvodů bychom neměli opomíjet ani nutnost udržení integrity databáze.

Veškeré hodnoty, s nimiž v databázi pracujeme, můžeme rozdělit do několika skupin zvaných datové typy. Každý datový typ představuje množinu hodnot, se kterými můžeme provádět společné operace a které mají společnou vnitřní reprezentaci. V mnoha případech je však velmi užitečné tyto datové typy slučovat nebo pomocí nich dokonce vytvářet nové tzv. abstraktní datové typy. Hlavním cílem abstraktních datových typů je zjednodušit a zpřehlednit program, který provádí operace s daným datovým typem [14]. Proto by bylo velmi užitečné, aby zvolený databázový systém podporovat abstraktní datové typy.

5.2 Volba databázového systému pro zpracování diplomové práce

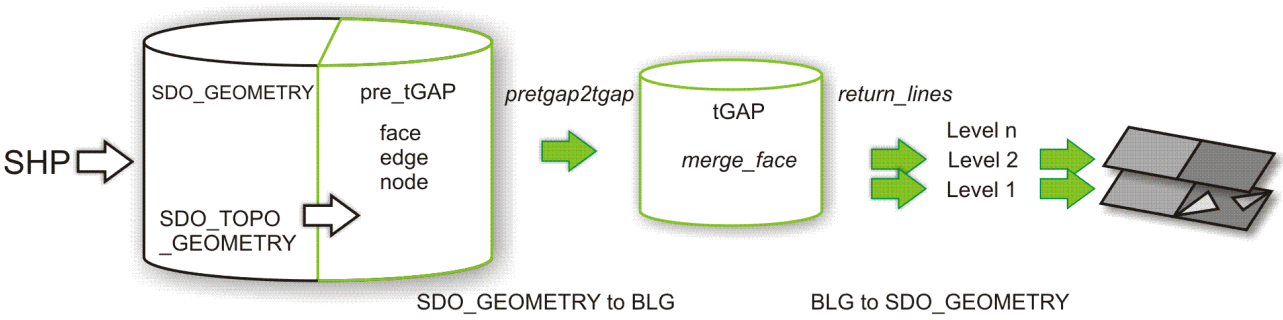
Na základě požadavků zmíněných v části 5.1 byl vybrán systém řízení báze dat Oracle. Systém byl vybrán z několika důvodů. Jedním z nich je podpora 3D indexace prostorových dat [12]. Při práci s datovou strukturou nabízí Oracle plně vyvinuté a funkční prostředí. Nalezneme i další důvody, proč si vybrat databázový systém Oracle. Např. uživatelská podpora a kvalitní dokumentace.

Závěrem je třeba zmínit, že předchozí práce, na které tato diplomová práce navazuje, využívaly systém Oracle. Tím je možné využít velké znalosti, které byly popsány či se vyhnout chybám, které byly eliminovány.

6 Implementace tGAP

Tato kapitola se bude zabývat implementací celé vybrané datové struktury pro generalizaci. Poskytne nám přehled o jednotlivých krocích a získaných poznatcích vyplývajících z práce se strukturou tGAP. Bude se zabývat jednotlivými aspekty implementace struktury do SŘDB Oracle.

Lepší představu o obsahu následující kapitoly získáme při pohledu na diagram **Chyba! Nenalezen zdroj odkazů.**, kde jsou schematicky znázorněny jednotlivé části řešené v této práci. Celý proces probíhá zleva doprava. Zelenou barvou jsou znázorněny kroky, které probíhají automaticky v procedurách a funkcích. Ostatní kroky je nutné řešit ruční úpravou dat. Z obrázku je patrné, že vstupní data jsou soubory typu shapefile, které jsou převedeny do databázového systému Oracle. V Oracle jsou data uložena v datovém typu SDO_GEOMETRY. Následujícím krokem je převedení dat do topologického datového modelu definovaného Oraclem. Z topologického modelu se některá data převedou do tabulek popsaných jako pre-tGAP. Poté z těchto tabulek automaticky proběhne import dat do struktury tGAP pomocí spuštěné procedury *pretgap2tgap*. Jakmile jsou data ve struktuře tGAP, stačí spustit proceduru *merge_face*, která vytvoří stromové struktury. Na závěr může uživatel spuštěním procedury *return_lines* shlédnout výsledky celého procesu. V této kapitole o implementaci se zaměříme jen na střední část diagramu, který definuje strukturu tGAP a práci s ní.



Obr. 6.1: Implementace struktury tGAP. Zeleně jsou vyznačeny oblasti, kterými se diplomová práce zabývá.

6.1 Postup implementace

Při zpracování této diplomové práce byla implementace řešena následovně: nejprve byl řešen datový model struktury tGAP, poté její naplnění a nakonec vytváření jednotlivých stromových struktur. To vše probíhalo pouze na testovacích datech, až později byla implementace testována na reálných datech. Následující text bude sledovat tento představený postup.

V kapitole 6.5 Fyzický datový model bude popsána celá struktura uložená v Oracle. V kapitole 6.6 Datový typ Binary Line Generalization (BLG) strom je zachycena realizace stromové struktury pro hrany. Kapitola 6.7 GAP face tree popisuje naplňování stromové struktury pro plochy. V kapitole 6.8 GAP edge forest je definováno slučování hran a v 6.10 Slučování BLG stromů je nastíněno slučování stromů hran. Práci s reálnými daty se věnuje kapitola 7.

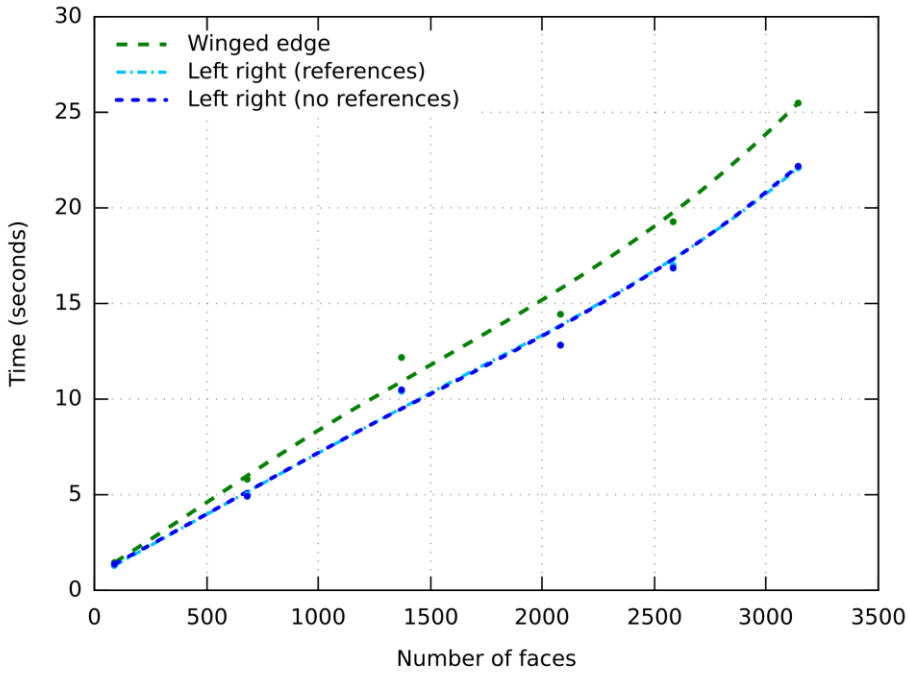
6.2 Výběr topologie

Hierarchická datová struktura tGAP, kterou chceme implementovat, využívá ukládání zjednodušených geografických informací do topologické struktury. Pro uložení topologie v SŘDB existuje několik balíčků, například *Radius Topology* nebo *Oracle Topology*, ale žádný není dostatečný vzhledem k speciálním požadavkům, které vychází ze samotné struktury, například slučování Binary Line Generalization stromů [12]. Již práce [12] se zabývala výběrem nejvhodnějšího uložení dat pomocí topologie. Jak bylo uvedeno v kapitole 3 Topologické datové modely, jsou možné tři způsoby:

- left right topologie bez odkazů na hrany.
- left right topologie s ukládáním odkazů na hrany.
- okřídlená hrana.

Každý z těchto způsobů nabízí určité výhody i nevýhody, aby bylo možné rozhodnout, který způsob je nejvhodnější, byly všechny implementovány v práci [12] a byla otestována jejich vhodnost pro strukturu tGAP. Byl porovnán čas, který byl potřebný při rekonstrukci ploch.

Výsledky jsou zobrazeny v grafu Obr. 6.2, díky němu máme dobré srovnání jednotlivých způsobů topologického uložení dat.



Obr. 6.2: Výkonost topologické rekonstrukce ploch z [12]

Implementace okřídlené hrany (Winged edge) je pomalejší než obě left-right topologie, což je pravděpodobně způsobeno tím, že využívá nezávislé dotazy, a tudíž je každý topologický dotaz v databázi řešen zvlášť [12]. Left-right topologie využívá hromadný dotaz do databáze, kde jsou topologické dotazy řešeny najednou [12]. Obě left-right topologie dodávají data přibližně ve stejném čase. Z toho lze vyvodit, že ukládáním odkazu navíc pro počáteční hrany nezískáme rychleji výsledek, dokonce ukládání extra referencí v modelu zabírá více místa. Stejně tak je problém udržet data konzistentní, pokud existuje více odkazů na jednotlivé hrany. Podobně komplikovaný problém může nastat při slučování ploch.

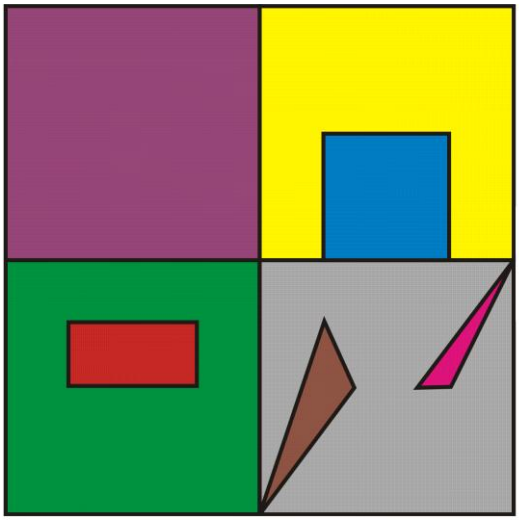
Na základě dvou hlavních důvodů, vycházejících z testování v [12]:

- více odkazů na hrany potřebuje více místa.
- více odkazů zkomplikuje proces slučování při vytváření GAP face tree.

Na základě těchto důvodů bylo rozhodnuto, že nejvhodnější je implementovat left right topologii bez odkazů na hrany 3.1 na rozdíl od předchozí práce [10], kde bylo spíše doporučováno použít okřídlených hran.

6.3 Testovací data

Pro testování a implementaci tGAP struktury byl potřeba soubor dat, na kterém by bylo možné strukturu testovat. Proto byl zvolen *testovací dataset*, s jehož pomocí bylo možné testovat jednotlivé kroky a zkoušet možné varianty slučovacího algoritmu. Soubor dat pro testování byl vytvořen co nejmenší, aby byly veškeré manipulace s daty co přehlednější. Zároveň byl volen dostatečně velký na to, aby bylo možné otestovat strukturu pro co nejvíce variant, které mohou nastat při slučování ploch.



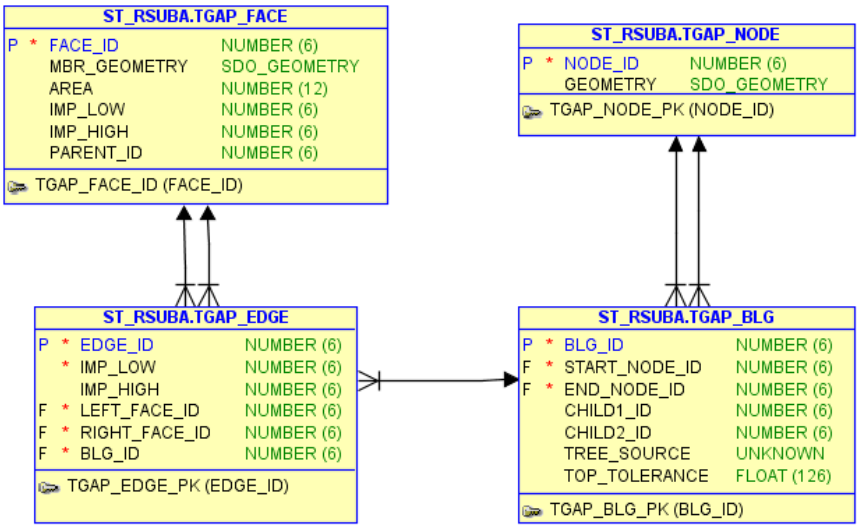
Obr. 6.3: Testovací dataset

V reálném světě může nastat velmi mnoho variant, které nelze odhadnout dopředu. Proto je takřka nemožné vymyslet cvičný soubor dat, ve kterém by všechny varianty nastaly. Přesto byl testovací dataset nejprve použit, abychom eliminovali největší „principiální chyby". Později jsme použili algoritmy, které už jsou vůči základním chybám imunní, na opravdová data.

Testovací dataset byt vytvořen ve formě tabulky, kde jednotlivé řádky reprezentovaly jednotlivé hrany ploch. Pomocí procedury byly jednotlivé záznamy převedeny na BLG a uloženy ve struktuře. Dalším krokem bylo vytvoření záznamů pro jednotlivé plochy společně s jejich odkazy. Tento krok byl proveden ručně. Následovalo testování a vytvoření algoritmů pro správný chod celé struktury.

6.4 Relační datový model

V kapitole 4 bylo popsáno, že struktura tGAP se skládá z GAP face tree a GAP edge forest. Tyto stromové struktury je nutné ukládat a pracovat s nimi v SŘDB. Obr. 6.4 popisuje jednotlivé tabulky, ve kterých je celá struktura tGAP uložena. Návrh tohoto datového modelu je výsledkem práce [12]. Je realizován pomocí několika tabulek, které si můžeme blíže popsat:



Obr. 6.4: Relační datový popisující uložení tGAP face tree a tGAP edge forest

TGAP_FACE

Tato tabulka obsahuje informace o všech plochách. Je zde uložen jednoznačný identifikátor společně s úrovní důležitosti, stejně tak zde můžeme nalézt odkaz na předchozí plochy, z kterých je konkrétní (sloučená) plocha složena.

TGAP_EDGE

Tabulka tgap_edge zaznamenává jednotlivé hrany, identifikátor hran a úroveň důležitosti. Stejně tak jejich vztah k plochám, tedy zaznamenává, která plocha leží vpravo či vlevo od hrany. Je zde uložen odkaz na BLG strom, který hranu reprezentuje. V této tabulce nenajdeme informace o jednotlivých bodech linie.

TGAP_BLG

Zde jsou uloženy jednotlivé BLG stromy, které přesně definují jednotlivé hrany. Zároveň jsou zde odkazy na počáteční a koncový bod jakékoliv linie. Každému uloženému BLG stromu je udělen jednoznačný identifikátor

TGAP_NODE

Jedinečný identifikátor v této tabulce popisuje počáteční a koncové body linií. Vždy souvisí s odpovídajícím BLG stromem potažmo s odpovídající hranou.

6.5 Fyzický datový model

Pro tuto implementaci vycházející z [12] a jsou vytvořeny čtyři tabulky pro tGAP. Dále budou tyto tabulky přesně popsány a definovány jednotlivé atributy.

```
Popis tabulky TGAP_FACE

name                      null ?     type
------------------------- -------- ------------------
face_id                   not null    number
mbr_geometry                          mdsys.sdo_geometry
area                                  number
imp_low                               number
imp_high                              number
parent_id                             number


Popis tabulky TGAP_EDGE

name                      null ?     type
------------------------- -------- ------------------
edge_id                   not null    number
imp_low                   not null    number
imp_high                              number
left_face_id                          number
right_face_id                         number
blg_id                                number


Popis tabulky TGAP_BLG

name                      null ?     type
------------------------- -------- ------------------
blg_id                    not null    number
start_node_id                         number
end_node_id                           number
child1_id                             number
child2_id                             number
tree_source                           blgtree
top_tolerance                         float (126)

Popis tabulky TGAP_NODE

name                      null ?     type
------------------------- -------- ------------------
node_id                   not null    number
geometry                              mdsys.sdo_geometry
```

Popis vybraných atributů:

- atributy edge_id a imp_low spolu tvoří kombinovaný primární klíč. Nestačí použít pouze edge_id, neboť se v průběhu slučování objevují nové záznamy se stejným id, ale rozdílnou důležitostí, jak bude vysvětleno dále.
- jednotlivé tabulky jsou v průběhu slučování doplňovány o nové záznamy, ale na počátku jsou jejich atributy důležitosti nastaveny na hodnotu *NULL* u atributu imp_high a hodnotu 0 pro atribut imp_low.

- atributy child1_id a child2_id slouží až při slučování blg stromů do blg lesů, využijí se při určování orientace dvou sloučených hran, jak je vysvětleno v kapitole 6.10 Slučování BLG stromů.

6.6 Datový typ Binary Line Generalization (BLG) strom

Jak jsme si doposud mohli všimnout ve struktuře tGAP je geometrie ukládána pouze v tabulce bodů (TGAP_NODE), kde jsou uloženy pouze počáteční a koncové body linií. Ostatní geometrie hran a jejich zjednodušení ve struktuře tGAP je ukládáno pomocí Binary Line Generalization (BLG) stromů, které jsou implementovány pomocí navrženého abstraktního datového typu. Z tohoto důvodu byl kladen důraz na podporu abstraktních datových typů při volbě databáze. Tento abstraktní datový typ byl použit v [12]. Nejprve bude popsán a později bude uvedeno, jak je tento datový typ naplňován.

Geometrie v tomto datovém typu je uložena velmi specifickým způsobem, je zpracována do struktury BLG stromů. To znamená, že jednotlivé lomové body linií jsou převedeny do podoby stromové struktury, a není tedy více třeba původní datový typ SDO_GEOMETRY.

SQL kód pro vytvoření BLG stromu v Oraclu [12]:

```
-- create the type for list of tolerances
create or replace type
float_list as
varray(524288) of float;
/

-- create the type for internal references
create or replace type
int_list as
varray(524288) of integer;
/

-- create the type for BLG node geometry
create or replace type
blgpoint as
object (
x number,
y number
);
/

-- create the type for list of points
create or replace type
point_list as
varray(524288) of blgpoint;
/
```

```
-- create the type for a BLG tree
create or replace type
blgtree as
object(
error float_list,
leftnode int_list,
rightnode int_list,
points point_list
);
/
```

Z ukázky lze vypozorovat, že se datový typ BLG strom skládá ze čtyř dalších abstraktních datových typů, float_list, dvakrát int_list a point_list a z toho jeden z nich (point_list) je složen z dalšího abstraktního typu - blgpoint. Je na místě abychom si jednotlivé části stromu trochu popsali, získáme tak lepší představu a podvědomí o jednotlivých částech. Tyto informace se nám budou hodit v dalším textu.

BLG strom se skládá ze čtyř datových typů:

error

Spíše než záznam chyb (z anglického error) by tento list měl být chápán jako pole vzdáleností. V tomto poli jsou postupně ukládány jednotlivé vzdálenosti od linie tak, jak byly spočteny při použití Douglas-Peucker algoritmu [4].

leftnode

Toto pole odkazuje na další uzel umístěný vlevo od současného (levý následovník). Pokud je v poli hodnota 0, žádný levý následovník neexistuje.

rightnode

Toto pole odkazuje na další uzel umístěný vpravo od současného (pravý následovník). Pokud je v poli hodnota 0, žádný pravý následovník neexistuje.

points

Zde jsou uloženy pomocí pole bodů jednotlivé body linie. Toto pomocné pole je též implementováno pomocí abstraktního datového typu. Je důležité poznamenat, že souřadnice bodů jsou uložena pouze jako čísla, nikoliv pomocí datového typu SDO_GEOMETRY navrhnutého Oraclem.

Celkově lze BLG strom chápat jako spojení čtyř datových polí dohromady. Každé ze čtyř polí je implicitně indexované. Stejně tak má každé pole stejný počet prvků jako ostatní a každému prvku s určitým indexem odpovídají vlastnosti uložené v ostatních třech polí se stejným indexem. Například informace uložená na pozici dva odpovídá druhému prvku stromu, se vzdáleností od

linie uloženou v poli též na druhé pozici. Celková velikost pole bude vždy o n-2 prvků menší, neboť počáteční a koncový bod jsou uloženy v tabulce bodů (TGAP_NODE). Pomocí odkazů na levého a pravého následovníka uloženého v atributech leftnode a rightnode není problém stromovou strukturu zrekonstruovat. Na první pozici se vždy nalézá prvek odpovídající kořenu stromu. Jestliže se v poli u levého (leftnode) a pravého (rightnode) následovníka nalézá 0, znamená to, že tento prvek stromu již nemá dalšího následovníka nalevo či napravo.

Tolerance (Error)	X_1	X_2	X_3	X_4	X_5	X_6
Left	2	0	4	0	6	0
Right	5	0	7	0	0	0
Points (coordinates)	$(y,x)_{\#3}$	$(y,x)_{\#2}$	$(y,x)_{\#5}$	$(y,x)_{\#4}$	$(y,x)_{\#7}$	$(y,x)_{\#6}$

Tabulka 6.1: Popis stromové strukturu Binary Line Generalization [12]. V prvním řádku je uložena vzdálenost, v druhém odkaz na levého následovníka, ve třetím odkaz na pravého následovníka a v posledním řádku jsou uloženy souřadnice bodu. Tabulka je naplněná daty z příkladu z kapitoly 4.3. Na první pozici je uložen nejvyšší bod celého stromu, tzv. kořen stromu. Z něho vedou dva následovníci, bod 2 vlevo a bod 5 vpravo. Souřadnice bodů v příkladu jsou zaznamenány symbolicky.

6.6.1 Použití struktury

Abstraktní datový typ Binary Line Generalization (BLG) tree je využívám pro uchování informací o jednotlivých liniích. K naplnění stromu se využije Douglas-Peuckerův algoritmus [4], tento proces byl popsán v kapitole 4.3. Na začátku jsou počáteční a koncové body uloženy pomocí datového typu SDO_GEOMETRY do tabulky TGAP_NODE, reference na tyto body se uloží do tabulky TGAP_BLG. Tímto krokem se velikost celkového BLG stromu zkrátí o n-2 prvků. A ve stromové struktuře jsou uloženy pouze mezilehlé body. Naplnění stromové struktury proběhne pro všechny linie na počátku a později už nejsou žádné úpravy prováděny. Pouze dochází ke spojování jednotlivých BLG stromů a vytváří se tak tzv. GAP EDGE forest. Tímto postupem se zamezí výskytu redundantních dat.

6.7 GAP face tree

Poté co jsou vytvořeny BLG stromy je možné strukturu tGAP naplnit. Naplnění struktury se skládá z dvou spojených kroků: vytvoření GAP face tree, tedy naplnění tabulky TGAP_FACE a vytvoření GAP edge forest, což je naplnění tabulky TGAP_EDGE a částečného doplnění tabulky TGAP_BLG [12].

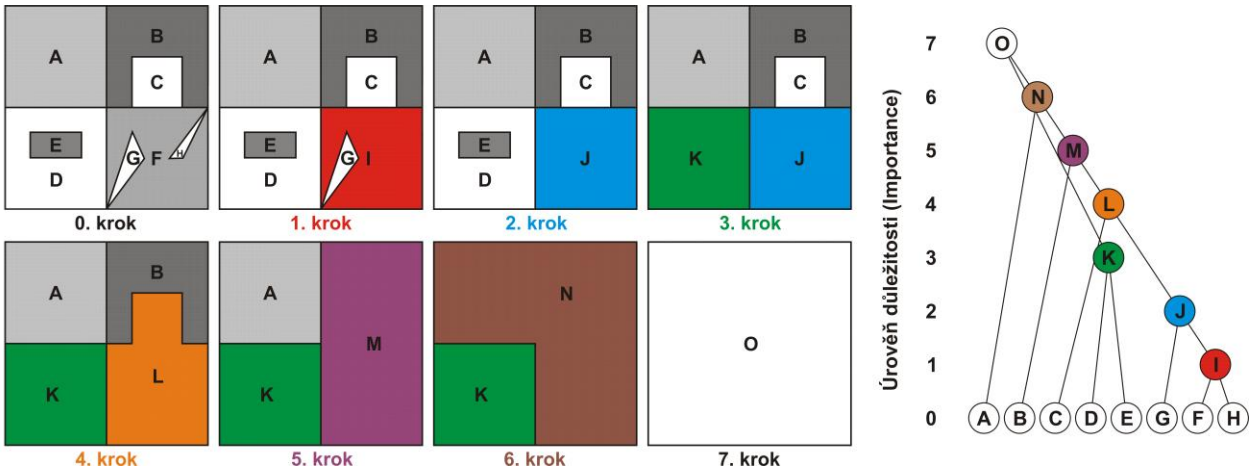
Generalizace pomocí GAP face tree znamená postupné slučování ploch a jeho zaznamenání ve stromové struktuře. Slučování ploch je řešeno tak, že nejmenší plocha se vždy sloučí s největší sousední plochou. Tedy je rozhodováno na základě atributu plocha (area), která je převzata při

vkládání dat do struktury. Bylo by možné využít i sofistikovanější způsoby[8] při slučování ploch, ale v rámci diplomové práce bylo rozhodnuto se zaměřit pouze na použití hierarchické datové struktury.

Vytvoření GAP face tree v Oracle bylo řešeno pomocí funkce v jazyce PL/SQL. Myšlenku této funkce lze popsat následovně [12]:

```
1. Vytvoř frontu všech ploch vhodných pro sloučení

2. Vyber nejmenší plochu z fronty

3. Vyber největšího souseda nejmenší plochy

4. Slouč tyto dvě plochy dohromady (vyřeš hrany slučovaných ploch), odstraň
   sloučené plochy z fronty

5. Sloučenou plochu přidej do fronty

6. Opakuj bod 2-6, dokud nebude existovat pouze jediná ploch ve frontě
```

Postup, který je zde popsaný, je též patrný na obr. 6.5, ten ilustruje jednotlivé kroky při slučování a z toho i vznikající stromovou strukturu. Pro přehlednost je ještě uvedena tabulka 6.2 obsahující informace o jednotlivých plochách.



Obr. 6.5: Vytváření GAP face tree – jednotlivé kroky. Krok nula ilustruje počáteční situaci. V kroku jedna je sloučena plocha H s plochou F. Společně vytvoří plochu I označenou červenou barvou. Takto proces pokračuje, dokud nezbyde pouze jediná plocha. Obrázek vychází z [12].

[8] Operace slučování by mohla být podmíněna dalšími pravidly například, o jaký typ plochy se jedná. Neboli pozemky stejného druhu se slučují nejdříve a až později se slučují s ostatními plochami. Metoda by využívala váhových matic, a tím by určovala nejvhodnější sousední plochu pro sloučení. Takovéto rozšíření by se snadno implementovalo a v práci [21] se toto rozšíření s názvem Constrained tGAP ukazuje jako velmi efektivní. Přesto se tato práce zaměřuje pouze na samotnou funkčnost struktury. Řešení vhodného „váhování“ jednotlivých ploch k dosazení nejvhodnějších výsledků, je natolik komplikovaný problém, který by mohl být řešen v další práci.

Funkce vytvářející GAP face tree, která byla nastíněné výše, se skládá z několika kroků a lze rozepsat následovně [12]:

Všem plochám v tabulce TGAP_FACE je nastavena počáteční hodnota důležitosti (imp_low, imp_high) na 0 pro atribut imp_low a na hodnotu NULL pro atribut imp_high. Plocha, která má imp_high nastavený na NULL, je stále ve frontě. Hodnota současné důležitosti (current importance) je nastavena na 1. Nyní se provádí proces do té doby, dokud nezbyde jedna plocha s atributem imp_high rovna NULL:

- najdi nejmenší plochu (plochu s nejnižší hodnotou v atributu area), která je ve frontě. Např. v kroku čtyři to je plocha C.
- najdi plochu, která sousedí s nejmenší plochou a současně má největší atribut area. Tato plocha je nalezena pomocí hrany nejmenší plochy. Např. Plocha J bude vybrána ve čtvrtém kroku.
- sluč tyto dvě plochy dohromady. To znamená - vytvoř záznam (vyřeš hrany slučovaných ploch). Odstraň sloučené plochy z fronty tím, že do atributu vysoká důležitost (imp_high) uložíš hodnotu současné důležitosti (current importance). Poté ulož do atributu parent_id sloučených ploch identifikátor nové plochy. Např. v kroku 4 bude plochám C a J nastavena vysoká důležitost na 4 a budou sloučeny do nové plochy označené L viz Tabulka 6.1.
- nové ploše vypočti plochu pomocí součtu ploch dvou předešlých.
- sloučenou plochu přidej do fronty, nízkou důležitost nastav na současnou důležitost a její vysokou důležitost na NULL (plocha bude ve frontě) Např. plocha L bude přidána do fronty.
- navyš hodnotu současné důležitosti o jedna.

Pro poslední plochu nastav vysokou důležitost na současnou důležitost.

FACE_ID	AREA	IMP_LOW	IMP_HIGH	PARENT_ID
A	$area_A$	0	6	N
B	$area_B$	0	5	M
C	$area_C$	0	4	L
D	$area_D$	0	3	K
E	$area_E$	0	3	K
F	$area_F$	0	1	I
G	$area_G$	0	2	J
H	$area_H$	0	1	I
I	$area_{H+F}$	1	2	J
J	$area_{G+I}$	2	4	L
K	$area_{E+D}$	3	7	O
L	$area_{C+J}$	4	5	M
M	$area_{B+L}$	5	6	N

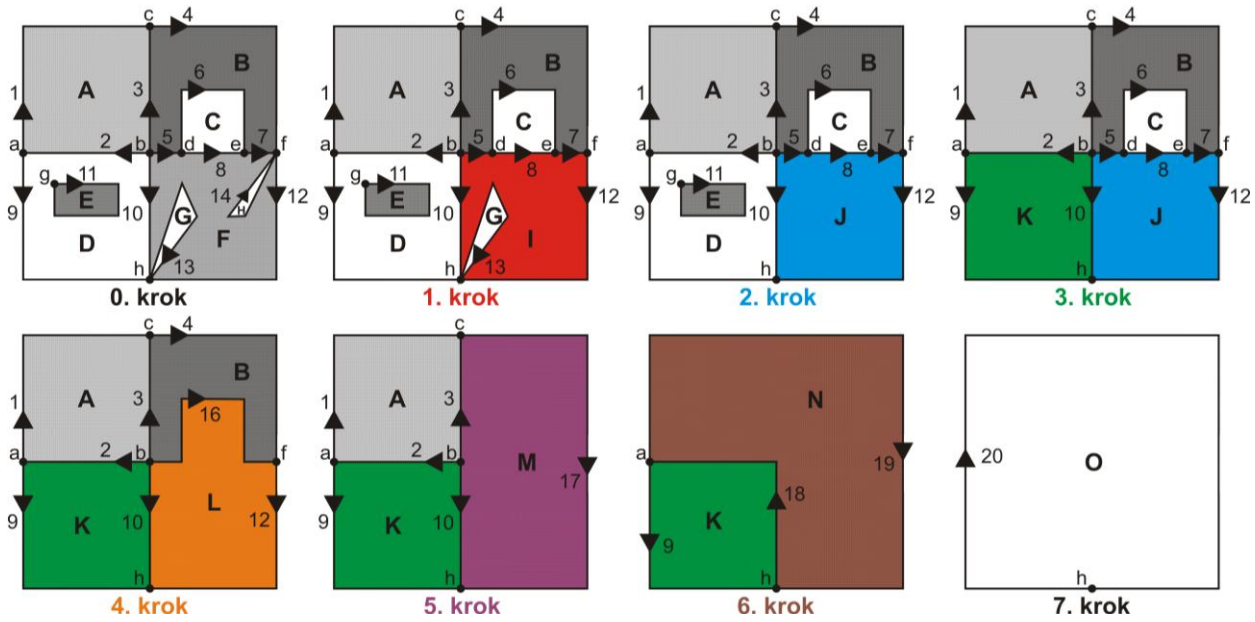
N	$area_{A+M}$	6	7	O
O	$area_{K+N}$	7	8	

Tabulka 6.2: Tabulka ploch tGAP po vytvoření GAP face tree. Plochy korespondují s jednotlivými kroky znázorněnými v obr. 6.5.

Zobrazení výsledků zaznamenané ve struktuře probíhá od nejvyššího prvku ve stromové struktuře po nejnižší. Prakticky to znamená, že nejprve je zobrazena pouze jedna plocha, která nemá žádnou hodnotu v atributu parent_id a poté následují jednotlivé plochy vždy odkazující na aktuální plochu. V našem případě by nejprve byla zobrazena plocha O, poté plocha K společně s N.

6.8 GAP edge forest

Struktura využívá left-right topologii, díky níž dostáváme pro každou plochu skupinu hran odkazující na společnou plochu. Počáteční informace o hranách jsou uloženy pří importu vstupních dat a jsou uloženy v tabulce TGAP_EDGE. Další informace jsou do této tabulky ukládány v průběhu slučování ploch. Pro přehlednost je uveden obr. 6.6.



Obr. 6.6: Vytváření GAP edge forest. Krok nula zachycuje počáteční stav. Velkými písmeny jsou označeny plochy, malými písmeny jsou označeny uzly a čísly jednotlivé hrany. Jednotlivé kroky slučování hran odpovídají slučování ploch z kapitoly 6.7. Obrázek vychází z [12].

Dle [12] pro danou hranu existují pouze tři varianty, které mohou při procesu slučování nastat:

- hrana musí být odstraněna, protože leží mezi dvěma plochami, které se slučují. Např. hrana 16 ve 4. kroku.

- hrana bude po procesu sloučení nadále používána. Je nutné přizpůsobit odkazy na plochy nově vzniklé situaci. Hrany musí obsahovat odkazy na nově vzniklou plochu. Např. hrany 2, 10 a 9 v kroku 3. Odkazy na předchozí plochu D se změní na plochu K.

- bude vytvořena nová hrana, protože dvě nebo více hran je nutné spojit. Např. hrany 1 a 17 v kroku 6, kde se sloučili do hrany 19.

Při vytvoření GAP edge forest využijeme postup, který lze nastínit takto [12]:

```
1. jakmile začne slučování ploch, vytvoř globální frontu ze všech hran.

2. s každým spojení dvou ploch vyber hrany, které tyto plochy definují, odeber
   je z globální fronty a vytvoř z nich lokální frontu.

3. projdi všechny hrany v lokální frontě a ověř zda:

a) jedná-li se o hranu mezi slučovanými plochami, odstraň ji z fronty bez
   další akce.

b) jedná-li se o hranu, která může být nadále používána, přizpůsob odkazy nově
   vzniklé situaci. Poté vrať hranu zpět do globální fronty.

c) jestliže se jedná o hranu, kterou je nutné spojit, je potřeba odstranit
   hrany, které se mají slučovat z lokální fronty a vložit do seznamu, s jehož
   pomocí se sloučí jednotlivé BLG stromy (více je tento krok rozebrán v další
   sekci). Po sloučení BLG stromů je „spojená verze" hrany přidána do globální
   fronty.
```

Postup pro vytváření GAP edge forest je ilustrován na obr. 6.6 a stejně tak ho velmi názorně zachycuje tabulka 6.3.

EDGE ID	IMP LOW	IMP HIGH	LEFT FACE	RIGHT FACE	BLG_ID
1	0	6		A	1
2	0	3	D	A	2
3	0	5	A	B	3
4	0	5		B	4
5	0	1	B	F	5
6	0	4	B	C	6
7	0	1	B	F	7
8	0	1	C	F	8
9	0	3	D		9
10	0	1	F	D	10
11	0	3	D	E	11
12	0	1		F	12
13	0	1	F	G	13
14	0	1	F	H	14
5	1	2	B	I	5
7	1	2	B	I	7
8	1	2	C	I	8
10	1	2	I	D	10
12	1	2		I	12
13	1	2	I	G	13

5	2	4	B	J	5
7	2	4	B	J	7
8	2	4	C	J	8
10	2	3	J	D	10
12	2	4		J	12
2	3	6	K	A	2
9	3	7	K		9
10	3	4	J	K	10
16	4	5	B	L	16 = ((5 ∩ 6) ∩ 7)
10	4	5	L	K	10
12	4	5		L	12
17	5	6		M	17 = (4 ∩ 12)
3	5	6	A	M	3
10	5	6	M	K	10
18	6	7	K	N	18 = (2 ∩ 10)
19	6	7		N	19 = (1 ∩ 17)
20	7	8		O	20 = (9 ∩ 19)

Tabulka 6.3: Zachycuje tabulku TGAP_ EDGE v databázi po procesu slučování ploch. Barvy korespondují s jednotlivými kroky slučování ploch v kapitole 6.7.

Vytváření GAP edge forest se skládá z několika kroků a lze je rozepsat následovně [12]:

Všem hranám v tabulce TGAP_EDGE je nastavena počáteční hodnota důležitosti (imp_low, imp_high) na nula pro atribut imp_low a na hodnotu NULL pro atribut imp_high. Hrana, která má imp_high nastavený na NULL, se nalézá v globální frontě. Při každém sloučení ploch musí proběhnout následovné:

- pro všechny hrany patřící k dvěma slučovaným plochám se změní hodnota vysoké důležitosti (imp_high) z NULL na současnou důležitost (current importence), tím se plochy přesunou z globální fronty do lokální.
- pro všechny hrany v lokální frontě se zkontroluje:

a) jedná-li se o hranu ležící mezi slučovanými plochami, neprovádí se další akce. (hraně zůstane nastavena hodnota imp_high na current_importance.

b) jedná-li se o hranu, která může být nadále používána, přizpůsobí se odkazy nově vzniklé situaci. Vytvoří se nový záznam v tabulce TGAP_EDGE. Nová hrana bude mít stejné id, ale nízkou důležitost (imp_low) bude nastavena na aktuální hodnotu. Vysoká důležitost bude nastavena na NULL, tím se hrana vrátí zpět do globální fronty.

c) jestliže se jedná o hranu, kterou je nutné spojit, je potřeba ji odstranit z lokální fronty a vložit do seznamu, s jehož pomocí se sloučí jednotlivé BLG stromy (viz dále). Po sloučení blg stromů se vygeneruje nové id a nová hrana se vloží do tabulky. Imp_low bude nastavena na současnou důležitost a imp_high na NULL.

Po prohlédnutí všech hran v lokální frontě, zůstane lokální fronta prázdná. Tím celý proces končí a může opět dojít ke sloučení ploch. Protože se mohou hrany se stejným identifikátorem (edge_id) vyskytovat na různých úrovních důležitosti, je primární klíč sloužen ze dvou atributů (edge_id a imp_low).

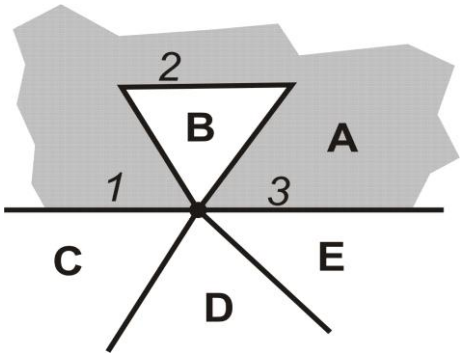
6.9 Princip slučování

6.9.1 Špatný přístup

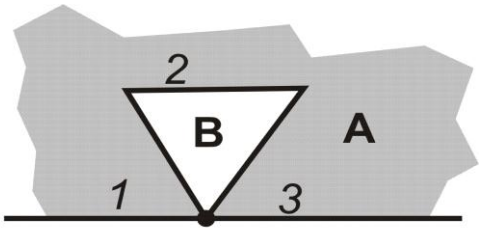
Při vytváření algoritmu, který řeší slučování hran, je nutné řádně rozmyslet, jakým způsobem bude tento proces probíhat. K lepšímu pochopení jsou uvedeny čtyři obrázky (viz níže). Tyto obrázky jsou řazeny od nejjednoduššího případu po nejnáročnější. Jelikož máme uloženy hrany s odkazy na jednotlivé plochy, víme která plocha je vlevo a které plocha je vpravo, není těžké identifikovat hranu, která má být v určitém kroku vymazána (odkazuje na nejmenší plochu a zároveň odkazuje také na největšího souseda). Díky této vlastnosti není těžké takovou hranu najít. V příkladu číslo 2 na Obr. 6.7 se jedná o hranu číslo 2.

Dokonce ani není těžké identifikovat, které hrany se mají sloučit, z důvodu, že počáteční nebo koncový bod rušené hrany navazuje na další hranu. V příkladu číslo 2 na Obr. 6.7 je opět snadné identifikovat, že se jedná o hrany 1 a 3. V příkladu číslo 1 je situace dokonce ještě jednodušší neboť sice identifikujeme hrany 1 a 3, které by se měly sloučit, ale jsou zde i další hrany a proto nemůže ke sloučení vůbec dojít, porušila by se tím topologie (pouze hrana 2 bude odstraněna).

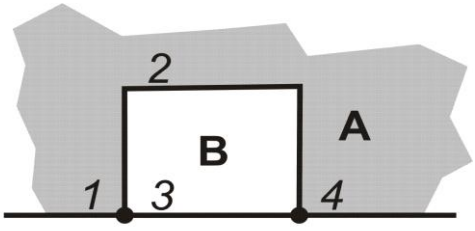
Situace se trochu zkomplikuje, pokud víme, že počáteční a koncový bod rušené hrany není totožný, viz Obr. 6.9. Z této situace jasně vyplývá sloučení tří hran 1, 3 a 4. Je velmi užitečné též identifikovat hranu ležící mezi počátečním a koncovým bodem, v příkladu 3 to je hrana číslo 3.



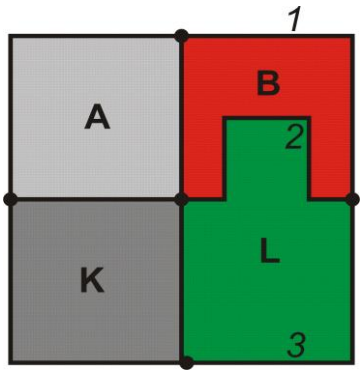
Obr. 6.7: Slučování hran, příklad 1



Obr. 6.8: Slučování hran, příklad 2



Obr. 6.9: Slučování hran, příklad 3



Obr. 6.10: Slučování hran, příklad 4

Ukazuje se, že podmínka pro slučování hran z třetího případu by bylo vhodné řešit dohromady. Tedy sloučení tří hran nastane, když má počáteční a koncový bod společnou hranu, která není rušená v tomto kroku, a zároveň má počáteční i koncový bod společné tři hrany (včetně rušené hrany). Bohužel, jak se ukáže v další textu práce, toto není vhodně definovaná podmínka.

Jestliže definujeme podmínku pro oba uzly najednou, dojde k patové situaci a algoritmus selže. Situace, kdy k tomu dojde, je zachycena na obrázku Obr. 6.9 (hrana 2 je vyhledána jako hrana, která se bude rušit. Spojnice mezi počátečním a koncovým bodem by ale neexistovala. Například počáteční bod linie by byl spojený pro čtyři linie, tedy nic by se nestalo a koncový bod by byl společný pouze pro tři hrany (1, 2 a 3, z toho hrana 2 je rušená). Algoritmus by v takovém případě předpokládal, že má dojít k sloučení pouze jedné hrany navazující na koncový bod s hranou, která leží mezi počátečním a koncovým bodem, která v tomto případě neexistuje. Vyřešení podmínky pro oba uzly (počáteční i koncový) dohromady by se tím ještě mnohem více zkomplikovalo.

6.9.2 Správný přístup

Mnohem efektivnější způsob je postupovat pouze po jednotlivých uzlech a řešit vždy pouze sloučení dvou hran, přestože se vzápětí ukáže, že sloučené hrany ležely hned vedle sebe. Příklad podmínky pro sloučení hran by mohl znít: Pokud je počáteční uzel společný pro tři hrany (včetně rušené) dojde ke sloučení těchto hran. Následně probíhá tato podmínka i pro koncový uzel. Nakonec i samotný algoritmus se velmi zjednoduší.

6.10 Slučování BLG stromů

Pro ukládání jednotlivých hran se využívá stromová struktura s názvem Binary Line Generalization tree (BLG strom). Každá hrana je uložena v tabulce TGAP_BLG pomocí této stromové struktury. Při řešení slučování jednotlivých ploch dohromady (viz předchozí část) je nutné některé hrany definující plochy sloučit dohromady tak, aby nebyla narušena topologie. Výhoda využití BLG stromů tkví v tom, že není třeba znovu ukládat souřadnice jednotlivých hran slučovaných ploch. Při slučování sice v tabulce vznikají nové záznamy odkazující na původní linie, avšak tyto záznamy už neobsahují BLG stromy, ale pouze odkazují na původní BLG, ze kterých vycházejí. Tímto postupem nedochází k duplicitě dat.

Slučování BLG stromů probíhá vždy po dvojicích. Algoritmus řešící slučování BLG stromů pracuje v principu následovně [12]:

- vytvoř globální frontu ze všech BLG záznamů.
- vytvoř seznam BLG stromů, které je potřeba spojit (dle hran, které se slučují.
- na základě seznamu BLG stromů, které se mají sloučit, projdi jednotlivé stromy a po dvojicích je spoj.
- pro nově vzniklý pár vytvoř záznam a opět přidej do globální fronty (Pár může být použit znovu).

BLG_ID	START NODE	END NODE	CHILD1	CHILD2	TREE SOURCE	TOP TOLERANCE
1	a	c			blgtree	-1
2	b	a			blgtree	-1
3	b	c			blgtree	-1
4	c	f			blgtree	-1
5	b	d			blgtree	-1
6	d	e			blgtree	-1
7	e	f			blgtree	-1
8	d	e			blgtree	-1
9	a	h			blgtree	-1
10	b	h			blgtree	-1
11	g	g			blgtree	-1
12	f	h			blgtree	-1
13	h	h			blgtree	-1
14	f	f			blgtree	-1
15	b	e	5	6		3
16	b	f	15	7		3
17	c	h	4	12		4
18	h	a	-10	2		5
19	a	h	1	17		5
20	h	h	-9	19		6

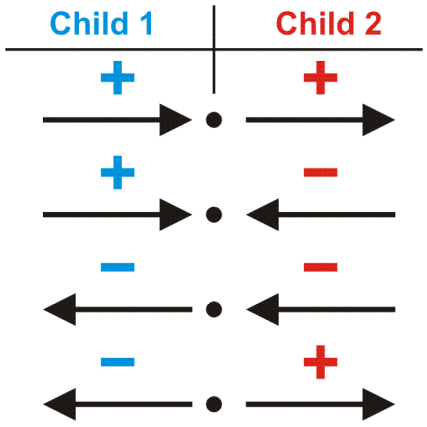
Tabulka 6.4: Tabulka TGAP_BLG, po procesu vytvoření edge forest, obsahující BLG stromy jednotlivých hran. Barvy zachycují kroky (viz. obr. 6.5), při kterých byly záznamy v tabulce TGAP_BLG vytvořeny.

Při vytváření tGAP edge forest, tedy naplňování tabulky TGAP_EDGE, jsou vytvářeny požadavky na sloučení BLG stromů. Jestliže jsou hrany v lokální frontě (viz. kapitola 6.8) a ukáže se, že je potřeba tyto hrany sloučit, spustí se proces sloučení BLG stromů. Příklad takového sloučení nastal na obr. 6.6 v kroku 4, kdy bylo potřeba sloučit hrany 5, 6 a 7.

Proces sloučení BLG stromů se skládá z několika kroků a lze rozepsat následovně [12]:

1. vezmi dva BLG stromy, které je nutné sloučit, a najdi společný uzel. (Společný uzel je možné zjistil z tabulky TGAP_BLG, kde jsou uloženy informace o počátečních a koncových uzlech. Např. Při sloučení BLG stromů číslo 5 a 6 byl společný uzel d.
2. spoj BLG stromy, které byly definováni v předchozím kroku. Vyřeš problém orientace hran. Vytvoř nový záznam pro spojení BLG stromů a přiřaď nové id. Např. V kroku 6 se slučují BLG stromy 10 a 2 se společným bodem b. Hrana 10 je orientovaná opačně, proto je do atributu child1 uložena s opačným znaménkem.
3. vrať hodnotu nového BLG stromu zpět sloučeným hranám.
4. pokračuj znovu od kroku 1, dokud nezbyde pouze jediný BLG strom.

Při slučování dvojic BLG stromů je nutné brát ohled na orientace slučovaných hran. Orientace jednotlivých hran se vždy rozhoduje na základě jejich společného uzlu. V Obr. 6.11 jsou znázorněny všechny varianty, které mohou při slučování nastat. Další dva případy zde nejsou zobrazeny, protože se jedná pouze o podmnožinu případů zde definovaných.



Obr. 6.11: Přehled variant při slučování BLG stromů. Uprostřed je znázorněn společný bod. Černé šipky reprezentují směry jednotlivých hran. Modrá a červená barva zachycuje znaménko u atributu Child1 a Child2.

Na základě toho, o jakou variantu se při slučování jedná, je atributům Child1 a Child2 přiřazeno znaménko. Je nezbytně nutné, aby přiřazování probíhalo stále stejně, protože jinak není možné např. zobrazit vybranou plochu.

7 Reálná data

Struktura tGAP, o které práce pojednává, není zamýšlena pro generalizaci testovacích dat. Hlavním účelem navrhované struktury je ukládání a zpracování skutečných dat např. katastru nemovitostí. V tomto případě byla reálná data poskytnuta Českým úřadem zeměměřickým

a katastrálním (dále jen ČÚZK). Jednalo se o jedno vybrané katastrálního území Plzeň 4 s kódem 722731 uložené ve formátu shapefile.

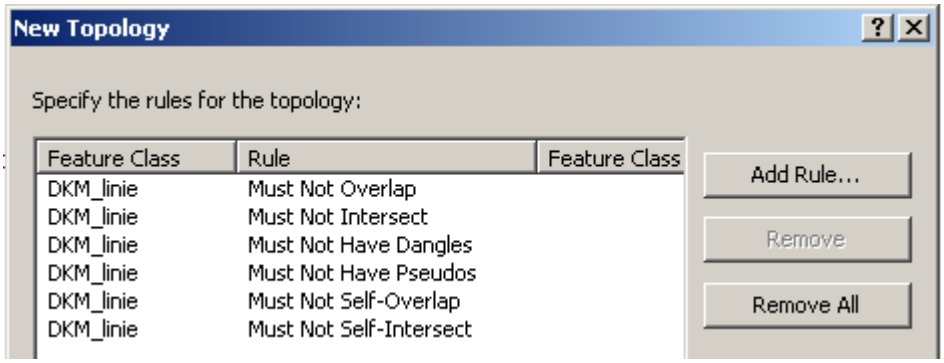
Data, která poskytnul ČÚZK, jsou uložena v geometrickém modelu, což znamená, že zachycují pouze souřadnice jednotlivých prvků (bodů, linií a ploch), ale neobsahují žádné informace o vzájemných vztazích. Např. žádný atribut nedefinuje sousední plochu. Abychom mohli data nahrát do struktury tGAP, která využívá data s topologickými vlastnostmi, je nejprve potřeba převést zdrojová data do topologického datového modelu. Takto uložená data lze upravit do tvaru, ze kterého je možné automaticky převést záznamy do struktury tGAP.

Tato kapitola popisuje jednotlivé kroky, pomocí kterých je možné nahrání dat do struktury tGAP anebo data uložená ve struktuře zpětně exportovat. Celý postup je zachycen na Obr. 6.1 v kapitole 6 Implementace tGAP. Nejprve bude popsán převod dat z formátu shapefile do databázového systému Oracle spolu s převodem do topologického datového modelu. Následně se text věnuje převodu dat z topologického datového modelu do stádia pre-tGAP, které jasně definuje podobu dat před automatickým převodem do struktury tGAP. Závěrem je vysvětleno, jakým způsobem je možné data ze struktury exportovat.

7.1 Předzpracování a import dat do Oracle

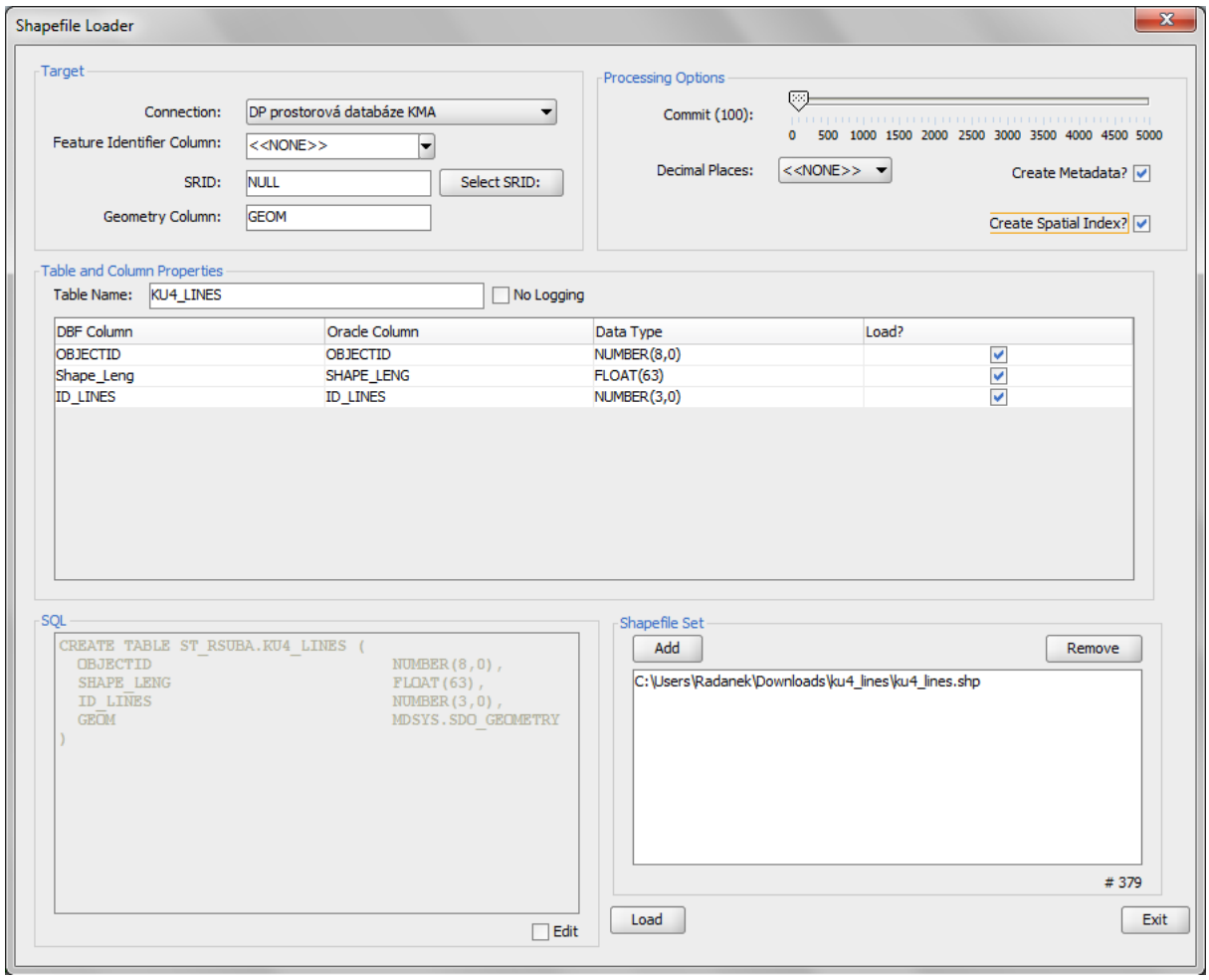
Pro uložení dat ve struktuře tGAP je nezbytné data upravit. Data musí být ve vhodném stavu pro uložení v topologickém modelu dat v databázovém systému Oracle. To znamená, že data se musí stát tzv. topologicky čistými, tedy takovými daty, nad kterými je možné vytvořit topologii, aniž by se jakkoli změnila jejich poloha [15]. Tím dosáhneme toho, že by data po jakékoliv úpravě měla být topologicky čistá, jakékoliv operace s daty budou probíhat korektně a neměl by se vyskytnout žádný neočekávaný problém související s geometrií objektu např. identická linie, která leží na jiné linii či křížení dvou linií bez společného bodu.

Topologickou kontrolu dat je možné řešit v různých softwarech různým způsobem. Vhodné řešení například nabízí program ArcGIS od společnosti ESRI. S využitím tohoto softwaru je možné přehledně a efektivně řešit topologické čištění dat. Uživatel definuje jednotlivá topologická pravidla a program následně kontroluje jejich porušení [16]. Následnou editací dat s vyznačenými topologickými chybami může uživatel data upravit. Příklad definovaných pravidel je vidět na Obr. 7.1.

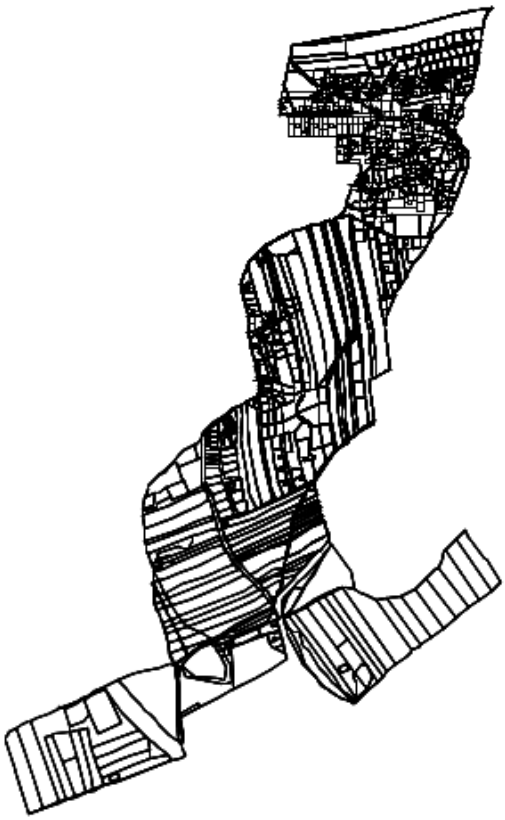


Obr. 7.1: Topologická pravidla v programu ArcGIS.

Po provedení kontroly topologie je možné katastrální data nahrát do databázového systému Oraclu, kde budou uložena v datovém typu SDO_GEOMETRY. Import lze realizovat přímo v SŘDB Oracle pomocí programu SQL Developer se zásuvným modulem GeoRaptor [17], který přímo nabízí import dat ve formátu shapefile viz Obr. 7.2. Následně je možné data zobrazit pomocí stejného zásuvného modulu Obr. 7.3.

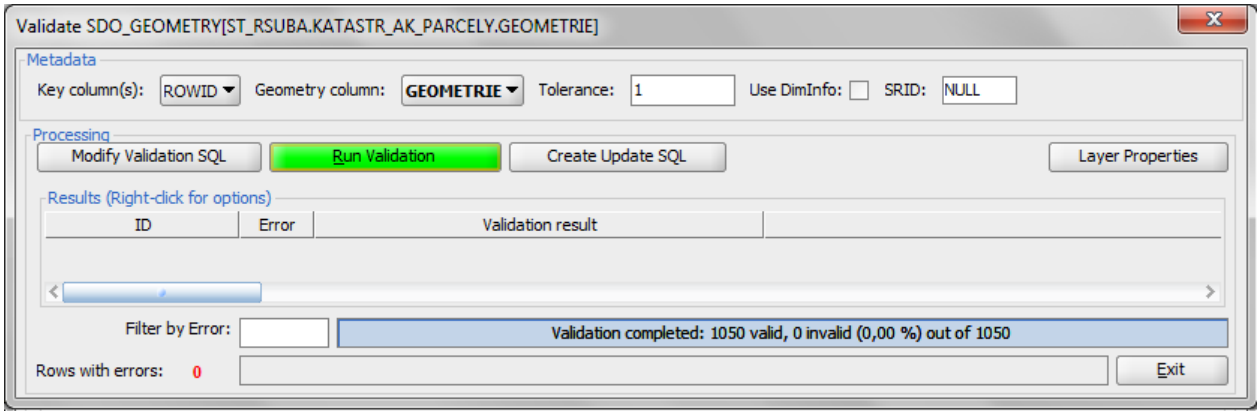


Obr. 7.2: GeoRaptor: Shapefile Loader.



Obr. 7.3: Zobrazená data po importu v SŘDB Oracle.

Jakmile jsou data uložena v SŘDB Oracle je možné je převést do topologického datového modelu. Ještě před samotným převodem je důležité ověřit, zda jsou prostorová data topologicky čistá. Je možné využít zásuvný modul GeoRaptor, který umožnuje provést test validity prostorových dat uložených pomocí objektového datového typu SDO_GEOMETRY. Úspěšný výsledek provedeného testu je znázorněn na Obr. 7.4.



Obr. 7.4: Kontrola validnosti geometrie.

Kontrola validnosti geometrie v Oraclu je již druhá kontrola topologie v pořadí (první kontrola byla provedena v programu ArcGIS), přesto ji nelze vynechat. SŘDB Oracle má mnohem větší požadavky na způsob, jakým jsou data uložena. Při druhé kontrole dat se stále můžou vyskytovat objekty, které nesplňují podmínky validity, přestože již první kontrolou prošli. Např. plocha v Oracle má jasně stanoveno, že směr hran ohraničující plochu je definován proti směru hodinových ručiček [18]. V programu ArcGIS nelze zkontrolovat, zda tato podmínka platí, a proto nemusí být plocha označena jako validní.

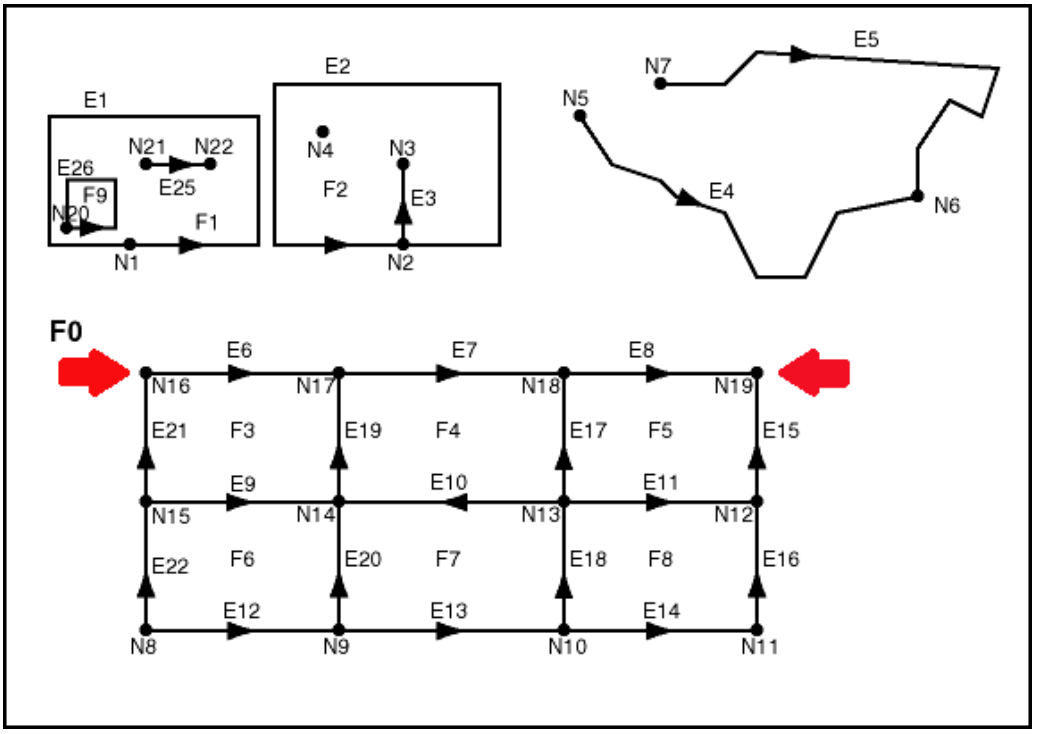
Samotný převod do topologického modelu přímo z datového typu SDO_GEOMETRY je dobře popsán v [17] a [11], a proto ho není třeba podrobně popisovat.

7.2 Z topologického modelu do pre-tGAP

Pokud jsou prostorová data uložena v topologickém modelu v databázovém systému firmy Oracle, jsou jednotlivá primitiva *plochy*, *hrany*, *body* uložena ve třech tabulkách, pro každé primitivum jedna. Jak jsou primitiva reprezentována v topologickém modelu je vidět na Obr. 7.5.

Poznámky k obrázku [11]:

- E elementy (E1, E2, …) jsou hrany, F elementy (F0, F1, …) jsou plochy a N elementy (N1, N2, …) jsou uzly.
- Šipka u každé hrany značí orientaci této hrany.
- F0 je tzv. univerzální plocha. F0 a je identifkována hodnotou ID = -1.
- Pro každou bodovou geometrii je vytvořen uzel, stejně tak pro každý počáteční a koncový bod. Např. hrana E25 má počáteční bod N21 a koncový bod N22.
- Izolovaný uzel (*island node*) je uzel, který je izolovaný uvnitř plochy. Např. uzel N4 uvnitř plochy F2.
- Izolovaná hrana (*island edge*) je hrana, která je izolována uvnitř plochy. Např. hrana E25 je isolovaná hrana uvnitř plochy F1.
- Smyčka (*loop edge*) je hrana, která má stejný počáteční a koncový bod. Např. hrana E1 je s počátečním a koncovým bodem N1.



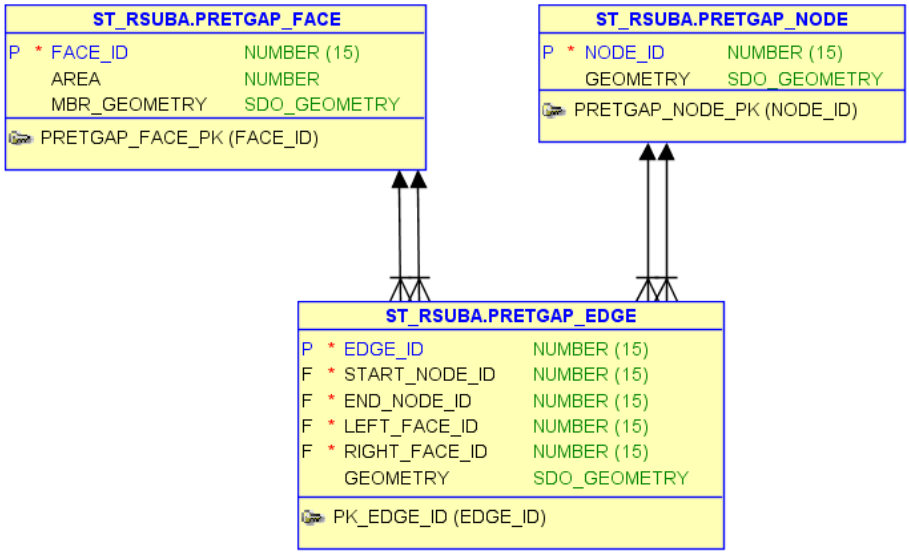
Obr. 7.5: Příklad topologických primitiv v Oracle [19]. Červenými šipkami jsou označeny části, které jsou problematické při importu do struktury tGAP.

Do struktury tGAP mohou vstupovat všechna primitiva, ale některé případy nejsou přípustné. Mezi nepřípustné patří izolované uzly, izolované hrany a hrany podobné hraně E3 na Obr. 7.5. Naproti tomu smyčka je ve struktuře tGAP přípustná. Další problém pak nastává s hranami, které mají počáteční nebo koncový bod společný pouze s další jedinou hranou, v Obr. 7.5 je takový příklad označen červenou šipkou. Přestože jsou tyto hrany přípustné v topologickém modelu Oracle, není možné je nahrát do struktury tGAP, protože by proces slučování hran neproběhl korektně. Je nutné takové případy vyhledat a eliminovat např. sloučením těchto hran dohromady, ale zároveň je důležité brát ohled na jejich orientaci. Pří realizaci diplomové práce k tomu sloučili procedury *ku_turn1*, *ku_turn2* a *ku_marge_lines*.

Po odstranění všech těchto nežádoucích případů můžeme data z topologického modelu uložit do tabulek označené jako pre-tGAP. Obsah Tabulek pre-tGAP je možné automaticky převést do struktury tGAP.

7.2.1 Pre-tGAP

Tabulky vhodné pro import do struktury tGAP jsou seskupeny do datového modelu s názvem pre-tGAP reprezentovaného na Obr. 7.6: Datový model pre-tGAP. V datovém modelu pre-tGAP můžeme vidět tabulky a jejich atributy, které budou při importu do struktury využity. Všechny ostatní atributy z topologického modelu např. next_left_edge_id nebo prev_left_edge_id jsou nepotřebné, a proto se v datovém modelu pre-tGAP ani nevyskytují. Jednotlivé tabulky může popsat následovně:



Obr. 7.6: Datový model pre-tGAP.

PRETGAP_FACE

Tato tabulka obsahuje informace o všech plochách vkládaných do struktury. Je zde uložen jednoznačný identifikátor společně s velikostí plochy (area) a *minimálním ohraničujícím obdélníkem*[9] (mbr_geometry).

PRETGAP_EDGE

Tabulka pretgap_edge ukládá jednotlivé hrany a jejich vztah k plochám. Zaznamenává, která plocha leží vpravo či vlevo od hrany. Je zde uložena geometrie každé hrany.

PRETGAP_NODE

Jedinečný identifikátor v této tabulce popisuje počáteční a koncové body linií. Každý bod je zde uložen v datovém typu SDO_GEOMETRY.

7.2.2 Fyzický datový model pre-tGAP

Tabulky obsažené v pre-tGAP jsou popsány a definovány následovně:

[9] Minimální ohraničující obdélník (Minimal Bounding Rectangle) je aproximace geometrického prvku obdélníkem. Slouží k indexaci prostorových dat. [26]

```
Popis tabulky PRETGAP_FACE

name                      null ?     type
------------------------ -------- ------------------
face_id                  not null    number(15)
mbr_geometry                         mdsys.sdo_geometry
area                                 number

Popis tabulky PRETGAP_EDGE

name                      null ?     type
------------------------ -------- ------------------
edge_id                  not null    number(15)
start_node_id            not null    number(15)
end_node_id              not null    number(15)
left_face_id             not null    number(15)
right_face_id            not null    number(15)
geometry                             mdsys.sdo_geometry

Popis tabulky PRETGAP_NODE

name                      null ?     type
------------------------ -------- ------------------
node_id                  not null    number(15)
geometry                             mdsys.sdo_geometry
```

Data uložená v pre-tGAP lze snadno převést pomocí procedury *pretgap2tgap* do struktury tGAP. Vše probíhá automaticky, a není tedy potřeba zásahu uživatele. Jakmile procedura nahraje data do struktury tGAP, může se přistoupit ke slučování ploch a vytváření stromových struktur. Vše lze aktivovat spuštěním procedury *merge_face*.

7.3 Z tGAP do SDO_GEOMETRY

Na začátku kapitoly 6 Implementace tGAP bylo zmíněno, že při vstupu do struktury tGAP jsou všechny linie převedeny do stromové struktury tzv. BLG stromů. Poté probíhá zjednodušování ploch a jejich hranic, jak bylo popsáno v kapitolách 6.7, 6.8 a 6.10. Problém nastává, pokud chceme zobrazit výsledky, kterými jsme využitím struktury dosáhli. SŘDB Oracle umožňuje zobrazovat výsledky za pomocí programu SQL Developer se zásuvným modulem GeoRaptor [17]. Zobrazovaná data však musí být datového typu SDO_GEOMETRY. Další možností pro zobrazení výsledků může být export dat do formátu KML a zobrazení v programu Google Earth. Tento způsob má navíc nevýhodu v tom, že data v Oraclu nemají přímo definovaný souřadnicový systém pro naše území, a zobrazení v Google Earth je tak ještě o něco problematičtější. Ať už je zvolen způsob zobrazení jakýkoliv, nevýhoda obou těchto možností je stejná: nejprve je důležité převést geometrická data ze struktury tGAP do datového typu SDO_GEOMETRY.

Pro převod dat ze struktury tGAP byla vytvořena procedura *return_lines,* která má parametr důležitost (imp_high). Tato procedura nám vrátí všechny hrany zvolené úrovně důležitosti v datovém typu SDO_GEOMETRY. Navíc však převede i hrany, které jsou ve vyšších úrovních podrobnosti (vyšší hodnota důležitosti). Výsledkem je kompletní vrstva dat až po nejpodrobnější zvolenou úroveň (zvolený imp_high). Využitím této procedury získáváme lepší přehled o stavu všech dat. Např. Podle toho jak je výsledná vrstva podobná původním datům dokážeme vytušit, zda proběhlo mnoho sloučení ploch nebo zda došlo pouze k několika základním.

V budoucnu by dalším způsobem převodu dat ze struktury tGAP mohla být procedura *return_face* s parametry: identifikátor hrany a důležitost (imp_high). Tato procedura by nám vrátila plochu zvolené úrovně důležitosti v datovém typu SDO_GEOMETRY. Pracovala by s daty velmi podobně jako procedura *return_lines*, ale navíc by musela řešit pospojování všech výsledných hran dohromady, aby výsledek byla právě jedna plocha. Při zpracování diplomové práce tato metoda nebyla tolik užitečná, protože nevrací data v „širším kontextu“. Proto nebyl její návrh realizován.

V další práci se strukturou tGAP by se užitečnost obou procedur pravděpodobně změnila. Zatímco v této chvíli je mnohem užitečnější zobrazovat prostorová data v širších souvislostech a tedy i prostorové prvky, které jsou převedeny víckrát. V budoucnu by tomu mohlo být přesně naopak a mnohem důležitější by bylo převést a zobrazit pouze minimální množství dat (sloučenou plochu) a co nejrychleji zobrazit sloučenou plochu. Druhá procedura (*return_face*) by se stala mnohem více užívanou naproti tomu, první procedura (*return_lines*) by se pravděpodobně přestala použít úplně.

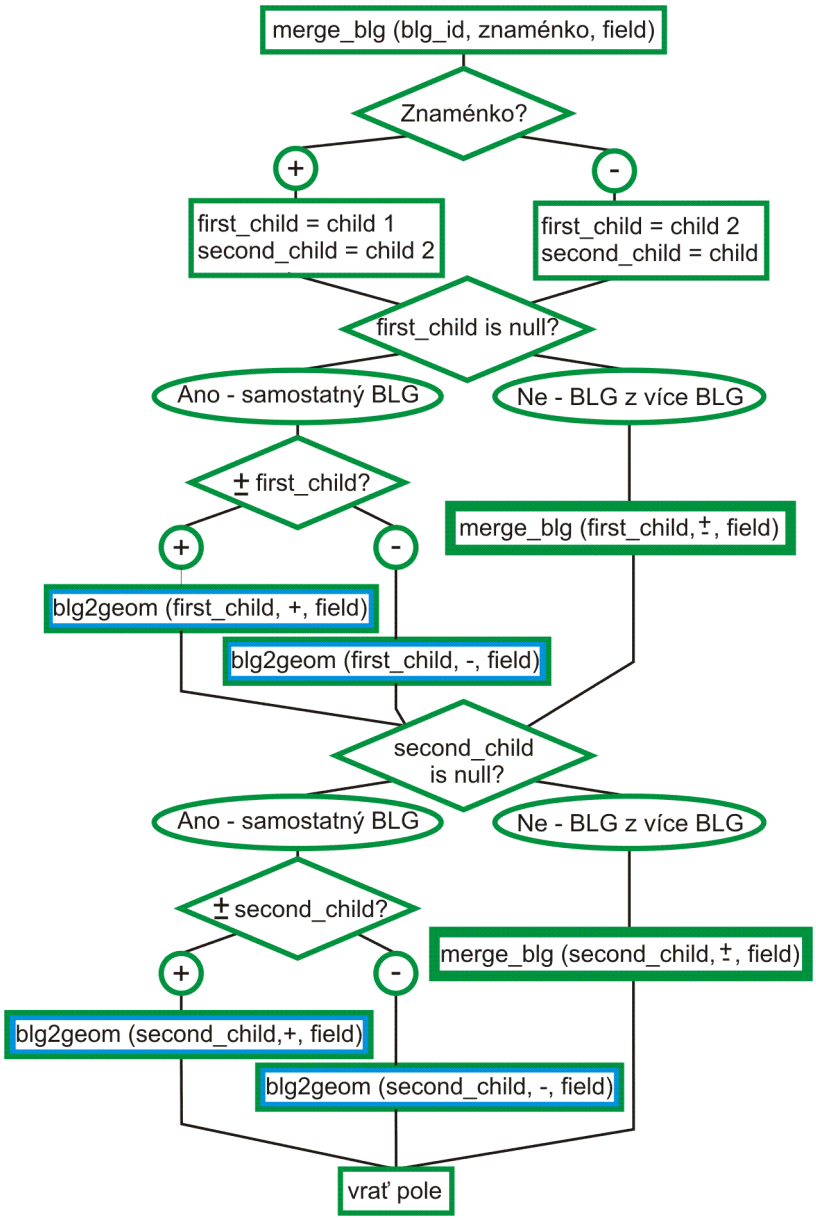
Obě tyto procedury pracují nebo by pracovali s výsledky dalších procedur. Celý tento proces funguje na principu postupného procházení jednotlivých hran, které odpovídají zvolené důležitosti. Každá nalezená hrana je převáděna z BLG stromů na pole souřadnic a je přidán počáteční a koncový bod. Při převádění jednotlivých hran ze stromové struktury na pole souřadnic je řešen i problém návaznosti jednotlivých dílčích úseků (BLG stromů), jejich orientace a výskyt duplicitních bodů při spojování.

Výsledkem procedury *return_lines* je tabulka RETURN_LINES_ALL, kde jednotlivé záznamy odpovídají jednotlivým převedeným hranám. Každá linie v tabulce je definována svým identifikátorem, úrovní důležitosti a datovým typem SDO_GEOMETRY. K zobrazení výsledků převodu a vlastně i vybraného kroku zjednodušení je možné opět využít GeoRaptor.

7.3.1 Stěžejní procedury při převodu

Obr. 7.6 a Obr. 7.7 zachycují dvě procedury a v závěru je popsána třetí, které jsou důležité pro převod dat ze struktury tGAP do datového typu SDO_GEOMETRY. Společně převádějí BLG stromy na pole souřadnic, každé části hrany přidávají počáteční a koncový bod a řeší problémy

s tím spojené. Např. návaznosti jednotlivých dílčích úseků či výskyt duplicitních bodů při spojování (koncový bod jedné hrany je stejný jako počáteční bod další).

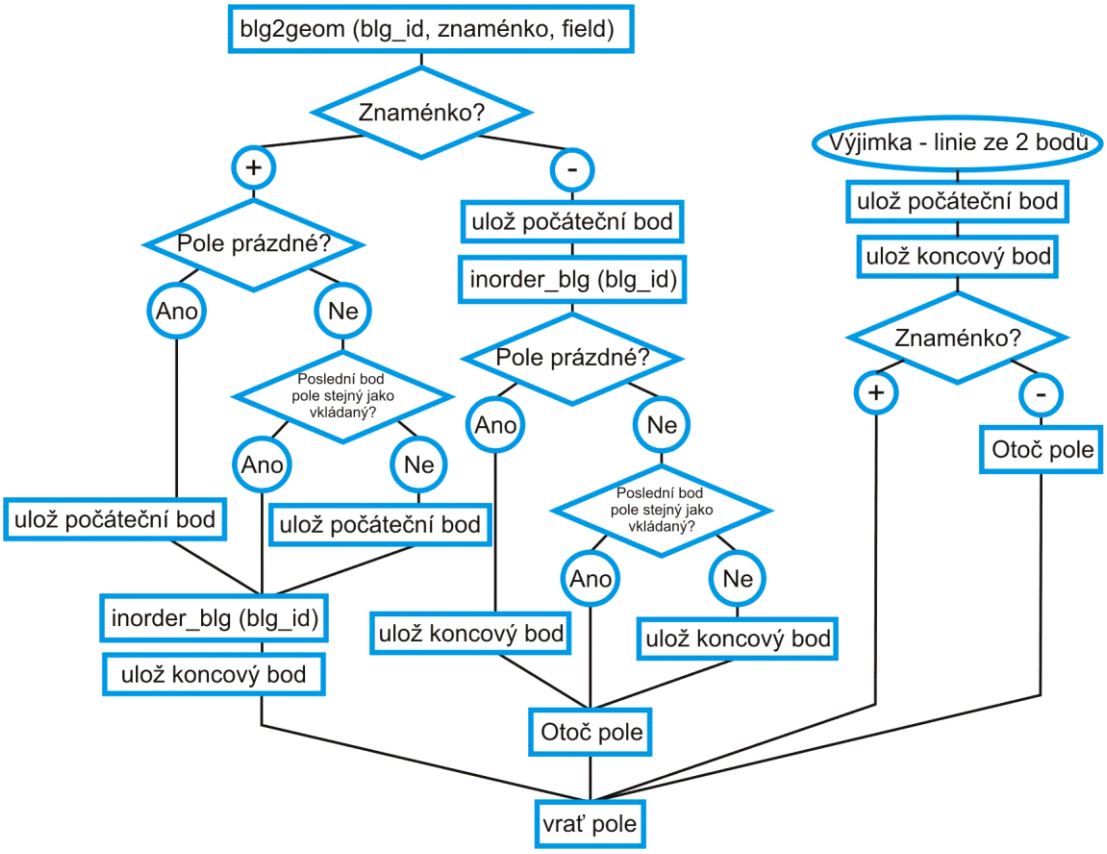


Obr. 7.7: Diagram procedury *merge_blg* procházející spojené BLG stromy. Modře jsou označeny volání procedury *blg2geom*. Dvojnásobně širokým zeleným obdélníkem je označeno rekurzivní volání procedury.

První procedura *merge_blg* na Obr. 7.7 je volána pro jednotlivé záznamy BLG stromů. Zjišťuje, zda je záznam složen z dalších BLG stromů. Pokud ano, procedura rekurzivně zavolá sama sebe. Pokud záznam další BLG stromy neobsahuje, zavolá proceduru *blg2geom* ,která bude vysvětlena dále.

Druhá procedura *blg2geom* na Obr. 7.8 je volána v případě, že BLG záznam v tabulce už neodkazuje na žádné další BLG stromy. Řeší orientace úseků hran a podle toho přidává počáteční

a koncové body. Jestliže je usek hrany orientován opačně, upraví pole souřadnic. Zároveň řeší i situaci kdy se hrana skládá pouze ze dvou bodů a tedy je BLG strom prázdný. V okamžiku kdy je přidán první bod seznamu souřadnic je volána procedura *inorder*, která prochází stromovou strukturu BLG a převádí jednotlivé body do pole souřadnic. Není použita jedině tehdy, pokud se hrana skládá pouze ze dvou bodů, tedy počátečního a koncového.



Obr. 7.8: Diagram procedury *blg2geom* řešící orientaci a ukládání počátečních a koncových bodů.

Třetí procedura *inorder* převádí hranu uloženou v BLG stromu na seznam souřadnic. Využívá algoritmus, který prohledá stromovou strukturu do hloubky. Podle pořadí, ve kterém se prochází uzly uspořádaného stromu, se uvádí jako metoda procházení *inorder*[10]. Výsledkem procházení stromu je pole obsahující jednotlivé prvky stromu v určitém pořadí.

Funkce v principu pracuje následovně:

```
1. vytvoří se pole souřadnic a vloží se počáteční bod linie.
2. proběhne algoritmus inorder (tím získáme pole souřadnic).
3. na závěr se přidá koncový bod linie.
```

[10] Procházení stromovou strukturou do hloubky *inorder* znamená, že se nejprve se projde levý podstrom, pak se provede akce a poté se projde se pravý podstrom [20].

Implementace algoritmu *inorder* řešící procházení stromové struktury a ukládání jednotlivých prvků do pole, vychází z obecného principu [20] a lze rozepsat následovně:

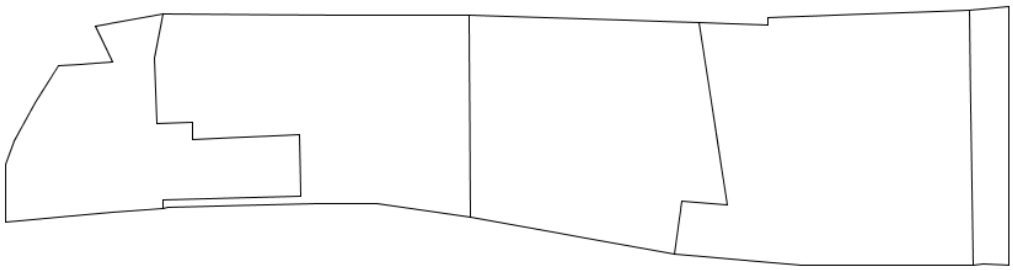
```
INORDER(vrchol, pole vrcholů)
1. zjisti levého následovníka (left_node).
2. zjisti pravého následovníka (right_node).
3. jestliže left_node != null INORDER (left_node, pole).
4. rozšíření pole a přidání vrcholu do pole.
5. jestliže right_node != null INORDER (left_node, pole).
```

Všechny tyto procedury mají jako vstupně-výstupní parametr pole souřadnic, do kterého jsou postupně ukládány jednotlivé souřadnice hran, v závěru procesu exportu je pak pole uloženo v datovém typu SDO_GEOMETRY. Takto se postupně zpracují všechny hrany.

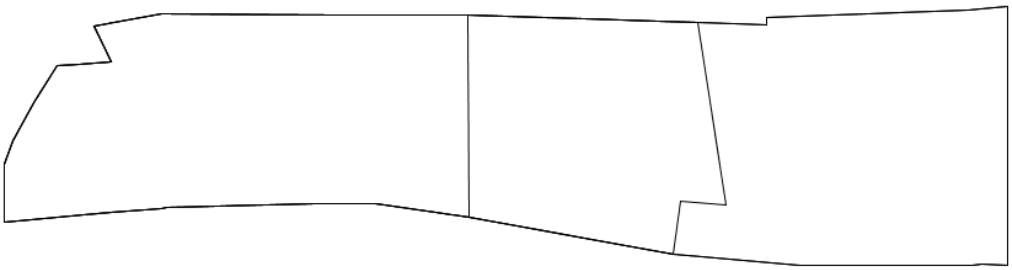
7.4 Výsledky práce s reálnými daty

Při práci s reálnými daty katastru nemovitostí jsem pracoval se souborem čítající 1050 ploch a zhruba 6000 linií a 6000 bodů. Testována byla funkčnost základní implementace struktury a byl řešen problém převodu topologicky uložených dat do tabulek zvaných pre-tGAP a následné nahrání do struktury tGAP. Úprava před vstupem, import do struktury a následné vytváření stromů proběhlo korektně. Avšak při kontrole dat ve struktuře tGAP bylo zjištěno, že proběhlo pouze 60 kroků sloučení ploch. Při opětovné úpravě dat dle kapitoly 7.2 bylo dosaženo sloučení ploch na úrovni důležitosti 120 později 206. Ostatní plochy při slučování nejsou zapojeny a atributy zůstávají nevyplněny, přestože proces slučování proběhne správně. Jedná se zhruba o dvě třetiny ploch, které do procesu slučování nevstoupí.

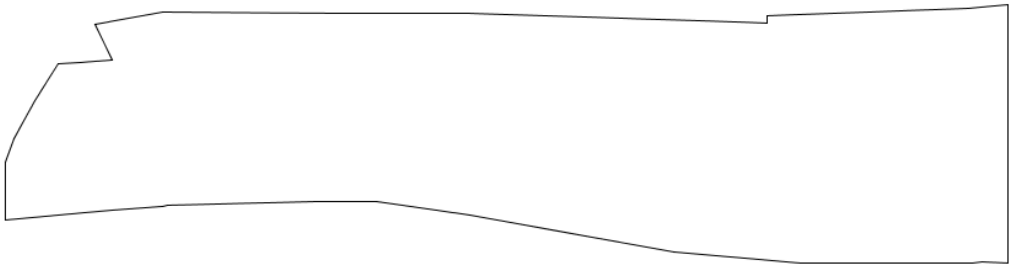
Jeden z příkladů reálných dat, u kterého proces fungoval, můžeme vidět na Obr. 7.9, Obr. 7.10 a Obr. 7.11. V tomto případě byl vybrán pouze malý vzorek dat, byly provedeny všechny zmíněné úpravy a proces sloučení ve struktuře tGAP. Vše proběhlo bez problémů. Z toho můžeme odvozovat, že základní implementace struktury pracuje správně. Problematickou částí tak zůstávají samotná data resp. jejich úprava před vstupem do struktury. Zdá se, že i přes opakovaně pozitivní výsledky kontroly validity nejsou data ve struktuře funkční.



Obr. 7.9: Příklad z reálných dat: Původní stav.



Obr. 7.10: Příklad z reálných dat: Průběh zjednodušení.



Obr. 7.11: Příklad z reálných dat: Závěrečná fáze.

8 SWOT analýza

Diplomová práce se zabývá datovou strukturou tGAP pro *on-the-fly* generalizaci. Navzdory tomu, že se jedná o základní implementaci této struktury, její výhody jsou:

- Největší přínos zvolené struktury spočívá ve využití topologického uložení dat. Díky tomu je eliminována duplicita dat.
- Je vhodná pro webové či mobilní aplikace, protože omezuje množství dat posílané uživateli a tím urychluje celý proces.
- Zároveň nabízí možnost vybrání libovolné úrovně podrobnosti. Můžeme si tedy vybrat libovolnou fázi slučování jednotlivých ploch a tu zobrazit.

- V neposlední řadě je možné strukturu použít na data z reálného světa a tím její význam nabývá na důležitosti.

Silné stránky struktury tGAP jsou jednoznačné, ale bohužel má struktura i jistá slabá místa. Těmi jsou převážně:

- Základní implementace struktury popsaná v této práci nepodporuje editaci dat. Není možné jakkoliv obsah struktury upravit nebo pozměnit.

- Zároveň nepodporuje vytvoření mapy vybraného měřítka. Kdybychom použili tGAP pro vytváření mapového díla, například v měřítku 1:5000, kde by jako referenční vrstva sloužila katastrální mapa v digitální podobě. Pomocí struktury bychom sice dostali takřka nekonečný počet úrovní podrobnosti, ale chybělo by nám jakési „logické napojení“ mapového měřítka na úroveň důležitosti struktury. Bylo by zapotřebí datové sady, z které bychom odvodili, jaký krok zjednodušení tGAP odpovídá měřítku mapy.

- Struktura tGAP nepodporuje jiné než plošné prvky. V budoucnu by bylo možné do struktury zahrnout i další prostorové prvky jako jsou osamocené linie či body (např. definiční body parcel). Vyvstává však další otázka, jak tyto „neplošné“ prvky propojit se současnými plošnými nebo jak vyřešit jejich vzájemné interakce. Např. sloučení dvou ploch do jedné, kde by v každé ploše byl jeden definiční bod.

- Slučování ploch pouze na základě jejich velikosti. Tímto postupem po několika krocích ztrácíme přehled o obsahu dat, např. budova byla sloučena s lesem. Takovýto postup není příliš vhodný, ale bohužel nemůžeme ovlivnit, které plochy se budou spojovat dohromady.

- V závěru je třeba zmínit, že struktura nepodporuje další generalizační operátory. Struktura tGAP zjednodušuje prostorová data pouze *slučováním* (*agregace*). Pomocí tohoto operátoru však nelze vhodně řešit všechny situace, např. generalizace vodního toku.

9 Závěr

Diplomová práce se zabývala problematikou generalizace *on-the-fly*. Nejprve se zaobírala tím, co přesně generalizace *on-the-fly* znamená a jak je možné generalizaci map v reálném čase řešit a jak lze k řešení přistupovat. Následně byl zvolen jeden přístup využívající hierarchické datové struktury, který byl popsán v několika posledních letech a vykazuje velmi slibné výsledky. Jedná se o *topological Generalized Area Partition,* zkráceně označován jako tGAP, který vychází z několika předešlých prací, ale na rozdíl od nich řeší i otázku redundance dat v uložené struktuře.

Datová struktura tGAP, která byla vybrána jako nejvhodnější pro provádění generalizace *on-the-fly,* byla implementována v prostředí Oracle 11g s využitím nadstavby Spatial. Následně byl vytvořen testovací dataset, na kterém byly realizovány a zkoušeny algoritmy pracující s tGAP, jednalo se například o vykreslování ploch poté, co jsou zjednodušeny a uloženy ve struktuře.

Následujícím krokem bylo použití reálných dat z katastru nemovitosti v základní implementaci struktury. Funkčnost struktury byla pozitivně otestována na několika příkladech pocházejících ze souboru reálných dat. Vždy se však jednalo pouze o plochy a linie v řádu maximálně desítek. Celý soubor dat čítající 1050 ploch a zhruba 6000 linií fungoval ve struktuře pouze v omezené míře. Bylo dosaženo sloučení pouze 206 ploch, přestože nedošlo k žádné chybě. Tento problém pravděpodobně souvisí s kvalitou reálných dat. Zdá se, že i přes opakovaně pozitivní výsledky kontroly validity nejsou data ve struktuře funkční. V budoucnu by proto bylo vhodné přijít na způsob vyhledání „problémových dat“ a jejich úpravu.

Diplomová práce řeší pouze základní podobu struktury tGAP, avšak je několik dalších možností, jak strukturu rozšiřovat. Většina z nich byla zmíněna v pracích [21] a [22]. Například použití Constrained tGAP je velmi nasnadě, protože jeho implementace není komplikovaná. Toto rozšíření tGAP řeší slučování ploch nejen na základě jejich velikosti, jak tomu je u tGAP, ale při slučování ploch je brán ohled i na to, o jaký typ plochy se jedná. Například pozemky stejného druhu se slučují nejdříve a až později se slučují s ostatními plochami. Metoda využívá váhových matic, a tím určuje nejvhodnější sousední plochu pro sloučení.

Nejaktuálnější rozšíření, které bylo uveřejněno v [22], popisuje využití struktury tGAP s dalším generalizačním operátorem známý pod názvem *kolaps* (*Collapse*). Jedná se vytváření koster jednotlivých ploch (skeleton) a na základě těchto koster jsou slučovány s okolními plochami. Za pomocí tohoto generalizačního operátoru je možné generalizovat i tak problematické objekty jako jsou řeky či vodní plochy. V [22] je rovněž diskutována problematika zjednodušování linie v průběhu její generalizace. S nižší úrovní podrobnosti klesá i podrobnost linie, a tím je ještě umocňována efektivita struktury. Pomocí těchto a dalších rozšíření, např. *umísťováním popisků* (*Laybel Placement*) se tato hierarchická struktura může stát opravdu plnohodnotným nástrojem na *on-the-fly* generalizaci.

Závěrem se nabízí i implementace struktury tGAP v open source. Možností by bylo použití například databázového systému PostgreSQL s geografickým modulem PostGIS. Takováto implementace by určitě nebyla bez problémů a musely by se řešit otázky, které by s implementací vyvstávaly. Např. rozdílné uložení topologie v PostgreSQL. Výhoda by byla zřejmá a to cena takovéhoto řešení.

10 Citovaná literatura

1. HOFMAN, A. *Application of generalisation in Rotterdam*. Rotterdam: 2007. A literature study. Delft University of Technology. Dostupné také z: http://www.gdmc.nl/publications/2007/Application_of_generalisation.pdf

2. ROBERT, W. a D. BURGHARDT. Generalization, On-the-Fly. In: SHARKKAR, S. a X. HUI. *Encyclopedia of GIS*. SpringerScience+Buisiness Media LLC. 2008, s. 339-44. ISBN: 978-0-387-35973-1.

3. VAN OOSTEROM, P. *Reactive Data Structures for Geographic Information Systems*. Leiden: 1990. Disertační práce. Leiden University, Department of Computer Science.

4. DOUGLAS, D. H. a T. K. PEUCKER. *Algorithms for the reduction of the number of points required to represent a digitized line or its …*. 1973, 112-122 s..

5. BAYER, T. *Generace bodových prvků/skupin. Generalizace liniových prvků/skupin. Generalizace plošných …*. 2010. Univerzita Karlova, Přírodovědecká fakulta, Katedra aplikované geoinformatiky a kartografie [cit. 2012-03-15]. Dostupné z: web.natur.cuni.cz/~bayertom/Adk/adk8.pdf

6. POKORNÝ, J. *Prostorové datové struktury a jejich použití pro indexaci prostorových objektu* [online]. Ostrava: 2000. Dostupné také z: http://gis.vsb.cz/GIS_Ostrava/GIS_Ova_2000/Sbornik/Pokorny/Referat.htm

7. BURGHARDT, D. R.S. PURVES a A.J. EDWARDES. *Techniques for on the-fly generalisation of thematic point data using hierarchical data structures*. Norwich: 28-30. April. 2004.

8. GUTTMANN, A. *R-Trees: A dynamic index structure for spatial searching* [online]. 1984 [cit. 2012-04-10]. Dostupné z: http://www.sai.msu.su/~megera/postgres/gist/papers/gutman-rtree.pdf

9. VAN OOSTEROM, P. a V. SCHENKELAARS. The Development of an Interactive Multi-Scale GIS [International Journal of Geographical Information System:]. The Hague (The Netherlands): 1993.

10. VAN OOSTEROM, P. Variable-scale Topological Data Structures Suitable for Progressive Data Transfer: The GAP-face …. *Cartography and Geographic Information Science*. 2005, s. 331-46. Dostupné také z: http://www.rgi-otb.nl/uwsm2/publications/RGI-233-01.pdf

11. JANEČKA, K. *Modelování konzistentní báze geodat*. Plzeň: 2009. Disertační práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd. Dostupné také z: http://home.zcu.cz/~kjanecka/Disertace-Karel-Janecka.pdf

12. MEIJERS, B. M. *Implementation and testing of variable scale topological data structures*. Delft (The Netherlands): 2006. Diplomová práce. Delft University of Technology, Section GIS Technology.

13. ZÍMA, M. Databázové systémy 2. *Základy PL/SQL* [studijní text]. Plzeň: 2011 [cit. 2012-04-12]. Dostupné z: http://www.kiv.zcu.cz/~zima/vyuka/db2/cviceni-plsql-zaklady.html

14. BATKO, M. Relační vs. objektově-relační vs. objektové databáze [online]. 2004 [cit. 2012-04-15]. Dostupné z: http://www.fi.muni.cz/~xbatko/oracle/compare.html

15. BŘEHOVSKÝM, M. a K. JEDLIČKA. Studijní článek: Topologické čištění dat. *Úvod do geografických informačních systémů (GIS)* [online]. 2007, verze 2007 [cit. 2012-04-18]. Dostupné z: http://www.gis.zcu.cz/studium/ugi/elearning/msgisu06s08cz/default.htm

16. ESRI. Topology rules. In: *ArcGIS 9.2. Desktop Help* [online]. 2007 [cit. 2012-05-15]. Dostupné z: http://webhelp.esri.com/arcgisdesktop/9.2/index.cfm?TopicName=Topology_rules

17. LAEBE, H. et al. *GeoRaptor*. [cit. 2012-04-17]. Dostupné z: http://sourceforge.net/apps/mediawiki/georaptor/index.php?title=Main_Page

18. MURRAYM, C. Oracle Spatial Developer's Guide, 11g Release 2 (11.2) [online]. E11830-09. 2011 [cit. 2012-05-16]. Dostupné z: http://docs.oracle.com/cd/E16338_01/appdev.112/e11830.pdf

19. ORACLE®. Spatial Topology and Network Data Models. In: *1 Topology Data Model Overview*. [cit. 2012-05-12]. Dostupné z: http://docs.oracle.com/cd/B19306_01/appdev.102/b14256/sdo_topo_concepts.htm#BABCJFAD

20. BAYER, T. *Dynamické datové struktury*. 2010. prezentace. Univerzita Karlova, Přírodovědecká fakulta, Katedra aplikované geoinformatiky a kartografie [cit. 2012-04-20]. Dostupné z: http://web.natur.cuni.cz/~bayertom/Prog2/prog2_2.pdf

21. HOFMAN, A. et al. *Using the constrained tGAP for generalisation of IMGeo to Top10NL model*. 2009, 23 s.. Dostupné také z: http://www.rgi-otb.nl/uwsm2/publications/233-42.pdf

22. MEIJERS, M. *Variable-scale Geo-information*. Delft: 2011. Disertační práce. Delft University of Technology, Section GIS Technology. 235 s.. Dostupné také z: http://

repository.tudelft.nl/assets/uuid:b477efe1-e656-4c90-931a-d16083737401/thesis_meijers_tudelft_online.pdf

23. DUNKARS, M. Automated Generaliation in a Multiple Representation Database. In: *Proc. 12th Int. Conf. on Geoinformatics − Geospatial Information Research: Bridging the Pacific and Atlantic* [online]. University of Gävle, Sweden, 7-9 June 2004 (Sweden): 7-9. June. 2004, s. 741-48 [cit. 2012-03-20]. Dostupné z: http://fromto.hig.se/~bjg/geoinformatics/files/p741.pdf

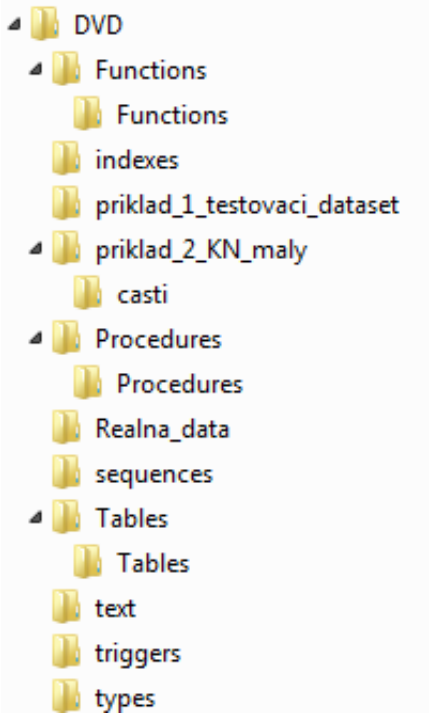
24. KRÁTKÝ, M. *Databázové a informační systémy* [online]. 2006-2007 [cit. 2012-03-25]. Dostupné z: http://www.cs.vsb.cz/kratky/courses/2006-07/tis/download/dbis.pdf

25. JEDLIČKA, K. *Možné chyby, které se mohou vyskytnout při vstupu dat* [učební text]. 29. Března 2006 [cit. 2012-03-29]. Dostupné z: http://www.gis.zcu.cz/studium/ugi/elearning/msgisu05s07cz/default.htm

26. RYCHTERA, R. *Reaktivní datové struktury*. Plzeň: 2007. Semestrální práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd [cit. 2012-04-18]. Dostupné z: http://gis.zcu.cz/studium/pdb/referaty/2007/Rychtera_ReaktivniDatoveStruktury/index.html

Příloha A

Obsah přiloženého DVD



Obr. A.1: Obsah přiloženého DVD

Obsah přiloženého DVD je znázorněn na obrázku A.1.

V adresáři **text** je uložen text této diplomové práce.

A adresářích **priklad_XY** jsou uvedeny jednotlivé příklady použití struktury tGAP. (XY je pak typ příkladu).

V dalších adresářích jsou pak jednotlivé kódy, které byly použity v průběhu zpracování diplomové práce.

Příloha B

Zdrojové kódy

Obsah přílohy je přiložen na DVD.

Jedná se o zdrojové kódy pro 4 funkce, 11 procedur, 5 sekvencí, 4 triggery a abstraktní datové typy.